

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

ZHIQIAN CHEN, of Computer Science and Engineering, Mississippi State University, U.S.A

FANGLAN CHEN, Computer Science, Virginia Tech, U.S.A

LEI ZHANG, Computer Science, Virginia Tech, U.S.A

TAORAN JI, Computer Science, Virginia Tech, U.S.A

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, JD.COM Silicon Valley Research Center, 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.

1 INTRODUCTION

Deep learning's performance in various machine learning tasks [96, 120, 145, 168, 170, 203] 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 [119], transportation networks [21], spread of epidemic disease [156], brain's neuronal networks [148], gene data on biological regulatory networks [58], telecommunication networks [66], and knowledge graph [137]. Previous deep learning algorithms, such as convolutional and recurrent neural networks, couldn't 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 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 [14, 38, 62, 87, 114, 188] and grouped into coarse-grained groupings like the spectral [88, 155, 204, 230, 234] and spatial [14, 87, 188] domains. GNNs have seen rising popularity recently for learning graph representations and quickly spread out to many application domains, such as physics [7, 113], chemistry [35, 57], knowledge graph [13, 211, 220, 229], recommender systems [49, 70, 193, 202], computer vision [78, 104, 131], natural language processing [20, 182, 200], combinatorial optimization [41, 80, 174], traffic network [39, 45, 103, 219], program representation [64, 198, 236], social network [24, 163, 202]. 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 [71, 85, 164], PageRank [30, 54, 115, 116], attention models [112, 118, 188], low-pass filters [29, 158], and message forwarding [81, 82]. Some preliminary research can only explain a few GNNs methods [82, 209, 222], 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 [88, 155, 204, 230, 234].

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.

1.1 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 [62, 89], whereas spatial theory is used to propose others [46, 87]. 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 paper, 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 in order 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

¹A short version is available at [52]

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 can be divided into three groups: **Existing Works Group 1 (Comprehensive Collection)**: Recently, many extensive surveys on graph neural networks have been compiled [37, 88, 204, 230, 234, 241]. Instead of studying their hierarchical and underlying mechanisms, most existing surveys focus on gathering newly published works and categorizing them into separate categories. A detailed survey, in particular, provides an overview of many examples of graph neural networks, classifying them as spatial or spectral-based techniques [37]. A taxonomy of graph types, training methods, and propagation processes was recently published in [234]. Another survey [230] categorized graph neural network advances as semi-supervised (graph convolution), unsupervised (graph auto-encoder), and latest advancements (graph recurrent neural network and graph reinforcement learning). Graph convolution, graph auto-encoder, graph recurrent neural network, and spatial-temporal graph neural networks are all included in [204]. These existing surveys, on the other hand, fail to integrate their categories into a cohesive framework. **Existing Works Group 2 (Particular Perspectives)**: The second thread of surveys for graph neural networks is from diverse theoretical perspectives. For example, in the field of graph neural networks with an attention mechanism, a comprehensive and concentrated survey was undertaken [121]. Another example demonstrated how many graph neural networks with negative sampling might be merged into an analytical matrix factorization framework [167]. One similar study offered a general view proving that network embedding techniques and matrix factorization are equal in terms of two objectives: one for similar nodes and the other for distant nodes [139]. One specific survey created four benchmark datasets with diverse features and user-friendly interfaces for 10 common algorithms, providing a unified paradigm for systematic categorization and analysis on several existing heterogeneous network embeddings approaches [213]. A recent work analyzed anonymous and degree-aware message-passing to study the distinguishing power of different classes [81]. However, this research is limited to a subset of the GNNs family and lacks a global perspective. **Existing Works Group 3 (Post-Hoc Explanation)**: Building post-hoc models and then identifying the underlying patterns from a statistical standpoint is another technique to analyze GNNs [18, 122, 136, 224]. Because neural networks are employed without any theories or domain expertise, this methodology is referred to as "black box". For this reason, post-hoc models have the potential to be biased, subject to adversarial attacks, and difficult to verify. Our research focuses on interpretable graph neural networks, which have a strong theoretical foundation. Previous surveys

	Existing Works 1	Existing Works 2	Existing Works 3	This Survey
Theoretical Support		●		●
Comprehensiveness	●		●	●

Table 1. Comparing this study with previous studies

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) [114], 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

In this section, we outline basic concepts, necessary preliminary, and problem setup of 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 [32], 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

Table 2. 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).
$\mathbf{U}, \mathbf{U}^T$	Eigenvector matrix and its transpose.
$\mathbf{U}_i \in \mathbf{U}, \mathbf{u}^T_i \in \mathbf{U}^T$	Single eigenvector and its transpose.
$\mathbf{D}$	The degree matrix of $\mathbf{A}$ and $\mathbf{D}_{ii} = \sum_{j=1}^n \mathbf{A}_{ij}$.
$\mathbf{X} \in \mathbb{R}^{N \times d}$	The feature matrix of a graph.
$\mathbf{Z} \in \mathbb{R}^{N \times b}$	New node feature matrix.
$\mathbf{H} \in \mathbb{R}^{N \times b}$	The node hidden feature matrix.
$\mathbf{h}_v \in \mathbb{R}^b$	The hidden feature vector of node v .
N	node number
b	dimension size of hidden feature
$\odot$	Element-wise product.
Θ, θ	Learnable model parameters.
$\mathbf{P}(\cdot), \mathbf{Q}(\cdot)$	Polynomial function.
$\mathcal{N}(v)$	Directed neighbors of node v

a graph topology, and outputting a filtered feature as:

$$f : G, \mathbf{X} \rightarrow \mathbf{Z}, \quad (1)$$

where we aim to find a mapping which 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 3, and most popular are symmetric normalized graph matrices.

In this survey, we use the graph Laplacian, adjacency matrix, and their modifications described in Table 3 to represent a graph. So yet, no experimental or theoretical evidence has been shown that any filter has a consistent advantage [195]. 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:

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

Table 3. 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

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 [180, 244] as $\tilde{\mathbf{L}} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top$, where $\mathbf{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues (i.e., $\Lambda_{ii} = \lambda_i$), and $\mathbf{U}$ represents eigenvectors. Further, the graph Fourier transform of a signal $\mathbf{X}$ is defined as $\hat{\mathbf{X}} = \mathbf{U}^\top \mathbf{X} \in \mathbb{R}^{N \times N}$ 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) [161], the convolution is defined in the Fourier domain such that

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

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

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

$$\mathbf{g} * \mathbf{X} = \mathbf{U} [\mathbf{g}(\mathbf{\Lambda}) \odot (\mathbf{U}^\top \mathbf{X})] = \mathbf{U} \text{diag}(\mathbf{g}(\mathbf{\Lambda})) \mathbf{U}^\top \mathbf{X}, \quad (3)$$

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:

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

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 [4, 6, 34, 56, 149, 162, 165, 166, 169, 186, 246]. 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 $\mathbf{L} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^{-1} = \mathbf{U} \mathbf{\Lambda} \mathbf{U}^\top$.

Chebyshev approximation on spectral filter g is applied [62, 89] so that it can be approximated with a polynomial with order k :

$$\begin{aligned}
g * X &= U g(\Lambda) U^T X \\
&\approx U \sum_k \theta_k T_k(\tilde{\Lambda}) U^T X & (\tilde{\Lambda} = \frac{2}{\lambda_{max}} \Lambda - I_N) \\
&= \sum_k \theta_k T_k(\tilde{L}) x & (U \Lambda^k U^T = (U \Lambda U^T)^k)
\end{aligned}$$

A most popular graph convolutional network [114] further simplifies this approximation by reducing the order to 1:

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

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 4 depicts many sample models that focus on node-level graph convolution. The spectral perspective was previously explored (SGWT [89]), 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 [38], ISGNN [94], ChebNet [62]). Furthermore, GCN [114], and GraphSAGE [87] 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 [51], AR [129], ARMA [25]). Meanwhile, focus shifts to the spatial domain, which has dominated GNNs to this point. Random walks (ParWalk [201], DeepWalk [164], LINE [184]) and CNN (DCNN [14]) were used in early spatial approaches. Following that, MPNN [82] solidified the message-passing mechanism in spatial techniques. High-order polynomial approximation has been studied [117, 173, 199, 209], but only within the context of ChebNet or DCNN. It's worth noting that while numerous publications described their suggested methods from both spatial and spectral perspectives, only a few GNNs are covered [117, 155]. 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 Def. 2.1).

