Bridging the Gap between Spatial and Spectral Domains: A Unified Framework for Graph Neural Networks

ZHIQIAN CHEN, Computer Science and Engineering, Mississippi State University, U.S.A
FANGLAN CHEN and LEI ZHANG, Computer Science, Virginia Tech, U.S.A
TAORAN JI, Computer Science, Texas A&M University - Corpus Christi
KAIQUN FU, Electrical Engineering & Computer Science, South Dakota State University, U.S.A
LIANG ZHAO, Computer Science, Emory University, U.S.A
FENG CHEN, Computer Science, The University of Texas at Dallas, U.S.A
LINGFEI WU, Pinterest, U.S.A
CHARU AGGARWAL, IBM T. J. Watson Research Center, U.S.A
CHANG-TIEN LU, Computer Science, Virginia Tech, U.S.A

Deep learning's performance has been extensively recognized recently. Graph neural networks (GNNs) are designed to deal with graph-structural data that classical deep learning does not easily manage. Since most GNNs were created using distinct theories, direct comparisons are impossible. Prior research has primarily concentrated on categorizing existing models, with little attention paid to their intrinsic connections. The purpose of this study is to establish a unified framework that integrates GNNs based on spectral graph and approximation theory. The framework incorporates a strong integration between spatial- and spectral-based GNNs while tightly associating approaches that exist within each respective domain.

CCS Concepts: • **Mathematics of computing** → **Spectra of graphs**; *Graph algorithms*; **Approximation algorithms**; • **Computing methodologies** → *Spectral methods;*

Additional Key Words and Phrases: Deep learning, graph neural networks, approximation theory, spectral graph theory, graph learning

This work was funded by the NSF IIS award # 2153369.
Authors' addresses: Z. Chen, Dept. of Computer Science and Engineering, Mississippi State University, Butler Hall, MS State, MS, 39762; e-mail: zchen@cse.msstate.edu; F. Chen, L. Zhang, and C.-T. Lu, Computer Science, Virginia Tech, 7054 Haycock Road, Falls Church, VA, 22043; e-mails: {fanglanc, zhanglei}@vt.edu, ctlu@vt.edu; T. Ji, Texas A&M University- Corpus Christi; e-mail: taoran.ji@tamucc.edu; K. Fu, Electrical Engineering & Computer Science, South Dakota State University; e-mail: kaiqun.fu@sdstate.edu; L. Zhao, Computer Science, Emory University, Fairfax; e-mail: liang.zhao@emory.edu; F. Chen, Computer Science, The University of Texas at Dallas, ECSS 3.901 UTD, Dallas, TX; e-mail: feng.chen@utdallas.edu; L. Wu, Pinterest, Mountain View, CA, 94043; e-mail: lwu@email.wm.edu; C. Aggarwal, IBM T. J. Watson Research Center; e-mail: charu@us.ibm.com.

0360-0300/2023/12-ART126
https://doi.org/10.1145/3627816

ACM Reference format:
Zhiqian Chen, Fanglan Chen, Lei Zhang, Taoran Ji, Kaiqun Fu, Liang Zhao, Feng Chen, Lingfei Wu, Charu Aggarwal, and Chang-Tien Lu. 2023. Bridging the Gap between Spatial and Spectral Domains: A Unified Framework for Graph Neural Networks. *ACM Comput. Surv.* 56, 5, Article 126 (December 2023), 42 pages.
https://doi.org/10.1145/3627816

1 INTRODUCTION

Deep learning's performance in various machine learning tasks [1–6] has been extensively recognized in recent decades, with amazing success on Euclidean data. In recent decades, a slew of new applications have emerged in which effective information analysis boils down to the non-Euclidean geometry of data represented by a graph, such as social networks, transportation networks, spread of epidemic disease, brain's neuronal networks, gene data on biological regulatory networks, telecommunication networks, and knowledge graph. Previous deep learning algorithms, such as convolutional and recurrent neural networks, could not handle such non-Euclidean problems on graph-structured data. Modeling data using a graph is difficult, because graph data is irregular, i.e., each graph has a different number of nodes, and each node in a graph has a varied number of neighbors, making some operations like convolutions inapplicable to the network structure.

There has recently been a surge in growing interest in applying deep learning to graph data. Inspired by deep learning's success, principles from deep learning models are used to handle the graph's inherent complexity. This growing trend has piqued the interest of the machine learning community, and a huge number of **graph neural networks (GNN)** models have been proposed based on diverse theories [7–12] and grouped into coarse-grained groupings such as the spectral [13–17] and spatial [10–12] domains. GNNs have seen rising popularity recently for learning graph representations and quickly spread out to many application domains, such as physics [18, 19], chemistry [20, 21], knowledge graph [22–25], recommender systems [26–29], computer vision [30–32], natural language processing [33–35], combinatorial optimization [36–38], traffic network [39–42], program representation [43–45], social network [26, 46, 47]. However, current research in methodology has not translated into a clear understanding of the mechanisms involved, nor has it given us insight into GNNs' effectiveness or physical meaning. As a result, several consequences will occur: (1) There is no underlying principle that connects all GNNs, which also limits their growth. (2) In high-stakes applications such as drug development, GNN models may carry potentially hazardous unknowns, since they are black boxes. Consequently, the necessity of dissecting GNNs is highlighted, thereby driving academics to hunt for a more universal framework. The main problem is that existing GNN models use a variety of techniques, including random walks [48–50], PageRank [51–54], attention models [12, 55, 56], low-pass filters [57, 58], and message forwarding [59, 60]. Some preliminary research can only explain a few GNNs methods [59, 61, 62], leaving the majority of GNN unaccounted for. Previous GNNs surveys have dealt mostly with classifying several existing models into multiple categories and expanding on each category individually, with no regard to the interrelationships between them [13–17].

This research[1] aims to provide a coherent framework for generalizing GNNs by bridging the divide between seemingly unrelated works in the spatial and spectral domains, as well as by linking methods within each domain. The study will build a unified theoretical framework that encompasses diverse GNNs. Our research is novel in that it connects disparate GNN models, allowing for direct rethinking and comparison of all GNN models.

[1]A short version is available at Reference [63].

1.1 Graph Neural Networks in Spatial and Spectral Domain

Over the past several years, GNNs have gained a lot of attention. However, the existence of numerous GNNs complicates model selection, because they are not easily understood within the same framework. Specifically, one uses spectral theory to implement early GNNs [9, 64], whereas spatial theory is used to propose others [10, 65]. The mismatch inherent to spectral and spatial approaches means that direct comparisons are difficult. Even in each area, there are numerous models, which makes it difficult to examine their strengths and weaknesses.

To untangle the mess, we present a unified framework that connects the spatial and spectral domains and reveals their intricate relationship. Furthermore, both domains' subcategories are proven to have a hierarchical link. The focus on a unified framework adds to the knowledge of how GNNs operate. The goal of this research is to use a combination of spectral graph theory and approximation theory to investigate the relationship between important categories, such as spatial- and spectral-based approaches. We give a detailed analysis of GNNs' current research findings in this article, as well as a discussion of trending topics such as over-smoothing. Many well-known GNNs will be used to demonstrate the universality of our architecture. This article's main motivation is twofold: **(1) Connecting the spectral and spatial domains.** The fundamental concepts, principles, and physical implications of spectral- and spatial-based GNNs are significantly different due to their distinct features. As a result, we present an overview of the fundamental principles and properties of spectral- and spatial-based GNNs to help people better grasp the problems, potential, and necessity of GNNs. Formally, a rigorous equivalence is established, indicating that spatial connection function is comparable to spectral filtering; **(2) Dissecting spectral and spatial domains, respectively.** In spectral techniques, filtering functions on eigenvalues are examined, and the filtering function choice can be matched with various tactics in approximation theory; while spatial methodologies are used to explore attribute aggregation, which may be understood from the size and direction within a predetermined region.

The structure of the article is summarized as follows: Basic concepts, distinctive principles, and properties of graph neural networks are covered in Section 2, as well as ways for encoding the graph, spectral-based GNNs, spatial-based GNNs, and essential fundamentals. The proposed framework is summarized in Section 3, which emphasizes the relevance of hierarchy. From Section 4 through Section 5, we explore exemplary GNN models in each domain using our proposed taxonomy structure. In Section 6, we go over the advantages and disadvantages of each domain in detail, as well as practical guidance on GNN model selection. Section 7 also includes a case study of our techniques, demonstrating our proposed framework with current and relevant GNNs concerns.

1.2 Related Surveys and Our Contributions

Existing works on GNNs can be categorized into three groups:

- **Group 1 (Comprehensive Surveys):** Most surveys catalog new GNN research into distinct classes, but they generally lack a unified framework [15–17, 66, 67].
- **Group 2 (Theoretical Perspectives):** The surveys undertaken in this context investigate GNNs from certain perspectives, however, their reach is frequently constrained [60, 68–71].
- **Group 3 (Post Hoc Explanation):** This methodology comprehends GNNs through the utilization of post hoc analysis, which provides valuable insights but poses challenges in terms of validation [72–75].

Our research aims to provide a more interpretable and theoretically grounded understanding of GNNs. Previous surveys either categorize several disparate groups or only address a few GNNs using a certain methodology. Following the overall goals of our framework, we want to create one framework that unifies GNNs across the spatial and spectral domains as well as within each domain via theoretical support. It should be noted that the majority of the work presented is related

to Graph Convolution Networks (GCN) [8], which is the most common type of GNNs, and that many other varieties of GNNs are still based on GCN. As a result, we do not differentiate between GNNs and GCN in this context and refer to GNNs in the following sections.

2 PROBLEM SETUP AND PRELIMINARY

We enumerate all the notations utilized in this survey within Table 1. Furthermore, we will discuss the fundamental concepts, necessary preliminaries, and the problem setup for learning node-level representation, which is the major task in the GNN literature. A simple graph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of n nodes and $\mathcal{E}$ represents edges. An entry $v_i \in \mathcal{V}$ denotes a node, and $e_{i,j} = \{v_i, v_j\} \in \mathcal{E}$ indicates an edge between nodes i and j. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ is defined by if there is a link between node i and j, $\mathbf{A}_{i,j} = 1$, and else 0. Node features $\mathbf{X} \in \mathbb{R}^{N \times F}$ is a matrix with each entry $x_i \in \mathbf{X}$ representing the feature vector on node i. Another popular graph matrix is the graph Laplacian, which is defined as $\mathbf{L} = \mathbf{D} - \mathbf{A} \in \mathbb{R}^{N \times N}$, where $\mathbf{D}$ is the degree matrix. Due to its generalization ability [76], the symmetric normalized Laplacian is often used, which is defined as $\tilde{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}}$. Another option is random walk normalization: $\tilde{\mathbf{L}} = \mathbf{D}^{-1} \mathbf{L}$. Note that normalization could also be applied to the adjacency matrix, and their relationship is $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$. Most GNNs focus on node-level embeddings, learning how a graph signal is modified by a graph topology, and outputting a filtered feature as:

Table 1. Commonly Used Notations

Notations	Descriptions
$\mathcal{G}$	A graph.
$\mathbf{V}$	The set of nodes in a graph.
$\mathbf{E}$	The set of edges in a graph.
$\mathbf{A}, \tilde{\mathbf{A}}$	The adjacency matrix and its normalization.
$\mathbf{L}, \tilde{\mathbf{L}}$	The graph Laplacian matrix and its normalization.
v	A node $v \in \mathbf{V}$.
e_{ij}	An edge $e_{ij} \in \mathbf{E}$.
$\lambda_i \in \mathbf{\Lambda}$	Eigenvalue(s).
$\mathbf{U}, \mathbf{U}^\intercal$	Eigenvector matrix and its transpose.
$\mathbf{U}_i \in \mathbf{U}, \mathbf{u}^\intercal{}_i \in \mathbf{U}^\intercal$	Single eigenvector and its transpose.
$\mathbf{D}$	The degree matrix of $\mathbf{A}$ and $\mathbf{D}_{ii} = \sum_{j=1}^{n} \mathbf{A}_{ij}$.
$\mathbf{X} \in \mathbf{R}^{N \times d}$	The feature matrix of a graph.
$\mathbf{Z} \in \mathbf{R}^{N \times b}$	New node feature matrix.
$\mathbf{H} \in \mathbf{R}^{N \times b}$	The node hidden feature matrix.
$\mathbf{h}_v \in \mathbf{R}^b$	The hidden feature vector of node v.
N	node number
b	dimension size of hidden feature
$\odot$	Element-wise product.
$\mathbf{\Theta}, \theta$	Learnable model parameters.
$\mathbf{P}(\cdot), \mathbf{Q}(\cdot)$	Polynomial function.
$\mathcal{N}(v)$	Directed neighbors of node v

$$f : G, \mathbf{X} \rightarrow \mathbf{Z}, \tag{1}$$

where we aim to find a mapping that can integrate graph structure and original node features, generating a update node representation $\mathbf{Z}$. G represents the graph connectivity, and many options are available as listed in Table 2, and most popular are symmetric normalized graph matrices. In this survey, we use the graph Laplacian, adjacency matrix and their modifications described in Table 2 to represent a graph. So, yet, no experimental or theoretical evidence has been shown that any filter has a consistent advantage [77]. This survey is investigating two specific groups of GNNs, namely, spectral- and spatial-based GNNs, which are defined as below:

Definition 2.1 (Spatial Method). By integrating graph connectivity G and node features $\mathbf{X}$, the updated node representations ($\mathbf{Z}$) are defined as:

$$\mathbf{Z} = f(G) \cdot \mathbf{X}, \tag{2}$$

where G is often implemented with $\mathbf{A}$ or $\mathbf{L}$ in existing works. Therefore, spatial methods focus on finding a **node aggregation function** $f(\cdot)$ that learns a proper aggregation with node features $\mathbf{X}$ to obtain a updated node embedding $\mathbf{Z}$.

Before introducing another definition, the necessary preliminary background is offered: **(1) graph Fourier transform**: The graph Laplacian $\mathbf{L}$ can be diagonalized [78, 79] as $\tilde{\mathbf{L}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal$, where $\mathbf{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues (i.e., $\mathbf{\Lambda}_{ii} = \lambda_i$), and $\mathbf{U}$ represents eigenvectors. Further, the graph Fourier transform of a signal $\mathbf{X}$ is defined as $\hat{\mathbf{X}} = \mathbf{U}^\intercal\mathbf{X} \in \mathbb{R}^{N\times N}$ and its inverse as $\mathbf{X} = \mathbf{U}\hat{\mathbf{X}}$. **(2) spectral convolution**: According to the Convolution Theorem (i.e., Fourier transform of a convolution of two signals is the element-wise product of their Fourier transforms) [80], the convolution is defined in the Fourier domain as:

Table 2. Representations for Graph Topology

Notations	Descriptions
$\mathbf{A}$	Adjacency matrix
$\mathbf{L}$	Graph Laplacian
$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$	Adjacency with self loop
$\mathbf{D}^{-1}\mathbf{A}$	Random walk row normalized adjacency
$\mathbf{A}\mathbf{D}^{-1}$	Random walk column normalized adjacency
$\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$	Symmetric normalized adjacency
$\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}}$	Left renormalized adjacency, $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$
$\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1}$	Right renormalized
$\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$	Symmetric renormalized
$(\tilde{\mathbf{D}}^{-1}\tilde{\mathbf{A}})^k$	Powers of left renormalized adjacency
$(\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1})^k$	Powers of right renormalized adjacency

$$f_1 * f_2 = \mathbf{U}\left[(\mathbf{U}^\intercal f_1) \odot (\mathbf{U}^\intercal f_2)\right],$$

where $\odot$ is the element-wise product, and f_1/f_2 are two signals defined on the spatial domain.

Definition 2.2 (Spectral Method). A node signal $f_2 = \mathbf{X}$ is filtered by spectral function $\mathbf{g} = \mathbf{U}^\intercal f_1$ as:

$$\mathbf{g} * \mathbf{X} = \mathbf{U}\left[\mathbf{g}(\mathbf{\Lambda}) \odot (\mathbf{U}^\intercal\mathbf{X})\right] = \mathbf{U}\,\mathrm{diag}(\mathbf{g}(\mathbf{\Lambda}))\,\mathbf{U}^\intercal\mathbf{X}, \tag{3}$$

where $\mathbf{g}$ is known as **frequency response function**. If $\mathbf{g}$ is polynomial, rational, or exponential function, then we can reduce the equation above to:

$$\mathbf{g} * \mathbf{X} = \mathbf{g}(\tilde{\mathbf{L}})\,\mathbf{X}. \tag{4}$$

In short, the objective of spectral methods is to learn a frequency response function $\mathbf{g}(\cdot)$.

The goal of both methods is to learn how to approximate node aggregation or a frequency response function using the data. A great deal of approximation techniques are utilized, and thus f or g can be efficiently estimated. Approximation theory is a branch of mathematics dedicated to discovering and quantifying the errors caused when functions are approximated using smaller functions. Despite the fact that polynomials have a more convenient form than rational functions and are more popular due to its efficiency, rational functions are better at approximating functions at singularities and on unbounded domains. The basic characteristics of rational functions are outlined in complex analytic literature [81–91]. As an important polynomial approximation, Chebyshev approximation is first introduced as spectral filtering for graph convolution: A real symmetric graph Laplacian $\mathbf{L}$ can be decomposed as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal$. Chebyshev approximation on spectral filter $\mathbf{g}$ is applied [9, 64] so it can be approximated with a polynomials with order k:

$$\begin{aligned}
\mathbf{g} * \mathbf{X} &= \mathbf{U}\,\mathbf{g}(\mathbf{\Lambda})\,\mathbf{U}^\intercal\mathbf{X} \\
&\approx \mathbf{U}\sum_k \theta_k T_k(\tilde{\mathbf{\Lambda}})\,\mathbf{U}^\intercal\mathbf{X} && \left(\tilde{\mathbf{\Lambda}} = \frac{2}{\lambda_{max}}\mathbf{\Lambda} - \mathbf{I}_N\right) \\
&= \sum_k \theta_k T_k(\tilde{\mathbf{L}})x && (\mathbf{U}\mathbf{\Lambda}^k\mathbf{U}^\intercal = (\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\intercal)^k)
\end{aligned}$$

Another graph convolutional network [8] further simplifies it by reducing the order to 1:

Table 3. Chronological List of Notable GNNs in Spatial and Spectral Domains

	Spatial	Spectral
Before 2015	ParWalk [92], DeepWalk [48], LINE [93]	Spectral GNN [7], ISGNN [94], Neural graph fingerprints [95]
2016	DCNN [11], Molecular Graph Convolutions [96], PATCHY-SAN [97]	GCN [8], ChebNet [9]
2017	MPNN [59], PGCN [98], GraphSAGE [10]	MoNet [13]
2018	GIN [61], Adapative GCN [99], Fast GCN [100] JKNet [101], Large Scale GCN [102]	RationalNet[103], AR [104], CayleyNet [105]
2019	SGCN [106], DeepGCN [107], MixHop [108], PPAP [53]	ARMA [109], GDC [110], EigenPool [111], GWNN [112], Stable GCNN[113]
2020	SIGN [114], Spline GNN [115], UaGGP [116], GraLSP [117], GraphSAINT [118], DropEdge [119], BGNN[120], ALaGNN[121] Continuous GNN [122], GCNII [123], PPRGo [124], DAGNN [125], H2GCN [126]	GraphZoom [127]
2021	ADC [128], UGCN [129], DGC [130], E(n)GNN [131], GRAND [132], C&S [133], LGNN [134]	Interpretable Spectral Filter [135], Expressive Spectral Perspective [136], S2GC [137], BernNet [138]
2022	GINR[139], Adaptive SGC [140], PGGNN [141], DIMP [142]	AGWN [143], ChebNetII [144], JacobiConv [145], SpecGNN [146], G^2CN [147], PGNN [148], ChebGibbsNet [149], SpecFormer[150], SIGN [151], Spectral Density [152]
2023	[153], [154], Low Rank GNN [155], Auto-HeG [156], [157]	DSF [158], F-SEGA [159], [160], [161]

$$
\begin{aligned}
\mathbf{g} * \mathbf{X} &\approx \theta_0 \mathbf{I}_{\mathcal{N}} x + \theta_1 \tilde{\mathbf{L}} \mathbf{X} && \text{(expand to 1st order)} \\
&= \theta_0 \mathbf{I}_{\mathcal{N}} x + \theta_1 \left(\frac{2}{\lambda_{max}} \mathbf{L} - \mathbf{I}_{\mathcal{N}} \right) \mathbf{X} && \left(\tilde{\mathbf{L}} = \frac{2}{\lambda_{max}} \mathbf{L} - \mathbf{I}_{\mathcal{N}}\right) \\
&= \theta_0 \mathbf{I}_{\mathcal{N}} x + \theta_1 (\mathbf{L} - \mathbf{I}_{\mathcal{N}}) \mathbf{X} && (\lambda_{max} = 2) \\
&= \theta_0 \mathbf{I}_{\mathcal{N}} x - \theta_1 \tilde{\mathbf{D}} \mathbf{A} \tilde{\mathbf{D}} \mathbf{X} && (\mathbf{L} = \mathbf{I}_{\mathcal{N}} - \tilde{\mathbf{D}} \mathbf{A} \tilde{\mathbf{D}}) \\
&= \theta_0 (\mathbf{I}_{\mathcal{N}} + \tilde{\mathbf{D}} \mathbf{A} \tilde{\mathbf{D}}) \mathbf{X} && (\theta_0 = -\theta_1) \\
&= \theta_0 (\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}) x && (\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N, \tilde{\mathbf{D}}_{ii} = \textstyle\sum_j \mathbf{A}_{ij}).
\end{aligned}
$$

As a result, θ_0 is the only parameter to learn. The learnable parameters in many different GNNs can vary based on the model design.

