Learning Graph Neural Networks on Complex Structured Data

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Jingwei Guo

Department of Electrical Engineering & Electronics
School of Electrical Engineering, Electronics and Computer Science

May 2024

Abstract

Graphs are a ubiquitous data structure in the real-world, representing entities and their relationships in areas such as biological networks, social sciences, and recommendation systems. Unlike grid-like data (e.g., images and sequences) with a regular structure, graph-structured data exhibits discrete, irregular, and non-independent and identically distributed (non-i.i.d.) nature. These inherent characteristics bring unique challenges to modern deep learning techniques, which are primarily designed for data with regular structures. In recent years, Graph Neural Networks (GNNs) have emerged as a powerful tool for graph data analysis via seamlessly exploiting both node features and graph topology. Classical GNN models often employ a proximity smoothing strategy, a method effective for simple, community-structured graphs where similar nodes are likely to be close, and vice versa. However, this approach faces limitations when applied to graphs with more complex structures, as it assumes homogeneity in node interactions and overlooks the intricate relationships that often characterize complex networks. Such complex-structured data, prevalent in domains like social and webpage networks, often embody intricacies like *entangled node relationships* and *heterophilic linking patterns*, which challenge the local smoothness assumptions underlying conventional GNNs. To this end, this thesis presents an extensive exploration and advancement of GNNs, extending from spatial to spectral domains. It encapsulates an array of algorithmic frameworks, with contributions to both understanding and handling the multifaceted nature of complex graph-structured data.

First, we confront the challenge of *entangled node relationships* in graph-structured

data, a critical issue often neglected by traditional GNNs due to their reliance on holistic representation learning with proximity smoothing. This oversight can potentially render the learned representations hardly explainable and less informative. To address this, we introduce the Local-Global Disentanglement (LGD) framework, which integrates both local and global graph information to effectively disentangle latent factors underlying node connections. Building upon existing neighborhood routing mechanisms, LGD incorporates a unique regularizer to promote inter-factor diversity and employs a global encoding scheme to enhance intra-factor consistency in the disentangled representations.

Second, we extend disentangled representation learning to address the challenge of *heterophilic linking patterns*, where a significant portion of connected nodes belong to different classes. This situation deviates from the common homophily assumption in conventional GNNs, introducing complexities that standard models do not inherently account for. Inspired by our insights on addressing *entangled node relationships*, we posit that both task-relevant and irrelevant factors underlie the graph to determine connections between nodes of the same and different classes. Specifically, we develop the Edge Splitting (ES) GNN framework, a novel approach tailored for heterophilic scenarios. ES-GNN dynamically partitions graphs into subgraphs based on edge relevance for targeted spatial aggregation, effectively disentangling task-relevant information from irrelevant data. This enhances GNNs' adaptability and robustness across varied network environments.

Third, in our exploration of heterophilic settings, we uncovered a notable phenomenon: besides nodes with different labels connecting, there is also a significant presence of regional heterogeneity. While certain spectral GNNs adapt to arbitrary label patterns using adaptive filters, we identified a critical shortfall in their approach: homogeneous spectral filtering often fails to capture local structural variations in complex networks. This observation led to the development of our novel Diverse Spectral Filtering (DSF) framework. DSF revolutionizes spectral GNNs with node-specific filter weights, merging a global component, shared among all nodes, and a locally varying one to reflect regional differences. This approach enables DSF to not only capture global graph characteristics

but also to discern diverse local patterns, taking into account the unique positions of nodes.

Lastly, building upon our advancements in spectral GNNs, we delve deeper into their spatial interpretability, marking a natural progression towards a more comprehensive understanding of GNN models. The focus of our previous study was on enhancing spectral GNNs within their native domain. Now, we shift to a cross-domain analysis, providing a fresh perspective by rethinking spectral GNNs from a spatial lens. We theoretically bridge spectral filtering and spatial aggregation, demonstrating how spectral GNNs inadvertently transform the original graph into a new, non-local graph in the spatial domain. Our findings challenge the traditional reliance on fixed-order polynomials in spectral GNNs, which often overlook global information. Based on this discovery, we propose the Spatially Adaptive Filtering (SAF) framework, which leverages this cross-domain insight to implement non-local aggregation effectively. It addresses long-range dependencies, especially in heterophilic graphs, further augmenting our ability to handle complex graph structures.

Acknowledgements

As I draft this thesis, I'm struck by how swiftly time has flown. The decision to pursue a PhD four years ago now feels like yesterday. Yet, it is this fleeting period has offered me invaluable opportunities for growing in both academic knowledge and personal development. Now, upon reaching this important milestone, I am filled with gratitude towards many people who have made my PhD journey so enriching.

First, I would like to thank my advisor, Prof. Kaizhu Huang, for giving the opportunity to pursue a PhD. His continuous support and encouragement, as well as the freedom he gave me to explore research areas of my interest, have been instrumental. I cherish the memories of working alongside Prof. Huang, especially recalling the days when I could easily step out of my office to knock on his door, seeking guidance and engaging in discussions on various research topics. I am really grateful to have Prof. Huang as my mentor. I also want to thank my secondary advisor, Prof. Xinping Yi, for his invaluable assistance in refining my papers and providing extensive advice on my research directions. His prompt and enthusiastic responses to my requests for help were always encouraging. Many thanks also go to the other faculty members of our research group, including Dr. Qiufeng Wang, Dr. Xiaobo Jin, and Dr. Rui Zhang, for their constructive feedback during group meetings and continuous support in managing general affairs.

The mentorship and support from senior members of our research group have been crucial in my journey. Special thanks to Dr. Shufei Zhang and Dr. Xi Yang, whose guidance in my first year was as impactful as that of my advisors. Their patience in

explaining fundamental machine learning concepts and the engaging discussions on the latest research ideas encouraged me to delve into theoretical research. Without their help, my growth would not have been possible. My gratitude also extends to Dr. Hang Dong, Dr. Zhiqiang Gao and Dr. Yuyao Yan, whose extensive research experience and insights were immensely helpful during my PhD journey.

I am also thankful to have met my girlfriend, Ms. Zixian Su, also a PhD student in our research group. Her belief in my abilities and daily encouragement have made me more confident and focused. The relaxation and happiness I feel in her company have been a source of great joy. She has also been my steadfast companion through countless days and nights spent working on academic papers. Additionally, I also want to thank my little cat, Youyu, whose companionship reminds me to enjoy life even during the intense periods of research and offers comfort during challenging times. I would also like to thank my colleagues and friends, Dr. Chenru Jiang, Dr. Haochuan Jiang, Dr. Guanyu Yang, Dr. Penglei Gao, Dr. Jing Li, Dr. Shuyi Qu, Mr. Zhuang Qian, Mr. Zihan Ye, Mr. Maizhen Ning, Ms. Wuwei Ma, and Mr. Yiming Lin. Their companionship during fitness activities, willingness to care for my cat during my travels, and shared moments of joy and humor have enriched my daily life.

Lastly, my deepest gratitude goes to my parents and family for their unconditional love and support. Their sacrifices and teachings from my childhood have laid the foundation for my dreams. I am really thankful for their patience, understanding, and support, especially during the most challenging phases of my PhD journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In this digital era, the importance of graphs in representing structured and complex data cannot be overlooked. Unlike grid-like data such as images or sequences, which is inherently limited in a regular structure, graphs offer a versatile means to capture the intrinsic relationships arising from our world. For instance, social networks can be encoded into a graph with nodes representing online users and edges depicting their relationship such as friendships. In the field of biology, a molecule can be described by a set of proteins and their interactions. This universal approach to data representation enables a deeper investigation into the nuances of real-world phenomena. By analyzing graph-structured data, we can discover invaluable knowledge, driving forward innovations that benefit human society at large.

Despite the ubiquity of real-world graphs, most machine learning models fall short in extracting the rich structural information. This limitation stems from their requirement for fixed-sized features and the assumption of independent and identically distributed (i.i.d.) samples – conditions not met by graph data due to its irregular shape and instance dependence. Although few efforts have been made to overcome these challenges based on graph kernels [6, 7], matrix factorization [8, 9], and random walk [10, 11] techniques, these methods often rely on manual feature engineering and utilize shallow architectures. As

a result, they mostly achieve sub-optimal representation quality and learning outcomes. This observation underscores the necessity for machine learning models with broader applicability, which are capable of adaptively analyzing and interpreting the diverse real-world structures.

With the advancement of deep learning, notably in computer vision, there has been an increasing exploration into adapting these techniques to accommodate the complex, irregular structures presented in graphs. This transition, however, introduces new challenges due to the inherent properties of structured data, such as varying node degrees, the lack of fixed node orderings, and arbitrary graph sizes. These challenges impede the application of the popular building block in deep learning — convolutional neural networks (CNNs), as they depend on grid-like structures with uniform spatial relationships for applying filters.

Recently, Graph Neural Networks (GNNs) have emerged as a universally applicable class of models for graph-structured data, marking a significant stride in research. Their ability to seamlessly integrate topological patterns with node features has made them indispensable across diverse fields, such as recommendation systems [12, 13], social network analysis [14, 15], chemistry [16, 17], and physics [18, 19]. GNNs can be broadly divided into spatial-based and spectral-based methods. The former is mainly built upon a message-passing framework [20] where nodes exchange information with their spatial neighbors. The latter is rooted in spectral graph theory [21], implementing convolution operations on graphs via spectral filters. Regardless of the category, traditional GNNs generally operate under an implicit assumption of local smoothness [4, 22], which aligns well with simple, community-structured graphs where proximal nodes display similar behaviors and attributes (see Figure 1.1). However, this assumption may not hold under more complex graph contexts, particularly those exhibiting *entangled node relationships* and *heterophilic linking patterns* (see Figure 1.2).

To address these issues, this thesis delves into the general GNN framework, scrutinizing its model capacity and unearthing how specific properties of diverse real-world structures can be harnessed as inductive biases. This endeavor aims to significantly

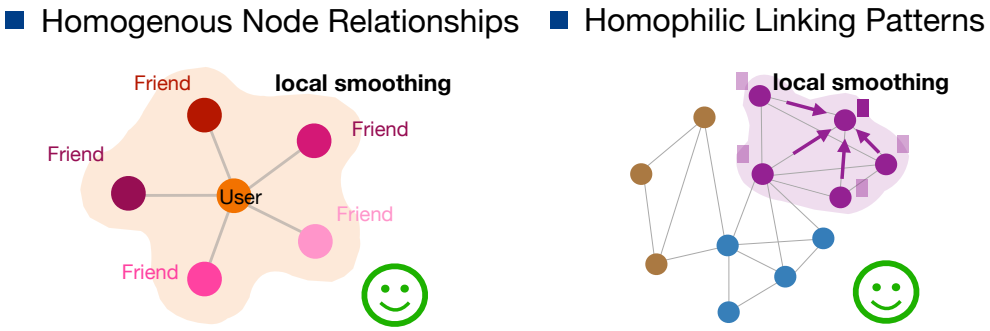


Figure 1.1: Simple graphs with homogeneous node relationships and homophilic linking patterns.

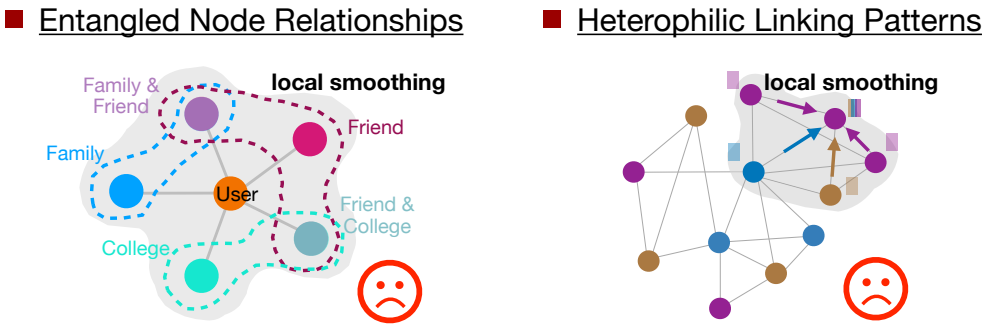


Figure 1.2: Complex graphs with entangled node relationships and heterophilic linking patterns.

enhance GNNs' performance and interpretability when dealing with complex structured data. Through our investigation, we seek to not only deepen the understanding of GNN models but also pave the way for their practical usage in solving real-world problems.

1.2 Contributions

The main contributions of this thesis are summarized as follows:

- We propose the LGD framework (Chapter 3) to address the challenge of *entangled node relationships* on graphs, aimed at disentangling latent factors behind node connections. LGD framework improves upon earlier studies by leveraging neighborhood routing locally and message passing globally. This dual approach not only amplifies the diversity of disentangled representations across varying factors but

also enhances their consistency within each factor. Therefore, it enables clearer interpretations of complex networks and significantly improves GNNs' performance.

- We generalize GNNs beyond the assumption of strong homophily by introducing the Edge Splitting (ES-) GNN framework (Chapter 4). ES-GNN employs a disentangled learning scheme (as inspired by our analysis from Chapter 3) by first dynamically partitioning the graph into two subgraphs based on edge relevance with respect to task-relevant and irrelevant factors underlying the graph. This division further allows for targeted spatial aggregations, effectively disentangling the task-relevant and irrelevant information as typically mixed in graphs with *heterophilic linking patterns*.

- We identify a notable presence of regional heterogeneity in graphs with *heterophilic linking patterns* and respond with the Diverse Spectral Filtering (DSF) framework (Chapter 5). DSF stands out by automatically learning node-specific filter weights, with awareness of node positions, to effectively capture both local structural variations and global graph characteristics. This advancement not only refines the capacity of GNNs in navigating real-world graph diversity but also enhance their interpretability across complex graph scenarios.

- We rethink spectral GNNs from a spatial viewpoint, uncovering that although these models are theoretically rooted in the spectral domain, their operation inherently modifies the spatial relations within the graph. This modification produces non-locality and signed edge weights, reflecting global label relationships among nodes. Inspired by this discovery, we propose the Spatially Adaptive Filtering (SAF) framework (Chapter 6), which exploits these cross-domain properties to better capture long-range dependencies especially on graphs with *heterophilic linking patterns*. This advancement not only enhances our ability to model complex graph structures but also deepens our understanding of the mechanisms driving GNNs.

The relationships among the developed models in this thesis are illustrated in Figure 1.3.

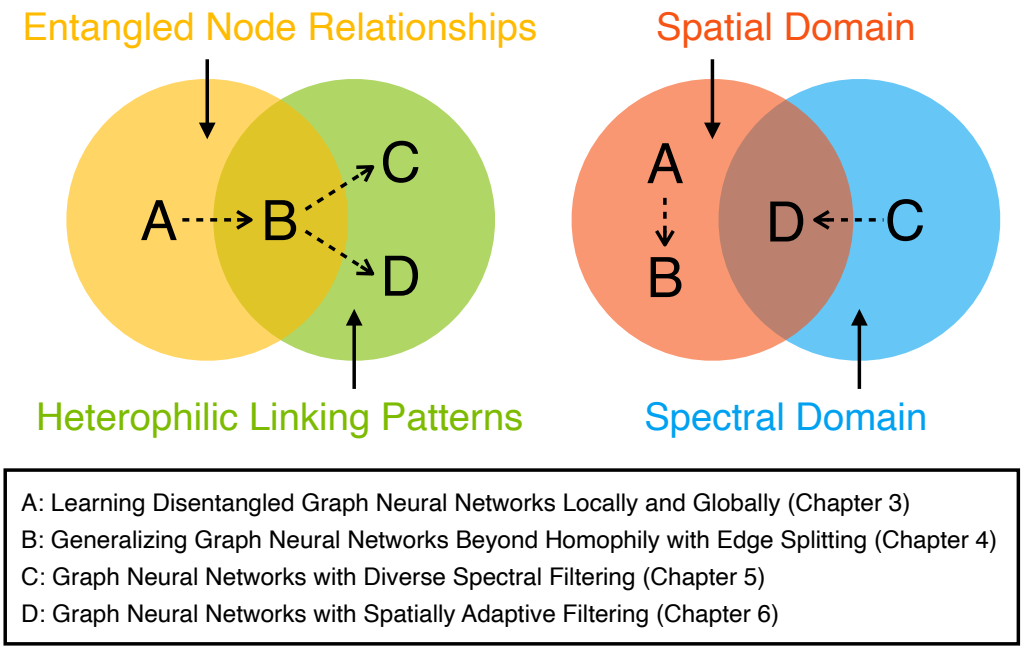


Figure 1.3: Relations among the developed models in this thesis, analyzed from both problem-centric (left) and model-centric (right) perspectives.

1.3 Thesis Outline

This thesis investigates how to enhance the learning and representational capabilities of Graph Neural Networks (GNNs) for complex structured environments, marked by *entangled node relationships* and *heterophilic linking patterns*. We start by introducing the motivation behind our research in **Chapter 1** and providing essential background knowledge in **Chapter 2**. Afterwards, this thesis is structured into four main chapters, each contributing to a cohesive exploration of our research topic.

Chapter 3 – Learning Disentangled Graph Neural Networks Locally and Globally. In this chapter, we tackle the issue of *entangled node relationships* in graph-structured data, where node connections are influenced by multiple latent factors. Despite their success, traditional GNNs tend to simplify these complex, entangled relationships, by adopting a holistic representation learning, and thereby ignore the nuanced distinctions within the graph caused by different factors. This oversight may

render the learned representations less explainable and informative. Furthermore, their overemphasis on local graph information may lead to a loss of global graph knowledge. To this end, we propose the Local-Global Disentanglement (LGD) framework, taking advantage of both local and global information for disentangling node representations in the latent space. Specifically, we propose to represent a disentangled latent continuous space with a statistical mixture model, by leveraging neighborhood routing mechanism locally. From the latent space, various new graphs can then be disentangled and learned, to overall reflect the hidden structures with respect to different factors. After that, a novel regularizer and global message passing is designed and employed to amplify the diversity of the disentangled representations across varying factors (inter-factor diversity) but also to enhance their consistency within each factor (intra-factor consistency). Extensive evaluations on both synthetic and real-world datasets show LGD offer clearer interpretations of complex networks and significantly improves GNNs' performance.

Chapter 4 – Generalizing Graph Neural Networks Beyond Homophily with Edge Splitting. In this chapter, we build upon the insights from Chapter 3 where we explored *entangled node relationships*, to tackle the challenge of *heterophilic linking patterns*. This challenge on graphs is characterized by a significant portion of edges connecting nodes from different classes – a situation that deviates from the standard homophily (or smoothness) assumption prevalent in conventional GNNs. To address this divergence, we propose a novel hypothesis named "disentangled smoothness assumption". It posits that both task-relevant and irrelevant factors shape the graph's structure, determining the connections between nodes within the same class as well as across different classes. In light of this, we develop the Edge Splitting (ES-) GNN framework to adaptively distinguish between graph edges either relevant or irrelevant to learning tasks. This process dynamically partitions the original graph into two subgraphs, each with the same node set but complementary edge sets. Following this, information is propagated separately on these subgraphs, and edge splitting is alternatively conducted, thus disentangling task-relevant from irrelevant features. We conduct extensive experiments to demonstrate the superiority of ES-GNN on graphs with varying homophily levels.

Chapter 5 – Graph Neural Networks with Diverse Spectral Filtering. In this chapter, we investigate the nuances of *heterophilic linking patterns*, revealing a notable presence of regional heterogeneity. While certain spectral GNNs are able to navigate arbitrary label patterns using learnable filters, we identified a critical shortfall in their approach: homogeneous spectral filtering often fails to capture the uncovered local structural variations among nodes in complex networks. This observation led to the development of our novel Diverse Spectral Filtering (DSF) framework. DSF revolutionizes spectral GNNs with node-specific filter weights to exploit the varying local structure properly. Particularly, the diverse filter weights consist of two components – A global one shared among all nodes, and a local one that varies along network edges to reflect node difference arising from distinct graph parts – to balance between local and global information. As such, not only can the global graph characteristics be captured, but also the diverse local patterns can be mined with awareness of different node positions. Interestingly, we formulate a novel optimization problem to facilitate the learning of diverse filters, which also enables the enhancement of spectral GNNs with our DSF framework.

Chapter 6 – Graph Neural Networks with Spatially Adaptive Filtering. In this chapter, we extend the analysis in Chapter 5 and delve deeper into the spatial implications of spectral GNNs, asking: what information is essentially encoded by spectral GNNs in the spatial domain? Previously, our efforts were concentrated on enhancing spectral GNNs within their native domain. Now, we shift our focus to a cross-domain exploration, offering a fresh perspective by rethinking spectral GNNs from a spatial lens. We establish a theoretical connection between spectral filtering and spatial aggregation, unveiling that spectral GNNs implicitly modify the original connections among nodes in the spatial domain. Both theoretical and empirical investigations reveal that the adapted new graph not only exhibits non-locality but also accommodates signed edge weights, reflecting global label relationships among nodes. These findings not only illustrate the interpretable role of spectral GNNs in the spatial domain but also challenge spectral GNNs' traditional reliance on fixed-order polynomials, which often

overlook global information. Built upon our theoretical findings, we propose a novel Spatially Adaptive Filtering (SAF) framework, which leverages the adapted new graph by spectral filtering for an auxiliary non-local aggregation. Notably, our proposed SAF comprehensively models both node similarity and dissimilarity from a global perspective, therefore alleviating persistent deficiencies of GNNs related to long-range dependencies and graph heterophily.

Finally, in **Chapter 7**, we conclude with a summary of the thesis content and outline future research directions.

Chapter 2

Research Background

In this chapter, we begin by presenting the preliminary knowledge essential for this thesis, followed by a discussion on several background topics. Particularly, we formalize graph notations in Section 2.1.1, outline the graph problem this thesis mainly investigates in Section 2.1.2, review the historical development of graph machine learning in Section 2.2.1, introduce Graph Neural Networks (GNN) in Section 2.2.2, and highlight recent advancements in complex graph learning in Section 2.2.3. A more detailed overview of related works is provided in subsequent chapters, where some models may be revisited to ensure each chapter is self-contained and to emphasize their relevance across different contexts.

2.1 Preliminary

In this section, we establish the foundational concepts necessary for understanding the subsequent discussions in this thesis. Specifically, we will formalize the graph notations and define the core problems that this thesis aims to address.

2.1.1 Notations

In this thesis, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is used to denote a static graph where $\mathcal{V}$ is a set of N nodes, and $\mathcal{E}$ represents a edge set satisfying $e_{i,j} = (v_i, v_j) \in \mathcal{E}$ if two distinct nodes $v_i, v_j \in \mathcal{V}$

Table 2.1: Common Notations.

Notations	Descriptions
$\mathcal{G}$	A graph.
$\mathcal{V}$	The node set of a graph.
$\mathcal{E}$	The edge set of a graph.
v	A node $v \in \mathcal{V}$.
$(v_i, v_j) \in \mathcal{E}$	A edge between nodes v_i and v_j.
$\mathbf{A}, \hat{\mathbf{A}}$	The adjacency matrix and its normalization.
$\mathbf{L}, \hat{\mathbf{L}}$	The Laplacian matrix and its normalization.
$\mathbf{D}$	The degree matrix of $\mathbf{A}$ and $\mathbf{D}_{i,i} = \sum_{j=1}^{N} \mathbf{A}_{i,j}$.
$\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_N)$	Eigenvalues of $\hat{\mathbf{L}}$ where each eigenvalue $\lambda_n \in [0, 2]$
$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N]$	Eigenvectors of $\hat{\mathbf{L}}$ where each eigenvector $\mathbf{u}_i \in \mathbb{R}^N$
$\mathbf{X} \in \mathbb{R}^{N \times f}$	The raw feature matrix of a graph.
$\mathbf{Z} \in \mathbb{R}^{N \times d}$	The hidden feature matrix of a graph.
$\mathbf{Y} \in \mathbb{R}^{N \times C}$	The ground truth label of nodes in one hot encoding.
$\hat{\mathbf{Y}} \in \mathbb{R}^{N \times C}$	The predicted label of nodes in one hot encoding.
$\mathbf{x}_i \in \mathbb{R}^f$	The raw features of a node v_i.
$\mathbf{z}_i \in \mathbb{R}^d$	The hidden features of a node v_i.
$\mathbf{y}_i \in \mathbb{R}^C$	The ground truth labels of a node v_i.
$\hat{\mathbf{y}}_i \in \mathbb{R}^C$	The predicted labels of a node v_i.
$\mathcal{N}_k(v)$	The k-hop neighborhood set of a node v.
$\mathbb{R}$	The set of real number.
N	The number of nodes in a graph.
C	The number of classes assigned to nodes.
f, d	The dimension of raw and hidden node features.
$\lvert \cdot \rvert$	The absolute value of a scalar or the size of a set.
$\lVert \cdot \rVert_2$	The L2 (Frobenius) norm of a vector (matrix).
$\lVert$	Column-wise vector concatenation.

are connected in the graph. We define the adjacency matrix as $\mathbf{A} \in \mathbb{R}^{N \times N}$ where $A_{i,j} = 1$ if there is a link between nodes v_i, v_j, and 0 otherwise. The graph Laplacian, as another important graph matrix, is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$ where $\mathbf{D}$ is the degree matrix. In practise, this matrix is often normalized into $\hat{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \hat{\mathbf{A}}$ for numerical stability and better generalization [23], where $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ refers to the normalized adjacency matrix. The graph Laplacian $\hat{\mathbf{L}}$ can be diagonalized [24] as $\hat{\mathbf{L}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$. Here, $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_N)$ consists of eigenvalues, and $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_N]$ represents eigenvectors. Based on this, the graph Fourier transform and inverse Fourier transform

can then be formulated as $\mathbf{S} = \mathscr{F}(\mathbf{X}) = \mathbf{U}^T\mathbf{X}$ and $\mathbf{X} = \mathscr{F}^{-1}(\mathbf{S}) = \mathbf{U}\mathbf{S}$, respectively, where $\mathbf{S}$ is called as the Fourier transformed features or Fourier coefficients of $\mathbf{X}$. For easy reading, we enumerate the commonly used notations throughout this thesis in Table 2.1.

2.1.2 Problem Definition

While there are plenty of graph learning problems, from link predictions to graph classifications, this thesis focuses on the foundational problem of transductive node classification on static graphs. This concentration enables a deeper investigation into the core principles of machine learning models, laying a solid foundation for broader graph-based applications. The formal problem definition is provided as follows, complemented by a straightforward toy example in Figure 2.1 for enhanced clarity.

Definition 1 (Transductive Node Classification)**.** Given a static graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}_L$ denotes the subset of nodes with known labels and $\mathcal{V}_U = \mathcal{V} \backslash \mathcal{V}_L$ denotes the rest without labels. The task is to construct a mapping rule or function $\mathbf{F}$ parameterized by $\mathbf{\Theta}$, from the graph topology $\mathbf{A}$ and node features $\mathbf{X}$ to the output $\hat{\mathbf{Y}} \in \mathbb{R}^{|\mathcal{V}| \times C}$, representing the predicted labels. Formally, we have $\mathbf{F}_\Theta(\mathbf{A}, \mathbf{X}, \mathbf{Y}_L) \rightarrow \hat{\mathbf{Y}}$, where $\mathbf{Y}_L \in \mathbb{R}^{|\mathcal{V}_L| \times C}$ are the known labels for $\mathcal{V}_L$. The function $\mathbf{F}$ aims to accurately fit the given node labels $\mathbf{Y}_L$ and robustly predict the class labels for the unlabeled nodes $\mathcal{V}_U$.

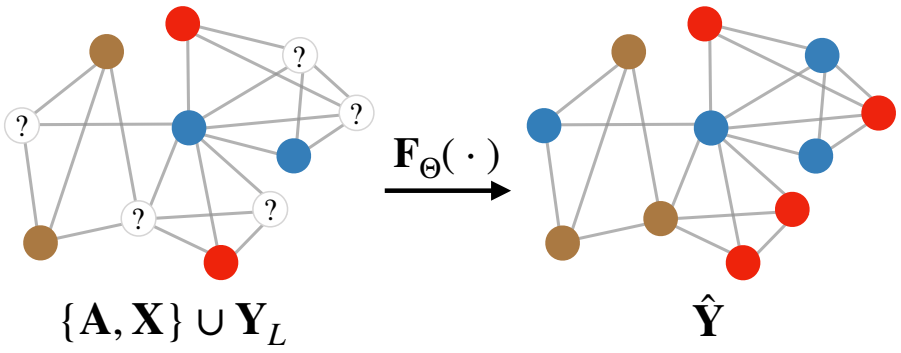


Figure 2.1: A toy example to show node classification tasks on a static graph with three node classes. Varying colors denote different labels, and nodes within the input graph that remain unfilled represent the targets for which we aim to predict labels.

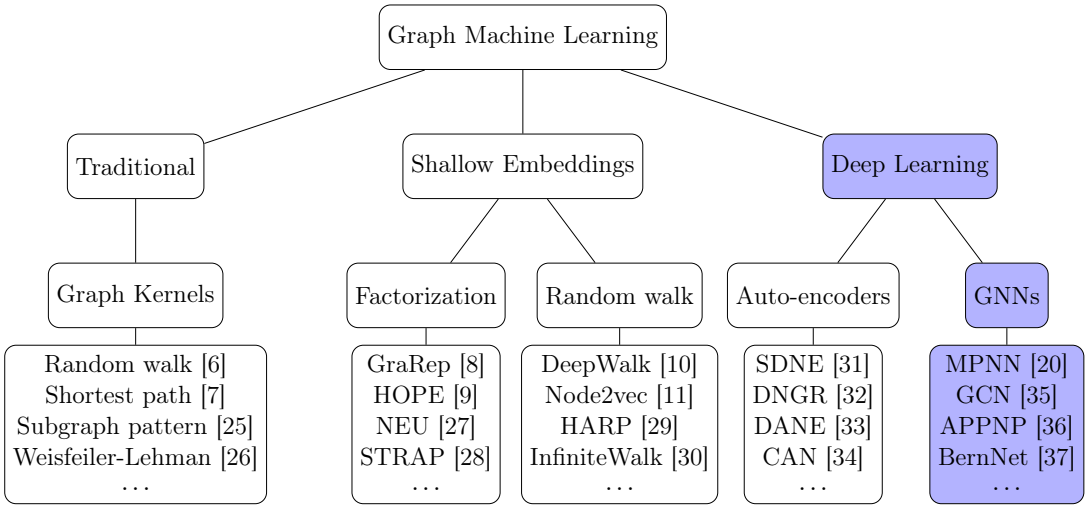


Figure 2.2: Taxonomy of graph machine learning models on static graphs. We divide them into traditional, shallow embeddings, and deep learning methods. This thesis focuses the field of Graph Neural Networks (GNNs) under the branch of deep learning.

2.2 Literature Review

In this section, we delve into the rich history of graph machine learning, tracing its evolution and pivotal developments (see Figure 2.2 for a concise taxonomy). We begin by exploring the traditional methodologies that have shaped the field. The focus then shifts to the latest research directions, particularly Graph Neural Networks (GNNs), which represent a significant stride forward in graph-based learning. Finally, we narrow our discussion to recent advancements in GNNs applied to complex graph learning, highlighting cutting-edge approaches and their implications for advancing the state of the art.

2.2.1 Historical Evolution of Graph Machine Learning

Over the past decades, the evolution of machine learning models for graph-structured data has been substantial. Drawing upon existing survey papers [38–46], in this section, we offer a concise overview of the traditional graph machine learning technologies. For a more comprehensive review, readers are encouraged to consult these seminal works.

Traditional Graph Kernels Graph kernels are historically dominant techniques in addressing classification tasks on graph-based data. These methods utilize kernel functions [47, 48] to measure the similarity between pairs of different graphs or atomic substructures within graphs, with detailed review available in work [42]. Despite their foundational role in modeling graph topology, graph kernel methods rely on predefined, hand-crafted criteria. This approach results in deterministic, rather than learnable, features, leading to models that cannot easily adjust to new or unseen data. Moreover, their intrinsic requirement for pairwise similarity calculations imposes significant computational challenges. This necessity limits scalability of graph kernel methods and introduces computational bottlenecks, preventing their application in environments where data volume are continuously expanding.

Shallow Embeddings In the past few years, there has been a surge in the development of graph embedding techniques. These methods seek to capture the structural essence of the graph, translating it into a low-dimensional vector embedding. The goal is to retain as much of the topological and attribute information within the embedding space, facilitating their integration with existing machine learning models. Unlike the earlier graph kernel methods, which relies heavily on manual feature engineering, current graph embedding approaches leverage machine learning algorithm to encode graph information in a data-driven way. Generally, existing graph embedding methods can be divided into the following categories: **1)** matrix factorization-based methods [8, 9, 27, 28] represent the initial attempts to learn graph embeddings. These methods usually first construct a matrix that captures the node proximity within the graph, with each matrix element $P_{i,j}$ reflecting the proximity measure between nodes v_i and v_j. Following this, dimension reduction techniques are applied to this matrix to derive graph embeddings. The core objective optimized in this process is $\min_{\mathbf{z}_i,\mathbf{z}_j} \sum_{v_i,v_j \in \mathcal{V}} |\mathbf{z}_i^T \mathbf{z}_j - P_{i,j}|$, where $\mathbf{z}_i$ refers to the learned embedding vector for node v_i; **2)** random walk-based methods [10, 11, 29] utilize random walks [49] to explore the graph structure, leveraging the generated paths to encode nodes' neighborhood information into their embeddings. This strategy ensures

that nodes co-occur frequently on these walks are considered similar, presenting a more flexible solution than the rigid proximity measures used in earlier factorization-based approaches. Despite success, these methods often suffered from the use of shallow architectures, struggling to exploit the depth and complexity of graph data.

Auto-encoders Shallow embedding methods struggle to capture the complex, non-linear structures often present in graphs. To address this limitation, graph auto-encoders were developed, utilizing deep neural networks for both encoding and decoding functions to better model these non-linearities. Unlike traditional methods that rely on graph regularization terms to reflect graph structure, auto-encoders encode the graph's adjacency matrix directly into the latent space. The architecture of auto-encoders includes multiple layers of non-linear transformations, with the encoder function being generally defined as $\mathbf{Z} = \mathbf{F}_{\text{enc}}(\mathbf{A}; \mathbf{\Theta})$. Training of these models focuses on minimizing the difference between the original graph structure and its reconstruction, with notable examples including SDNE [31] and DNGR [32]. Subsequently, DANE [33] introduced an advancement by incorporating node attributes $\mathbf{X}$ into the encoding process, enhancing the model's ability to capture the intricacies of graph data. Despite effectiveness, the inability of auto-encoders to fully integrate graph topology and node features limits their potential in accurately modeling the comprehensive dynamics of graph data, necessitating advancements to better synergize these critical components.

2.2.2 Graph Neural Networks

With the rapid success of deep learning, Graph Neural Networks (GNNs) have emerged as a forefront solution for analyzing graph-structured data, prompting the focus of this thesis on exploring GNNs as a pivotal method in graph machine learning. The choice to center on GNNs stems from their remarkable ability to seamlessly integrate both graph topology and node features. This integration is achieved via an end-to-end trainable fashion, which not only allows for a more efficient learning process but also significantly improves generalization across various graph types. Specifically, GNNs can be divided

into spatial- and spectral-based methods [39, 41, 45, 50], each offering unique viewpoints on processing graph-structured data. In the following, we offer their formal definitions, and right after each definition, we introduce several typical models within each category.

Definition 2 (Spatial-based Methods)**.** Spatial GNNs utilize the graph's spatial structure to update node representations through a three-step process: message-passing, aggregation, and feature update. This can be formulated as:

$$\mathbf{z}_i = \text{UPDATE}\left(\mathbf{x}_i, \text{AGGREGATE}\left(\left\{\text{MESSAGE}(\mathbf{x}_i, \mathbf{x}_j, e_{i,j}), \forall v_j \in \mathcal{N}(v_i)\right\}\right)\right),$$

where MESSAGE$(\cdot)$ allows nodes to exchange information with their neighbors $v_j \in \mathcal{N}(v_i)$ along edge $e_{i,j}$; AGGREGATE$(\cdot)$ combines the received messages into a single representation; UPDATE$(\cdot)$ updates the node's features by integrating its own features with the aggregated neighborhood information.

GraphSAGE [51]. GraphSAGE generates node representations by aggregating features from the sampled neighborhood for each node. Specifically, using the mean aggregator as an example, it computes the average features of a node and its neighbors, i.e.,

$$\mathbf{z}_i = \sigma\left(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{x}_i\} \cup \{\mathbf{x}_j | \forall v_j \in \mathcal{N}_{\text{sam}}(v_i)\})\right),$$

where $\mathbf{W}$ is the learnable weights, $\mathcal{N}_{\text{sam}}(v_i) \subset \mathcal{N}(v_i)$ refers to the sampled neighborhood of node v_i, σ is an activation function, and other symbols are early defined in the texts.

GIN [52]. Drawing inspiration from the Weisfeiler-Lehman (WL) test, the GIN introduces a set of principles to enhance the capabilities of GNNs, offering a straightforward yet effective architecture. GIN leverages strong theoretical foundations from WL test's functionality, updating the representations of nodes as follows (where ϵ can be a learnable parameter or a fixed scalar):

$$\mathbf{z}_i = \mathbf{W}\left((1+\epsilon)\ \mathbf{x}_i + \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{x}_j\right)$$

GAT [53]. GAT enhances the capacity of GNNs by incorporating attention mechanisms to weigh the significance of each neighbor's features dynamically. It computes the node representations using a multi-head technique as follows:

$$\mathbf{z}_i = \Big\Vert_{k=1}^{K} \sigma\left(\sum_{v_j \in \mathcal{N}(v_i)} \alpha_{i,j}^k \mathbf{W}^{(k)} \mathbf{x}_j \right)$$

where $\mathbf{W}^{(k)}$ is the learnable weights in the $k-$th head, and $\alpha_{i,j}$ is an attention coefficient measuring the importance of node v_j to node v_i, specifically computed as:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_j]\right)\right)}{\sum_{v_k \in \mathcal{N}(v_i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{x}_i \| \mathbf{W}\mathbf{x}_k]\right)\right)},$$

where $\mathbf{a}^T \in \mathbb{R}^{1\times 2d}$ is a transposed learnable vector and $\|$ denotes vector concatenation.

Definition 3 (Spectral-based Methods)**.** Spectral GNNs leverage the graph's spectral domain for convolution or, alternatively, spectral filtering. It selectively shrinks or amplifies the Fourier coefficients of node features and usually take the form as

$$\mathbf{Z} = \mathbf{g} *_{\mathcal{G}} \mathbf{X} = \mathbf{U}\ [\mathbf{g}(\mathbf{\Lambda}) \odot (\mathbf{U}^T\mathbf{X})] = \mathbf{U}\ \text{diag}(\mathbf{g}(\mathbf{\Lambda}))\ \mathbf{U}^T\mathbf{X} \tag{2.1}$$

where $g(\cdot) : [0, 2] \to \mathbb{R}$ is known as frequency response function. For models taking polynomial approximation, the equation can be directly formulated as $\mathbf{Z} = \mathbf{g}(\hat{\mathbf{L}})\mathbf{X} = \sum_{k=0}^{K} \psi_k P_k(\hat{\mathbf{L}})\mathbf{X}$ with $P_k(\cdot)$ and ψ_k denoting polynomial basis and coefficient, respectively.

Vanilla GCN [35]. The vanilla GCN truncates Chebyshev polynomials to a simple first-order for efficient graph convolution, which functions as a low-pass filter, i.e., $\mathbf{Z} = (\mathbf{I}+\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{X}$. To ensure numerical stability, it further leverages a renormalization trick to replace $\mathbf{I} + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ with $(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-\frac{1}{2}}$ and derives each graph convolution layer as

$$\mathbf{Z}^{(k+1)} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{Z}^{(k)}\mathbf{W}^{(k)},$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$, $\mathbf{Z}^{(0)} = \mathbf{X}$, and $\mathbf{W}^{(k)}$ is learnable weights in the k-th layer.

GPR-GNN [54]. GPR-GNN leverages the Generalized PageRank to approximate spectral graph filters with Monomial polynomial basis. The model architecture is formulated as

$$\mathbf{Z} = \sum_{k=0}^{K} \psi_k \hat{\tilde{\mathbf{A}}}^k f_\varphi(\mathbf{X}),$$

where $\hat{\tilde{\mathbf{A}}} = \tilde{\mathbf{D}}^{\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{\frac{1}{2}}$, and $f_\varphi(\mathbf{X})$ refers to a linear map with parameters φ. The approximated graph filter is $g_\psi(\tilde{\lambda}) = \sum_{k=0}^{K} \psi_k (1-\tilde{\lambda})^k$ where $\tilde{\lambda}$ is the eigenvalue of $\hat{\tilde{\mathbf{L}}} = \mathbf{I} - \hat{\tilde{\mathbf{A}}}$.

BernNet [37]. BernNet approximates graph filters with Bernstein polynomials to attain non-negative property, yielding the model expression as

$$\mathbf{Z} = \sum_{k=0}^{K} \psi_k \frac{1}{2^K} \binom{K}{k} (2\mathbf{I} - \hat{\mathbf{L}})^{K-k} \hat{\mathbf{L}}^k f_\varphi(\mathbf{X}).$$

The graph filter is defined as $g_\psi(\lambda) = \sum_{k=0}^{K} \psi_k \frac{1}{2^K} \binom{K}{k} (2-\lambda)^{K-k} \lambda^k$.

Spatial GNNs v.s. Spectral GNNs As two fundamental approaches operating in distinct domains, spatial and spectral GNNs each exhibit unique strengths and weaknesses in graph learning. We will discuss and summarize their relationship and implications as follows:

- Spatial GNNs primarily rely on a message-passing framework to update and learn node features within the graph's spatial domain. This approach excels at modeling local graph structures, offering an intuitive understanding of the learning process since it occurs within the visibly interpretable spatial domain. However, this focus on local features can sometimes lead to an oversight of global graph knowledge.

- Spectral GNNs implement convolution operations on graphs and filter graph signals in the spectral domain. This method effectively captures global information as it compresses all nodes into the graph spectrum, providing a robust mechanism for integrating broader graph context. Yet, the operations in the spectral domain are often more implicit and less intuitive, which can make it challenging to grasp and may overlook important local graph details.

Table 2.2: Summary of GNN models discussed, analyzed, and compared in this thesis. We divide them into "Spatial" for spatial-based methods, "Spectral" for spectral-based methods, and "Unified" for methods unifying both categories. Among them, LGD, ES-GNN, DSF, and SAF are the key models proposed within our research.

Year	**Spatial**	**Spectral**	**Unified**
2016		ChebNet [55]	
2017	MPNN [20], GraphSAGE [51]	MoNet [56], GCN [35]	
2018	GAT [53], JK-Net [57], GIN [52]		
2019	DisenGCN [1], APPNP [36]	SGC [58]	
2020	GCNII [59], IPGDN [2], FactorGCN [3], Geom-GCN [60], H2GCN [4]		
2021	FAGCN [61], Non-Local GNNs [62], GEN [63], WRGAT [64], PDE-GCN [65]	ARMA [66], GPR-GNN [54], BernNet [37]	GNN-LF [67], GNN-HF [67], ADA-UGNN [68]
2022	LGD [69], ES-GNN [70], NodeFormer [71], GloGNN++ [72]	PA-GNN [73], JacobiConv [74], ChebNetII [75]	
2023	MGNN [76]	DSF [77], Specformer [78], LON-GNN [79], OptBasisGNN [80]	ClenshawGCN [81], FE-GNN [82]
2024			SAF [83]

- Recognizing these differences has spurred various research efforts within the community, including the works developed in this thesis. From a model perspective, this thesis engages in nuanced discussions around spatial and spectral GNNs, unveiling subtle and distinct principles that may not be immediately apparent. This exploration contributes to a deeper understanding of how these methodologies can be harnessed and combined to enhance graph machine learning. Specifically, in Table 2.2, we list all GNN models discussed, compared, and analyzed in this thesis.

2.2.3 Complex Graph Learning

In this section, we introduce the intricate nature of graph data, focusing on two prevalent types of complexities – *entangled node relationships* and *heterophilic linking patterns*. These complexities present significant challenges in graph analysis, especially revealing the learning limitations in conventional Graph Neural Networks (GNNs) that employ local smoothness. We also offer a concise review of diverse strategies developed to

overcome these challenges within GNN frameworks.

Entangled Node Relationships Entangled node relationships in graphs represent a complex challenge where the connections between nodes are influenced by multiple latent factors. This complexity is particularly evident in social networks where individuals connect for many reasons such as family, work, and school. The multifaceted nature of these relationships even leads to multiple dimensions concealed within a single edge where the connected users are both friends and colleagues. Traditional Graph Neural Networks (GNNs), however, tend to simplify these complex, entangled relationships by adopting a holistic approach to representation learning. They aggregate neighborhood information as a whole without distinguishing between the different latent factors that contribute to the formation of each edge. This approach limits their ability to accurately interpret the entangled node relationships, rendering the learned representations hardly explainable and less informative.

The advent of disentangled representation learning has brought a new perspective to the forefront of graph data analysis. Initially gaining popularity in image representation learning [84–88], this approach aims to reveal the explanatory variables underlying the data, thereby producing representations that are not only interpretable [89, 90] but also inherently resistant to complex variations and adversarial attacks [91]. Yet, how to learn representations that disentangle the latent factors behind a graph remains a challenge due its irregular shape and non-independent and identically distributed (non-i.i.d.) samples.

One pioneering work in applying disentangled representation learning to graph data is DisenGCN [1]. Specifically, DiscenGCN employs a neighborhood routing mechanism to partition each node's neighborhood into multiple clusters, from which the information is aggregated independently to produce disentangled representations that describe different node aspects. Since then, several works have extended this concept within the graph learning domain [2, 69, 70, 92–94], each exploring different mechanisms for disentangling node relationships. For instance, Innovations such as IPGDN [95] and

LGD [69] improves upon DisenGCN by promoting independence between different latent factors and incorporating both local and global graph information, respectively. FactorGCN [3] takes a different approach by factorizing the original graph into multiple subgraphs, thereby capturing different graph aspects and facilitating graph disentangled representation learning.

These efforts highlight the growing interest in addressing the challenge of *entangled node relationships* within complex-structured data. By developing methods to disentangle the complex interplay of factors behind graph edges, researchers are paving the way for more nuanced and effective graph analysis techniques, ultimately enhancing the ability of GNNs to learn from and interpret graph data accurately.

Heterophilic Linking Patterns Early wisdom in the community was primarily dedicated to learning from graphs demonstrating strong homophiliy, where most connected nodes share similar attributes and same label [35, 36, 53]. It wasn't until 2020 that researchers began to emphasize the importance of exploring Graph Neural Networks (GNNs) on graphs with *heterophilic linking patterns* – scenarios where a considerable portion of connected nodes belong to different classes. This was notably marked by the studies of [60] and [4], introducing a new dimension to graph categorization based on the edge homophily ratio – $\mathcal{H} = \frac{\{|(v_i,v_j)|(v_i,v_j)\in\mathcal{E},\mathbf{y}_i=\mathbf{y}_j|\}}{|\mathcal{E}|}$ – which measures the fraction of graph edges that connect nodes from the same class.

Traditional GNNs often struggle in heterophilic environments due to their inherent inductive bias towards homophily–the assumption that connected nodes are likely to exhibit similar features or labels. This bias is embedded in the initial GNNs' mechanism for aggregating and propagating information, essentially functioning as low-pass filters that smooth information across a node's neighborhood, as documented in various studies [58, 96–98]. Researchers in works [67, 68] further suggests that many GNN models, including GCN [35], SGC [58], GAT [53], and APPNP [36], essentially address a graph signal denoising problem [24] under the smoothness assumption for connected nodes. However, this approach misaligns with heterophilic graphs, where node features

and labels vary more significantly, leading to an over-smoothing effect that masks the essential distinctions between inherently different nodes.

Addressing heterophilic linking patterns, recent efforts have sought to properly leverage a node's neighborhood. While attention mechanisms, as employed in works like [53, 99], offer a way forward, they often still enforce smoothness within a node's neighborhood, albeit focusing on key members only [67, 68, 98]. Innovatively, FAGCN [61] proposes an adaptive mechanism to model both similarities and differences among adjacent nodes. GPR-GNN [54] introduces a universal polynomial graph filter that utilizes learnable weights to capture both low- and high-frequency information effectively. Subsequent models like BernNet [37], JacobiConv [74], ChebNetII [75], LON-GNN [79], and OptBasisGNN [80] have been developed from various perspectives to further enhance the learning capabilities of spectral GNNs on heterophilic graphs through the regularized design of polynomial graph filters.

Another significant line of research focuses on capturing long-range information beyond immediate node proximity, proving particularly effective in heterophilic contexts where a node's relevance might extend beyond its direct connections. For example, Geom-GCN [60] extends traditional message passing with geometric aggregation in latent space, and H2GCN [4] explores higher-order neighborhoods to capture homophily-dominant information. WRGAT [64] reimagines the input graph as a multi-relational graph to enhance assortativity from both proximity and structural lens, while GEN [63] estimates the optimal graph for GNNs' learning with multi-order neighborhood information and Bayesian inference as guide. Non-local GNNs [62] implement non-local aggregation based on an efficient attention-guided sorting strategy. Both Nodeformer [71] and Specformer [78] take advantage of Transformer [100] to addresses long-range dependencies for spatial and spectral GNNs, respectively. GloGNN and GloGNN++ [72] capture global homophily beyond immediate neighborhoods by learning signed edge weights, demonstrating a grouping effect [101]. More recently, MGNN [76] leverages congruent-insensitivity and distance geometry principles to effectively handle heterophilic graphs within spatial domain.