Table 4. Chronological list of notable GNNs in spatial and spectral domains

	Spatial	Spectral
Before 2015	ParWalk [201], DeepWalk [164], LINE [184]	Spectral GNN [38], ISGNN [94], Neural graph fingerprints [67]
2016	DCNN [14], Molecular Graph Convolutions [109], PATCHY-SAN [157]	GCN [114], ChebNet [62]
2017	MPNN [82], PGCN [74], GraphSAGE [87]	MoNet [155]
2018	GIN [209], Adaptive GCN [100], Fast GCN [47] JKNet [210], Large Scale GCN [76]	RationalNet[51], AR [129], CayleyNet [124]
2019	SGCN [199], DeepGCN [126], MixHop [3], PPAP [115]	ARMA [25], GDC [117], EigenPool [146], GWNN [207], Stable GCNN[191]
2020	SIGN [173], Spline GNN [242], UaGGP [141], GraLSP [106], GraphSAINT [227], DropEdge [172], BGNN[237], ALaGNN[206] Continuous GNN [205], GCNII [50], PPRGo [31], DAGNN [138], H2GCN [239]	GraphZoom [63]
2021	ADC [231], UGCN [105], DGC [196], E(n)GNN [175], GRAND [42], C&S [97], LGNN [140]	Interpretable Spectral Filter [110], Expressive Spectral Perspective [17], S2GC [238], BernNet [93]
2022	GINR[84], Adaptive SGC [43], PG-GNN [101], DIMP [215]	AGWN [150], ChebNetII [92], JacobiConv [194], SpecGNN [216], G ² CN [128], PGNN [75], ChebGibbsNet [9], SpecFormer[11], SIGN [10], Spectral Density [197]

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., g in Def. 2.2). In Section 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:

$$(A-1) \text{ LINEAR AGGREGATION} \rightrightarrows (A-2) \text{ POLYNOMIAL AGGREGATION} \rightrightarrows (A-3) \text{ RATIONAL AGGREGATION},$$

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:

$$(B-1) \text{ LINEAR APPROXIMATION} \rightrightarrows (B-2) \text{ POLYNOMIAL APPROXIMATION} \rightrightarrows (B-3) \text{ RATIONAL APPROXIMATION},$$

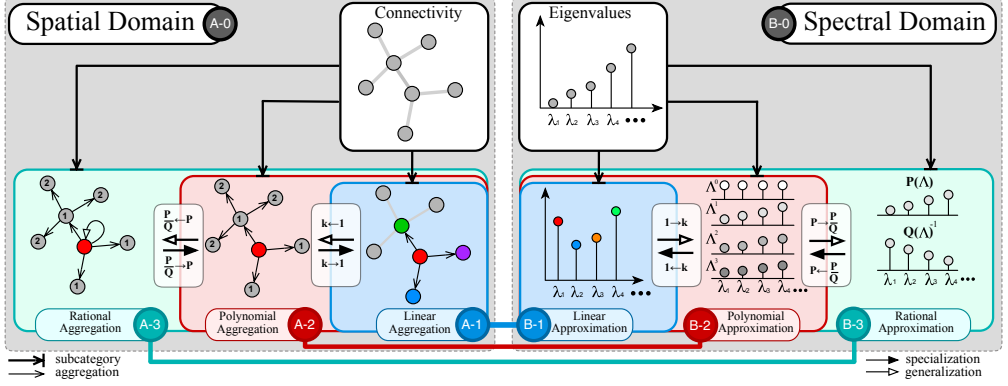


Fig. 1. Illustration of major graph neural operations and their relationship. Spatial and spectral methods are divided into three groups, respectively. Groups A-1, A-2, and A-3 are strongly-correlated by generalization and specialization, so are groups B-1, B-3, and B-3. The equivalence among them is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

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:

$$(A-1) \text{ LINEAR AGGREGATION} \Leftrightarrow (B-1) \text{ LINEAR APPROXIMATION},$$

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 uses a linear filter on eigenvalues with a negative slope, i.e., $g(\Lambda) = -\Lambda + a$. 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.,

$$(A-2) \text{ POLYNOMIAL AGGREGATION} \Leftrightarrow (B-2) \text{ POLYNOMIAL APPROXIMATION}.$$

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)

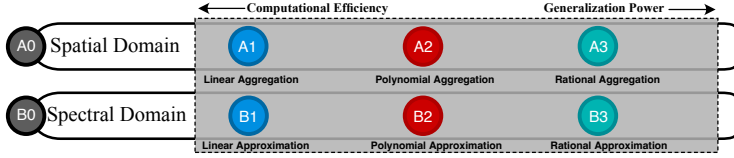


Fig. 2. Category and subcategory comparison.

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 them is that if the order is set too large, 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

$$(A-3) \text{ RATIONAL AGGREGATION} \Leftrightarrow (B-3) \text{ RATIONAL APPROXIMATION},$$

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. [4, 34, 56, 149, 165, 166, 186]

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 trade-off between computational efficiency and generalization capability. Table 5 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.

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) [82, 87, 164, 188, 208, 209]. The supervisory signal patterns are revealed by altering the weights for the node and its first order neighbors. The revised node embeddings, $Z(v_i)$, can be

Table 5. Structure of section 4 and 5.

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

represented in the following way:

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

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:

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

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 [107]. 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:

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

Equations 7 can be generalized as an identical form:

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

where $\tilde{\mathbf{A}}$ denotes normalized $\mathbf{A}$, which could be 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 [114] adds a self-loop to nodes, and applies a *renormalization* trick which 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:

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

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,

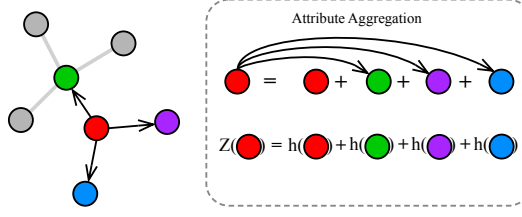


Fig. 3. Illustration of A-1: the current node (red) updates itself with its original representation plus neighbors.

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 [87] applies an aggregation among its neighbors. Take mean aggregator as example, it averages a node with its neighbors, i.e.,

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

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:

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

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 [209] develops conditions to maximize the power of GNNs, proposing a simple architecture, Graph Isomorphism Network (GIN). With strong theoretical support, GIN generalizes the WL test and updates node representations as:

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

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 [188] applies attention mechanism by adjusting neighbors' weights, instead of using uniform weights in many related works:

$$\mathbf{Z} = (\mathbf{W}_{att} \otimes \mathbf{A})\mathbf{X}, \quad (13)$$

where $\mathbf{W}_{att} \in \mathbb{R}^{N \times N}$ is a matrix, $\otimes$ denote element-wise multiplication, and calculated by a forward neural network $\mathbf{W}_{att}(i, j) = f(\mathbf{h}_i, \mathbf{h}_j)$ 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 [155] is similar to GAT, since its update follows:

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

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

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

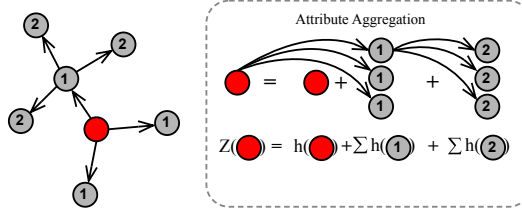


Fig. 4. Illustration of A-2: This current node (red) is using its original representation plus its first and second-order neighbors to update itself.

where μ_j are learnable $d \times d$ and $d \times 1$ 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 a attention model:

$$Z = (W_{MoNet} \otimes A) X, \quad (16)$$

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 research use higher orders of neighbors to integrate deeper structural information [14, 62, 85, 184, 199]. Because direct neighbors aren't always enough to describe a node's surroundings. Excessive order, on the other hand, generally averages all node representations, resulting in over-smoothing and a loss of emphasis on the immediate neighborhood [129]. 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 [3].

Define the shortest distance between node i and j as $d_G(i, j)$, and $\partial N(i, \tau)$ 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:

$$Z(v_i) = \Phi(v_i) h(v_i) + \underbrace{\sum_{u_j \in N(v_i, \tau=1)} \Psi^{(\tau=1)} h(u_j)}_{\text{1st-order neighbor}} + \underbrace{\sum_{u_j \in N(v_i, \tau=2)} \Psi^{(\tau=1)} h(u_j)}_{\text{2nd-order neighbor}} + \dots + \underbrace{\sum_{u_j \in N(v_i, \tau=k)} \Psi^{(\tau=k)} h(u_j)}_{\text{k-th order neighbor}} + \dots, \quad (17)$$

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:

$$Z = (\phi I + \sum_{j=1}^k \psi_j A^j) X = P_k(A) X, \quad (18)$$

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

$$Z = (\phi I + \sum_{j=1}^k \psi_j (D^{-\frac{1}{2}} A D^{-\frac{1}{2}})^j) X = (\phi I + \sum_{i=1}^k \psi_i \tilde{A}^i) X = (\sum_{i=0}^k \psi_i \tilde{A}^i) X = P_k(\tilde{A}) X, \quad (19)$$

where $\phi = \psi_0$, and A could also be normalized by random walk normalization: $\tilde{A} = D^{-1} A$. 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 [69, 94]. Introducing truncated Chebyshev polynomial for estimating wavelet in graph signal processing, ChebNet [62] embeds a novel neural network layer for the convolution operator. Specifically, ChebNet can be written as:

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

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:

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

which is exactly Equation 19. The predefined K is the order number of Chebyshev polynomial, and a larger K mean 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 [164] 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 walk is large enough, the transition probability of random walk on a graph can be represented as:

$$\tilde{\mathbf{A}} = \mathbf{D}^{-1} \mathbf{A}. \quad (22)$$

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:

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

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 [14] 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:

$$\mathbf{Z} = \mathbf{W} \odot \tilde{\mathbf{A}}^* \mathbf{X} = [w_1, w_2, w_3, \dots, w_k]^\top \odot [\tilde{\mathbf{A}}, \tilde{\mathbf{A}}^2, \tilde{\mathbf{A}}^3, \dots, \tilde{\mathbf{A}}^k]^\top \mathbf{X}, \quad (24)$$

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:

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

4.2.4 Scalable Inception Graph Neural Networks (SIGN). By generalizing GCN [114], GAT [188] and SGC [199], SIGN [173] constructs a block linear diffusion operators along with non-linearity. For node-wise classification tasks, SIGN has the form:

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

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

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

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:

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

4.2.5 Graph diffusion convolution (GDC). GDC [117] defines a generalized graph diffusion via the diffusion matrix:

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

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 which 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 :

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

4.2.6 Node2Vec. Node2Vec [85] defines a second-order random walk to control the balance between BFS (breath-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:

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

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., $t \rightarrow v \rightarrow t$), 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:

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

which can be transformed and reorganized as:

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

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

4.2.7 LINE[137] / SDNE[192]. 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:

$$\mathbf{Z} = \tilde{\mathbf{A}} \mathbf{X}. \quad (34)$$

Second-order: the pair of nodes are 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:

$$\mathbf{Z} = \tilde{\mathbf{A}}^2 \mathbf{X}. \quad (35)$$

Then the final learned node embeddings are formulated as:

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

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 [199] removes non-linear function between neighboring graph convolution layers, and combines graph aggregation in one single layer:

$$\mathbf{Z} = \tilde{\mathbf{A}}^k \mathbf{X} \quad (37)$$

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:

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

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 [115]. 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 [33, 165, 185]. Several works use a rational function on the adjacency matrix to perform self-aggregation, either explicitly or implicitly [25, 51, 102, 115, 124, 130, 143, 187].

Because generic label propagation is achieved by multiplying the graph Laplacian, self-aggregation may be achieved by multiplying the inverse graph Laplacian as follows:

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

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 $\mathbf{P}$, and as that of neighbors with probability $(1-\mathbf{P})$. The difference of A-3 beyond A-2 is that A-3 can avoid over-smoothing issue in an automatic manner [129, 232]. 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

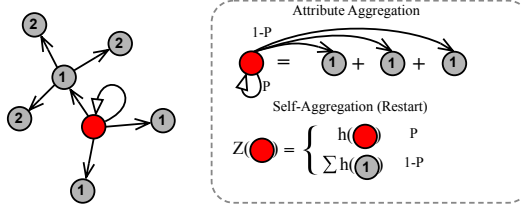


Fig. 5. Illustration of A-3: These current nodes (red) are using the representation that predates this iteration and the surrounding nodes to compute the total. In A-3, the ratio of the original representation remains stable, whereas A-1 does not control the ratio.

convolution can be described as an optimization problem [130, 158, 232, 241], e.g., (1) minimizing the supervised loss and (2) keeping the local neighborhood similar:

$$Z = \arg \min_Z \left\{ \underbrace{\|Z - Y\|_2^2}_{(1) \text{ supervised loss}} + \underbrace{\alpha \text{Tr}(Z^T L Z)}_{(2) \text{ neighborhood regularization}} \right\}, \quad (40)$$

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

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

However, because α 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 [50, 98, 172, 173, 210], 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.

4.3.1 Auto-Regressive. Label propagation (LP) [22, 233, 243] 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:

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

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

4.3.2 Personalized PageRank (PPNP). Obtaining node's representation via teleport (restart), PPNP [30, 115, 223] keeps the original representation (self-aggregation) X with probability α . Therefore, $1 - \alpha$ is the probability of performing the normal label propagation:

$$Z = \alpha (I - (1 - \alpha) \tilde{A})^{-1} X = \frac{\alpha}{I - (1 - \alpha) \tilde{A}} X, \quad (43)$$

where $\tilde{A} = D^{-1} A$ 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 [25] filter approximates any desired filter response function with updates as:

$$Z = \frac{b}{I - a \tilde{A}} X. \quad (44)$$

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

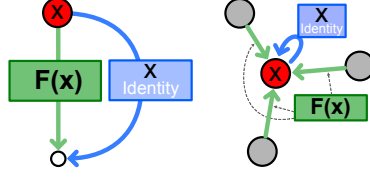


Fig. 6. Left: Residual Learning $x' = F(x) + x$; Right: Rational Aggregation: $x' = F(x) + x$

4.3.4 ParWalks. A partially absorbing random walk is a second-order Markov chain with partial absorption at each state. [201] shows that with proper absorption, the absorption probabilities can well capture the global graph structure. Note that the concept "absorption" in [201] is similar to "teleport" or "restart" in PPNP [115]. ParWalks defines the aggregation as:

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

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.

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

where α is redefined as a regularizer in the original paper [201]. 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:

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

The author also discussed the over-smoothing issue: when $\mathbf{A} = \mathbf{I}$ and as $\alpha \rightarrow 0$, 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 [51] 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 [91]. As shown in the work PPAP [164], the author proposed an iterative algorithm called APPAP, which is

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

where $\mathbf{Z}^k$ means the intermediate representations at k -th layer. This format is exactly the same as residual learning as illustrated in Figure 6: sum of multiple graph convolutions serve as $F(x)$, and each time identify input will be added. The slight difference is that Equation 48 has normalized the weights, i.e., $(1 - \alpha) + \alpha = 1$. Because Rational Aggregation (A-3) includes the inverse of a matrix, it has a high computational cost. Iterative methods are commonly used to efficiently determine the inverse of a matrix [25, 115, 124]. In this subsection, we only demonstrate one layer

or iteration of Auto-Regression, PPNP, ARMA, and ParWalks. Multiple iterations will result in a more sophisticated rational function. Higher orders are achieved by several iterations or layers. 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. On the other hand, calculating the inverse of the graph Laplacian is expensive, making iterative fashion a key object in online learning algorithms [95]. By leveraging the concept of conductance, with f as a heat distribution over the vertexes, $L(f)$ indicates the flux induced by f over the graph. Then based on the representer theorem [12, 176], $f(V_i) = L^{-1}(L(f))$ could be interpreted as the heat at each vertex been expressed concerning or derived from the flux through every vertex. Thus, when L sends a heat distribution f over each node to flux through each vertex, L^{-1} sends some of the fluxes over the graph back to the original heat distribution (i.e., keep 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 $P(L(f))$. M -th degree of $P(\cdot)$ means that each vertex can update M -th neighbors at most. Then using another updated flux in the reverse direction, $Q(L(f))^{-1}$ will adjust or reduce flux within N -th 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 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.[115].

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) in order 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 6. 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 [130], 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