3 FRAMEWORK OVERVIEW

The development of GNNs is briefly discussed below with representative studies before we introduce our proposed theoretical framework. Table 3 depicts many sample models that focus on node-level graph convolution. The spectral perspective was previously explored (SGWT [64]),

and it serves as the technical foundation for all subsequent spectral methods, including spectral convolution and approximation. Researchers continue to add to this thread, demonstrating that spectral methods have the ability to handle graphs (SGNN [7], ISGNN [94], ChebNet [9]). Furthermore, GCN [8] and GraphSAGE [10] create effective training methodologies, gaining considerable attention from various communities. After that, spectral techniques development stagnated, with the exception of a few publications on rational filtering (RationalNet [103], AR [104], ARMA [109]). Meanwhile, focus shifts to the spatial domain, which has dominated GNNs to this point. Random walks (ParWalk [92], DeepWalk [48], LINE [93]) and CNN (DCNN [11]) were used in early spatial approaches. Following that, MPNN [59] solidified the message-passing mechanism in spatial techniques. High-order polynomial approximation has been studied [61, 106, 110, 114], but only within the context of ChebNet or DCNN. It is worth noting that while numerous publications described their suggested methods from both spatial and spectral perspectives, only a few GNNs are covered [13, 110]. Until recently, spectral research has demonstrated a resurgence.

In this survey, we provide a framework to fully comprehend spectral methods from a spatial perspective and vice versa. A cross-domain perspective is used to integrate spatial and spectral techniques into a coherent framework. As shown in Figure 1, the proposed framework divides GNNs into two domains: spatial (A-0) and spectral (B-0), each of which is further separated into three subcategories (A-1/A-2/A-3 and B-1/B-2/B-3). A-0 is separated into linear (A-1), polynomial (A-2), and rational (A-3) aggregations based on the types of neighbor aggregation (i.e., f in Definition 2.1). Operations on first-order neighbors only are considered in linear aggregation (A-1), whereas high-order neighbors are incorporated in polynomial aggregation (A-2). In addition, rational aggregation (A-3) includes self-aggregation. According to the types of approximation techniques, the spectral methods are divided into linear (B-1), polynomial (B-2), and rational (B-3) approximation, depending on the types of frequency filtering (i.e., $\mathbf{g}$ in Definition 2.2). In Sections 4 and 5, each category and subcategory will be explained in detail with examples.

3.1 Inside the Spatial and Spectral Domain

The hierarchical link between the spatial and spectral domains is depicted in this subsection. Spatial-based techniques can be divided into three types, with specialization and generalization relationships existing:

$$\text{(A-1) Linear Aggregation} \rightleftarrows \text{(A-2) Polynomial Aggregation} \rightleftarrows \text{(A-3) Rational Aggregation},$$

where it is a generalization from left to right, and specialization from right to left. Specifically, Linear Aggregation (A-1) comprises all algorithms for learning a linear function among neighbors in the first-order. Higher-order neighbors are included in Polynomial Aggregation (A-2), and the order number is defined by the polynomials. Additionally, Rational Aggregation (A-3) utilizes self-aggregation, since the inclusion of higher-order neighbors causes linear aggregation (A-1) to transform into polynomial aggregation (A-2), and polynomial aggregation (A-2) to transform into rational aggregation (A-3) if self-aggregation is added. The approaches falling under the general category of spectral analysis can be grouped into three distinct groups:

$$\text{(B-1) Linear Approximation} \rightleftarrows \text{(B-2) Polynomial Approximation} \rightleftarrows \text{(B-3) Rational Approximation},$$

which includes left-to-right generalization and right-to-left specialization. Concretely, (B-1) outlines all models that aggregate frequency components using a linear function, while (B-2) uses polynomial approximation, and (B-3) applies rational approximation. Therefore, (B-1) can be generalized as (B-2) if replacing linear approximation with polynomial approximation, and (B-2) is generalized as (B-3) if replacing polynomial approximation with rational approximation.

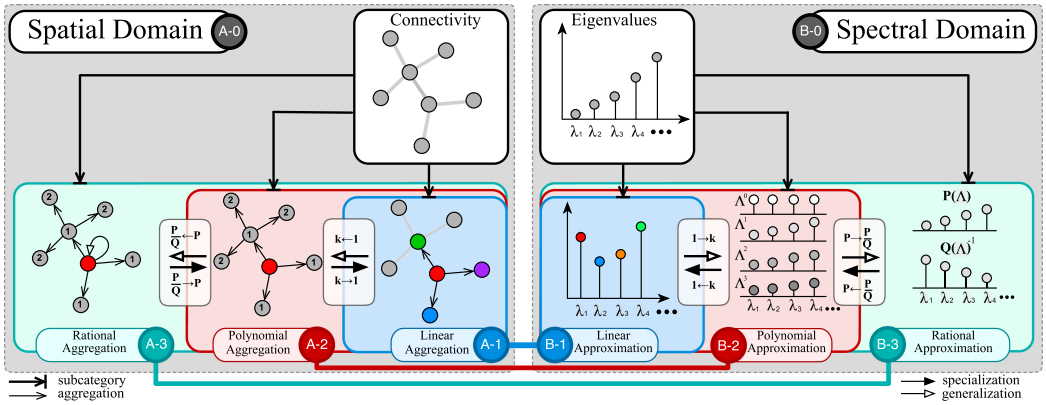

Fig. 1. Illustration of major graph neural operations and their relationship. Spatial and spectral methods are divided into three groups, respectively. Groups A-1, A-2, and A-3 are strongly correlated by generalization and specialization, so are groups B-1, B-3, and B-3. The equivalence among them is marked in the same color.

3.2 Between the Spatial and Spectral Domain

In this section, we go through the link between the spatial and spectral domains over the border. There is also equivalency by modifying the analytical form of these subcategories, as shown below. Linear Aggregation (A-1) and Linear Approximation (B-1) are the initial equivalences:

$$\text{(A-1) Linear Aggregation} \Leftrightarrow \text{(B-1) Linear Approximation},$$

which means that adjusting weights on neighbors in linear aggregation equates to adjusting weights on frequency components in linear approximation using a linear function. Linear Aggregation (A-1) and Linear Approximation (B-1) have the same linear function and can seamlessly convert to each other in closed form. The main difference between them is that Linear Aggregation (A-1) recovers the signal as a linear function of the frequency component, whereas Linear Approximation (B-1) models the target signal as a linear function of neighbor nodes. Both Linear Aggregation (A-1) and Linear Approximation (B-1) aggregate the representations of neighbors by tweaking the weights of each neighbor or use a linear filter on eigenvalues with a negative slope, i.e., $\mathbf{g}(\mathbf{\Lambda}) = -\mathbf{\Lambda} + a$. Because the low-frequency components are given a higher weight by g than their original values, this is referred to as low-pass filtering (i.e., eigenvalues). This group's main advantages are (1) its low computational cost and (2) the large number of real-world scenarios that are subject to the homophily assumption (i.e., neighbors are similar). The fundamental disadvantage is that not every network guarantees homophily. Polynomial Aggregation (A-2) and Polynomial Approximation (B-2) are identical in terms of actual operation, i.e.,

$$\text{(A-2) Polynomial Aggregation} \Leftrightarrow \text{(B-2) Polynomial Approximation}.$$

This means that in polynomial aggregation, aggregating higher orders of neighbors can be expressed as the sum of different orders of frequency components in polynomial approximation. Both use higher-order neighbors in addition to first-order neighbors, increasing the capacity to simulate a more complex relationship among the neighbors. It is theoretically more powerful than (A-1)/(B-1) from a spectral standpoint, because (A-2)/(B-2) is a polynomial approximation as a spectral filter, whereas (A-1)/(B-1) is linear regression. As a result, one flaw is the cost of border neighborhood, which leads to higher computational complexity than (A-1)/(B-1). Another flaw of

Table 4. Structure of Sections 4 and 5

	Section 4 (A-0) **Spatial:** function of adjacency matrix	**Section 5 (B-0)** **Spectral:** function of eigenvalues
Linear $l(\cdot)$	**Section 4.1 (A-1)** $l(A) = a_1 A + a_0 A^0 = a_1 A + a_0 I$	**Section 5.1 (B-1)** $l(\Lambda) = a_1 \Lambda + a_0 \Lambda^0 = a_1 \Lambda + a_0$
Polynomial $\mathbf{P}(\cdot)$	**Section 4.2 (A-2)** $\mathbf{P}(A) = a_m A^m + \cdots + a_k A^k + \cdots + a_0 A^0$	**Section 5.2 (B-2)** $\mathbf{P}(\Lambda) = a_m \Lambda^m + \cdots + a_k \Lambda^k + \cdots + a_0 \Lambda^0$
Rational $\frac{\mathbf{P}(\cdot)}{\mathbf{Q}(\cdot)}$	**Section 4.3 (A-3)** $\frac{\mathbf{P}(A)}{\mathbf{Q}(A)} = \frac{a_m A^m + \cdots + a_k A^k + \cdots + a_0 A^0}{a_m A^m + \cdots + a_k A^k + \cdots + a_0 A^0}$	**Section 5.3 (B-3)** $\frac{\mathbf{P}(\Lambda)}{\mathbf{Q}(\Lambda)} = \frac{a_m \Lambda^m + \cdots + a_k \Lambda^k + \cdots + a_0 \Lambda^0}{a_m \Lambda^m + \cdots + a_k \Lambda^k + \cdots + a_0 \Lambda^0}$

them is that if the order is set too large, then it will over-smooth (i.e., all nodes will look the same), and there is no golden rule for order size because it is based on data. Note that K-layer (A-1) or (B-1) is equivalent to K-order of (A-2)/(B-2), hence stacking K-layer (A-1) or (B-1) causes over-smoothing (B-1). Similarly, the last equivalence relationship is

$$\text{(A-3) Rational Aggregation} \Leftrightarrow \text{(B-3) Rational Approximation},$$

in which rational aggregation defines a label aggregation with self-aggregation, while rational approximation adjusts filter function with rational approximation. Both alleviate the over-smoothing issue by introducing self-aggregation, which limits the intensity of uni-directional aggregation in the spatial domain. From a spectral perspective, this advantage can be interpreted as the superiority of rational approximation (A-3/B-3) over polynomial approximation (A-2/B-2). In particular, the rational approximation is more powerful and precise, particularly when estimating some abrupt signals like discontinuity [82, 84–87, 89, 90].

As a result, we may summarize the advantages and disadvantages of each combination as illustrated in Figure 2. The category selection is based on the data complexity and the efficiency required, as there is a tradeoff between computational efficiency and generalization capability. Table 4 shows the structure of Sections 4 and 5, where we will discuss details of these two threads, respectively, and exemplify using several representative graph neural networks. The second column denotes spatial perspective (Section 4, A-0), which can treat popular GNNs as learning a function of adjacency matrix, or *node aggregation function*. Similarly, the third column means spectral view (Section 5, B-0), which sees GNN as learning a function of eigenvalues or frequency response functions. The cell at the intersection of the second row and second column means that this category of GNNs can be treated as a linear function of adjacency matrix, or, say, its node aggregation function is linear. The other intersection cells follow a similar schema. Note that categories within the same row have the same format and function. For instance, in the second row, A-1 and B-1 share the format of a linear function, but their parameters need not be identical.

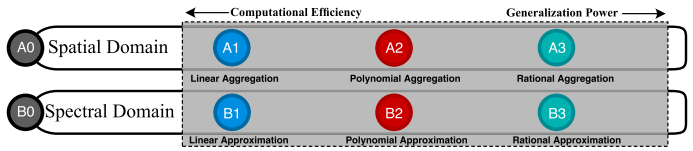

Fig. 2. Category and subcategory comparison.

4 SPATIAL-BASED GNNS (A-0)

In the current literature, spatial approaches such as self-loop, normalization, high-order neighbors, aggregation, and node combination are often explored. We established a new taxonomy for spatial-based GNNs based on these operations, dividing them into three separate groupings as below.

4.1 Linear Aggregation (A-1)

A lot of research has gone into understanding the aggregation among first-order neighbors (i.e., direct neighbors) [10, 12, 48, 59, 61, 162]. The supervisory signal patterns are revealed by altering the weights for the node and its first-order neighbors. The revised node embeddings, $\mathbf{Z}(v_i)$, can be represented in the following way:

$$\mathbf{Z}(v_i) = \overbrace{\Phi(v_i)\,\mathbf{h}(v_i)}^{\text{self node}} + \overbrace{\sum_{u_j \in \mathcal{N}(v_i)} \Psi(u_j)\,\mathbf{h}(u_j)}^{\text{neighbors' aggregation}}, \tag{5}$$

where u_j denotes a neighbor of node v_i, $\mathbf{h}(\cdot)$ is their representations, and Φ/Ψ indicate the weight functions. The first item on the right-hand side denotes the weighted representation of node v_i, while the second represents the update from its neighbors. By applying random walk normalization (i.e., dividing neighbors by degree of the current node), Equation (5) and its symmetric normalization can be written as:

$$\mathbf{Z}(v_i) = \Phi(v_i)\,\mathbf{h}(v_i) + \sum_{u_j \in \mathcal{N}(v_i)} \Psi(u_j)\frac{\mathbf{h}(u_j)}{d_i}, \qquad \tilde{\mathbf{Z}}(v_i) = \Phi(v_i)\,\mathbf{h}(v_i) + \sum_{u_j \in \mathcal{N}(v_i)} \Psi(u_j)\frac{\mathbf{h}(u_j)}{\sqrt{d_i d_j}}, \tag{6}$$

where d_i represents the degree of node v_i. Normalization has better generalization capacity, which is not only due to some implicit evidence but also because of a theoretical proof on performance improvement [163]. In a simplified configuration, weights for all the neighbors are the same and is a scalar value ψ, while the weight for self node ϕ is another scalar value. Therefore, they can be rewritten in matrix form as:

$$\mathbf{Z} = \phi\,\mathbf{X} + \psi\,\mathbf{D}^{-1}\,\mathbf{A}\,\mathbf{X} = (\phi\,\mathbf{I} + \psi\,\mathbf{D}^{-1}\,\mathbf{A})\,\mathbf{X}, \qquad \tilde{\mathbf{Z}} = \phi\,\mathbf{X} + \psi\,\mathbf{D}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{D}^{-\frac{1}{2}}\,\mathbf{X} = (\phi\,\mathbf{I} + \psi\,\mathbf{D}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{D}^{-\frac{1}{2}})\,\mathbf{X}. \tag{7}$$

Equation (7) can be generalized as an identical form:

$$\mathbf{Z} = (\phi\,\mathbf{I} + \psi\tilde{\mathbf{A}})\,\mathbf{X}, \tag{8}$$

where $\tilde{\mathbf{A}}$ denotes normalized $\mathbf{A}$, which is implemented by random walk or symmetric normalization. As shown in Figure 3, the new representation of the current node (in red) is updated as the sum of the previous representations of itself and its neighbors. (A-1) may adjust the weights of the neighbors. The following are a few state-of-the-art approaches that were chosen to demonstrate this schema:

4.1.1 Graph Convolutional Network (GCN). As the one state-of-the-art, GCN [8] adds a self-loop to nodes, and applies a *renormalization* trick that changes degree matrix from $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ to $\hat{\mathbf{D}}_{ii} = \sum_j (\mathbf{A} + \mathbf{I})_{ij}$. Specifically, GCN can be written as:

$$\mathbf{Z} = \hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\,\mathbf{X} = \hat{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{I} + \mathbf{A})\hat{\mathbf{D}}^{-\frac{1}{2}}\,\mathbf{X} = (\mathbf{I} + \tilde{\mathbf{A}})\,\mathbf{X}, \tag{9}$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$, and $\tilde{\mathbf{A}}$ is normalized adjacency matrix with self-loop. Therefore, Equation (9) is equivalent to Equation (8) when setting $\phi = 0$ and $\psi = 1$ with the *renormalization* trick. Besides, GCN takes the sum of each node and average of its neighbors as new node embeddings. Note that the normalization of GCN is different from the others, but the physical meaning is the same.

4.1.2 GraphSAGE. Computing intermediate representations of each node and its neighbors, GraphSAGE [10] applies an aggregation among its neighbors. Take mean aggregator as example, it averages a node with its neighbors, i.e.,

$$\mathbf{Z}(v_i) = \text{MEAN}\left(\{\mathbf{h}(v_i)\} \cup \left\{\mathbf{h}(u_j), \forall u_j \in \mathcal{N}(v_i)\right\}\right), \tag{10}$$

where $\mathbf{h}$ indicates the intermediate representation, and $\mathcal{N}$ denotes the neighbor nodes. Equation (10) can be written in matrix form after implementing MEAN using symmetric normalization:

$$\mathbf{Z} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{I} + \mathbf{A})\,\mathbf{D}^{-\frac{1}{2}}\,\mathbf{X} = (\mathbf{I} + \tilde{\mathbf{A}})\,\mathbf{X}, \tag{11}$$

which is equivalent to Equation (8) with $\phi = 1$ and $\psi = 1$. Note that the key difference between GCN and GraphSAGE is the normalization strategy: The former is symmetric normalization with renormalization trick, and the latter is random walk normalization.

4.1.3 Graph Isomorphism Network (GIN). Inspired by the **Weisfeiler-Lehman (WL)** test, GIN [61] develops conditions to maximize the power of GNNs, proposing a simple architecture, Graph Isomorphism Network. With strong theoretical support, GIN generalizes the WL test and updates node representations as:

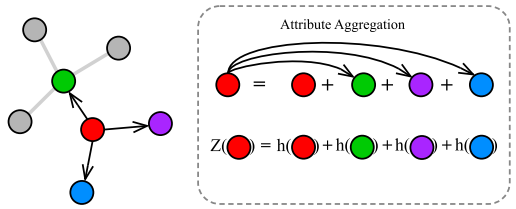

Fig. 3. A-1: the red node updates itself with neighbors.

$$\mathbf{Z} = (1+\epsilon)\cdot\mathbf{h}(v) + \sum_{u_j} \in \mathcal{N}(v_i)\,\mathbf{h}(u_j) = [(1+\epsilon)\,\mathbf{I} + \mathbf{A}]\,\mathbf{X}, \tag{12}$$

which is equivalent to Equation (8) with $\phi = 1 + \epsilon$ and $\psi = 1$. Note that GIN does not perform any normalization.