The exploration of *heterophilic linking patterns* within complex structured enviroments using GNNs represent a significant advancement in graph representation learning. This thesis also contributes to this growing body of research by investigating and addressing the unique challenges posed by heterophilic graphs. By focusing on both the refinement of neighborhood utilization and the incorporation of long-range dependencies, we aim to enhance the adaptability and performance of GNNs in diverse graph settings, paving the way for effective graph analysis techniques and broader graph-based applications.

Chapter 3

Learning Disentangled Graph Neural Networks Locally and Globally

In this chapter, we confront the challenge of *entangled node relationships* within complex structured environments, a critical oversight of traditional GNNs. These models typically aggregate each node neighborhood as a whole, while overlooking the variations in node relationships caused by entangled graph factors. Moreover, their overemphasis on local details may compromise global graph knowledge. To address these, we propose the Local-Global Disentanglement (LGD) framework, aimed at disentangling latent factors behind complex node relationships. LGD employs a regularized neighborhood routing locally and a decoupled message passing globally. This dual aproach not only amplifies the diversity of disentangled representations across varying factors but also enhances their consistency within each factor. This work has been published in the IEEE Transactions on Neural Networks and Learning Systems (TNNLS) 2022.

3.1 Introduction

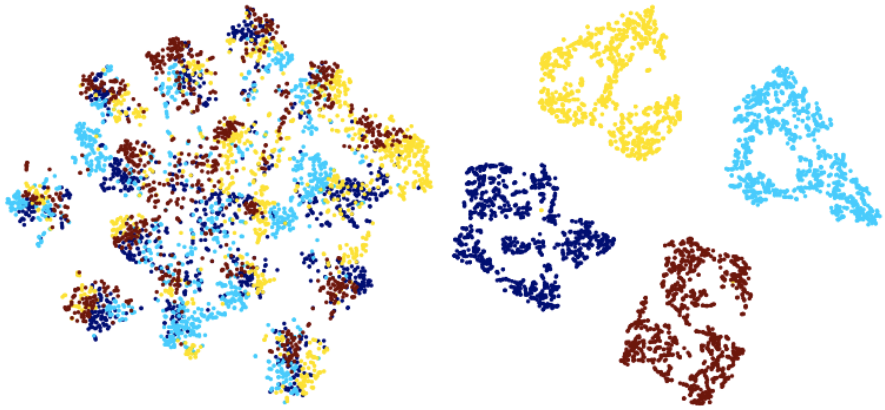
Graphs are emerging as an insightful structured modeling technique for capturing the similarity between data samples and identifying the relationship between entities [102–106]. To mine the domain-specific knowledge in graph-structured data, Graph Neural

Networks (GNNs) were proposed to integrate topological patterns and content features for node classification [35]. In the past years, GNNs have demonstrated excellent expressive power that leads to growing popularity in various graph learning tasks, such as node classification, link prediction, and recommendation [13, 44, 45].

Notably, most existing GNN models in the literature [35, 51, 53, 57, 58] focus on exploiting the local graph information and take a holistic approach, i.e., they interpret the node neighborhood as a perceptual whole while ignoring the within-distinctions. Yet a real-world graph typically contains heterogeneous node relations, driven by the entanglement of many latent factors. For example, a user in a social network usually links with others for various reasons, such as family, work, and/or hobby, which typically stores partial information in different types. The holistic approaches fail to capture the expressive partial information due to the neglect of the underlying factors, thereby rendering the learned representations heavily-entangled and less informative. On the other hand, while the benefits of modeling data both locally and globally have been well demonstrated in various machine learning models [107], there are few variants of GNNs (e.g., [108, 109]) incorporating both local and global graph information.

Recently, several works [1, 2, 92–94] make attempts to disentangle the latent factors behind graph data through neighborhood partition. Despite the novel design, they mostly rely on local node neighborhood only, similarly to most GNNs, which may bring unexpected issues. First, the information from local ranges can be significantly varied across the entire graph. Solely depending on it, they could easily produce latent representations that loses consensus cluster centroids with respect to different factors. This consequently may weaken the intra-factor correlation and inter-factor separability between disentangled features, therefore leading to a diminished interpretability. Second, the local neighborhood information can be scarce and limited especially in sparse graphs, which prohibits models from learning informative node aspects and yielding favourable performance boost. A detailed discussion will be given later on Section 3.3.

In this work, to address above issues, we propose a novel Local-Global Disentanglement (LGD) framework for Graph Neural Networks. The core idea is that we learn disentangled



(a) DisenGCN (Local Approach) (b) LGD (Our Approach)

Figure 3.1: Visualization of the disentangled latent units w.r.t. four latent factors on a synthetic graph. Several works have been made towards graph disentanglement learning, e.g. DisenGCN [1] and IPGDN [2]. They are all heavily relying on the local graph information, and here we only take DisenGCN, the basic one, as an example for intuitive comparison. Points with a different color indicate the disentangled latent units (for all nodes) of a different latent factor. In sharp contrast to DisenGCN, our LGD displays a highly disentangled pattern with strong inter-factor diversity and intra-factor consistency; it indicates high (low) correlations between intra-factor (inter-factor) features.

node representations by mining both local and global graph information. In particular, we first present a local disentanglement on nodes by partitioning their observable neighbors with the neighborhood routing mechanism. Then, the global information is attained by modeling the overall densities of nodes while considering different factors, and further disclosing the hidden node relations from different angles.

To this end, a statistical mixture modeling is performed on disentangled latent units to derive a latent continuous space. This enables a different density covering all the nodes, specific to a latent factor, in a different subspace [110, 111]. Accordingly, a novel regularizer is developed for promoting *inter-factor diversity*. It encourages the separability between these latent units according to different factors, and captures the uncorrelated information. After that, we manage to build a different new graph with sparse property by only connecting nearby neighbors within a different spatial region. These new graphs overall reflect the underlying data structures, i.e., the hidden node relations, in different aspects. Employing a message passing scheme over them

can efficiently encode the global information specific to different factors. This further strengthens *intra-factor consistency*, i.e., the correlation between disentangled features w.r.t. the same factor. Therefore, the disentangled informativeness of model can be enhanced in the output space. In sharp contrast to the local approach for graph disentanglement, Figure 3.1 clearly visualizes the benefit of learning disentangled node representations *both locally and globally*. In a nutshell, our contributions are summarized as below:

- We show through empirical analysis that the existing disentangled approaches may produce latent representations with weakly disentangled factors when solely relying on the local graph information. Therefore, the performance gain of these disentangled approaches becomes marginal when it comes to sparse graphs.
- To overcome the above limitations, we propose a novel LGD framework for Graph Neural Networks to disentangle the latent factors underlying the graph data in a more effective way. Specifically, by leveraging neighborhood routing *locally* and message passing *globally*, LGD can disentangle node representations with promoted *inter-factor diversity* and strengthened *intra-factor consistency*.
- Extensive evaluations on synthetic and five real-world data sets show the superiority of the proposed LGD over the state-of-the-arts both quantitatively and qualitatively. Specifically, LGD averagely outperforms the disentangled state-of-the-arts by 9.8%, 19.3%, 5.9%, 6.1%, and 3.5% on Blogcatalog, Flickr, Cora, Citeseer, and Pubmed data sets, respectively.

3.2 Related Works

Disentangled Representation Learning. Disentangled representation learning aims to reveal the explanatory latent variables behind data for generating a meaningful representation to admit intuitive explanations [89, 90]. Massive works have been developed on that topic [84–88]. It has been proved that disentangled representation is less susceptible

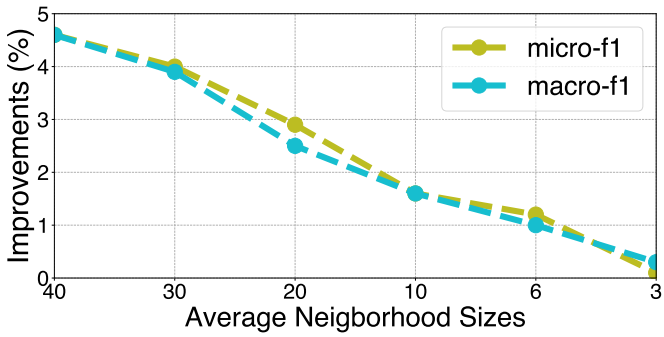


Figure 3.2: Relative improvements of DisenGCN upon GCN while varying the average neighborhood sizes of the synthetic graph. It can be observed that the boost performance of DisenGCN is becoming minor as the average neighborhood sizes decreasing.

to complex variants and more robust to adversarial attacks [89, 91]. However, most works are only applicable to Euclidean data structure. Recently, DisenGCN [1] made the first attempt towards disentangled node representation learning. Since then, works [2, 92–94] extending it start to emerge in the field of graph learning.

DisenGCN [1] hypothesizes that nodes on the graph are connected mainly due to different kinds of relationship caused by different factors m $(m = 1, 2, ..., M)$. It aims to identify these latent factors so as to learn disentangled node representations. Specifically, the node features, $\{\mathbf{x}_i \in \mathbb{R}^f | \forall v_i \in \mathcal{V}\}$, are first projected onto M subspaces in different channels. In each channel m, the hidden states $\mathbf{z}_{i,m} \in \mathbb{R}^{\frac{d}{M}}$ for each node v_i is given by

$$\mathbf{z}_{i,m} = \frac{\sigma(\mathbf{W}_m^T \mathbf{x}_i + \mathbf{b}_m)}{\|\sigma(\mathbf{W}_m^T \mathbf{x}_i + \mathbf{b}_m)\|_2} \tag{3.1}$$

where $\mathbf{W}_m \in \mathbb{R}^{f \times \frac{d}{M}}$ and $\mathbf{b}_m \in \mathbb{R}^{\frac{d}{M}}$ are learnable parameters, and σ is an activation function. Then, DisenGCN employs a neighborhood routing mechanism [1, Algorithm-1] (denoted as NRM in this work) to partition each node neighborhood into M clusters, from which the information is aggregated independently to produce disentangled latent units, e.g., $\{\bar{\mathbf{z}}_{i,1}, \bar{\mathbf{z}}_{i,2}, ..., \bar{\mathbf{z}}_{i,M}\}$ for node v_i, describing different node aspects. Finally, the disentangled node representation, $\bar{\mathbf{x}}_i \in \mathbb{R}^d$, can be attained by column-wise vector concatenation:

$$\bar{\mathbf{x}}_i = \bar{\mathbf{z}}_{i,1} \parallel \bar{\mathbf{z}}_{i,2} \parallel ... \parallel \bar{\mathbf{z}}_{i,M}. \tag{3.2}$$

Graph Structure Learning. Structure information plays a key role in graph learning and GNNs. Unfortunately, real-world data are typically noisy and incomplete and hence tend to generate imperfect or even poor graph structures [112]. To address this issue, many researchers try to remove noisy edges and infer the hidden relations among nodes so that an appropriate graph structure can be better learned [113]. These studies can generally be divided into two categories. The first one takes the approach of metric learning. It aims to learn a metric to decide whether two nodes are connected or not based on their features, and update the original structure with the new one by, e.g., interpolation. Representative ideas include [114–117]. The other one assumes that graphs are generated by sampling edges from a certain distribution [118–120]. These works focus on probabilistic modeling and train graph neural networks with the sampled graphs. Our work focuses on inferring the underlying node relations in the disentangled feature space, thereby falling into the group of metric learning. Another work related to ours is FactorGCN [3], which factorizes a graph into multiple subgraphs by edge clipping, and uses them to learn graph-level representation from different angles. Different from it, our approach generates multiple new graphs from a latent space, and employs a message passing along them to characterize nodes in different aspects.

3.3 Motivation

While the current disentangled state-of-the-arts reveal certain latent factors and demonstrate good performance in many scenarios [92–94], we argue that they are prone to produce weakly disentangled representations, and yield limited performance boost because of their heavy reliance on local graph information. To illustrate, we conduct two empirical investigations over a graph synthesized with four latent factors (see details in Section 3.6.1 for data generation). For simplification, we only take DisenGCN [1] as an example, because the other disentangled state-of-the-arts are mainly built on it.

First, we visualize the disentangled latent units of DisenGCN using t-SNE [121] in Figure 3.1 (a). At the micro-level, we can observe the separability between points with

different colors in some regions. When it comes to the macro-level, all points unexpectedly fall into discrete clusters and mixed together, indicating a weak disentanglement. As DisenGCN generates disentangled latent units that preserve some specific micro-meanings of the factor but lose the consistent macro-meaning (*intra-factor consistency*), this limits its potential in attaining higher performance. Additionally, DisenGCN only considers disentangling representations in different channels without ensuring their diversity w.r.t. different factors (*inter-factor diversity*). The learned representations are thus prone to preserve redundant information, as illustrated in Figure 3.1 (a).

Second, we further augment the synthetic graph by tuning the p value (which controls the density of the synthetic graph as described in Section 3.6.1) to generate graphs with different average neighborhood sizes. We then train GCN [35], the typical holistic approach, and DisenGCN for multi-label classification, and plot the relative improvements of DisenGCN upon GCN in Figure 3.2. From the figure, the improvements drop from approximately 5% to 0% as the average neighborhood size decreasing from 40 to 3. The results are consistent with the theoretical analysis in that DisenGCN relies on the local information only. When the graph is sparse (with limited local information), DisenGCN is inevitably getting less effective.

3.4 Methodology

We present a novel Local-Global Disentanglement (LGD) framework for Graph Neural Networks, to learn disentangled node representations both *locally and globally*, as presented in Figure 3.3. By leveraging the neighborhood routing mechanism [1] firstly, we attain disentangled latent units preserving *local* graph information w.r.t. different factors. Then, we propose to further incorporate *global* graph information. To this end, our LGD discloses the underling factor-aware relations among nodes, and utilize them to learn better disentangled representations with promoted *inter-factor diversity* and strengthened *intra-factor consistency*.

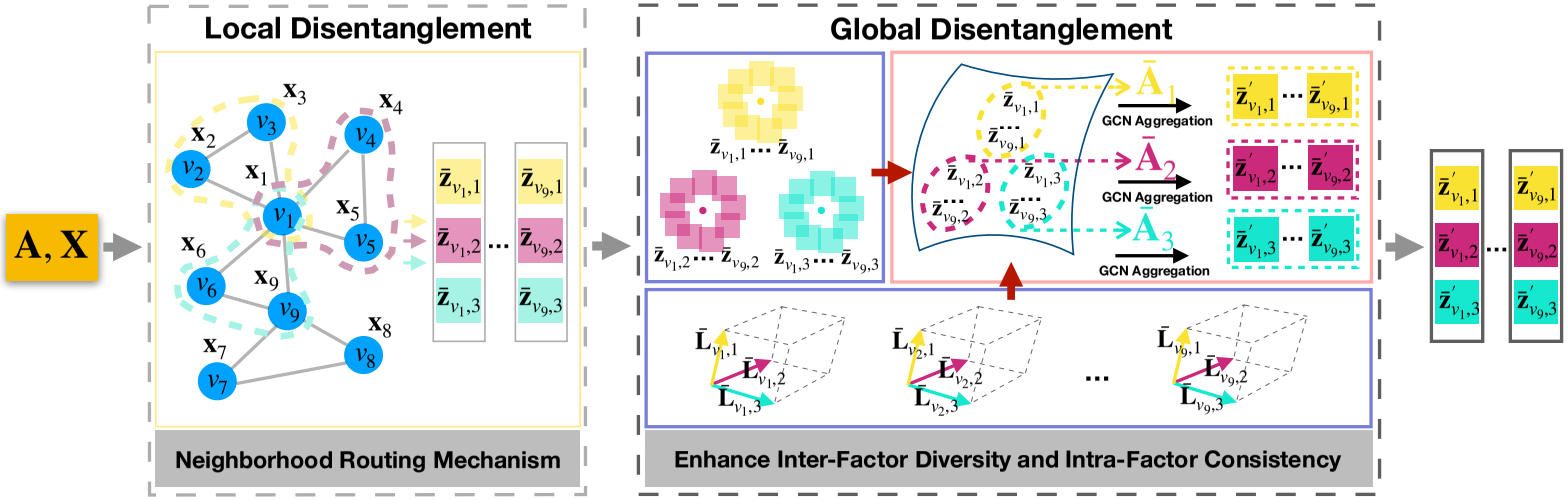


Figure 3.3: Illustration of our LGD layer with $M = 3$ latent factors, where $\mathbf{A}$ and $\mathbf{x}$ denote the adjacency matrix and feature matrix of the input graph, respectively. First, the node representations are locally disentangled by leveraging the neighborhood routing mechanism. These disentangled representations are then modeled in a latent continuous space, promoted with inter-factor diversity, from which various new graphs are built for further aggregation to strengthen intra-factor consistency.

3.4.1 Modeling Latent Continuous Space

We assume that the disentangled latent unit $\bar{\mathbf{z}}$ follows a Gaussian mixture distribution expressed as: $p(\bar{\mathbf{z}}) = \sum_{m=1}^{M} q(m)\mathcal{N}(\bar{\mathbf{z}}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$, where $\boldsymbol{\mu}_m \in \mathbb{R}^{\frac{d}{M}}$ and $\boldsymbol{\Sigma}_m \in \mathbb{R}^{\frac{d}{M} \times \frac{d}{M}}$ are the mean and covariance associated with factor m in latent space, and $q(m)$ is the prior probability of factor m and set as $\frac{1}{M}$ for equal consideration. To employ this assumption for space modeling, we maximize the conditional likelihood of disentangled latent units given their associated factor, i.e., $p(\bar{\mathbf{z}}_{i,m}|m)$ for each node v_i and factor m. It turns out to be equivalent to minimizing the negative log term after removing constants:

$$\mathcal{L}_{i,m}^{\text{space}} = (\bar{\mathbf{z}}_{i,m} - \boldsymbol{\mu}_m)^T \boldsymbol{\Sigma}_m^{-1} (\bar{\mathbf{z}}_{i,m} - \boldsymbol{\mu}_m). \tag{3.3}$$

Its functionality is quite similar to some supervised embedding methods [111, 122–126] that collapse categories into a low-dimensional embedding with the reduced within-class pairwise distance. Instead of constraining pairwise samples, we introduce Mahalanobis Distance [127] between each latent unit $\bar{\mathbf{z}}_{i,m}$ and its globally inferred center $\boldsymbol{\mu}_m$. Accordingly, a latent continuous space can be derived with a more compact data (sub-)manifolds, where the latent units are encouraged to be more discriminative with respect to their factor density. Finally, the regularization for space modeling is given by averaging over nodes and factors:

$$\mathcal{L}_{\text{space}} = \frac{1}{NM} \sum_{v_i \in \mathcal{V}} \sum_{m=1}^{M} \mathcal{L}_{i,m}^{\text{space}}. \tag{3.4}$$

3.4.2 Promoting Inter-Factor Diversity

Diversity-promoting learning aims to encourage different components in latent space models to stay mutually uncorrelated and different, and has been widely studied [128, 129]. In the previous section, we have derived a latent continuous space by independently modeling the density of each disentangled factor. However, the approximated distributions with different factors could still be overlapped [128]. Consequently, the disentangled latent units may preserve redundant information and lose informativeness. To cope with this problem, we propose to promote the diversity among different latent factors so as

to capture the uncorrelated information.

Particularly, we define the factor diversity with respect to the probabilities of a sampled latent unit staying close to different factor densities. Inspired by Determinant Point Process [130], we formulate the factor diversity for each node v_i as

$$\mathcal{F}_i^{div} = \det(\hat{\mathbf{P}}_i^T \hat{\mathbf{P}}_i), \tag{3.5}$$

where $\hat{\mathbf{P}}_i = [\frac{\mathbf{P}_{i,1}}{\|\mathbf{P}_{i,1}\|_2}, ..., \frac{\mathbf{P}_{i,M}}{\|\mathbf{P}_{i,M}\|_2}]$, and $\mathbf{P}_{i,m} = [\mathcal{N}(\bar{\mathbf{z}}_{i,m}; \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1), ..., \mathcal{N}(\bar{\mathbf{z}}_{i,m}; \boldsymbol{\mu}_M, \boldsymbol{\Sigma}_M)]^T \in \mathbb{R}^M$ contains the probabilities of the $\bar{\mathbf{z}}_{i,m}$ belonging to M different Gaussian distributions (w.r.t. M latent factors). By the property of Determinant [131], $\mathcal{F}_i^{div}$ is equal to the square of the volume spanned by the set $\{\frac{\mathbf{P}_{i,1}}{\|\mathbf{P}_{i,1}\|_2}, ..., \frac{\mathbf{P}_{i,M}}{\|\mathbf{P}_{i,M}\|_2}\}$, which offers elegantly an intuitive geometric interpretation as shown in Figure 3.3. To promote factor diversity, we introduce the diversity promoting regularizer as

$$\mathcal{L}_{\text{div}} = -\frac{1}{N} \sum_{v_i \in \mathcal{V}} \log(\mathcal{F}_i^{div}), \tag{3.6}$$

with the following proposition.

Proposition 1. Minimizing $\mathcal{L}_{\text{space}}$ and $\mathcal{L}_{\text{div}}$ simultaneously as $\mathcal{L} = \lambda_{\text{space}}\mathcal{L}_{\text{space}} + \lambda_{\text{div}}\mathcal{L}_{\text{div}}$ encourages the separatability between different factor densities, where λ_{space} and λ_{div} are positive regularization constants.

Proof. Penalizing $\mathcal{L}_{\text{space}}$ in Eq. (3.4) models the density of each factor with a latent continuous space, where most latent units will stay close to their centers. In other words, they will have the largest conditional likelihood according to their factors, e.g., the maximum values in $\mathbf{P}_{i,m} \in \mathbb{R}^M$ lie in its m-th element, i.e., $\mathcal{N}(\bar{\mathbf{z}}_{i,m}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$. On the other hand, as $\frac{\mathbf{P}_{i,m}}{\|\mathbf{P}_{i,m}\|_2}$ is normalized, the maximum value of $\mathcal{F}_i^{div}$ is 1 and can only be attained when $\mathbf{P}_{i,1}, ..., \mathbf{P}_{i,M}$ are orthogonal to each other. Therefore, minimizing $\mathcal{L}_{\text{div}}$ in Eq. (3.6) emphasizes the maximal value within $\mathbf{P}_{i,m}$ and promotes its discretization, thereby disjointing the possible overlaps between different factor densities. As a result, the disentangled latent units are encouraged to be separated w.r.t. different factors. □

This process can essentially prune the redundancy, enhance the disentangled informativeness, and finally promote the *inter-factor diversity*.

3.4.3 Strengthening Intra-factor Consistency

Although node relations can naturally be available in a graph, we believe that they are imperfect for disentangled graph learning due to data corruption or missing information. Take a huge and sparse graph as an example. It is difficult for most nodes to absorb sufficient information from their small neighborhood, especially in case of the average neighborhood size being much less than the number of latent factors to be disentangled. On the other hand, the original graph is essentially constructed from the raw feature space of nodes, and may not contain the desired topology after projecting node features in different channels for disentangling. To alleviate this issue, we propose to disclose the hidden relations among nodes from a latent space, and utilize them to encode more information, which proves beneficial.

The modeled latent space as described in Section 3.4.1 embeds the disentangled latent units of all the nodes, specific to a different factor, into a different subspace, from which a new graph can naturally be constructed by connecting nearby neighbors. These graphs are expected to reflect the overall structured information from different angles, and disclose the hidden node relations pertinent to different factors. Then, the disentangled latent units are allowed to propagate on their latent graphs followed by a neighborhood aggregation. As such, the factor-specific information can be encoded globally and selectively for nodes, further strengthening the *intra-factor consistency*. There are many ways for latent graph construction, and we list three popular ones below.

1) **k-Nearest-Neighbors [132] (kNN):** Two samples i and j are connected in a kNN graph if either of them belongs to k-nearest neighbors of the other. Formally, the adjacency matrix is given as:

$$\mathbf{A}^{\mathrm{kNN}}_{[i,j]} = \begin{cases} 1 & \mathcal{P}(i,j) \leq \mathcal{P}(i,i_k) \text{ or } \mathcal{P}(j,j_k) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{P}(\cdot)$ denotes pairwise distance, and i_k and j_k are the k-th nearest neighbors of samples i and j, respectively.

2) **Continuous k-Nearest-Neighbors [133] (CkNN):** It provides a more continuous version of kNN in case of modeling samples that are non-uniformly distributed:

$$\mathbf{A}^{\text{CkNN}}_{[i,j]} = \begin{cases} 1 & \mathcal{P}(i,j) < \delta\sqrt{\mathcal{P}(i,i_k)\mathcal{P}(j,j_k)} \\ 0 & \text{otherwise} \end{cases}$$

where δ is a scalar parameter controlling the density of the generated graphs.

3) **ϵ-Ball [134]:** Two samples i and j are connected in a ϵ-Ball graph if their distance is smaller than some scalar value $\epsilon > 0$:

$$\mathbf{A}^{\epsilon\text{-Ball}}_{[i,j]} = \begin{cases} 1 & \mathcal{P}(i,j) < \epsilon \\ 0 & \text{otherwise} \end{cases}.$$

In this work, we apply kNN [132] algorithm on $\{\mathbf{z}_{i,m} | v_i \in \mathcal{V}\}$ with $\mathcal{P}(\cdot)$ being Euclidean distance, and attain a latent graph for each factor m with adjacency matrix $\mathbf{A}_m$. On the other hand, there are also multiple choices on message passing frameworks from the basic to state-of-the-arts, e.g., GCN [35], GAT [53], GIN [52], FAGCN [61], etc., for encoding information on these latent graphs. In our LGD, we found that simply applying the basic GCN-aggregator gives us satisfactory results and a privilege of low computational cost. Therefore, the disentangled latent units are updated as follows:

$$\bar{\mathbf{Z}}'_m \leftarrow \tilde{\mathbf{D}}_m^{-\frac{1}{2}} \tilde{\mathbf{A}}_m \tilde{\mathbf{D}}_m^{-\frac{1}{2}} \bar{\mathbf{Z}}_m, \tag{3.7}$$

where $\tilde{\mathbf{A}}_m = \mathbf{A}_m + \mathbf{I}$, $\tilde{\mathbf{D}}_m = \mathbf{D}_m + \mathbf{I}$, $\mathbf{D}_m$ is the degree matrix, and $\bar{\mathbf{Z}}_m$ represents the feature matrix with the i-th row being $\bar{\mathbf{z}}^T_{i,m}$. Particularly, we call this proposed module as Global Aggregation, termed as $\mathbb{GA}$.

Algorithm 1 One Layer of LGD

Input: $\{\mathbf{x}_i \in \mathbb{R}^{d_{in}} | \forall v_i \in \mathcal{V}\}$, where $d_{in} = f$ in the first layer or d in the hidden layer.

$\mathbf{z}_{i,1}, \mathbf{z}_{i,2}, ..., \mathbf{z}_{i,M} \leftarrow \mathbf{x}_i \ \forall v_i \in \mathcal{V}$ by Eq. (4.2).
// Leverage Neighborhood Routing Mechanism [1] with T routing iterations.
for $v_i \in \mathcal{V}$ **do**
  $\bar{\mathbf{z}}_{i,m} \leftarrow \{\mathbf{z}_{i,m}\} \cup \{\mathbf{z}_{j,m} | \forall j \in \mathcal{N}(v_i)\}, \forall m = 1, 2, ..., M.$
end for
// Promoting Inter-factor Diversity.
Minimize $\mathcal{L}_{\text{space}}$ and $\mathcal{L}_{\text{div}}$ by Eq. (3.4) and (3.6).
// Strengthening Intra-factor Consistency.
for $m = 1, 2, ..., M$ **do**
  $\mathbf{A}_m \leftarrow \{\bar{\mathbf{z}}_{i,m} | \forall v_i \in \mathcal{V}\}$ by k-Nearest-Neighbors.
  $\{\bar{\mathbf{z}}'_{i,m} | \forall v_i \in \mathcal{V}\} \leftarrow \{\bar{\mathbf{z}}_{i,m} | \forall v_i \in \mathcal{V}\}$ with $\mathbf{A}_m$ by Eq. (3.7).
end for

Output: $\{\mathbf{x}'_i = \bar{\mathbf{z}}'_{i,1} \parallel \bar{\mathbf{z}}'_{i,2} \parallel \cdots \parallel \bar{\mathbf{z}}'_{i,M} | \forall v_i \in \mathcal{V}\}$.

3.5 Overall Algorithm

3.5.1 Network Architecture

We detail the general network architecture of the proposed LGD in this section. The pseudocode of a LGD's layer is presented in Algorithm 1, which can be stacked to exploit graph data sufficiently. In this work, by appending one single layer of our model after DisenGCN, we can even observe significant performance gain as later discussed in the section of experiments. Specifically, we adopt the ReLU activation function in Eq. (4.2) and apply dropout [135] in the end of each LGD's layer, and is only enabled in training. We can then have the output of l-th layer as $\{\mathbf{x}_i^{(l)} | \forall v_i \in \mathcal{V}\} = \text{Dropout}(\mathbf{F}_{\theta^{(l)}}(\{\mathbf{x}_i^{(l-1)} | \forall v_i \in \mathcal{V}\}))$, where $1 \leq l \leq L$, L denotes the total number of layers, $\mathbf{x}_i^{(0)}$ is initialized with the raw features $\mathbf{x}_i$, and $\mathbf{F}_{\theta^{(l)}}$ refers to LGD's l-th layer. This work focuses on node classification tasks with the output representations $\{\mathbf{x}_i^{(L)} \in \mathbb{R}^d | \forall v_i \in \mathcal{V}\}$. We denote $\hat{\mathbf{y}}_i \in \mathbb{R}^C$ as the class prediction for node v_i, which can then be calculated as $\sigma(\mathbf{W}^T \mathbf{x}_i^{(L)} + \mathbf{b})$ with σ being softmax and sigmoid respectively for single-label (multi-class) and multi-label node classification, where $\mathbf{W} \in \mathbb{R}^{d \times C}$, $\mathbf{b} \in \mathbb{R}^C$, and C denotes the number of class. Suppose we have training set $\mathcal{V}_{\text{trn}}$ and the ground truth label set $\{\mathbf{y}_i \in \mathbb{R}^C | \forall v_i \in \mathcal{V}_{\text{trn}}\}$ in one hot encoding. Then, the loss for classification task $\mathcal{L}_{\text{pred}}$ can be expressed as

Algorithm 2 Optimization Procedure of LGD

Params: $\{(\boldsymbol{\mu}_m^{(l)}, \boldsymbol{\Sigma}_m^{(l)}) | \forall m = 1, 2, ..., M, \forall l = 1, 2, ..., L\}$ and $\Theta = \{\theta^{(1)}, \theta^{(2)}, ..., \theta^{(L)}\}$

for number of training epochs **do**
  // Forward propagation.
  for $l = 1, 2, ..., L$ **do**
    $\{\mathbf{x}_i^{(l)} | \forall v_i \in \mathcal{V}\} \leftarrow \mathbf{F}_{\theta^{(l)}}(\{\mathbf{x}_i^{(l-1)} | \forall v_i \in \mathcal{V}\})$
    Calculate $\mathcal{L}_{\text{space}}^{(l)}$ and $\mathcal{L}_{\text{div}}^{(l)}$ by Eq. (3.4) and (3.6).
  end for
  Calculate $\mathcal{L}_{\text{total}}$ with $(\lambda_{\text{space}}, \lambda_{\text{div}})$ by Eq. (3.8).
  // Back propagation.
  for $l = 1, 2, ..., L$ **do**
    // Update $\theta^{(l)}$ with learning rate η.
    $\theta^{(l)} \leftarrow \theta^{(l)} - \eta \nabla_{\theta^{(l)}} \mathcal{L}_{\text{total}}$
    // Update $\{(\boldsymbol{\mu}_m^{(l)}, \boldsymbol{\Sigma}_m^{(l)})\}_{m=1}^M$ with update rate α.
    $\{\mathbf{x}_i^{(l)} = \bar{\mathbf{z}}_{i,1}^{(l)} \parallel \bar{\mathbf{z}}_{i,2}^{(l)} \parallel \cdots \parallel \bar{\mathbf{z}}_{i,M}^{(l)} | \forall v_i \in \mathcal{V}\} \leftarrow \mathbf{F}_{\theta^{(l)}}(\{\mathbf{x}_i^{(l-1)} | \forall v_i \in \mathcal{V}\})$
    for $m = 1, 2, ..., M$ **do**
      // Calculate the new values of mean and covariance.
      $\boldsymbol{\mu}_m^{\text{new}} \leftarrow \frac{1}{N} \sum_{v_i \in \mathcal{V}} \bar{\mathbf{z}}_{i,m}^{(l)}$
      $\boldsymbol{\Sigma}_m^{\text{new}} \leftarrow \frac{1}{N} \sum_{v_i \in \mathcal{V}} (\bar{\mathbf{z}}_{i,m}^{(l)} - \boldsymbol{\mu}_m^{\text{new}})(\bar{\mathbf{z}}_{i,m}^{(l)} - \boldsymbol{\mu}_m^{\text{new}})^T$
      // Update the values of mean and covariance.
      $\boldsymbol{\mu}_m^{(l)} \leftarrow (1 - \alpha)\boldsymbol{\mu}_m^{(l)} + \alpha \boldsymbol{\mu}_m^{\text{new}}$
      $\boldsymbol{\Sigma}_m^{(l)} \leftarrow (1 - \alpha)\boldsymbol{\Sigma}_m^{(l)} + \alpha \boldsymbol{\Sigma}_m^{\text{new}}$
    end for
  end for
end for

$-\frac{1}{|\mathcal{V}_{\text{trn}}|} \sum_i^{\mathcal{V}_{\text{trn}}} \mathbf{y}_i^T \log(\hat{\mathbf{y}}_i)$ and $-\frac{1}{|\mathcal{V}_{\text{trn}}|} \sum_i^{\mathcal{V}_{\text{trn}}} [\mathbf{y}_i^T \log((\hat{\mathbf{y}}_i)) + (\mathbf{1} - \mathbf{y}_i)^T \log(\mathbf{1} - \hat{\mathbf{y}}_i)]$ for single-label (multi-class) and multi-label node classification, respectively. Combing the derived regularization terms in Eq. (3.4) and (3.6), we have the final optimization objective:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{pred}} + \sum_{l=1}^{L} \lambda^{(l)} (\lambda_{\text{space}} \mathcal{L}_{\text{space}}^{(l)} + \lambda_{\text{div}} \mathcal{L}_{\text{div}}^{(l)}), \tag{3.8}$$

where $\mathcal{L}_{\text{space}}^{(l)}$ and $\mathcal{L}_{\text{div}}^{(l)}$ are the regularization terms calculated in l-th layer, λ_{space} and λ_{div} are the corresponding regularization coefficients, and $\lambda^{(l)}$ is taken as 10^{l-L} to grow the impact of $\mathcal{L}_{\text{space}}^{(l)}$ and $\mathcal{L}_{\text{div}}^{(l)}$ as the layer goes deeper within a proper range.

3.5.2 Computational Complexity

In this section, we provide a brief computational analysis on the proposed LGD. Compared to DisenGCN in training, our LGD additionally needs to compute the means and covariance matrixes of Gaussian mixtures in Eq. (3.4). In our work, instead of learning them with stochastic gradient, we employ iterative updating with newly computed values from latent features as detailed in Algorithm 2. This optimization technique is not unique in our work but has been widely adopted in multiple research fields [136–138]. It not only enables a lighter computational cost in a low dimensional latent space, but also provides a stable convergent property as empirically verified in our experiments. On the other hand, we have theoretically analyze the time complexity of our model as $O(fdN + 2|\mathcal{E}|d + (N + (d-1)k)N)$, where the overhead part compared to DisenGCN is $O((N + (d-1)k)N)$ brought by the inference of latent graphs. It suggests that the additional computational cost is mostly influenced by data size, of which the empirical study is provided in Section 3.7.2. We argue that our proposed LGD is still reasonably efficient in practice, especially when we consider the significant performance gains as verified in our experiments.

3.6 Experiments

In this section, we show the effectiveness of our LGD with experiments on five real-world and one synthetic data sets in node classification and factor disentanglement. We also study the generalization behavior and computational complexity of our model in comparison with DisenGCN. Finally, parameter sensitivity and module ablation study are provided.

3.6.1 Experimental Setup

Synthetic Data. To investigate the behavior of LGD on graphs with arbitrary number of latent factors, we also construct synthetic graphs. In detail, we first generate m Erdős-Rényi random graphs with 1,000 nodes and 16 classes, where nodes connect each

Table 3.1: Dataset Statistics

Datasets	# Nodes	# Edges	# Features	# Classes	# Train	# Validation	# Test
Blogcatalog	5,196	171,743	8,189	6	519	1,039	3,638
Flickr	7,575	239,738	12,047	9	757	1,515	5,303
Cora	2,708	5,429	1,433	7	140	500	1,000
Citeseer	3,327	4,732	3,703	6	120	500	1,000
Pubmed	19,717	44,338	500	3	60	500	1,000

other with probability p if they are in the same class, with probability q otherwise. Then, we merge these generated graphs, by summing the adjacency matrix and turning the element-value bigger than zero to one, to obtain the final synthetic graphs with m latent factors. There are $16\times m$ classes, and each node is assigned with m labels according to the original ones in the m random graphs. The rows of the adjacency matrix are taken as the node representations. Following [1], we set q to $3e^{-5}$ and tune p value such that the average neighborhood size is around 40.

Real-World Datasets. Blogcatalog [139] is a community of online blogging where users are connected by following each other, labeled by predefined categories of interests, and given features generated based on their personal descriptions. Flickr [139] is an multimedia sharing platform, where the users follow each other online with interest tags and joined groups respectively being their features and labels. Cora, Citeseer, and Pubmed [140] are three typically sparse citations networks whose average neighborhood sizes are 3.9, 2.8, and 4.5, respectively. Their nodes are documents connected by undirected citations, and assigned with one topic for each as well as features of bags-of-words. Data statistics are listed in Table 3.1.

Baselines. We compare our model with a number of mainstream GCN methods, including the state-of-the-arts, as the baselines: (1) MoNet [56] makes the first attempt to generalize convolutional neural networks to non-Euclidean graph data; (2) GCN [35] approximates graph Laplacian with Chebyshev expainsion; (3) GraphSAGE [51] is an inductive framework for large graph learning where we only consider one of its variant with GCN aggregator; (4) GAT [53] combines the attention mechanism with graph neural networks to aggregate information with important neighbors; (5) SGC [58] simplifies

GCN by removing nonlinearities; (6) JK-Net [57] leverage multi-hop neighborhood of nodes to capture structure-aware information; (7) DisenGCN [1] partitions node neighborhood to learn disentangled node representations; (8) IPGDN [2] further extends DisenGCN [1] with promoted independence between different factors; (9) FactorGCN [3] employs a graph factorization to disentangle different graph aspects.

Implementation Details. For all the baselines and our model, we set $d = 64$ as the hidden dimension for fair comparison, and tune the hyper-parameters on the validation split of each data set using Optuna [141] for efficiency. Following DisenGCN, we set $T = 7$ as the number of routing iterations. For semi-supervised node classification on real-world data sets, we apply dropout $\sim \{0, 0.05, ..., 1\}$ with step 0.05, learning rate $\sim$ [1e−3, 5e−1], weight decay $\sim$ [1e−4, 5e−1], update rate $\sim [0.1, 0.9]$ for $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$, the number of layers $\sim \{1, 2, ..., 10\}$, the number of channel $M \sim \{2, 4, ..., 16\}$ with step 2, and $m\lfloor \frac{d}{m} \rfloor$ as the hidden dimension in our model when m is not divisible by d. For multi-label classification on the synthetic data set, with a slight difference, we apply learning rate $\sim$ [5e−4, 5e−3], and weight decay $\sim$ [1e−3, 1e−2]. With the best hyper-parameters, we train models within $1,000$ epochs using the early-stopping strategy with a patience of 100 epochs, and report the average performance in 10 runs on the test split.

3.6.2 Quantitative Evaluation

In this section, we evaluate our model quantitatively in tasks of semi-supervised node classification and multi-label node classification.

Semi-supervised Node Classification. We evaluate models on the standard splits provided by [120] for Blogcatalog and Flickr datasets. For Cora, Citeseer, and Pubmed, we follow the experimental protocol established by [35, 53] and use their provided data splits. The hyper-parameters are tuned over the validation split and the average performance with 10 different model initializations is reported in Table 3.2. As observed, for social networks, the disentangled approaches including DisenGCN, IPGDN, and ours outperform the holistic approaches. This is partially because the users tend to have multiple different relationships (family, friend, and/or college) between their neighbors,

Table 3.2: Semi-supervised node classification accuracies (%).

Methods	**Blogcatalog**	**Flickr**	**Cora**	**Citeseer**	**Pubmed**
MoNet	74.7±0.4	61.7±0.7	79.6±1.5	70.2±1.3	78.0±0.5
GCN	73.8±0.3	56.6±0.4	81.8±1.0	71.8±1.3	78.7±0.5
GraphSAGE	73.7±0.3	56.3±0.4	81.9±0.9	71.3±1.3	79.0±0.6
GAT	56.7±5.0	45.1±1.0	81.9±0.8	73.1±0.8	78.8±0.7
SGC	74.5±0.3	61.4±0.2	82.4±0.5	72.4±0.5	79.4±0.2
JK-Net	76.5±0.3	64.6±0.4	82.0±0.9	73.0±0.9	79.1±0.4
DisenGCN	86.5±1.3	75.8±0.6	81.9±0.9	72.5±0.8	79.7±0.6
IPGDN	86.9±0.9	75.9±0.5	83.0±0.5	72.7±1.4	80.0±0.5
FactorGCN	78.4±1.3	47.0±1.7	72.9±2.2	59.6±1.8	74.4±0.8
LGD	**93.7±0.4**	**85.5±0.6**	**85.2±0.6**	**74.4±0.5**	**81.5±0.6**

and learning representations that recognize and disentangle the underlying factors could better describe the users from different angles. Although FactorGCN also considers different types of node relations, it fails on some networks and even performs worse than the holistic approaches. One main reason is that FactorGCN focuses on capturing different graph aspects from a global view while ignoring the local details important for node-level classification. On the other hand, our model achieves significant performance gains upon the disentangled state-of-the-arts averagely by 9.8% and 19.3% on Blogcatalog and Flickr, respectively. This demonstrates the benefits brought by further capturing rich global information. Importantly, the real-world social networks may be updated quickly, i.e., the users could frequently follow, or meet new friends. Therefore, the static network in a certain state cannot reflect the "true" relations between users. In this circumstance, the proposed LGD is able to connect the far reached but potentially related users by globally inferring the underlying graph structures, which may explain why significant performance improvement can be attained. In particular, for citation networks which are typically sparse, our model is able to boost the performance by a margin of 1.9% on average, showing its effectiveness in absorbing extra information from a global range.

Multi-label Node Classification. To validate the disentangling ability of the proposed LGD quantitatively, we apply MLP, i.e. a multi-layer perception, GCN, DisenGCN, and

Table 3.3: Micro-F1 (Left) and Macro-F1 (Right) scores (%) on synthetic graphs with different number of latent factors.

Methods	**4**		**6**		**8**		**10**		**12**	
MLP	79.3±0.5	77.9±0.7	55.5±0.4	54.8±0.6	37.0±0.8	36.0±0.8	25.9±0.6	24.5±0.7	21.2±0.8	20.1±0.9
GCN	74.5±0.8	78.3±0.9	56.3±0.7	55.6±0.9	38.2±0.9	37.2±1.0	28.0±0.7	26.9±0.5	23.1±0.8	22.2±0.9
DisenGCN	84.1±1.0	82.9±1.1	60.4±0.9	59.9±1.0	41.4±1.3	40.2±1.2	29.4±0.7	28.1±0.7	24.2±0.8	23.4±0.7
LGD	**87.2±0.5**	**86.1±0.5**	**65.0±0.5**	**64.2±0.6**	**43.6±0.7**	**42.5±0.6**	**30.2±0.5**	**28.8±0.5**	**26.1±0.5**	**25.1±0.5**

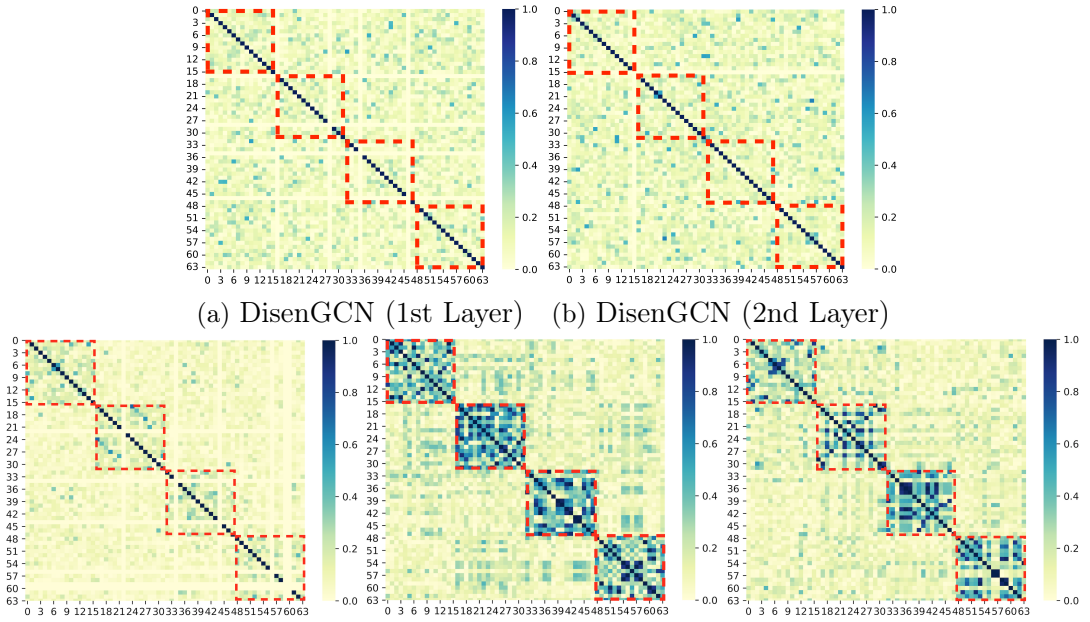
our model to train synthetic graphs with various number of latent factors for multi-label node classification. Specifically, we randomly split each data set into train/validation/test as 0.6/0.2/0.2, measure model performance in both Micro-F1 and Macro-F1 scores, and report them in Table 3.3. From the results, our model consistently outperforms others while varying the number of latent factors. Especially, LGD significantly outperforms DisenGCN by (Micro-F1) 4.6% and (Macro-F1) 4.3% on the synthetic graph with six latent factors.

3.6.3 Qualitative Evaluation

To gain more understanding of our proposed method, we conduct various qualitative experiments to take closer examinations in parallel with DisenGCN. These evaluations focus on the disentanglement performance and the learned embeddings' informativeness.

Visualization of Disentangled Representations. We plot in Figure 3.1 (b) a 2D visualization of the learned representations w.r.t. four latent factors on the synthetic graph. Compared to that of DisenGCN in Figure 3.1 (a), our model displays a highly disentangled pattern, evidenced by the intra-factor compactness and inter-factor separability. It also indicates that the common type of factor-specific information is captured, and globally shared by all nodes.

Correlation of Disentangled Features. The correlation analysis of the latent features learned by DisenGCN and our model is presented in Figure 3.4. As observed, our model showcases a more block-wise correlation pattern, which becomes denser in the second layer. We also analyze the feature correlation of our model while ablating the module



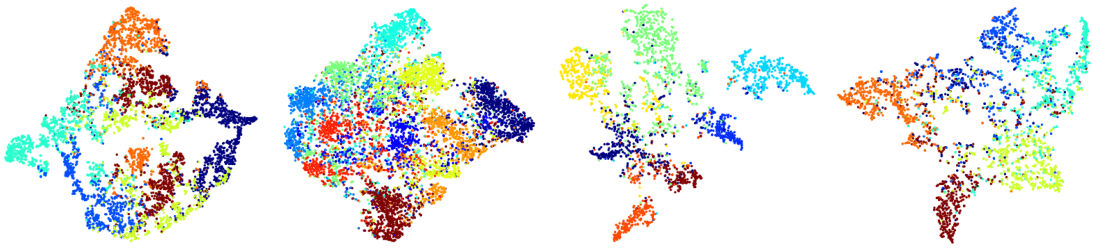
(a) DisenGCN (1st Layer) (b) DisenGCN (2nd Layer)

(c) LGD (1st Layer) (d) LGD (2nd Layer) (e) LGD w/o $\mathbb{GA}$ (2nd Layer)

Figure 3.4: Feature correlation analysis. The latent features are obtained on the test split of the synthetic graph with four latent factors.