Table 6. Summary of Representative GNNs

	(A-1) linear function of Λ	(B-1) linear function of Λ
GCN	$\mathbf{I} + \mathbf{A}$	$2 - \Lambda$
GraphSAGE	$\hat{\mathbf{D}}^{-1} + \tilde{\mathbf{A}}$	$2 - \Lambda$
GIN	$(1 + \epsilon) \mathbf{I} + \mathbf{A}$	$2 + \epsilon - \Lambda$
	(A-2) polynomial function of Λ	(B-2) polynomial function of Λ
ChebNet	$\phi \mathbf{I} + \sum_{i=1}^k \psi_i \tilde{\mathbf{A}}^i$	$\theta_0 \cdot 1 + \theta_1 \Lambda + \theta_2 \Lambda^2 + \dots$
DeepWalk	$\frac{1}{t+1} (\mathbf{I} + \tilde{\mathbf{A}} + \tilde{\mathbf{A}}^2 + \dots + \tilde{\mathbf{A}}^t)$	$\frac{1}{t+1} [\dots + (-1)^{t-1} + \binom{1}{t} (-1)^{t-1} + \dots]$
DCNN	$\psi_1 \tilde{\mathbf{A}} + \psi_2 \tilde{\mathbf{A}}^2 + \psi_3 \tilde{\mathbf{A}}^3 + \dots$	$\theta_1 \Lambda + \theta_2 \Lambda^2 + \theta_3 \Lambda^3 + \dots$
GDC	$\psi_1 \tilde{\mathbf{A}} + \psi_2 \tilde{\mathbf{A}}^2 + \psi_3 \tilde{\mathbf{A}}^3 + \dots$	$\theta_1 \Lambda + \theta_2 \Lambda^2 + \theta_3 \Lambda^3 + \dots$
Node2Vec	$\frac{1}{p} \mathbf{I} + \left(1 - \frac{1}{q}\right) \tilde{\mathbf{A}} + \frac{1}{q} \tilde{\mathbf{A}}^2$	$\left(1 + \frac{1}{p}\right) - \left(1 + \frac{1}{q}\right) \Lambda + \frac{1}{q} \Lambda^2$
LINE/SDNE	$\psi_1 \tilde{\mathbf{A}} + \psi_2 \tilde{\mathbf{A}}^2$	$\theta_1 \Lambda + \theta_2 \Lambda^2$
SGC	$0 \cdot \mathbf{I} + 0 \cdot \tilde{\mathbf{A}} + 0 \cdot \tilde{\mathbf{A}}^2 + \dots + 1 \cdot \tilde{\mathbf{A}}^K$	$\binom{K}{0} + \binom{K}{1} \Lambda^1 + \binom{K}{2} \Lambda^2 + \dots + \Lambda^n$
	(A-3) rational function of Λ	(B-3) rational function of Λ
Auto-Regress	$\frac{1}{(1+\alpha) \mathbf{I} - \alpha \tilde{\mathbf{A}}}$	$\frac{1}{1+\alpha(1-\Lambda)}$
PPNP	$\frac{\alpha}{1 - (1-\alpha) \tilde{\mathbf{A}}}$	$\frac{\alpha}{\alpha \mathbf{I} + (1-\alpha) \Lambda}$
ARMA	$\frac{b}{(1-\alpha) \tilde{\mathbf{A}}}$	$\frac{b}{(1-\alpha+a\Lambda)}$
ParWalk	$\frac{\alpha}{\alpha \mathbf{I} + 1 - \tilde{\mathbf{A}}}$	$\frac{\beta}{\beta + \Lambda}$

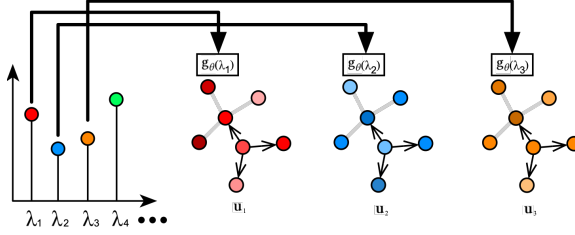


Fig. 7. Illustration of B-1: A linear function g maps the eigenvalues to new values.

component weights during aggregation. A linear g function is used in particular:

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

where $\mathbf{u}_i$ is the i -th eigenvector, and g is *frequency filter function* or *frequency response function* controlled by parameters θ , with selected l lowest frequency components. The goal of 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 ($\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3 \dots$) as $g_\theta(\lambda)$ 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:

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

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

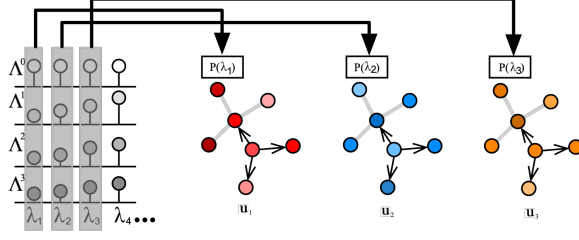


Fig. 8. Illustration of B-2: A polynomial function P maps the eigenvalues to new values.

eigenvector. The range of the eigenvalue is in $[0, 2)$ for the normalized Laplacian matrix [55]. If let *bias* be larger than or equal to 2, we have:

$$w(\lambda_i) = \frac{\text{bias} - \lambda_i}{N \cdot \text{bias} - \sum_j \lambda_j} = \underbrace{\frac{-1}{N \cdot \text{bias} - \sum_j \lambda_j}}_{\text{slope}} \lambda_i + \underbrace{\frac{\text{bias}}{N \cdot \text{bias} - \sum_j \lambda_j}}_{\text{intercept}}, \quad (51)$$

when *bias* is larger or equal than 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:

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

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

Remark: Polynomial approximation, in theory, gets more accurate as the order grows [6, 56, 162, 166, 186]. It's 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 9a, linear functions cannot accurately approximate *sign*(*x*). The situation improves dramatically when polynomial approximation is used, as demonstrated in Figure 9b. The variance will be greatly decreased if the order of the polynomial function is increased (Figure 9c). 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 [62], DeepWalk [164], Diffusion CNN [14], Simple Graph Convolution [199]).

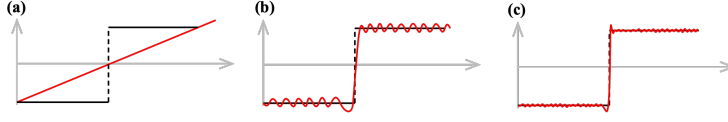


Fig. 9. Approximation for $\text{sign}(x)$ (in black): (a) linear approximation (b) polynomial approximation with low orders, (c) polynomial approximation with high orders.

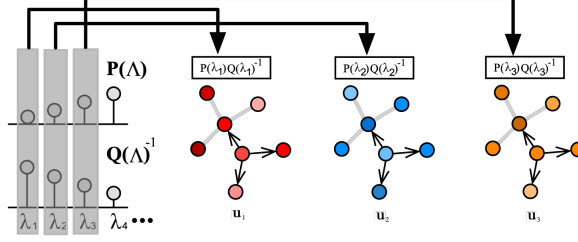


Fig. 10. Illustration of B-3: A rational function maps the eigenvalues to new values.

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, on the other hand, 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:

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

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 ($\mathbf{u}_1, \mathbf{u}_2, \dots$) as $\mathbf{g}_\theta(\lambda)$ which is a rational function.

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 [51].

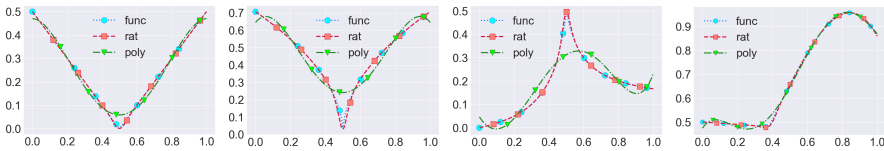


Fig. 11. Rational (rat) and polynomial (poly) approximation for several functions with discontinuity (func). From left to right: $\sqrt{|x-0.5|}$; $|x-0.5|$; $\frac{x}{10|x-0.5|+1}$; $\max(0.5, \sin(x+x^2)) - \frac{x}{20}$. Figures are from [51].

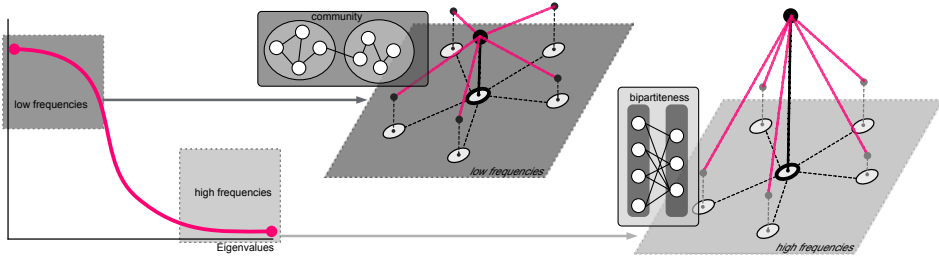


Fig. 12. Connection between the Spatial and Spectral Perspective

6 THEORETICAL ANALYSIS

In terms of volume, spatial-based approaches outnumber spectrum-based methods in the literature [2, 37, 204, 230, 234], 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 trade-off; 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 v.s. 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, whilst the last few eigenvalues indicate the graph’s bipartiteness [60, 61]. 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 is 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 [60]. It appears at first glance that global and 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 [73]. Specifically,

$$\Delta_t^2 \Delta_\omega^2 \geq \frac{1}{4}, \quad (54)$$

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 trade-off 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 [5]. 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

Table 7. 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

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 [29, 133, 189, 228, 240, 245]. 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 trade-off between. As shown in Table 7, four aspects are compared: **Methodology:** Spatial approaches describe local regions while working bottom-up, and identify global patterns using a graph frequency approach. On the other hand, 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, 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 Learning:** spatial method is easily converted to distributed learning [135, 178], whereas spectral is difficult to transfer. Even if it is possible to approximate spectrum technique using a neural network model [177] and then use distributed learning on a neural network [190], new nodes and edges must be retrained from scratch. Alternatively, the spatial technique can effectively manage online learning with streaming data [79]. **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 [15, 90]. This can be remedied by employing clustering before to the spatial technique [179]. 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 AX . Accordingly, polynomial and rational method are analyzed in Table 8 where K is the order number. To compare their expressive power, the convergence rate on challenging jump signal is employed as a benchmark [51] (a smooth signal cannot distinguish them). As shown in Table 8,

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 trade-off 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.