4.1.4 Graph Attention Model (GAT). GAT [12] applies attention mechanism by adjusting neighbors' weights, instead of using uniform weights in many related works:

$$\mathbf{Z} = (W_{att} \otimes \mathbf{A})\,\mathbf{X}, \tag{13}$$

where $W_{att} \in \mathbb{R}^{N\times N}$ is a matrix, $\otimes$ denotes element-wise multiplication, and calculated by a forward neural network $W_{att}(i,j) = f(\mathbf{h}_i, \mathbf{h}_j)$ with a pair of node representations as input. GAT can be treated as learning dynamic weight on the neighbors, since their weights are not uniform. MoNet [13] is similar to GAT, since its update follows:

$$\mathbf{Z}(v) = \sum_{u\in\mathcal{N}(v)} w_j(\mathbf{u}(\mathbf{h}_i, \mathbf{h}_j))\,\mathbf{h}_j, \tag{14}$$

where $\mathbf{u}$ is a d-dimensional vector of pseudo-coordinates $\mathbf{u}(x, y)$, and

$$w_j(\mathbf{u}) = \exp\left(-\frac{1}{2}(\mathbf{u} - \boldsymbol{\mu}_j)^\top \Sigma_j^{-1}(\mathbf{u} - \boldsymbol{\mu}_j)\right), \tag{15}$$

where μ_j are learnable d × d and d × 1 covariance matrix and mean vector of a Gaussian kernel, respectively. Let $W_{Monet} = w(u(\cdot))$ as a weight function of a pair of node representations representation, then it is also an attention model:

$$\mathbf{Z} = (W_{MoNet} \otimes \mathbf{A})\,\mathbf{X}\,. \tag{16}$$

These works do not consider updating nodes with their original representations, i.e., $\phi = 0$ and ψ is replaced with matrix parameter W in Equation (8). However, it is easy to extend them with self node.

4.2 Polynomial Aggregation (A-2)

Several researches use higher orders of neighbors to integrate deeper structural information [9, 11, 49, 93, 106], because direct neighbors are not always enough to describe a node's surroundings. Excessive order, however, generally averages all node representations, resulting in over-smoothing and a loss of emphasis on the immediate neighborhood [104]. Many models are motivated by this to fine-tune the aggregation strategy based on different orders of neighbors. As a result, adequate constraint and order flexibility are essential for node representation. Challenging signals, such as Gabor-like filters, have been shown to have a high order of neighbors [108].

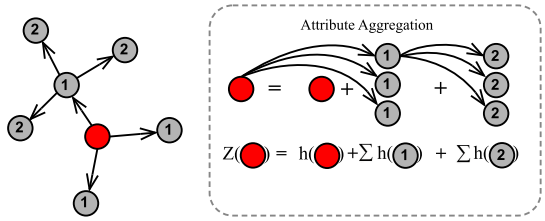

Fig. 4. A-2: This red node updates its representation with its first- and second-order neighbors to update itself.

Define the shortest distance between node i and j as $d_G(i,j)$, and $\partial\mathcal{N}(i,\tau)$ to be the set of nodes j that satisfies $d_G(i,j)=\tau$, i.e., τ-order neighbors. Formally, this type of work can be written as:

$$\mathbf{Z}(v_i) = \Phi(v_i)\,\mathbf{h}(v_i) + \overbrace{\sum_{u_j\in\mathcal{N}(v_i,\tau=1)} \Psi^{(\tau=1)}\,\mathbf{h}(u_j)}^{\text{1st-order neighbor}} + \cdots + \overbrace{\sum_{u_j\in\mathcal{N}(v_i,\tau=k)} \Psi^{(\tau=k)}\,\mathbf{h}(u_j)}^{k\text{th order neighbor}} + \cdots, \tag{17}$$

where $\Psi^{(\tau)}$ indicates a scalar parameter for all τ-order neighbors. Setting the same order neighbors to share the same weights, Equation (17) can be rewritten in matrix form:

$$\mathbf{Z} = \left(\phi\,\mathbf{I} + \sum_{j=1}^{k} \psi_j\,\mathbf{A}^j\right)\mathbf{X} = \mathbf{P}_k(\mathbf{A})\,\mathbf{X}, \tag{18}$$

where $\mathbf{P}_k(\cdot)$ is a polynomial function with order number k. Applying symmetric normalization, Equation (18) can be rewritten in matrix form as:

$$\mathbf{Z} = \left(\phi\,\mathbf{I} + \sum_{j=1}^{k} \psi_i(\mathbf{D}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{D}^{-\frac{1}{2}})^j\right)\mathbf{X} = \left(\phi\,\mathbf{I} + \sum_{i=1}^{k} \psi_i\,\tilde{\mathbf{A}}^i\right)\mathbf{X} = \left(\sum_{i=0} \psi_i\,\tilde{\mathbf{A}}^i\right)\mathbf{X} = \mathbf{P}_k(\tilde{\mathbf{A}})\,\mathbf{X}, \tag{19}$$

where $\phi=\psi_0$ due to the fact that $\mathbf{I}=\tilde{\mathbf{A}}^0$, and $\mathbf{A}$ could also be normalized by random walk normalization: $\tilde{\mathbf{A}}=\mathbf{D}^{-1}\,\mathbf{A}$. As shown in Figure 4, the new representation of the current node (in red) is updated as the sum of the previous representations of itself, its first- and second-order neighbors. Note that the weights among those representations are learnable. Several existing works are analyzed below, showing that they are variants of Equation (18) or (19).

4.2.1 ChebNet. To bridge the gap, spectral convolutional operation is generalized, which requires expensive steps of spectral decomposition and matrix multiplication [94, 164]. Introducing truncated Chebyshev polynomial for estimating wavelet in graph signal processing, ChebNet [9] embeds a novel neural network layer for the convolution operator. Specifically, ChebNet can be written as:

$$\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}})\,\mathbf{X} = (\tilde{\theta}_0\,\mathbf{I} + \tilde{\theta}_1\tilde{\mathbf{L}} + \tilde{\theta}_2\tilde{\mathbf{L}}^2 + \cdots)\,\mathbf{X}, \tag{20}$$

where $T_k(\cdot)$ denotes the Chebyshev polynomial and θ_k is the Chebyshev coefficient. $\tilde{\theta}$ is the coefficient after expansion and reorganization. Since $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$, we have:

$$\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}})\,\mathbf{X} = [\tilde{\theta}_0\,\mathbf{I} + \tilde{\theta}_1(\mathbf{I}-\tilde{\mathbf{A}}) + \tilde{\theta}_2(\mathbf{I}-\tilde{\mathbf{A}})^2 + \cdots]\,\mathbf{X} = \left(\phi\,\mathbf{I} + \sum_{i=1}^{k} \psi_i\,\tilde{\mathbf{A}}^i\right)\mathbf{X} = \mathbf{P}_k(\tilde{\mathbf{A}})\,\mathbf{X}, \tag{21}$$

which is exactly Equation (19). The predefined K is the order number of Chebyshev polynomial, and a larger K means higher approximation accuracy in estimating the function of eigenvalues. Equation (21) shows that K also can be explained as the highest order of the neighbors.

4.2.2 DeepWalk. Applying random walk, DeepWalk [48] first draws a group of random paths from a graph and applies a skip-gram algorithm to extract node features. Assuming the number of random walks is sufficient, the transition probability on a graph can be represented as:

$$\tilde{\mathbf{A}} = \mathbf{D}^{-1}\,\mathbf{A}\,. \tag{22}$$

Let the window size of skip-gram be $2t + 1$ and the current node is the $(t + 1)$-th one along each sampled random walk path, so the farthest neighbor current node can reach is the first one and the last one. A node and its neighbors are likely to appear in the same random walk path, and the neighbors follow the transition probability (Equation (22)) to appear in the same path. Therefore, the updated representation is as follows:

$$\mathbf{Z} = \frac{1}{t+1}(\mathbf{I} + \tilde{\mathbf{A}} + \tilde{\mathbf{A}}^2 + \cdots + \tilde{\mathbf{A}}^t)\,\mathbf{X} = \frac{1}{t+1}\,\mathbf{P}_k(\tilde{\mathbf{A}})\,\mathbf{X}, \tag{23}$$

where $\mathbf{I}$ means that DeepWalk always considers previous representation of the current node as one element, $\tilde{\mathbf{A}}$ represents the direct neighbors' transition probability, and $\tilde{\mathbf{A}}^2$ denotes that of the second-order neighbors. It will have at most t-order neighbors, depending on the predefined length of the random walk (i.e., $2t + 1$).

4.2.3 Diffusion Convolutional Neural Networks (DCNN). DCNN [11] uses a degree-normalized transition matrix (i.e., renormalized adjacency matrix: $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}\,\mathbf{A}$) as graph representation and performs node embedding update as:

$$\mathbf{Z} = W \odot \tilde{\mathbf{A}}^*\,\mathbf{X} = [w_1, w_2, w_3, \ldots, w_k]^\intercal \odot [\tilde{\mathbf{A}}, \tilde{\mathbf{A}}^2, \tilde{\mathbf{A}}^3 \ldots, \tilde{\mathbf{A}}^k]^\intercal\,\mathbf{X}, \tag{24}$$

where $\tilde{\mathbf{A}}^*$ denotes a tensor containing the power series of $\tilde{\mathbf{A}}$, and the $\odot$ operator represents element-wise multiplication. It can be transformed as:

$$\mathbf{Z} = (w_1\,\tilde{\mathbf{A}} + w_2\,\tilde{\mathbf{A}}^2 + w_3\,\tilde{\mathbf{A}}^3 + \cdots w_k\,\tilde{\mathbf{A}}^k)\,\mathbf{X} = \mathbf{P}_k(\tilde{\mathbf{A}})\,\mathbf{X}\,. \tag{25}$$

4.2.4 Scalable Inception Graph Neural Networks (SIGN). By generalizing GCN [8], GAT [12], and SGC [106], SIGN [114] constructs a set of linear diffusion operators along with non-linearity. For node-wise classification tasks, SIGN has the form:

$$\begin{aligned} \mathbf{Z} &= \sigma\left([\mathbf{X}\Theta_0, \mathbf{A}_1\mathbf{X}\Theta_1, \ldots, \mathbf{A}_r\mathbf{X}\Theta_r]\right), \\ \mathbf{Y} &= \xi(\mathbf{Z}\Omega), \end{aligned} \tag{26}$$

where $[\cdot, \cdot, \ldots,]$ is concatenation, and r denotes the power number. SIGN can be rewritten as:

$$\mathbf{Z} = [\mathbf{X}\Theta_0, \mathbf{A}_1\mathbf{X}\Theta_1, \ldots, \mathbf{A}_r\mathbf{X}\Theta_r]\,\Omega = \omega_0\sigma(\mathbf{X}\Theta_0) + \omega_1\sigma(\mathbf{A}_1\mathbf{X}\Theta_1), \ldots, \omega_r\sigma(\mathbf{A}_r\mathbf{X}\Theta_r), \tag{27}$$

where $\sigma(\mathbf{A}_r\mathbf{X}\Theta_r)$ could be treated as refined representation of each order of label aggregation by non-linear function σ and fully connected layer Ω, i.e., $\widehat{\mathbf{A}^r\mathbf{X}}$. Replacing A with normalized adjacency matrix, it can be rewritten as:

$$\hat{\mathbf{Z}} = \omega_0\widehat{\tilde{\mathbf{A}}^0\mathbf{X}} + \omega_1\widehat{\tilde{\mathbf{A}}^1\mathbf{X}}, \ldots, \omega_r\widehat{\tilde{\mathbf{A}}^r\mathbf{X}} = \sum_r \omega_r\widehat{\tilde{\mathbf{A}}^r\mathbf{X}} = \widehat{\mathbf{P}(\tilde{\mathbf{A}})\mathbf{X}}. \tag{28}$$

4.2.5 Graph Diffusion Convolution (GDC). GDC [110] defines a generalized graph diffusion via the diffusion matrix:

$$\mathbf{Z} = \sum_{k=0}^{\infty} \theta_k T^k, \tag{29}$$

where θ_k are the weighting coefficients with $\sum_{k=0}^{\infty}\theta_k = 1, \theta_k \in [0,1]$, T is a generalized undirected transition matrix that includes the random walk transition matrix $T_{rw} = \mathbf{A}\,\mathbf{D}^{-1}$, and the symmetric transition matrix $T_{sym} = \mathbf{D}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{D}^{-\frac{1}{2}}$. In the general case, it can be written as :

$$\mathbf{Z} = \sum_{k=0}^{\infty} \theta_k\,\tilde{\mathbf{A}}^k = \mathbf{P}(\tilde{\mathbf{A}}). \tag{30}$$

4.2.6 Node2Vec. Node2Vec [49] defines a second-order random walk to control the balance between **BFS (breadth-first search)** and **DFS (depth-first search)**. Consider a random walk that traversed an edge between node t and v, denoted as (t, v), and now it resides at node v. Then, the transition probabilities to next stop x from node t is defined as:

$$P(t \to x) = \begin{cases} \frac{1}{p} & \text{if } d(t,x) = 0, \text{ return to the source} \\ 1 & \text{if } d(t,x) = 1, \text{ BSF} \\ \frac{1}{q} & \text{if } d(t,x) = 2, \text{ DFS,} \end{cases} \tag{31}$$

where $d(t,x)$ denotes the shortest path between nodes t and x. $d(t,x) = 0$ indicates a second-order random walk returns to its source node, (i.e., $t \to v \to t$), while $d(t,x) = 1$ means that this walk goes to a BFS node, and $d(t,x) = 2$ to a DFS node. The parameters p and q control the distribution of the three cases. Assuming the random walk is sufficiently sampled, Node2Vec can be rewritten in matrix form:

$$\mathbf{Z} = \left(\frac{1}{p}\cdot\overbrace{\mathbf{I}}^{\text{source}} + \overbrace{\tilde{\mathbf{A}}}^{\text{BFS}} + \frac{1}{q}\overbrace{(\tilde{\mathbf{A}}^2 - \tilde{\mathbf{A}})}^{\text{DFS}}\right)\mathbf{X}, \tag{32}$$

which can be transformed and reorganized as:

$$\mathbf{Z} = \left[\frac{1}{p}\mathbf{I} + \left(1 - \frac{1}{q}\right)\tilde{\mathbf{A}} + \frac{1}{q}\tilde{\mathbf{A}}^2\right]\mathbf{X} = \mathbf{P}_{k=2}(\tilde{\mathbf{A}})\,\mathbf{X}, \tag{33}$$

where transition probabilities $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\,\mathbf{A}$ is random walk normalized adjacency matrix.

4.2.7 LINE [165]/SDNE [166]. These two models consider first-order and second-order neighbors as the constraints for learning node embeddings. First-order: The nodes representation is forced to be similar to its neighbors, which is equivalent to:

$$\mathbf{Z} = \tilde{\mathbf{A}}\,\mathbf{X}\,. \tag{34}$$

Second-order: The pair of nodes is forced to be similar if their neighbors are similar, which is equivalent to make second-order neighbors similar, therefore, we can get the second-order connectivity by taking the power of the original adjacency:

$$\mathbf{Z} = \tilde{\mathbf{A}}^2\,\mathbf{X}\,. \tag{35}$$

Then the final learned node embeddings are formulated as:

$$\mathbf{Z} = \tilde{\mathbf{A}}\,\mathbf{X} + \alpha\,\tilde{\mathbf{A}}^2\,\mathbf{X} = \mathbf{P}_{k=2}(\tilde{\mathbf{A}})\,\mathbf{X}\,. \tag{36}$$

Since LINE uses concatenation between the representations constrained by first and second-order, $\alpha = 1$; for SDNE, α is pre-defined.

4.2.8 Simple Graph Convolution (SGC). To reduce the computational overhead, SGC [106] removes non-linear function between neighboring graph convolution layers and combines graph aggregation in one single layer:

$$\mathbf{Z} = \tilde{\mathbf{A}}^k\,\mathbf{X}, \tag{37}$$

where $\tilde{\mathbf{A}}$ is renormalized adjacency matrix, i.e., $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\,\mathbf{A}\,\tilde{\mathbf{D}}^{-\frac{1}{2}}$, where $\tilde{\mathbf{D}}^{-\frac{1}{2}}$ is degree matrix with self loop. Therefore, it can be easily rewritten as:

$$\mathbf{Z} = (0\cdot\mathbf{I} + 0\cdot\tilde{\mathbf{A}} + 0\cdot\tilde{\mathbf{A}}^2 + \cdots + 1\cdot\tilde{\mathbf{A}}^k)\,\mathbf{X} = \mathbf{P}_k(\tilde{\mathbf{A}})\,\mathbf{X}, \tag{38}$$

which only has the highest-order term.

4.3 Rational Aggregation (A-3)

Most works merely consider label propagation from the node to its neighbors (i.e., gathering information from its neighbors) but ignore self-aggregation. Self-aggregation means that labels or attributes can be propagated back to themselves or restart propagating with a certain probability. This reverse behavior can avoid over-smoothing issue [53]. Note that Polynomial Aggregation (A-2) may manually change the order number to relieve the over-smoothing issue, but Rational Aggregation (A-3) can do so automatically. Theoretically, rational function approximation is more effective than polynomial and has been researched in machine learning problems [86, 167, 168]. Several works use a rational function on the adjacency matrix to perform self-aggregation, either explicitly or implicitly [53, 103, 105, 109, 169–172]. Because generic label propagation is achieved by multiplying the graph Laplacian, self-aggregation may be achieved by multiplying the inverse graph Laplacian as follows:

$$\mathbf{Z} = \mathbf{P}_m(\tilde{\mathbf{A}})\,\mathbf{Q}_n(\tilde{\mathbf{A}})^{-1}\,\mathbf{X} = \frac{\mathbf{P}_m(\tilde{\mathbf{A}})}{\mathbf{Q}_n(\tilde{\mathbf{A}})}\,\mathbf{X}, \tag{39}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are two different polynomial functions, and the bias of $\mathbf{Q}$ is often set to 1 to normalize the coefficients. As shown in Figure 5, the new representations of the current node (in red) are updated as the previous one with probability P, and as that of neighbors with probability (1-P). The difference of A-3 beyond A-2 is that A-3 can avoid over-smoothing issue in an automatic manner [104, 173]. Over-smoothing issue happens when GNNs go deep, which would drive node features to a stationary point and average all the information from raw node representations. Graph convolution can be described as an optimization problem [57, 67, 170, 173], e.g., (1) minimizing the supervised loss and (2) keeping the local neighborhood similar:

$$\mathbf{Z} = \arg\min_{\mathbf{Z}}\{\ \underbrace{\|\mathbf{Z}-Y\|_2^2}_{\text{(1) supervised loss}} + \underbrace{\alpha\,\mathrm{Tr}\left(\mathbf{Z}^\top\,\mathbf{L}\,\mathbf{Z}\right)}_{\text{(2) neighborhood regularization}}\ \}, \tag{40}$$

where α is the controlling weight between the two constraints. The problem has analytical solution:

$$\mathbf{Z} = (\mathbf{I} + \alpha\,\tilde{\mathbf{L}})^{-1}Y = ((1+\alpha)\,\mathbf{I} - \tilde{\mathbf{A}})Y. \tag{41}$$

However, because *alpha* increases with the number of times graph convolution is done, it is prone to over-smoothing. Over-smoothing is addressed in a variety of ways [101, 114, 119, 123, 174], including Rational Aggregation (A-3), which does so by retaining a portion of the original representation no matter how many iterations it does, greatly reducing over-smoothing.

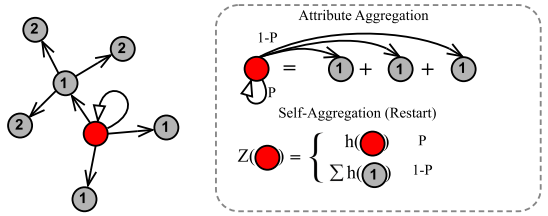

Fig. 5. A-3: The red nodes (red) update its representation with self-loop and neighbors. In A-3, the ratio of the original representation remains stable, whereas A-1 does not control the ratio.

4.3.1 Auto-regressive. **Label propagation (LP)** [175–177] is a widely used methodology for graph-based learning. The objective of LP is two-fold: one is to extract embeddings that match with the node label, the other is to become similar to neighboring vertices. The label can be treated as part of node attributes, so we have:

$$\mathbf{Z} = (\mathbf{I} + \alpha\,\tilde{\mathbf{L}})^{-1}\,\mathbf{X} = \frac{\mathbf{I}}{\mathbf{I} + \alpha(\mathbf{I} - \tilde{\mathbf{A}})}\,\mathbf{X} = \frac{\mathbf{I}}{(1+\alpha)\,\mathbf{I} - \alpha\,\tilde{\mathbf{A}}}\,\mathbf{X}, \tag{42}$$

which is the closed-form solution and also equivalent to the form of Equation (39), i.e., $\mathbf{P} = \mathbf{I}$ and $\mathbf{Q} = (1+\alpha)\,\mathbf{I} - \alpha\,\tilde{\mathbf{A}}$.