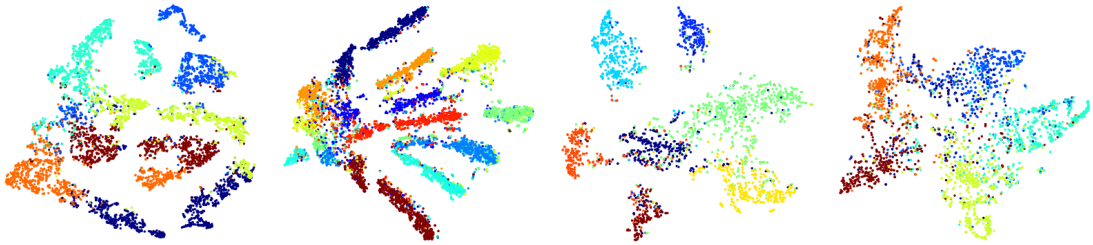
$\mathbb{GA}$, denoted as LGD* in Figure 3.4 (e). Though the block-wise pattern in Figure 3.4 (e) can still be observed, it is obviously weaker than that of LGD in Figure 3.4 (d). This verifies the significance of $\mathbb{GA}$. The captured global information, specific to each latent factor, strengthens the correlation between the intra-factor features, and enhances the interpretability and disentangling power.

Visualization of Node Embeddings. Figure 3.5 and Figure 3.6 provide an intuitive comparison between the learned node embeddings of DisenGCN and our model. It can be observed that the proposed LGD generally learns better node embeddings and exhibits a high intra-class similarity and inter-class difference. By absorbing rich global information specific to each latent factor, our model learns more informative node aspects and thus leads to superior discriminative power.



(a) Blogcatalog (b) Flickr (c) Cora (d) Citeseer

Figure 3.5: Visualization of node embeddings learned by DisenGCN



(a) Blogcatalog (b) Flickr (c) Cora (d) Citeseer

Figure 3.6: Visualization of node embeddings learned by LGD

3.7 Analysis and Discussion

3.7.1 Parameter and Ablation Study

In this subsection, we investigate the sensitivity of three essential hyper-parameters and perform ablation analysis over the proposed different modules.

Analysis of Space Modeling Coefficient λ_{space}. We plot the learning performance of our model w/o $\mathcal{L}_{\text{div}}$ while varying λ_{space} in Eq. (3.8). For example, we adopt a range of $\{0, 0.01, 0.05, 0.1, 0.5, 1\}$ on Flickr and report the learning performance in Figure 4.9 (b). In general, the accuracy goes up first and then drops; promising result can be attained by choosing λ_{space} from $[0.05, 0.5]$. Similar trends can also be observed on the other four data sets.

Analysis of Diversity Coefficient λ_{div}. We also examine the effect of λ_{div} by varying it value. For example, λ_{div} is changed from 0 to 0.5 on Citeseer. The results are shown in Figure 4.9)(d)-(f). Basically, λ_{div} is relatively robust within $[0, 0.1]$ for all the data

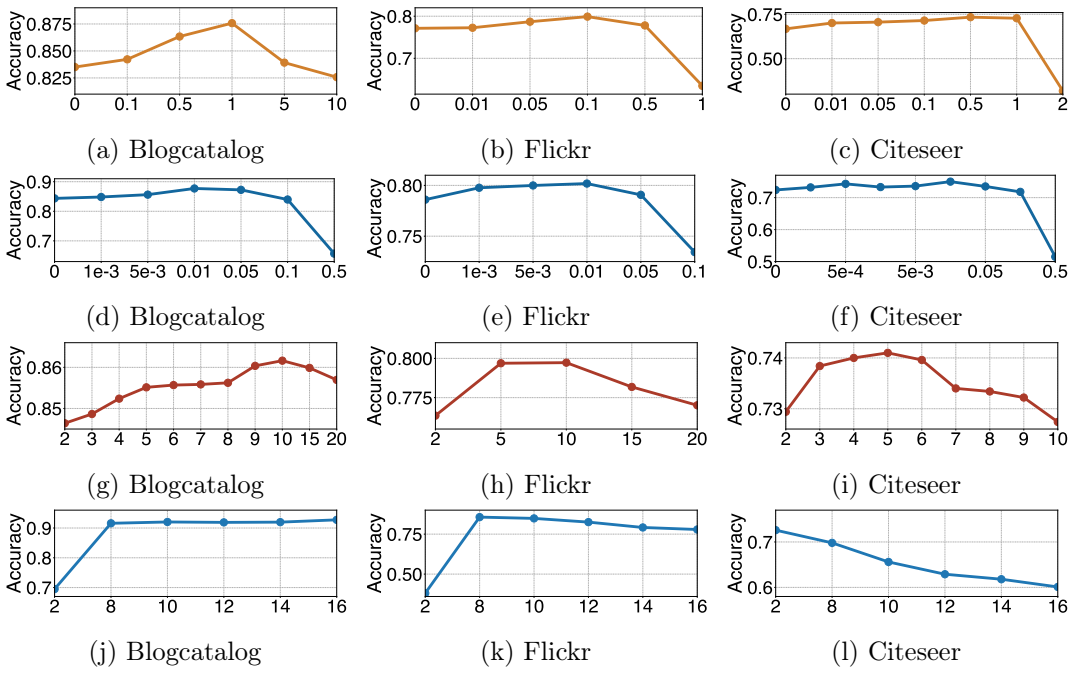


Figure 3.7: Analysis of parameters λ_{space} (orange), λ_{div} (blue), k (red), and M (green) from top to bottom rows.

sets except for Flickr whose stable range is $[0, 0.05]$. Once out of that range, the results drop to a low point, suggesting that overly emphasizing diversity could be harmful to model performance.

Analysis of Density Parameter k. Figure 4.9(g)-(i) displays the impact of k. The results are relatively stable while selecting k around 4 for Citeseer, and 10 for the rest. However, as k is larger, the accuracy performance deteriorates obviously. Such trend may be caused by noisy edges in cases of a large k, leading to inappropriate information sharing.

Analysis of Channel Number M. We test the effect of channel number M on real-world data sets on Figure 4.9(j)-(l). As can be observed, LGD attains its highest accuracy with the channel number around 16 for Blogcatalog and 8 for Flickr. In comparison, the ideal channel number is relatively smaller on citation networks, where LGD performs the best with M being 4. We also study its influence on the synthetic graph with eight predefined factors as an typical example. From Figure 3.8, our model performs the best

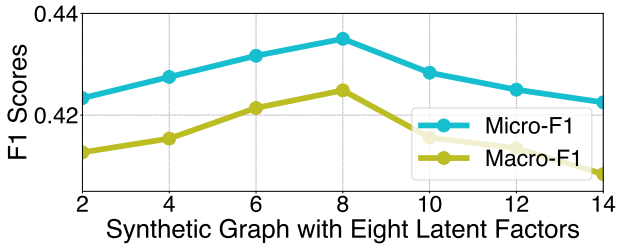


Figure 3.8: Analysis of parameter M on a synthetic graph

Table 3.4: Ablation analysis

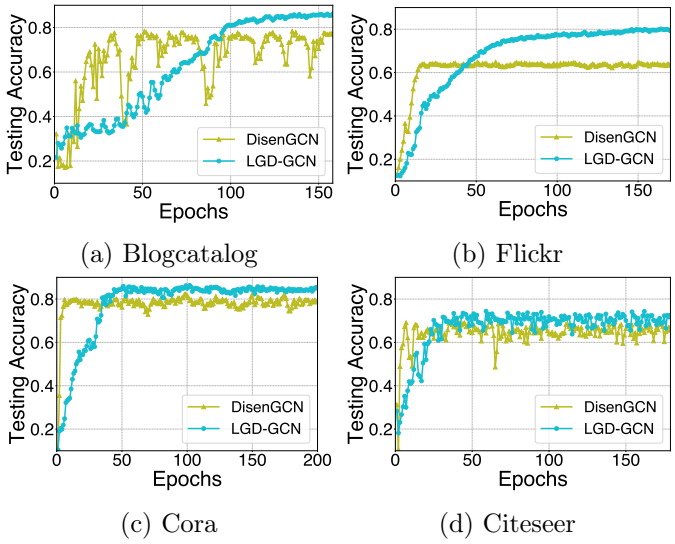
Variants	**Blogcatalog**	**Flickr**	**Cora**	**Citeseer**	**Pubmed**
LGD	93.7±0.4	85.5±0.6	85.2±0.6	74.4±0.5	81.5±0.6
w/o $\mathcal{L}_{\text{space}}$	87.9±1.7	78.9±2.1	81.1±1.2	70.1±0.9	79.3±1.6
w/o $\mathcal{L}_{\text{div}}$	93.7±0.5	85.2±0.7	84.0±0.9	74.0±0.8	80.8±0.7
w/o $\mathbb{GA}$	91.5±0.3	70.5±1.1	83.4±0.4	73.0±0.9	79.2±0.8
w/o NRM	83.3±0.3	76.2±2.1	61.9±1.6	22.1±2.9	77.5±0.6
w/ only NRM	86.1±0.5	69.6±0.6	81.2±1.2	68.9±1.5	78.5±1.1

when the number of channels is around 8, the true number of the latent factors.

Ablation Study. We first validate the contributions of the proposed modules denoted by $\mathcal{L}_{\text{space}}$, $\mathcal{L}_{\text{div}}$, and $\mathbb{GA}$ in node classification. As indicated in Table 3.4, removing any of these components typically results in significant drops in accuracy and increased statistical deviations. Furthermore, we investigate the impact of local and global modeling on model performance from an overall perspective. Specifically, we refer to global modeling as using LGD without the neighborhood routing mechanism (NRM) [1, Algorithm-1], and to local modeling as using the model with only NRM. From the results in the last two rows of Table 3.4, it is clear that while global modeling can enhance the baseline model's performance, relying solely on global modeling results in poor performance across many datasets, often worse than using local modeling alone (w/ only NRM). This highlights the critical role of local modeling, which establishes a stable foundation for global modeling, thereby allowing it to more effectively realize its value, as demonstrated by the results of our LGD model that integrates both local and global modeling strategies.

3.7.2 Generalization Behavior and Complexity Analysis.

In Figure 3.9, we plot the evolution of testing accuracy for both DisenGCN and our proposed LGD model as training epochs increases. It is empirically observed that although DisenGCN converges faster, its accuracy trajectory is marked by significant fluctuations across epochs and a tendency to get stuck into a local optimum, indicating a lower generalization capability on test datasets. On the other hand, our LGD model demonstrates a more stable and consistent increase in testing accuracy, ultimately converging to a notably higher peak. This suggests that LGD possesses superior generalization abilities, making it a more robust model for handling unknown environments as compared to DisenGCN. Regarding the pronounced fluctuations observed in Figure 3.9 (d), these can be interpreted as convergence within a relatively larger error threshold, as it occurs periodically within a roughly bounded range. Additionally, the similar patterns displayed by both DisenGCN and LGD suggest that this phenomenon is largely driven by the data rather than the models It should be noted that in this analysis, we have omitted the Pubmed dataset for the sake of time efficiency. The other two datasets, Cora and Citeseer, which are also citation networks, have been tested, suggesting that Pubmed would likely exhibit similar results. For complexity analysis, we first report the average training time (ms) per epoch in Table 5.1. On average, LGD is around 11.4% and 53.3% slower than DisenGCN on small data sets (Cora and Citeseer) and large datasets (Pubmed, Blogcatalog, and Flickr). That is mainly caused by the computation of latent graphs with the complexity of $O(N^2)$. However, with the significant performance gain, we believe that such costs may be worthwhile especially when the computational capability and stability are being steadily empowered. On the other hand, as we have the time complexity of our model as $O(fdN + 2|\mathcal{E}|d + (N + (d-1)k)N)$, there should be no trade-off between the channel number M and running time. However, our experimental findings show that the inference time of LGD gets approximately 0.02 ms increased with one channel added, which is mainly owing to the additional time consumed by looping channels in our implementation.



(a) Blogcatalog (b) Flickr

(c) Cora (d) Citeseer

Figure 3.9: Generalization behavior of DisenGCN and the proposed LGD w.r.t. training epochs.

Table 3.5: Average Training Time (ms) Per Epoch

Methods	**Blogcatalog**	**Flickr**	**Cora**	**Citeseer**	**Pubmed**
DisenGCN	15.9	22.0	28.4	35.4	116.8
LGD	26.4	33.8	31.6	39.5	163.9

3.7.3 Limitations and Future work

Despite the considerable success of our model in achieving robust results across various benchmarks, it is important to recognize that there are still limitations that need to be addressed. In this subsection, we outline these limitations and suggest potential directions for future research to enhance the model's performance and applicability.

- In this work, we have chosen to model the disentangled latent space using a Gaussian mixture model, weighting each mixture equally for simplicity. While this approach enhances computational efficiency, it may not always be optimal or even appropriate in many real-world scenarios where the complexity of data could require a more nuanced weighting strategy. The equal weighting scheme overlooks the potential variability in the significance of different mixtures, which might lead to suboptimal disentanglement performance in diverse settings. Future

research should explore more adaptable and realistic methods for characterizing the disentangled latent space to enhance the model's applicability and accuracy.

- Furthermore, when comparing our approach, LGD, to other state-of-the-art disentangled models that primarily utilize local graph information, it is evident that LGD incurs a higher algorithmic complexity. This is largely due to its increased computational demands associated with inferring latent graph information on a global scale. Although this comprehensive approach facilitates a deeper understanding and utilization of graph data, it also results in significantly higher resource consumption. An intriguing direction for future work would be to develop a more efficient algorithm that maintains the informativeness of the global graph insights but with reduced computational overhead. Such advancements could make the LGD approach more practical for large-scale applications and real-time systems.

3.8 Conclusion

We argue that most GNNs have inherited issues due to their entangled representations and/or heavy reliance on the local graph information. Motivated by this problem, we propose a novel framework termed as LGD to learn disentangled node representations both *locally and globally*. LGD is capable of disclosing the hidden node relations pertinent to each latent factor. Specifically, we first present a disentangled latent continuous space with Gaussian mixtures, from which various new graphs w.r.t. different factors can be learned and disentangled. These graphs reflect the latent structure information, i.e. the hidden relations among nodes, overall from different angles. We then utilize them to aggregate and capture the factor-specific information *globally*, which strengthens the *intra-factor consistency*. Moreover, to avoid the mistakenly preserved confounding of the factors, we also promote the *inter-factor diversity* by a novelly designed regularizer along with the latent space modeling. Extensive experiments over synthetic and five real-world data sets well demonstrate the improved classification accuracy and disentangling ability over the state-of-the-arts both quantitatively and qualitatively.

Chapter 4

Generalizing Graph Neural Networks Beyond Homophily with Edge Splitting

In this chapter, we tackle the challenge of *heterophilic linking patterns*. This graph phenomenon is characterized by a notable portion of connected nodes from different classes – a situation that deviates from the homophily assumption in conventional GNNs. Inspired by the insights from addressing *entangled node relationships* in the previous chapter, we posit that there exists task-relevant and irrelevant graph factors influencing node connections within the same and cross different classes. Built upon this hypothesis, we develop the Edge Splitting (ES-) GNN framework that disentangles the input graph as two subgraphs respectively denoting task-relevant and irrelevant relations among nodes. Node features are then aggregated on each subgraph to produce disentangled representations, which essentially decouple the task-relevant and irrelevant information, facilitating the learning of GNNs especially under heterophily. This work has been submitted to the IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI) and is currently under revision.

4.1 Introduction

As a ubiquitous data structure, graph can symbolize complex relationships between entities in different domains. For example, knowledge graphs describe the inter-connections between real-world events, and social networks store the online interactions between users. With the flourishing of deep learning models on graph-structured data, graph neural networks (GNNs) emerge as one of the most powerful techniques in recent years. Owing to their remarkable performance, GNNs have been widely adopted in multiple graph-based learning tasks, such as link prediction, node classification, and recommendation [13, 44, 142, 143].

Modern GNNs are mainly built upon a message passing framework [20], where nodes' representations are learned by aggregating their transformed neighbors iteratively. From the graph signal denoising viewpoint, this mechanism could be seen as a low-pass filter [58, 67, 68, 98] that smooths the signals between adjacent nodes. Several works [4, 22, 54, 63, 64, 68, 144] refer this to smoothness or homophily assumption in GNNs. Notably, they work well on homophilic (assortative) graphs, from which the proximity information of nodes can be utilized to predict their labels [96]. However, real-world networks are typically abstracted from complex systems, and sometimes display heterophilic (disassortative) properties whereby the opposite objects are attracted to each other [145]. For instance, different types of amino acids are mostly interacted in many protein structures [4], and most people in heterosexual dating networks prefer to link with others of the opposite gender. Recent studies [4, 22, 54, 61–64, 72, 144, 146, 147] have shown that the conventional neighborhood aggregation strategy may not only cause the over-smoothing problem [148] but also severely hinder the generalization performance of GNNs beyond homophily.

One reason why current GNNs perform poorly on heterophilic graphs, could be the mismatch between the labeling rules of nodes and their linking mechanism. The former is the target that GNNs are expected to learn for classification tasks, while the latter specifies how messages pass among nodes for attaining this goal. In homophilic

scenarios, both of them are similar in the sense that most nodes are linked because of their commonality which therefore leads to identical labels. In heterophilic scenarios, however, the motivation underlying why two nodes get connected may be ambiguous to the classification task. Let us take the social network within a university as an example, where students from different clubs can be linked usually due to taking the same classes and/or being roommates but not sharing the same hobbies. Namely, the task-relevant and irrelevant (or even harmful) information is typically mixed into node neighborhood under heterophily. However, current methods usually fail to recognize and differentiate these two types of information within nodes' proximity, as illustrated in Figure 4.1. As a consequence, the learned representations are prone to be entangled with false information, leading to non-robustness and sub-optimal performance.

Once the issue of GNNs' learning beyond homophily is identified, a natural question arises: *Can we design a new type of GNNs that is adaptive to both homophilic and heterophilic scenarios?* Well formed designs should be able to recognize the node connections irrelevant to learning tasks, and substantially extract the most correlated information for prediction. However, the assortativity of real-world networks is usually agnostic. Even worse, the features of nodes are typically full of noises, where similarity or dissimilarity between connected ones may not actually reflect their class relations. Existing techniques including [53, 61, 149] usually parameterize graph edges with node similarity or dissimilarity, and cannot well assess the correlation between node connections and the downstream target.

In this work, we propose ES-GNN, an end-to-end graph learning framework that generalizes GNNs on graphs with either homophily or heterophily. Without loss of generality, we make an assumption that two nodes get connected mainly because they share some similar features, which are however unnecessarily just relevant to the learning task. In other words, nodes may be linked due to similar features, either relevant or irrelevant to the task. This implicitly divides the original graph edges into two complementary sets, each of which represents a latent relation between nodes. Thanks to the proximity smoothness, aggregating node features individually on each edge set should

disentangle the task-relevant and irrelevant features. Meanwhile, these disentangled representations potentially reflect node similarity in two aspects (task-relevant and irrelevant). As such, they can be better utilized to split the original graph edges more precisely. Motivated by this, the proposed framework integrates GNNs with an interpretable edge splitting (ES), to jointly partition network topology and disentangle node features.

Technically, we design a residual scoring mechanism, executed within each ES-layer, to distinguish the task-relevant and irrelevant graph edges. The node features are then aggregated separately on these connections to produce disentangled representations, based on which graph edges can be classified more accurately in the next ES-layer. Finally, the task-relevant representations are granted for prediction. Meanwhile, an Irrelevant Consistency Regularization (ICR) is developed to regulate the task-irrelevant representations with the potential label-disagreement between adjacent nodes, for further reducing the classification-harmful information from the final predictive target. To interpret our new algorithm theoretically, generalizing the *standard smoothness assumption* [68], we also conduct some analysis on ES-GNN and establish its connection with a *disentangled graph signal denoising problem*. In summary, the main contributions are four-fold:

- We propose a novel framework called ES-GNN for node classification tasks with one plausible hypothesis, which enables GNNs to go beyond the strong homophily assumption on graphs.
- We theoretically prove that our ES-GNN is equivalent to solving a graph denoising problem with a *disentangled smoothness assumption*, which interprets its good performance on different types of networks.
- Extensive evaluations on 11 benchmark and 1 synthetic datasets show that ES-GNN consistently outperforms the state-of-the-art GNNs on graphs with various homophily levels, and gives the largest error reduction 17.4% on average.
- Importantly, ES-GNN is able to alleviate the over-smoothing problem, and enjoys

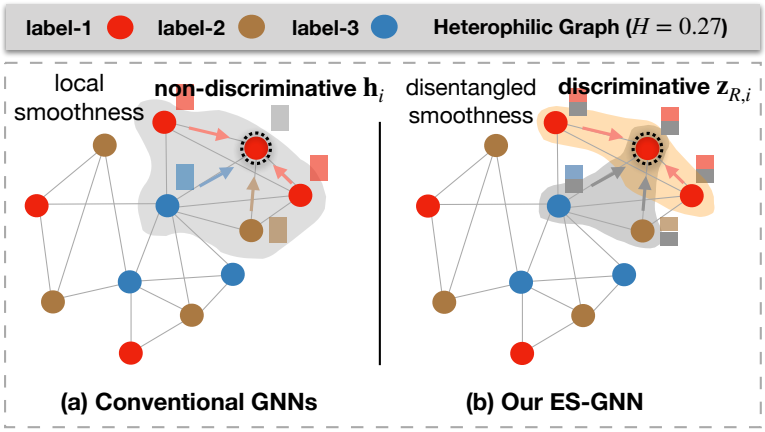


Figure 4.1: A toy example to show differences between conventional GNNs and our ES-GNN in aggregating node features. Standard GNNs with local smoothness tend to produce non-discriminative representations on heterophilic graphs, while our ES-GNN is able to disentangle and exclude the task-harmful features from the final prediction.

remarkable robustness against adversarial graphs. This shows that ES-GNN could still lead to excellent performance even if the *disentangled smoothness assumption* may not hold practically.

4.2 Related Works

Homophily and Heterophily on Graphs. On graphs, homophily and heterophliy (or low homophily) typically refer to the similarity and dissimilarity between adjacent nodes, including but not limited to labels and features. Since this work primarily investigates Graph Neural Networks (GNNs) within the realm of node classification tasks, we focus on the concepts of homophily and heterophily at the level of class labels, dividing real-world graphs into homophilic and heterophilic ones based on different homophily levels. To quantify this level of homophily, researchers have developed various metrics. Among these, edge homophily $\mathcal{H}$ is a widely used metric that calculates the proportion of edges connecting nodes with identical labels, expressed as:

$$\mathcal{H} = \frac{|\{(v_i, v_j) | \mathbf{y}_i = \mathbf{y}_j, \forall (v_i, v_j) \in \mathcal{E}\}|}{|\mathcal{E}|}. \tag{4.1}$$

It ranges from 0 to 1, of which the higher values suggest higher homophily (lower heterophily), and otherwise. Recently, a more nuanced metric such as class homophily $\mathcal{H}_{\text{class}}$ have been proposed in works [150]. This metric take into account potential class imbalance on the graph data, offering a more accurate estimation.

Graph Neural Networks with Heterophily. The central idea of most GNNs is to utilize nodes' proximity information for building their representations for tasks, based on which great effort has been made in developing different variants [3, 35, 53, 57, 58, 66, 151–154], and understanding the nature of GNNs [67, 68, 155–158]. Several works have proved that GNNs essentially behave as a low pass filter that smooths information within node surrounding [58, 96–98]. In line with this view, [67] and [68] further show that a number of GNN models, such as GCN [35], SGC [58], GAT [53], and APPNP [36], can be seen as different optimization solvers to a graph signal denoising problem with a *smoothness assumption* upon connected nodes. All these results indicate that GNNs are mostly tailored for the strong homophily hypothesis on the observed graphs while largely overlooking the important setting of heterophily, where node features and labels vary unsmoothly on graphs. Recent studies [59, 146] also connect this to the over-smoothing problem [148].

To extend GNNs on heterophilic graphs, several works leverage the long-range information beyond nodes' proximity. Geom-GCN [60] extends the standard message passing with geometric aggregation in latent space. H2GCN [4] directly models the higher order neighborhoods for capturing the homophily-dominant information. WRGAT [64] transforms the input graph into a multi-relational graph, for modeling structural information and enhancing the assortativity level. GEN [63] estimates a suitable graph for GNNs' learning with multi-order neighborhood information and Bayesian inference as guide. Another line of work emphasizes the proper utilization of node neighbors. The most common works employ attention mechanism [53, 99], however, they are still imposing smoothness within nodes' neighborhood albeit on the important members only [67, 68, 98]. Compared to that, FAGCN [61] adaptively models both similarities and dissimilarities between adjacent nodes. GPR-GNN [54] introduces a universal polynomial

graph filter, by associating different hop neighbors with learnable weights in both positive and negative signs, so as to extract both low- and high-frequency information.

However, none of them analyzes the motivations why two nodes get connected, nor do they associate them with learning tasks, which is analyzed as one of the keys to generalize GNNs beyond homophily in this work. In contrast, ES-GNN distinguishes graph edges as either relevant (R) or irrelevant (IR) to the task. Such information acts as a guide to disentangle and exclude classification-harmful information from the final predictive target, and thus boosts GNNs' performance under heterophily. Meanwhile, detailed analyses on the limited performance of the existing state-of-the-arts are provided in Section 4.6.2.

Disentangled Representation Learning. Disentangled representation learning is to learn decomposed vector representations which disentangle the explanatory latent variables underlying the observed data and encode them as separate dimensions [89, 159]. Existing efforts concerning that topic are mainly made on computer vision [86, 160–162], while a couple of works recently emerge to explore the potential of disentangled learning in graph-structured domains [1, 3, 95, 163]. For example, DisenGCN [1] employs a neighborhood routing mechanism to iteratively partition node neighborhood into multiple separated parts. NED-VAE [164], a deep unsupervised generative approach for graph disentanglement learning, automatically discovers the independent latent factors in both edges and nodes. SND-VAE [165] emerged as the first disentangled deep generative model for spatial networks, by discovering the independent and dependent latent factors of spatial and networks. FactorGCN [3] factorizes the original graph into multiple subgraphs by clipping edges so as to capture different graph aspects.

We notice that our work shares a similarity with FactorGCN [3]: to learn multiple subgraphs from the original network topology for disentangling features. Nevertheless, there are three main differences. First, FactorGCN could assign one edge to multiple groups, i.e., the factorized subgraphs may share overlapped edges, while our ES-GNN employs an edge splitting to partition the original network topology into two mutually complementary ones satisfying $\mathbf{A}_{\mathrm{R}}+\mathbf{A}_{\mathrm{IR}}=\mathbf{A}$. Second, despite the disentangling property,

FactorGCN merely interprets the inferred subgraphs as different graph aspects without providing any concrete meanings, and the predefined number of latent factors requires to be tuned differently across graphs. Differently, our model adaptively produces two interptable task-relevant and irrelevant topolgies for all kinds of input graphs. Last, FactorGCN models all disentangled parts towards final prediction, while we target at decoupling the task-relevant and task-irrelevant features whereby the classification-harmful information can be excluded from the final predictive target and disentangled in the task-irrelevant parts. Experimental results also validate that our proposed model substantially outperform FactorGCN on all the datasets used in the paper (see Section 6.6).

4.3 Methodology

In this section, we propose an end-to-end graph learning framework, ES-GNN, generalizing Graph Neural Networks (GNNs) to arbitrary graph-structured data with either homophilic or heterophilic properties. An overview of ES-GNN is given in Figure 4.2. The central idea is to integrate GNNs with an interpretable edge splitting (ES) layer that adaptively partitions the network topology as guide to disentangle the task-relevant and irrelevant node features.

4.3.1 Edge Splitting Layer

The goal of this layer is to infer the latent relations underlying adjacent nodes on the observed graph, and distinguish between graph edges which could be relevant or irrelevant to learning tasks. Given a simple graph with an adjacency matrix $\mathbf{A}$ and node feature matrix $\mathbf{X}$, an ES-layer splits the original graph edges into two complementary sets, and thereby produces two partial network topologies with adjacency matrices $\mathbf{A}_{\mathrm{R}}, \mathbf{A}_{\mathrm{IR}} \in \mathbb{R}^{N \times N}$ satisfying $\mathbf{A}_{\mathrm{R}} + \mathbf{A}_{\mathrm{IR}} = \mathbf{A}$. We would expect $\mathbf{A}_{\mathrm{R}}$ storing the most correlated graph edges to the classification task, of which the rest is excluded and disentangled in $\mathbf{A}_{\mathrm{IR}}$. Therefore, analyzing the correlation between node connections and

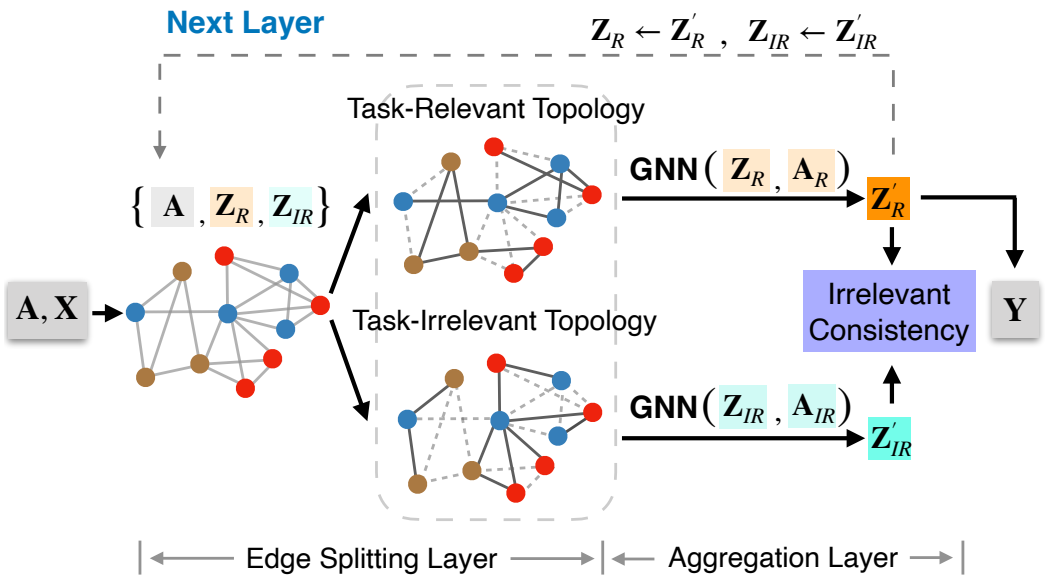


Figure 4.2: Illustration of ES-GNN framework.

learning tasks comes into the first step.

However, existing techniques [53, 61, 149] mainly parameterize graph edges with node similarity or dissimilarity, while failing to explicitly correlate them with the prediction target. Even worse, as the assortativity of real-world networks is usually agnostic and node features are typically full of noises, the captured similarity/dissimilarity may not truly reflect the label-agreement/disagreement between nearby nodes. Consequently, the harmful-similarity between pairwise nodes from different classes could be mistakenly preserved for prediction. To this end, we present one plausible hypothesis below, whereby the explicit correlation between node connections and learning tasks is established automatically.

Hypothesis 1. Two nodes get connected in a graph mainly due to their similarity in some features, which could be either relevant or irrelevant (even harmful) to the learning task.

This hypothesis is assumed without losing generality to both homophilic and heterophilic graphs. For a homophilic scenario, e.g., in citation networks, scientific papers tend to cite or be cited by others from the same area, and both of them usually possess

the common keywords uniquely appearing in their topics. For a heterophilic scenario, students having different interests are likely be connected because of the same classes and/or dormitory they take and/or live in, but neither has direct relation to the clubs they have joined. This inspires us to classify graph edges by measuring the similarity between adjacent nodes in two different aspects, i.e., *a graph edge is more relevant to a classification task if connected nodes are more similar in their task-relevant features, or otherwise.* Our experimental analysis in Section 4.7.3 further provides evidences that even when our Hypothesis 1 may not hold, most adversarial edges (considered as the task-irrelevant ones) can still be recognized though neither types of node similarity exists.

It is worthy mentioning that our hypothesis is not in contradiction to the "opposites attract", which could be intuitively explained by linking due to different but matching attributes. We believe the inherent cause to connection even in "opposites attract" may still be certain commonalities. For example, in heterosexual dating networks, people of the opposite sex are most likely connected because of their similar life values. Although these similarities may be inappropriate (or even harmful) in distinguishing genders, modeling and disentangling them from the final predictive target might be still of great importance.

An ES-layer consists of two channels to respectively extract the task-relevant and irrelevant information from nodes. As only the raw feature matrix $\mathbf{X}$ is provided in the beginning, we will project them into two different subspaces before the first ES-layer:

$$\mathbf{Z}_s^{(0)} = \sigma(\mathbf{W}_s^T \mathbf{X} + \mathbf{b}_s), \tag{4.2}$$

where $\mathbf{W}_s \in \mathbb{R}^{f \times \frac{d}{2}}$ and $\mathbf{b}_s \in \mathbb{R}^{\frac{d}{2}}$ are the learnable parameters in channel $s \in \{\mathrm{R}, \mathrm{IR}\}$, d is the number of node hidden states, and σ is a nonlinear activation function.

Given Hypothesis 1, we adopt a flexible approach for classifying node connections by using continuous edge weights from 0 to 1, reflecting the varying degrees to which edges are task-relevant or irrelevant. Nevertheless, applying metrics to independently determine

$\mathbf{A}_{\mathrm{R}}$ and $\mathbf{A}_{\mathrm{IR}}$ based on node similarity may not fully capture the complex interplay between different channels and could diminish the focus on topological distinctions. To address this, for edges where $\mathbf{A}_{(i,j)} = 1$, we parameterize the difference between $\mathbf{A}_{\mathrm{R}(i,j)}$ and $\mathbf{A}_{\mathrm{IR}(i,j)}$, by solving the linear equation:

$$\begin{cases} \mathbf{A}_{\mathrm{R}(i,j)} - \mathbf{A}_{\mathrm{IR}(i,j)} = \alpha_{i,j} \\ \mathbf{A}_{\mathrm{R}(i,j)} + \mathbf{A}_{\mathrm{IR}(i,j)} = 1 \end{cases}. \tag{4.3}$$

This gives us $\mathbf{A}_{\mathrm{R}(i,j)} = \frac{1+\alpha_{i,j}}{2}$ and $\mathbf{A}_{\mathrm{IR}(i,j)} = \frac{1-\alpha_{i,j}}{2}$ with $-1 \leq \alpha_{i,j} \leq 1$. To effectively quantify the interaction (or relative importance) between the task-relevant and irrelevant aspects of each edge, we propose a residual scoring mechanism:

$$\alpha_{i,j} = \tanh(\mathbf{g}^T \left[\mathbf{Z}^T_{\mathrm{R}[i,:]} \parallel \mathbf{Z}^T_{\mathrm{IR}[i,:]} \parallel \mathbf{Z}^T_{\mathrm{R}[j,:]} \parallel \mathbf{Z}^T_{\mathrm{IR}[j,:]}\right]). \tag{4.4}$$

Here, both of the task-relevant and irrelevant node features are first concatenated and convoluted by learnable $\mathbf{g} \in \mathbb{R}^{2d\times 1}$, and then passed to the tangent activation function to produce a floating value between -1 and 1. Similar learning scheme can be found in works [53, 61, 149]. To further enhance the distinction between $\mathbf{A}_{\mathrm{R}}$ and $\mathbf{A}_{\mathrm{IR}}$, while acknowledging their inherent continuous nature, one can apply techniques, such as softmax with temperature in Eq. (4.5), Gumbel-Softmax [166, 167] in Eq. (4.6), or thresholding in Eq. (4.7). These methods aim to bring their values closer to 0 or 1, thereby strengthening the clarity of task relevance and promoting graph disentanglement.

$$\mathbf{A}'_{s(i,j)} = \frac{\exp(\mathbf{A}_{s(i,j)}/\tau)}{\sum_{\kappa\in\{\mathrm{R,IR}\}} \exp(\mathbf{A}_{\kappa(i,j)}/\tau)} \tag{4.5}$$

$$\mathbf{A}'_{s(i,j)} = \frac{\exp((\log(\mathbf{A}_{s(i,j)}) + \gamma)/\tau)}{\sum_{\kappa\in\{\mathrm{R,IR}\}} \exp((\log(\mathbf{A}_{\kappa(i,j)}) + \gamma)/\tau)} \tag{4.6}$$

$$\mathbf{A}'_{s(i,j)} = \begin{cases} 1 & \mathbf{A}_{s(i,j)} > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

where $s \in \{\mathrm{R}, \mathrm{IR}\}$, τ is a hyper-parameter mediating discreteness degree, and $\gamma \sim Gumbel(0, 1)$ is a Gumbel random variable. However, in this work, we find good results without adding any additional discretization techniques, and will leave this investigation to the future work.

4.3.2 Aggregation Layer

As the split network topologies disclose the partial relations among nodes in different latent spaces, they can be utilized to aggregate information for learning different node aspects. Specifically, we leverage a simple low-pass filter with scaling parameters $\{\epsilon_{\mathrm{R}}, \epsilon_{\mathrm{IR}}\}$ for both task-relevant and irrelevant channels, from the k^{th} to $k+1^{th}$ layer:

$$\mathbf{Z}_s^{(k+1)} = \epsilon_s \mathbf{Z}_s^{(0)} + (1-\epsilon_s)\mathbf{D}_s^{-\frac{1}{2}}\mathbf{A}_s\mathbf{D}_s^{-\frac{1}{2}}\mathbf{Z}_s^{(k)}. \tag{4.8}$$

$s \in \{\mathrm{R}, \mathrm{IR}\}$ denotes the task-relevant or irrelevant channel, and $\mathbf{D}_s$ is the degree matrix associated with the adjacency matrix $\mathbf{A}_s$. Derivation of Eq. (4.8) is detailed in our theoretical analysis. Importantly, by incorporating proximity information in different structural spaces, the task-relevant and irrelevant information can be better disentangled in $\mathbf{Z}_{\mathrm{R}}^{(k+1)}$ and $\mathbf{Z}_{\mathrm{IR}}^{(k+1)}$, based on which the next ES-layer can make a more precise partition on the raw topology.

4.3.3 Irrelevant Consistency Regularization

Stacking ES-layer and aggregation layer iteratively lends itself to disentangling different features of nodes into two distinct representations, denoted by $\mathbf{Z}_{\mathrm{R}}$ and $\mathbf{Z}_{\mathrm{IR}}$. First, $\mathbf{Z}_{\mathrm{R}}$, informed and shaped by $\mathbf{A}_{\mathrm{R}}$, is tuned for prediction, with its development guided by the minimization of the classification loss $\mathcal{L}_{\mathrm{pred}}$. This process not only makes $\mathbf{Z}_{\mathrm{R}}$ predictive of node labels but also implicitly reinforces the task-relevant nature of $\mathbf{A}_{\mathrm{R}}$ via message passing. However, only supervising one channel (R) risks neglecting the meaningfulness of the other (IR), potentially leading to the preservation of erroneous information in predictions. To this end, we introduce a Irrelevant Consistency Regularization (ICR) loss

$\mathcal{L}_{\text{ICR}}$, designed to regulate $\mathbf{Z}_{\text{IR}}$ as the opposite of $\mathbf{Z}_{\text{R}}$, i.e., identifying the classification-harmful information within the observed graph.

The key rationale is to explore the similarities among nodes that are detrimental to classification tasks within $\mathbf{Z}_{\text{IR}}$. Given any node pairs $(v_i, v_j) \in \mathcal{E}$, we would expect $\mathbf{Z}_{\text{IR}[i,:]}$ and $\mathbf{Z}_{\text{IR}[j,:]}$ to be close in the latent space if they possess different labels. Specifically, our ICR can be formulated as:

$$\mathcal{L}_{\text{ICR}} = \sum_{(v_i, v_j) \in \mathcal{E}} (1 - \delta(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)) \|\mathbf{Z}_{\text{IR}[i,:]} - \mathbf{Z}_{\text{IR}[j,:]}\|_2^2, \tag{4.9}$$

where δ is a Kronecker function returning 1 if $\mathbf{y}_i = \mathbf{y}_j$ and 0 otherwise, and $\|\cdot\|_2$ denotes l_2 norm. As such, $\mathbf{Z}_{\text{IR}}$ is constrained with a local consistency between adjacent nodes from different classes. As a benefit, the classification-harmful information between nodes can be further excluded from task-relevant features $\mathbf{Z}_{\text{R}}$, and disentangled in the task-irrelevant ones $\mathbf{Z}_{\text{IR}}$.

Several powerful techniques [149, 168] have been developed to measure the label-agreement between pairwise nodes. In this work, however, we find that using directly the joint probability from model prediction works well, which also offers advantages in low computational complexity as no additional trainable parameters are required.

4.4 Overall Algorithm

4.4.1 Network architecture

The overall pipeline of ES-GNN is detailed in Algorithm 3. Specifically, we adopt ReLU activation function in Eq. (4.2) to first map node features into two different channels, and then pass them with the adjacency matrix to an ES-layer for splitting the raw network topology into two complementary parts. After that, these two partial network topologies are utilized to aggregate information in different structural spaces. Alternatively stacking ES-layer and aggregation layer not only enables more accurate disentanglement but also explores the graph information beyond local neighborhood. Finally, a fully connected

Algorithm 3 Framework of ES-GNN

Input: Nodes set: $\mathcal{V}$, Edge set: $\mathcal{E}$, Adjacency matrix: $\mathbf{A} \in \mathbb{R}^{N\times N}$, Node feature matrix: $\mathbf{X} \in \mathbb{R}^{|V|\times f}$, layer number: K, Scaling parameters: $\{\epsilon_{\mathrm{R}}, \epsilon_{\mathrm{IR}}\}$, Irrelevant consistency coefficient: λ_{ICR}, and ground truth training labels: $\{\mathbf{y}_i \in \mathbb{R}^C | \forall v_i \in \mathcal{V}_{\mathrm{trn}}\}$.

Param: $\mathbf{W}_{\mathrm{R}}, \mathbf{W}_{\mathrm{IR}} \in \mathbb{R}^{f\times d}$, $\mathbf{W}_F \in \mathbb{R}^{d\times C}$, $\mathbf{b}_F \in \mathbb{R}^C$, $\{\mathbf{g}^{(k)} \in \mathbb{R}^{1\times 2d} | k = 0, 1, ..., K-1\}$

// *Project node features into two subspaces.*
for $s \in \{\mathrm{R}, \mathrm{IR}\}$ **do**
  $\mathbf{Z}_s^{(0)} \leftarrow \sigma(\mathbf{W}_s^T \mathbf{X} + \mathbf{b}_s)$.
  $\mathbf{Z}_s^{(0)} \leftarrow \mathrm{Dropout}(\mathbf{Z}_s^{(0)})$ // *Enabled only for training.*
end for
// *Stack Edge Splitting and Aggregation Layers.*
for layer number $k = 0, 1, ..., K-1$ **do**
  Initialize $\mathbf{A}_{\mathrm{R}}, \mathbf{A}_{\mathrm{IR}} \in \mathbb{R}^{N\times N}$ with zeros.
  for $(v_i, v_j) \in \mathcal{E}$ **do**
    $\alpha_{i,j} \leftarrow \tanh(\mathbf{g}^{(k)} \left[\mathbf{Z}_{\mathrm{R}[i,:]}^{(k)} \parallel \mathbf{Z}_{\mathrm{IR}[i,:]}^{(k)} \parallel \mathbf{Z}_{\mathrm{R}[j,:]}^{(k)} \parallel \mathbf{Z}_{\mathrm{IR},[j,:]}^{(k)}\right]^T)$.
    $\alpha_{i,j} \leftarrow \mathrm{Dropout}(\alpha_{i,j})$ // *Enabled only for training.*
    $\mathbf{A}_{\mathrm{R}(i,j)} \leftarrow \frac{1+\alpha_{i,j}}{2}$, $\mathbf{A}_{\mathrm{IR}(i,j)} \leftarrow \frac{1-\alpha_{i,j}}{2}$.
  end for
  $\mathbf{Z}_s^{(k+1)} \leftarrow \epsilon_s \mathbf{Z}_s^{(0)} + (1-\epsilon_s)\mathbf{D}_s^{-\frac{1}{2}}\mathbf{A}_s\mathbf{D}_s^{-\frac{1}{2}}\mathbf{Z}_s^{(k)}$, $\forall s \in \{\mathrm{R}, \mathrm{IR}\}$.
end for
$\hat{\mathbf{y}}_i = \mathrm{softmax}(\mathbf{W}_F^T \mathbf{Z}_{\mathrm{R}[i,:]}^{(K)} + \mathbf{b}_F), \forall v_i \in \mathcal{V}$. // *Prediction.*
$\mathcal{L}_{\mathrm{ICR}} = \sum_{(v_i,v_j)\in\mathcal{E}} (1 - \delta(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)) \|\mathbf{Z}_{\mathrm{IR}[i,:]} - \mathbf{Z}_{\mathrm{IR}[j,:]}\|_2^2$. // *ICR Loss.*
$\mathcal{L}_{\mathrm{pred}} = -\frac{1}{|\mathcal{V}_{\mathrm{trn}}|} \sum_{i\in\mathcal{V}_{\mathrm{trn}}} \mathbf{y}_i^T \log(\hat{\mathbf{y}}_i)$.
Minimize $\mathcal{L}_{\mathrm{pred}} + \lambda_{\mathrm{ICR}}\mathcal{L}_{\mathrm{ICR}}$.

layer is appended to project the learned representations into class space $\mathbb{R}^C$. We integrate $\mathcal{L}_{\mathrm{ICR}}$ into the optimization process with a irrelevant consistency coefficient λ_{ICR} to have final objective function below, where $\mathcal{L}_{\mathrm{pred}} = -\frac{1}{|\mathcal{V}_{\mathrm{trn}}|} \sum_{v_i\in\mathcal{V}_{\mathrm{trn}}} \mathbf{y}_i^T \log(\hat{\mathbf{y}}_i)$.

$$\mathcal{L} = \mathcal{L}_{\mathrm{pred}} + \lambda_{\mathrm{ICR}}\mathcal{L}_{\mathrm{ICR}}. \tag{4.10}$$

It is noted that the method ES-GNN employs in Eq. (4.4) for learning edge weights diverges from the L_2 space metrics used in our ICR loss. While parameterizing edges with node similarity in L_2 space seems straightforward, this method is only independently feasible for task-relevant and irrelevant channels. Such an approach may not fully capture the intricate interactions across different channels, possibly reducing the focus

Table 4.1: Time complexity of the comparison models with one hidden layer as an example. N_e denotes the number of graph aspects assumed in FactorGCN [3], D_{max} represents the maximum node degree, and $|\mathcal{E}_2|$ is the total number of neighbors in the second hop of nodes. Other symbols are earlier defined in this thesis.

Models	**Complexity**
GCN [35]	$\mathcal{O}((f+C)\|\mathcal{E}\|d)$
GAT [53]	$\mathcal{O}(((2+f)N+(4+C)\|\mathcal{E}\|)d)$
FactorGCN [3]	$\mathcal{O}(N_eN+(Nf+(3+C)\|\mathcal{E}\|)d)$
H2GCN [4]	$\mathcal{O}(fd+\|\mathcal{E}\|D_{max}+(\|\mathcal{E}\|+\|\mathcal{E}_2\|)d)$
FAGCN [61]	$\mathcal{O}(((1+C+f)N+\|\mathcal{E}\|)d)$
GPR-GNN [54]	$\mathcal{O}((fN+\|\mathcal{E}\|C)d)$
ES-GNN (Ours)	$\mathcal{O}(((1+C+f)N+\|\mathcal{E}\|)d)$

on topological distinctions. Alternatively, our strategy employs a flexible attention mechanism, unlike direct metric computation, allowing for the nuanced learning of weight residuals between different channels. Importantly, the use of learnable $\mathbf{g}$ in Eq. (4.4) lends our method a universal fitting capability, enabling it to adapt and bridge potential inconsistencies between different framework components. This flexibility ensures that our model effectively integrates and responds to the diverse dynamics within the graph structure, maintaining our focus on graph disentanglement.

4.4.2 Computational Complexity

Finally, we also report in Table 4.1 the complexity of the proposed ES-GNN method in comparison with the state-of-the-arts which will be evaluated in the experimental section. Clearly, our model displays the same complexity to FAGCN [61] while being slightly overhead compared to GPR-GNN [54]. Here, we omit the related works, GEN [63] and WRGAT [64], as their complexity is obviously higher than others by involving reconstructing the whole graph.

4.5 Theoretical Analysis

In this section, we investigate two important problems: (1) what limits the generalization power of the conventional GNNs on graphs beyond homophily, and (2) how the proposed ES-GNN breaks this limit and performs well on different types of networks. We will answer these questions by first analyzing the typical GNNs as graph signal denoising from a more generalized viewpoint, and then impose our Hypothesis 1 to derive ES-GNN.