Table 8. 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	$O(1)$	$O(1/K)$	$O(\exp^{-\sqrt{K}})$

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 [114], polynomial filter [62], and rational filter [25]. Note that we only distinguish them in the function of A or Λ , keeping all the other configurations the same. The dataset includes representative homophily and heterophily datasets [43, 75, 105, 132, 144, 214]. The evaluation code is released ². The implementation is based on the official Pytorch Geometric [1]. Each model on each dataset is evaluated 50 times, and the results are averaged.

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. Rational method involve more matrix operation as ChebNet does, so rational method sometimes goes beyond log 1, but due to optimization in implementation (iterative algorithm), which make it a little faster than ChebNet, but still consistently slower 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.

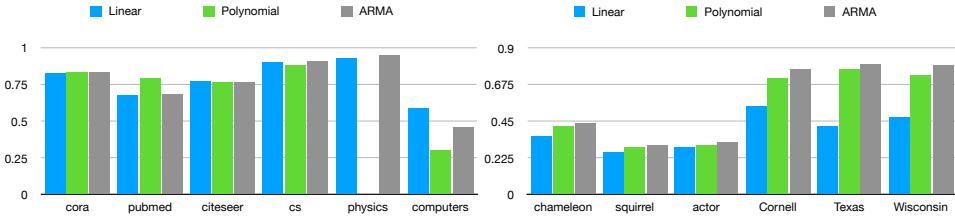


Fig. 13. Performance Comparison. **Left:** homophily dataset; **Right:** heterophily dataset

²https://github.com/aquastar/csur_bridge_spectral_spatial_gnn_survey

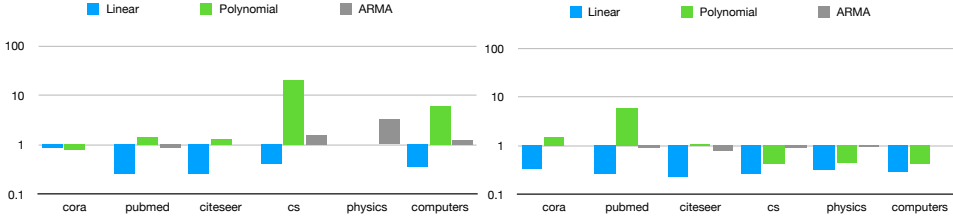


Fig. 14. Runtime Comparison. **Left:** homophily dataset; **Right:** heterophily dataset. (Y-axis is log-based)

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’ll 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* [87] applies uniform node sampling for each batch, which is equivalent to subgraph sampling. The likelihood of transfer therefore follows random normalization (i.e., $\hat{\mathbf{A}} = \mathbf{D}^{-1} \mathbf{A}$), which makes it part of Linear Aggregation (A-1). In the majority of follow-up works, the same methodology is used [227]: (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 [46, 48, 77, 100, 218, 227].

To avoid the recursive neighborhood expansion, FastGCN [46] 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 reduce 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 is selected for one node and activation is zero. To avoid taxing calculation of activation, Stochastic GCN [48] further uses the historical activation in the previous layer to avoid redundant re-evaluation. With adaptive sampling, nodes on subsequent layers are sampled in order to speed up GraphSAGE and FastGCN [100]. Learnable graph convolutional layer (LGCL) [77] selects a fixed number of neighboring nodes for each feature based on value ranking, and transform graph into 1-D data which 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 [226]. Cluster-GCN [53] samples a community of nodes determined by a graph clustering algorithm, and compute the graph convolution within each community.

7.1.2 Random Walk. To derive node-level representations with word2vec [151–153], various random walk algorithms are proposed [85, 164, 184, 192, 217, 227]. 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 have been performed on the paths.

As analyzed in previous section, *DeepWalk* [164] draws a number of random paths from the graph, which makes the transfer probability of random walk is, i.e., $\tilde{\mathbf{A}} = \mathbf{D}^{-1} \mathbf{A}$. Let the window size of skip-gram be $2t + 1$ and the index of current node is $t + 1$. Therefore, the updated representation is as $\mathbf{Z} = \frac{1}{t+1} \mathbf{P}(\tilde{\mathbf{A}}) \mathbf{X}$ (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 [125], SGNS is implicitly factorizing:

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

where $\mathbf{A}$ and $\mathbf{B}$ 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, [167] derived a 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 represents edge number and step size, respectively. Therefore, the target matrix to decompose is still a polynomials of $\mathbf{A}$. *Node2Vec* [85] defines a 2nd order random walk to control the balance between Breath 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 [167]:

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

where Node2Vec is demonstrated to be polynomial methods. *LINE* [184] and *SDNE* [192] 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 [227] employs multiple sampling polices but the best one is random walk. *PinSAGE* [223] improves the efficiency of GraphSAGE [87] 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 of the enormous number of steps may be needed before convergence.

7.2 Over-smoothing Point of View

We carve out two conditions under which neighborhood aggregation is not helpful: (1) when a node's neighbors are highly dissimilar and (2) when a node's embedding is already similar to that of its neighbors.

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 [28, 44, 50, 99, 126, 127, 138, 154, 160, 172, 210, 212, 232, 235, 240] can be reduced to one category of our proposed framework. H2GCN [240] proposed a method that combines direct neighbors with higher-order, which is equivalent to Polynomial Aggregation (A-2). Deep GCN [126, 127, 212] 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 [54] generalize PageRank and found the equivalence of GPR and Polynomial Approximation (B-2). JKNet [210] also follows the same residue methodology as Deep GCN. DAGNN [138] stacks multiple layers which uses different orders of propagation with the learnable weights, which makes it belong to Polynomial Aggregation (A-2). PairNorm [232] 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. [28] design 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 [99, 172] 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 [50] 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 [115] 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.

8 LIMIT, OPEN CHALLENGES AND CONCLUSION

In this paper, we present a unifying paradigm for comprehending GNNs created under various processes. Our study shows that the subcategories are closely related via generalization and specialization links within their domains, as well as equivalence ties across domains. We show the framework’s generalization power by reformulating existing GNNs models. As introduced in the sections above, spatial methods are designed by various ideas, and they can be interpreted well by the unified spectral theory. Therefore, we will discuss the potential that spectral theory, as a unified theoretical framework, may also extend to emerging directions.

Despite the an increasing number of emerging GNNs models [68] made in recent years, spectral methods so far has been intensively studied in node-level graph convolution only, leaving the other graph learning problem uncovered. In recent years, graph learning has been successfully extended to various tasks such as (sub)graph-level tasks, combinatorial optimization, explainability, domain application (brain, PDE solver, circuit, molecule, protein), generative graph, graph transformer, contrast learning, heterogeneous graph [18, 86, 142, 159, 222]. However, a unified framework is lacking, which is due to the underlying theories from spectral graph theory and graph signal processing applied in node-level graph convolution application by the first few pioneer work [62, 89, 114], but little attention has been paid to the theoretical study on the emerging topics rather than node-level convolution. Therefore, most new topics in graph machine learning is based on intuitive design or isolated theory, lacking a unified framework to compare and understand these separate learning models. In this section, we will list the possibility that all the selected topics can be looked at through a unified framework.

Theoretical Understanding -The vast majority of recent research has examined spatial and spectral approaches individually, and even from a theoretical standpoint. [111, 194, 209]. 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, however it is currently unclear how the learning approach can be optimally structured to execute this advantage. Using the partial differential equation [183], in which the diffusion equation and wave equation are analogous to polynomial and rational filters, provides an alternate 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 [108], 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 [123], spectral filters can be defined on directed graphs represented by non-symmetric adjacency matrices with Hermitian transpose.

Dynamic Graph - 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 [36, 147, 181]. 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 the relationship with the whole graph [40].

Higher-Order Interaction and Combinatorial Optimization - Most current work falls into first-order relationship at node-level. For example, the most spatial method is to learn the relationship between the current node and neighbors, but higher-order interaction is either ignored or implicitly included. Existing explainable learning also focuses on the neighbors' identification [225]. 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 [16, 72, 221] Hodge Laplacian [134, 171] and simplicial complex [19, 23, 27] provide more theoretical tools for modeling this combinatorial effect.

Multilayer Network - In numerous realistic biological and engineering systems, however, the units can be interconnected and interdependent via multiple interdependent and heterogeneous networks. Failure of interdependent nodes between linked networks may result in cascading failures inside and across the networks. Similar interactions exist in many cyber-physical systems, where the spread of misinformation about infectious diseases via social media can result in risky daily plans at the group level, resulting in an epidemic outbreak. 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 [8, 26, 59, 65]. Graph neural network research on this crucial area is scarce [83]. One popular technique to generalize principles from monolayer networks to multilayer networks is to "flatten" adjacency tensors into matrices (called "supra-adjacency matrix"), and spectral theory is available and worth researching.