4.3.2 Personalized PageRank (PPNP). Obtaining node's representation via teleport (restart), PPNP [51, 53, 178] keeps the original representation (self-aggregation) $\mathbf{X}$ with probability α. Therefore, 1-α is the probability of performing the normal label propagation:

$$\mathbf{Z} = \alpha\left(\mathbf{I} - (1-\alpha)\,\tilde{\mathbf{A}}\right)^{-1}\mathbf{X} = \frac{\alpha}{\mathbf{I} - (1-\alpha)\,\tilde{\mathbf{A}}}\,\mathbf{X}, \tag{43}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\,\mathbf{A}$ is random walk normalized adjacency matrix with self-loop. Equation (43) is with a rational function whose numerator is a constant.

4.3.3 ARMA Filter. ARMA [109] filter approximates any desired filter response function with updates as:

$$\mathbf{Z} = \frac{b}{\mathbf{I} - a\,\tilde{\mathbf{A}}}\,\mathbf{X}\,. \tag{44}$$

Note that ARMA filter is an unnormalized version of PPNP. When a + b = 1, ARMA becomes PPNP.

4.3.4 ParWalks. A partially absorbing random walk is a second-order Markov chain with partial absorption at each state. Reference [92] shows that with proper absorption, the absorption probabilities can well capture the global graph structure. Note that the concept "absorption" in Reference [92] is similar to "teleport" or "restart" in PPNP [53]. ParWalks defines the aggregation as:

$$p_{ij} = \begin{cases} \frac{\alpha_i}{\alpha_i + d_i}, & i = j \\ \\ \frac{w_{ij}}{\alpha_i + d_i}, & i \neq j, \end{cases} \tag{45}$$

where α is defined as a variable to control the level of absorption, w_{ij} and d_i indicate non-negative matrix of pairwise affinities between vertex i and j, and degree of vertex i, respectively.

$$\mathbf{Z} = \frac{\alpha\,\mathbf{I}}{\alpha + \tilde{\mathbf{L}}}\,\mathbf{X} = \frac{\alpha}{\alpha\,\mathbf{I} + \mathbf{I} - \tilde{\mathbf{A}}}\,\mathbf{X}, \tag{46}$$

where α is redefined as a regularizer in the original paper [92]. When $\alpha = 1$, all nodes follow the same absorbing behavior. Otherwise, each node has an independent absorbing policy. Also,

ParWalks model is equivalent to ARMA filter ($a = b = \frac{1}{2}$) when $\alpha = 1$ and with normalized Laplacian:

$$\mathbf{Z} = \frac{\mathbf{I}}{\mathbf{I}+\tilde{\mathbf{L}}}\mathbf{X} = \frac{\mathbf{I}}{\mathbf{I}+(\mathbf{I}-\tilde{\mathbf{A}})}\mathbf{X} = \frac{\frac{1}{2}\mathbf{I}}{\mathbf{I}-\frac{1}{2}\tilde{\mathbf{A}}}\mathbf{X}. \tag{47}$$

The author also discussed the over-smoothing issue: When $\mathbf{\Lambda} = \mathbf{I}$ and as $\alpha \to 0$, a ParWalk would converge to the constant distribution $1/n$, regardless of the starting vertex.

4.3.5 RationalNet. To leverage higher order of neighbors, RationalNet [103] proposes a general rational function with a predefined order number, and it is optimized by Remez algorithm. The analytic form is exactly Equation (39). The major difference beyond PPNP or ARMA filter is that RationalNet generalized them, and the order can be any number.

Remark: The optimization towards rational aggregation (A-3) is exactly the same as the *residual learning* that was first and widely used in image recognition [179]. As shown in the work PPAP [48], the author proposed an iterative algorithm called APPAP, which is

$$\mathbf{Z}^{(k+1)} = (1-\alpha)\tilde{A}\mathbf{Z}^{(k)} + \alpha\mathbf{Z}^{0} = \overbrace{\sum_{i}^{k}\alpha(1-\alpha)^{i}\tilde{A}^{i}\mathbf{Z}^{(0)}}^{F(x)\text{ where }x=\mathbf{Z}^{(0)}} + \overbrace{\alpha\mathbf{Z}^{0}}^{\text{identity }x}, \tag{48}$$

where $\mathbf{Z}^{k}$ means the intermediate representations at kth layer. This format is exactly the same as residual learning, as illustrated in Figure 6: sum of multiple graph convolutions serves as $F(x)$, and each time identify input will be added. The difference is that Equation (48) has normalized the weights ($(1-\alpha)+\alpha = 1$). Because the matrix inversion in Rational Aggregation is computationally costly, iterative methods are commonly used to efficiently determine the matrix inversion [53, 105, 109]. We only demonstrate one iteration of Auto-Regression, PPNP, ARMA, and ParWalks. Multiple iterations will result in a more sophisticated rational function. The main distinction between rational and polynomial aggregation is whether or not the inverse graph Laplacian polynomial exists. In each cycle of rational aggregation, a fixed ratio for the original representation is always reserved, whereas polynomial aggregation does not. However, calculating the inverse of the graph Laplacian is expensive, making iterative fashion a key object in online learning algorithms [180]. Utilizing the concept of conductance, where f represents a heat distribution across the vertices, $\mathbf{L}(f)$ signifies the flux induced by f throughout the graph. Consequently, in accordance with the representer theorem [181, 182], $f(\mathcal{V}_i) = \mathbf{L}^{-1}(\mathbf{L}(f))$ can represent an operation that reconstructs the heat distribution f from this flux. Thus, when $\mathbf{L}$ sends a heat distribution f over each node to flux through each vertex, $\mathbf{L}^{-1}$ sends some of the fluxes over the graph back to the original heat distribution (i.e., keeps part of fluxes itself). Going back to the graph learning application, we first translate our updated "heat distribution" to flux through all of those nodes by calculating $\mathbf{P}(\mathbf{L}(f))$. Mth degree of $\mathbf{P}(\cdot)$ means that each vertex can update Mth neighbors at most. Then, using another updated flux in the reverse direction, $\mathbf{Q}(\mathbf{L}(f))^{-1}$ will adjust or reduce flux within Nth neighbors. Polynomial aggregation with more layers or a higher degree tends to involve more neighbors, increasing capacity. When utilizing too many layers or degrees, over-smoothing is almost always unavoidable (e.g., all nodes are

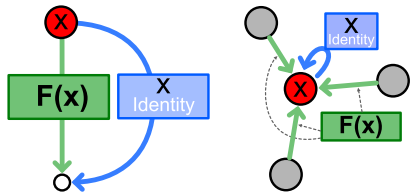

Fig. 6. Left: Residual Learning $x' = F(x) + x$; Right: Rational Aggregation: $x' = F(x) + x$

Table 5. Summary of Representative GNNs

	(**A-1**) linear function of $\mathbf{A}$	(**B-1**) linear function of $\mathbf{\Lambda}$
GCN	$\mathbf{I}+\tilde{\mathbf{A}}$	$2-\mathbf{\Lambda}$
GraphSAGE	$\hat{\mathbf{D}}^{-1}+\tilde{\mathbf{A}}$	$2-\mathbf{\Lambda}$
GIN	$(1+\epsilon)\mathbf{I}+\mathbf{A}$	$2+\epsilon-\mathbf{\Lambda}$
	(**A-2**) polynomial function of $\mathbf{A}$	(**B-2**) polynomial function of $\mathbf{\Lambda}$
ChebNet	$\phi\mathbf{I}+\sum_{i=1}^{k}\psi_i\tilde{\mathbf{A}}^i$	$\tilde{\theta}_0\cdot 1+\tilde{\theta}_1\mathbf{\Lambda}+\tilde{\theta}_2\mathbf{\Lambda}^2+\cdots$
DeepWalk	$\frac{1}{t+1}\left(\mathbf{I}+\tilde{\mathbf{A}}+\tilde{\mathbf{A}}^2+\cdots+\tilde{\mathbf{A}}^t\right)$	$\frac{1}{t+1}\left[\cdots+\left((-1)^{t-1}+\binom{1}{t}(-1)^{t-1}\right)+\cdots\right]$
DCNN	$\psi_1\tilde{\mathbf{A}}+\psi_2\tilde{\mathbf{A}}^2+\psi_3\tilde{\mathbf{A}}^3+\cdots$	$\theta_1\mathbf{\Lambda}+\theta_2\mathbf{\Lambda}^2+\theta_3\mathbf{\Lambda}^3+\cdots$
GDC	$\psi_1\tilde{\mathbf{A}}+\psi_2\tilde{\mathbf{A}}^2+\psi_3\tilde{\mathbf{A}}^3+\cdots$	$\theta_1\mathbf{\Lambda}+\theta_2\mathbf{\Lambda}^2+\theta_3\mathbf{\Lambda}^3+\cdots$
Node2Vec	$\frac{1}{p}\mathbf{I}+\left(1-\frac{1}{q}\right)\tilde{\mathbf{A}}+\frac{1}{q}\tilde{\mathbf{A}}^2$	$\left(1+\frac{1}{p}\right)-\left(1+\frac{1}{q}\right)\mathbf{\Lambda}+\frac{1}{q}\mathbf{\Lambda}^2$
LINE/SDNE	$\psi_1\tilde{\mathbf{A}}+\psi_2\tilde{\mathbf{A}}^2$	$\theta_1\mathbf{\Lambda}+\theta_2\mathbf{\Lambda}^2$
SGC	$0\cdot\mathbf{I}+0\cdot\tilde{\mathbf{A}}+0\cdot\tilde{\mathbf{A}}^2+\cdots+1\cdot\tilde{\mathbf{A}}^K$	$\binom{K}{0}+\binom{K}{1}\mathbf{\Lambda}^1+\binom{K}{2}\mathbf{\Lambda}^2+\cdots+\mathbf{\Lambda}^n$
	(**A-3**) rational function of $\mathbf{A}$	(**B-3**) rational function of $\mathbf{\Lambda}$
Auto-Regress	$\frac{\mathbf{I}}{(1+\alpha)\mathbf{I}-\alpha\tilde{\mathbf{A}}}$	$\frac{1}{1+\alpha(1-\mathbf{\Lambda})}$
PPNP	$\frac{\alpha}{\mathbf{I}-(1-\alpha)\tilde{\mathbf{A}}}$	$\frac{\alpha}{\alpha\mathbf{I}+(1-\alpha)\mathbf{\Lambda}}$
ARMA	$\frac{b}{(\mathbf{I}-a\tilde{\mathbf{A}})}$	$\frac{b}{(1-a+a\mathbf{\Lambda})}$
ParWalk	$\frac{\alpha}{\alpha\mathbf{I}+\mathbf{I}-\tilde{\mathbf{A}}}$	$\frac{\beta}{\beta+\mathbf{\Lambda}}$

similar). However, unless all of the layers or degrees are tried, determining the appropriate number of layers or degrees is difficult. The over-smoothing problem is largely overcome by the "sending back" (i.e., teleport) behavior of rational aggregation, in which the out-degree flux is restrained even if excesses of graph convolutional layers or approximation degrees are added [53].

Three groups of spatial methods introduced above (i.e., A-1, A-2, A-3) are strongly connected under *generalization* and *specialization* relationship, as shown in Figure 1. **Generalization:** By adding more neighbors of higher rank, Linear Aggregation (A-1) can be expanded to Polynomial Aggregation (A-2). By adding reverse aggregation, Polynomial Aggregation (A-2) can be advanced to Rational Aggregation (A-3); **Specialization:** Linear Aggregation (A-1) is a special case of Polynomial Aggregation where the order is set to 1. (A-2). Rational Aggregation (A-3) degenerates into Polynomial Aggregation when reverse aggregation is removed (A-2).

5 SPECTRAL-BASED GNNS (B-0)

The use of eigen-decomposition and analysis of the weight-adjusting function (i.e., frequency filter function or frequency response function) on eigenvalues of graph matrices are both parts of graph spectral theory. In spectral-based GNNs (B-0), weights are applied to frequency components (eigenvectors) to recover the target signal using the filter's output. Accordingly, we propose a new taxonomy for graph neural networks, dividing spectral-based GNNs into three subgroups, depending on the types of response filtering functions. In addition, the same set of representative models discussed in Section 4 will be analyzed under spectral view. To facilitate comprehension of the analysis, their spatial and spectral analytical forms are listed in Table 5. The detailed transformation of equations in category B-0 is deferred to the Appendix.

5.1 Linear Approximation (B-1)

Changing the weights of frequency components in the spectrum domain has been the subject of several research. The filter function's objective is to suit the intended output by adjusting eigenvalues. Many of them have been shown to be low-pass filters [170], which implies that only low-frequency components are highlighted, i.e., the first few eigenvalues are increased, while the rest are decreased. There are several studies that may be understood as changing frequency component weights during aggregation. A linear $\mathbf{g}$ function is used in particular:

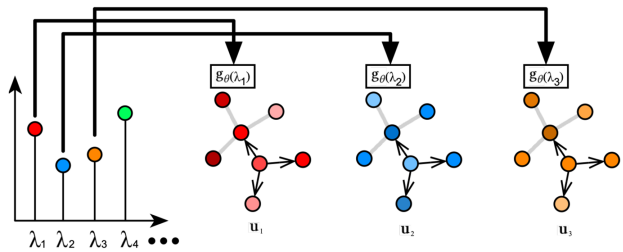

Fig. 7. Illustration of B-1: A linear function g of the eigenvalues.

$$\mathbf{Z} = \left(\sum_{i=0}^{l} \theta_i \lambda_i \, \mathbf{u}_i \, \mathbf{u}^\intercal{}_i \right) \mathbf{X} = \mathbf{U} \, \mathbf{g}_\theta(\mathbf{\Lambda}) \, \mathbf{U}^\intercal \, \mathbf{X}, \tag{49}$$

where $\mathbf{u}_i$ is the ith eigenvector, and $\mathbf{g}$ is *frequency filter function* or *frequency response function* controlled by parameters θ, with selected l lowest frequency components. The goal of $\mathbf{g}$ is to change the weights of eigenvalues to fit the target output. As shown in Figure 7, B-1 updates the weights of eigenvectors $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_2 \ldots)$ as $\mathbf{g}_\theta(\lambda)$, which is a linear function. Several state-of-the-art methods introduced in Section 4 are analyzed to provide a better understanding of this scheme.

Remark: The aforementioned methods apply linear low-pass filtering, and the only difference among them is that the bias is different (i.e., 2 for GCN, 2 for GraphSAGE, and $2+\epsilon$ for GIN). Therefore, we study the influence of bias on the filter function and define a metric:

$$w(\lambda_i) = \frac{|bias - \lambda_i|}{\sum_j |bias - \lambda_j|}, \tag{50}$$

which indicates the overall proportion change of each eigenvalue after applying the response function. A large adjusted value means that the filtering will enlarge the influence of the corresponding eigenvector. The range of the eigenvalue is in $[0, 2)$ for the normalized Laplacian matrix [183]. If let *bias* be larger than or equal to 2, then we have:

$$w(\lambda_i) = \frac{bias - \lambda_i}{N \cdot bias - \sum_j \lambda_j} = \overbrace{\frac{-1}{N \cdot bias - \sum_j \lambda_j}}^{\text{slope}} \lambda_i + \overbrace{\frac{bias}{N \cdot bias - \sum_j \lambda_j}}^{\text{intercept}}, \tag{51}$$

when $bias \geq 2$, the slope is negative, which means that the filter function is low-pass filtering: As the bias increases, the slope becomes larger, and larger weights are assigned to low-frequency spectral components. Therefore, the bias of all studies in this subsection is larger or equal to 2.

5.2 Order of Approximation (B-2)

Considering higher order of frequency, filter function can approximate any smooth filter function, because it is equivalent to applying the polynomial approximation. Therefore, introducing higher-order of frequencies boosts the representation power of filter function in simulating spectral signal.

Formally, this type of work can be written as:

$$\mathbf{Z} = \left(\sum_{i=0}^{l} \sum_{j=0}^{k} \theta_j \lambda_i^j \, \mathbf{u}_i \, \mathbf{u}^\intercal{}_i \right) \mathbf{X} = \mathbf{U} \, \mathbf{P}_\theta(\mathbf{\Lambda}) \, \mathbf{U}^\intercal \, \mathbf{X}, \tag{52}$$

where $\mathbf{g}(\cdot) = \mathbf{P}_\theta(\cdot)$ is a polynomial function. As shown in Figure 8, B-2 updates the weights of eigenvectors $(\mathbf{u}_1, \mathbf{u}_2, \ldots)$ as $\mathbf{P}_\theta(\lambda)$, which is a polynomial function.

Remark: Polynomial approximation, in theory, gets more accurate as the order grows [81–85]. It is worth noting that Linear Approximation (B-1) can be thought of as a polynomial approximation of order 1. We look into polynomial approximation on the $sign(x)$ function, comparing and contrasting all of the cases in Polynomial Approximation (B-2). Because it is difficult for any straight line to suit a jump signal, as shown in Figure 9(a), linear functions cannot accurately approximate $sign(x)$. The situation improves dramatically when polynomial approximation is used, as demonstrated in Figure 9(b).

The variance will be greatly decreased if the order of the polynomial function is increased (Figure 9(c)). To recapitulate, higher-order polynomial approximation is more accurate than lower-order polynomial approximation, but it comes at the expense of increased computational complexity. Node2Vec/LINE/SDNE with an order of 2 have lesser approximation power than those with more than 2 layers/orders, because the latter's order is predefined and can be as large as possible (e.g., ChebNet [9], DeepWalk [48], Diffusion CNN [11], Simple Graph Convolution [106]).

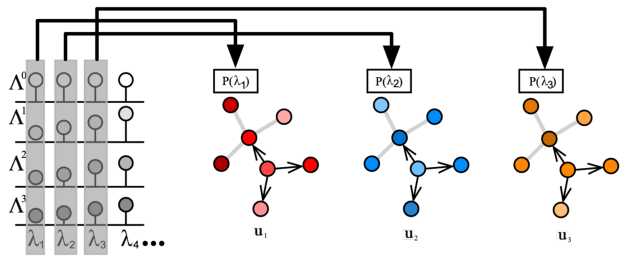

Fig. 8. Illustration of B-2: A polynomial function $\mathbf{P}$ of the eigenvalues.

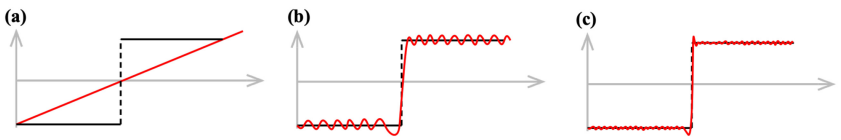

Fig. 9. Approximation for $sign(x)$ (in black): (a) linear approximation (b) polynomial approximation with low orders, (c) polynomial approximation with high orders.

5.3 Approximation Type (B-3)

Despite its widespread use and experimental success, polynomial approximation only works when applied to a smooth spectral signal. Real-world signals, however, cannot be guaranteed to be smooth. As a result, the rational approximation is employed to improve the accuracy of non-smooth signal modeling. An example of a rational kernel-based technique is as follows:

$$\mathbf{Z} = \left(\sum_{i}^{l} \frac{\sum_{j=0}^{k} \theta_j \lambda_i^j}{\sum_{m=1}^{n} \phi_m \lambda_i^m + 1} \, \mathbf{u}_i \, \mathbf{u}^\intercal{}_i \right) \mathbf{X} = \mathbf{U} \, \frac{\mathbf{P}_\theta(\mathbf{\Lambda})}{\mathbf{Q}_\phi(\mathbf{\Lambda})} \, \mathbf{U}^\intercal \, \mathbf{X}, \tag{53}$$

where $\mathbf{g}(\cdot) = \frac{\mathbf{P}_\theta(\cdot)}{\mathbf{Q}_\phi(\cdot)}$ is a rational function, and $\mathbf{P}, \mathbf{Q}$ are independent polynomial functions. Spectral methods process graph as a signal in the frequency domain. As shown in Figure 10, B-3 updates the weights of eigenvectors $(\mathbf{u}_1, \mathbf{u}_2, \ldots)$ as $\mathbf{g}_\theta(\lambda)$, which is a rational function.