4.5.1 Potential Flaws of Conventional GNNs

Recent studies [67, 68] have proved that most GNNs can be regarded as solving a graph signal denoising problem:

$$\arg\min_{\mathbf{Z}} \quad \|\mathbf{Z}-\mathbf{X}\|_2^2 + \xi \cdot tr(\mathbf{Z}^T\mathbf{L}\mathbf{Z}), \tag{4.11}$$

where $\mathbf{X} \in \mathbb{R}^{N\times f}$ is the input signal, $\mathbf{L} = \mathbf{D} - \mathbf{A} \in \mathbb{R}^{N\times N}$ is the graph laplacian matrix, and ξ is a constant coefficient. The first term guides $\mathbf{Z}$ to be close to $\mathbf{X}$, while the second term $tr(\mathbf{Z}^T\mathbf{L}\mathbf{Z})$ is the laplacian regularization, which enforces smoothness between connected nodes. One fundamental assumption made here is that similar nodes should have a higher tendency to connect each other, and we refer it as *standard smoothness assumption* on graphs. However, real-world networks typically exhibit diverse linking patterns of both assortativity and disassortativity. Constraining smoothness on each node pair is prone to mistakenly preserve both of the task-relevant and irrelevant (or even harmful) information for prediction. Given that, we divide the original graph into two subgraphs with the same nodes sets but complementary edge sets, i.e., $\mathcal{E} = \mathcal{E}_{\mathrm{R}} \cup \mathcal{E}_{\mathrm{IR}}$, and reformulate Eq. (4.11) as:

$$\arg\min_{\mathbf{Z}} \quad \|\mathbf{Z}-\mathbf{X}\|_2^2 + \xi \cdot tr(\mathbf{Z}^T\mathbf{L}_{\mathrm{R}}\mathbf{Z}) + \xi \cdot tr(\mathbf{Z}^T\mathbf{L}_{\mathrm{IR}}\mathbf{Z}). \tag{4.12}$$

In this formulation, we define $\mathbf{L}_{\mathrm{R}} = \mathbf{D}_{\mathrm{R}} - \mathbf{A}_{\mathrm{R}}$ and $\mathbf{L}_{\mathrm{IR}} = \mathbf{D}_{\mathrm{IR}} - \mathbf{A}_{\mathrm{IR}}$, where $\mathbf{A}_{\mathrm{R}}$ and $\mathbf{A}_{\mathrm{IR}}$ represent the adjacency matrices for task-relevant and task-irrelevant node relations,

captured in $\mathcal{E}_{\mathrm{R}}$ and $\mathcal{E}_{\mathrm{IR}}$, respectively. The operation $tr(\mathbf{Z}^T\mathbf{L}_{\mathrm{R}}\mathbf{Z}) = \sum_{(v_i,v_j)\in\mathcal{E}_{\mathrm{R}}} \|\mathbf{Z}_i - \mathbf{Z}_j\|_2^2$ emphasizes enhancing similarity among adjacent nodes in $\mathbf{A}_{\mathrm{R}}$, which is advantageous for retaining only information relevant to the task. Conversely, smoothing over node pairs in $\mathbf{A}_{\mathrm{IR}}$, indicated by $tr(\mathbf{Z}^T\mathbf{L}_{\mathrm{IR}}\mathbf{Z}) = \sum_{(v_i,v_j)\in\mathcal{E}_{\mathrm{IR}}} \|\mathbf{Z}_i - \mathbf{Z}_j\|_2^2$, might maintain similarities detrimental to classification, potentially limiting the predictive performance of GNNs.

4.5.2 Disentangled Smoothness Assumption in ES-GNN

Our Hypothesis 1 suggests that the original graph topology can be partitioned into two complementary ones, wherein connected nodes displays high similarity with either task-relevant or irrelevant features only. We further interpret this result as *disentangled smoothness assumption*, based on which the conventional graph signal denoising problem in Eq. (4.11) can be generalized as:

$$\begin{aligned} \underset{\mathbf{Z}_{\mathrm{R}},\mathbf{Z}_{\mathrm{IR}}}{\arg\min} \quad & \|\mathbf{Z}_{\mathrm{R}} - \mathbf{X}_{\mathrm{R}}\|_2^2 + \|\mathbf{Z}_{\mathrm{IR}} - \mathbf{X}_{\mathrm{IR}}\|_2^2 + \xi \cdot tr(\mathbf{Z}_{\mathrm{R}}^T\mathbf{L}_{\mathrm{R}}\mathbf{Z}_{\mathrm{R}}) + \xi \cdot tr(\mathbf{Z}_{\mathrm{IR}}^T\mathbf{L}_{\mathrm{IR}}\mathbf{Z}_{\mathrm{IR}}) \\ \text{where} \quad & \mathbf{L}_{\mathrm{R}} = \mathbf{D}_{\mathrm{R}} - \mathbf{A}_{\mathrm{R}}, \mathbf{L}_{\mathrm{IR}} = \mathbf{D}_{\mathrm{IR}} - \mathbf{A}_{\mathrm{IR}} \\ \text{s.t.} \quad & \mathbf{A}_{\mathrm{R}} + \mathbf{A}_{\mathrm{IR}} = \mathbf{A}, \quad \forall \mathbf{A}_{\mathrm{R}(i,j)}, \mathbf{A}_{\mathrm{IR}(i,j)} \in [0,1]. \end{aligned} \tag{4.13}$$

Here, $\mathbf{A}_{\mathrm{R}(i,j)}$ and $\mathbf{A}_{\mathrm{IR}(i,j)}$ measure the degree to which the node connection (v_i, v_j) are relevant and irrelevant to the learning task, respectively. We further name this optimization as *disentangled graph denoising problem*, and finally derive the following theorem:

Theorem 1. The proposed ES-GNN is equivalent to the solution of the disentangled graph denoising problem in Eq. (4.13).

Proof. Let $\mathbf{X}_{\mathrm{R}} \in \mathbb{R}^{\frac{d}{2}}$ and $\mathbf{X}_{\mathrm{IR}} \in \mathbb{R}^{\frac{d}{2}}$ be the results of mapping $\mathbf{X}$ into different channels in Eq. (4.2), i.e., $\mathbf{X}_{\mathrm{R}} = \mathbf{Z}_{\mathrm{R}}^{(0)}$ and $\mathbf{X}_{\mathrm{IR}} = \mathbf{Z}_{\mathrm{IR}}^{(0)}$. Hypothesis 1 motivates us to define $\mathbf{A}_{\mathrm{R}(i,j)}$ and $\mathbf{A}_{\mathrm{IR}(i,j)}$ as node similarity in two aspects. Combining above constraints, we have a

linear system in case of $\mathbf{A}_{(i,j)} = 1$:

$$\begin{cases} \mathbf{A}_{\mathrm{R}(i,j)} + \mathbf{A}_{\mathrm{IR}(i,j)} = 1 \\ \mathbf{A}_{\mathrm{R}(i,j)} - \mathbf{A}_{\mathrm{IR}(i,j)} = \phi_{\mathrm{res}}(\mathbf{Z}_{\mathrm{R}[i,:]}, \mathbf{Z}_{\mathrm{IR}[i,:]}, \mathbf{Z}_{\mathrm{R}[j,:]}, \mathbf{Z}_{\mathrm{IR}[j,:]}) \end{cases}, \tag{4.14}$$

where $\phi_{\mathrm{res}}(\cdot)$ outputs the residual between $\mathbf{A}_{\mathrm{R}(i,j)}$ and $\mathbf{A}_{\mathrm{IR}(i,j)}$ considering both task-relevant and irrelevant node information, and can be formulated with our residual scoring mechanism in Eq. (4.4). Solving above equations, we can express both $\mathbf{A}_{\mathrm{R}}$ and $\mathbf{A}_{\mathrm{IR}}$ in terms of $\mathbf{Z}_{\mathrm{R}}$ and $\mathbf{Z}_{\mathrm{IR}}$, i.e.,

$$\mathbf{A}_{\mathrm{R}(i,j)} = \frac{1 + \mathbf{\Omega}_{(i,j)}}{2}, \ \mathbf{A}_{\mathrm{IR}(i,j)} = \frac{1 - \mathbf{\Omega}_{(i,j)}}{2} \tag{4.15}$$

where $\mathbf{\Omega}_{(i,j)} = \phi_{\mathrm{res}}(\mathbf{Z}_{\mathrm{R}[i,:]}, \mathbf{Z}_{\mathrm{IR}[i,:]}, \mathbf{Z}_{\mathrm{R}[j,:]}, \mathbf{Z}_{\mathrm{IR}[j,:]})$. So far, the optimization problem in Eq. (4.13) is only made up of variables $\mathbf{X}_{\mathrm{R}}$, $\mathbf{X}_{\mathrm{IR}}$, $\mathbf{Z}_{\mathrm{R}}$, and $\mathbf{Z}_{\mathrm{IR}}$. Directly solving it is still however not easy, as the mixing variables of $\mathbf{Z}_{\mathrm{R}}$ and $\mathbf{Z}_{\mathrm{IR}}$, and the introduced non-linear operator in $\phi_{\mathrm{res}(\cdot)}$ result in a complicated differentiation process.

Instead, we can approach this problem by decoupling the learning of $\mathbf{A}_{\mathrm{R}}, \mathbf{A}_{\mathrm{IR}}$ from the optimization target, and employ an alternative learning between stages. Suppose we have attained the task-relevant and irrelevant node features in the k^{th} round, i.e., $\mathbf{Z}_{\mathrm{R}}^{(k)}$ and $\mathbf{Z}_{\mathrm{IR}}^{(k)}$. In the first stage, we can compute $\mathbf{A}_{\mathrm{R}(i,j)}^{(k+1)}$ and $\mathbf{A}_{\mathrm{IR}(i,j)}^{(k+1)}$ using $\{\mathbf{Z}_{\mathrm{R}[i,:]}^{(k)}, \mathbf{Z}_{\mathrm{IR}[i,:]}^{(k)}, \mathbf{Z}_{\mathrm{R},[j,:]}^{(k)}, \mathbf{Z}_{\mathrm{IR}[j,:]}^{(k)}\}$ with Eq. (4.15) and Eq. (4.15), which in fact turns out to be our ES-layer in Section 4.3.1.

In the second stage, injecting the computed values of $\mathbf{A}_{\mathrm{R}(i,j)}^{(k+1)}$ and $\mathbf{A}_{\mathrm{IR}(i,j)}^{(k+1)}$ relaxes the mixture of variables $\mathbf{Z}_{\mathrm{R}}$ and $\mathbf{Z}_{\mathrm{IR}}$, and the original optimization problem can then be disentangled into two independent targets (as all four penalized terms are positive):

$$\arg\min_{\mathbf{Z}_{\mathrm{R}}^*} \|\mathbf{Z}_{\mathrm{R}}^* - \mathbf{Z}_{\mathrm{R}}^{(0)}\|_2^2 + \xi \cdot tr(\mathbf{Z}_{\mathrm{R}}^{*T} \mathbf{L}_{\mathrm{R}}^{(k)} \mathbf{Z}_{\mathrm{R}}^*) \tag{4.16}$$

$$\arg\min_{\mathbf{Z}_{\mathrm{IR}}^*} \|\mathbf{Z}_{\mathrm{IR}}^* - \mathbf{Z}_{\mathrm{IR}}^{(0)}\|_2^2 + \xi \cdot tr(\mathbf{Z}_{\mathrm{IR}}^{*T} \mathbf{L}_{\mathrm{IR}}^{(k)} \mathbf{Z}_{\mathrm{IR}}^*) \tag{4.17}$$

where $\mathbf{L}_{\mathrm{R}}^{(k)} = \mathbf{D}_{\mathrm{R}}^{(k)} - \mathbf{A}_{\mathrm{R}}^{(k)}$ and $\mathbf{L}_{\mathrm{IR}}^{(k)} = \mathbf{D}_{\mathrm{IR}}^{(k)} - \mathbf{A}_{\mathrm{IR}}^{(k)}$ are fixed values. Lemma 2, on the R channel as an example, further shows that our aggregation layer, on the task-relevant and irrelevant topologies, in Section 4.3.2 is approximately solving these two optimization problems in Eq. (4.16) and Eq. (4.17).

Therefore, stacking ES- and aggregation layers iteratively is equivalent to the above alternative learning for solving the *disentangled graph denoising problem* in Eq. (4.13) with $\mathbf{X}_{\mathrm{R}} = \mathbf{Z}_{\mathrm{R}}^{(0)}$ and $\mathbf{X}_{\mathrm{IR}} = \mathbf{Z}_{\mathrm{IR}}^{(0)}$. Finally, given $\mathbf{Z}_{\mathrm{R}}^{(K)}$ and $\mathbf{Z}_{\mathrm{IR}}^{(K)}$, we minimize the prediction loss $\mathcal{L}_{\mathrm{pred}}$ and the Irrelevant Consistency Regularization $\mathcal{L}_{\mathrm{ICR}}$ in Eq. (4.10) with Adam [169] algorithm, which imposes concrete meanings on different channels, and simultaneously ensures the convergence of our described alternative learning. □

Lemma 2. When adopting the normalized laplacian matrix $\mathbf{L}_{\mathrm{R}} = \mathbf{I} - \mathbf{D}_{\mathrm{R}}^{-\frac{1}{2}}\mathbf{A}_{\mathrm{R}}\mathbf{D}_{\mathrm{R}}^{-\frac{1}{2}}$, the feature aggregation operator in Eq. (4.8) with channel $s = \mathrm{R}$ can be regarded as solving Eq. (4.16) using iterative gradient descent with stepsize $\beta = \frac{1}{2+2\xi}$ and $\xi = \frac{1}{\epsilon_{\mathrm{R}}} - 1$.

Proof. We take iterative gradient descent with the stepsize β to solve the denoising problem in Eq. (4.16) (referred as $\mathcal{L}_{\mathrm{R}}$) as follows:

$$\mathbf{Z}_{\mathrm{R}}^{(k+1)} = \mathbf{Z}_{\mathrm{R}}^{(k)} - \beta \cdot \frac{\partial \mathcal{L}_{\mathrm{R}}}{\partial \mathbf{Z}_{\mathrm{R}}^{*}}|_{\mathbf{Z}_{\mathrm{R}}^{*}=\mathbf{Z}_{\mathrm{R}}^{(k)}} \tag{4.18}$$

$$= 2\beta\mathbf{Z}_{\mathrm{R}}^{(0)} + 2\beta\xi(\mathbf{D}_{\mathrm{R}}^{-\frac{1}{2}}\mathbf{A}_{\mathrm{R}}\mathbf{D}_{\mathrm{R}}^{-\frac{1}{2}})\mathbf{Z}_{\mathrm{R}}^{(k)} + (1 - 2\beta - 2\beta\xi)\mathbf{Z}_{\mathrm{R}}^{(k)}. \tag{4.19}$$

Setting β as $\frac{1}{2+2\xi}$ gives us:

$$\mathbf{Z}_{\mathrm{R}}^{(k+1)} = \frac{1}{1+\xi}\mathbf{Z}_{\mathrm{R}}^{(0)} + \frac{\xi}{1+\xi}(\mathbf{D}_{\mathrm{R}}^{-\frac{1}{2}}\mathbf{A}_{\mathrm{R}}\mathbf{D}_{\mathrm{R}}^{-\frac{1}{2}})\mathbf{Z}_{\mathrm{R}}^{(k)}, \tag{4.20}$$

which is equivalent to Eq. (4.8) while choosing $\xi = \frac{1}{\epsilon_{\mathrm{R}}} - 1$, i.e.,

$$\mathbf{Z}_{\mathrm{R}}^{(k+1)} = \epsilon_{\mathrm{R}}\mathbf{Z}_{\mathrm{R}}^{(0)} + (1 - \epsilon_{\mathrm{R}})(\mathbf{D}_{\mathrm{R}}^{-\frac{1}{2}}\mathbf{A}_{\mathrm{R}}\mathbf{D}_{\mathrm{R}}^{-\frac{1}{2}})\mathbf{Z}_{\mathrm{R}}^{(k)}. \tag{4.21}$$

□

As the possible classification-harmful similarity between nodes (hidden in $\mathbf{A}_{\mathrm{IR}}$) can

be excluded from $\mathbf{Z}_{\mathrm{R}}$ and disentangled in $\mathbf{Z}_{\mathrm{IR}}$ while optimizing Eq. (4.13), our ES-GNN presents a universal approach that theoretically guarantees good performance on different types of networks.

4.5.3 Aligning Disentangled and Conventional Problems

It is noted that Eq. (4.11) can be interpreted as a special case of Eq. (4.13) in specific graph scenarios. This situation arises in graphs where only edges connecting nodes with identical labels exist, indicating that the similarity between adjacent nodes should be beneficial in predicting their shared label. In such cases, task-irrelevant edges may not exist, as all connections inherently support the task. Consequently, in this scenario, the objective term involving $\mathbf{L}_{\mathrm{IR}}$ in Eq. (4.13) becomes redundant, amounting to zero, and the need for disentangling $\mathbf{Z}_{\mathrm{R}}$ and $\mathbf{Z}_{\mathrm{IR}}$ is obviated. This leads to the simplification of Eq. (4.13) into the conventional graph denoising problem Eq. (4.11), conforming to the standard smoothness assumption across the entire graph.

In practice, completely smooth graphs devoid of edges linking nodes with different labels are rare. Nevertheless, for homophilic graphs where most edges connect nodes from the same class, such as in the citation network Cora with homophily ratio 0.810, Eq. (4.11) serves as a close approximation of Eq. (4.13). This approximation holds as the term involving $\mathbf{L}_{\mathrm{IR}}$ in Eq. (4.13) becomes negligible and the inherently classification-harmful information in the graph is almost non-existent. Empirical evidence from Figure 4.5 in our study reinforces this understanding, showing that on homophilic graphs like Cora, the majority of informative content is retained in the task-relevant channel, underscoring the minimal presence of classification-harmful information.

4.6 Experiments

We empirically evaluate our ES-GNN for node classification using both synthetic and real-world datasets in this section.

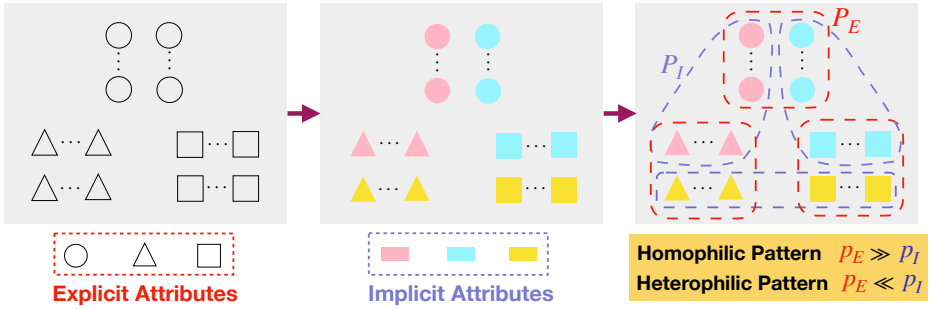


Figure 4.3: Synthetic graphs with varying levels of homophily. Node shape and color refer to the explicit and implicit attributes, respectively. Nodes sharing the same shape (or color) are connected with a probability of P_{E} (or P_{I}) and are classified into three categories only based on their different shapes. In this context, "shape" attributes represent task-relevant features, whereas "color" attributes denote task-irrelevant ones. It can be intuitively observed that adequate disentanglement of these attributes is crucial for classification tasks; otherwise, model prediction will inevitably suffer, as misled by the task-irrelevant "color" information.

4.6.1 Experimental Setup

Synthetic Data. To investigate the behavior of GNNs on graphs with arbitrary levels of homophily and heterophily, we consider the contextual stochastic block model (CSBM) [170, 171] to construct synthetic graphs with our Hypothesis 1 as guide. The central idea is to define links among nodes under two conditions independently, of which only one is correlated with the classification task. We consider 1,200 nodes, 3 equal-size classes, and 500 node features made up of both explicit and implicit attributes. The explicit attributes determine the label assignment, while implicit ones model dependency across different classes. Figure 4.3 further illustrates their allocation to nodes with "shape" and "color" as an example. Notably, all these attributes in six types (three explicit and three implicit ones) are randomly sampled from different Gaussian distributions, each pair of them are combined via element-wise addition to attain the final node features. For instance, the features of a node (from i-th class) with explicit attribute-i and implicit attribute-j are defined as the addition of two random vectors respectively sampled from $\mathcal{N}(\boldsymbol{\mu}_{\mathrm{E},i}, \boldsymbol{\sigma}_{\mathrm{E},i})$ and $\mathcal{N}(\boldsymbol{\mu}_{\mathrm{I},j}, \boldsymbol{\sigma}_{\mathrm{I},j})$, where $\boldsymbol{\mu}_{\mathrm{E},i}, \boldsymbol{\mu}_{\mathrm{I},j} \in \mathbb{R}^{f_{\mathrm{syn}}}$ are means, $\boldsymbol{\sigma}_{\mathrm{E},i}, \boldsymbol{\sigma}_{\mathrm{I},j} \in \mathbb{R}^{f_{\mathrm{syn}} \times f_{\mathrm{syn}}}$ are the associated covariance matrixes, and $f_{\mathrm{syn}} = 500$ is the feature dimensions.

Table 4.2: Parameters for synthesizing graphs with varying homophily ratios $\mathcal{H}_{\text{syn}}$.

$\mathcal{H}_{\text{syn}}$	**0.0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1.0**
P_{E}	0.02	0.06	0.1	0.2	0.4	0.4	0.6	0.7	0.8	0.9	0.96
P_{I}	0.72	0.81	0.6	0.7	0.9	0.6	0.6	0.45	0.3	0.15	0.045
ω	0.1	0.084	0.1	0.075	0.05	0.062	0.05	0.05	0.05	0.05	0.051

Then, we connect nodes with probability P_{E} if they are from the same class (the task-relevant condition), with probability P_{I} if they share different labels but posses implicit attributes from the same distribution (the task-irrelevant condition). For all other cases, we connect nodes with probability q in a small value, 1e−5 in this work for ensuring a connected graph. Since no class-imbalance problem exists here, the homophily ratios of our generated graphs are measured with Eq. (4.1). Intuitively, we could anticipate heterophilic connecting pattern when setting $P_{\text{E}} \ll P_{\text{I}}$, and strong homophily otherwise. Quantitatively, the relationship between the homophily ratio $\mathcal{H}_{\text{syn}}$ and parameters P_{E}, P_{I} can be derived with the simple knowledge on combinatorics and statistics while omitting the small value of q:

$$\mathcal{H}_{\text{syn}}(P_{\text{E}}, P_{\text{I}}) = \frac{3(n-3)}{3(n-3) + 2n\frac{P_{\text{I}}}{P_{\text{E}}}}, \tag{4.22}$$

with n being the total number of nodes. Clearly, we have $\mathcal{H}_{\text{syn}} \to 0$ while $P_{\text{I}} \gg P_{\text{E}}$, and $\mathcal{H}_{\text{syn}} \to 1$ while $P_{\text{I}} \ll P_{\text{E}}$. To avoid possible computational overhead, we also need to control the average node degree of our synthetic graphs. Similarly, we can approximately derive it as the function of P_{I} and P_{E}:

$$\mathcal{T}(P_{\text{E}}, P_{\text{I}}) = \frac{n-3}{3}P_{\text{E}} + \frac{4n}{9}P_{\text{I}}. \tag{4.23}$$

From Eq. (4.22) and Eq. (4.23), we have that $\mathcal{H}_{\text{syn}}(\cdot)$ is a function of the fraction between P_{E} and P_{I} with fixed n, and $\mathcal{T}(\cdot)$ is linearly correlated with P_{E} and P_{I}. As such, given fixed P_{E} and P_{I} attaining certain $\mathcal{H}_{\text{syn}}$, we can almost attain the average node degree in any values with a scaling parameter ω, i.e., average degree $= \omega \cdot \mathcal{T}(P_{\text{E}}, P_{\text{I}}) = \mathcal{T}(\omega \cdot P_{\text{E}}, \omega \cdot P_{\text{I}})$

Table 4.3: Statistics of real-world datasets, where both $\mathcal{H}$ and $\mathcal{H}_{\text{class}}$ (considering class-imbalance problem) measure graph homophily ratio from 0 to 1.

Dataset	# Nodes	# Edges	# Features	# Classes	$\mathcal{H}$	$\mathcal{H}_{\text{class}}$
Squirrel	5,201	217,073	2,089	5	0.22	0.03
Chameleon	2,227	36,101	2,325	5	0.23	0.06
Wisconsin	251	499	1,703	5	0.21	0.09
Cornell	183	295	1,703	5	0.30	0.05
Texas	183	309	1,703	5	0.11	0.00
Twitch-DE	9,498	153,138	2,545	2	0.63	0.14
Actor	7,600	33,544	931	5	0.22	0.01
Cora	2,708	5,429	1,433	7	0.81	0.77
Citeseer	3,327	4,732	3,703	6	0.74	0.63
Pubmed	19,717	44,338	500	3	0.80	0.66
Polblogs	1,222	16,714	/	2	0.91	0.81

without changing $\mathcal{H}_{\text{syn}}$. In this work, we tune all these parameters such that the average degree is around 20, and list the tested values in Table 4.2.

Real-World Datasets. We consider 11 widely used benchmark datasets including both seven heterophilc graphs, i.e., Chameleon, Squirrel [172], Wisconsin, Cornell, Texas [60] (webpage networks), Actor [173] (co-occurrence network), and Twitch-DE [5, 172] (social network), as well as four homophilic graphs including Cora, Citeseer, Pubmed [140] (citation networks), and Polblogs [174, 175] (community network) with statistics shown in Table 6.1. For Polblogs dataset, since node features are not provided, we use the rows of the adjacency matrix. In certain compact sections, we use four-letter abbreviations for dataset names.

Data Splitting. For heterophilic graphs and our synthetic graphs, we divides each dataset into 60%/20%/20% corresponding to training/validation/testing to follow [4, 54, 60]. For homophilic graphs, we adopt the popular sparse splitting [35, 53, 58], i.e., 20 nodes per class, 500 nodes, and 1,000 nodes to train, validate, and test models. For each dataset, 10 random splits are created for evaluation.

Baselines. We compare our ES-GNN with 9 baselines including the state-of-the-art GNNs: 1) GCN [35] adopts Chebyshev expansion to approximate the graph laplacian efficiently; 2) SGC [58] simplifies GCN [35] by removing non-linearity; 3) GAT [53]

employs an attention mechanism to adaptively utilize neighborhood information; 4) FactorGCN [3]; 5) GEN [63]; 6) WRGAT [64]; 7) H2GCN [4]; 8) FAGCN [61]; 9) GPR-GNN [54], of which baselines from 4) to 9) have been briefly introduced in the Section 6.2.

Implementation Details. For all the baselines and our model, we set $d = 64$ as the number of hidden states for fair comparison, and tune the hyper-parameters on the validation split of each dataset using Optuna [141] for 200 trials. With the best hyper-parameters, we train models in 1,000 epochs using the early-stopping strategy with a patience of 100 epochs. We then report the average performance in 10 runs on the test set for each random split. For reproducibility, we provide the searching space of our hyper-parameters: learning rate $\sim$ [1e−2, 1e−1], weight decay $\sim$ [1e−6, 1e−3], dropout $\sim \{0, 0.1, ..., 0.8\}$ with step 0.1, the number of layers $K \sim \{1, 2, ..., 8\}$ with step 1, scaling parameter $\epsilon_{\text{R}}, \epsilon_{\text{IR}} \sim \{0.1, 0.2, ..., 1\}$ with step 0.1, and irrelevant consistency coefficient $\lambda_{\text{ICR}} \sim [0, 1]$ for Cora, Citeseer, Pubmed, and Twitch-DE, [5e−8, 5e−6] for Chameleon, Wisconsin, Cornell, and Texas, [5e−5, 5e−3] for Squirrel, and [5e−3, 5e−2] for Actor.

4.6.2 Overall Evaluation

Results on Synthetic Graphs We examine the learning ability of various models on graphs across the homophily or heterophily spectrum. From Figure 4.4, we have the following observations: **1)** Looking through the overall trend, we obtain a "U" pattern on graphs from the lowest to the highest homophily ratios. That suggests GNNs' prediction performance is not monotonically correlated with graph homophily levels in a strict manner. When it comes to the extreme heterophilic scenario, GNNs tend to alternate node features completely between different classes, thereby still making nodes distinguishable w.r.t. their labels, which coincides with the findings in [176]. **2)** Despite the attention mechanism for adaptively utilizing relevant neighborhood information, GAT turns out to be the least robust method to arbitrary graphs. The entangled information in the mixed assortativity and disassortativity provides weak supervision signals for learning the attention weights. FactorGCN employs a graph

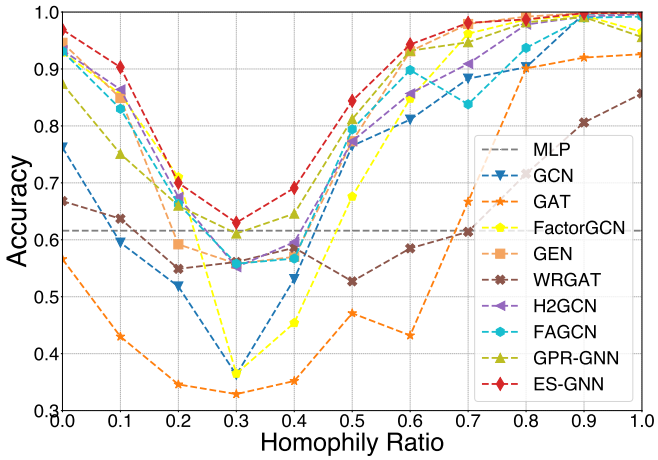


Figure 4.4: Results of different models on synthetic graphs with varied homophily ratios, where ES-GNN constantly outperform all the baselines including conventional GNNs and the state-of-the-arts explicitly designed for heterophilic graphs.

factorization to disentangle different graph aspects but still adopts all of them for prediction without judgement, thereby performing poorly especially on the tough cases of $\mathcal{H}_{\text{syn}} = 0.3, 0.4,$ and 0.5. **3)** Both FAGCN and GPR-GNN model the dissimilarity between nearby nodes to go beyond the smoothness assumption in conventional GNNs, and display some superiority under heterophily. However, the correlation between graph edges and classification tasks is not explicitly defined and emphasized in their designs. In other words, the classification-harmful information still could be preserved in their node dissimilarity. Experimental results also show that these methods are constantly beaten by our disentangled approach. **4)** The proposed ES-GNN consistently outperforms, or matches, others across different graphs with different homophily levels, especially in the hardest case with $\mathcal{H}_{\text{syn}} = 0.3$ where some baselines even perform worse than MLP. This is mainly because our ES-GNN is able to distinguish between task-relevant and irrelevant graph links, and makes prediction with the most correlated features only.

Results on Real-World Graphs Table 5.3 summaries node classification accuracies on real-world datasets in 100 runs with multiple random splits and different model initializations. In general, ES-GNN achieves state-of-the-art performance on all the eleven datasets, and consistently outperforms all the baselines including four popular

Table 4.4: Node classification accuracies (%) over 100 runs. Error Reduction gives the average improvement of our ES-GNN upon the second place models, which are explicitly designed for heterophilic graphs.

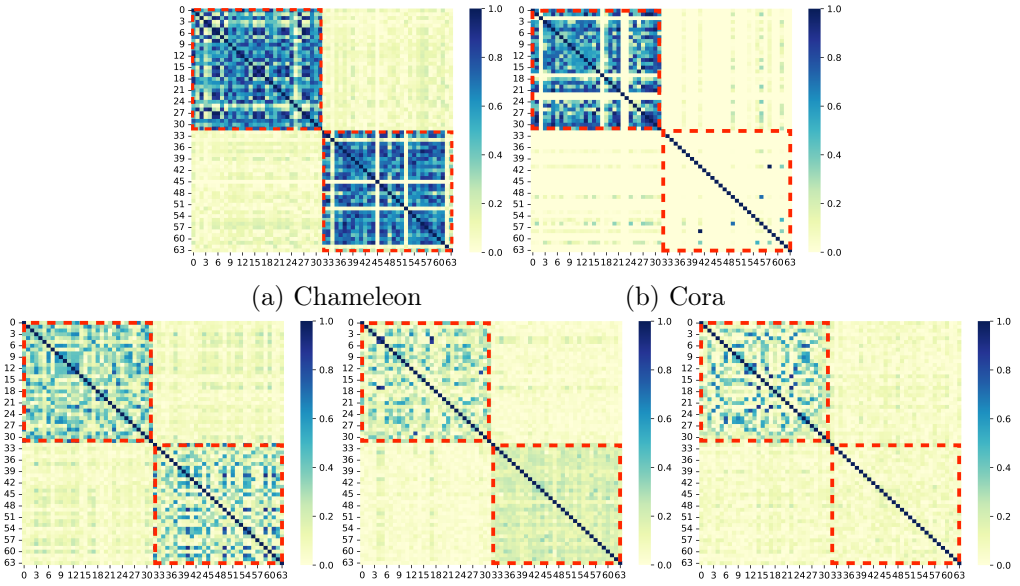
Datasets	**Heterophilic Graphs**							**Homophilic Graphs**			
	Squi.	**Cham.**	**Wisc.**	**Corn.**	**Texas**	**Twit.**	**Actor**	**Cora**	**Cite.**	**Pubm.**	**Polb.**
GCN [35]	55.2±1.5	67.6±2.0	59.5±3.6	52.8±6.0	61.7±3.7	74.0±1.2	31.2±1.3	79.7±1.2	69.5±1.7	78.7±1.6	89.4±0.9
SGC [58]	50.7±1.3	61.9±2.6	53.7±3.9	51.2±0.9	51.4±2.2	73.9±1.3	30.9±0.6	79.1±1.0	69.9±2.0	76.6±1.3	89.0±1.5
GAT [53]	54.8±2.2	67.3±2.2	57.9±4.5	50.4±5.9	55.4±5.9	73.7±1.3	30.5±1.2	82.0±1.1	69.9±1.7	78.6±2.0	87.4±1.1
FactorGCN [3]	56.6±2.4	69.8±2.0	64.2±4.8	50.6±1.8	69.5±6.5	73.1±1.4	29.0±1.4	75.2±1.6	61.6±2.0	72.9±2.3	87.9±1.7
GEN [63]	36.0±4.0	57.6±3.1	83.3±3.6	81.0±3.9	78.3±8.0	74.1±1.4	37.3±1.4	79.8±1.3	69.7±1.6	78.9±1.7	89.6±1.4
WRGAT [64]	39.6±1.4	57.7±1.6	82.9±4.5	79.2±3.5	80.5±6.1	70.0±1.3	38.6±1.1	71.7±1.5	64.1±1.9	73.3±2.1	88.2±1.2
H2GCN [4]	45.1±1.9	62.9±1.9	82.6±4.0	79.6±4.9	79.8±7.3	73.1±1.5	38.4±1.0	81.4±1.4	68.7±2.0	78.0±2.0	89.0±1.0
FAGCN [61]	50.4±2.6	68.9±1.8	82.3±4.4	79.4±5.5	80.3±5.5	74.1±1.4	37.9±1.0	82.6±1.3	70.3±1.6	80.0±1.7	89.3±1.1
GPR-GNN [54]	54.1±1.6	69.6±1.7	82.7±4.1	79.9±5.3	81.7±4.9	74.0±1.6	38.0±1.1	81.5±1.5	69.6±1.7	79.8±1.3	89.5±0.8
ES-GNN (ours)	**62.4**±1.4	**72.3**±2.1	**85.3**±4.6	**82.2**±4.0	**82.3**±5.7	**74.7**±1.1	**38.9**±0.8	**83.0**±1.1	**70.7**±1.7	**80.7**±1.4	**89.7**±0.9
Error Reduction	**17.4%**	**9.0%**	**2.5%**	**2.4%**	**2.2%**	**1.6%**	**0.9%**	**3.6%**	**2.2%**	**2.7%**	**0.6%**

graph neural network models and five recent state-of-the-arts which explicitly considers heterophily on graphs. Specifically, compared to the second place models, our method achieves significant performance gains by 17.4% and 9.0% separately on the heterophilic graphs Squirrel and Chameleon, and we have the relative error reductions of 2.5%, 2.4%, and 2.2% on Wisconsin, Cornell, and Texas, respectively. For Actor and Twitch-DE, ES-GNN wins by an average margin of 1.3%. We notice that FactorGCN surprisingly has relative good performance on Squirrel and Chameleon datasets. This phenomenon can be explained by its ability on separating channels for learning disentangled graph aspects, which further verifies our speculation in Section 4.1, i.e., the different types of information are typically mixed and entangled in the node neighborhood under heterophily. However, FactorGCN dose not continue to perform well on other five heterophilic graphs, mainly because it fails to distinguish between the useful and useless (even harmful) channels, and takes the whole parts for classification. For graphs with strong homophily, ES-GNN maintains competitiveness and exhibits averagely 2.3% superiority upon the state-of-the-art models. We will show our ES-GNN could demonstrate remarkable robustness on homophilic graphs in case of perturbation or noisy links in Section 4.7.3.

4.7 Analysis and Discussion

4.7.1 Correlation Analysis

To better understand our proposed method, we investigate the disentangled features on Chameleon, Cora, and three synthetic graphs as typical examples in Figure 4.5. Clearly, on the strong heterophilic graph Chameleon with $\mathcal{H} = 0.23$, correlation analysis of learned latent features displays two clear block-wise patterns, each of which represents task-relevant or task-irrelevant aspect respectively. In contrast, on the citation network Cora with $\mathcal{H} = 0.81$, the node connections are in line with the classification task, since scientific papers mostly cite or are cited by others in the same research topic. Thus, most information will be retained in the task-relevant topology, while very minor information could be disentangled in the task-irrelevant topology (see Figure 4.5 (b)).



(a) Chameleon (b) Cora

(c) $\mathcal{H}_{\text{syn}} = 0.1$ (d) $\mathcal{H}_{\text{syn}} = 0.5$ (e) $\mathcal{H}_{\text{syn}} = 0.9$

Figure 4.5: Feature correlation analysis. Two distinct patterns (task-relevant and task-irrelevant topologies) can be learned on Chameleon with $\mathcal{H} = 0.23$, while almost all information is retained in the task-relevant channel (0-31) on Cora with $\mathcal{H} = 0.81$. On synthetic graphs in (c), (d), and (e), block-wise pattern in the task-irrelevant channel (32-63) is gradually attenuated with the incremental homophily ratios across 0.1, 0.5, and 0.9. ES-GNN presents one general framework which can be adaptive for both heterophilic and homophilic graphs.

On the other hand, the results on synthetic graphs from Figure 4.5 (c)-(e) display an attenuating trend on the second block-wise pattern with the incremental homophily ratios across $0.1, 0.5,$ and 0.9. This correlation analysis empirically verifies that our ES-GNN successfully disentangles the task-relevant and irrelevant features, and also demonstrates its universal adaptivity on different types of networks.

4.7.2 Edge Analysis

We analyze the split edges from our ES-layer using synthetic graphs as an example in this section. According to Section 4.6.1, the synthetic edges are defined as the task-relevant connections if they link nodes from the same class, and the task-irrelevant ones otherwise. Therefore, we calculate the percentages of heterophilic node connections, which are

Table 4.5: Edge Analysis of our ES-GNN on synthetic graphs with various homophily ratios. Removed Het. gives the percentage (%) of heterophilic node connections excluded from the task-relevant topology and disentangled in the task-irrelevant topology. The last two rows give the corresponding node classification accuracies (%) of ES-GNN and its variant while ablating ES-layer.

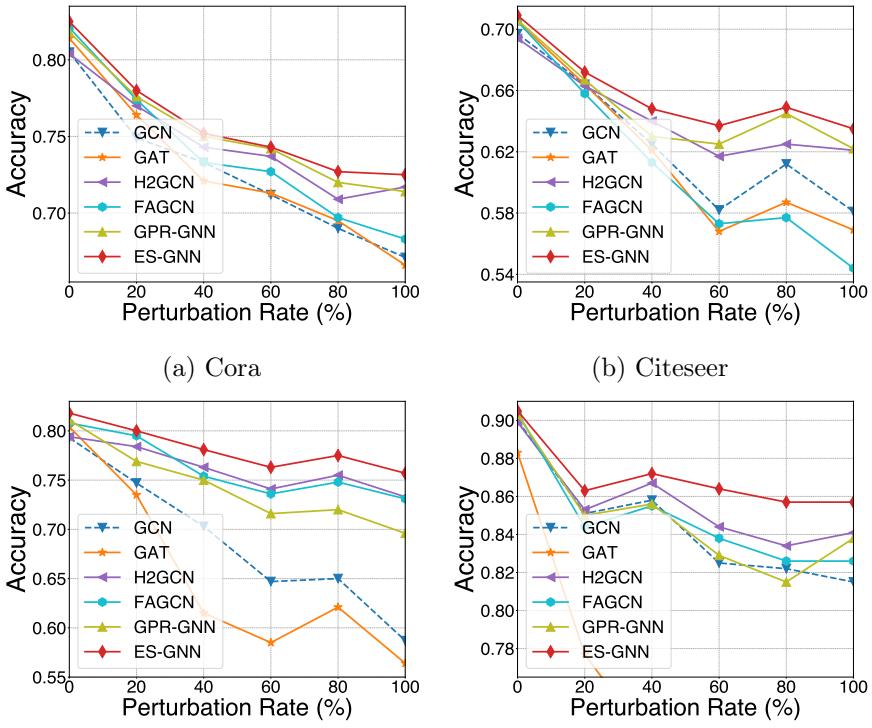
$\mathcal{H}_{\text{syn}}$	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**Avg.**
Removed Het.	41.9	53.2	60.8	70.4	74.2	80.7	86.7	87.8	89.9	**71.7**
ES-GNN	90.0	69.6	62.1	69.6	85.4	93.8	98.3	99.2	100.0	**85.3**
ES-GNN w/o ES	84.6	57.9	53.3	53.8	74.2	81.7	86.3	90.4	96.7	**75.4**

excluded from our task-relevant topology and disentangled in the task-irrelevant one, so as to investigate the discerning ability of ES-GNN between edges in different types. As can be observed in Table 4.5, 71.7% task-irrelevant edges are identified on average across various homophily ratios. On the other hand, we also report the classification accuracies of ES-GNN and its variant while ablating ES-layer, from which approximately 10% degradation can be observed. All of these strongly validate the effectiveness of our ES-layer and reasonably interprets the good performance of ES-GNN.

4.7.3 Robustness Analysis

By splitting the original graph edge set into task-relevant and task-irrelevant subsets, our proposed ES-GNN enjoys strong robustness particularly on homophilic graphs, since perturbed or noisy aspects of nodes could be purified from the task-relevant topology and disentangled in the task-irrelevant topology. To examine this, we randomly inject fake edges into graphs with perturbed rates from 0% to 100% with a step size of 20%. Adversarially perturbed examples are generated from graphs with strong homophily, such as Cora, Citseer, Pubmed, and Polblogs. As shown in Figure 4.6, models considering graphs beyond homophily, i.e., H2GCN, FAGCN, GPR-GNN, and our model, consistently display a more robust behavior than GCN and GAT. That is mainly because fake edges may connect nodes across different labels, and consequently cause erroneous information sharing in the conventional methods.

On the other hand, our ES-GNN beats all the state-of-the-arts by an average margin

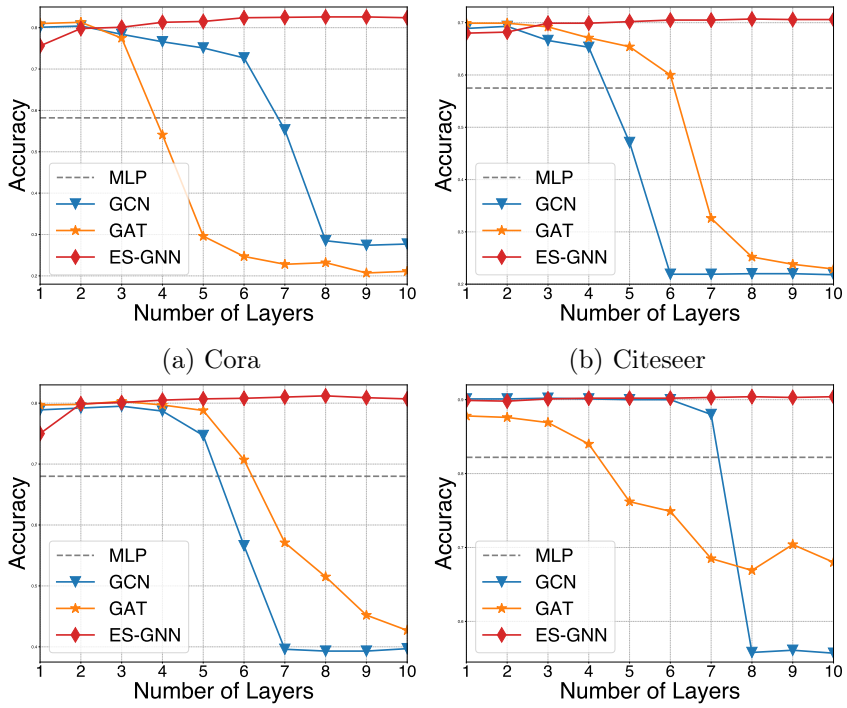


(a) Cora (b) Citeseer

(c) Pubmed (d) Polblogs

Figure 4.6: Results of different models on perturbed homophilic graphs. ES-GNN is able to identify the falsely injected (the task-irrelevant) graph edges, and exclude these connections from the final predictive learning, thereby displaying relative robust performance against adversarial edge attacks.

of 2% to 3% on Citeseer, Pubmed, and Polblogs while displaying relatively the same results on Cora. We attribute this to the capability of our model in associating node connections with learning tasks. Take Pubmed dataset as an example. We investigate the learned task-relevant topologies and find that 81.0%, 73.0%, 82.1%, 83.0%, 82.6% fake links get removed on adversatial graphs with perturbation rates from 20% to 100%. This also offers evidences supporting that our ES-layer is able to distinguish between task-relevant and irrelevant node connections. Therefore, despite a large number of false edge injections, the proximity information of nodes can still be reasonably mined in our model to predict their labels. Importantly, these empirical results also indicate that ES-GNN can still identify most of the task-irrelevant edges though no clear similarity or



(a) Cora (b) Citeseer

(c) Pubmed (d) Polblogs

Figure 4.7: Classification accuracy vs. model depths.

association between the connected nodes exists in the adversarial setting.

4.7.4 Alleviating Over-smoothing Problem

Graph over-smoothing is a prevalent issue in deep GNN models, where node features become indistinguishable as the number of layers increases, leading to degraded performance. This problem arises because repeated application of the smoothing operation, inherent in common GNNs' working mechanism, tends to homogenize node features across the graph. In order to verify whether ES-GNN alleviates the over-smoothing problem, we compare it with GCN and GAT by varying the layer number in Figure 4.7. It can be observed that these two baselines attain their highest results when the number of layers reaches around two. As the layer goes deeper, the accuracies of both GCN and GAT gradually drop to a lower point. On the contrary, our ES-GNN presents a stable curve. In spite of starting from a relative lower point, the performance of ES-GNN keeps

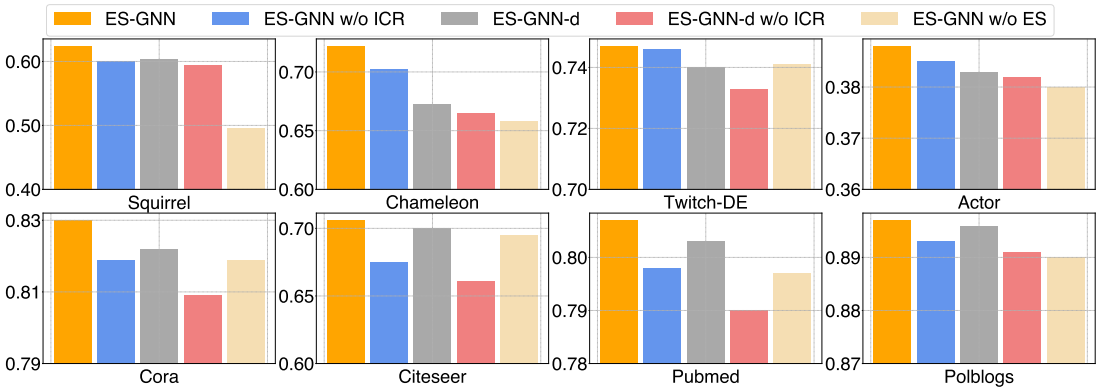


Figure 4.8: Ablation study of ES-GNN on eight datasets in node classification.

improving as the model depths increase, and eventually outperforms both GCN and GAT. The main reason is that, our ES-GNN can adaptively utilize proper graph edges in different layers to attain the task-optimal results with enlarged receptive fields. In other words, once an edge stops passing useful information or starts passing harmful messages, ES-GNN tends to identify it and remove it from learning the task-correlated representations, thereby having the ability of mitigating the over-smoothing problem.

4.7.5 Channel Analysis and Ablation Study

In this section, we compare ES-GNN with its variant ES-GNN-d which takes dual (both the task-relevant and irrelevant) channels for prediction, and perform an ablation study. Figure 5.5 provides comparison on eight real-world datasets as examples. Here, we first specify some annotations including 1) "w/o ICR": without regularization loss $\mathcal{L}_{\text{ICR}}$, and 2) "w/o ES": without edge splitting (ES-) layer. Overall, two conclusions can be drawn from Figure 5.5. First, ES-GNN is consistently better than ES-GNN-d, implying that the task-irrelevant channels indeed capture some false information where model performance downgrades even with the doubled feature dimensions. Second, removing either ICR or ES-layer from both ES-GNN and ES-GNN-d leads to a clear accuracy drop. That validates the effectiveness of our model designs.