REFERENCES

- [1] [n.d.]. Pyg Documentation. <https://pytorch-geometric.readthedocs.io/en/latest/>
- [2] 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*
- [3] 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).
- [4] Naum I Achieser. 2013. *Theory of approximation*. Courier Corporation.
- [5] Ameya Agaskar and Yue M Lu. 2013. A Spectral Graph Uncertainty Principle. *IEEE Transactions on Information Theory* 59, 7 (2013), 4338–4356.
- [6] Lars V Ahlfors. 1953. Complex analysis: an introduction to the theory of analytic functions of one complex variable. *New York, London* (1953), 177.
- [7] 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 *International Conference on Machine Learning*. PMLR, 212–222.
- [8] Alberto Aleta and Yamir Moreno. 2018. Multilayer networks in a nutshell. *arXiv preprint arXiv:1804.03488* (2018).
- [9] Anonymous. 2023. From ChebNet to ChebGibbsNet. In *Submitted to The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=2a5Ru3JtNe0> under review.
- [10] Anonymous. 2023. Sign and Basis Invariant Networks for Spectral Graph Representation Learning. In *Submitted to The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=Q-UHqMorzil> under review.
- [11] Anonymous. 2023. Specformer: Spectral Graph Neural Networks Meet Transformers. In *Submitted to The Eleventh International Conference on Learning Representations*. <https://openreview.net/forum?id=0pdSt3oyJa1> under review.
- [12] Andreas Argyriou, Charles A Micchelli, and Massimiliano Pontil. 2009. When is there a representer theorem? Vector versus matrix regularizers. *Journal of Machine Learning Research* 10, Nov (2009), 2507–2529.
- [13] Siddhant Arora. 2020. A survey on graph neural networks for knowledge graph completion. *arXiv preprint arXiv:2007.12374* (2020).
- [14] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*. 1993–2001.
- [15] 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.
- [16] Song Bai, Feihu Zhang, and Philip HS Torr. 2021. Hypergraph convolution and hypergraph attention. *Pattern Recognition* 110 (2021), 107637.
- [17] 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 *International Conference on Learning Representations (ICLR)*.
- [18] Federico Baldassarre and Hossein Azizpour. 2019. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686* (2019).
- [19] 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. *Physics Reports* 874 (2020), 1–92.
- [20] Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. *arXiv preprint arXiv:1806.09835* (2018).
- [21] Michael GH Bell, Yasunori Iida, et al. 1997. Transportation network analysis. (1997).
- [22] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. 2006. 11 label propagation and quadratic criterion. (2006).
- [23] Austin R Benson, Rediet Abebe, Michael T Schaub, Ali Jadbabaie, and Jon Kleinberg. 2018. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences* 115, 48 (2018), E11221–E11230.
- [24] 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*, Vol. 34. 549–556.
- [25] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2019. Graph neural networks with convolutional arma filters. *CoRR* (2019).
- [26] Ginestra Bianconi. 2018. *Multilayer networks: structure and function*. Oxford university press.
- [27] Ginestra Bianconi. 2021. *Higher-order networks*. Cambridge University Press.
- [28] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. [n.d.]. Beyond Low-frequency Information in Graph Convolutional Networks. *arXiv preprint arXiv:2101.00797* ([n.d.]).

- [29] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-frequency Information in Graph Convolutional Networks. *Thirty-Fifth AAAI Conference on Artificial Intelligence* (2021).
- [30] 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 *ACM SIGKDD*. 2464–2473.
- [31] 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 & #38; Data Mining (KDD)*. 2464–2473.
- [32] Béla Bollobás. 2004. *Extremal graph theory*. Courier Corporation.
- [33] Nicolas Boulle, Yuji Nakatsukasa, and Alex Townsend. 2020. Rational neural networks. *Advances in Neural Information Processing Systems* 33 (2020), 14243–14253.
- [34] John P Boyd. 2001. *Chebyshev and Fourier spectral methods*. Courier Corporation.
- [35] John Bradshaw, Matt J Kusner, Brooks Paige, Marwin HS Segler, and José Miguel Hernández-Lobato. 2018. A generative model for electron paths. *arXiv preprint arXiv:1805.10970* (2018).
- [36] 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.
- [37] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.
- [38] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLIS, April 2014*. <http://openreview.net>.
- [39] Khac-Hoai Nam Bui, Jiho Cho, and Hongsuk Yi. 2021. Spatial-temporal graph neural network for traffic forecasting: An overview and open research issues. *Applied Intelligence* (2021), 1–12.
- [40] Defu Cao, Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. 2021. Spectral temporal graph neural network for trajectory prediction. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1839–1845.
- [41] 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).
- [42] 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).
- [43] Sudhanshu Chanpuriya and Cameron Musco. 2022. Simplified Graph Convolution with Heterophily. In *Advances in Neural Information Processing Systems (NeurIPS)* 35. Curran Associates, Inc.
- [44] 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*, Vol. 34. 3438–3445.
- [45] Fanglan Chen, Zhiqian Chen, Subhodie 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.
- [46] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. Fastgcn: fast learning with graph convolutional networks via importance sampling. *International Conference on Learning Representations* (2018).
- [47] Jie Chen, Tengfei Ma, and Cao Xiao. 2018. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *International Conference on Learning Representations (ICLR)*.
- [48] Jianfei Chen, Jun Zhu, and Le Song. 2018. Stochastic Training of Graph Convolutional Networks with Variance Reduction. In *International Conference on Machine Learning*. 942–950.
- [49] 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*, Vol. 34. 27–34.
- [50] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. (2020).
- [51] Z. Chen, F. Chen, R. Lai, X. Zhang, and C. Lu. 2018. Rational Neural Networks for Approximating Graph Convolution Operator on Jump Discontinuities. (2018), 59–68. <https://doi.org/10.1109/ICDM.2018.00021>
- [52] 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).
- [53] 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 *ACM SIGKDD*.
- [54] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2020. Adaptive Universal Generalized PageRank Graph Neural Network. In *International Conference on Learning Representations*.

- [55] Fan RK Chung. 1997. *Spectral graph theory*. Number 92. American Mathematical Soc.
- [56] Harold Cohen. 2011. *Numerical approximation methods*. Springer.
- [57] Hanjun Dai, Chengtao Li, Connor Coley, Bo Dai, and Le Song. 2019. Retrosynthesis prediction with conditional graph logic network. *Advances in Neural Information Processing Systems* 32 (2019).
- [58] Eric H Davidson, Jonathan P Rast, Paola Oliveri, Andrew Ransick, Cristina Calestani, Chiou-Hwa Yuh, Takuya Minokawa, Gabriele Amore, Veronica Hinman, Cesar Arenas-Mena, et al. 2002. A genomic regulatory network for development. *science* 295, 5560 (2002), 1669–1678.
- [59] Manlio De Domenico, Clara Granell, Mason A Porter, and Alex Arenas. 2016. The physics of spreading processes in multilayer networks. *Nature Physics* 12, 10 (2016), 901–906.
- [60] Siemon de Lange, Marcel de Reus, and Martijn Van Den Heuvel. 2014. The Laplacian spectrum of neural networks. *Frontiers in computational neuroscience* 7 (2014), 189.
- [61] 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.
- [62] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*. 3844–3852.
- [63] 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 *International Conference on Learning Representations (ICLR)*.
- [64] 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 *International Conference on Learning Representations (ICLR)*.
- [65] Gaogao Dong, Dongli Duan, and Yongxiang Xia. 2021. A briefing survey on advances of coupled networks with various patterns. *Frontiers in Physics* (2021), 737.
- [66] James Howard Drew and Hui Liu. 2008. Diagnosing fault patterns in telecommunication networks. US Patent 7,428,300.
- [67] 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 *Advances in Neural Information Processing Systems (NeurIPS)* 28. Curran Associates, Inc., 2224–2232.
- [68] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. 2019. A fair comparison of graph neural networks for graph classification. *arXiv preprint arXiv:1912.09893* (2019).
- [69] Joan Bruna Estrach, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *2nd International Conference on Learning Representations, ICLR 2014*.
- [70] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.
- [71] 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).
- [72] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3558–3565.
- [73] Gerald B Folland and Alladi Sitaram. 1997. The uncertainty principle: a mathematical survey. *Journal of Fourier analysis and applications* 3, 3 (1997), 207–238.
- [74] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. 2017. Protein Interface Prediction using Graph Convolutional Networks. In *Advances in Neural Information Processing Systems (NeurIPS)* 30. Curran Associates, Inc., 6530–6539.
- [75] 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)*. 6878–6917.
- [76] 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)*. 1416–1424.
- [77] 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.
- [78] 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*, Vol. 33. 8303–8311.
- [79] Zhan Gao, Fernando Gama, and Alejandro Ribeiro. 2022. Wide and deep graph neural network with distributed online learning. *IEEE Transactions on Signal Processing* 70 (2022), 3862–3877.
- [80] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. 2019. Exact combinatorial optimization with graph convolutional neural networks. *Advances in Neural Information Processing Systems* 32 (2019).