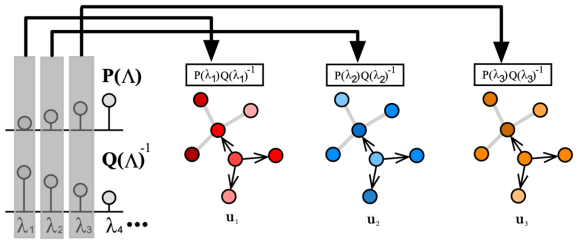

Fig. 10. Illustration of B-3: A rational function of the eigenvalues.

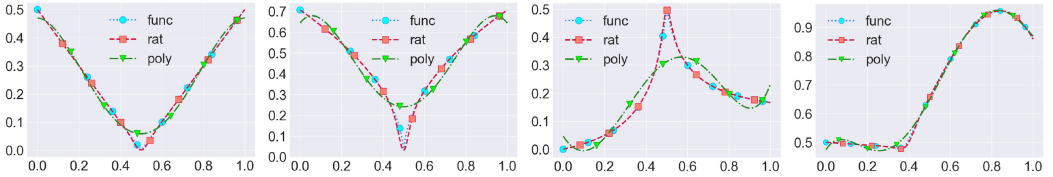

Fig. 11. Rational (rat) and polynomial (poly) approximation for several functions with discontinuity (func). From left to right: $\sqrt{|x-0.5|}$; $|x-0.5|$; $\frac{x}{10|x-0.5|+1}$; $max(0.5, sin(x+x^2)) - \frac{x}{20}$. Figures are from Reference [103].

Remark: When the function to approximate contains discontinuities, rational function has overwhelming advantage over the polynomials or linear functions. Figure 11 illustrates the difference between rational and polynomial approximation. Theoretically, rational approximation only needs exponentially less orders than that of polynomial functions [103].

6 THEORETICAL ANALYSIS

In terms of volume, spatial-based approaches outnumber spectrum-based methods in the literature [15–17, 66, 184], owing to the following reasons: (1) Spectral-based methods have a much higher computing overhead than spatial-based methods, and spectral methods are less intuitive than spatial methods. (2) Spatial-based approaches are convenient for model construction and scalability. However, from a spatial and spectral perspective, there is a tradeoff; neither has a major advantage over the other. This section outlines numerous viewpoints that demonstrate the merits and limitations of such views.

6.1 Uncertainty Principle: Global vs. Local Perspectives

Spectral-based approaches decompose data into orthogonal frequency components and examine graph filtering from the spectral domain with a global perspective. Each frequency indicates a global basis: Low-frequency components emphasize local weights with little variation, whereas high-frequency components are linked to significant variance in neighborhood. In other words, the Laplacian spectrum reflects topological properties: The first few eigenvalues are related with substantial community structure, while the last few eigenvalues indicate the graph's bipartiteness [185, 186]. A typical low-pass filtering function for eigenvalues is shown on the left of Figure 12, which raises small eigenvalues while decreasing adjusted values for large eigenvalues. Only low-frequency components are maintained in this scenario, and neighbors have little variance.

Filtering patterns from the local neighborhood are characterized by spatial-based approaches. Most GNNs assume homophily among neighbors, so signals traversing across the neighborhood are smooth or with little variation, which is exactly the same concept as low-pass filtering. The relationship between low-pass filtering in the spectral domain (left) and its effects in the spatial domain (middle and right) is depicted in Figure 12 [186]. It appears at first glance that global and

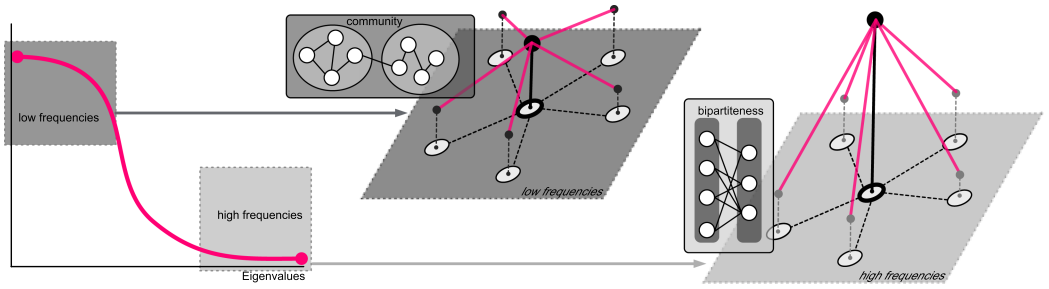

Fig. 12. Connection between the spatial and spectral perspective.

local viewpoints differ greatly, but on closer inspection, they depict the same signal in very different ways: Filtering in the spectral domain that does low-pass/high-pass filtering is analogous to learning which neighbors are similar/dissimilar. While it is true that the two types of observations yield similar results, this does not entail that they are identical. It is impossible to know an unknown quantity's value with absolute confidence in quantum mechanics because of the Heisenberg's uncertainty principle [187]. Specifically,

$$\Delta_t^2 \Delta_\omega^2 \geq \frac{1}{4}, \tag{54}$$

where Δ_t and Δ_ω denote time spread and frequency spread, respectively. Signal concentration can also be impacted by the concentration of time and frequency. A graph representing the tradeoff between a signal's localization on a graph and in its spectral domain is created, which is influenced by the uncertainty principle of quantum mechanics [188]. A lower bound on the product of the two spreads is obtained by quantifying the spreads in the vertex and the spectral domain of a graph signal x. This result suggests that applying spatial-based GNNs results in accuracy loss in the spectral domain, while using spectral-based GNNs results in accuracy loss in the spatial domain. In sum: **(1) Global and local perspectives are strong connected:** Global observation is the process of generalizing details in local settings, while localized understanding provides details of the global picture. **(2) Global and local perspectives outperform each other in their own domains:** Clear local context tends to muddy the big picture, but a focus on global context diminishes the smaller picture. GNNs are able to expand the range of options with the addition of other preexisting works, which bridge the gap between global and local views or between spectral and spatial information, to improve the expressive potential of GNNs [58, 189–193]. According to the information above, we can state that no model can be flawless from a global or local perspective; it is only possible to have a proper tradeoff between. As shown in Table 6, four aspects are compared: **Methodology:** Spatial approaches describe local regions while working bottom-up and identify global patterns using a graph frequency approach. However, spectral methods work top-down, beginning with a graph and ending with a global observation. **Computation:** To use spatial approaches, one has to carry out a number of steps on their local region before convergence is achieved. With spectral approaches, you can get a critical component with a single-step computation. **Space Complexity:** The high space complexity of spectral approaches is associated with the massive memory storage required to load the full graph. If memory is sufficient, then the full graph can be covered by using spatial methods. However, for a smaller graph, you can choose to cover it using samplings such as sampling of regions or paths. **Stability:** To create accurate, consistent results, spatial analysis methods need to apply iterative algorithms, therefore the outcomes will vary. Eigen-decomposition is a unique feature of spectral approaches if no same eigenvalues.

Guidance of Choosing Spatial and Spectral Methods. The user can choose between spatial or spectral GNNs, depending on their previously described properties. **Distributed and Online**

Table 6. Comparison between the Spatial (A-0) and Spectral (B-0) Methods

	Methodology	Computation	Space Complexity	Stability
Spectral	Global	One-step	High	Exact
Spatial	Local	Iterative	Low	Approximate

Table 7. Comparison on Time Complexity and Expressive Power

	Linear (A-1, B-1)	Polynomial (A-2, B-2)	Rational (A-3, B-3)
Time	$O(N^2F)$	$O(N^{K+1}F)$	$O(N^{K+1}F + N^3)$
Expressivity	$O(1)$	$O(1/K)$	$O(\exp^{-\sqrt{K}})$

Learning: Spatial method is easily converted to distributed learning [194, 195], whereas spectral is difficult to transfer. Even if it is possible to approximate spectrum technique using a neural network model [196] and then use distributed learning on a neural network [197], new nodes and edges must be retrained from scratch. Alternatively, the spatial technique can effectively manage online learning with streaming data [198]. **Global View:** The spectral method may provide a global perspective that the spatial method lacks. In situations where the group form is not spherical, the spatial method may disregard this influence and continue to follow the circular shape as a prospective group [199, 200]. This can be remedied by employing clustering before to the spatial technique [201]. This also renders spatial methods more locally interpretable and obscures their global perspective.

6.2 Comparison between Linear, Polynomial, and Rational Methods

Linear methods (A-1 and B-1) have a time complexity of $O(N^2F)$ due to the matrix multiplication of $\mathbf{A}\,\mathbf{X}$. Accordingly, polynomial and rational method are analyzed in Table 7 where K is the order number. To compare their expressive power, the convergence rate on challenging jump signal is employed as a benchmark [103] (a smooth signal cannot distinguish them). As shown in Table 7, rational methods (A-3 and B-3) converge exponentially faster than linear methods (A-1/B-1), and polynomial methods (A-2/B-2) converge linearly faster than linear methods (A-1/B-1). Therefore, there is a tradeoff between the expressive power and computational efficiency. Linear methods (A-1/B-1) have the best efficiency but only capture the linear relationship. Rational methods (A-3/B-3) consume the most considerable overhead but could tackle more challenging signals.

Experiments are undertaken to highlight our theoretical analysis of spatial and spectral approaches by comparing the differences between the three underlying groups. We chose one typical technique for linear filter [8], polynomial filter [9], and rational filter [109]. Note that we only distinguish them in the function of $\mathbf{A}$ or $\mathbf{\Lambda}$, keeping all the other configurations the same. The dataset includes representative homophily and heterophily datasets [129, 140, 148, 202–204]. The evaluation code is released at https://github.com/XGraph-Team/Spectral-Graph-Survey.

As shown in Figure 13 (Left), there is no significant difference in classification accuracy in the homophily dataset between the three models, with the exception of ChebNet, which performs marginally better in *PubMed* and encounters an out-of-memory error in *physics* dataset. ChebNet exhibits inferior accuracy on the *Computers* dataset, whereas GCN is significantly superior. Figure 13 (right) illustrates that ARMA consistently outperforms the others when performing the same task on the heterophily dataset, while ChebNet consistently outperforms GCN. This demonstrates the benefits of the advanced filter function, as theoretically analyzed in Section 3.2. In terms of runtime, as shown in Figure 14, ChebNet is often beyond $\log 1$, while GCN never goes beyond $\log 1$. The Rational method involves a higher volume of matrix operations compared to Cheb-Net, occasionally exceeding the "$\log 1$" threshold. However, its iterative optimization process enables

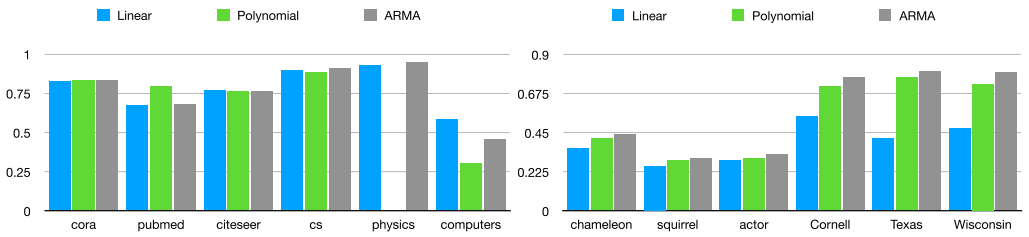

Fig. 13. Performance comparison. **Left**: homophily dataset; **Right**: heterophily dataset.

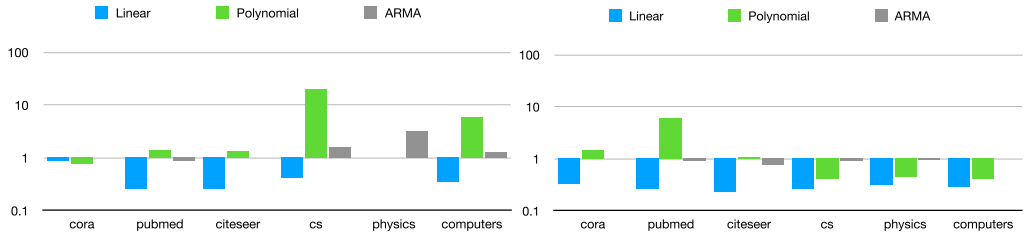

Fig. 14. Runtime comparison. **Left**: homophily dataset; **Right**: heterophily dataset. (Y-axis is log-based).

it to execute slightly faster than ChebNet. Despite this, the Rational method consistently exhibits slower performance than GCN. In terms of runtime, as shown in Figure 14, ChebNet frequently exceeds log 1, whereas GCN never does. Rational method involves more matrix operations than ChebNet, so it sometimes exceeds log 1. However, due to optimization in implementation (iterative algorithm), the rational method is slightly faster than ChebNet but consistently slower than GCN. This verifies our analysis in Section 6.2.

7 EXEMPLIFY THE PROPOSED FRAMEWORK

Over-smoothing and large-scale difficulties are two of the most difficult issues for existing GNNs, and numerous recent publications have proposed various approaches to address them. We will show in this section that all of the enhancements are still covered by our framework.

7.1 Sampling Point of View

The sampling mechanism is used as a spatial method for managing large graphs. Subgraph sampling and random walk are popular approaches.

7.1.1 Subgraph Sampling. For an early work, *GraphSAGE* [10] applies uniform node sampling for each batch, which is equivalent to subgraph sampling. The likelihood of transfer, therefore, follows random normalization (i.e., $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\mathbf{A}$), which makes it part of Linear Aggregation (A-1). In the majority of follow-up works, the same methodology is used [118]: (1) build a local graph convolution for the input graph, (2) sample nodes in each layer, and (3) optimize parameters in graph convolution. Steps (2) and (3) proceed iteratively to update the weights via stochastic gradient descent [65, 99, 118, 205–207].

To avoid the recursive neighborhood expansion, FastGCN [65] treats graph convolutions as integral transformation of embedding functions and proposes Monte Carlo approach to estimate the integral. FastGCN employs importance sampling independently for each layer and reduces variance, cutting down the number of sampling nodes to constant size for all layers, exponentially shrinking the computational cost. FastGCN is proved to be importance sampling, which is better

than uniform sampling, but still suffers from unstable learning when no neighbors are selected for one node and activation is zero. To avoid taxing calculation of activation, Stochastic GCN [205] further uses the historical activation in the previous layer to avoid redundant re-evaluation. With adaptive sampling, nodes on subsequent layers are sampled to speed up GraphSAGE and FastGCN [99]. **Learnable graph convolutional layer (LGCL)** [206] selects a fixed number of neighboring nodes for each feature based on value ranking and transforms graph into 1-D data that is compatible with normal convolution networks. Similarly, A scalable GCN samples a fixed number of nodes, with different sampling policy called **frontier sampling (FS)**. FS maintains a constant size frontier set consisting of several vertices, which is randomly popped out with a degree-based probability distribution [208]. Cluster-GCN [209] samples a community of nodes determined by a graph clustering algorithm and computes the graph convolution within each community.

7.1.2 Random Walk. To derive node-level representations with word2vec [210–212], various random walk algorithms are proposed [48, 49, 93, 118, 166, 213]. Paths are viewed as complete sentences, and nodes are viewed as individual words. Transition probability among nodes approximates to a random walk normalized adjacency matrix if enough random walks or uniform sampling has been performed on the paths.

In the preceding section's analysis, it was noted that *DeepWalk* [48] generates multiple random paths from a graph. This results in a transfer probability for the random walk characterized by $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-1}\boldsymbol{A}$. Let the window size of skip-gram be $2t + 1$ and the index of current node is $t + 1$. Therefore, the updated representation is $\mathbf{Z} = \frac{1}{t+1}\mathbf{P}(\tilde{\mathbf{A}})\,\mathbf{X}$ (Equation (23)). One popular word2vec configuration, i.e., **skip-gram with Negative Sampling (SGNS)**, assumes a corpus of words w and their context c. Following the work by Levy and Goldberg [214], SGNS is implicitly factorizing:

$$\log\left(\frac{|(w,c)|\cdot|\mathcal{D}|}{|w|\cdot|c|}\right) - \log b = \boldsymbol{A}\boldsymbol{B}^{\top},$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ denote matrix of current node and its neighbors, respectively. $|(w,c)|$, $|w|$, $|c|$, and $\mathcal{D}$ denote the number of times word-context pair (w,c), word w, context c, and corpus size, respectively; b is the number of negative samples. Accordingly, Reference [69] derived an exact format as:

$$\log\left(\mathbf{P}(\tilde{\mathbf{A}})\right) - \log(b) = \log\left(\frac{|E|}{T}\left(\sum_{r=1}^{T}(\mathbf{D}^{-1}\,\mathbf{A})^{r}\right)\mathbf{D}^{-1}\right) - \log(b),$$

where $|E|$ and T represent edge number and step size, respectively. Consequently, the matrix designated for decomposition remains a polynomial of $\mathbf{A}$. *Node2Vec* [49] defines a second-order random walk to control the balance between **Breadth First Search (BFS)** and **Depth First Search (DFS)**. Assuming the random walk is sufficiently sampled, Node2Vec's second order can be rewritten to decompose matrix [69]:

$$\log\left(\mathbf{P}(\tilde{\mathbf{A}})\right) - \log(b) = \log\left(\frac{\frac{1}{2T}\sum_{r=1}^{T}\left(\sum_u X_{w,u}\underline{\tilde{\mathbf{A}}}^{r}_{c,w,u} + \sum_u X_{c,u}\underline{\tilde{\mathbf{A}}}^{r}_{w,c,u}\right)}{\left(\sum_u X_{w,u}\right)\left(\sum_u X_{c,u}\right)}\right) - \log b,$$

where Node2Vec is demonstrated to be polynomial methods. *LINE* [93] and *SDNE* [166] learn the node representations within the first- and second-order neighbors, which can be treated as unconstrained version of Node2Vec:

$$\log\left(\mathbf{P}(\tilde{\mathbf{A}})\right) - \log(b) = \log\left(|E|\,\tilde{\mathbf{A}}\right) - \log b.$$

GraphSAINT [118] employs multiple sampling polices but the best one is random walk. *PinSAGE* [178] improves the efficiency of GraphSAGE [10] by taking the top several neighbors with highest normalized visit counts.

Remark: Sampling methods, as a spatial methodology, seek both variance reduction and efficiency. Subgraph and random walk are equal with enough samples, since they traverse the entire network with the transition probability associated with graph connection. A-3 and B-3 do not, however, include any sample methods, mainly due to their higher computational complexity. Space complexity may be improved when sampling methods are used, but there is no guarantee that the time complexity will be much reduced because the enormous number of steps may be needed before convergence.

7.2 Over-smoothing Point of View

When a node's neighbors are highly dissimilar and when a node's embedding is already similar to its neighbors, neighborhood aggregation is not advantageous. Most GNNs perform poorly when stacking many layers, which is called the over-smoothing issue. Many related works aiming to solve the over-smoothing issue [101, 107, 119, 123, 125, 173, 193, 215–222] can be reduced to one category of our proposed framework. H2GCN [193] proposed a method that combines direct neighbors with higher-order, which is equivalent to Polynomial Aggregation (A-2). Deep GCN [107, 215, 218] developed a model with a residue module, dense connection, and dilated aggregation, which learns the weights of all different orders of neighbors. This is equivalent to Polynomial Aggregation (A-2). GPR [52] generalizes PageRank and found the equivalence of GPR and Polynomial Approximation (B-2). JKNet [101] also follows the same residue methodology as Deep GCN. DAGNN [125] stacks multiple layers, which uses different orders of propagation with the learnable weights, which makes it belong to Polynomial Aggregation (A-2). PairNorm [173] presents a two-step method that includes centering and re-scaling, which mitigates the over-smoothing from graph convolution. Therefore, PairNorm is equivalent to Rational Propagation (A-3), since re-scaling is similar to do propagation and restart at the same time. Reference [216] designed an adaptive method to dynamically adjust the weights between low-frequency and high-frequency components, resulting in two peaks in the spectral domain. This could also be modeled by Rational Aggregation (A-3) with its accuracy in jump signals. DropEdge [119, 219] randomly drops a certain number of edges to avoid over-smoothing, which can be categorized as Rational Aggregation (A-3), since dropping edge prevents the propagation and thereby provides a probability of keeping the original values of nodes. GCNII [123] applies *initial residual*, which combines the smoothed representation with an initial residual connection to the first layer and *identity mapping*, which adds an identity matrix to the weight matrix. *Initial residual* is a trick that PPAP [53] uses, which enables itself to retard the over-smoothing by keeping partial previous representations. *Identity mapping* further remains one original representation to slow down the spreading of over-smoothing propagation.