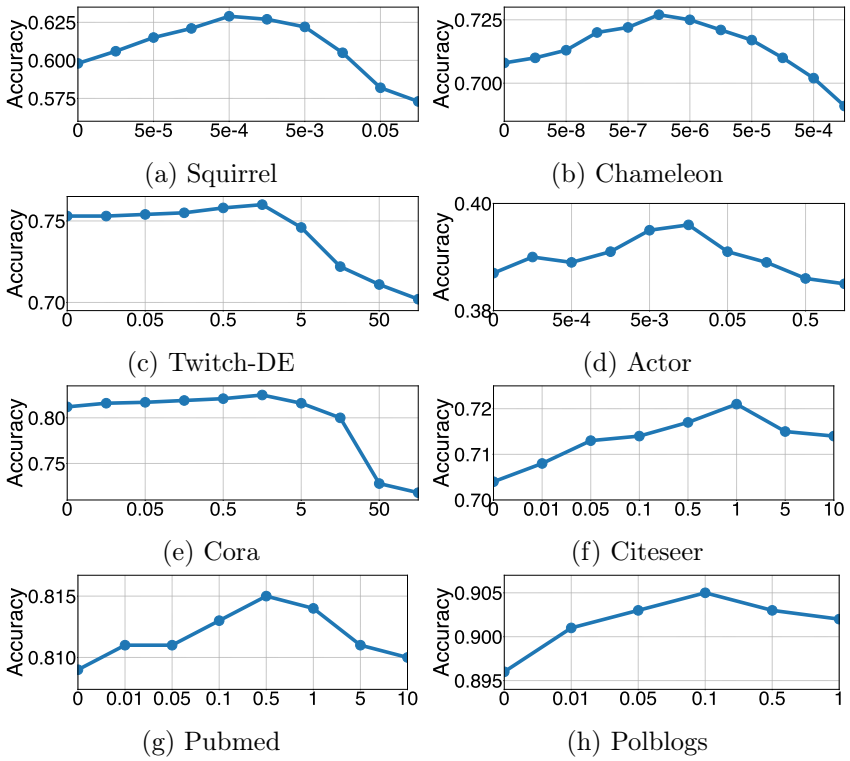


Figure 4.9: Sensitivity analysis of coefficient λ_{ICR}.

4.7.6 Sensitivity Analysis

We test the effect of the irrelevant consistency coefficient λ_{ICR}, and plot the learning performance of our model on eight real-world datasets as examples in Figure 4.9 by varying λ_{ICR} with different values. For example, the classification accuracy on Squirrel in Figure 4.9(a) goes up first and then gradually drops. Promising results can be attained by choosing λ_{ICR} from $[5\text{e}{-5}, 5\text{e}{-3}]$. Similar trends can be also observed on the other datasets, where λ_{ICR} is relatively robust within a wide albeit distinct interval.

4.7.7 Limitations and Future work

In this subsection, we turn our attention to the limitations of our proposed method. While our model has demonstrated strong performance across a range of benchmarks, it is crucial to acknowledge the areas where improvements are necessary. Here, we

discuss these limitations and propose directions for future research that could potentially enhance both the effectiveness and the scope of the model.

- Our current edge splitting strategy is relatively simplistic, primarily based on differentiating between intra-class and inter-class connections to define task-relevant and irrelevant edges. This approach may be too restrictive for more complex scenarios where the downstream task involves more than straightforward label classification problems. Future work could explore more nuanced strategies that account for the multifaceted relationships and dynamics within data, potentially leading to a more accurate and robust model in diverse environments.
- The current focus of our model is mainly confined to node-level tasks within a transductive learning framework. Extending our learning paradigm to handle graph-level tasks could significantly enhance the model's inductive inference capabilities. Investigating methods to adapt the model for such tasks would not only broaden its applicability but also strengthen its utility in scenarios where holistic graph understanding is crucial.

4.8 Conclusion

In this work, we develop a novel graph learning framework which enables GNNs to go beyond the strong homophily assumption on graphs. We manage to establish correlation between node connections and learning tasks through one plausible hypothesis, based on which ES-GNN is derived with an interpretable edge splitting. Our ES-GNN essentially partitions the original graph structure into the task-relevant and irrelevant topologies as guide to disentangle node features, whereby the classification-harmful information can be disentangled and excluded from the final prediction target. Theoretical analysis illustrates our motivation and offers interpretations on the expressive power of ES-GNN on different types of networks. To provide empirical verification, we conduct extensive experiments over 11 benchmark and 1 synthetic datasets. The node classification results show that ES-GNN constantly outperforms the other 9 competitive GNNs (including 5

state-of-the-arts explicitly designed for heterophily) on graphs with either homophily or heterophily. In particular, we also conduct analysis on the split edges, correlation among disentangled features, model robustness, and the ablated variants. All of these results demonstrate the success of ES-GNN in identifying graph edges between different types, which also validates the effectiveness of our interpretable edge splitting.

Chapter 5

Graph Neural Networks with Diverse Spectral Filtering

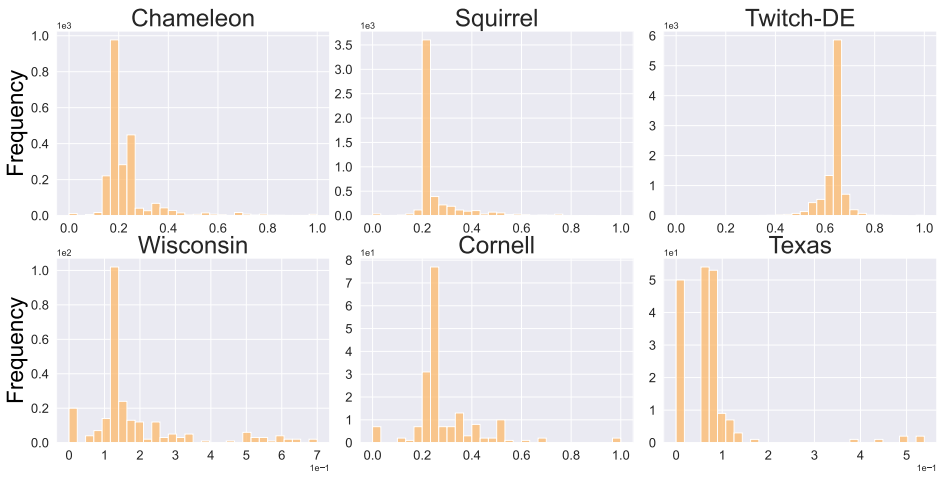
In this chapter, we investigate the nuances of *heterophilic linking patterns* that we tackle in the previous chapter, uncovering a notable presence of regional heterogeneity (or local structural variations) on the graph. While certain spectral GNNs have been proposed to learn from arbitrary label patterns using learnable filters, we identified a critical shortfall in their approach: homogeneous spectral filtering often fails to capture the regional heterogeneity prevalent in complex networks. This observation led to the development of our novel Diverse Spectral Filtering (DSF) framework. DSF automatically learns node-specific filter weights with awareness of node positions, adeptly capturing both local structural variations and global characteristics. This advancement not only refines GNNs' capacity to navigate real-world graph diversity but also enhance their interpretability under complex graph scenarios. This work has been published in the Proceedings of the ACM Web Conference (WWW) 2023.

5.1 Introduction

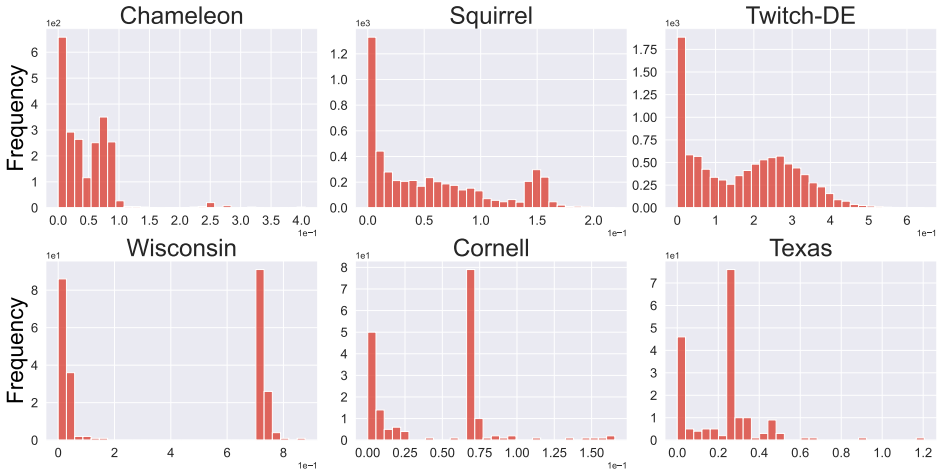
Recent years have witnessed the explosive growth of learning from graph-structured data. As an emerging technique, Graph Neural Networks (GNNs) have recently attracted

significant attention in handling data with complex relationships between entities. Capable of exploiting node features and graph topology simultaneously and adaptively, GNNs achieve state-of-the-art performance in a wide variety of graph analytical tasks, such as social analysis and recommendation system [13, 45]. Spectral GNNs are a class of GNN models that implement convolution operations in the spectral domain. Recent studies show that modern variants mostly function as polynomial spectral filters [35, 37, 54, 55, 66, 74]. Specifically, these filters transform the input source (node features) into a new desired space by selectively attenuating or amplifying its Fourier coefficients induced by the graph Laplacian. Existing efforts either design or learn the polynomial coefficients to simulate different types of filters including low, band, and/or high-pass. Despite their success, high degree polynomials are necessary as typically required by their expressive power [37, 54, 177] so as to reach high-order neighborhoods. Nevertheless, most spectral GNNs would fail practically due to the overfitting and/or over-squashing problem [54, 74, 178]. They usually end up enforcing *identical* filter weights among nodes, albeit lying in different network areas, to mine their distinct local contexts (see details in Section 5.4.1). Namely, the existing spectral GNN models (including GPR-GNN [54], BernNet [37], and JacobiConv [74]) are mostly restricted in the *homogeneous* spectral filtering framework, and focus on the uniform filter weights learning. Such limitation is induced by simplifying diverse regional graph patterns as homogeneous ones at different localities.

However, real-world networks typically exhibit *heterogeneous* mixing pattern [64], i.e., different graph parts may possess diverse characteristics (e.g. local assortative level could vary across the graph as illustrated in Figure 5.1). Apparently, GNNs with classic *homogeneous* spectral filtering are inadequate to model the varying regional pattern; this could result in poor interpretability on the micro graph mining which is important in accomplishing node-level tasks. A single shared weight set tends to pull the model in many opposite directions, which may lead to a biased model that merely captures the most common graph patterns while leaving others not well covered. Ideally, in order to properly mine different local contexts, distinct filter weights might be needed for different



(a) Local Label Homophily



(b) Local Graph Frequency

Figure 5.1: Distributions of two graph properties on real-world graphs (see Section 5.4.1).

nodes. To do so, one may parameterize each node a separate set of trainable filter weights. Unfortunately, this would substantially increase model complexity and cause severe overfitting to local noises, especially in case of large-scale graphs with complex linking patterns (see Section 5.7.1).

This work focuses on adapting spectral GNN models to graphs with diverse mixing patterns. We argue that, instead of parameterizing each node arbitrarily different filter weights or a uniform weight, a reasonable design should be *built upon a shared global model whilst locally adapted to each node with awareness of its location in the graph.*

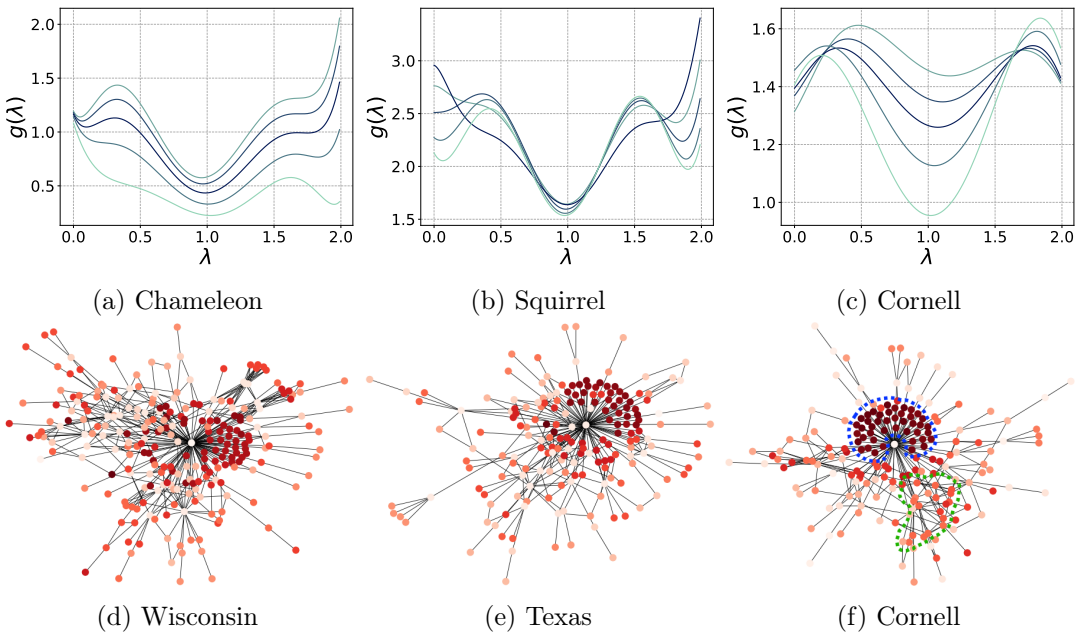
Such a proposition is well evidenced by the key observation that nearby nodes tend to display similar local contexts because of their overlapped neighborhoods. For nodes far apart, they are likely to have more possibilities, e.g., even if residing at disjoint graph regions, these nodes may also possess akin local structures due to their similar positions such as graph borders (see Figure 5.2(d)-(f)). As such, we aim to take advantage of such rationale as a guide to make node-wise adjustments. To this end, we formulate a novel optimization problem to first encode the positional information of nodes. It embeds graph vertices into a low-dimensional coordinate space, based on which we learn node-specific coefficients properly to alter the original filter weights. Accordingly, the global graph filter can be localized on the micro level, allowing individual nodes to adaptively exploit their diverse local contexts. Meanwhile, some beneficial invariant graph properties can still be preserved by virtue of the mutual filter weights. The proposed framework, named *diverse* spectral filtering (DSF), flexibly handles the complex graph and makes a proper balance between its conformal and disparate regional patterns with enhanced interpretability, as shown in Figure 5.2. Besides, our DSF is easy to implement and can be readily plug-and-play in any spectral GNNs with considerable performance gains.

To summarize, the main contributions of this work are three-fold:

- We show that many existing spectral GNNs are restricted in the form of *homogeneous* spectral filtering, and identify the need to break this ceiling to deal with complex graphs with regional *heterogeneity*.

- We propose a novel *diverse* spectral filtering (DSF) framework to learn diverse and interpretable spectral filters on the micro level, which consistently leads to performance gains for many spectral GNNs.

- We showcase our DSF framework on three state-of-the-arts including GPR-GNN [54], BernNet [37], and JacobiConv [74]. Extensive evaluations on 10 real-world datasets demonstrate the superiority of DSF framework in node classification tasks.



(a) Chameleon (b) Squirrel (c) Cornell

(d) Wisconsin (e) Texas (f) Cornell

Figure 5.2: (a)-(c) Diverse filters learned from real-world networks, where five representative curves are plotted for illustration. On each graph, these filters display similar overall shapes but different local details in function curves, showing the capability of our DSF in capturing both the global graph structure and locally varied linking patterns. (d)-(f) Visualization of node-specific filter weights, where alike color indicates similar filter weights between nodes. Overall, nodes can be differentiated based on their disjoint underlying regions as circled by the blue and green dashed lines, and far-reaching nodes can still learn similar filter weights due to their akin local structures. E.g., vertices on the graph border are mostly ingrained in a line subgraph such as $\bullet - \bullet - \bullet$, and some unusual cases can be handled (see details in Section 5.7.2). These results justify the enhanced model interpretability by learning diverse spectral filters on the micro level.

5.2 Related Works

Spectral Graph Neural Networks. Existing GNNs are often divided into spatial-based and spectral-based methods. The former is mainly built upon a message-passing framework [20] where nodes exchange information with their spatial neighbors. The latter is rooted in spectral graph theory [21, 24], mostly operating as polynomial spectral filters divided into two categories [37, 54, 74]. One class is spectral GNNs with fixed filters: GCN [35] truncates Chebyshev polynomials to a simple first-order, and works as a low-pass filter [58]. APPNP [36] constructs filter functions with personalized

PageRank [179] to overcome the over-smoothing problem [180]. GNN-LF/HF [67] is derived from the perspective of graph optimization to simulate low/high-pass filters. In the second class, spectral GNNs are mostly designed with trainable filters: ChebNet [55] approximates the filtering operation with Chebyshev polynomials whose coefficients are learnable. AdaGNN [181] learns adaptive filters to model each feature channel independently. GPR-GNN [54] extends APPNP [36] by directly parameterizing its filter weights and training them with gradient descent. ARMA [66] takes rational filter functions while approximating them still with polynomials. BernNet [37] employs positive weight constraints in learning spectral filters with the Bernstein polynomial approximation. Wang and Zhang [74] further analyze the expressive power of spectral GNNs in a general form, and propose JacobiConv with an orthogonal polynomial basis and feature-wise filter learning.

For both types, it is identified that most spectral GNNs apply a *homogeneous* setting for spectral filtering. These present methods tend to focus on the most frequent graph layouts while under-exploring the rich and diverse local patterns. To alleviate this issue, we introduce a *diverse* spectral filtering (DSF) framework to enhance spectral GNNs with trainable diverse filters. It is worth noting that though both JacobiConv [74] and AdaGNN [181] learn multiple filters in a seemingly similar way, they are essentially different from our method in nature. Concretely, their adaptive filters are mainly for studying each feature channel independently, whilist our diverse filters aim at individual context modeling for each node.

Graphs with Complex Linking Patterns. Early wisdom in the community was mainly dedicated to learning from graphs with strong homophiliy (assortativity) where most connected nodes share similar attributes and same label [35, 36, 53]. Until 2020, [60] and [4] first emphasized the importance of studying GNNs in the heterophily (disassortativity) setting, thus categorizing real-world networks into homophilic and heterophilic graphs based on edge homophily ratio. Recently massive works [54, 61, 70, 144, 182] have been done which focus more on complex graph scenarios. For example, FAGCN [61] captured both similarity and dissimilarity between pairwise nodes. As much attention

from the community was paid in analyzing graph patterns on the macro/global-level, some researchers [64, 183] start looking into the micro/local graph structures surrounding nodes. In particular, [64] introduced a node-level assortativity to show heterogeneous mixing patterns inherent in real-world graphs. However, these existing works mostly tackle this regional heterogeneity phenomenon under the intuitive message passing framework [20]. Surprisingly, none of them attempts to solve it from the spectral perspective, a theoretically more elegant framework. To this end, we explore in this work how to learn spectral GNNs on graphs with diverse mixing patterns and propose a novel framework called *diverse* spectral filtering.

It is noted that a recent proposal [73] shares some similarity with our work. This method takes the idea of dynamic neural networks [184], and introduce PA-GNN based on GPR-GNN [54] by learning node-specific weight offsets. Though PA-GNN is also originated in a spectral-based framework, it leverages a simple but less-justified node-specific aggregation scheme and encodes different (but probably inconsistent) sources of information for prediction. On one hand, this would fail to learn interpretable filters; on the other hand, such drawback limits the accuracy gain and may even hurt the performance, which can be later seen in the experiment part.

Positional Encoding. Positional encoding (PE) aims to quantify the global position of, e.g., pixels in images, words in texts, and nodes in graphs. It plays a crucial role in facilitating various neural networks such as CNNs [185], RNNs [186], and Transformer [187]. For GNNs, PE has been widely used to increase their model expression [154, 188, 189] as bounded by Weisfeiler-Leman (WL) graph isomorphism test [52, 190, 191]. However, existing focus is mainly put on message-passing GNNs [20], while few research has been conducted on the models defining graph filter in the spectral domain. Recently, [74] has established a connection between WL test and the expressive power of spectral GNNs, which motivates us to investigate further how to leverage PE for spectral GNN models. We note that PA-GNN [73] also extract latent positional embeddings from the graph structure, which however cannot be better changed/adjusted to tasks and is mixed with other (even incompatible) attributes. Compared to this, our DSF framework model the

positional information of nodes in an independent channel via an iterative updating, thereby producing more expressive and task-beneficial representations. The advantages of this decoupled learning paradigm on node positional and structural features are also verified by other recent works [188, 189].

5.3 Preliminaries

In this section, we lay the groundwork for the mathematical concepts crucial to understanding spectral GNNs discussed later in this work. Moving beyond the general background introduced previously, we now concentrate specifically on the mathematical details of Laplacian decomposition and graph spectral filtering.

Laplacian Decomposition. Let $\hat{\mathbf{L}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ denote the eigen decomposion of $\hat{\mathbf{L}}$, where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N] \in \mathbb{R}^{N\times N}$ is a matrix of eigenvectors, and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_N)$ is the diagonal matrix of eigenvalues. As $\hat{\mathbf{L}}$ is positive semidefinite, we have $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$, of which $\{\mathbf{u}_n\}_{n=1}^N$ is also called the Laplacian eigenbases. We then have

$$\lambda_n = \mathbf{u}_n^T\hat{\mathbf{L}}\mathbf{u}_n = \sum_{(v_p,v_q)\in\mathcal{E}} (\frac{1}{\sqrt{\text{deg}_p}}\mathbf{u}_{n,p} - \frac{1}{\sqrt{\text{deg}_q}}\mathbf{u}_{n,q})^2 \tag{5.1}$$

which measures the frequency or smoothness level of each eigenbasis $\mathbf{u}_n$ on the graph. In this work, we refer to λ_n as the global graph frequency. Based on all above, the graph Fourier transform and inverse Fourier transform can be respectively formulated as $\mathbf{S} = \mathscr{F}(\mathbf{X}) = \mathbf{U}^T\mathbf{X}$ and $\mathbf{X} = \mathscr{F}^{-1}(\mathbf{S}) = \mathbf{U}\mathbf{S}$, where $\mathbf{S}$ is often called as the Fourier transformed features or Fourier coefficients of $\mathbf{X}$.

Graph Spectral Filtering. The central idea of spectral GNNs is to transform the graph signal (an instance of node features) in the Fourier space by applying graph spectral filters. They usually take the form as $\mathbf{Z} = g(\hat{\mathbf{L}})\mathbf{X} = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^T\mathbf{X}$ where $g : [0, 2] \rightarrow \mathbb{R}$ is a filter function defined on the spectrum of graph Laplacian. This function creates frequency response to filter different components of $\mathbf{X}$ on Laplacian eigenbases. Take one-channeled $\mathbf{X}$ as an example. As $\mathbf{S} = \mathbf{U}^T\mathbf{X} = [s_1, s_2, ..., s_N]^T$, we

have $\mathbf{X} = \sum_{n=1}^{N} s_n \cdot \mathbf{u}_n$ and $\mathbf{Z} = \sum_{n=1}^{N} \big(g(\lambda_n)s_n\big) \cdot \mathbf{u}_n$ with scalar s_n. It can be seen that node feature matrix $\mathbf{X}$ is mapped into a new $\mathbf{Z}$, by either decreasing or increasing Fourier coefficients $\mathbf{S}$ via $s_n \mapsto g(\lambda_n)s_n$ selectively. Recent studies have shown that most spectral GNN models implement the filter function $g(\cdot)$ with polynomials [37, 66, 74] following $\mathbf{Z} = \sum_{k=0}^{K} \omega_k \hat{\mathbf{L}}^k \mathbf{X} = \sum_{k=0}^{K} \alpha_k P_k(\hat{\mathbf{L}})\mathbf{X}$, where both ω_k and α_k denote the polynomial coefficient (also called filter weight), and $P_k : [0, 2] \to \mathbb{R}$ is a polynomial basis in the k^{th} order. Taking one-channel $\mathbf{X}$ as an example, the existing spectral GNNs can then be unified as

$$\mathbf{Z} = \sum_{k=0}^{K} \alpha_k P_k(\hat{\mathbf{L}})\mathbf{X} = \sum_{n=1}^{N} \hat{s}_n \cdot \mathbf{u}_n \tag{5.2}$$

where $\hat{s}_n = \sum_{k=0}^{K} \alpha_k P_k(\lambda_n) s_n$ for brief symbolization. Present efforts either fix or learn filter weights with different classes of polynomial basis. For instance, APPNP [36] leverages personalized PageRank [179] to make polynomial basis $P_k(\lambda) = (1 - \lambda)^k$ and set $\alpha_k = \frac{\alpha^k}{1-\alpha}$ with constant hyper-parameter α. As an extension of APPNP, GPR-GNN [54] directly trains α_k with gradient descent. A comprehensive summarization of various spectral GNNs as polynomial spectral filters can be found in [74].

5.4 Diverse Spectral Filtering

In this section, the motivation of *diverse* spectral filtering is first provided with both theoretical and empirical analysis. We then present our novel diverse filtering framework.

5.4.1 Motivations

The unified formula in Eq. (5.2) can be considered as the *homogeneous* spectral filtering, where all nodes share the identical coefficient $\hat{s}_n$ equally operated on their basis signals, i.e., all elements in $\mathbf{u}_n$, for feature transformation. It seems reasonable as one can learn arbitrary $\hat{s}_n$ with a polynomial graph filter [24], which formally requires high-degree polynomials and reaching high-order node neighborhood [37, 54, 177]. However, aggregating/passing information across a long path via $\hat{\mathbf{L}}^k \mathbf{X}$ with $k \to \infty$ is prone to cause overfitting to noises and/or over-squashing problem [178]. [54] practically show

that the polynomial coefficients in Eq. (5.2) converges to zero as k gets larger. With this empirical finding, [74] even propose strategies to optimize $\{\alpha_k\}_{k=0}^{K}$ with decreasing scale. All the above reveal the local modeling nature of the existing spectral GNNs. In other words, nodes, albeit lying in different graph parts, are enforced to mine their distinct local contexts with the identical filter weights $\{\alpha_k\}_{k=0}^{K}$. Such filtering scheme implicitly assumes the similar distributions between different graph regions.

This hypothesis however may not be accurate due to the intrinsic complexity in forming real-world networks. To make further investigation, we define two essential graph properties on the local graph level to empirically observe their changing behaviors across the graph. The definitions are given below.

Definition 4 (Local Label Homophily)**.** We define the Local Label Homophily as a measure of the local homophily level surrounding each node v_i:

$$h_i = \frac{|\{(v_p, v_q)|\mathbf{y}_p = \mathbf{y}_q \wedge (v_p, v_q) \in \mathcal{E}_{i,k}\}|}{|\mathcal{E}_{i,k}|}$$

Here, h_i directly computes the edge homophily ratio[1] on the subgraph made up of the k-hop neighbors, and $\mathcal{E}_{i,k} = \{(v_p, v_q)|v_p, v_q \in \mathcal{N}_{i,k} \wedge (v_p, v_q) \in \mathcal{E}\}$ denotes its edge set.

Definition 5 (Local Graph Frequency)**.** The Local Graph Frequency is defined by measuring the local smoothness level of the decomposed Laplacian eigenbases, and for each node v_i we have:

$$\lambda_{n,i} = \sum_{(v_p, v_q) \in \mathcal{E}_{i,k}} (\frac{1}{\sqrt{\deg_p}}\mathbf{u}_{n,p} - \frac{1}{\sqrt{\deg_q}}\mathbf{u}_{n,q})^2$$

where $\lambda_{n,i}$ denotes the frequency or smoothness level of each Laplacian eigenbasis $\mathbf{u}_n$ upon the subgraph induced by the k-hop neighbors. Since all summed elements in Eq. (5.1) are positive and $\mathcal{E}_{i,k} \subseteq \mathcal{E}$, we can always have a $\xi_i \in (0, 1)$ such that $\lambda_{n,i} = \xi_i \lambda_n$.

[1]This widely used metric measures the fraction of edges connecting nodes within the same class [4]. It is defined as $\mathcal{H} = \frac{|\{(v_i,v_j)|\mathbf{y}_i=\mathbf{y}_j \wedge (v_i,v_j)\in\mathcal{E}\}|}{|\mathcal{E}|}$ with values ranging from 0 to 1. Higher values of $\mathcal{H}$ indicate stronger homophily (or conversely, weaker heterophily) among the nodes.

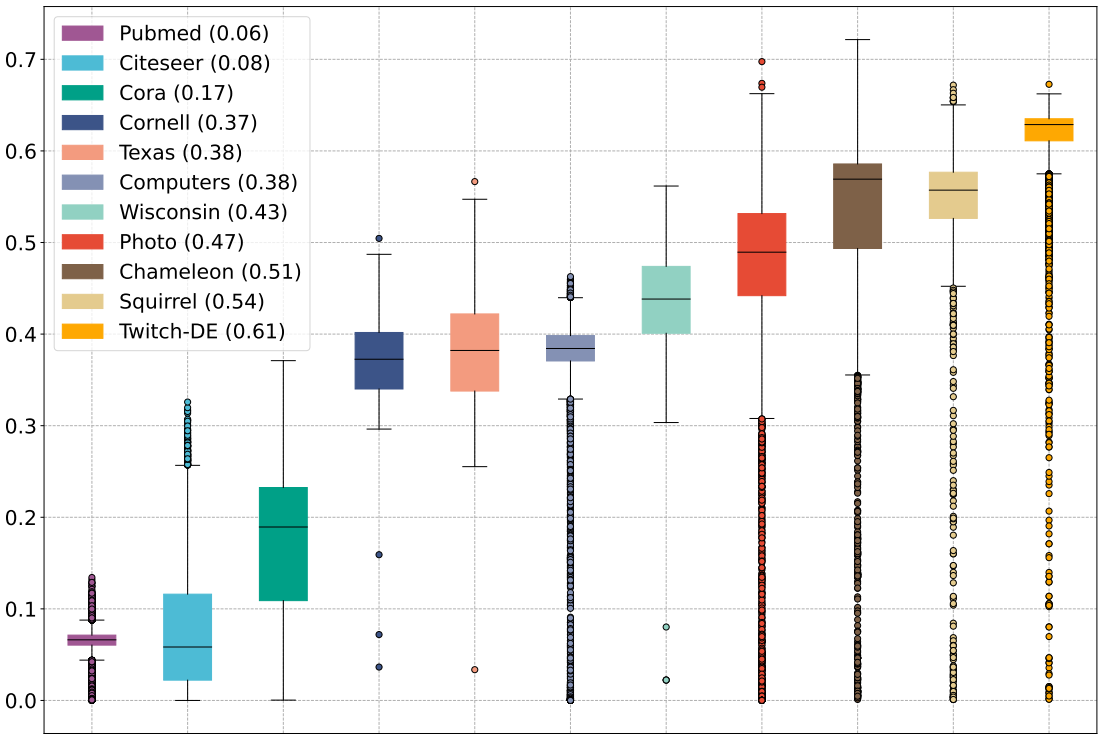


Figure 5.3: Diversity distributions of local graph frequency on various networks, where numbers nearby data names are the mean diversity values.

In this work, we take two-hop neighborhood to illustrate local graph patterns. For Local Graph Frequency, as we usually have a large set of the decomposed eigenbases $\{\mathbf{u}_n\}_{n=1}^N$, visualizing them all is not feasible. Thus, we propose to quantify the diversity degree of local graph frequency w.r.t. each eigenbase, focusing on the most representative one (nearest to the mean). This measure is defined as below.

Definition 6 (Diversity of Local Graph Frequency)**.** For each group of Local Graph Frequency $\{\lambda_{n,i}\}_{n=1}^N$, we first rank the data, divide them into M bins, and count the case number $p_{n,m}$ in each bin. Then, we normalize the results into $\hat{\mathbf{P}}_n = \{\hat{p}_{n,m}\}_{m=1}^M$ where $\hat{p}_{n,m} = p_{n,m}/\sum_{j=1}^M p_{n,j}$. Finally, we have the Diversity of Local Graph Frequency w.r.t. the n-th eigenvector as:

$$\tau_n = \max\{\mathrm{JS}(\hat{\mathbf{P}}_n||\mathbf{Q}_1), \mathrm{JS}(\hat{\mathbf{P}}_n||\mathbf{Q}_2), ..., \mathrm{JS}(\hat{\mathbf{P}}_n||\mathbf{Q}_M)\} \tag{5.3}$$

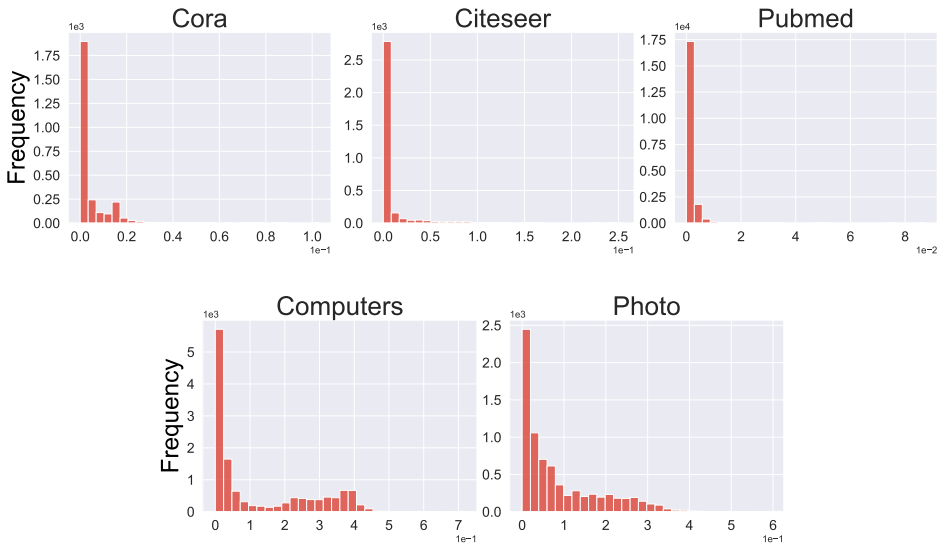


Figure 5.4: Additional distributions of Local Graph Frequency. Different from Figure 5.1, more uniform distributions can be observed on homophilic graphs.

where $\mathbf{Q}_i$ is a unit vector at position i, JS denotes the Jensen-Shannon divergence, and τ_n ranges from 0 to 1 where the higher value indicates high degree of diversity.

We displays the distribution of $\{\tau_n\}_{n=1}^{N}$ across networks with different homophily levels (see dataset information in Table 6.1) in Figure 5.3. It is can be seen that only homophilic graphs like Cora, Citeseer, and Pubmed, exhibit lower degrees of diversity. This is intuitive since connections in homophilic graphs typically follow a uniform pattern of linking similar nodes together. Conversely, in other datasets, especially those with heterophilic graphs, a relatively higher degree of diversity can be observed.

Figure 5.1 shows the distribution of "Local Label Homophily" and "Local Graph Frequency" on various real-world graphs. Overall, we observe skewed and even multi-modal distributions. These phenomena imply that local structural patterns are not uniformly distributed between different graph regions, but exhibiting evident heterogeneity (see more examples in Figure 5.4). More importantly, in the spectral domain, the global graph frequency λ_n usually fails to capture the diverse local characteristics of $\mathbf{u}_n$ as shown in Figure 5.2(b). Thus, weighing $\mathbf{u}_n$ with the only one scalar coefficient, $\hat{s}_n$ computed

as a function of λ_n, may not be appropriate and tends to cause ineffective modeling. In other words, a diverse filtering framework appears necessary so that one could fully exploit the heterogeneous mixing patterns for adaptive micro graph learning.

5.4.2 Diverse Filtering Framework

To implement diverse filtering, we aim to improve the classic *homogeneous* spectral filtering by endowing each node a different set of transforming coefficients on the basis signals. Particularly, the scalar $\hat{s}_n$ is expanded as a vector $\hat{\mathbf{s}}_n = [\hat{\mathbf{s}}_{n,1}, \hat{\mathbf{s}}_{n,2}, ..., \hat{\mathbf{s}}_{n,N}]^T$ with the same dimensions as the eigenbasis $\mathbf{u}_n$. Eq. (5.2) can be thereby enhanced into $\mathbf{Z} = \sum_{n=1}^{N} \hat{\mathbf{s}}_n \odot \mathbf{u}_n$ where $\odot$ denotes element-wise multiplication, and each element in $\hat{\mathbf{s}}_n$ independently operates on the corresponding signal in $\mathbf{u}_n$. Since $\hat{s}_n$ is originally expressed as a polynomial function of λ_n, it is reasonable to make

$$\hat{\mathbf{s}}_{n,i} = f(\lambda_{n,i}) = \sum_{k=0}^{K} \alpha_k P_k(\lambda_{n,i}) s_n \tag{5.4}$$

based on our analysis in the previous section, where $\lambda_{n,i}$ denotes the local graph frequency specified in Definition 5. However, it would be computationally expensive to calculate $\lambda_{n,i}$, which requires not only Laplacian decomposition but also subgraph extraction. To mitigate this issue, we turn to exploiting the substitution using $\lambda_{n,i} = \xi_i \lambda_n$ s.t. $0 < \xi_i < 1$ with the following proposition.

Proposition 1. Suppose a K-order polynomial function $f : [0, 2] \to \mathbb{R}$ with polynomial basis $P_k(\cdot)$ and coefficients $\{\alpha_k\}_{k=0}^{K}$ in real number. For any pair of variables $x, \hat{x} \in [0, 2]$ satisfying $x = \xi\hat{x}$ where ξ is a constant real number, we always have a function $g : [0, 2] \to \mathbb{R}$ with the same polynomial basis but a different set of coefficients $\{\beta_k\}_{k=0}^{K}$ such that $f(x) = g(\hat{x})$.

Proof. We denote $f(x) = \sum_{k=0}^{K} \alpha_k P_k(x)$, and substituting $x = \xi\hat{x}$ gives us $f(\xi\hat{x}) = \sum_{k=0}^{K} \alpha_k P_k(\xi\hat{x})$. As the maximum order on variable x is K, we can always express $f(\xi\hat{x})$ in a power series with new coefficient set $\{\omega_k\}_{k=0}^{K}$ where $\omega_k \in \mathbb{R}$, i.e., $f(\xi\hat{x}) = \sum_{k=0}^{K} \omega_k (\xi\hat{x})^k$. Moreover, since ξ is a constant, we can view $f(\xi\hat{x})$ as a function of

variable $\hat{x}$, i.e., $g(\hat{x}) = \sum_{k=0}^{K}(\omega_k \xi^k)\hat{x}^k$, Similarly, with the expressive power of polynomial basis $P_k(\cdot)$, we can always find a new coefficient set $\{\beta_k\}_{k=0}^{K}$ where $\beta_k \in \mathbb{R}$ making $g(\hat{x}) = \sum_{k=0}^{K}\beta_k P_k(\hat{x})$. Therefore, we have $f(x) = g(\hat{x})$ where these two functions are made up of the same polynomial basis $P_k; [0, 2] \rightarrow \mathbb{R}$ and two different coefficient sets $\{\alpha_k\}_{k=0}^{K}$ and $\{\beta_k\}_{k=0}^{K}$. □

Proposition 1 suggests that the polynomial $f(\lambda_{n,i})$ computing $\hat{\mathbf{s}}_{n,i}$ in Eq. (5.4) can be reformulated into another function of variable λ_n, using the same basis $P_k(\cdot)$ but a different coefficient set $\{\beta_k\}_{k=0}^{K}$, i.e., $\hat{\mathbf{s}}_{n,i} = f(\lambda_{n,i}) = \sum_{k=0}^{K}\alpha_k P_k(\xi_i \lambda_n)s_n = \sum_{k=0}^{K}\beta_{k,i}P_k(\lambda_n)s_n$. Therefore, our *diverse* spectral filtering can be formulated as:

$$\mathbf{Z} = \sum_{n=1}^{N} \hat{\mathbf{s}}_n \odot \mathbf{u}_n = \sum_{k=0}^{K} \mathrm{diag}(\beta_{k,1}, \beta_{k,2}, ..., \beta_{k,N}) P_k(\hat{\mathbf{L}})\mathbf{X}$$

where each node v_i is parameterized with a different set of filter weights $\{\beta_{k,i}\}_{k=0}^{K}$. The remaining issue is then how to learn these weights. Existing state-of-the-art spectral GNNs [37, 54, 74] usually set filter weight as free parameters to be directly trained. However, in our framework, doing this would not only lead to a high computational complexity, but also could cause severe overfitting to local noises. In the following, we introduce two strategies so as to deal with the issue.

Position-aware Filter Weights. It has been shown that $\{\beta_{k,i}\}_{k=0}^{K}$ is utilized to mine the local context of each node v_i. The differences among these filter weight sets are meant to capture regional heterogeneity on the graph. From another angle, if the filter weights are learned to be similar between nodes, they are more likely to lie in the same region sharing almost identical local structural patterns. While, distant node pairs may have more possibilities, e.g., even if residing at disjoint graph regions, these vertices could still possess alike local subgraphs due to their similar positions in the network such as graph borders. This motivates us to make use of node positional information as a guide to learn diverse filter weights.

To do so, the first step is to encode the node positions into a latent space while preserving their graph-based distance. To attain this, inspired by graph signal denois-

ing [67] and Laplacian loss [192, 193], we formulate an novel optimization problem with the objective $\mathcal{L}_p$:

$$\arg\min_{\mathbf{P}} \ \mathcal{L}_p = \|\mathbf{X}_p - \mathbf{P}\|_F^2 + \kappa_1 tr(\mathbf{P}^T \hat{\mathbf{L}} \mathbf{P}) + \kappa_2 \|\mathbf{P}^T\mathbf{P} - \mathbf{I}_d\|_F^2 \tag{5.5}$$

where $\mathbf{P} = [\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_N]^T \in \mathbb{R}^{N\times d}$ is a matrix of node positional embeddings, $\mathbf{X}_p$ initializes $\mathbf{P}$ (more information can be seen in Section 5.5), $\mathbf{I}_d$ is an $d \times d$ identity matrix, and κ_1, κ_2 are two non-negative trade-off coefficients. The first term guides $\mathbf{P}$ to be close to $\mathbf{X}_p$, while the middle term enforces adjacent nodes to stay closer in the positional latent space. A penalty term is lastly appended to ensure orthogonal feature channels for attaining a valid coordinate system. Minimizing $\mathcal{L}_p$ therefore enables a canonical positioning of nodes in the graph. We take an iterative gradient method to solve Eq. (5.5), and derive the iterative updating rule:

$$\mathbf{P}^{(k+1)} = \eta_1 \mathbf{X}_p + (1-\eta_1)\big((1+\eta_2)\hat{\mathbf{A}} - \eta_2(\mathbf{P}^{(k)}\mathbf{P}^{(k)^T})\big)\mathbf{P}^{(k)} \tag{5.6}$$

where $\mathbf{P}^{(0)} = \mathbf{X}_p$, $\eta_1 = \frac{1}{1+\kappa_1-2\kappa_2}$, $\eta_2 = \frac{2\kappa_2}{\kappa_1-2\kappa_2}$, and the stepsize is set as $\frac{\eta_1}{2}$. Both η_1 and η_2 are constant hyper-parameters searched from $\{0, 0.1, ..., 1.0\}$ by 0.1, and the case $\eta_1 = 1.0$ examines the effectiveness of the initial $\mathbf{X}_p$. By iteratively updating $\mathbf{P}^{(k)}$, the objective $\mathcal{L}_P$ can be progressively minimized to solve the optimization problem. In practical training, we need to normalize $\mathbf{P}^{(k)}\mathbf{P}^{(k)^T}$ to ensure numerical stability and benefit computational efficiency. Thus, Eq. (5.6) is enhanced into

$$\mathbf{P}^{(k+1)} = \eta_1 \mathbf{X}_p + (1-\eta_1)\big((1+\eta_2)\hat{\mathbf{A}} - \eta_2\sigma(\mathbf{P}^{(k)}\mathbf{W}\mathbf{P}^{(k)^T})\big)\mathbf{P}^{(k)} \tag{5.7}$$

where σ is a sigmoid function to produce values between 0 to 1, and $\mathbf{W} \in \mathbb{R}^{d\times d}$ is a learnable mapping matrix to improve model capacity. Besides, we also add a tanh activation function between updating steps to make both positive and negative values in the derived coordinate system, i.e., $\mathbf{P}^{(k+1)} \leftarrow \text{Tanh}(\mathbf{P}^{(k+1)})$. We further refer to this process as iterative positional encoding (IPE).

So far, the positional information of nodes can be encoded into a low-dimensional metric space by applying Eq. (5.7) recursively. To learn polynomial filter weights with awareness of node positions, it is empirically found that a simple yet effective non-linear mapping works well:

$$\beta_{k,i} = \sigma_p(\mathbf{W}^{(k)^T}\mathbf{P}_i^{(k)} + \mathbf{b}^{(k)}) \tag{5.8}$$

where $\mathbf{W}^{(k)} \in \mathbb{R}^d$ and $\mathbf{b}^{(k)} \in \mathbb{R}$ are learnbale parameters for polynomial order k, and σ_p is an activation function. As such, we manage to train models appropriately with guidance from node positions, while avoiding the possible overfitting induced by parameterizing each node arbitrarily with different filter weights. Moreover, model complexity can also be greatly reduced with the lowered number of trainable parameters from $N \times (K+1)$ to $(d+1)\times(K+1)$ where $d \ll N$ is feature dimension. Besides, we show our DSF framework, albeit with the simple mapping formula in Eq. (5.8), is able to deal with complex or even unusual graph cases in Section 5.7.2.

Local and Global Weight Decomposition. Though real-world networks exhibit rich and diverse local patterns, the global graph structure still matters, as it encodes some invariant graph properties while simultaneously pruning local noises. Accordingly, we decompose our node-specific filter weights $\beta_{k,i}$ into two independent coefficients γ_i and $\theta_{k,i}$ with multiplication, i.e., $\beta_{k,i} = \gamma_i\theta_{k,i}$. We call $\gamma_i \in \mathbb{R}$ the global filter weight responsible for capturing the global graph structure, and name $\theta_{k,i} \in (-1, 1)$ as local filter weight which is learned by the non-linear mapping in Eq. (5.8). As a benefit, the local coefficients can flexibly rescale and/or flip the sign of the global ones to capture node differences, while global connecting patterns can also be mined with diminished noisy information. In this work, we call this technique Local and Global Weight Decomposition (LGWD). Additionally, we find that JacobiConv [74] also leverages a similar technique called PCD that decomposes the filter weight as $\alpha_k = \pi_k \prod_{s=1}^{k} \rho_s$ (we replace their symbols to avoid confusions with ours). This design aims to facilitate model training with a decreasing scale on $\{\alpha_k\}_{k=0}^{K}$ as k grows, and all the nodes still share the same parameter set. In contrast, our method works on individual vertices through disentangling the

globally shared and locally varied node coefficients.

5.5 Overall Algorithm

5.5.1 Implementation Details

As our DSF framework is independent of any underlying model, it can flexibly improve any spectral GNNs. In our experiments, we showcase it over three SOTA baselines including GPR-GNN, BernNet, and JacobiConv. A comprehensive summary of their designed trainable spectral filters can be found in [74]. In practice, we find that the term $\mathbf{P}^{(k)}\mathbf{P}^{(k)^T}$ in Eq. (5.6) involves a high computational complexity in $\mathcal{O}(N^2)$, and possibly causes a memory leak while running models on large-scale graphs. To alleviate this, we remove the corresponding term in the objective function, i.e., $\left\|\mathbf{P}^T\mathbf{P} - \mathbf{I}_d\right\|_F^2$ in Eq. (5.5), and reformulate it into a regularizer:

$$\mathcal{L}_{\text{Orth}} = \left\|\hat{\mathbf{P}}^{(K)}\hat{\mathbf{P}}^{(K)} - \mathbf{I}_d\right\|_2^2 \tag{5.9}$$

where $\hat{\mathbf{P}}^{(K)}$ is normalized from $\mathbf{P}^{(K)}$ such that each column of $\hat{\mathbf{P}}^{(K)}$ has zero mean and one l_2 norm, and $\mathbf{I}_d \in \mathbb{R}^{d\times d}$ is an identity matrix. Accordingly, we set $\eta_2 = 0$ in Eq. (5.7) and have another hyper-parameter λ_{Orth} called orthogonal regularization parameter. In training, $\mathcal{L}_{\text{Orth}}$ is penalized with the task loss as $\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{Orth}}\mathcal{L}_{\text{Orth}}$. We name this variant as DSF-x-R where $x \in \{\text{GPR}, \text{Bern}, \text{Jacobi}\}$ denotes the backbone GNN, while referring to the original one as DSF-x-I. The overall pipeline of the proposed DSF framework is detailed in Algorithm 4, and we also provide other important implementation details in the following.

GPR-GNN [54] as backbone. The authors experiment GPR-GNN with several initializing strategies on $\{\alpha_k\}_{k=0}^K$ and take the optimal one for the final evaluation. To take advantage of this, we adopt the same strategy to initialize our $\{\gamma_k\}_{k=0}^K$ before training. We call the resulted variants DSF-GPR-I and DSF-GPR-R.

BernNet [37] as backbone. As stated in the original paper, α_k is constrained to be

Algorithm 4 Framework of *diverse* spectral filtering

Input: Node set: $\mathcal{V}$, Laplacian matrix: $\hat{\mathbf{L}}$, Raw node content and positional features: $\mathbf{X} \in \mathbb{R}^{N \times f}, \mathbf{X}_p \in \mathbb{R}^{N \times f_p}$, Polynomial basis: $P_k(\cdot)$, Hyper-parameters: $K, \eta_1, \eta_2, \lambda_{\text{Orth}}$, Ground truth labels for training: $\{\mathbf{y}_i \in \mathbb{R}^C | \forall v_i \in \mathcal{V}_{\text{trn}}\}$, Activation function in Eq. (5.8): $\sigma_p(\cdot)$, and DSF-mode: $\phi \in \{\text{I}, \text{R}\}$.