- [81] Floris Geerts, Filip Mazowiecki, and Guillermo Perez. 2021. Let’s agree to degree: Comparing graph convolutional networks in the message-passing framework. In *International Conference on Machine Learning*. PMLR, 3640–3649.
- [82] 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-Volume 70*. JMLR. org, 1263–1272.
- [83] 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).
- [84] Daniele Grattarola and Pierre Vandergheynst. 2022. Generalised Implicit Neural Representations. In *Advances in Neural Information Processing Systems (NeurIPS) 35*. Curran Associates, Inc.
- [85] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.
- [86] Ruocheng Guo, Jundong Li, and Huan Liu. 2020. Learning individual treatment effects from networked observational data. *ACM International Conference on Web Search and Data Mining* (2020).
- [87] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.
- [88] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).
- [89] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.
- [90] Honglei He, Yuxuan He, Fang Wang, and Wenming Zhu. 2022. Improved K-means algorithm for clustering non-spherical data. *Expert Systems* 39, 9 (2022), e13062.
- [91] 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.
- [92] Mingguo He, Zhewei Wei, and Ji-Rong Wen. 2022. Convolutional Neural Networks on Graphs with Chebyshev Approximation, Revisited. In *Advances in Neural Information Processing Systems (NeurIPS) 35*. Curran Associates, Inc.
- [93] Mingguo He, Zhewei Wei, Hongteng Xu, et al. 2021. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems* 34 (2021), 14239–14251.
- [94] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* (2015).
- [95] 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.
- [96] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. 2012. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine* 29 (2012).
- [97] 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 *International Conference on Learning Representations (ICLR)*.
- [98] 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).
- [99] 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*
- [100] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. 2018. Adaptive sampling towards fast graph representation learning. *Advances in neural information processing systems* 31 (2018), 4558–4567.
- [101] 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)*. 9377–9409.
- [102] E. Isufi, A. Loukas, A. Simonetto, and G. Leus. 2017. Autoregressive Moving Average Graph Filtering. *IEEE Transactions on Signal Processing* 65, 2 (Jan 2017), 274–288. <https://doi.org/10.1109/TSP.2016.2614793>
- [103] Weiwei Jiang and Jiayun Luo. 2022. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications* (2022), 117921.
- [104] 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 Transactions on Artificial Intelligence* (2022).
- [105] Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. 2021. Universal Graph Convolutional Networks. In *Advances in Neural Information Processing Systems (NeurIPS) 34*. Curran Associates, Inc., 10654–10664.
- [106] Yilun Jin, Guojie Song, and Chuan Shi. 2020. GraLSP: Graph Neural Networks with Local Structural Patterns. In *Proceedings of the Thirty-Fourth Conference on Association for the Advancement of Artificial Intelligence (AAAI)*. 4361–4368.

- [107] Rie Johnson and Tong Zhang. 2007. On the effectiveness of Laplacian normalization for graph semi-supervised learning. *Journal of Machine Learning Research* 8, Jul (2007), 1489–1517.
- [108] Bo Kågström and Axel Ruhe. 1980. An algorithm for numerical computation of the Jordan normal form of a complex matrix. *ACM Transactions on Mathematical Software (TOMS)* 6, 3 (1980), 398–419.
- [109] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. 2016. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design (CAMD)* 30, 8 (2016), 595–608.
- [110] 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)*. 5388–5397.
- [111] Henry Kenlay, Dorina Thanou, and Xiaowen Dong. 2021. Interpretable stability bounds for spectral graph filters. In *International conference on machine learning*. PMLR, 5388–5397.
- [112] Dongkwan Kim and Alice Oh. 2020. How to find your friendly neighborhood: Graph attention design with self-supervision. In *International Conference on Learning Representations*.
- [113] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. 2018. Neural relational inference for interacting systems. In *International Conference on Machine Learning*. PMLR, 2688–2697.
- [114] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations* (2017).
- [115] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. (2018).
- [116] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Combining Neural Networks with Personalized PageRank for Classification on Graphs. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=H1gL-2A9Ym>
- [117] Johannes Klicpera, Stefan Weissenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems*. 13354–13366.
- [118] Boris Knyazev, Graham W Taylor, and Mohamed Amer. 2019. Understanding Attention and Generalization in Graph Neural Networks. *Advances in Neural Information Processing Systems* 32 (2019), 4202–4212.
- [119] David Lazer, Alex Sandy Pentland, Lada Adamic, Sinan Aral, Albert Laszlo Barabasi, Devon Brewer, Nicholas Christakis, Noshir Contractor, James Fowler, Myron Gutmann, et al. 2009. Life in the network: the coming age of computational social science. *Science (New York, NY)* 323, 5915 (2009), 721.
- [120] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.
- [121] 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 Article 62 (Nov. 2019), 25 pages. <https://doi.org/10.1145/3363574>
- [122] Jure Leskovec. 2019. GNNExplainer: Generating Explanations for Graph Neural Networks. (2019).
- [123] 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.
- [124] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. 2018. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing* 67, 1 (2018), 97–109.
- [125] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems* 27 (2014), 2177–2185.
- [126] 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.
- [127] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. 2020. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739* (2020).
- [128] Mingjie Li, Xiaojun Guo, Yifei Wang, Yisen Wang, and Zhouchen Lin. 2022. G²CN: Graph Gaussian Convolution Networks with Concentrated Graph Filters. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*. 12782–12796.
- [129] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.
- [130] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. 2019. Label Efficient Semi-Supervised Learning via Graph Filtering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.
- [131] 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.
- [132] 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)*. 13242–13256.
- [133] 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.
- [134] Lek-Heng Lim. 2020. Hodge Laplacians on graphs. *Siam Review* 62, 3 (2020), 685–715.

- [135] 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).
- [136] Wanyu Lin, Hao Lan, and Baochun Li. 2021. Generative Causal Explanations for Graph Neural Networks. *Proceedings of the International conference on Machklicpera2018predictine learning* (2021).
- [137] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion.. In *AAAI*, Vol. 15. 2181–2187.
- [138] 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.
- [139] 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 Twelfth ACM International Conference on Web Search and Data Mining*. 375–383.
- [140] Zemin Liu, Yuan Fang, Chenghao Liu, and Steven C.H. Hoi. 2021. Node-wise Localization of Graph Neural Networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*. 1520–1526.
- [141] 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 Thirty-Fourth Conference on Association for the Advancement of Artificial Intelligence (AAAI)*. 4957–4964.
- [142] Andreas Loukas. 2019. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199* (2019).
- [143] A. Loukas, A. Simonetto, and G. Leus. 2015. Distributed Autoregressive Moving Average Graph Filters. *IEEE Signal Processing Letters* 22, 11 (Nov 2015), 1931–1935. <https://doi.org/10.1109/LSP.2015.2448655>
- [144] 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 *Advances in Neural Information Processing Systems (NeurIPS)* 35. Curran Associates, Inc.
- [145] Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1412–1421.
- [146] 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)*. 723–731.
- [147] Muhammad Zubair Malik and Sarfraz Khurshid. 2012. Dynamic Shape Analysis Using Spectral Graph Properties. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*. 211–220. <https://doi.org/10.1109/ICST.2012.101>
- [148] Vivien Marx. 2012. High-throughput anatomy: charting the brain’s networks. *Nature* 490, 7419 (2012), 293.
- [149] John C Mason and David C Handscomb. 2002. *Chebyshev polynomials*. CRC Press.
- [150] 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 2022 SIAM International Conference on Data Mining (SDM)*. SIAM, 100–108.
- [151] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Workshop at International Conference on Learning Representations* (2013).
- [152] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013), 3111–3119.
- [153] Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *NAACL*. 746–751.
- [154] Yimeng Min, Frederik Wenkel, and Guy Wolf. 2020. Scattering gcn: Overcoming oversmoothness in graph convolutional networks. *arXiv preprint arXiv:2003.08414* (2020).
- [155] 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.
- [156] Mark EJ Newman. 2002. Spread of epidemic disease on networks. *Physical review E* 66, 1 (2002), 016128.
- [157] 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)*. 2014–2023.
- [158] Hoang NT and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550* (2019).
- [159] Kenta Oono and Taiji Suzuki. [n.d.]. GRAPH NEURAL NETWORKS EXPONENTIALLY LOSE EXPRESSIVE POWER FOR NODE CLASSIFICATION. ([n. d.]).
- [160] Kenta Oono and Taiji Suzuki. 2019. Graph Neural Networks Exponentially Lose Expressive Power for Node Classification. In *International Conference on Learning Representations*.