Remark: A-1 lacks state-of-the-art methods, implying it is vulnerable to over-smoothing. Applying A-1 many times with learnable weights for different ordering equates to A-2. So, A-2 might balance low (raw representations) and high orders (smoothed representations). However, too many A-1 operations may result in over-smoothing. So, for A-2, precise order configuration is required. No matter the number of orders or layers, A-3 reserves a proportion of the final representation as raw representation, making itself robust to over-smoothing.

7.3 Inverse Problem Point of View

Let us consider a link represented by the variable e, which establishes a connection between two nodes, denoted as i and j. **Graph neural networks (GNNs)** can be formally expressed as a function $f : (v_i, e) \rightarrow v_j$. The function denoted as f, which is equivalent to Equation (19), incorporates the application of graph convolution and neural networks:

$$f : (\mathbf{A}, X_i) \rightarrow X_j, \tag{55}$$

which implies that the features of node i, denoted as X_i, undergo a smoothing process to become part of the features of node j, represented as X_j.

The reconstruction of graphs [223] poses an inverse problem to that of Equation (55). The issue at hand is frequently illustrated through a task within the realm of **knowledge graphs (KGs)**, which is commonly referred to as link completion. The objective can be articulated in the following manner:

$$f : (X_i, X_j) \rightarrow \mathbf{A}, \quad \text{or in single-instance format} \quad f : (h, t) \rightarrow r, \tag{56}$$

where relation between node i and j is denoted by $\mathbf{A}$, while the head and tail entity vertices connected by relation r are represented by h/t. The semantic similarity-based scoring function is a prevalent approach in KGs for assessing the veracity of facts by means of semantic matching, as reported by Reference [224]. The conventional method involves utilizing a broad multiplicative structure, denoted as $h^\top \mathbf{A} \approx t^\top$, in which the function $\mathbf{A}$ is dependent on the relationship r. This formulation exhibits a high degree of consistency with the graph convolution operator. It is worth noting that empirical evidence indicates that multiplicative models ($h^\top \mathbf{A} \approx t^\top$) outperform additive models ($h + r \approx t$) in the context of KGs [225, 226].

Instead of allowing the usage of arbitrary linear maps represented by random matrices $\mathbf{A}$ to depict relations, a particular collection of matrices has been investigated for linear maps that demonstrate favorable characteristics. A theoretical analysis posits that normal matrices possess desirable theoretical properties that are commonly sought in relational modeling [227]. The diagonalization of these entities can be achieved through unitary means, thereby enabling their analysis through the spectral theorem as described in Reference [228]. The Graph Laplacian matrix is a normal matrix that is utilized to characterize the feature-wise relationship or covariance of entities, as per its definition. This matrix can elucidate the physical significance of the learned $\mathbf{A}$. The aim of link completion is to establish the feature-wise relationship ($\mathbf{A}$) between training pairs of (h, t) by utilizing fundamental statistical measures such as covariance. Following this, an objective function is optimized to precisely match the training data, i.e., $\min \Delta(h^\top P(\mathbf{A}), t)$, where Δ denotes a difference metric function between $h^\top P(\mathbf{A})$ and t. The degree of inconsistency can be mathematically represented as $< h^\top P(\mathbf{A}), t > = h^\top P(\mathbf{A})t$, which bears a resemblance to the notion of Dirichlet energy of a signal across a graph framework. Assuming similarity between h and t, and their smoothness over a function of feature-wise graph Laplacian $P(\mathbf{A})$, it is expected that the value of $h^\top P(\mathbf{A})t$ will be minimized. Otherwise, the magnitude of the value will be significant. As demonstrated in Reference [226], the diagonal matrix $\mathbf{A}$ can be utilized to detect features that display no covariance, thereby signifying the independence of all features during the learning phase. Rewrite the above-mentioned link completion task in matrix format (suppose multiple pairs of (h, t)), and we have *feature-level* graph convolution:

$$\overbrace{H}^{N\times d}\ \overbrace{P(\mathbf{A})}^{d\times d} \rightarrow \overbrace{T}^{N\times d}, \tag{57}$$

where N is used to denote the number of instances, whereas d is used to represent the number of features. Thus, the aforementioned equation, denoted as Equation (57), converts the given dimension d of H into d dimensions of T. Given that matrix $\mathbf{A}$ is normal, it is possible to consider matrix P as a filter function applied to its eigenvalues, as previously discussed. Existing studies on knowledge graphs have observed that P is modeled as a linear function (e.g., References [229, 230]) with the form of (A-1/B-1), or as a polynomial function (e.g., Reference [231]) with the form of (A-2/B-2). Furthermore, GNN functions as a nascent encoder for entity graph and employs prevalent GNN models in KGs [230, 232–234]. Performing graph convolution at the *entity-level*

can be accomplished through a standard GNN scheme, which is straightforward to implement:

$$\overbrace{P(\mathbf{A})}^{N\times N}\ \overbrace{H}^{N\times d}\ \rightarrow\ \overbrace{T}^{N\times d}, \tag{58}$$

where the convolution of H is performed by $P(\mathbf{A})$ at the entity level, whereby each node is averaged by its neighbors. It is worth noting that Equations (57) and (58) operate at distinct levels, with the latter typically utilized for node classification as opposed to link prediction.

8 LIMIT, OPEN CHALLENGES, AND CONCLUSION

Theoretical Understanding - Most recent research has examined spatial and spectral approaches individually, and even from a theoretical standpoint. [61, 145, 235]. However, it is unclear how these two distinct interpretations can be connected. This survey demonstrates that the theoretical advantage of rational function over the others may be proved, It is still unclear how the learning approach can be optimally structured to execute this advantage. Using the partial differential equation [236], in which the diffusion equation and wave equation are analogous to polynomial and rational filters, provides an alternative viewpoint to integrate spatial and spectral views.

Directed Graph - Most contemporary work, especially spectral methods, only handle undirected graphs, due to symmetric graph matrix is readily available with off-the-shelf techniques. Many related works integrate or sum up asymmetric adjacency matrices from bi-direction into symmetric matrix, which avoids decomposing asymmetric matrix directly. It is possible to decompose asymmetric matrix by some techniques such as the Jordan norm [237]; asymmetric matrices can be used to express properties such as graph and filter complexity. Directed graphs and their decomposition can also be achievable in a certain type of geometric space called a graph manifold. As shown in Section 2.4 of Reference [238], spectral filters can be defined on directed graphs represented by non-symmetric adjacency matrices with Hermitian transpose.

Dynamic Graph - The existing work model with GNNs and RNNs is built using graph convolutional networks and recurrent neural networks, which lack transparency. Due to the limited expressive power of RNNs, the task is constrained to prediction, and it is unable to perform long-term sequence processing well. Graph dynamics has a large number of valuable tasks, such as inferring the structure of a graph, constructing a joint dynamic of structure and attributes, and exploring the connection between structure and mass flow. It is possible to use graph spectra to detect the patterns in dynamics of graph [239–241]. Due to the challenge of the long-dependency of a path, the spectral method can also provide a potential way to model trajectory prediction, as the spectra of trajectory implicitly reveal its relationship with the whole graph [242].

Higher-order Interaction and Combinatorial Optimization - Most current work falls into the first-order relationship at the node-level, but higher-order interaction is either ignored or implicitly included. Existing explainable learning also focuses on the neighbors' identification [243]. Also, the existing work pays the most attention to neighbors, but remote connection or higher-order relationship with the other nodes receives little attention. Hypergraphs provide a possibility to model the combinatorial effect [244–246] Hodge Laplacian [247, 248] and simplicial complex [249–251] provide more theoretical tools for modeling this combinatorial effect.

Multilayer Network - In real-world biological and engineering systems, units are often interconnected across multiple networks. Failure in one network can trigger cascading failures across systems. Likewise, misinformation spread via social media in cyber-physical systems can lead to group-level risks like epidemic outbreaks. Such interconnected networks can be represented by *multilayer networks* that produce new degrees of freedom via coupling interactions. Such "new

physics" is prevalent in multilayer systems, but they are still poorly understood [252–255]. Graph neural network research in this crucial area is scarce [256].

APPENDICES

Details of equation transformation for Section 5 are listed below.

A B-1

A.1 Graph Convolutional Network (GCN)

Rewriting GCN [8] in spectral domain, we have:

$$\mathbf{Z} = \tilde{\mathbf{A}}\,\mathbf{X} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\,\mathbf{D}^{-\frac{1}{2}}\,\mathbf{X} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D}-\mathbf{L}+\mathbf{I})\,\mathbf{D}^{-\frac{1}{2}}\,\mathbf{X} = (\mathbf{I}-\mathbf{D}^{-\frac{1}{2}}\,\mathbf{L}\,\mathbf{D}^{-\frac{1}{2}}+\mathbf{I})\,\mathbf{X} = \mathbf{U}(2-\mathbf{\Lambda})\,\mathbf{U}^{\intercal}\,\mathbf{X}, \tag{59}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A}+\mathbf{I})\mathbf{D}^{-\frac{1}{2}}$ is renormalization of $\tilde{\mathbf{A}}$. Therefore, the frequency response function is $\mathbf{g}(\mathbf{\Lambda}) = 1-\mathbf{\Lambda}$, which is a low-pass filter, i.e., smaller eigenvalues that correspond to low frequency components are assigned with a larger value.

A.2 GraphSAGE

Considering the MEAN aggregation as example, we can rewrite GraphSAGE [10] in matrix form:

$$\mathbf{Z} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{I}+\mathbf{A})\,\mathbf{D}^{-\frac{1}{2}}\,\mathbf{X} = (\mathbf{I}+\tilde{\mathbf{A}})\,\mathbf{X} = (2\,\mathbf{I}-\tilde{\mathbf{L}})\,\mathbf{X} = \mathbf{U}(2-\mathbf{\Lambda})\,\mathbf{U}^{\intercal}\,\mathbf{X}\,. \tag{60}$$

Hence, the frequency response function is $\mathbf{g}(\mathbf{\Lambda}) = 2-\mathbf{\Lambda}$, which is a low-pass filtering. Note that GraphSAGE's normalization is different from GCN, which utilizes the renormalization trick.

$$(I + D^{-1}A)\,\mathbf{X} \tag{61}$$

$$\mathbf{U}\left(1+\Lambda_P\right)\mathbf{U}^{\intercal}\,\mathbf{X}, \tag{62}$$

where $P = D^{-1}A$

A.3 Graph Isomorphism Network (GIN)

A multi-layer neural network has the capability to conform to the normalization scale (i.e., normalization) [8], so GIN [61] can be rewritten as:

$$\mathbf{Z} = \mathbf{D}^{-\frac{1}{2}}[(1+\epsilon)\,\mathbf{I}+\mathbf{A}]\,\mathbf{D}^{-\frac{1}{2}}\,\mathbf{X} = \mathbf{D}^{-\frac{1}{2}}[(2+\epsilon)\,\mathbf{I}-\tilde{\mathbf{L}}]\,\mathbf{D}^{-\frac{1}{2}}\,\mathbf{X} = \mathbf{U}(2+\epsilon-\mathbf{\Lambda})\,\mathbf{U}^{\intercal}\,\mathbf{X}\,. \tag{63}$$

GIN can be seen as a generalization of GCN or GraphSAGE without normalized adjacency matrix $\mathbf{A}$. The frequency response function is $\mathbf{g}(\mathbf{\Lambda}) = 2+\epsilon-\mathbf{\Lambda}$, which is low-pass.

B B-2

B.1 ChebNet

As analyzed in Equation (21), ChebNet [64] can be written as:

$$\sum_{k=0}^{K-1}\theta_k T_k(\tilde{\mathbf{L}})\,\mathbf{X} = (\tilde{\theta_0}\,\mathbf{I}+\tilde{\theta_1}\tilde{\mathbf{L}}+\tilde{\theta_2}\tilde{\mathbf{L}}^2+\cdots)\,\mathbf{X}, \tag{64}$$

where $T_k(\cdot)$ is the Chebyshev polynomial and θ_k is the Chebyshev coefficient. $\tilde{\theta}$ is the coefficient after expansion and reorganization. Therefore, we can rewrite it as:

$$\sum_{k=0}^{K-1}\theta_k T_k(\tilde{\mathbf{L}})\,\mathbf{X} = \mathbf{U}(\tilde{\theta_0}\cdot 1+\tilde{\theta_1}\,\mathbf{\Lambda}+\tilde{\theta_2}\mathbf{\Lambda}^2+\cdots)\,\mathbf{U}^{\intercal}\,\mathbf{X}, \tag{65}$$

where spectral response function is $\mathbf{g}(\mathbf{\Lambda}) = \tilde{\theta_0}\cdot 1+\tilde{\theta_1}\,\mathbf{\Lambda}+\tilde{\theta_2}\mathbf{\Lambda}^2+\cdots = \mathbf{P}(\mathbf{\Lambda})$.

B.2 DeepWalk

Starting from Equation (23), DeepWalk [48] can be rewritten as:

$$
\begin{aligned}
\mathbf{Z} &= \frac{1}{t+1}(\mathbf{I}+\tilde{\mathbf{A}}+\tilde{\mathbf{A}}^2+\cdots+\tilde{\mathbf{A}}^t)\,\mathbf{X} \\
&= \frac{1}{t+1}(\mathbf{I}+(\mathbf{I}-\tilde{\mathbf{L}})+(\mathbf{I}-\tilde{\mathbf{L}})^2+\cdots+(\mathbf{I}-\tilde{\mathbf{L}})^t)\,\mathbf{X} \\
&= \frac{1}{t+1}\left(2\,\mathbf{I}+(-1-2-3-\cdots)\,\tilde{\mathbf{L}}+(1+3+6+\cdots)\,\tilde{\mathbf{L}}^2+\cdots\left((-1)^{t-1}+\begin{pmatrix}1\\ t\end{pmatrix}(-1)^{t-1}\right)\tilde{\mathbf{L}}^{t-1}+(-1)^t\,\tilde{\mathbf{L}}^t\right)\mathbf{X} \\
&= (\theta_0\,\mathbf{I}+\theta_1\,\tilde{\mathbf{L}}+\theta_2\,\tilde{\mathbf{L}}^2+\cdots+\theta_t\,\tilde{\mathbf{L}}^t)\,\mathbf{X} \\
&= \mathbf{U}(\theta_0+\theta_1\,\mathbf{\Lambda}+\theta_2\,\mathbf{\Lambda}^2+\cdots+\theta_t\,\mathbf{\Lambda}^t)\,\mathbf{U}^\intercal\,\mathbf{X} \\
&= \mathbf{U}\,\mathbf{P}_{k=r}(\mathbf{\Lambda})\,\mathbf{U}^\intercal\,\mathbf{X},
\end{aligned}
$$

where $\mathbf{g}(\mathbf{\Lambda}) = \theta_0+\theta_1\,\mathbf{\Lambda}+\theta_2\,\mathbf{\Lambda}^2+\cdots+\theta_t\,\mathbf{\Lambda}^t$, and all parameters θ_i are determined by the predefined step size t.

B.3 Scalable Inception Graph Neural Networks (SIGN)

Substituting $\tilde{\mathbf{A}} = \mathbf{I}-\tilde{\mathbf{L}}$ in Equation (28), it can be rewritten as:

$$
\hat{\mathbf{Z}} = \sum_r \omega_r \widehat{(\mathbf{I}-\tilde{\mathbf{L}})^r}\,\mathbf{X} = \mathbf{U}\,\widehat{\mathbf{P}_{k=r}(\mathbf{\Lambda})}\,\mathbf{X}\,\mathbf{U}^\intercal. \tag{66}
$$

B.4 Graph Diffusion Convolution (GDC)

Substituting $\tilde{\mathbf{A}} = \mathbf{I}-\tilde{\mathbf{L}}$, general case in Equation (30) can be written as:

$$
\mathbf{Z} = \sum_{k=0}^{\infty}\theta_k(\mathbf{I}-\tilde{\mathbf{L}}) = \mathbf{U}\sum_{k=0}^{\infty}\theta_k(1-\mathbf{\Lambda})^k\,\mathbf{U}^\intercal = \mathbf{U}\,\mathbf{P}(\mathbf{\Lambda})\,\mathbf{X}\,\mathbf{U}^\intercal. \tag{67}
$$

B.5 Diffusion Convolutional Neural Networks (DCNN)

As analyzed in Equation (24), DCNN [11] can be transformed with $\tilde{\mathbf{A}} = \mathbf{I}-\tilde{\mathbf{L}}$ as:

$$
\mathbf{Z} = \mathbf{P}(\mathbf{I}-\tilde{\mathbf{L}})\,\mathbf{X} = \mathbf{U}\,\mathbf{P}(\mathbf{\Lambda})\,\mathbf{X}\,\mathbf{U}^\intercal, \tag{68}
$$

which is equivalent to ChebNet, and parameters θ_i are learnable.

B.6 Node2Vec

Node2Vec [49] can be rewritten in matrix form as Equation (32). Then it can be transformed and reorganized after substituting $\tilde{\mathbf{A}} = \mathbf{I}-\tilde{\mathbf{L}}$:

$$
\mathbf{Z} = \left[\left(1+\frac{1}{p}\right)\mathbf{I}-\left(1+\frac{1}{q}\right)\tilde{\mathbf{L}}+\frac{1}{q}\tilde{\mathbf{L}}^2\right]\mathbf{X} = \mathbf{U}\left[\left(1+\frac{1}{p}\right)-\left(1+\frac{1}{q}\right)\mathbf{\Lambda}+\frac{1}{q}\mathbf{\Lambda}^2\right]\mathbf{U}^\intercal\,\mathbf{X}. \tag{69}
$$

Therefore, Node2Vec's frequency response function is:

$$
\mathbf{g}(\mathbf{\Lambda}) = \left(1+\frac{1}{p}\right)-\left(1+\frac{1}{q}\right)\mathbf{\Lambda}+\frac{1}{q}\mathbf{\Lambda}^2, \tag{70}
$$

which integrates a second-order function of $\mathbf{\Lambda}$ with predefined parameters, i.e., p and q.

B.7 LINE/SDNE

As described in Equation (36), LINE [165] and SDNE [166] can be rewritten as:

$$
\mathbf{Z} = \tilde{\mathbf{A}}\,\mathbf{X}+\alpha\,\tilde{\mathbf{A}}^2\,\mathbf{X} = (\mathbf{I}-\tilde{\mathbf{L}})\,\mathbf{X}+\alpha(\mathbf{I}-\tilde{\mathbf{L}})^2\,\mathbf{X} = \mathbf{U}[(\mathbf{I}-\mathbf{\Lambda})+\alpha(1-\mathbf{\Lambda})^2]\,\mathbf{X}\,\mathbf{U}^\intercal = \mathbf{U}\,\mathbf{P}_{k=2}(\mathbf{\Lambda})\,\mathbf{U}^\intercal\,\mathbf{X}, \tag{71}
$$

where response function $\mathbf{g}(\mathbf{\Lambda}) = \mathbf{\Lambda}+\alpha\,\mathbf{\Lambda}^2$ is a polynomial function with order 2.

B.8 Simple Graph Convolution (SGC)

As analyzed in Equation (38), SGC can be transformed as:

$$\begin{aligned}\mathbf{Z} = (\mathbf{I} - \tilde{\mathbf{L}})^k \mathbf{X} &= \left[\binom{k}{0} \mathbf{I} + \binom{k}{1} \tilde{\mathbf{L}}^1 + \binom{k}{2} \tilde{\mathbf{L}}^2 + \cdots + \tilde{\mathbf{L}}^k \right] \mathbf{X} \\ &= \mathbf{U} \left[\binom{k}{0} + \binom{k}{1} \mathbf{\Lambda}^1 + \binom{k}{2} \mathbf{\Lambda}^2 + \cdots + \mathbf{\Lambda}^k \right] \mathbf{U}^\top \mathbf{X},\end{aligned}$$

where spectral response function is a polynomial function of order k:

$$\mathbf{g}(\mathbf{\Lambda}) = \binom{k}{0} + \binom{k}{1} \mathbf{\Lambda}^1 + \binom{k}{2} \mathbf{\Lambda}^2 + \cdots + \mathbf{\Lambda}^k .$$

B.9 Improved GCN (IGCN)

By stacking multiple layers, IGCN [170] is proposed as:

$$\mathbf{Z} = \tilde{\mathbf{L}}^k \mathbf{X} = \mathbf{U} \mathbf{\Lambda}^k \mathbf{U}^\top \mathbf{X}, \tag{72}$$

where the spectral response function is a polynomial function with order k.