Param: $\mathbf{W}_x \in \mathbb{R}^{f \times d}$, $\mathbf{b}_x \in \mathbb{R}^d$, $\mathbf{W}_p \in \mathbb{R}^{f_p \times d}$, $\mathbf{b}_p \in \mathbb{R}^d$, $\mathbf{W} \in \mathbb{R}^{d \times d}$, $\mathbf{W}_F \in \mathbb{R}^{d \times C}, \mathbf{b}_F \in \mathbb{R}^C$, $\{\mathbf{W}^{(k)} \in \mathbb{R}^d, \mathbf{b}^{(k)} \in \mathbb{R} | k = 0, 1, ..., K\}$, and $\{\gamma_k \in \mathbb{R} | \forall k = 0, 1..., K\}$.

Set $\eta_2 = 0$ if ϕ is R

$\mathbf{X}_i \leftarrow \text{ReLU}(\mathbf{W}_x^T \mathbf{X}_i + \mathbf{b}_x)$, $\mathbf{P}_i^{(0)} \leftarrow \text{Tanh}(\mathbf{W}_p^T \mathbf{X}_{p_i} + \mathbf{b}_b)$ for all $v_i \in \mathcal{V}$. // *Latent Space Projection.*

$\mathbf{X} \leftarrow \text{Dropout}(\mathbf{X})$, $\mathbf{P}^{(0)} \leftarrow \text{Dropout}(\mathbf{P}^{(0)})$. // *Enabled only for training.*

$\beta_{0,i} \leftarrow \gamma_0 \sigma_p(\mathbf{W}^{(0)^T} \mathbf{P}_i^{(0)} + \mathbf{b}^{(0)})$ for all $v_i \in \mathcal{V}$. // *Initialization.*

$\mathbf{Z}^{(0)} \leftarrow \text{diag}(\beta_{0,1}, \beta_{0,2}, ..., \beta_{0,N}) P_0(\hat{\mathbf{L}}) \mathbf{X}$.

for $k = 1, 2, ..., K$ **do**

  $\mathbf{P}^{(k)} \leftarrow \mathbf{P}^{(k-1)}$ using Eq. (5.7) with $\eta_1, \eta_2, \mathbf{W}$. // *Update node positional features.*

  $\beta_{k,i} \leftarrow \gamma_k \sigma_p(\mathbf{W}^{(k)^T} \mathbf{P}_i^{(k)} + \mathbf{b}^{(k)})$ for all $v_i \in \mathcal{V}$. // *Update node hidden states.*

  $\mathbf{Z}^{(k)} \leftarrow \mathbf{Z}^{(k-1)} + \text{diag}(\beta_{k,1}, \beta_{k,2}, ..., \beta_{k,N}) P_k(\hat{\mathbf{L}}) \mathbf{X}$.

end for

$\hat{\mathbf{y}}_i = \text{softmax}(\mathbf{W}_F^T \mathbf{Z}_{[i,:]}^{(K)} + \mathbf{b}_F), \forall v_i \in \mathcal{V}$. // *Prediction.*

$\mathcal{L}_{\text{task}} = -\frac{1}{|\mathcal{V}_{\text{trn}}|} \sum_{v_i \in \mathcal{V}_{\text{trn}}} \mathbf{y}_i^T \log(\hat{\mathbf{y}}_i)$. // *Training.*

if ϕ is R **then**

  Minimize $\mathcal{L}_{\text{task}} + \lambda_{\text{Orth}} \mathcal{L}_{\text{Orth}}$ with $\mathcal{L}_{\text{Orth}}$ computed in Eq. (5.9).

else

  Minimize $\mathcal{L}_{\text{task}}$.

end if

non-negative. To follow up, we apply the same limit to our $\beta_{k,i} = \gamma_k \theta_{k,i}$ by making $\gamma_i \leftarrow \text{ReLU}(\gamma_i)$ and taking σ_p as a sigmoid function in Eq 5.8 to restrict $\theta_{k,i}$ within $(0, 1)$. The resulted models are named as DSF-Bern-I and DSF-Bern-R.

JacobiConv [74] as backbone. The authors leverage a technique called PCD, which decomposes the filter weight into multiple coefficients, such as $\alpha_k = \pi_k \prod_{s=1}^k \rho_s$. To deploy our DSF framework over JacobiConv, we make $\pi_k = \gamma_k$, and transform ρ_s into $\rho_{s,i}$ which is learned using Eq. (5.8). The produced variants are finally referred to as DSF-Jacobi-I and DSF-Jacobi-R.

Initialization on IPE. The choice of initializing node positional embeddings $\mathbf{X}_p$ is important, which usually requires to be permutation-invariant and distance-sensitive. In this work, we leverage two popular and efficient methods. The first one is widely

used and called Laplacian Positional Encoding [192] (LapPE). It basically takes the decomposed eigenvectors $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_N\}$, and each node v_i is assigned with $\mathbf{P}_i^{\text{Lap}} = [\mathbf{u}_{1,i}, \mathbf{u}_{2,i}, ..., \mathbf{u}_{f_p,i}]^T \in \mathbb{R}^{f_p}$, where $f_p \ll N$ is the predefined feature number. The other approach was recently proposed based on the random walk diffusion process, named RWPE [189]. It aims to moderate the sign ambiguity issue in LapPE, and is formulated as $\mathbf{P}_i^{\text{RW}} = [\mathbf{RW}_{i,i}^1, \mathbf{RW}_{i,i}^2, ..., \mathbf{RW}_{i,i}^{d_f}]^T \in \mathbb{R}^{f_p}$, where $\mathbf{RW} = \mathbf{AD}^{-1}$. To increase model capacity and efficiency, we map $\mathbf{X}_p$ into a latent space with dimension d (see line 2 in Algorithm 4), and node' positional features are iteratively updated along with their hidden states (see the lines 11-17). Similar approaches can be found in [189].

Remarks. The proposed DSF framework extends the existing spectral GNNs as $\sum_{k=0}^{K} \alpha_k P_k(\hat{\mathbf{L}})\mathbf{X} \rightarrow \sum_{k=0}^{K} \gamma_k \text{diag}(\theta_{k,1}, \theta_{k,2}, ..., \theta_{k,N}) P_k(\hat{\mathbf{L}})\mathbf{X}$ where a functional space made up of diverse filters, $\{g_i(\cdot) = \sum_{k=0}^{K} \gamma_k \theta_{k,i} P_k(\cdot) | \forall v_i \in \mathcal{V}\}$, is derived to enable node-wise learning. Specifically, the underlying graph region of individual node is mined with a different filter function. Existing spectral GNN models mostly learn with one filter function, and thereby can be strengthened with our DSF framework. Besides, in comparison with the advocated interpretability in BernNet [37], our DSF framework is able to offer better interpretability by further differentiating micro graph structures with learned diverse filters, as we demonstrate in Section 5.7.2.

5.5.2 Computational Complexity

Since our framework requires node-wise computations with positional features, compared to its underlying GNNs, the model complexity is increased by $O(N(f_p d + 2Kd + 2d + K +$

Table 5.1: Average running time per epoch (ms)/average total running time (s). Although DSF-GPR-I is less efficient on large networks, DSF-GPR-R, (our major model) can reduce it by more than 75% on average (though reasonably slower than GPR-GNN).

Datasets	**Small-scale**	**Large-scale**	**Average**
GPR-GNN	1.10/2.24	0.98/5.01	1.08/2.74
DSF-GPR-I	5.96/12.19	40.34/131.77	12.21/33.93
DSF-GPR-R	2.49/6.29	3.02/14.48	2.59/7.78

$1)+2|\mathcal{E}|Kd+2N^2Kd)$ in our DSF-x-I. By the regularization term $\mathcal{L}_{\text{Orth}}$, we further mange to reduce it by $\mathcal{O}(N^2Kd)$, and introduce our major model named DSF-x-R. The average running time is reported on Table 5.1. Despite the slightly higher computational overhead, we argue that DSF framework works still reasonably efficient in practice, especially considering the remarkable performance gains and the enhanced model interpretability (see Section 5.6.2 and 5.7.2).

5.6 Experiments

In this section, we design experiments to answer the following research questions: (**RQ1**) How effective is our DSF framework in improving state-of-the-art spectral GNNs for node classification? (**RQ2**) Is there a negative impact on accuracy when each node is parameterized by a separate set of trainable weights? If so, would the proposed strategies, i.e., Position-aware Filter Weights and Local and Global Weight Decomposition (LGWD) take effect in alleviating this? and (**RQ3**) Could the proposed DSF framework indeed learn diverse and interpretable filters capturing both the common graph structure and regional heterogeneity?

5.6.1 Experimental Setup

Datasets. We examine models over 11 real-world datasets from various domains including 6 heterophilic graphs as Chameleon, Squirrel [172], Wisconsin, Cornell, Texas [60] (webpage networks), and Twitch-DE [5, 172] (social network), as well as 5 homophilic graphs, i.e., Cora, Citeseer, Pubmed [140] (citation networks), Computers, and Photo [194, 195] (the Amazon co-purchase graphs). Detailed statistics are provided in Table 6.1, and in certain compact sections, we use four-letter abbreviations for dataset names. We divide each dataset into 60%/20%/20% for training/validation/testing by following [37, 54, 74], and create 10 random splits for evaluation.

Baselines. To verify the effectiveness of the proposed DSF framework, we implement it upon three state-of-the-art spectral GNNs with trainable polynomial filters, i.e.,

Table 5.2: Statistics of real-world datasets, where ⋆ denotes large-scale graphs. Both $\mathcal{H}$ [4] and $\mathcal{H}_{\text{class}}$ [5] (considering class-imbalance problem) measure graph homophily ratio from 0 to 1. Albeit the relative high value given by $\mathcal{H} = 0.63$, Twitch-DE is essentially a heterophlic graph with class-imbalanced issue, as suggested by $\mathcal{H}_{\text{class}} = 0.14$.

Dataset	# Nodes	# Edges	# Features	# Classes	$\mathcal{H}$	$\mathcal{H}_{\text{class}}$
Chameleon	2,227	36,101	2,325	5	0.23	0.06
Squirrel	5,201	217,073	2,089	5	0.22	0.03
Wisconsin	251	499	1,703	5	0.21	0.09
Cornell	183	295	1,703	5	0.30	0.05
Texas	183	309	1,703	5	0.11	0.00
Twitch-DE	9,498	153,138	2,545	2	0.63	0.14
Cora	2,708	5,429	1,433	7	0.81	0.77
Citeseer	3,327	4,732	3,703	6	0.74	0.63
Pubmed⋆	19,717	44,338	500	3	0.80	0.66
Computers⋆	13,752	245,861	767	10	0.78	0.70
Photo	7,650	119,081	745	8	0.83	0.77

GPR-GNN [54], BernNet [37], and JacobiConv [74]. Therefore, we have six variants with names formatted as DSF-x-ϕ for $x \in \{\text{GPR}, \text{Bern}, \text{Jacobi}\}$ and $\phi \in \{\text{I}, \text{R}\}$. For a more comprehensive comparison, we also consider another 8 baseline GNNs including GCN [35], GAT [53], ChebNet [55], APPNP [36], GNN-LF/HF [67], FAGCN [61], and PA-GNN [73].

Setup. We fix the number of hidden features $d = 64$ for all models, and set the polynomial order $K = 10$ to follow [37, 54, 74]. For each dataset, we tune the hyper-parameters of all models, including baselines with their specified parameter ranges, on the validation split using Optuna [141] for 200 trails. With the best hyper-parameters, we train models in 1,000 epochs using early-stopping strategy and a patience of 100 epochs. The average performance over 100 runs (10 runs × 10 splits) are reported.

5.6.2 Overall Evaluation

To answer RQ1, we report the average node classification accuracies with a 95% confidence interval on both heterophilic and homophilic graphs. From Table 5.3, we have the following observations: **1)** Spectral GNNs with trainable filters generally yield better

Table 5.3: Node classification accuracies (%) ± 95% confidence interval over 100 runs.

Datasets	**Heterophilic Graphs**						**Homophilic Graphs**				
	Cham.	**Squi.**	**Wisc.**	**Corn.**	**Texas**	**Twit.**	**Cora**	**Cite.**	**Pubm.**	**Comp.**	**Photo**
GCN	67.22±0.43	54.21±0.41	59.45±0.72	52.76±1.17	61.66±0.71	73.94±0.15	88.13±0.25	77.00±0.27	89.07±0.11	91.06±0.12	93.99±0.12
GAT	67.72±0.41	52.26±0.58	57.94±0.89	50.20±0.93	55.37±1.10	73.00±0.15	88.47±0.22	77.23±0.27	88.30±0.11	91.69±0.11	94.55±0.11
ChebNet	64.85±0.44	48.14±0.33	80.93±0.72	77.98±1.00	75.83±1.20	73.73±0.14	87.64±0.21	76.93±0.24	89.91±0.11	91.65±0.12	95.27±0.07
APPNP	53.66±0.33	36.08±0.36	81.23±0.64	81.29±0.78	79.42±1.05	72.65±0.11	88.70±0.21	77.77±0.24	89.93±0.09	91.62±0.10	94.92±0.09
GNN-LF	54.29±0.36	36.87±0.33	59.85±0.60	62.90±0.98	61.88±0.95	73.03±0.13	88.90±0.25	77.35±0.29	88.89±0.10	91.12±0.11	95.13±0.08
GNN-HF	55.22±0.42	35.45±0.30	68.17±0.72	72.98±1.02	66.66±1.34	71.92±0.13	89.01±0.19	77.74±0.23	89.53±0.10	90.73±0.10	95.26±0.09
FAGCN	68.38±0.51	50.08±0.60	82.11±0.85	79.00±0.93	81.00±0.95	74.15±0.13	88.82±0.20	77.65±0.29	90.13±0.11	91.90±0.11	95.25±0.10
GPR-GNN	69.01±0.50	55.39±0.33	82.72±0.85	80.81±0.78	81.66±1.02	74.07±0.18	89.03±0.20	77.63±0.28	90.10±0.44	92.34±0.13	95.34±0.09
DSF-GPR-I	71.18±0.52	57.08±0.29	**87.64**±0.79	84.76±0.90	85.44±1.05	74.58±0.16	**89.64**±0.20	78.03±0.26	90.26±0.08	92.49±0.12	95.64±0.07
DSF-GPR-R	**71.64**±0.55	**58.44**±0.30	87.43±0.74	**84.93**±0.90	**85.56**±0.93	**74.81**±0.14	89.63±0.17	**78.22**±0.29	**90.51**±0.07	**92.80**±0.12	**95.73**±0.08
Improv.	2.63%	3.05%	4.92%	4.12%	3.9%	0.74%	0.61%	0.59%	0.41%	0.46%	0.39%
PA-GNN*	0.66%	1.28%	–	–	–	–	-0.09%	-0.74%	-0.03%	1.03%	0.02%
BernNet	70.59±0.42	56.63±0.32	85.00±0.94	82.10±0.95	82.20±0.98	74.45±0.15	88.72±0.23	77.52±0.29	90.21±0.46	92.57±0.10	95.42±0.08
DSF-Bern-I	72.95±0.53	59.45±0.32	**88.23**±0.81	**85.07**±0.93	**84.59**±1.07	74.96±0.15	89.05±0.22	**78.32**±0.27	90.40±0.10	92.76±0.10	95.73±0.07
DSF-Bern-R	**73.60**±0.53	**59.99**±0.30	88.02±0.91	84.29±0.93	84.42±1.00	**75.00**±0.15	**89.10**±0.22	78.27±0.26	**90.52**±0.10	**92.84**±0.10	**95.79**±0.06
Improv.	3.01%	3.36%	3.23%	2.97%	2.39%	0.55%	0.38%	0.80%	0.31%	0.27%	0.37%
JacobiConv	73.71±0.42	57.22±0.24	83.21±0.68	82.34±0.88	82.42±0.90	74.34±0.12	89.24±0.19	77.81±0.29	89.50±0.47	92.26±0.10	95.62±0.06
DSF-Jacobi-I	74.88±0.39	58.26±0.26	85.34±0.74	**84.54**±0.81	83.68±1.12	74.65±0.13	89.54±0.19	78.18±0.26	89.78±0.09	92.38±0.11	**95.76**±0.07
DSF-Jacobi-R	**75.00**±0.38	**59.23**±0.27	**86.13**±0.70	84.39±0.88	**84.46**±0.81	**74.75**±0.15	**89.66**±0.19	**78.23**±0.25	**90.07**±0.10	**92.44**±0.11	95.75±0.08
Improv.	1.29%	2.01%	2.92%	2.20%	2.04%	0.41%	0.42%	0.42%	0.41%	0.18%	0.14%

* This row lists the relative improvements of PA-GNN [73] upon GPR-GNN [54] based on the results obtained from its paper, where – denotes values not provided.

classification results than other baseline models. This is because conventional GNNs typically fail to deal with complex linking patterns, e.g., in heterophilic graphs, using their fixed frequency response filters. Contrastively, GPR-GNN, BernNet, and JacobiConv can simulate different types of filters to learn from both assortative and disassortative label patterns. FAGCN is able to capture both low- and high-frequency information but is limited as they can only model pairwise node relationship. **2)** The proposed DSF framework consistently produces performance boost over its underlying models, especially on heterophilic graphs with the maximal improvement up to 4.92%. This can be mainly explained by the diverse characteristics inherent in their local graph patterns, as shown in Figure 5.1. By comparison, the existing spectral GNN models assume the *homogeneous* spectral filtering, and neglect regional *heterogeneity* at different graph localities. **3)** For homophilic graphs such as Cora, Citeseer, and Pubmed, our framework achieves only a slight improvement. This limited enhancement stems from the inherent similarity and uniform distribution of local structural patterns throughout these graphs, as evidenced by the concentrated histogram in Figure 5.4. Such structural uniformity lends itself well to classic *homogeneous* spectral filtering techniques. The modest gains in classification accuracy by DSF primarily arise from addressing sudden structural changes at graph boundaries, for instance, between distinct community zones or social groups. **4)** We conduct comparisons with PA-GNN [73] which also tries to learn node-specific parameter offsets. As no codes are publicly available for PA-GNN, we simply compute the relative improvements upon its base model GPR-GNN by copying the values from their paper (listed in the row of PA-GNN [73]*). In general, PA-GNN shows marginal or even negative performance gains. We conjecture that PA-GNN might encode different sources of information for predicting the offsets with small value constraint, thus limiting their performance. **5)** Interestingly, it is noted our variant DSF-x-R not only decreases the model complexity of DSF-x-I but also achieves higher performance gains on average. This is partially because DSF-x-I minimizes the orthogonal penalty, i.e., the last term in Eq. (5.5), mainly by means of the iterative aggregation on a non-sparse graph computed by $\mathbf{P}^{(k)}\mathbf{W}\mathbf{P}^{(k)^T}$. Despite the theoretical convergence, aggregating features on dense graph

is prone to mistakenly preserve noises and cause model overfitting. On the other hand, DSF-x-R with the regularization loss $\mathcal{L}_{\text{Orth}}$ offers a more flexible and accurate control, provides extra supervisory signals directly operated on model parameters, and could also benefit from the advanced optimization technique such as Adam [169] algorithm.

5.7 Analysis and Discussion

5.7.1 Ablation Study

This subsection aims to validate our designs through ablation study. We earlier argue that it is inappropriate to directly parameterize each node a separate set of trainable filter weights. To provide empirical evidences, we first experiment with our DSF framework in node classification tasks while ablating the module of iterative positional encoding (IPE). That is to directly make $N \times (K+1)$ filter weights w.r.t. nodes to be trained as model parameters. We then report the downgraded model performance compared to the underlying models in Table 5.4, where six datasets are experimented for illustration. As observed, learning without IPE leads to a clear accuracy drop, notably on networks Chameleon and Squirrel with complex connecting patterns and relative a large number of nodes. This confirms our early conjecture as well as the importance of the proposed IPE strategy. In this work, we also constrain the channel orthogonality while encoding positional features, and introduce a technique called Local and Global Weight Decomposition (LGWD). To examine their effectiveness, we conduct comprehensive ablation study over six datasets in node classification. For simplicity,

Table 5.4: Reduced classification accuracies (%) of our DSF framework compared to base models while learning without IPE.

Datasets	**Cham.**	**Squi.**	**Wisc.**	**Corn.**	**Texas**	**Photo**
DSF-GPR w/o IPE	22.62	22.55	5.81	11.20	11.10	1.51
DSF-Bern w/o IPE	17.47	18.65	4.72	6.25	6.54	3.59
DSF-Jacobi w/o IPE	24.64	26.93	1.10	3.10	5.88	1.53
Average Reduction	**21.58**	**22.71**	**3.88**	**6.85**	**7.84**	**2.21**

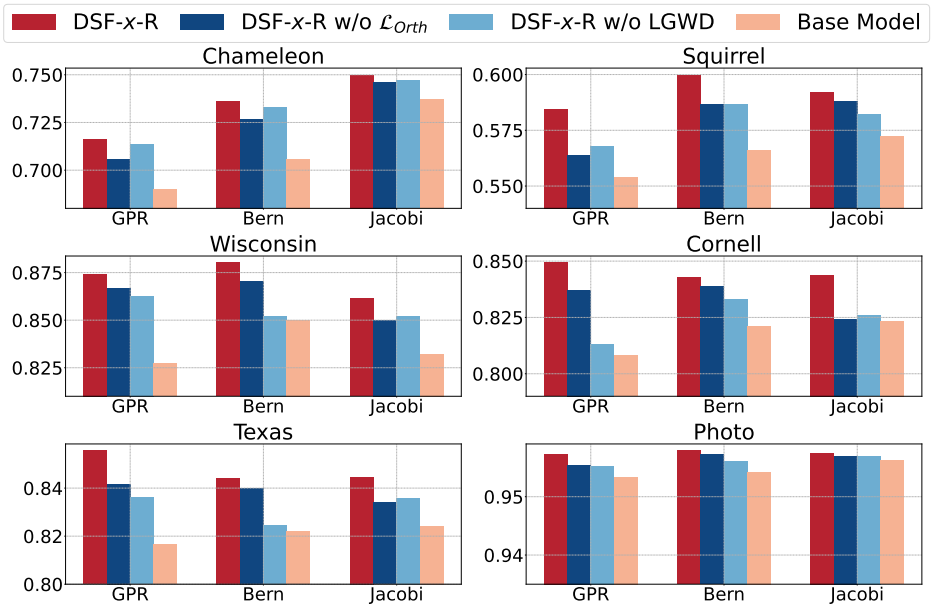
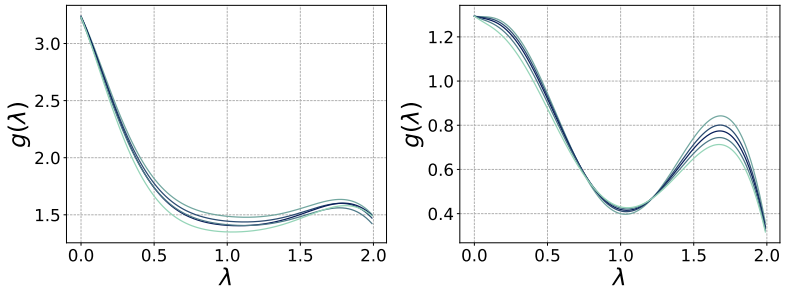


Figure 5.5: Ablation study of DSF framework on six datasets with our variants DSF-x-R for all $x \in \{\text{GPR}, \text{Bern}, \text{Jacobi}\}$ as an example.

we take DSF-x-R, one variant of our framework, as an example. Similar results can be obtained on the other variants. From Figure 5.5, two conclusions can be drawn. First, removing either $\mathcal{L}_{\text{Orth}}$ or LGWD from our framework causes an evident performance downgrade, validating the usefulness of these two developed techniques. Second, the ablated variants still outperform their underlying models. This further underpins the advantages offered by learning diverse filters with awareness of positional information.

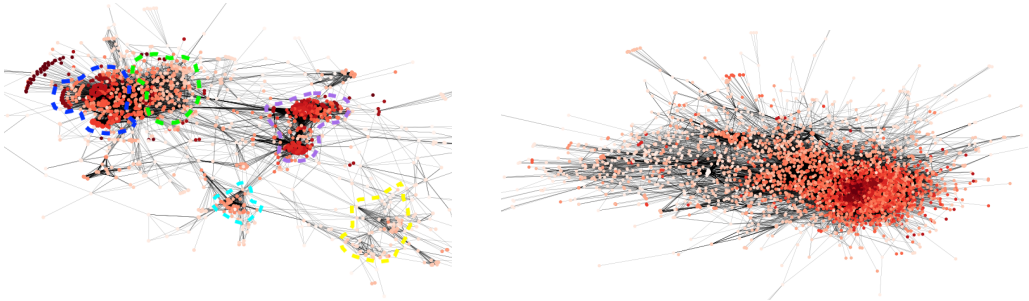
5.7.2 Analysis on Diverse Filters

We now answer RQ3 by first plotting the diverse filter functions learned by our DSF with BernNet as the illustrative base model. Without loss of generality, we cluster the node-specific filter weights, i.e., $\{[\beta_{0,i}, \beta_{1,i}, ..., \beta_{K,i}]^T | \forall v_i \in \mathcal{V}\}$, into five groups with k-means algorithm [196], and only plot the filters w.r.t. the representative centroids for better visualization. From Figure 5.2 on heterophilc graphs, we observe a group of function curves showing similar overall shapes but different local aspects. This implies that the proposed DSF framework is able to grasp both conformal and disparate regional



(a) Citeseer (b) Photo

Figure 5.6: Diverse filters on homophilic graphs, which are learned to be similar due to the intrinsic assortative linking patterns distributed uniformly on these networks. Our DSF presents one general framework which can be adaptive to different types of networks.



(a) Chameleon (b) Squirrel

Figure 5.7: Visualization of node-specific filter weights on Chameleon and Squirrel datasets, where a few border nodes are cropped away for better picturing.

information on the graph.

In addition, we also draw the diverse filters learned from homophilic graphs including Citeseer and Photo. These graph networks have assortative mixing patterns with homogeneous local structures. The learned filter functions produce almost identical curves fluctuating within a reasonable interval in Figure 5.6. It further shows our DSF framework could work on different types of graphs. On the other hand, we present t-SNE [197] visualization of the node-specific filter weights. The color likeness reflects the corresponding similarity. From Figure 5.2f, disparate regional patterns can be distinguished, and far-reaching nodes with conformal local subgraphs still learn similar filter weights. Besides, we notice the node in graph center displays a salient

white color, obviously divergent from its neighborhood. This is because such vertex possesses the unique local context characterized by the densest graph neighborhood, and thereby deserves a special treatment. This phenomenon shows the flexibility of our DSF framework in dealing with complex or even unusual cases, instead of learning some strict relationships, e.g., nearby nodes ought to possess similar local structures (similar filter weights) and otherwise. We also present visualizations of larger graphs in Figure 5.7, where the regional distinctions, captured in Chameleon dataset, are annotated with irregular circles in different colors. For Squirrel dataset, we can see a gradual shift on the color depth of nodes from graph center to the border, which coincides with our conclusions in the main text about our DSF framework capturing regional heterogeneity. These analytical results demonstrate the strong interpretability of our DSF framework.

5.7.3 Limitations and Future work

In this subsection, we reflect on the limitations of our proposed method. While our model has shown impressive performance across multiple benchmarks, it is essential to address certain constraints that could hinder its broader applicability. Below, we identify key areas where the model could be improved and explore potential avenues for future research.

- The learning of node-specific filtering weights in our model largely depends on nodes' position, an assumption valid in conventional networks. However, the effectiveness of this inductive bias is less certain in less common networks, such as those encountered in biomedical fields like molecular structures. Investigating whether this assumption holds true in more complex and varied scenarios, and extending the model to accommodate such complexities, presents an exciting research opportunity.

- Our current methodology requires the learning of node-wise parameters, leading to high computational complexity, particularly in large-scale graphs. Balancing the need to accurately model the graph's local variability with maintaining model

efficiency is crucial. Developing strategies to reduce the computational demand without compromising the model's ability to capture local graph dynamics is a promising direction for future work.

5.8 Conclusion

This work focuses on learning GNNs on complex graphs with regional *heterogeneity* from the spectral perspective. We show that most existing spectral GNNs implicitly assume invariance between local networking patterns, and are restricted in the *homogeneous* spectral filtering, thus limiting their performance. To this end, we propose a novel *diverse* spectral filtering (DSF) framework generalizing spectral GNNs to better exploit rich and diverse local graph information. Both theoretical and empirical investigations validate the effectiveness of our DSF framework and its enhanced interpretability by learning diverse filters.

Chapter 6

Graph Neural Networks with Spatially Adaptive Filtering

The preceding chapter focused on enhancing spectral GNNs within their spectral domain. This chapter, however, embarks on a cross-domain exploration, revisiting spectral GNNs from a spatial perspective. Despite their grounding in the spectral domain, our investigation reveals that these GNNs fundamentally modify spatial relationships within the graph, introducing non-locality and signed edge weights to delineate global label relationships among nodes. This insight challenges the conventional reliance of fixed-order polynomials in spectral GNNs, which typically neglect the richness of global information. Motivated by these findings, we propose the Spatially Adaptive Filtering (SAF) framework. SAF capitalizes on our cross-domain discoveries to facilitate non-local aggregation, adeptly capturing long-range dependencies and excelling particularly in heterophilic graph contexts. This work has been submitted to a recent academic conference.

6.1 Introduction

Graph Neural Networks (GNNs) have shown remarkable abilities to uncover the intricate dependencies within graph-structured data, and achieved tremendous success in graph

machine learning [41, 45, 198]. Spectral GNNs are a class of GNNs rooted in spectral graph theory [21, 24], implementing graph convolutions via spectral filters [35, 55]. Whilst various spectral filtering strategies [37, 54, 74, 75, 77, 79, 80, 199, 200] have been proposed for spectral GNNs, their practical implementations invariably resort to approximating graph filters with fixed-order polynomials for computational efficiency [74, 75]. This truncated approach essentially relies on the direct extraction of spatial features from the local regions of nodes. As such, the spatial domain of a graph, albeit loosely connected to spectral GNNs in theory, still plays a crucial role in effectively learning node representations.

However, there is a notable lack of research examining spectral GNNs from the spatial perspective. Though recent studies analyze both spectral and spatial GNNs to elucidate their similarities in model formulations [40, 81, 201], outcomes [67, 68], and expressiveness [59, 74, 82, 98], they ignore exploring the interpretability that could arise mutually from the other domain. Specifically, whilst most spectral GNNs have well explained their learned graph filters in the spectral domain [37, 74, 75, 77], understandings from the spatial viewpoint are merely limited to fusing multi-scale graph information [202]; this unfortunately lacks a deeper level of interpretability in the vertex domain. Therefore, a natural question arises: *what information is essentially encoded by spectral GNNs in the spatial domain?*

In this work, we attempt to answer this question by exploring the connection between spectral filtering and spatial aggregation. The former is the key component in spectral GNNs, while the latter is closely associated with spatial GNNs utilizing recursive neighborhood aggregation. In existing GNN frameworks, these two approaches rarely interact each other at the risk of domain information trade-offs due to uncertainty principles [203–205]. Recognizing the spatial significance in spectral filtering, [37] have recently considered non-negative constraints as part of a generalized graph optimization problem. Notably, however, spatial aggregation meanwhile resembles the optimizing trajectory of the same optimization problem through iterative steps, which may be easily overlooked. Inspired by such observation, we examine, for the first time, the theoretical

interaction between spectral filtering and spatial aggregation. This exploration has led us to uncover an intriguing theoretical interplay, *i.e.*, spectral filtering implicitly modifies the original graph, transforming it into a new one that explicitly functions as a computation graph for spatial aggregation. Delving deeper, we discover that the adapted new graph enjoys some desirable properties, enabling a direct link among nodes that originally require multiple hops to do so, thereby exhibiting nice non-locality. Moreover, we find that the new graph edges allow signed weights, which turns out capable of distinguishing between label agreement and disagreement of the connected nodes.

Overall, these findings underscore the interpretable role and significance of spectral GNNs in the spatial domain, inspiring us to rethink graph spectral filters beyond the fixed-order polynomials, which limit the effective propagation range of models and hinder their ability to capture long-range dependencies. Concretely, we propose a novel Spatially Adaptive Filtering (SAF) framework, for fully exploring spectral GNNs in the spatial domain. SAF leverages the adapted new graph by spectral filtering for auxiliary spatial aggregation and allows individual nodes to flexibly balance between spectral and spatial features. By performing non-local aggregation with signed edge weights, our SAF adeptly overcomes the limitations of truncated polynomials, enabling the model to capture both node similarity and dissimilarity at a global scale. As a benefit, it can mitigate persistent deficiencies of GNNs regarding long-range dependencies and graph heterophily. The contributions are summarized as follows:

- Our investigation into spectral GNNs in the spatial domain reveals that graph spectral filtering fundamentally alters the original graph, imbuing it with non-locality and signed edge weights that discern label consistency among nodes.
- We propose Spatially Adaptive Filtering (SAF) framework, a paradigm-shifting approach to spectral GNNs that jointly leverages graph learning in both spatial and spectral domains, making it a powerful tool for capturing long-range dependencies and handling graph heterophily.
- Extensive experiments over 13 node classification benchmarks exhibit notable

improvements of up to 15.37%, and show that SAF beats the best-performing spectral GNNs on average.

6.2 Related Works

Graph Neural Networks. GNNs can be broadly divided into spatial-based and spectral-based models. Spatial GNNs leverage the spatial connections among nodes to perform message passing, also known as spatial aggregation [20, 51]. For a thorough review, we direct readers to the works [45, 143]. Spectral GNNs leverage the graph's spectral domain for convolution or, alternatively, spectral filtering [35, 55, 181]. Prevailing approaches focus on developing polynomial graph filters, by either learning polynomial coefficients, such as GPR-GNN [54], BernNet [37], ChebNetII [75], and JacobiConv [74], or concurrently optimizing the polynomial basis for better real-world adaption, as seen in models like LON-GNN [79] and OptBasisGNN [80]. Diverging from this trend, ARMA [66] employs rational filter functions while still approximating them with polynomials. Although these methods are theoretically grounded in the spectral domain, their practical reliance on polynomial approximation hints at a profound linkage to the spatial domain. However, the spatial-domain interpretation of spectral GNNs is rarely examined. To this end, we delve into in this work the intrinsic information spectral GNNs convey within the spatial context.

Unified Viewpoints for GNNs. Several works have explored the nuances between spatial and spectral GNNs. Early studies by [201] and [40] examined their similarities in model formulations. [59] proved their spatial GNN's anti-oversmoothing ability via spectral analysis. [68] and [67] utilized the graph signal denoising problem to integrate both GNN types. [98] and [74] further explored their expressiveness equivalence. Recently, [82] have highlighted the feature space constraints of both spatial and spectral GNNs, while [81] attempted to combine them via a residual connection module. Though these studies effectively bridge spectral and spatial GNNs, they remain focused on congruencies. Unlike them, our work represents the first endeavor to delve into the

interpretability of spectral GNNs in the spatial domain, emphasizing the theoretical synergy between spectral filtering and spatial aggregation. The empirical success of our proposed method (as compared to unified GNNs in Tables 6.2 and 6.3), stemming from this in-depth analysis, further underscores our practical contributions to the literature.

Long-range Dependencies. While substantial efforts have been directed towards capturing long-range dependencies in spatial GNNs [36, 59, 60, 62, 65, 71], the exploration of the same challenge in spectral GNNs remains under-studied. To fill this gap, we propose a SAF framework, which emerges as a valuable consequence of analyzing spectral GNNs in the spatial domain, enhancing their long-range dependency capture. Concurrently, [78] also introduced Specformer to addresses long-range dependencies for spectral GNNs, using a Transformer based set-to-set spectral filter. However, it lacks spatial-domain interpretability and introduce more trainable parameters. In contrast, our approach creates a non-local new graph without learning additional parameters, simultaneously elucidating the interpretive implications of spectral GNNs in the spatial domain.

Graph Heterophily. Graph heterophily [4, 60], where different labeled nodes connect, challenges GNNs operating under the homophily assumption [145]. Although many GNNs have been crafted to manage heterophilic connections [54, 61, 63, 76, 206, 207], our proposed SAF stands out in addressing graph heterophily. Specifically, SAF innovatively conducts an auxiliary non-local aggregation using signed edge weights, emphasizing both intra-class similarity and inter-class difference on a global scale. One should note that a recent work [72] bear some resemblance to ours, introducing GloGNN and GloGNN++ to capture global homophily beyond immediate neighborhoods by learning signed edge weights. However, their approach, albeit demonstrating a grouping effect [101], restricts the optimization objective into a K-hop neighborhood, focusing on similar local structural information. Conversely, our SAF framework ensures the theoretical properties of non-local learning (as proved in Section 6.4.2), while effectively modeling label relationships. This capability directly benefits downstream classification tasks, offering a notable superiority on real-world applications.

Eigendecomposition Eigendecomposition breaks down a matrix into its eigenvalues and eigenvectors, offering insights into matrix properties, especially for the graph Laplacian. Despite computational demands, this technique has attracted surging interest in the graph learning community due to its theoretical richness, and it can be practically expedited for larger graphs using Lanczos [208] and Sparse Generalized Eigenvalue [209] algorithms. Recent advancements also underscore its value in various applications such as graph positional encoding [192, 210], spectral graph convolution [202], graph domain adaptation [211], and graph robustness [212]. For example, Laplacian eigenvectors have been widely used in identifying global position of nodes in the graph [188], particularly in recent popular graph transformers [213–215], enhancing their expressiveness. Innovations like SignNet, BasisNet [216], and Sign Equivariant [217] have further optimized the processing of these eigenvectors. When exploring the expressive power of GNN models, Specformer [78] employs eigendecomposition for learning set-to-set graph filters, TEDGCN [206] leverages it for adaptive weighting of eigengraphs, and FE-GNN [82] taps into singular value decomposition (SVD) for graph feature expansion. In line with these developments, our method, SAF, also utilizes eigendecomposition to explicitly create a new graph, enabling efficient non-local aggregation with signed weights to tackle long-range dependency and graph heterophily.

6.3 Preliminaries

In this section, we focus on the mathematical underpinnings of spectral filtering and spatial aggregation, essential for understanding the discussions on both spectral and spatial GNNs that follow. This shift from the broader background presented earlier to more specialized topics allows us to explore the foundational techniques at the core of GNN operations.

Spectral Filtering. Spectral filtering is essential in spectral GNNs. It selectively shrinks or amplifies the Fourier coefficients of node features [55] and usually take the

form as

$$\mathbf{Z} = g_\psi(\hat{\mathbf{L}})\mathbf{X} = \mathbf{U}g_\psi(\mathbf{\Lambda})\mathbf{U}^T\mathbf{X}. \tag{6.1}$$

Here, $g_\psi : [0, 2] \to \mathbb{R}$ defines a graph filter function, which are often approximated by a K-order polynomial in practice. Specifically, we have $g_\psi(\lambda) = \sum_{k=0}^{K} \psi_k P_k(\lambda) = \sum_{k=0}^{K} \omega_k \lambda^k$ where $P_k : [0, 2] \to \mathbb{R}$ refers to a polynomial basis and both ψ_k and ω_k denote the polynomial coefficient.

Spatial Aggregation. Spatial Aggregation is a central component of spatial GNNs, facilitating the propagation of node information along graph edges and its subsequent aggregation within node neighborhood. To provide a more formal illustration of spatial aggregation, let's consider the widely adopted GNN model, APPNP [36]. This model begins by applying a feature transformation, given by $\mathbf{Z}^{(0)} = f(\mathbf{X})$. The propagation then proceeds as:

$$\mathbf{Z}^{(k)} = (1 - \eta)\mathbf{Z}^{(0)} + \eta\hat{\tilde{\mathbf{A}}}\mathbf{Z}^{(k-1)}, \quad k = 1, 2, \cdots, K, \tag{6.2}$$

where $\hat{\tilde{\mathbf{A}}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and η refers to the update rate.

6.4 Rethinking Spectral GNNs from the Spatial Perspective

In this section, we provide both theoretical and empirical analyses to examine spectral GNNs from the spatial perspective and answer the question, *i.e.*, what information is essentially encoded by spectral GNNs in the spatial domain?

6.4.1 Interplay of Spectral and Spatial Domains through the Lens of Graph Optimization

The graph signal denoising problem [24] was initially leveraged in [67, 68] as a means to interpret GNNs with smoothness assumption, which yet does not always hold in certain real-world graph scenarios such as heterophily [4]. Without loss of generality, in this

work, we consider a more generalized graph optimization problem[1]

$$\arg\min_{\mathbf{Z}} \ \mathcal{L} = \alpha\|\mathbf{X}-\mathbf{Z}\|_F^2 + (1-\alpha)\cdot \mathrm{tr}(\mathbf{Z}^T\gamma_\theta(\hat{\mathbf{L}})\mathbf{Z}) \tag{6.3}$$

where $\mathbf{Z}\in\mathbb{R}^{N\times d}$ refers to node representations, $\gamma_\theta(\hat{\mathbf{L}})$ determines the rate of propagation [218] by operating on the graph spectrum, i.e., $\gamma_\theta(\hat{\mathbf{L}}) = \mathbf{U}\gamma_\theta(\mathbf{\Lambda})\mathbf{U}^T$, and $\alpha\in(0,1)$ is a trade-off coefficient. In case of setting $\gamma_\theta(\hat{\mathbf{L}}) = \hat{\mathbf{L}}$, Eq. (6.3) turns into the well-known graph signal denoising problem. To ensure the convexity of the objective in Eq. (6.3), a positive semi-definite constraint is imposed on $\gamma_\theta(\hat{\mathbf{L}})$, *i.e.*, $\gamma_\theta(\lambda)\geq 0$ for $\lambda\in[0,2]$. Then, one can address this minimization problem through either closed-form or iterative solutions.

Closed-form Solution. The closed-form solution can be obtained by setting the derivative of the objective function $\mathcal{L}$ to 0, i.e., $\frac{\partial\mathcal{L}}{\partial\mathbf{Z}} = 2\alpha(\mathbf{Z}-\mathbf{X}) + 2(1-\alpha)\gamma_\theta(\hat{\mathbf{L}})\mathbf{Z} = 0$. Let $g_\psi(\lambda) = (1+\frac{1-\alpha}{\alpha}\gamma_\theta(\breve{})\breve{})^{-1}$, we can observe that the closed-form solution in Eq. (6.4) is equivalent to the spectral filtering in Eq. (6.1).

$$\mathbf{Z}^* = (\mathbf{I} + \frac{1-\alpha}{\alpha}\gamma_\theta(\hat{\mathbf{L}}))^{-1}\mathbf{X} = g_\psi(\hat{\mathbf{L}})\mathbf{X} = \mathbf{U}g_\psi(\mathbf{\Lambda})\mathbf{U}^T\mathbf{X}. \tag{6.4}$$

As $\gamma_\theta(\lambda)\geq 0$, this establishes a more stringent constraint for the graph filter in spectral GNNs, i.e., $0 < g_\psi(\lambda) \leq \frac{\alpha}{\alpha+(1-\alpha)\cdot 0} = 1$, which is termed as a non-negative constraint in this work.

Iterative Solution. Alternatively, we can take an iterative gradient descent method such that $\mathbf{Z}^{(k)} = \mathbf{Z}^{(k-1)} - b\frac{\partial\mathcal{L}}{\partial\mathbf{Z}}|_{\mathbf{Z}=\mathbf{Z}^{(k-1)}}$ with a step size $b=\frac{1}{2}$, which yields a concise iterative solution in Eq. (6.5) with $\hat{\mathbf{A}}^{\text{new}} = \mathbf{I} - \gamma_\theta(\hat{\mathbf{L}})$. Notably, by taking $\hat{\mathbf{A}}^{\text{new}}$ as a new computation graph, this solution closely mirrors the spatial aggregation in Eq. (6.2).

$$\mathbf{Z}^{(k)} = \alpha\mathbf{X} + (1-\alpha)\hat{\mathbf{A}}^{\text{new}}\mathbf{Z}^{(k-1)}, \quad k=1,2,\cdots,K \tag{6.5}$$

Theoretical Interaction — the Adapted New Graph. With the non-negative

[1]This problem was first introduced in [37] for theoretically grounded graph filters. However, in this study, we repurpose it as a bridge between spectral filtering and spatial aggregation.

constraint, it is evident that both spectral filtering and spatial aggregation effectively address the generalized graph optimization problem in Eq. (6.3), despite their distinctive forms and operation domains. Upon closer examination, we discover a compelling relationship between the graph filter $g_\psi(\lambda)$ in Eq. (6.4) and the new graph $\hat{\mathbf{A}}^{\text{new}}$ in Eq. (6.5), given $g_\psi(\lambda) = (1 + \frac{1-\alpha}{\alpha}\gamma_\theta(\breve{}))^{-1}$,

$$\hat{\mathbf{A}}^{\text{new}} = \mathbf{I} - \gamma_\theta(\hat{\mathbf{L}}) = \mathbf{I} - \frac{\alpha}{1-\alpha}(g_\psi(\hat{\mathbf{L}})^{-1} - \mathbf{I}) \tag{6.6}$$

which unveils an intrinsic inter-play, *i.e.*, spectral filtering implicitly leads the original graph to an adapted new graph, explicitly computed for spatial aggregation.

What are the differences between $\hat{\mathbf{A}}^{\text{new}}$ and $g_\psi(\hat{\mathbf{L}})$? Whereas the former as the uncovered new graph elucidates the inherent spatial node relationships, the latter is a graph operation, primarily processing graph features within spectral domain. It is crucial to understand that $g_\psi(\hat{\mathbf{L}})$ may not result in a dense matrix, especially with fixed-order polynomial approximation. This is because it captures up to only a K-hop neighborhood, i.e., $g_\psi(\hat{\mathbf{L}}) = \sum_{k=0}^{K} \psi_k P_k(\hat{\mathbf{L}}) = \sum_{k=0}^{K} \omega_k \hat{\mathbf{A}}^k$, practically limiting spectral GNNs' effective propagation range. In contrast, our newfound graph $\hat{\mathbf{A}}^{\text{new}}$ intrinsically enjoys a non-local property, as confirmed in the following section. Building upon this discovery, we further devise a framework to break domain barriers, overcoming the limitations of current spectral GNNs due to truncated polynomials (see details in Section 6.5).

6.4.2 In-depth Analysis of the adapted new graph

To deepen our understanding of the interpretability produced by spectral GNNs in the spatial domain, we embark upon a blend of theoretical and empirical inquiries into the adapted new graph.

Non-locality. Our examination of the adapted new graph illuminates its non-local (or alternatively global) nature, particularly evident in the infinite series expansion of the original graph's adjacency matrix. To elucidate, we first introduce an pivotal mathematical construct, the Neumann series, in the following lemma.

Lemma 3. Let $\mathbf{M} \in \mathbb{R}^{N\times N}$ be a matrix with eigenvalues λ_n, if $|\lambda_n| < 1$ for all $n = 1, 2, ..., N$, then $(\mathbf{I}-\mathbf{M})^{-1}$ exists and can be expanded as an infinite series, i.e., $(\mathbf{I}-\mathbf{M})^{-1} = \sum_{t=0}^{\infty} \mathbf{M}^t$, which is known as Neumann series.

With the established non-negative constraint on graph filters, specifically $0 < g_\psi(\lambda) \leq 1$, it becomes evident that the eigenvalues of $\mathbf{I} - g_\psi(\hat{\mathbf{L}})$ falls into the interval permitting Neumann series expansion, as shown in lemma 3 [219]. Building on this observation, we present a non-trivial property of the new graph in the following proposition.

Proposition 2. Given adjacency matrix $\hat{\mathbf{A}}^{\text{new}}$ formulated in Eq. (6.6), the adapted new graph exhibits non-locality. Specifically, $\hat{\mathbf{A}}^{\text{new}}$ is expressible as an infinite series expansion of the original graph's adjacency matrix $\hat{\mathbf{A}}$. Formally, we have $\hat{\mathbf{A}}^{\text{new}} = \mathbf{I} - \frac{\alpha}{1-\alpha}\sum_{t=1}^{\infty}(\mathbf{I} - \sum_{k=0}^{K}\pi_k\hat{\mathbf{A}}^k)^t = \sum_{t=0}^{\infty}\phi_t\hat{\mathbf{A}}^t$ where π_k and ϕ_t refer to the constant coefficients computed from $\{\psi_0, \psi_1, ..., \psi_K\}$ in distinct ways.