C B-3

C.1 Auto-regressive Filter

Label propagation (LP) [175–177] is a prevailing methodology for graph-based learning. The objective of LP is two-fold: one is to extract embeddings that match with the label, the other is to be similar with neighboring vertices. Label can be treated as part of node attributes, so we can have:

$$\mathbf{Z} = (\mathbf{I} + \alpha \tilde{\mathbf{L}})^{-1} \mathbf{X} = \mathbf{U} \frac{1}{1 + \alpha(1 - \mathbf{\Lambda})} \mathbf{U}^\top \mathbf{X} . \tag{73}$$

C.2 PPNP

Personalized PageRank (PPNP) [53] can obtain node's representation via teleport (restart) probability α, which indicates the ratio of keeping the original representation:

$$\mathbf{Z} = \frac{\alpha}{\mathbf{I} - (1 - \alpha)(\mathbf{I} - \tilde{\mathbf{L}})} \mathbf{X} = \mathbf{U} \frac{\alpha}{\alpha + (1 - \alpha)\mathbf{\Lambda}} \mathbf{U}^\top \mathbf{X}, \tag{74}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-1} \mathbf{A}$ is random-walk normalized adjacency matrix with self-loop. Equation (74) is with a rational function whose numerator is a constant.

C.3 ARMA Filter

Substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$, Equation (44) can be rewritten as:

$$\mathbf{Z} = \frac{b}{\mathbf{I} - a(\mathbf{I} - \tilde{\mathbf{L}})} \mathbf{X} = \mathbf{U} \frac{b}{(1 - a) + a\mathbf{\Lambda}} \mathbf{U}^\top \mathbf{X} . \tag{75}$$

Note that ARMA filter is an unnormalized version of PPNP. When a + b = 1, ARMA filter becomes PPNP. Therefore, ARMA filter is more generalized than PPNP due to its unnormalization.

C.4 ParWalks

Upon decomposing the graph Laplacian, ParWalks can be expressed as follows:

$$\mathbf{Z} = \mathbf{U} \frac{\beta}{\beta + \mathbf{\Lambda}} \mathbf{X} \mathbf{U}^\top, \tag{76}$$

when setting $\beta = \frac{\alpha}{1-\alpha}$, it becomes PPNP:

$$\mathbf{Z} = \mathbf{U} \frac{\frac{\alpha}{1-\alpha}}{\frac{\alpha}{1-\alpha} + \mathbf{\Lambda}} \mathbf{X}\mathbf{U}^\intercal = \mathbf{U} \frac{\alpha}{\alpha + (1-\alpha)\mathbf{\Lambda}} \mathbf{X}\mathbf{U}^\intercal . \tag{77}$$

C.5 RationalNet

Substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$, Equation (39) can be transformed to Equation (53). The frequency response function is a generalized rational function.

REFERENCES

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 779–788.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 91–99.

[4] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Sig. Process. Mag.* 29 (2012).

[5] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).

[6] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 1412–1421.

[7] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *Proceedings of the International Conference on Learning Representations (ICLR'14)*. Retrieved from: http–openreview

[8] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations* (2017).

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 3844–3852.

[10] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 1024–1034.

[11] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 1993–2001.

[12] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

[13] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M. Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5115–5124.

[14] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[15] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2018. Deep learning on graphs: A survey. *arXiv preprint arXiv:1812.04202* (2018).

[16] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434* (2018).

[17] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596* (2019).

[18] Ferran Alet, Adarsh Keshav Jeewajee, Maria Bauza Villalonga, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Kaelbling. 2019. Graph element networks: Adaptive, structured computation and memory. In *Proceedings of the International Conference on Machine Learning*. PMLR, 212–222.

[19] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. 2018. Neural relational inference for interacting systems. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2688–2697.
[20] Hanjun Dai, Chengtao Li, Connor Coley, Bo Dai, and Le Song. 2019. Retrosynthesis prediction with conditional graph logic network. *Proceedings of the International Conference on Advances in Neural Information Processing Systems*.
[21] John Bradshaw, Matt J. Kusner, Brooks Paige, Marwin H. S. Segler, and José Miguel Hernández-Lobato. 2018. A generative model for electron paths. *arXiv preprint arXiv:1805.10970* (2018).
[22] Siddhant Arora. 2020. A survey on graph neural networks for knowledge graph completion. *arXiv preprint arXiv:2007.12374* (2020).
[23] Zi Ye, Yogan Jaya Kumar, Goh Ong Sing, Fengyan Song, and Junsong Wang. 2022. A comprehensive survey of graph neural networks for knowledge graphs. *IEEE Access* 10 (2022), 75729–75741.
[24] Yuyu Zhang, Xinshi Chen, Yuan Yang, Arun Ramamurthy, Bo Li, Yuan Qi, and Le Song. 2020. Efficient probabilistic logic reasoning with graph neural networks. *arXiv preprint arXiv:2001.11850* (2020).
[25] Xiaoran Xu, Wei Feng, Yunsheng Jiang, Xiaohui Xie, Zhiqing Sun, and Zhi-Hong Deng. 2019. Dynamically pruned message passing networks for large-scale knowledge graph reasoning. *arXiv preprint arXiv:1909.11334* (2019).
[26] Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. 2020. Graph convolutional networks with Markov random field reasoning for social spammer detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1054–1061.
[27] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 27–34.
[28] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *Proceedings of the World Wide Web Conference*. 417–426.
[29] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z. Sheng, Mehmet A. Orgun, Longbing Cao, Francesco Ricci, and Philip S. Yu. 2021. Graph learning based recommender systems: A review. *arXiv preprint arXiv:2105.06339* (2021).
[30] Licheng Jiao, Jie Chen, Fang Liu, Shuyuan Yang, Chao You, Xu Liu, Lingling Li, and Biao Hou. 2022. Graph representation learning meets computer vision: A survey. *IEEE Trans. Artif. Intell.* 4, 1 (2022), 2–22.
[31] Junyu Gao, Tianzhu Zhang, and Changsheng Xu. 2019. I know the relationships: Zero-shot action recognition via two-stream graph convolutional networks and knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 8303–8311.
[32] Ruiyu Li, Makarand Tapaswi, Renjie Liao, Jiaya Jia, Raquel Urtasun, and Sanja Fidler. 2017. Situation recognition with graph neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*. 4173–4182.
[33] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. 2021. Graph neural networks for natural language processing: A survey. *arXiv preprint arXiv:2106.06090* (2021).
[34] Daniil Sorokin and Iryna Gurevych. 2018. Modeling semantics with gated graph neural networks for knowledge base question answering. *arXiv preprint arXiv:1808.04126* (2018).
[35] Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. *arXiv preprint arXiv:1806.09835* (2018).
[36] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. 2019. Approximation ratios of graph neural networks for combinatorial problems. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*.
[37] Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. 2021. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544* (2021).
[38] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. 2019. Exact combinatorial optimization with graph convolutional neural networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*.
[39] Weiwei Jiang and Jiayun Luo. 2022. Graph neural network for traffic forecasting: A survey. *Expert Syst. Applic.* 207 (2022), 117921.
[40] Khac-Hoai Nam Bui, Jiho Cho, and Hongsuk Yi. 2022. Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues. *Appl. Intell.* 52, 3 (2022), 2763–2774.
[41] Jiexia Ye, Juanjuan Zhao, Kejiang Ye, and Chengzhong Xu. 2020. How to build a graph-based deep learning architecture in traffic domain: A survey. *IEEE Trans. Intell. Transport. Syst.* 23, 5 (2020), 3904–3924.
[42] Fanglan Chen, Zhiqian Chen, Subhodip Biswas, Shuo Lei, Naren Ramakrishnan, and Chang-Tien Lu. 2020. Graph convolutional networks with Kalman filtering for traffic prediction. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*. 135–138.
[43] Elizabeth Dinella, Hanjun Dai, Ziyang Li, Mayur Naik, Le Song, and Ke Wang. 2020. Hoppity: Learning graph transformations to detect and fix bugs in programs. In *Proceedings of the International Conference on Learning Representations (ICLR'20)*.

[44] Jiayi Wei, Maruth Goyal, Greg Durrett, and Isil Dillig. 2020. LambdaNet: Probabilistic type inference using graph neural networks. *arXiv preprint arXiv:2005.02161* (2020).
[45] Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. 2019. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*.
[46] Tian Bian, Xi Xiao, Tingyang Xu, Peilin Zhao, Wenbing Huang, Yu Rong, and Junzhou Huang. 2020. Rumor detection on social media with bi-directional graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 549–556.
[47] Hao Peng, Jianxin Li, Qiran Gong, Yangqiu Song, Yuanxing Ning, Kunfeng Lai, and Philip S. Yu. 2019. Fine-grained event categorization with heterogeneous graph convolutional networks. *arXiv preprint arXiv:1906.04580* (2019).
[48] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 701–710.
[49] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 855–864.
[50] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. 2020. Graph random neural network for semi-supervised learning on graphs. *arXiv preprint arXiv:2005.11079* (2020).
[51] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate PageRank. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2464–2473.
[52] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2020. Adaptive universal generalized PageRank graph neural network. In *Proceedings of the International Conference on Learning Representations*.
[53] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized PageRank. In *International Conference on Learning Representations*.
[54] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Combining neural networks with personalized PageRank for classification on graphs. In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=H1gL-2A9Ym
[55] Boris Knyazev, Graham W. Taylor, and Mohamed Amer. 2019. Understanding attention and generalization in graph neural networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 4202–4212.
[56] Dongkwan Kim and Alice Oh. 2020. How to find your friendly neighborhood: Graph attention design with self-supervision. In *Proceedings of the International Conference on Learning Representations*.
[57] Hoang N. T. and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).
[58] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.
[59] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*. JMLR. org, 1263–1272.
[60] Floris Geerts, Filip Mazowiecki, and Guillermo Perez. 2021. Let's agree to degree: Comparing graph convolutional networks in the message-passing framework. In *Proceedings of the International Conference on Machine Learning*. PMLR, 3640–3649.
[61] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=ryGs6iA5Km
[62] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNN explainer: A tool for post-hoc explanation of graph neural networks. *CoRR*. arXiv preprint arXiv:1903.03894 (2019).
[63] Zhiqian Chen, Fanglan Chen, Lei Zhang, Taoran Ji, Kaiqun Fu, Liang Zhao, Feng Chen, and Chang-Tien Lu. 2020. Bridging the gap between spatial and spectral domains: A survey on graph neural networks. *arXiv preprint arXiv:2002.11867* (2020).
[64] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Appl. Comput. Harmon. Anal.* 30, 2 (2011), 129–150.
[65] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *Proceedings of the International Conference on Learning Representations*.
[66] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: Going beyond Euclidean data. *IEEE Sig. Process. Mag.* 34, 4 (2017), 18–42.
[67] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and unifying graph neural networks with an optimization framework. *arXiv preprint arXiv:2101.11859* (2021).

[68] John Boaz Lee, Ryan A. Rossi, Sungchul Kim, Nesreen K. Ahmed, and Eunyee Koh. 2019. Attention models in graphs: A survey. *ACM Trans. Knowl. Discov. Data* 13, 6, Article 62 (Nov. 2019), 25 pages. DOI: http://dx.doi.org/10.1145/3363574

[69] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying DeepWalk, LINE, PTE, and node2vec. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 459–467.

[70] Xin Liu, Tsuyoshi Murata, Kyoung-Sook Kim, Chatchawan Kotarasu, and Chenyi Zhuang. 2019. A general view for network embedding as matrix factorization. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 375–383.

[71] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. 2020. Heterogeneous network representation learning: Survey, benchmark, evaluation, and beyond. *arXiv preprint arXiv:2004.00216* (2020).

[72] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. Gnnexplainer: Generating explanations for graph neural networks. *Advances in Neural Information Processing Systems* 32 (2019).

[73] Federico Baldassarre and Hossein Azizpour. 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686* (2019).

[74] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2020. Explainability in graph neural networks: A taxonomic survey. *arXiv preprint arXiv:2012.15445* (2020).

[75] Wanyu Lin, Hao Lan, and Baochun Li. 2021. Generative causal explanations for graph neural networks. In *Proceedings of the International Conference on Machine Learning*.

[76] Béla Bollobás. 2004. *Extremal Graph Theory*. Courier Corporation.

[77] Yewen Wang, Ziniu Hu, Yusong Ye, and Yizhou Sun. 2020. Demystifying Graph Neural Network Via Graph Filter Assessment. Retrieved from https://openreview.net/forum?id=r1erNxBtwr

[78] David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Sig. Process. Mag.* 30, 3 (2013), 83–98.

[79] Xiaofan Zhu and Michael Rabbat. 2012. Approximating signals supported on graphs. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'12)*. IEEE, 3921–3924.

[80] Alan V. Oppenheim, John R. Buck, and Ronald W. Schafer. 2001. *Discrete-time Signal Processing. Vol. 2*. Prentice Hall, Upper Saddle River, NJ.

[81] Lars V. Ahlfors. 1981. *Complex Analysis: An Introduction to Theory of Analytic Functions of One Complex Variable*. McGraw Hill International Book Company. https://www.google.com/books/edition/Complex_Analysis_an_Introduction_to_Theo/HGPrzAEACAAJ?hl=en

[82] Lloyd N. Trefethen. 2013. *Approximation Theory and Approximation Practice*. Vol. 128. Siam.

[83] Ricardo Pachon. 2010. *Algorithms for Polynomial and Rational Approximation*. Ph.D. Dissertation. University of Oxford.

[84] Michael James David Powell. 1981. *Approximation Theory and Methods*. Cambridge University Press.

[85] Harold Cohen. 2011. *Numerical Approximation Methods*. Springer.

[86] Penco Petrov Petrushev and Vasil Atanasov Popov. 2011. *Rational Approximation of Real Functions*. Vol. 28. Cambridge University Press.

[87] Naum I. Achieser. 2013. *Theory of Approximation*. Courier Corporation.

[88] Eric Ziegel. 1987. Numerical recipes: The art of scientific computing. *Technometrics* 29, 4 (November 1, 1987), 501–502. https://doi.org/10.1080/00401706.1987.10488304

[89] John P. Boyd. 2001. *Chebyshev and Fourier Spectral Methods*. Courier Corporation.

[90] John C. Mason and David C. Handscomb. 2002. *Chebyshev Polynomials*. CRC Press.

[91] Eugene Y. Remez. 1934. Sur la détermination des polynômes d'approximation de degré donnée. *Comm. Soc. Math. Kharkov* 10 (1934), 41–63.

[92] Xiao-Ming Wu, Zhenguo Li, Anthony M. So, John Wright, and Shih-Fu Chang. 2012. Learning with partially absorbing random walks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 3077–3085.

[93] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*.

[94] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).

[95] David K. Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P. Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'15)*. Curran Associates, Inc., 2224–2232.

[96] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: Moving beyond fingerprints. *J. Comput.-aid. Molec. Des.* 30, 8 (2016), 595–608.

[97] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *Proceedings of the 33rd International Conference on Machine Learning (ICML'16)*. 2014–2023.

[98] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein interface prediction using graph convolutional networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'17)*. Curran Associates, Inc., 6530–6539.

[99] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive sampling towards fast graph representation learning. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 4558–4567.

[100] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast learning with graph convolutional networks via importance sampling. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*.

[101] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536* (2018).

[102] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'18)*. 1416–1424.

[103] Zhiqian Chen, Feng Chen, Rongjie Lai, Xuchao Zhang, and Chang-Tien Lu. 2018. Rational neural networks for approximating graph convolution operator on jump discontinuities. In *2018 IEEE International Conference on Data Mining (ICDM)*, IEEE, 59–68.

[104] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

[105] Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. 2018. CayleyNets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Trans. Sig. Process.* 67, 1 (2018), 97–109.

[106] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *Proceedings of the International Conference on Machine Learning*. 6861–6871.

[107] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs go as deep as CNNs? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9267–9276.

[108] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. *arXiv preprint arXiv:1905.00067* (2019).

[109] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2021. Graph neural networks with convolutional ARMA filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 7 (2021), 3496–3507.

[110] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 13354–13366.

[111] Yao Ma, Suhang Wang, Charu C. Aggarwal, and Jiliang Tang. 2019. Graph convolutional networks with EigenPooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19)*. 723–731.

[112] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. 2019. Graph wavelet neural network. In *Proceedings of the International Conference on Learning Representations (ICLR'19)*.

[113] Saurabh Verma and Zhi-Li Zhang. 2019. Stability and generalization of graph convolutional neural networks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19)*. 1539–1548.

[114] Emanuele Rossi, Fabrizio Frasca, Ben Chamberlain, Davide Eynard, Michael Bronstein, and Federico Monti. 2020. SIGN: Scalable inception graph neural networks. *arXiv preprint arXiv:2004.11198* (2020).

[115] Shichao Zhu, Lewei Zhou, Shirui Pan, Chuan Zhou, Guiying Yan, and Bin Wang. 2020. GSSNN: Graph smoothing splines neural network. In *Proceedings of the 34th Conference on Association for the Advancement of Artificial Intelligence (AAAI'20)*. 7007–7014.

[116] Zhao-Yang Liu, Shao-Yuan Li, Songcan Chen Yao Hu, and Sheng-Jun Huang. 2020. Uncertainty aware graph Gaussian process for semi-supervised learning. In *Proceedings of the 34th Conference on Association for the Advancement of Artificial Intelligence (AAAI'20)*. 4957–4964.

[117] Yilun Jin, Guojie Song, and Chuan Shi. 2020. GraLSP: Graph neural networks with local structural patterns. In *Proceedings of the 34th Conference on Association for the Advancement of Artificial Intelligence (AAAI'20)*. 4361–4368.

[118] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. GraphSaint: Graph sampling based inductive learning method. In *Proceedings of the International Conference on Learning Representations*.

[119] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. DropEdge: Towards deep graph convolutional networks on node classification. In *Proceedings of the International Conference on Learning Representations*.

[120] Hongmin Zhu, Fuli Feng, Xiangnan He, Xiang Wang, Yan Li, Kai Zheng, and Yongdong Zhang. 2020. Bilinear graph neural network with neighbor interactions. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI'20)*. 1452–1458.

[121] Yiqing Xie, Sha Li, Carl Yang, Raymond Chi-Wing Wong, and Jiawei Han. 2020. When do GNNs Work: Understanding and improving neighborhood aggregation. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI'20)*. 1303–1309.

[122] Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. 2020. Continuous graph neural networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. 10432–10441.

[123] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, PMLR, 1725–1735.

[124] Aleksandar Bojchevski, Johannes Klicpera, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. 2020. Scaling graph neural networks with approximate PageRank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'20)*. 2464–2473.

[125] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 338–348.

[126] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*. Curran Associates, Inc., 7793–7804.

[127] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. 2020. GraphZoom: A multi-level spectral approach for accurate and scalable graph embedding. In *Proceedings of the International Conference on Learning Representations (ICLR'20)*.

[128] Jialin Zhao, Yuxiao Dong, Ming Ding, Evgeny Kharlamov, and Jie Tang. 2021. Adaptive diffusion in graph neural networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'21)*. Curran Associates, Inc., 23321–23333.

[129] Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. 2021. Universal graph convolutional networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'21)*. Curran Associates, Inc., 10654–10664.

[130] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. 2021. Dissecting the diffusion process in linear graph convolutional networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'21)*. Curran Associates, Inc., 5758–5769.

[131] Víctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. 2021. E(n) equivariant graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*. 9323–9332.

[132] Benjamin Paul Chamberlain, James Rowbottom, Maria Gorinova, Stefan Webb, Emanuele Rossi, and Michael M. Bronstein. 2021. GRAND: Graph neural diffusion. *arXiv preprint arXiv:2106.10934* (2021).

[133] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin Benson. 2021. Combining label propagation and simple models out-performs graph neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR'21)*.

[134] Zemin Liu, Yuan Fang, Chenghao Liu, and Steven C. H. Hoi. 2021. Node-wise localization of graph neural networks. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21)*. 1520–1526.

[135] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. 2021. Interpretable stability bounds for spectral graph filters. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*. 5388–5397.

[136] Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. 2021. Analyzing the expressive power of graph neural networks in a spectral perspective. In *Proceedings of the International Conference on Learning Representations (ICLR'21)*.