Proof. We begin with the assertion that the eigenvalues of $\mathbf{I} - g_\psi(\hat{\mathbf{L}})$ are positive and strictly less than 1, which fulfills the necessary condition for the Neumann series expansion stated in Lemma 3. As such, we can deduce $g_\psi(\hat{\mathbf{L}})^{-1} = (\mathbf{I} - (\mathbf{I} - g_\psi(\hat{\mathbf{L}})))^{-1} = \sum_{t=0}^{\infty}(\mathbf{I} - g_\psi(\hat{\mathbf{L}}))^t$. Owning to the prevalent polynomial approximation, we are eligible to express $g_\psi(\hat{\mathbf{L}})$ w.r.t. adjacency matrix $\hat{\mathbf{A}}$, i.e., $g_\psi(\hat{\mathbf{L}}) = g_\psi(\mathbf{I} - \hat{\mathbf{A}}) = \sum_{k=0}^{K}\pi_k\hat{\mathbf{A}}^k$ where π_k refers to the new coefficients made of up $\{\psi_m\}_{m=0}^{K}$. Substituting this polynomial representation into our Neumann expansion, we obtain $g_\psi(\hat{\mathbf{L}})^{-1} = \sum_{n=0}^{\infty}(\mathbf{I} - \sum_{k=0}^{K}\pi_k\hat{\mathbf{A}}^k)^t$. Now, revisiting $\hat{\mathbf{A}}^{\text{new}}$ in Eq. (6.6), we have $\hat{\mathbf{A}}^{\text{new}} = \mathbf{I} - \frac{\alpha}{1-\alpha}(g_\psi(\hat{\mathbf{L}})^{-1} - \mathbf{I}) = \mathbf{I} - \frac{\alpha}{1-\alpha}\sum_{t=1}^{\infty}(\mathbf{I} - \sum_{k=0}^{K}\pi_k\hat{\mathbf{A}}^k)^t = \sum_{t=0}^{\infty}\phi_t\hat{\mathbf{A}}^t$ where ϕ_t is a constant coefficient made up of $\{\pi_m\}_{m=0}^{K}$. □

This proposition implies that the new graph engenders immediate links between nodes that originally necessitate multiple hops for connection. To further underpin this theoretical claim, we analyze the general connection status on the new graph by BernNet [37], a spectral GNN adhering to the non-negative constraint. From Figure 6.1, it is apparent that nodes originally separated by multiple hops achieve direct connections in the new graph.

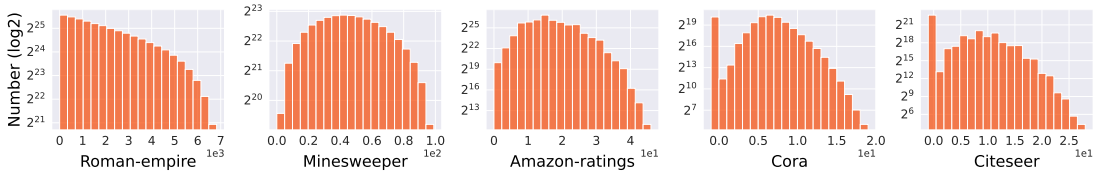


Figure 6.1: Distributions of connected nodes in the new graph based on their geodesic or shortest-path distance (as $\Delta_{i,j}$) in the original graph. Nodes, distant in the original graph ($\Delta_{i,j} > 1$ in x-axis), can be linked in the new graph (Number > 0 in y-axis).

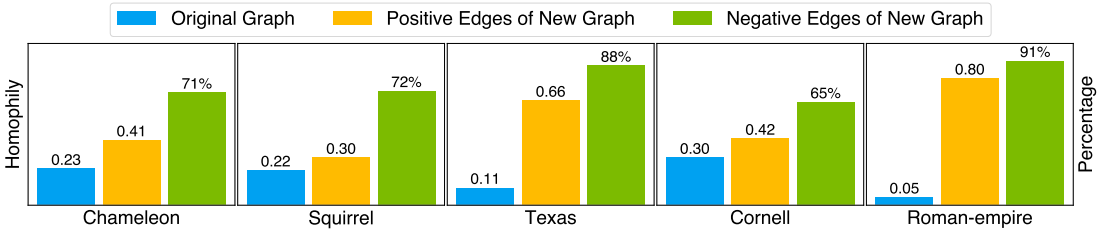


Figure 6.2: Left y-axis: Homophily comparison between original and new graphs, considering only positive edges (blue and yellow bars). Right y-axis: Percentage of edges connecting nodes from different classes, identified by negative edges (green bar).

Signed Edge Weights — Discerning Label Consistency. Upon further scrutinizing the adapted new graph, we make a notable discovery that it readily accommodates both positive and negative edge weights. A more granular analysis in Figure 6.2 reveals that a considerable portion of positive edge weights are assigned to the same-class node pairs, enhancing graph homophily (exemplified by edge homophily ratio [4]). Conversely, edges parameterized with negative weights tend to bridge nodes with different labels. These findings demonstrate the newfound graph's adeptness in discerning label consistency among nodes. To theoretically explain this phenomenon, we further present the proposition below:

Proposition 3. Let $\mathbf{Z}^*$ be the node representations optimized by Eq. (6.3). For $\mathbf{Z}^*$ to be effective in label prediction, it is a necessary condition that $\hat{\mathbf{A}}^{\text{new}}$ accommodates both positive and negative edge weights s.t. for any node pairs $v_i, v_j \in \mathcal{V}$, $\hat{A}^{\text{new}}_{i,j} > 0$ if $\mathbf{y}_i = \mathbf{y}_j$ and $\hat{A}^{\text{new}}_{i,j} < 0$ if $\mathbf{y}_i \neq \mathbf{y}_j$.

Proof. Let us commence the proof by contradiction. Let C denote the condition described

in proposition 3. Assume, for the sake of contradiction, that C is not requisite for the optimal node representations $\mathbf{Z}^*$ to be predictive of node labels. Under this assumption, there are node pairs $v_i, v_j \in \mathcal{V}$ such that: (1) if $\mathbf{y}_i = \mathbf{y}_j$, $\hat{A}^{\text{new}}_{i,j} < 0$; (2) if $\mathbf{y}_i \neq \mathbf{y}_j$, $\hat{A}^{\text{new}}_{i,j} > 0$. Without loss of generality, given the non-locality as proved in proposition 2, we exclude cases where $\hat{A}^{\text{new}}_{i,j} = 0$ from our consideration. Now, consider the second objective term $\text{tr}(\mathbf{Z}^T \gamma_\theta(\hat{\mathbf{L}})\mathbf{Z})$ in Eq. (6.3). Using the relationship $\gamma_\theta(\hat{\mathbf{L}}) = \mathbf{I} - \hat{\mathbf{A}}^{\text{new}}$, we can expand this term into $\sum_{v_i, v_j \in \mathcal{V}} \hat{A}^{\text{new}}_{i,j} \|\mathbf{Z}_i - \mathbf{Z}_j\|_2^2$. Under (1), for same-class nodes v_i, v_j with $\hat{A}^{\text{new}}_{i,j} < 0$, minimizing the objective term pulls $\mathbf{Z}_i$ and $\mathbf{Z}_j$ apart in the latent space. This behavior violates the canonical understanding that nodes from the same class should exhibit similar representations. Under (2), for different-class nodes v_i, v_j with $\hat{A}^{\text{new}}_{i,j} > 0$, the optimization encourages $\mathbf{Z}_i$ and $\mathbf{Z}_j$ to be more similar. This is in direct opposition to the basic classification principle that nodes from different classes should have distinct representations. Given these contradictions stemming from the mathematical implications in optimization, we must reject assumptions (1) and (2), affirming the necessary condition C for accurate label prediction by $\mathbf{Z}^*$. □

Proposition 3 provides a theoretical foundation of our empirical findings on the new graph. The essence lies in the objective in Eq. (6.3), particularly the trace term $\text{tr}(\mathbf{Z}^T \gamma_\theta(\hat{\mathbf{L}})\mathbf{Z})$. For clarity, let us reinterpret this trace term as $\text{tr}(\bar{\mathbf{Z}}^T(\mathbf{D}^{\text{new}} - \mathbf{A}^{\text{new}})\bar{\mathbf{Z}})$, where $\mathbf{D}^{\text{new}}$ denotes the related degree matrix and $\bar{\mathbf{Z}}$ is derived from rescaling $\mathbf{Z}$. Clearly, this term evaluates label smoothness among adjacent nodes in the new graph, which, given its non-local nature, includes both intra-class ($=$) and inter-class ($\neq$) node connections such that $\mathbf{A}^{\text{new}} = \mathbf{A}^{\text{new}}_{=} + \mathbf{A}^{\text{new}}_{\neq}$. Drawing from proposition 3, we can further dissect the original trace term, splitting it into $\text{tr}(\bar{\mathbf{Z}}^T(\mathbf{D}^{\text{new}}_{=} - \mathbf{A}^{\text{new}}_{=})\bar{\mathbf{Z}}) - \text{tr}(\bar{\mathbf{Z}}^T|(\mathbf{D}^{\text{new}}_{\neq} - \mathbf{A}^{\text{new}}_{\neq})|\bar{\mathbf{Z}})$ where the $|\cdot|$ operation denotes absolute values. As such, it becomes evident that minimizing this trace term not only enhances the representational proximity for same-class node pairs but also strengthens the distinctiveness for different-class nodes pairs. Such nuanced behaviors, inherent to the optimization in Eq. (6.3), are necessary for GNN models to achieve accurate label predictions.

To summarize, our investigation into spectral GNNs in the spatial domain reveals

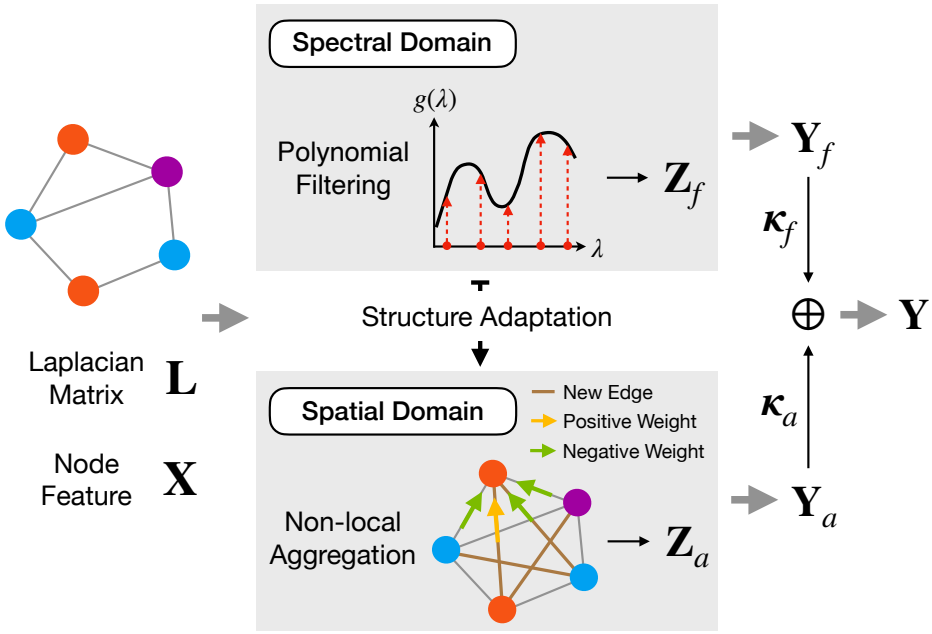


Figure 6.3: Illustration of the proposed SAF framework, where varying node colors represent different node labels.

that graph spectral filtering fundamentally alters the original graph, imbuing it with non-locality and signed edge weights that capture label consistency among nodes. These findings highlight the interpretable role of spectral GNNs in the spatial domain, and prompt us to rethink current spectral GNNs beyond the truncated polynomial filters.

6.5 Spatially Adaptive Filtering Framework

Building on our discoveries, we re-evaluate the state-of-the-art spectral GNNs and put forth a paradigm-shifting framework, Spatially Adaptive Filtering (SAF), for joint exploitation of graph-structured data across both spectral and spatial domains (refer to Figure 6.3). SAF leverages the adapted new graph by spectral filtering for an auxiliary non-local aggregation, addressing enduring challenges in GNNs related to long-range dependencies and graph heterophily.

6.5.1 Non-negative Spectral Filtering

The proposed SAF requires explicit computation of the newfound graph, as outlined in Eq. (6.6). This further necessitates the graph filter $g_\psi : [0, 2] \to \mathbb{R}$ to satisfy the non-negative constraint from Eq. (6.3): $0 \leq g_\psi(\lambda) \leq 1$. However, not all extant graph filters fulfill this prerequisite. For instance, the filter use by GCN [35], $g_\psi(\lambda) = 1 - \lambda$, takes negative values when $\lambda > 1$. In this research, we approximate the graph filter using Bernstein polynomials [220], which are known for their non-negative traits [221] and are essential in a preeminent spectral GNN, BernNet [37]. For $g_\psi(\lambda) \leq 1$ part, we rescale Bernstein polynomials with the following proposition.

Proposition 4. Let $B_{k,K}(x)$ denote the Bernstein polynomial basis of index k and degree K, which is defined as $B_{k,K}(x) = \binom{K}{k}(1-x)^{K-k}x^k$ for $x \in [0, 1]$. Let ψ_k denote the k-th coefficient of a polynomial $p(x)$ of degree K, where $p(x) = \sum_{k=0}^{K} \psi_k B_{k,K}(x)$ with $\psi_k \geq 0$ for all k. Then for all $x \in [0, 1]$, we have $g_\psi(x) \leq \max\{\psi_k\}_{k=0}^{K}$.

Proof. We denote $p(x) = \sum_{k=0}^{K} \psi_k \binom{K}{k}(1-x)^{K-k}x^k$ as a Bernstein polynomial with $\psi_k \geq 0$ for all k and $\psi_{\max} = \max\{\psi_k\}_{k=0}^{K}$. Given $x \in [0, 1]$, we can derive the following inequality as

$$\begin{aligned} p(x) &= \sum_{k=0}^{K} \psi_k \binom{K}{k}(1-x)^{K-k}x^k \leq \psi_{\max} \sum_{k=0}^{K} \binom{K}{k}(1-x)^{K-k}x^k \\ &= \psi_{\max}(1 - x + x)^K = \psi_{\max}. \end{aligned}$$

Therefore, we have $p(x) \leq \max\{\psi_k\}_{k=0}^{K}$ for all $x \in [0, 1]$. □

Proposition 4 suggests that the Bernstein polynomial function attains its maximum value in $\psi_{\max} = \max\{\psi_k\}_{k=0}^{K}$. Therefore, $g_\psi(\lambda)$ can be rescaled within $[0, 1]$ by $\hat{g}_\psi(\lambda) = \frac{1}{\psi_{\max}} \sum_{k=0}^{K} \psi_k B_{k,K}(\frac{\lambda}{2})$, enabling us to formulate the spectral filtering in SAF as

$$\mathbf{Z}_f = \hat{g}_\psi(\hat{\mathbf{L}}) f_\varphi(\mathbf{X}) = \frac{1}{\psi_{\max}} \sum_{k=0}^{K} \psi_k \frac{1}{2^K} \binom{K}{k} (2\mathbf{I} - \hat{\mathbf{L}})^{K-k} \hat{\mathbf{L}}^k f_\varphi(\mathbf{X})$$

where $f_{\varphi}(\cdot)$, a two-layer MLP, maps $\mathbf{X}$ from F to C dimensions using 64 hidden units, and $\{\psi_k\}_{k=1}^{K}$ are non-negative learnable parameters. Note that SAF also permits alternative implementations such as using Chebyshev polynomials [55, 222] for graph filter learning, enhancing models like ChebNetII [75] (see details in Section 6.7.3).

6.5.2 Non-local Spatial Aggregation

Once acquiring a suitable spectral filter $\hat{g}_{\psi}(\lambda)$, we compute the adapted new graph as $\hat{\mathbf{A}}^{\text{new}} = \mathbf{I} - \tau(\mathbf{U}_{\mathbf{m}} g_{\psi}(\mathbf{\Lambda}_{\mathbf{m}})^{-1} \mathbf{U}_{\mathbf{m}}{}^{T} - \mathbf{I})$ by Eq. (6.6) where $\tau = \frac{\alpha}{1-\alpha}$ is a scaling parameter and a partial eigendecomposition can be employed to obtain only m extremal eigenvalues [208], producing a low-rank, robust structure for $\hat{\mathbf{A}}^{\text{new}}$. Equipped with this newfound graph, we proceed to perform non-local aggregation:

$$\mathbf{Z}^{(l)} = (1-\eta)\mathbf{Z}^{(0)} + \eta \hat{\mathbf{A}}^{\text{new}} \mathbf{Z}^{(l-1)}, \quad l = 1, 2, \cdots, L$$

where η refers to the update rate and $\mathbf{Z}^{(0)} = f_{\varphi}(\mathbf{X})$. The iteratively aggregated results are denoted as $\mathbf{Z}_a$. Recognizing the potential noise from the non-local nature of $\hat{\mathbf{A}}^{\text{new}}$, we apply a sparsification technique, leveraging a positive threshold ϵ, and retain only essential elements outside the $[-\epsilon, \epsilon]$ interval. For clarity, this refined model is referred to as SAF-ϵ.

6.5.3 Node-wise Prediction Amalgamation

To leverage information from different graph domains, we employ an attention mechanism, allowing nodes to determine the importance of each space. This mechanism produces pairwise weights for a nuanced amalgamation during prediction. Specifically, the weight pair is computed as $\boldsymbol{\kappa}_f = \text{Sigmoid}(\mathcal{P}_f(\mathbf{Z}_f))$, $\boldsymbol{\kappa}_a = \text{Sigmoid}(\mathcal{P}_a(\mathbf{Z}_a))$ where $\boldsymbol{\kappa}_f, \boldsymbol{\kappa}_a \in \mathbb{R}^N$ contain the weights for each node, and $\mathcal{P}_f(\cdot)$ and $\mathcal{P}_a(\cdot)$ are two different mappings from $\mathbb{R}^C$ to $\mathbb{R}$. For simplicity, we implement them using two one-layer MLPs. Given domain

predictions $\mathbf{Y}_f, \mathbf{Y}_a \in \mathbb{R}^C$, the final model prediction is attained as

$$\mathbf{Y} = \mathrm{diag}(\boldsymbol{\kappa}_f) \cdot \mathbf{Y}_f + \mathrm{diag}(\boldsymbol{\kappa}_a) \cdot \mathbf{Y}_a \tag{6.7}$$

where a normalization $\left[\boldsymbol{\kappa}_f, \boldsymbol{\kappa}_a\right] \leftarrow \frac{\left[\boldsymbol{\kappa}_f, \boldsymbol{\kappa}_a\right]}{\max\{\|\{\boldsymbol{\kappa}_f, \boldsymbol{\kappa}_a\}\|_1, \delta\}}$ is performed beforehand to maintain $\boldsymbol{\kappa}_f + \boldsymbol{\kappa}_a = \mathbf{1}$ with small value δ preventing zero division. Similar schemes can be founded in works [223–225].

6.5.4 Computational Complexity

SAF augments spectral GNNs with non-local aggregation and node-wise amalgamation. The first part entails creating a new graph and information propagation. In SAF-$\boldsymbol{\epsilon}$, these two steps are separated by sparsification, culminating in $O(N^3 + N^2 + nnz(\hat{\mathbf{A}}^{\text{new}})d)$ complexity, where nnz denotes non-zero element count. Conversely, SAF, viewing non-local aggregation holistically, can reduce complexity to $O(2dN^2 + dN)$ when $d \ll N$. For node-wise amalgamation, its parallelizable nature ensures computational efficiency. Our method also requires a precomputation of eigendecomposition[2] which, naively complex at $O(N^3)$, can be reduced to $O(m^2 + nnz(\hat{\mathbf{L}})m)$ using Lanczos method [208] for larger graphs with $m \ll N$ denoting iterative steps and is reusable for both training and inference. We present empirical studies on both time and space overheads in Section 6.7.4.

6.6 Experiments

6.6.1 Experimental Setup

Datasets. We evaluate models over 13 real-world datasets from various domains. These include three well-known homophilic graphs: Cora, Citeseer, and Pubmed [140], five commonly used heterophilic graphs: Chameleon, Squirrel [172], Cornell, Texas [60], and Actor [173], as well as five recently introduced benchmarks: Minesweeper (synthetic graph), Tolokers (crowdsourcing platform worker network), Amazon-ratings (product

[2]For more on its modern applications and discussions, please refer to section 6.2 (including Specformer [78] & FE-GNN [82] featured in our experiments) and 6.7.4.

Table 6.1: Statistics of real-world datasets. Δ represents graph diameter referring to the longest geodesic distance between nodes on the graph. For Penn94, due to multiple subgraphs, we report Δ of the largest connected component.

Dataset	# Nodes	# Edges	# Features	# Classes	Δ	$\mathcal{H}$	$\mathcal{H}_{\text{class}}$	$\mathcal{H}_{\text{adjusted}}$
Chameleon	2,227	36,101	2,325	5	11	0.23	0.06	0.03
Squirrel	5,201	217,073	2,089	5	10	0.22	0.03	0.01
Texas	183	309	1,703	5	8	0.11	0.00	-0.23
Cornell	183	295	1,703	5	8	0.30	0.05	-0.08
Actor	7,600	33,544	931	5	12	0.22	0.01	0.00
Cora	2,708	5,429	1,433	7	19	0.81	0.77	0.77
Citeseer	3,327	4,732	3,703	6	28	0.74	0.63	0.67
Pubmed	19,717	44,338	500	3	18	0.80	0.66	0.69
Minesweeper	10,000	39,402	7	2	99	0.68	0.01	0.01
Tolokers	11,758	519,000	10	2	11	0.59	0.18	0.09
Amazon-ratings	24,492	93,050	300	5	46	0.38	0.13	0.14
Roman-empire	22,662	32,927	300	18	6,824	0.05	0.02	-0.05
Penn94	41,554	1,362,229	5	2	8	0.47	0.05	0.02

co-purchasing), Roman-empire (word dependency graph) [226], and Penn94 [150] (social network). Detailed statistics are summarized in Table 6.1. Alongside standard data attributes, we also provide the longest geodesic (shortest-path) distance between graph nodes for better illustrating the non-local property that we investigate in Figures 6.1. Moreover, we adopt three metrics - edge homophily [4] $\mathcal{H}$, class homophily [150] $\mathcal{H}_{\text{class}}$, and adjusted homophily [227] $\mathcal{H}_{\text{adjusted}}$ - to assess the graph's homophily ratio, which ranges from 0 (high heterophily) to 1 (high homophily). While the first is a commonly used index, the latter two, considering class variability and potential imbalance, have been recently introduced for more accurate estimation. For our analysis regarding the adapted new graph, we primarily rely on the edge homophily metric, defined as $\mathcal{H} = |\{(v_i, v_j)|(v_i, v_j) \in \mathcal{E} \wedge \mathbf{y}_i = \mathbf{y}_j\}|/|\mathcal{E}|$, given its simplicity and wide usage. In certain compact sections, we use four-letter abbreviations for dataset names.

Baselines. We compare SAF with 22 models: (1) MLP; (2) Basic GNNs: GCN [35] and APPNP [36]; (3) Spectral GNNs: ARMA [66], GPR-GNN [54], BernNet [37], ChebNetII [75], JacobiConv [74], Specformer [78], LON-GNN [79] and OptBasisGNN [80]; (4) Spatial GNNs: GCNII [59], PDE-GCN [65], GEN [63], NodeFormer [71], GloGNN++ [72] and MGNN [76]; (5) Unified GNNs: GNN-LF [67], GNN-HF [67], ADA-UGNN [68],

FE-GNN [82] and ClenshawGCN [81]. In our experiments, we leverage the Pytorch Geometric library [228] implementations for GCN and APPNP. For MLP, we include a sequence of linear layers, each of which is followed by batch normalization, ReLU activation, and dropout. The number of MLP layers are tuned from 1 to 5. For the remaining baselines, we resort to their publicably released code. Besides, for the work [72], only the most effective model variant, GloGNN++, is included in our experiments. Models like Geom-GCN [60] and Non-Local GNNs [62], surpassed by these SOTA methods, are excluded from our comparison.

Setup. To follow [37, 74, 75], we fix $K = 10$. For each dataset, we perform a grid search to tune the hyper-parameters of all models. With the best hyper-parameters, we train models with Adam optimizer [169] in 1,000 epochs using early-stopping strategy and a patience of 200 epochs, and report the mean classification accuracies with a 95% confidence interval on 10 random data splits. As [75] have made a comprehensive evaluation and share the same experimental protocol with us, we directly leverage their results for models: MLP, GCN, APPNP, ARMA, GPR-GNN, BernNet, ChebNetII, GCNII and PDE-GCN on datasets including Chameleon, Squirrel, Texas, Cornell, Actor, Cora, Citeseer, and Pubmed. For JacobiConv, LON-GNN, and OptBasisGNN, we also report their results from corresponding papers [74, 79, 80]. Other experiments are performed on a machine equipped with an NVIDIA GeForce RTX 3090 (24GB) and an Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz (20 cores).

Hyper-parameters Setting. We perform a grid search on the hyper-parameters of all models (including baselines) for each dataset using the open-source package Optuna [141] To accommodate extensive experiments across diverse datasets in both semi- and full-supervised setting, we define a broad searching space as: learning rate lr $\sim$ {1e-3, 5e-3, 1e-2, 5e-2, 0.1}, weight decay L_2 $\sim$ {0.0, 1e-6, 5e-6, 1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2}, dropout $\sim$ {0.0, 0.1, ..., 0.8} with step 0.1, non-local aggregation step $L \sim$ {1,2, ...,10} with step 1, scaling parameter $\tau \sim$ {0.1, 0.2, ..., 1.0} with step 0.1, update rate $\eta \sim$ {0.1, 0.2, ..., 1.0} with step 0.1, and threshold $\epsilon \sim$ {0.0, 1e-5, 5e-5, 1e-4, 5e-4, 1e-3, 5e-3, 1e-2}. For other parameters specific to different base models, we strictly follow

Table 6.2: Semi-supervised node classification accuracy (%) ± 95% confidence interval.

Method	Cham.	Squi.	Texas	Corn.	Actor	Cora	Cite.	Pubm.
MLP	26.36±2.85	21.42±1.50	32.42±9.91	36.53±7.92	29.75±0.95	57.17±1.34	56.75±1.55	70.52±2.01
GCN	38.15±3.77	31.18±0.93	34.68±9.07	32.36±8.55	22.74±2.37	79.19±1.37	69.71±1.32	78.81±0.84
APPNP	32.73±2.31	24.50±0.89	34.79±10.11	34.85±9.71	29.74±1.04	82.39±0.68	69.79±0.92	79.97±1.58
ARMA	37.42±1.72	24.15±0.93	39.65±8.09	28.90±10.07	27.02±2.31	79.14±1.07	69.35±1.44	78.31±1.33
GPR-GNN	33.03±1.92	24.36±1.52	33.98±11.90	38.95±12.36	28.58±1.01	82.37±0.91	69.22±1.27	79.28±2.25
BernNet	27.32±4.04	22.37±0.98	43.01±7.45	39.42±9.59	29.87±0.78	82.17±0.86	69.44±0.97	79.48±1.47
ChebNetII	**43.42±3.54**	**33.96±1.22**	46.58±7.68	42.19±11.61	30.18±0.81	82.42±0.64	69.89±1.21	79.51±1.03
JacobiConv	36.67±1.63	29.38±0.71	48.50±5.90	43.01±11.92	31.69±0.71	82.93±0.55	70.25±1.02	79.53±1.28
Specformer	36.05±3.47	29.64±0.88	50.00±8.33	43.76±5.84	31.45±0.68	81.44±0.63	66.11±0.98	78.05±1.03
LON-GNN	35.17±1.85	30.25±1.04	45.38±7.92	35.32±8.09	31.51±1.23	81.93±0.74	70.41±1.10	79.57±1.08
OptBasisGNN	35.56±2.86	31.25±1.06	37.11±5.09	32.31±7.11	31.73±0.50	78.69±0.86	63.46±1.30	77.38±0.98
GNN-LF	26.49±2.00	22.01±1.04	39.02±6.24	36.65±9.60	28.28±0.71	81.96±0.92	69.80±1.36	79.50±1.28
GNN-HF	35.57±2.26	22.36±1.26	44.80±5.67	38.79±11.62	29.15±0.78	81.15±0.78	69.68±0.73	79.10±1.19
ADA-UGNN	39.39±2.02	25.65±0.49	47.86±6.65	42.89±8.09	30.78±1.00	82.52±1.04	70.18±1.40	79.78±1.32
FE-GNN	38.23±1.66	31.67±1.60	47.40±5.90	41.21±8.96	26.20±0.76	77.00±0.74	61.24±1.26	75.63±1.33
ClenshawGCN	38.29±2.44	31.24±1.27	49.42±6.01	46.76±12.83	31.84±0.75	82.38±0.77	69.23±1.21	79.76±1.03
SAF	41.82±1.74	31.77±0.69	58.04±3.76	52.49±8.56	33.50±0.55	83.57±0.66	71.07±1.08	79.51±1.12
SAF-ϵ	41.88±2.04	32.05±0.40	**58.38±3.47**	**53.41±5.55**	**33.84±0.58**	**83.79±0.71**	**71.30±0.93**	**80.16±1.25**
Improv.*	14.56%	9.68%	15.37%	13.99%	3.97%	1.62%	1.86%	0.68%

* This row lists the relative improvement of best performing SAF variants over its base model BernNet [37].

their instructions in the original papers.

6.6.2 Overall Evaluation

Semi-supervised Node Classification. In this task, we follow the experimental protocol established by [75] and compare our models with MLP, two basic GNNs, eight popular polynomial spectral GNNs, and five unified GNNs. For data splitting on homophilic graphs (Cora, Citeseer, and Pubmed), we apply the standard division [229] with 20 nodes per class for training, 500 nodes for validation, and 1,000 nodes for testing. On the other five heterophilic graphs, we leverage the sparse splitting [54] with 2.5%/2.5%/95% samples respectively for training/validation/testing. The results are reported in Table 6.2, where the best results are bold and the underlined letters denote the second highest accuracy. We first observe that both SAF and SAF-ϵ substantially boosts its base model, BernNet, with gains reaching a notable 15.37%. This impressive enhancement is credited to their capacity to effectively exploit the task-beneficial information, which is implicitly encoded by spectral filtering in the spatial

Table 6.3: Full-supervised node classification accuracy (%) ± 95% confidence interval.

Method	Cham.	Squi.	Texas	Corn.	Actor	Cora	Cite.	Pubm.
MLP	46.59±1.84	31.01±1.18	86.81±2.24	84.15±3.05	40.18±0.55	76.89±0.97	76.52±0.89	86.14±0.25
GCN	60.81±2.95	45.87±0.88	76.97±3.97	65.78±4.16	33.26±1.15	87.18±1.12	79.85±0.78	86.79±0.31
APPNP	52.15±1.79	35.71±0.78	90.64±1.70	91.52±1.81	39.76±0.49	88.16±0.74	80.47±0.73	88.13±0.33
ARMA	60.21±1.00	36.27±0.62	83.97±3.77	85.62±2.13	37.67±0.54	87.13±0.80	80.04±0.55	86.93±0.24
GPR-GNN	67.49±1.38	50.43±1.89	92.91±1.32	91.57±1.96	39.91±0.62	88.54±0.67	80.13±0.84	88.46±0.31
BernNet	68.53±1.68	51.39±0.92	92.62±1.37	92.13±1.64	41.71±1.12	88.51±0.92	80.08±0.75	88.51±0.39
ChebNetII	71.37±1.01	57.72±0.59	93.28±1.47	92.30±1.48	41.75±1.07	88.71±0.93	80.53±0.79	88.93±0.29
JacobiConv	74.20±1.03	57.38±1.25	93.44±2.13	92.95±2.46	41.17±0.64	88.98±0.46	80.78±0.79	89.62±0.41
Specformer	75.06±1.10	**65.05±0.96**	90.33±3.12	90.00±2.79	42.55±0.67	88.85±0.46	80.68±0.90	91.25±0.31
LON-GNN	73.00±2.20	60.61±1.69	87.54±3.45	84.47±3.45	39.10±1.59	89.44±1.12	81.41±1.15	90.98±0.64
OptBasisGNN	74.26±0.74	63.62±0.76	91.15±1.97	89.84±2.46	42.39±0.52	87.96±0.71	80.58±0.82	90.30±0.19
GCNII	63.44±0.85	41.96±1.02	80.46±5.91	84.26±2.13	36.89±0.95	88.46±0.82	79.97±0.65	89.94±0.31
PDE-GCN	66.01±1.56	48.73±1.06	93.24±2.03	89.73±1.35	39.76±0.74	88.62±1.03	79.98±0.97	89.92±0.38
GEN	68.82±0.96	56.05±1.04	92.30±2.30	90.49±1.80	41.08±2.08	89.21±0.54	79.40±0.59	90.40±0.24
NodeFormer	53.02±1.58	34.25±1.96	87.71±2.13	90.00±3.45	41.74±0.61	86.93±1.22	79.58±0.85	91.27±0.39
GloGNN++	72.36±0.85	60.60±1.04	91.48±1.48	89.84±3.62	41.87±1.02	87.21±0.59	79.89±0.61	86.89±0.33
MGNN	72.65±1.16	55.40±1.13	87.05±2.46	85.25±3.28	41.06±0.87	86.59±0.77	78.47±1.53	90.53±0.75
GNN-LF	53.74±1.29	36.15±0.86	76.07±2.62	78.36±2.46	38.39±0.81	88.51±0.89	79.84±0.56	89.86±0.23
GNN-HF	55.97±1.05	35.29±0.72	81.15±2.62	85.41±3.12	38.96±0.77	88.28±0.64	80.04±0.93	90.35±0.30
ADA-UGNN	61.09±1.51	42.02±1.26	84.92±3.12	83.61±3.44	41.10±0.62	88.74±0.85	79.81±1.11	90.61±0.44
FE-GNN	73.00±1.31	63.28±0.81	88.03±1.80	86.07±3.12	41.74±0.67	89.21±0.71	80.26±1.06	90.80±0.30
ClenshawGCN	74.36±0.59	62.94±1.04	91.48±1.97	91.15±2.46	41.98±0.65	88.93±0.85	78.05±0.97	91.10±0.43
SAF	**75.30±0.96**	63.63±0.81	94.10±1.48	92.95±1.97	42.93±0.79	89.80±0.69	80.61±0.81	91.49±0.29
SAF-ϵ	74.84±0.99	64.00±0.83	**94.75±1.64**	**93.28±1.80**	**42.98±0.61**	**89.87±0.51**	**81.45±0.59**	**91.52±0.30**
Improv.*	6.77%	12.61%	2.13%	1.15%	1.27%	1.36%	1.37%	3.01%

domain. This ability is particularly advantageous in contexts with limited supervision, where it allows effective leveraging of extra prior knowledge during training. Generally, our models outperform competitors on all datasets except for Chameleon and Squirrel, where SAF maintains a second-place rank with considerable improvements on BernNet by 14.56% and 9.68%. In these cases, ChebNetII initially surpasses our model, yet, with more training samples, our SAF manages to beats it by margins of 3.93% and 6.28% (see Table 6.3). Moreover, it can be seen that SAF-ϵ averagely delivers better results than SAF, benefiting from its thresholding sparsity that reduces non-local noise for efficient graph learning. However, this enhancement also incurs higher computational costs, as illustrated in both Section 6.5 and Section 6.7.4.

Full-supervised Node Classification.[3] To bolster our evaluation, we expand the previously compared baselines to include six cutting-edge spatial GNN models: GCNII

[3]We borrow this terminology from [59, 75] to denote 60%/20%/20% data splitting.

Table 6.4: Evaluations on new heterophilic graph datasets

Method	Mine.	Tolo.	Amaz.	Roma.	Penn94
MLP	50.61±0.87	74.58±0.69	45.50±0.38	66.11±0.33	74.58±0.37
GCN	72.25±0.60	76.56±0.85	48.06±0.39	53.49±0.33	82.47±0.27
APPNP	68.48±1.20	74.13±0.62	48.12±0.37	72.99±0.46	75.29±0.27
GPR-GNN	89.76±0.53	75.82±0.50	49.06±0.25	73.19±0.24	81.38±0.16
ChebNetII	83.62±1.51	78.95±0.49	49.76±0.36	74.52±0.54	83.12±0.22
JacobiConv	89.88±0.33	77.24±0.39	43.89±0.28	74.30±0.50	83.35±0.11
FE-GNN	84.68±0.36	79.31±0.59	49.46±0.29	74.50±0.30	82.30±0.54
ClenshawGCN	90.36±0.92	**80.94±0.52**	50.14±0.52	73.14±0.51	**84.71±0.31**
BernNet	77.75±0.61	75.35±0.63	49.84±0.52	74.56±0.74	82.47±0.21
SAF	**90.54±0.30**	79.38±0.58	**50.49±0.28**	**74.87±0.22**	83.86±0.26
Improv.	12.79%	4.03%	0.65%	0.31%	1.39%

& PDE-GCN for capturing long-range dependency, GEN & MGNN for handling graph heterophily, and NodeFormer & GloGNN++ addressing both. For all datasets, we randomly divide them into 60%/20%/20% for training/validation/testing by following [37, 75]. Table 6.3 summarizes the mean classification accuracies. Our methods demonstrate superior performance across most datasets, with an exception on Squirrel where they achieve comparable results to Specformer. This notable performance is primarily attributed to our SAF's effective non-local aggregation, utilizing signed edge weights to model global label relationships. This enables our methods to outperform GNNs that are specifically tailored for long-range dependency and/or graph heterophily. Besides the eight standard datasets for node classification, our study also extends to five new benchmarks [150, 226] that focus on graph heterophily. Due to space limit, we present detailed results in Section 6.6.2.

New Benchmarks for Graph Heterophily For a more extensive evaluation across various domains, we also test SAF on five recently introduced datasets, including Minesweeper, Tolokers, Amazon-ratings, Roman-empire, and Penn94 [150, 226]. In this context, we draw comparisons solely with MLP, GCN, APPNP, along with six GNN models that have previously shown promising results in prior tasks, namely GPR-GNN, BernNet, ChebNetII, JacobiConv, FE-GNN, and ClenshawGCN. Table 6.4 lists the average classification accuracies that are obtained over random splits provided by [150, 226], with a distribution of 50%/25%/25% for training/validation/testing.

In summary, SAF achieves significant performance gains of 12.79% and 4.03% on Minesweeper and Tolokers datasets, respectively, while maintaining competitiveness on the other three datasets.

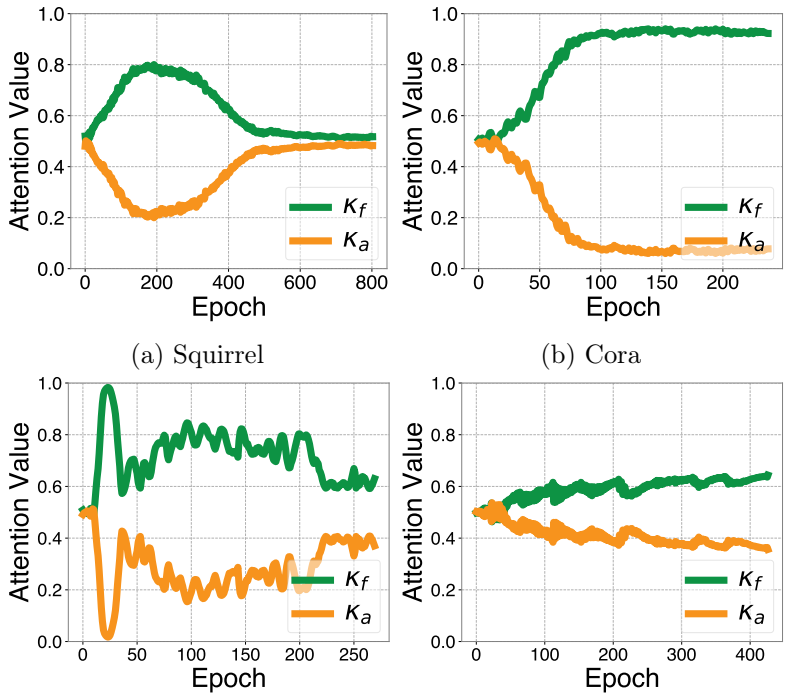
6.7 Analysis and Discussion

6.7.1 Analysis of Attention Trends

We analyze the changing trends of the pair-wise attention weights during training SAF. From Figure 6.4, the average weights for filtering and aggregation start similarly but diverge throughout training, showing different trends in heterophilic and homophilic graphs. On the heterophilic graph Squirrel, both weights converge to similar values, demonstrating their mutual importance in modeling complex connectivity. Conversely, $\boldsymbol{\kappa}_f$ becomes dominant on the homophilic graph Cora due to the sufficiency of node proximity information for label prediction, thereby diminishing the relevance of $\boldsymbol{\kappa}_a$ and non-local aggregation. It is noteworthy that, despite Pubmed being a homophilic graph with $\mathcal{H} = 0.80$, non-local aggregation maintains a pivotal role within SAF, as shown in Figure 6.4(d), diverging from the patterns observed on the Cora dataset. This discrepancy owes much to the sizable node count of Pubmed, where the capability of non-local aggregation in capturing long-range dependencies proves advantageous for improving model performance.

6.7.2 Parameter and Ablation Study

Parameter Analysis. This section presents the sensitivity analysis of hyper-parameters including τ, η, ϵ, and L. Figure 6.5 visualizes how varying these parameters within a broad range influences learning performance, showcasing our model's robust stability over diverse settings. Beyond empirical observation, we also provides deeper insights into parameter understanding and rationalizes the chosen ranges for parameter searching: **1)** The scaling parameter $\tau = \frac{\alpha}{1-\alpha}$, crucial in new graph construction in Eq. (6.6), stems from the trade-off parameter $\alpha \in (0, 1)$ within the graph optimization problem in



(a) Squirrel (b) Cora

(c) Chameleon (d) Pubmed

Figure 6.4: Attention changing trends w.r.t. training epochs.

Eq.(6.3). While theoretically we have $0 = \frac{0}{1-0} < \tau < \frac{1}{1-1} = \infty$, practical considerations for extracting structural information suggest a larger penalty on the trace objective term $\mathrm{tr}(\mathbf{Z}^T \gamma_\theta(\hat{\mathbf{L}})\mathbf{Z})$, i.e., keeping $\alpha < 0.5$, thereby limiting $\tau < \frac{0.5}{1-0.5} = 1$. This rationale substantiates our selection of τ within the set $\{0.1, 0.2, ..., 1\}$ as stated in Section 6.6.1, aligning with the observed optimal performance in Figures 6.5(a)-(c). When addressing graphs with noisy structure, we may adjust the upper limit of α to $t \in (0, 1)$, setting τ's maximum possible value to $\frac{t}{1-t}$. For graph benchmarking evaluations in this work, where extracting structural information is important, we practically set $t = 0.5$. **2)** For the non-local aggregation layer number L, a noticeable decline in model performance is observed when L exceeds 10. This is attributed to the non-local nature of our new graph, which facilitates efficient information exchange between nodes. Exceeding a certain number of layers may potentially lead to oversmoothing, where there is an overemphasis on global information, thus degrading model performance. However, choosing the number

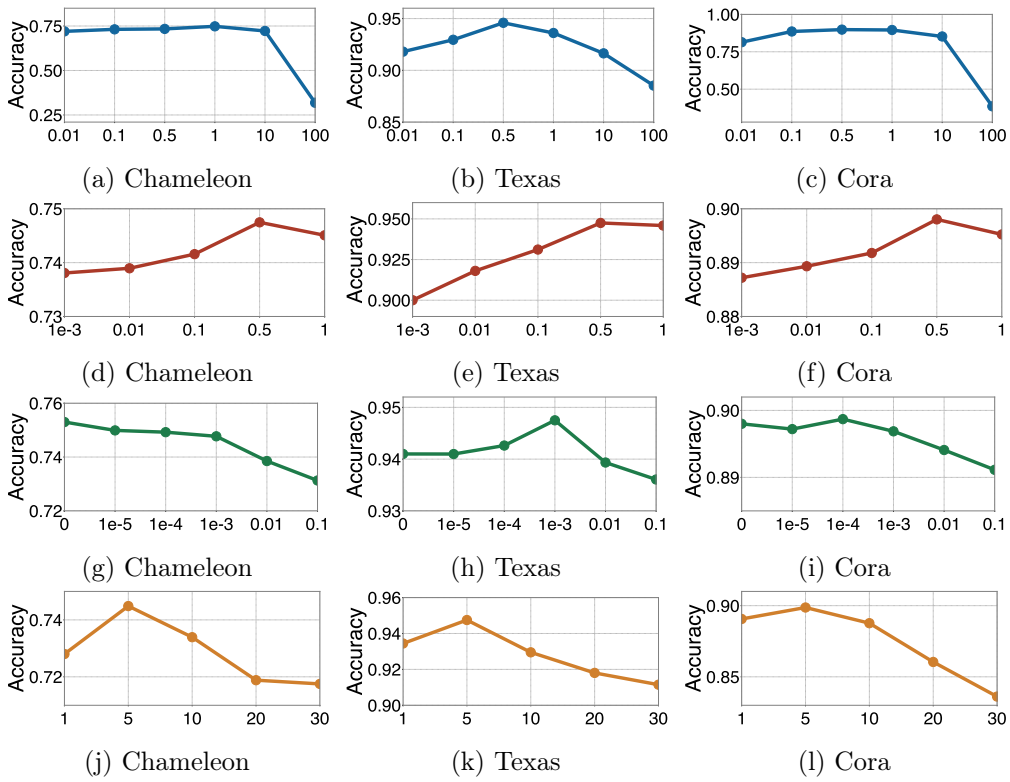


Figure 6.5: Sensitivity analysis for hyper-parameters: τ (blue), η (red), ϵ (green), and L (orange) from top to bottom rows.

of layers within a reasonable range generally ensures consistent and impressive model performance, as verified in Figures 6.5(j)-(l).

Ablation Study. This section aims to validate our designs by comparing SAF with its three ablated variants – SAF w/o Atte., SAF w/o Spec., and SAF w/o Spat. – in full-supervised node classification. Specifically, Atte., Spec., and Spat. respectively refers to: attention mechanism in "Node-wise Prediction Amalgamation", "Non-negative Spectral Filtering", and "Non-local Spatial Aggregation". For SAF w/o Atte., we remove the attention mechanism and equally blend predictions from different domains. SAF w/o Spec. abandons the spectral filtering phase, practically setting $\boldsymbol{\kappa}_f = \mathbf{0}, \boldsymbol{\kappa}_a = \mathbf{1}$ in Eq. (6.7). As the SAF w/o Spat. configuration is equivalent to BernNet model, the corresponding results are posted directly. From Figure 6.6, we can draw several insights: **1)** The impact of Atte. module on our SAF varies by datasets, e.g., on Squirrel and

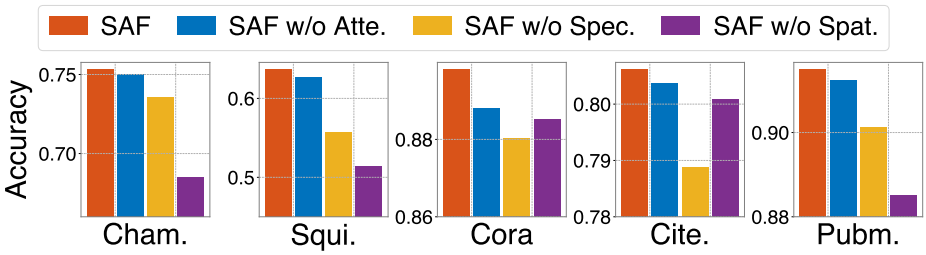


Figure 6.6: Ablation study of SAF framework on six datasets.

Table 6.5: Full-supervised node classification accuracy (%). SAF-Cheb refers to SAF implementation using Chebyshev polynomials.

Method	**Cham.**	**Squi.**	**Texas**	**Corn.**	**Actor**	**Cora**	**Cite.**	**Pubm.**
ChebNetII	71.37±1.01	57.72±0.59	93.28±1.47	92.30±1.48	41.75±1.07	88.71±0.93	80.53±0.79	88.93±0.29
SAF-Cheb	74.97±0.66	64.06±0.59	94.43±1.81	92.62±2.13	42.65±1.01	89.56±0.64	80.68±0.68	91.27±0.34
SAF-Cheb-ϵ	75.25±0.96	64.42±0.82	94.26±1.64	93.12±1.64	42.79±1.04	89.61±0.71	81.08±0.68	91.73±0.18
Improv.	3.88%	6.70%	1.15%	0.82%	1.04%	0.90%	0.15%	2.80%

Chameleon , showing a slight performance reduction upon its removal. This observation aligns with our observation that their optimal attention values are close to an even split, as suggested in Figures 6.4(a) and (c). Conversely, Cora dataset exhibits a notable drop, due to its optimal attention weights being far from even, as depicted in Figure 6.4(b). **2)** Spectral filtering (Spec. module) remains vital for discriminative node representation learning. Specifically, the quality of the adapted new graph fundamentally hinges on the graph spectral filters' training, as underscored by their theoretical interaction in Eq. (6.6). Practically, the absence of spectral filtering markedly reduces model accuracy, confirming its importance in SAF. **3)** This visualization not only reaffirms the pivotal role of the non-local aggregation (Spat. module), but also underscores its position as the most crucial component in advancing spectral GNNs within the SAF framework.

6.7.3 SAF with ChebNetII as Base Model

To expand the versatility of our SAF framework, we introduced ChebNetII [75] as an alternative base model, chosen for its adherence to the non-negative constraint, critical in our model design as stated in Section 6.5.1. The rationale behind this choice is

ChebNetII's use of Chebyshev interpolation for learning Chebyshev polynomials, where the constraint can be ensured by keeping its learnable parameters $\{\gamma_j\}_{j=0}^{K}$ non-negative. Our experiments, as shown in Table 6.5, confirm that SAF can significantly enhances ChebNetII's performance, underscoring the framework's flexibility with different spectral filters. Interestingly, we observed that SAF, utilizing Bernstein polynomials (SAF-Bern), slightly surpasses its performance with Chebyshev polynomials (SAF-Cheb) in most datasets. The margin of improvement over the base model is also more pronounced with SAF-Bern. This phenomenon could be attributed to the $g_\phi(\lambda) \leq 1$ constraint within SAF (refer to Section 6.5.1), necessitating the rescaling of filter functions by their maximum values. For Bernstein polynomials, this maximum is readily obtained as the largest polynomial coefficient $\max\{\phi_k\}_{k=0}^{K}$, as per Proposition 3. However, for Chebyshev polynomials, the best theoretical upper bound is the sum of absolute coefficients, $\sum_{k=0}^{K}|\phi_k|$, which is comparatively less precise. This difference may impact the quality of graph construction and, subsequently, compromise the model's performance.

6.7.4 Time and Space Overheads

Eigendecomposition. Our SAF framework pre-computes eigendecomposition once per graph and reuses it in Eq. (6.6). This aspect is crucial, as the forward-pass cost in model training often exceeds the preprocessing expense of eigendecomposition. To empirically validate this, we compare the time overheads of eigendecomposition with the training times of various models in Table 6.6. It is evident that for most datasets, the time consumed by decomposition is significantly less than the time required for model training. For medium-sized graphs such as Pubmed, while the full decomposition time exceeds that of BernNet, it still maintains efficiency against more advanced GNNs (ChebNetII, ClenshawGCN, SAF). Moving to the large-scale graph, Penn94, where only partial eigendecomposition with 100 extremal eigenvalues is considered, the computation time is markedly reduced compared to all models mentioned.

Model Comparison. To evaluate the efficiency of our model, we compare the running times of its variants, SAF and SAF-ϵ, against three notable spectral GNNs (BernNet,

Table 6.6: Time overheads (s) / Space overheads (MB). For large-scale graph Penn94 (with 41,554 nodes and 1,362,229 edges), we only employ a partial eigendecomposition with 100 extremal eigenvalues to balance between model effectiveness and efficiency.

Method	**Cham.**	**Squi.**	**Texas**	**Corn.**	**Actor**	**Cora**	**Cite.**	**Pubm.**	**Penn94**
BernNet	8.36/72	13.74/232	3.92/5	4.16/5	4.88/292	5.24/64	5.52/152	6.06/1546	24.05/1902
ChebNetII	22.82/72	30.73/231	11.47/5	9.64/5	14.88/291	19.96/63	16.14/152	36.91/1584	41.67/1850
ClenshawGCN	15.58/98	26.21/398	2.98/5	4.13/5	4.43/313	3.72/68	12.48/152	24.97/1833	191.30/3017
SAF	11.55/112	18.78/440	4.38/5	4.70/5	5.36/733	6.04/120	6.12/237	18.43/4515	23.49/8491
Decomposition	0.58/141	1.59/540	0.02/1	0.02/1	3.93 /1206	1.00/140	0.77 /239	21.34/7641	4.76/4

Table 6.7: Average running time per epoch (ms) / average total running time (s).

Method	**Cham.**	**Squi.**	**Texas**	**Corn.**	**Actor**	**Cora**	**Cite.**	**Pubm.**
BernNet	14.60/8.36	17.40/13.74	16.10/3.92	14.30/4.16	14.40/4.88	14.90/5.24	16.20/5.52	15.10/6.06
ChebNetII	39.30/22.82	40.70/30.73	42.30/11.47	39.60/9.64	41.80/14.88	39.20/19.96	40.70/16.14	40.60/36.91
JacobiConv	11.10/6.54	11.20/10.01	11.60/3.86	11.20/2.93	11.70/4.22	11.50/5.54	11.30/6.02	11.40/8.47
NodeFormer	135.00/58.96	135.10/79.66	49.10/14.29	67.60/18.89	158.20/66.20	56.70/19.25	73.10/32.00	150.60/68.57
GloGNN++	53.60/35.63	123.70/68.31	14.80/4.47	9.80/3.00	204.80/73.13	89.60/32.68	49.60/12.35	6369.20/5266.53
ClenshawGCN	20.6/15.58	39.7/26.21	11.0/2.98	14.5/4.13	13.7/4.43	11.0/3.72	29.7/12.48	68.1/24.97
SAF-ϵ	30.70/19.11	59.50/54.71	27.30/6.95	29.70/7.86	121.20/33.14	30.20/10.32	32.00/17.19	2054.80/837.10
SAF	19.30/11.55	20.90/18.78	17.70/4.38	18.60/4.70	18.80/5.36	18.90/6.04	18.10/6.12	43.30/18.43

ChebNetII, JacobiConv), two non-local GNNs (NodeFormer, GloGNN++), and one unified GNN model (ClenshawGCN), as detailed in Table 6.7. One can observe that SAF, while slightly slower than its base model (BernNet) due to the integration of non-local spatial aggregation, remains more efficient than or comparable to other SOTA methods. On the other hand, SAF-ϵ costs more time (but still faster than the non-local GNN, GloGNN++), a consequence of its quadratic complexity from non-local sparsification. However, this trade-off allows SAF-ϵ to produce an high-quality new graph, better capturing long-range interconnections among nodes and addressing graph heterophily.

6.7.5 Limitations and Future Work

In this subsection, we critically examine the limitations of our proposed method. While our model has shown considerable success across diverse benchmarks, it remains essential to recognize and address its inherent limitations. In what follows, we highlight these limitations and propose directions for future research that could potentially improve both the effectiveness and broader applicability of the model.

- Our model employs non-negative constraints on the proposed SAF framework applied to graph filters. While this approach ensures stability and interpretability, it may restrict the model's expressiveness, thereby limiting its ability to capture more complex patterns within the data. There is a need for theoretical refinement to explore whether relaxing these constraints could lead to enhanced model performance without sacrificing stability.

- Although this study focuses on a node-level investigation, it raises intriguing questions about the implications of spectral GNNs at the graph-level in the spatial domain. Future work could expand this examination by relaxing theoretical constraints or exploring the cross-domain interplay from a broader graph-level viewpoint.

6.8 Conclusion

This paper introduces a fresh spatial perspective on spectral GNNs, shedding light on their interpretability. We reveal that spectral GNNs fundamentally leads the original graph to an adapted new one, which exhibits non-locality and accommodates signed edge weights. This insight leads to our proposed Spatially Adaptive Filtering (SAF) framework, enhancing spectral GNNs for more effective and versatile graph representation learning.

Chapter 7

Conclusion and Future Work

In this thesis, we embarked on a comprehensive journey to explore and advance Graph Neural Networks (GNNs) in the context of complex structured data. We delved into the challenges posed by intricate graph structures, such as *entangled node relationships* and *heterophilic linking patterns*, that conventional GNNs struggle to effectively capture and represent. Through this exploration, we not only identified critical gaps in the existing methodologies but also proposed innovative models designed to navigate and interpret the complexity inherent in such data. The following sections summarize the key contributions of this thesis and outline interesting directions for future work.

7.1 Conclusion

This thesis unfolds four major components, each dedicated to exploring different dimensions of Graph Neural Networks within complex structured data environments. We delve into the fundamental framework of GNNs across both spatial and spectral domains, assessing their model capacity and highlighting how distinct features of diverse real-world graphs can act as powerful inductive biases. Contributions are made in advancing GNNs in term of both performance and interpretability, enhancing their ability to decode complex graph data.

Particularly, we have introduced the Local-Global Disentanglement (LGD) framework,

specifically designed to address the *entangled node relationships* in complex structured data. Our approach uniquely combines neighborhood routing at the local scale with a global message passing, aiming to infer the latent factors behind node connections. This strategy not only addresses the core challenge of entanglement but also promotes inter-factor diversity and intra-factor consistency on the disentangled representations. Moreover, our evaluations on both real-world and synthetic datasets demonstrate the superiority of LGD in both quantitative and qualitative aspects.

Then, we investigate the substantial outcomes of *entangled node relationships*, focusing on *heterophilic linking patterns* where most connected nodes belong to different classes. Such a scenario starkly contrasts with the homophily principle that underpins traditional GNNs, introducing significant modeling challenges. To navigate this complexity, we propose the Edge Splitting (ES-) GNN framework, specifically designed for heterophilic graphs. ES-GNN dynamically partitions graphs into subgraphs based on the edge relevance for targeted spatial aggregation, efficiently disentangling the task-relevant information from the noisy, irrelevant data. Empirically, we have shown ES-GNN's robust performance across various datasets, notably surpassing existing models, especially in environments featured by strong heterophily.

Furthermore, we delve deeper into the nuances of *heterophilic linking patterns* and uncovered a notable presence of regional heterogeneity. While certain spectral GNNs is able to learn from arbitrary label patterns with adaptive filters, we pinpoint a fundamental limitation: homogeneous spectral filtering often fails to capture local structural variations in complex networks. This observation leads to the development of our novel Diverse Spectral Filtering (DSF) framework. DSF revolutionizes spectral GNNs by incorporating node-specific filter weights, integrating both a global component for all nodes and distinct local elements to mirror regional disparities. This approach enables DSF to flexibly handles complex graph landscapes and achieves a proper balance between uniformity and diversity with enhanced interpretability. In practise, our DSF is applicable to any spectral GNNs and make considerable performance gains, especially in heterophilic graph settings.

Finally, we revisit current spectral GNNs from the spatial lens and revealed that these models, despite their spectral foundation, inadvertently alters spatial connectivity among nodes. This modification introduces non-locality and produces signed edge, depicting global relationship among node labels. Inspired by this discovery, we propose the Spatially Adaptive Filtering (SAF) framework, which exploits these identified properties to better capture long-range dependencies and address graph heterophily. This advancement not only enhances our ability to model complex graph structures but also deepens our understanding of the mechanisms driving GNN models. Our evaluations demonstrate that SAF can significantly improve the efficacy of spectral GNNs, with notable performance enhancements.

Discussion on Model Relationships This subsection summarizes the relationships among the models developed throughout this thesis, highlighting their interconnected advancements. Our initial work, as detailed in Chapter 3, tackles the challenge of *entangled node relationships*. Building on this foundation, the second work in Chapter 4 extends the disentangled representation learning paradigm to address *heterophilic linking patterns*, which are seen as a significant outcome of *entangled node relationships*. This exploration into *heterophilic linking patterns* led us further propose a shift in focus from the edge-level to the subgraph-level, uncovering regional disparities hidden beneath these patterns, i.e., notable differences in how nodes are structured across different graph regions. Continuing this trajectory, the third work in Chapter 5 investigates how to leverage this newfound prior graph knowledge to enhance graph learning. Specifically, it focuses on applying these insights to improve spectral GNNs' modeling capabilities on heterophilic graphs. As our previous studies primarily enhanced spectral GNNs within their native domain, the fourth work, presented in Chapter 6, proposes a cross-domain analysis of GNN models. This investigation into spectral GNNs in the spatial domain has revealed many overlooked cross-domain insights, which in turn are utilized by us to significantly enhance GNN performance in capturing long-range dependency and addressing graph heterophily. An intuitive visualization of the relationships among

these models, from both a problem-centric and model-centric perspective, is presented in Figure 1.3. This visual aid effectively showcases the depth and breadth of the research conducted in this thesis, tracing the evolutionary path of our studies in a cohesive manner.

Limitations In this subsection, we acknowledge the limitations of the research conducted in this thesis and outline areas that warrant further exploration. While our studies have provided significant insights, they are subject to certain constraints that could be expanded to enhance their relevance and applicability in broader contexts.

- **Node-level Transductive Setting:** The majority of our GNN research has been conducted within a transductive setting, focusing on node-level tasks where the graph structure is usually fixed. However, real-world applications often demand inductive inference ability, adapting to new and unseen data. Extending our methodologies to accomplish graph-level tasks not only has profound theoretical implications but is also vital for practical application in dynamic environments.
- **Application of Foundational Models:** Our work has primarily revolved around the foundational GNN models. While establishing a strong theoretical base is crucial, the true value of these models is often more fully realized when they are effectively applied to real-world problems. Future research should, therefore, focus on applying the insights and methodologies developed in this thesis to address practical issues encountered in everyday applications.

To address these limitations, detailed future research directions are proposed in Section 7.2. These recommendations aim to guide subsequent studies towards overcoming the identified challenges and enhancing the practical utility of our research findings.

7.2 Future Work

Building on the foundational insights and advancements detailed in this thesis, this section outlines several promising directions for future research.

Graph Robustness. The exploration of complex structured data in this thesis can be seen as adding noise to clean data, such as linking nodes from different classes as adversarial noise, transforming original homophilic graphs into heterophilic ones. Thus, our proposed methods inherently possess a degree of adversarial robustness, offering enhanced resilience compared to standard GNNs. While Chapter 3 tested the robustness of ES-GNN through random edge insertions, a systematic evaluation against various attack methods was not conducted. The robustness observed is more of a beneficial side effect than a direct objective. Investigating the potential relationship between complex graph learning and graph robustness, and leveraging techniques developed within this domain to bolster model robustness, should be a promising future direction. This could involve a more focused study on constructing GNNs that not only withstand adversarial attacks but also capitalize on the inherent noise within graph data to improve resilience.

Graph Anomaly Detection. The issue of graph heterophily, central to this thesis, also features prominently in graph anomaly detection tasks, such as identifying fraudulent activities within banking systems, where fraudsters often connect with regular users. Unlike general heterophilic graphs, transaction networks may exhibit a significant class imbalance, with a vast majority of normal users and a small fraction of anomalous ones. This imbalance, coupled with the mixing pattern of links that may obscure the activities of fraudsters, adds complexity to the detection task. Adapting GNNs tailored for heterophilic graphs to overcome these challenges and effectively detect anomalies in such skewed environments represents an intriguing future direction. Exploring solutions that address class imbalance and enhance the detection of anomalous patterns within heterophilic linking contexts could significantly advance the field.

Deep Geometric Learning. This thesis primarily investigates node-level tasks, such as node classification through transductive learning, where the graph structure is fixed, and only a subset of labels is missing. However, real-world applications often demand inductive inference capabilities, for instance, in deep geometric learning such as drug discovery, where the goal is to predict the properties of new molecules not seen during training. Expanding the methodologies and insights from this thesis to graph-level tasks

and enhancing inductive learning capabilities hold profound theoretical and practical significance. Exploring how to generalize the proposed models to accommodate unseen graphs and applying these advancements to geometric learning domains like drug discovery and 3D point cloud could open new avenues for impactful research.

Publications

1. **Guo, J.**, Huang, K., Yi, X. and Zhang, R., 2022. Learning disentangled graph convolutional networks locally and globally. IEEE Transactions on Neural Networks and Learning Systems (TNNLS).

2. **Guo, J.**, Huang, K., Yi, X. and Zhang, R., 2023, April. Graph Neural Networks with Diverse Spectral Filtering. In Proceedings of the ACM Web Conference 2023 (WWW).

3. **Guo, J.**, Huang, K., Yi, X. and Zhang, R., 2024. ES-GNN: Generalizing Graph Neural Networks Beyond Homophily with Edge Splitting. IEEE transactions on pattern analysis and machine intelligence (Under Revision, TPAMI).

4. **Guo, J.**, Huang, K., Yi, X., Su, Z. and Zhang, R., 2024. Rethinking Spectral Graph Neural Networks with Spatially Adaptive Filtering. arXiv preprint arXiv:2401.09071.

5. Su, Z., **Guo, J.**, Yao, K., Yang, X., Wang, Q. and Huang, K., 2024. Unraveling Batch Normalization for Realistic Test-Time Adaptation. In Proceedings of the AAAI Conference on Artificial Intelligence (AAAI).

Appendix

This appendix provides a comprehensive reassessment of the models introduced in this thesis, namely LGD (Chapter 3), ES-GNN (Chapter 4), DSF (Chapter 5), and SAF (Chapter 6). Despite these models being designed for identical tasks and evaluated on largely overlapping datasets, discrepancies in experimental settings arose due to the need to compare with contemporary baseline methods in line with their experimental protocols. To ensure a comprehensive evaluation of our proposed models, we have reevaluated them following the model evaluation criteria detailed in Chapter 6. This reevaluation encompasses seven real-world datasets under both semi-supervised (with approximately 2.5%/2.5%/95% data splits) and full-supervised (60%/20%/20% data splits) settings. The detailed results are respectively provided in Tables 7.1 and 7.2. Specifically, for the DSF model, we selected BernNet [37] as its base model to maintain consistency with our proposed SAF. For time efficiency, we only examine its regularized version, DSF-Bern-R, still referred to as DSF in the following Tables.

Table 7.1: Semi-supervised node classification accuracy (%) ± 95% confidence interval.

Method	**Cham.**	**Squi.**	**Texas**	**Corn.**	**Cora**	**Cite.**	**Pubm.**
MLP	26.36±2.85	21.42±1.50	32.42±9.91	36.53±7.92	57.17±1.34	56.75±1.55	70.52±2.01
GCN	38.15±3.77	31.18±0.93	34.68±9.07	32.36±8.55	79.19±1.37	69.71±1.32	78.81±0.84
ARMA	37.42±1.72	24.15±0.93	39.65±8.09	28.90±10.07	79.14±1.07	69.35±1.44	78.31±1.33
APPNP	32.73±2.31	24.50±0.89	34.79±10.11	34.85±9.71	82.39±0.68	69.79±0.92	79.97±1.58
GPR-GNN	33.03±1.92	24.36±1.52	33.98±11.90	38.95±12.36	82.37±0.91	69.22±1.27	79.28±2.25
BernNet	27.32±4.04	22.37±0.98	43.01±7.45	39.42±9.59	82.17±0.86	69.44±0.97	79.48±1.47
ChebNetII	43.42±3.54	33.96±1.22	46.58±7.68	42.19±11.61	82.42±0.64	69.89±1.21	79.51±1.03
LGD	38.72±1.88	27.49±1.66	35.49±11.39	39.19±9.54	81.03±1.22	70.00±1.19	80.75±1.07
ES-GNN	39.43±1.77	31.82±0.89	47.98±8.04	45.90±7.23	82.48±0.66	70.14±0.97	78.96±1.05
DSF	38.97±2.20	28.90±1.12	57.23±5.15	43.06±10.87	83.01±0.94	70.29±1.26	79.71±0.85
SAF	41.82±1.74	31.77±0.69	58.04±3.76	52.49±8.56	83.57±0.66	71.07±1.08	79.51±1.12

Table 7.2: Full-supervised node classification accuracy (%) ± 95% confidence interval.

Method	**Cham.**	**Squi.**	**Texas**	**Corn.**	**Cora**	**Cite.**	**Pubm.**
MLP	46.59±1.84	31.01±1.18	86.81±2.24	84.15±3.05	76.89±0.97	76.52±0.89	86.14±0.25
GCN	60.81±2.95	45.87±0.88	76.97±3.97	65.78±4.16	87.18±1.12	79.85±0.78	86.79±0.31
ARMA	60.21±1.00	36.27±0.62	83.97±3.77	85.62±2.13	87.13±0.80	80.04±0.55	86.93±0.24
APPNP	52.15±1.79	35.71±0.78	90.64±1.70	91.52±1.81	88.16±0.74	80.47±0.73	88.13±0.33
GPR-GNN	67.49±1.38	50.43±1.89	92.91±1.32	91.57±1.96	88.54±0.67	80.13±0.84	88.46±0.31
BernNet	68.53±1.68	51.39±0.92	92.62±1.37	92.13±1.64	88.51±0.92	80.08±0.75	88.51±0.39
ChebNetII	71.37±1.01	57.72±0.59	93.28±1.47	92.30±1.48	88.71±0.93	80.53±0.79	88.93±0.29
LGD	65.34±0.99	43.22±1.21	82.46±4.10	84.26±3.77	88.44±0.89	80.26±1.00	87.76±0.60
ES-GNN	71.79±0.83	59.15±1.76	86.39±3.28	82.13±2.95	89.49±0.59	80.19±0.90	90.89±0.31
DSF	73.17±0.81	59.35±1.03	94.26±1.15	92.30±2.30	89.48±0.58	80.46±0.79	91.72±0.29
SAF	75.30±0.96	63.63±0.81	94.10±1.48	92.95±1.97	89.80±0.69	80.61±0.81	91.49±0.29

Bibliography

[1] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu, "Disentangled graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4212–4221.

[2] Y. Liu, X. Wang, S. Wu, and Z. Xiao, "Independence promoted graph disentangled networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4916–4923.

[3] Y. Yang, Z. Feng, M. Song, and X. Wang, "Factorizable graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[4] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7793–7804, 2020.

[5] D. Lim, X. Li, F. Hohne, and S.-N. Lim, "New benchmarks for learning on non-homophilous graphs," *arXiv preprint arXiv:2104.01404*, 2021.

[6] T. Gärtner, P. Flach, and S. Wrobel, "On graph kernels: Hardness results and efficient alternatives," in *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*. Springer, 2003, pp. 129–143.

[7] K. M. Borgwardt and H.-P. Kriegel, "Shortest-path kernels on graphs," in *Fifth IEEE International Conference on Data Mining*. IEEE, 2005, pp. 8–pp.

[8] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, 2015, pp. 891–900.

[9] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1105–1114.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710.

[11] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 855–864.

[12] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 27–34.

[13] T. Chen and R. C.-W. Wong, "Handling information loss of graph neural networks for session-based recommendation," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1172–1180.

[14] Y. Wu, D. Lian, Y. Xu, L. Wu, and E. Chen, "Graph convolutional networks with markov random field reasoning for social spammer detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 1054–1061.

[15] T. Bian, X. Xiao, T. Xu, P. Zhao, W. Huang, Y. Rong, and J. Huang, "Rumor detection on social media with bi-directional graph convolutional networks," in

Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 01, 2020, pp. 549–556.

[16] H. Dai, C. Li, C. Coley, B. Dai, and L. Song, "Retrosynthesis prediction with conditional graph logic network," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] J. Bradshaw, M. J. Kusner, B. Paige, M. H. Segler, and J. M. Hernández-Lobato, "A generative model for electron paths," *arXiv preprint arXiv:1805.10970*, 2018.

[18] F. Alet, A. K. Jeewajee, M. B. Villalonga, A. Rodriguez, T. Lozano-Perez, and L. Kaelbling, "Graph element networks: Adaptive, structured computation and memory," in *International Conference on Machine Learning*. PMLR, 2019, pp. 212–222.

[19] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," in *International Conference on Machine Learning*. PMLR, 2018, pp. 2688–2697.

[20] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1263–1272.

[21] F. R. K. Chung, "Spectral graph theory," 1996.

[22] J. Zhu, R. A. Rossi, A. Rao, T. Mai, N. Lipka, N. K. Ahmed, and D. Koutra, "Graph neural networks with heterophily," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021, pp. 11 168–11 176.

[23] B. Bollobás, *Extremal graph theory*. Courier Corporation, 2004.

[24] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[25] N. Shervashidze, S. Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt, "Efficient graphlet kernels for large graph comparison," in *Artificial Intelligence and Statistics*. PMLR, 2009, pp. 488–495.

[26] N. Shervashidze, P. Schweitzer, E. J. Van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, no. 9, 2011.

[27] C. Yang, M. Sun, Z. Liu, and C. Tu, "Fast network embedding enhancement via high order proximity approximation," in *Twenty-Sixth International Joint Conference on Artificial Intelligence*, vol. 17, 2017, pp. 3894–3900.

[28] Y. Yin and Z. Wei, "Scalable graph embeddings via sparse transpose proximities," in *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019, pp. 1429–1437.

[29] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "Harp: Hierarchical representation learning for networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[30] S. Chanpuriya and C. Musco, "Infinitewalk: Deep network embeddings as laplacian embeddings with a nonlinearity," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1325–1333.

[31] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234.

[32] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[33] H. Gao and H. Huang, "Deep attributed network embedding," in *Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018.

[34] Z. Meng, S. Liang, H. Bao, and X. Zhang, "Co-embedding attributed networks," in *Proceedings of the twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 393–401.

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

[36] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *International Conference on Learning Representations*, 2019.

[37] M. He, Z. Wei, H. Xu *et al.*, "Bernnet: Learning arbitrary graph spectral filters via bernstein approximation," *Advances in Neural Information Processing Systems*, vol. 34, pp. 14 239–14 251, 2021.

[38] W. Ju, Z. Fang, Y. Gu, Z. Liu, Q. Long, Z. Qiao, Y. Qin, J. Shen, F. Sun, Z. Xiao *et al.*, "A comprehensive survey on deep graph representation learning," *Neural Networks*, p. 106207, 2024.

[39] S. Khoshraftar and A. An, "A survey on graph representation learning methods," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 1, pp. 1–55, 2024.

[40] Z. Chen, F. Chen, L. Zhang, T. Ji, K. Fu, L. Zhao, F. Chen, L. Wu, C. Aggarwal, and C.-T. Lu, "Bridging the gap between spatial and spectral domains: A unified framework for graph neural networks," *arXiv preprint arXiv:2107.10234*, 2021.

[41] I. Chami, S. Abu-El-Haija, B. Perozzi, C. Ré, and K. Murphy, "Machine learning on graphs: A model and comprehensive taxonomy," *Journal of Machine Learning Research*, vol. 23, no. 89, pp. 1–64, 2022.

[42] N. M. Kriege, F. D. Johansson, and C. Morris, "A survey on graph kernels," *Applied Network Science*, vol. 5, no. 1, pp. 1–42, 2020.

[43] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[44] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249–270, 2020.

[45] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4–24, 2021.

[46] P. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Çaglar Gülçehre, H. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *ArXiv*, vol. abs/1806.01261, 2018.

[47] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[48] B. Schölkopf and A. J. Smola, *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[49] G. F. Lawler and V. Limic, *Random walk: A modern introduction*. Cambridge University Press, 2010, vol. 123.

[50] Z. Chen, F. Chen, L. Zhang, T. Ji, K. Fu, L. Zhao, F. Chen, L. Wu, C. Aggarwal, and C.-T. Lu, "Bridging the gap between spatial and spectral domains: A unified framework for graph neural networks," *ACM Computing Surveys*, vol. 56, no. 5, pp. 1–42, 2023.

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

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

[53] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," *International Conference on Learning Representations*, 2018.

[54] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized pagerank graph neural network," in *International Conference on Learning Representations*, 2021.

[55] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[56] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," *2017 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5425–5434, 2017.

[57] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5453–5462.

[58] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6861–6871.

[59] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 1725–1735.

[60] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," *arXiv preprint arXiv:2002.05287*, 2020.

[61] D. Bo, X. Wang, C. Shi, and H. Shen, "Beyond low-frequency information in graph convolutional networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021, pp. 3950–3957.

[62] M. Liu, Z. Wang, and S. Ji, "Non-local graph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 10 270–10 276, 2021.

[63] R. Wang, S. Mou, X. Wang, W. Xiao, Q. Ju, C. Shi, and X. Xie, "Graph structure estimation neural networks," in *Proceedings of the Web Conference 2021*, 2021, pp. 342–353.

[64] S. Suresh, V. Budde, J. Neville, P. Li, and J. Ma, "Breaking the limit of graph neural networks by improving the assortativity of graphs with local mixing patterns," *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021.

[65] M. Eliasof, E. Haber, and E. Treister, "Pde-gcn: novel architectures for graph neural networks motivated by partial differential equations," *Advances in Neural Information Processing Systems*, vol. 34, pp. 3836–3849, 2021.

[66] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, "Graph neural networks with convolutional arma filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[67] M. Zhu, X. Wang, C. Shi, H. Ji, and P. Cui, "Interpreting and unifying graph neural networks with an optimization framework," in *Proceedings of the Web Conference 2021*, 2021, pp. 1215–1226.

[68] Y. Ma, X. Liu, T. Zhao, Y. Liu, J. Tang, and N. Shah, "A unified view on graph neural networks as graph signal denoising," in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 2021, pp. 1202–1211.

[69] J. Guo, K. Huang, X. Yi, and R. Zhang, "Learning disentangled graph convolutional networks locally and globally," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[70] ——, "Es-gnn: Generalizing graph neural networks beyond homophily with edge splitting," *arXiv preprint arXiv:2205.13700*, 2022.

[71] Q. Wu, W. Zhao, Z. Li, D. P. Wipf, and J. Yan, "Nodeformer: A scalable graph structure learning transformer for node classification," *Advances in Neural Information Processing Systems*, vol. 35, pp. 27 387–27 401, 2022.

[72] X. Li, R. Zhu, Y. Cheng, C. Shan, S. Luo, D. Li, and W. Qian, "Finding global homophily in graph neural networks when meeting heterophily," in *International Conference on Machine Learning*. PMLR, 2022, pp. 13 242–13 256.

[73] Y. Yang, Y. Liang, and M. Zhang, "Pa-gnn: Parameter-adaptive graph neural networks."

[74] X. Wang and M. Zhang, "How powerful are spectral graph neural networks," in *International Conference on Machine Learning*. PMLR, 2022, pp. 23 341–23 362.

[75] M. He, Z. Wei, and J.-R. Wen, "Convolutional neural networks on graphs with chebyshev approximation, revisited," *Advances in neural information processing systems*, vol. 35, pp. 7264–7276, 2022.

[76] G. Cui and Z. Wei, "Mgnn: Graph neural networks inspired by distance geometry problem," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 335–347.

[77] J. Guo, K. Huang, X. Yi, and R. Zhang, "Graph neural networks with diverse spectral filtering," in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 306–316.

[78] D. Bo, C. Shi, L. Wang, and R. Liao, "Specformer: Spectral graph neural networks meet transformers," *arXiv preprint arXiv:2303.01028*, 2023.

[79] Q. Tao, Z. Wang, W. Yu, Y. Li, and Z. Wei, "Longnn: Spectral gnns with learnable orthonormal basis," *arXiv preprint arXiv:2303.13750*, 2023.

[80] Y. Guo and Z. Wei, "Graph neural networks with learnable and optimal polynomial bases," in *International Conference on Machine Learning*. PMLR, 2023, pp. 12 077–12 097.

[81] ——, "Clenshaw graph neural networks," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 614–625.

[82] J. Sun, L. Zhang, G. Chen, P. Xu, K. Zhang, and Y. Yang, "Feature expansion for graph neural networks," in *International Conference on Machine Learning*. PMLR, 2023, pp. 33 156–33 176.

[83] J. Guo, K. Huang, X. Yi, Z. Su, and R. Zhang, "Rethinking spectral graph neural networks with spatially adaptive filtering," *arXiv preprint arXiv:2401.09071*, 2024.

[84] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," *Advances in neural information processing systems*, vol. 28, 2015.

[85] B. Paige, J.-W. Van De Meent, A. Desmaison, N. Goodman, P. Kohli, F. Wood, P. Torr *et al.*, "Learning disentangled representations with semi-supervised deep generative models," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[86] R. Lopez, J. Regier, M. I. Jordan, and N. Yosef, "Information constraints on auto-encoding variational bayes," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[87] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *International Conference on Learning Representations*, 2017.

[88] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," *Advances in neural information processing systems*, vol. 29, 2016.

[89] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[90] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel, "Variational lossy autoencoder," *ArXiv*, vol. abs/1611.02731, 2017.

[91] A. A. Alemi, I. S. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *ArXiv*, vol. abs/1612.00410, 2017.

[92] L. Hu, S. Xu, C. Li, C. Yang, C. Shi, N. Duan, X. Xie, and M. Zhou, "Graph neural news recommendation with unsupervised preference disentanglement," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 4255–4264.

[93] X. Wang, H. Jin, A. Zhang, X. He, T. Xu, and T.-S. Chua, "Disentangled graph collaborative filtering," *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[94] X. Wang, R. Wang, C. Shi, G. Song, and Q. Li, "Multi-component graph convolutional collaborative filtering," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6267–6274.

[95] Y. Liu, X. Wang, S. Wu, and Z. Xiao, "Independence promoted graph disentangled networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4916–4923.

[96] H. Nt and T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters," *arXiv preprint arXiv:1905.09550*, 2019.

[97] Y. Min, F. Wenkel, and G. Wolf, "Scattering gcn: Overcoming oversmoothness in graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 498–14 508, 2020.

[98] M. Balcilar, R. Guillaume, P. Héroux, B. Gaüzère, S. Adam, and P. Honeine, "Analyzing the expressive power of graph neural networks in a spectral perspective," in *International Conference on Learning Representations*, 2021.

[99] Y. Hou, J. Zhang, J. Cheng, K. Ma, R. T. Ma, H. Chen, and M.-C. Yang, "Measuring and improving the use of graph information in graph neural networks," in *International Conference on Learning Representations*, 2019.

[100] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[101] X. Li, B. Kao, C. Shan, D. Yin, and M. Ester, "Cast: A correlation-based adaptive spectral clustering algorithm on multi-scale data," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 439–449.

[102] Y.-M. Zhang, K. Huang, G.-G. Geng, and C.-L. Liu, "Mtc: A fast and robust graph-based transductive learning method," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 9, pp. 1979–1991, 2015.

[103] M. Shi, Y. Tang, and X. Zhu, "Mlne: Multi-label network embedding," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3682–3695, 2020.

[104] U. S. Shanthamallu, J. J. Thiagarajan, H. Song, and A. Spanias, "Gramme: Semisupervised learning using multilayered graph attention models," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 10, pp. 3977–3988, 2020.

[105] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge

graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021.

[106] J. Wu, S. Pan, X. Zhu, C. Zhang, and P. S. Yu, "Multiple structure-view learning for graph classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 7, pp. 3236–3251, 2018.

[107] K. Huang, H. Yang, I. King, and M. Lyu, "Modeling data locally and globally," *Springer Verlag ISBN-13: 978-3540794516*, 2008.

[108] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," *arXiv preprint arXiv:1809.10341*, 2018.

[109] H. Li, B. Wang, L. Cui, L. Bai, and E. R. Hancock, "Lgl-gnn: Learning global and local information for graph neural networks," in *Structural, Syntactic, and Statistical Pattern Recognition*. Springer International Publishing AG, 2021, pp. 129–138.

[110] Z. Ghahramani and G. E. Hinton, "The em algorithm for mixtures of factor analyzers," 1996.

[111] L. Hajderanj, I. Weheliye, and D. Chen, "A new supervised t-sne with dissimilarity measure for effective data visualization and classification," in *Proceedings of the 2019 8th International Conference on Software and Information Engineering*, 2019, pp. 232–236.

[112] D. Luo, W. Cheng, W. Yu, B. Zong, J. Ni, H. Chen, and X. Zhang, "Learning to drop: Robust graph neural network via topological denoising," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021, pp. 779–787.

[113] Y. Zhu, W. Xu, J. Zhang, Q. Liu, S. Wu, and L. Wang, "Deep graph structure learning for robust representations: A survey," *arXiv preprint arXiv:2103.03036*, 2021.

[114] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[115] D. Yu, R. Zhang, Z. Jiang, Y. Wu, and Y. Yang, "Graph-revised convolutional network," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part III*. Springer, 2021, pp. 378–393.

[116] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 313–11 320.

[117] Y. Chen, L. Wu, and M. Zaki, "Iterative deep graph learning for graph neural networks: Better and robust node embeddings," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[118] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1972–1982.

[119] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang, "Robust graph representation learning via neural sparsification," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 458–11 468.

[120] T. Zhao, Y. Liu, L. Neves, O. J. Woodford, M. Jiang, and N. Shah, "Data augmentation for graph neural networks," in *AAAI*, 2021.

[121] L. V. D. Maaten and G. E. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.

[122] M. Vlachos, C. Domeniconi, D. Gunopulos, G. Kollios, and N. Koudas, "Non-linear dimensionality reduction techniques for classification and visualization," in

Proceedings of the 8th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2002, pp. 645–651.

[123] X. Geng, D.-C. Zhan, and Z.-H. Zhou, "Supervised nonlinear dimensionality reduction for visualization and classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 6, pp. 1098–1107, 2005.

[124] S.-q. Zhang, "Enhanced supervised locally linear embedding," *Pattern Recognition Letters*, vol. 30, no. 13, pp. 1208–1218, 2009.

[125] J. Cheng, H. Liu, F. Wang, H. Li, and C. Zhu, "Silhouette analysis for human action recognition based on supervised temporal t-sne and incremental learning," *IEEE Transactions on Image Processing*, vol. 24, no. 10, pp. 3203–3217, 2015.

[126] M. Yu, S. Zhang, L. Zhao, and G. Kuang, "Deep supervised t-sne for sar target recognition," in *2017 2nd International Conference on Frontiers of Sensors Technologies*. IEEE, 2017, pp. 265–269.

[127] R. De Maesschalck, D. Jouan-Rimbaud, and D. L. Massart, "The mahalanobis distance," *Chemometrics and intelligent laboratory systems*, vol. 50, no. 1, pp. 1–18, 2000.

[128] P. Xie, "Diversity-promoting and large-scale machine learning for healthcare," 2018.

[129] B. Xie, Y. Liang, and L. Song, "Diversity leads to generalization in neural networks," *ArXiv*, vol. abs/1611.03131, 2016.

[130] A. Kulesza and B. Taskar, "Determinantal point processes for machine learning," *Found. Trends Mach. Learn.*, vol. 5, pp. 123–286, 2012.

[131] D. Bernstein, "Matrix mathematics: Theory, facts, and formulas with application to linear systems theory," 2005.

[132] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[133] T. Berry and T. Sauer, "Consistent manifold representation for topological data analysis," *Foundations of Data Science*, 2019.

[134] Z. Liu and M. Barahona, "Graph-based data clustering via multiscale community detection," *Applied Network Science*, vol. 5, no. 1, pp. 1–20, 2020.

[135] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[136] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 499–515.

[137] S. Zhang, K. Huang, J. Zhu, and Y. Liu, "Manifold adversarial training for supervised and semi-supervised learning," *Neural Networks*, vol. 140, pp. 282–293, 2021.

[138] S. Zhang, Z. Qian, K. Huang, Q. Wang, R. Zhang, and X. Yi, "Towards better robust generalization with shift consistency regularization," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12 524–12 534.

[139] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, 2017, pp. 731–739.

[140] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[141] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2019, pp. 2623–2631.

[142] G. Ciano, A. Rossi, M. Bianchini, and F. Scarselli, "On inductive–transductive learning with graph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 2, pp. 758–769, 2021.

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

[144] L. Yang, W. Zhou, W. Peng, B. Niu, J. Gu, C. Wang, X. Cao, and D. He, "Graph neural networks beyond compromise between attribute and topology," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1127–1135.

[145] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: Homophily in social networks," *Annual review of sociology*, vol. 27, no. 1, pp. 415–444, 2001.

[146] Y. Yan, M. Hashemi, K. Swersky, Y. Yang, and D. Koutra, "Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks," *arXiv preprint arXiv:2102.06462*, 2021.

[147] Z. Fang, L. Xu, G. Song, Q. Long, and Y. Zhang, "Polarized graph neural networks," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1404–1413.

[148] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," in *International Conference on Learning Representations*, 2020.

[149] D. Kim and A. Oh, "How to find your friendly neighborhood: Graph attention design with self-supervision," in *International Conference on Learning Representations*, 2021.

[150] D. Lim, F. Hohne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S. N. Lim, "Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 887–20 902, 2021.

[151] K.-H. Lai, D. Zha, K. Zhou, and X. Hu, "Policy-gnn: Aggregation optimization for graph neural networks," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 461–471.

[152] E. Isufi, F. Gama, and A. Ribeiro, "Edgenets: Edge varying graph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[153] Y. Gao, Y. Feng, S. Ji, and R. Ji, "HgnnΘ+: General hypergraph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[154] G. Bouritsas, F. Frasca, S. P. Zafeiriou, and M. Bronstein, "Improving graph neural network expressivity via subgraph isomorphism counting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[155] M. Balcilar, P. Héroux, B. Gauzere, P. Vasseur, S. Adam, and P. Honeine, "Breaking the limits of message passing graph neural networks," in *International Conference on Machine Learning*. PMLR, 2021, pp. 599–608.

[156] L. Faber, A. K. Moghaddam, and R. Wattenhofer, "When comparing to ground truth is wrong: On evaluating gnn explanation methods," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021, pp. 332–341.

[157] X. Wang, Y. Wu, A. Zhang, F. Feng, X. He, and T.-S. Chua, "Reinforced causal explainer for graph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[158] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schutt, K.-R. Muller, and G. Montavon, "Higher-order explanations of graph neural networks via relevant walks." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PP, 2021.

[159] I. Higgins, D. Amos, D. Pfau, S. Racaniere, L. Matthey, D. Rezende, and A. Ler-

chner, "Towards a definition of disentangled representations," *arXiv preprint arXiv:1812.02230*, 2018.

[160] L. Ma, Q. Sun, S. Georgoulis, L. Van Gool, B. Schiele, and M. Fritz, "Disentangled person image generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 99–108.

[161] Z. Zhang, L. Tran, F. Liu, and X. Liu, "On learning disentangled representations for gait recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[162] C. Eom, W. Lee, G. Lee, and B. Ham, "Is-gan: Learning disentangled representation for robust person re-identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[163] H. Li, X. Wang, Z. Zhang, Z. Yuan, H. Li, and W. Zhu, "Disentangled contrastive learning on graphs," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 872–21 884, 2021.

[164] X. Guo, L. Zhao, Z. Qin, L. Wu, A. Shehu, and Y. Ye, "Interpretable deep graph generation with node-edge co-disentanglement," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1697–1707.

[165] X. Guo, Y. Du, and L. Zhao, "Deep generative models for spatial networks," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021, pp. 505–515.

[166] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[167] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[168] O. Stretcu, K. Viswanathan, D. Movshovitz-Attias, E. Platanios, S. Ravi, and A. Tomkins, "Graph agreement models for semi-supervised learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[169] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[170] Y. Deshpande, S. Sen, A. Montanari, and E. Mossel, "Contextual stochastic block models," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[171] J. Palowitch, A. Tsitsulin, B. Perozzi, and B. A. Mayer, "Synthetic graph generation to benchmark graph learning," in *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*, 2022.

[172] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," *Journal of Complex Networks*, vol. 9, no. 2, p. cnab014, 2021.

[173] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2009, pp. 807–816.

[174] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: Divided they blog," in *Proceedings of the 3rd International Workshop on Link Discovery*, 2005, pp. 36–43.

[175] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 66–74.

[176] Y. Ma, X. Liu, N. Shah, and J. Tang, "Is homophily a necessity for graph neural networks?" *ArXiv*, vol. abs/2106.06134, 2021.

[177] V. Lingam, C. Ekbote, M. Sharma, R. Ragesh, A. Iyer, and S. Sellamanickam, "A piece-wise polynomial filtering approach for graph neural networks," *arXiv preprint arXiv:2112.03499*, 2021.

[178] U. Alon and E. Yahav, "On the bottleneck of graph neural networks and its practical implications," in *International Conference on Learning Representations*, 2021.

[179] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[180] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[181] Y. Dong, K. Ding, B. Jalaian, S. Ji, and J. Li, "Adagnn: Graph neural networks with adaptive frequency response filter," in *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, 2021, pp. 392–401.

[182] L. Yang, M. Li, L. Liu, C. Wang, X. Cao, Y. Guo *et al.*, "Diverse message passing for attribute with heterophily," *Advances in Neural Information Processing Systems*, vol. 34, pp. 4751–4763, 2021.

[183] X. Ma, Q. Chen, Y. Ren, G. Song, and L. Wang, "Meta-weight graph neural network: Push the limits beyond global homophily," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1270–1280.

[184] Y. Han, G. Huang, S. Song, L. Yang, H. Wang, and Y. Wang, "Dynamic neural networks: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[185] M. A. Islam, S. Jia, and N. D. Bruce, "How much position information do convolutional neural networks encode?" *arXiv preprint arXiv:2001.08248*, 2020.

[186] A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[187] P. Dufter, M. Schmitt, and H. Schütze, "Position information in transformers: An overview," *Computational Linguistics*, vol. 48, no. 3, pp. 733–763, 2022.

[188] H. Wang, H. Yin, M. Zhang, and P. Li, "Equivariant and stable positional encoding for more powerful graph neural networks," in *International Conference on Learning Representations*, 2022.

[189] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Graph neural networks with learnable structural and positional representations," in *International Conference on Learning Representations*, 2022.

[190] B. Weisfeiler and A. Leman, "The reduction of a graph to canonical form and the algebra which appears therein," *NTI, Series*, vol. 2, no. 9, pp. 12–16, 1968.

[191] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 4602–4609.

[192] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[193] R. Lai and S. Osher, "A splitting method for orthogonality constrained problems," *Journal of Scientific Computing*, vol. 58, no. 2, pp. 431–449, 2014.

[194] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 43–52.

[195] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," *Relational Representation Learning Workshop, NeurIPS 2018*, 2018.

[196] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[197] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[198] C. Gao, Y. Zheng, N. Li, Y. Li, Y. Qin, J. Piao, Y. Quan, J. Chang, D. Jin, X. He *et al.*, "A survey of graph neural networks for recommender systems: Challenges, methods, and directions," *ACM Transactions on Recommender Systems*, vol. 1, no. 1, pp. 1–51, 2023.

[199] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[200] M. Ju, S. Hou, Y. Fan, J. Zhao, Y. Ye, and L. Zhao, "Adaptive kernel graph neural network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6, 2022, pp. 7051–7058.

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

[202] R. Liao, Z. Zhao, R. Urtasun, and R. S. Zemel, "Lanczosnet: Multi-scale deep graph convolutional networks," *arXiv preprint arXiv:1901.01484*, 2019.

[203] W. Heisenberg, "Über den anschaulichen inhalt der quantentheoretischen kinematik und mechanik," *Zeitschrift für Physik*, vol. 43, no. 3-4, pp. 172–198, 1927.

[204] G. B. Folland and A. Sitaram, "The uncertainty principle: A mathematical survey," *Journal of Fourier analysis and applications*, vol. 3, pp. 207–238, 1997.

[205] A. Agaskar and Y. M. Lu, "A spectral graph uncertainty principle," *IEEE Transactions on Information Theory*, vol. 59, no. 7, pp. 4338–4356, 2013.

[206] Y. Yan, Y. Chen, H. Chen, M. Xu, M. Das, H. Yang, and H. Tong, "From trainable negative depth to edge heterophily in graphs," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[207] J. Yoo, M.-C. Lee, S. Shekhar, and C. Faloutsos, "Less is more: Slimg for accurate, robust, and interpretable graph mining," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 3128–3139.

[208] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," 1950.

[209] Y. Cai, G. Fang, and P. Li, "A note on sparse generalized eigenvalue problem," *Advances in Neural Information Processing Systems*, vol. 34, pp. 23 036–23 048, 2021.

[210] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *Journal of Machine Learning Research*, vol. 24, no. 43, pp. 1–48, 2023.

[211] Y. You, T. Chen, Z. Wang, and Y. Shen, "Graph domain adaptation via theory-grounded spectral regularization," in *International Conference on Learning Representations*, 2022.

[212] H. Chang, Y. Rong, T. Xu, Y. Bian, S. Zhou, X. Wang, J. Huang, and W. Zhu, "Not all low-pass filters are robust in graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 25 058–25 071, 2021.

[213] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 618–21 629, 2021.

[214] J. Kim, D. Nguyen, S. Min, S. Cho, M. Lee, H. Lee, and S. Hong, "Pure transformers are powerful graph learners," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14 582–14 595, 2022.

[215] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini, "Recipe for a general, powerful, scalable graph transformer," *Advances in Neural Information Processing Systems*, vol. 35, pp. 14 501–14 515, 2022.

[216] D. Lim, J. Robinson, L. Zhao, T. Smidt, S. Sra, H. Maron, and S. Jegelka, "Sign and basis invariant networks for spectral graph representation learning," *arXiv preprint arXiv:2202.13013*, 2022.

[217] D. Lim, J. Robinson, S. Jegelka, and H. Maron, "Expressive sign equivariant networks for spectral geometric learning," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[218] D. Spielman, "Spectral graph theory," *Combinatorial scientific computing*, vol. 18, p. 18, 2012.

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

[220] R. T. Farouki, "The bernstein polynomial basis: A centennial retrospective," *Computer Aided Geometric Design*, vol. 29, no. 6, pp. 379–419, 2012.

[221] V. Powers and B. Reznick, "Polynomials that are positive on an interval," *Transactions of the American Mathematical Society*, vol. 352, no. 10, pp. 4677–4692, 2000.

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

[223] O. Sagi and L. Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, p. e1249, 2018.

[224] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "Am-gcn: Adaptive multi-channel graph convolutional networks," in *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1243–1253.

[225] S. Zhu, S. Pan, C. Zhou, J. Wu, Y. Cao, and B. Wang, "Graph geometry interaction learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 7548–7558, 2020.

[226] O. Platonov, D. Kuznedelev, M. Diskin, A. Babenko, and L. Prokhorenkova, "A critical look at the evaluation of gnns under heterophily: are we really making progress?" *arXiv preprint arXiv:2302.11640*, 2023.

[227] O. Platonov, D. Kuznedelev, A. Babenko, and L. Prokhorenkova, "Characterizing graph datasets for node classification: Homophily-heterophily dichotomy and beyond," *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[228] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[229] Z. Yang, W. Cohen, and R. Salakhudinov, "Revisiting semi-supervised learning with graph embeddings," in *International Conference on Machine Learning*. PMLR, 2016, pp. 40–48.