[137] Hao Zhu and Piotr Koniusz. 2021. Simple spectral graph convolution. In *Proceedings of the International Conference on Learning Representations (ICLR'21)*.

[138] Mingguo He, Zhewei Wei, Zengfeng Huang, and Hongteng Xu. 2021. BernNet: Learning arbitrary graph spectral filters via Bernstein approximation. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 14239–14251.

[139] Daniele Grattarola and Pierre Vandergheynst. 2022. Generalised implicit neural representations. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'22)*. Curran Associates, Inc.

[140] Sudhanshu Chanpuriya and Cameron Musco. 2022. Simplified graph convolution with heterophily. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'22)*. Curran Associates, Inc.

[141] Zhongyu Huang, Yingheng Wang, Chaozhuo Li, and Huiguang He. 2022. Going deeper into permutation-sensitive graph neural networks. In *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*. 9377–9409.
[142] Liang Yang, Chuan Wang, Xiaochun Cao, Junhua Gu, Bingxin Niu, Yuanfang Guo, Weixun Li, Cheng Chen, and Dongxiao He. 2022. Self-supervised graph neural networks via diverse and interactive message passing. In *Proceedings of the 36th Conference on Association for the Advancement of Artificial Intelligence (AAAI'22)*. 4327–4336.
[143] Guangyu Meng, Qisheng Jiang, Kaiqun Fu, Beiyu Lin, Chang-Tien Lu, and Zhqian Chen. 2022. Early forecasting of the impact of traffic accidents using a single shot observation. In *Proceedings of the SIAM International Conference on Data Mining (SDM'22)*. SIAM, 100–108.
[144] Mingguo He, Zhewei Wei, and Ji-Rong Wen. 2022. Convolutional neural networks on graphs with Chebyshev approximation, revisited. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'22)*. Curran Associates, Inc.
[145] Xiyuan Wang and Muhan Zhang. 2022. How powerful are spectral graph neural networks. In *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*. 23341–23362.
[146] Mingqi Yang, Yanming Shen, Rui Li, Heng Qi, Qiang Zhang, and Baocai Yin. 2022. A new perspective on the effects of spectrum in graph neural networks. In *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*. 25261–25279.
[147] Mingjie Li, Xiaojun Guo, Yifei Wang, Yisen Wang, and Zhouchen Lin. 2022. G^2CN: Graph Gaussian convolution networks with concentrated graph filters. In *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*. 12782–12796.
[148] Guoji Fu, Peilin Zhao, and Yatao Bian. 2022. p-Laplacian based graph neural networks. In *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*. 6878–6917.
[149] Jie Zhang, Bo Hui, Po-wei Harn, Min-Te Sun, and Wei-Shinn Ku. 2023. From ChebNet to ChebGibbsNet. https://openreview.net/forum?id=2a5Ru3JtNe0
[150] Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. 2023. *Specformer: Spectral Graph Neural Networks Meet Transformers*. arXiv preprint arXiv:2303.01028 (2023).
[151] Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. 2022. Sign and basis invariant networks for spectral graph representation learning. arXiv preprint arXiv:2202.13013 (2022).
[152] Yongyu Wang, Zhiqiang Zhao, and Zhuo Feng. 2022. Scalable graph topology learning via spectral densification. In *Proceedings of the 15th ACM International Conference on Web Search and Data Mining (WSDM'22)*. Association for Computing Machinery, New York, NY, 1099–1108. DOI: http://dx.doi.org/10.1145/3488560.3498480
[153] Zeyu Zhang, Jiamou Liu, Xianda Zheng, Yifei Wang, Pengqian Han, Yupan Wang, Kaiqi Zhao, and Zijian Zhang. 2023. RSGNN: A model-agnostic approach for enhancing the robustness of signed graph neural networks. In *Proceedings of the ACM Web Conference*. 60–70.
[154] Yu Wang, Yuying Zhao, Yi Zhang, and Tyler Derr. 2023. Collaboration-aware graph convolutional network for recommender systems. In *Proceedings of the ACM Web Conference*. 91–101.
[155] Liang Yang, Qiuliang Zhang, Runjie Shi, Wenmiao Zhou, Bingxin Niu, Chuan Wang, Xiaochun Cao, Dongxiao He, Zhen Wang, and Yuanfang Guo. 2023. Graph neural networks without propagation. In *Proceedings of the ACM Web Conference*. 469–477.
[156] Xin Zheng, Miao Zhang, Chunyang Chen, Qin Zhang, Chuan Zhou, and Shirui Pan. 2023. Auto-HeG: Automated graph neural network on heterophilic graphs. *arXiv preprint arXiv:2302.12357* (2023).
[157] Taoran Fang, Zhiqing Xiao, Chunping Wang, Jiarong Xu, Xuan Yang, and Yang Yang. 2022. DropMessage: Unifying random dropping for graph neural networks. *arXiv preprint arXiv:2204.10037* (2022).
[158] Jingwei Guo, Kaizhu Huang, Xinping Yi, and Rui Zhang. 2023. Graph neural networks with diverse spectral filtering. In *Proceedings of the ACM Web Conference*. 306–316.
[159] Dongqi Fu, Dawei Zhou, Ross Maciejewski, Arie Croitoru, Marcus Boyd, and Jingrui He. 2023. Fairness-aware clique-preserving spectral clustering of temporal graphs. In *Proceedings of the ACM Web Conference*. 3755–3765.
[160] Jincheng Huang, Lun Du, Xu Chen, Qiang Fu, Shi Han, and Dongmei Zhang. 2023. Robust mid-pass filtering graph convolutional networks. *arXiv preprint arXiv:2302.08048* (2023).
[161] Yuan Gao, Xiang Wang, Xiangnan He, Zhenguang Liu, Huamin Feng, and Yongdong Zhang. 2023. Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. In *Proceedings of the ACM Web Conference*. 1528–1538.
[162] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
[163] Rie Johnson and Tong Zhang. 2007. On the effectiveness of Laplacian normalization for graph semi-supervised learning. *J. Mach. Learn. Res.* 8, July (2007), 1489–1517.
[164] Joan Bruna Estrach, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR'14)*.

[165] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2181–2187.
[166] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1225–1234.
[167] Matus Telgarsky. 2017. Neural networks and rational functions. In *Proceedings of the International Conference on Machine Learning*. PMLR, 3387–3393.
[168] Nicolas Boullé, Yuji Nakatsukasa, and Alex Townsend. 2020. Rational neural networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 14243–14253.
[169] Martin Trimmel, Mihai Zanfir, Richard Hartley, and Cristian Sminchisescu. 2022. ERA: Enhanced rational activations. In *Proceedings of the European Conference on Computer Vision*. Springer, 722–738.
[170] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. 2019. Label efficient semi-supervised learning via graph filtering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'19)*.
[171] A. Loukas, A. Simonetto, and G. Leus. 2015. Distributed autoregressive moving average graph filters. *IEEE Sig. Process. Lett.* 22, 11 (Nov. 2015), 1931–1935. DOI: http://dx.doi.org/10.1109/LSP.2015.2448655
[172] E. Isufi, A. Loukas, A. Simonetto, and G. Leus. 2017. Autoregressive moving average graph filtering. *IEEE Trans. Sig. Process.* 65, 2 (Jan. 2017), 274–288. DOI: http://dx.doi.org/10.1109/TSP.2016.2614793
[173] Lingxiao Zhao and Leman Akoglu. 2019. PairNorm: Tackling oversmoothing in GNNs. In *Proceedings of the International Conference on Learning Representations*.
[174] Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R. Benson. 2020. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993* (2020).
[175] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the 20th International Conference on Machine Learning*. 912–919.
[176] Dengyong Zhou, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 321–328.
[177] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. 11 label propagation and quadratic criterion. located at: https://www.researchgate.net/profile/Y_Bengio/publication/238675708_Label_Propagation_and_Quadratic_Criterion/links/0f3175320aae4ada34000000/Label-Propagation-and-Quadratic-Criterion.pdf
[178] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.
[179] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 770–778.
[180] Mark Herbster, Massimiliano Pontil, and Lisa Wainer. 2005. Online learning over graphs. In *Proceedings of the 22nd International Conference on Machine Learning*. ACM, 305–312.
[181] Andreas Argyriou, Charles A. Micchelli, and Massimiliano Pontil. 2009. When is there a representer theorem? Vector versus matrix regularizers. *J. Mach. Learn. Res.* 10, Nov. (2009), 2507–2529.
[182] B. Schölkopf, R. Herbrich, and A. J. Smola. 2001. A generalized representer theorem. In *International Conference on Computational Learning Theory*, Springer Berlin Heidelberg, Berlin, Heidelberg, 416–426.
[183] Fan R. K. Chung. 1997. *Spectral Graph Theory*. Number 92. American Mathematical Society.
[184] Sergi Abadal, Akshay Jain, Robert Guirado, Jorge López-Alonso, and Eduard Alarcón. 2020. Computing Graph Neural Networks: A Survey from Algorithms to Accelerators. *arXiv:cs.LG/2010.00130*
[185] Siemon C. de Lange, Martijn P. van den Heuvel, and Marcel A. de Reus. 2016. The role of symmetry in neural networks and their Laplacian spectra. *NeuroImage* 141 (2016), 357–365.
[186] Siemon de Lange, Marcel de Reus, and Martijn Van Den Heuvel. 2014. The Laplacian spectrum of neural networks. *Front. Comput. Neurosci.* 7 (2014), 189.
[187] Gerald B. Folland and Alladi Sitaram. 1997. The uncertainty principle: A mathematical survey. *J. Fourier Anal. Applic.* 3, 3 (1997), 207–238.
[188] Ameya Agaskar and Yue M. Lu. 2013. A spectral graph uncertainty principle. *IEEE Trans. Inf. Theor.* 59, 7 (2013), 4338–4356.
[189] Petar Veličković, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. 2018. Deep graph infomax. In *Proceedings of the International Conference on Learning Representations*.
[190] Xiaomin Liang, Daifeng Li, and Andrew Madden. 2020. Attributed network embedding based on mutual information estimation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 835–844.
[191] Chenyi Zhuang and Qiang Ma. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the World Wide Web Conference*. 499–508.

[192] Songyang Zhang, Xuming He, and Shipeng Yan. 2019. LatentGNN: Learning efficient non-local relations for visual recognition. In *Proceedings of the International Conference on Machine Learning*. PMLR, 7374–7383.

[193] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*.

[194] Haiyang Lin, Mingyu Yan, Xiaochun Ye, Dongrui Fan, Shirui Pan, Wenguang Chen, and Yuan Xie. 2022. A comprehensive survey on distributed training of graph neural networks. *arXiv preprint arXiv:2211.05368* (2022).

[195] Yingxia Shao, Hongzheng Li, Xizhi Gu, Hongbo Yin, Yawen Li, Xupeng Miao, Wentao Zhang, Bin Cui, and Lei Chen. 2022. Distributed graph neural network training: A survey. *arXiv preprint arXiv:2211.00216* (2022).

[196] Uri Shaham, Kelly Stanton, Henry Li, Boaz Nadler, Ronen Basri, and Yuval Kluger. 2018. SpectralNet: Spectral clustering using deep neural networks. *arXiv preprint arXiv:1801.01587* (2018).

[197] Joost Verbraeken, Matthijs Wolting, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S. Rellermeyer. 2020. A survey on distributed machine learning. *ACM Comput. Surv.* 53, 2 (2020), 1–33.

[198] Zhan Gao, Fernando Gama, and Alejandro Ribeiro. 2022. Wide and deep graph neural network with distributed online learning. *IEEE Trans. Sig. Process.* 70 (2022), 3862–3877.

[199] Honglei He, Yuxuan He, Fang Wang, and Wenming Zhu. 2022. Improved K-means algorithm for clustering non-spherical data. *Expert Syst.* 39, 9 (2022), e13062.

[200] Martin Azizyan, Aarti Singh, and Larry Wasserman. 2015. Efficient sparse clustering of high-dimensional non-spherical Gaussian mixtures. In *Artificial Intelligence and Statistics*. PMLR, 37–45.

[201] Yan Shi, Jun-Xiong Cai, Yoli Shavit, Tai-Jiang Mu, Wensen Feng, and Kai Zhang. 2022. ClusterGNN: Cluster-based coarse-to-fine graph neural network for efficient feature matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12517–12526.

[202] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2022. Revisiting heterophily for graph neural networks. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'22)*. Curran Associates, Inc.

[203] Liang Yang, Mengzhe Li, Liyang Liu, Bingxin Niu, Chuan Wang, Xiaochun Cao, and Yuanfang Guo. 2021. Diverse message passing for attribute with heterophily. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems (NeurIPS'21)*. Curran Associates, Inc., 4751–4763.

[204] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. 2022. Finding global homophily in graph neural networks when meeting heterophily. In *Proceedings of the 39th International Conference on Machine Learning (ICML'22)*. 13242–13256.

[205] Jianfei Chen, Jun Zhu, and Le Song. 2018. Stochastic training of graph convolutional networks with variance reduction. In *Proceedings of the International Conference on Machine Learning*. 942–950.

[206] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1416–1424.

[207] Kai-Lang Yao and Wu-Jun Li. 2021. Blocking-based neighbor sampling for large-scale graph neural networks. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI'21)*. 3307–3313.

[208] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Accurate, efficient and scalable graph embedding. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS'19)*. IEEE, 462–471.

[209] Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. 2019. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

[210] Tomáš Mikolov, Wen-Tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the Annual Conference of the North American Chapter of the ACL (NAACL'13)*. 746–751.

[211] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the Workshop at the International Conference on Learning Representations*.

[212] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 3111–3119.

[213] Tianmeng Yang, Yujing Wang, Zhihan Yue, Yaming Yang, Yunhai Tong, and Jing Bai. 2022. Graph pointer neural networks. In *Proceedings of the 36th Conference of the Association for the Advancement of Artificial Intelligence (AAAI'22)*. 8832–8839.

[214] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*. 2177–2185.

[215] Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. 2020. Revisiting "over-smoothing" in deep GCNs. *arXiv preprint arXiv:2003.13663* (2020).

[216] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 5 (2021), 3950–3957.
[217] Kenta Oono and Taiji Suzuki. 2019. Graph neural networks exponentially lose expressive power for node classification. In *Proceedings of the International Conference on Learning Representations*.
[218] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. 2020. DeeperGCN: All you need to train deeper GCNs. *arXiv preprint arXiv:2006.07739* (2020).
[219] Wenbing Huang, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. 2020. Tackling Over-smoothing for General Graph Convolutional Networks. *arXiv:cs.LG/2008.09864*
[220] Kaixiong Zhou, Xiao Huang, Yuening Li, Daochen Zha, Rui Chen, and Xia Hu. 2020. Towards Deeper Graph Neural Networks with Differentiable Group Normalization. *arXiv:cs.LG/2006.06972*
[221] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. 2020. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3438–3445.
[222] Yimeng Min, Frederik Wenkel, and Guy Wolf. 2020. Scattering GCN: Overcoming oversmoothness in graph convolutional networks. *arXiv preprint arXiv:2003.08414* (2020).
[223] Antonio G. Marques, Santiago Segarra, and Gonzalo Mateos. 2020. Signal processing on directed graphs: The role of edge directionality when processing and learning from network data. *IEEE Sig. Process. Mag.* 37, 6 (2020), 99–116.
[224] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Trans. Neural Netw. Learn. Syst.* 33, 2 (Feb. 2022), 494–514. DOI: http://dx.doi.org/10.1109/TNNLS.2021.3070843
[225] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. *ICML*, 11, 10.5555 (2011), 3104482–3104584.
[226] Bishan Yang, Wen-Tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).
[227] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical inference for multi-relational embeddings. In *Proceedings of the International Conference on Machine Learning*. PMLR, 2168–2178.
[228] Nelson Dunford and Jacob T. Schwartz. 1988. *Linear Operators, Part 1: General Theory*. Vol. 10. John Wiley & Sons.
[229] Canran Xu and Ruijiang Li. 2019. Relation embedding with dihedral group in knowledge graph. *arXiv preprint arXiv:1906.00687* (2019).
[230] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-end structure-aware convolutional networks for knowledge base completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3060–3067.
[231] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the International Conference on Advances in Neural Information Processing Systems*.
[232] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *Proceedings of the 15th International Conference on the Semantic Web (ESWC'18)*. Springer, 593–607.
[233] Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. 2019. Composition-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1911.03082* (2019).
[234] Deepak Nathani, Jatin Chauhan, Charu Sharma, and Manohar Kaul. 2019. Learning attention-based embeddings for relation prediction in knowledge graphs. *arXiv preprint arXiv:1906.01195* (2019).
[235] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. 2021. Interpretable stability bounds for spectral graph filters. In *Proceedings of the International Conference on Machine Learning*. PMLR, 5388–5397.
[236] Walter A. Strauss. 2007. *Partial Differential Equations: An Introduction*. John Wiley & Sons.
[237] Bo Kågström and Axel Ruhe. 1980. An algorithm for numerical computation of the Jordan normal form of a complex matrix. *ACM Trans. Math. Softw.* 6, 3 (1980), 398–419.
[238] Ron Levie, Wei Huang, Lorenzo Bucci, Michael M. Bronstein, and Gitta Kutyniok. 2021. Transferability of spectral graph convolutional neural networks. *J. Mach. Learn. Res.* 22 (2021), 272–1.
[239] Muhammad Zubair Malik and Sarfraz Khurshid. 2012. Dynamic shape analysis using spectral graph properties. In *Proceedings of the IEEE 5th International Conference on Software Testing, Verification and Validation*. 211–220. DOI: http://dx.doi.org/10.1109/ICST.2012.101
[240] Arlei Silva, Ambuj Singh, and Ananthram Swami. 2018. Spectral algorithms for temporal graph cuts. In *Proceedings of the World Wide Web Conference (WWW'18)*. International World Wide Web Conferences Steering Committee, 519–528. DOI: http://dx.doi.org/10.1145/3178876.3186118
[241] Jared C. Bronski and Lee DeVille. 2014. Spectral theory for dynamics on graphs containing attractive and repulsive interactions. *SIAM J. Appl. Math.* 74, 1 (2014), 83–105.

[242] Defu Cao, Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. 2021. Spectral temporal graph neural network for trajectory prediction. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'21)*. IEEE, 1839–1845.
[243] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2022. Explainability in graph neural networks: A taxonomic survey. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 5 (2022), 5782–5799.
[244] Song Bai, Feihu Zhang, and Philip H. S. Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recog.* 110 (2021), 107637.
[245] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 3558–3565.
[246] Jaehyuk Yi and Jinkyoo Park. 2020. Hypergraph convolutional recurrent neural network. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 3366–3376.
[247] Emily Ribando-Gros, Rui Wang, Jiahui Chen, Yiying Tong, and Guo-Wei Wei. 2022. Graph and Hodge Laplacians: Similarity and difference. *arXiv preprint arXiv:2204.12218* (2022).
[248] Lek-Heng Lim. 2020. Hodge Laplacians on graphs. *Siam Rev.* 62, 3 (2020), 685–715.
[249] Ginestra Bianconi. 2021. *Higher-order Networks*. Cambridge University Press.
[250] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. 2018. Simplicial closure and higher-order link prediction. *Proc. Nat. Acad. Sci.* 115, 48 (2018), E11221–E11230.
[251] Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. 2020. Networks beyond pairwise interactions: Structure and dynamics. *Phys. Rep.* 874 (2020), 1–92.
[252] Manlio De Domenico, Clara Granell, Mason A. Porter, and Alex Arenas. 2016. The physics of spreading processes in multilayer networks. *Nat. Phys.* 12, 10 (2016), 901–906.
[253] Ginestra Bianconi. 2018. *Multilayer Networks: Structure and Function*. Oxford University Press.
[254] Alberto Aleta and Yamir Moreno. 2018. Multilayer networks in a nutshell. *arXiv preprint arXiv:1804.03488* (2018).
[255] Gaogao Dong, Dongli Duan, and Yongxiang Xia. 2021. A briefing survey on advances of coupled networks with various patterns. *Front. Phys.* 9 (2021), 795279.
[256] Marco Grassia, Manlio De Domenico, and Giuseppe Mangioni. 2021. mGNN: Generalizing the graph neural networks to the multilayer case. *arXiv preprint arXiv:2109.10119* (2021).

Received 5 August 2021; revised 15 May 2023; accepted 18 September 2023