- [161] Alan V Oppenheim, John R Buck, and Ronald W Schafer. 2001. *Discrete-time signal processing*. Vol. 2. Upper Saddle River, NJ: Prentice Hall.
- [162] Ricardo Pachon. 2010. *Algorithms for polynomial and rational approximation*. Ph.D. Dissertation. University of Oxford.
- [163] 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).
- [164] 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 and data mining*. ACM, 701–710.
- [165] Penco Petrov Petrushev and Vasil Atanasov Popov. 2011. *Rational approximation of real functions*. Vol. 28. Cambridge University Press.
- [166] Michael James David Powell. 1981. *Approximation theory and methods*. Cambridge university press.
- [167] 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 Eleventh ACM International Conference on Web Search and Data Mining*. 459–467.
- [168] 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.
- [169] Eugene Y Remez. 1934. Sur la détermination des polynômes d’approximation de degré donnée. *Comm. Soc. Math. Kharkov* 10 (1934), 41–63.
- [170] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99.
- [171] Emily Ribando-Gros, Rui Wang, Jiahui Chen, Yiyong Tong, and Guo-Wei Wei. 2022. Graph and Hodge Laplacians: Similarity and Difference. *arXiv preprint arXiv:2204.12218* (2022).
- [172] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2019. Droppedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*.
- [173] 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).
- [174] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. 2019. Approximation ratios of graph neural networks for combinatorial problems. *Advances in Neural Information Processing Systems* 32 (2019).
- [175] 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)*. 9323–9332.
- [176] B Schölkopf, R Herbrich, and AJ Smola. 2001. A generalized representer theorem Computational Learning Theory ed D Helmbold and B Williamson.
- [177] 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).
- [178] 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).
- [179] 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.
- [180] 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 Signal Processing Magazine* 30, 3 (2013), 83–98.
- [181] Arlei Silva, Ambuj Singh, and Ananthram Swami. 2018. Spectral Algorithms for Temporal Graph Cuts (*WWW ’18*). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 519–528. <https://doi.org/10.1145/3178876.3186118>
- [182] Daniil Sorokin and Iryna Gurevych. 2018. Modeling semantics with gated graph neural networks for knowledge base question answering. *arXiv preprint arXiv:1808.04126* (2018).
- [183] Walter A Strauss. 2007. *Partial differential equations: An introduction*. John Wiley & Sons.
- [184] 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*.
- [185] Matus Telgarsky. 2017. Neural networks and rational functions. In *International Conference on Machine Learning*. PMLR, 3387–3393.
- [186] Lloyd N Trefethen. 2013. *Approximation theory and approximation practice*. Vol. 128. Siam.
- [187] Martin Trimmel, Mihai Zanfir, Richard Hartley, and Cristian Sminchisescu. 2022. ERA: Enhanced Rational Activations. In *European Conference on Computer Vision*. Springer, 722–738.
- [188] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. (2018).

- [189] Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2018. Deep Graph Infomax. In *International Conference on Learning Representations*.
- [190] Joost Verbraeken, Matthijs Woltling, Jonathan Katzy, Jeroen Kloppenburg, Tim Verbelen, and Jan S Rellermeyer. 2020. A survey on distributed machine learning. *Acm computing surveys (csur)* 53, 2 (2020), 1–33.
- [191] 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)*. 1539–1548.
- [192] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1225–1234.
- [193] 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).
- [194] Xiyuan Wang and Muhao Zhang. 2022. How Powerful are Spectral Graph Neural Networks. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*. 23341–23362.
- [195] Yewen Wang, Ziniu Hu, Yusong Ye, and Yizhou Sun. 2020. Demystifying Graph Neural Network Via Graph Filter Assessment. <https://openreview.net/forum?id=r1erNxBtwr>
- [196] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. 2021. Dissecting the Diffusion Process in Linear Graph Convolutional Networks. In *Advances in Neural Information Processing Systems (NeurIPS)* 34. Curran Associates, Inc., 5758–5769.
- [197] Yongyu Wang, Zhiqiang Zhao, and Zhuo Feng. 2022. Scalable Graph Topology Learning via Spectral Densification (WSDM '22). Association for Computing Machinery, New York, NY, USA, 1099–1108. <https://doi.org/10.1145/3488560.3498480>
- [198] Jiayi Wei, Maruth Goyal, Greg Durrett, and Isil Dillig. 2020. Lambdanet: Probabilistic type inference using graph neural networks. *arXiv preprint arXiv:2005.02161* (2020).
- [199] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *International Conference on Machine Learning*. 6861–6871.
- [200] 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).
- [201] Xiao-Ming Wu, Zhenguo Li, Anthony M So, John Wright, and Shih-Fu Chang. 2012. Learning with partially absorbing random walks. In *Advances in neural information processing systems*. 3077–3085.
- [202] 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*, Vol. 34. 1054–1061.
- [203] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google’s neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* (2016).
- [204] 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).
- [205] Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. 2020. Continuous Graph Neural Networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 10432–10441.
- [206] 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 Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*. 1303–1309.
- [207] Bingbing Xu, Huawei Shen, Qi Cao, Yunqi Qiu, and Xueqi Cheng. 2019. Graph Wavelet Neural Network. In *International Conference on Learning Representations (ICLR)*.
- [208] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
- [209] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=ryGs6iA5Km>
- [210] 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).
- [211] 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).
- [212] Chaoqi Yang, Ruijie Wang, Shuochao Yao, Shengzhong Liu, and Tarek Abdelzaher. 2020. Revisiting" over-smoothing" in deep gcns. *arXiv preprint arXiv:2003.13663* (2020).
- [213] 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).

- [214] Liang Yang, Mengzhe Li, Liyang Liu, Bingxin Niu, Chuan Wang, Xiaochun Cao, and Yuanfang Guo. 2021. Diverse Message Passing for Attribute with Heterophily. In *Advances in Neural Information Processing Systems (NeurIPS) 34*. Curran Associates, Inc., 4751–4763.
- [215] 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 Thirty-Sixth Conference on Association for the Advancement of Artificial Intelligence (AAAI)*. 4327–4336.
- [216] 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)*. 25261–25279.
- [217] Tianmeng Yang, Yujing Wang, Zhihan Yue, Yaming Yang, Yunhai Tong, and Jing Bai. 2022. Graph Pointer Neural Networks. In *Proceedings of the Thirty-Sixth Conference on Association for the Advancement of Artificial Intelligence (AAAI)*. 8832–8839.
- [218] Kai-Lang Yao and Wu-Jun Li. 2021. Blocking-based Neighbor Sampling for Large-scale Graph Neural Networks. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI)*. 3307–3313.
- [219] 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 Transactions on Intelligent Transportation Systems* (2020).
- [220] 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.
- [221] Jaehyuk Yi and Jinkyoo Park. [n.d.]. Hypergraph Convolutional Recurrent Neural Network. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*.
- [222] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. 2019. GNN Explainer: A Tool for Post-hoc Explanation of Graph Neural Networks. (2019).
- [223] 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 *ACM SIGKDD*. 974–983.
- [224] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2020. Explainability in Graph Neural Networks: A Taxonomic Survey. *arXiv preprint arXiv:2012.15445* (2020).
- [225] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. 2022. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).
- [226] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Accurate, efficient and scalable graph embedding. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 462–471.
- [227] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. Graphsaint: Graph sampling based inductive learning method. *International Conference on Learning Representations* (2019).
- [228] Songyang Zhang, Xuming He, and Shipeng Yan. 2019. Latentgcn: Learning efficient non-local relations for visual recognition. In *International Conference on Machine Learning*. PMLR, 7374–7383.
- [229] 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).
- [230] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2018. Deep learning on graphs: A survey. *arXiv preprint arXiv:1812.04202* (2018).
- [231] Jialin Zhao, Yuxiao Dong, Ming Ding, Evgeny Kharlamov, and Jie Tang. 2021. Adaptive Diffusion in Graph Neural Networks. In *Advances in Neural Information Processing Systems (NeurIPS) 34*. Curran Associates, Inc., 23321–23333.
- [232] Lingxiao Zhao and Leman Akoglu. 2019. PairNorm: Tackling Oversmoothing in GNNs. In *International Conference on Learning Representations*.
- [233] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. 2004. Learning with local and global consistency. In *Advances in neural information processing systems*. 321–328.
- [234] 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).
- [235] 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*
- [236] 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. *Advances in neural information processing systems* 32 (2019).
- [237] 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 Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI)*. 1452–1458.

- [238] Hao Zhu and Piotr Koniusz. 2021. Simple Spectral Graph Convolution. In *International Conference on Learning Representations (ICLR)*.
- [239] 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 *Advances in Neural Information Processing Systems (NeurIPS)* 33. Curran Associates, Inc., 7793–7804.
- [240] 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. *Advances in Neural Information Processing Systems* (2020).
- [241] 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).
- [242] Shichao Zhu, Lewei Zhou, Shirui Pan, Chuan Zhou, Guiying Yan, and Bin Wang. 2020. GSSNN: Graph Smoothing Splines Neural Network. In *Proceedings of the Thirty-Fourth Conference on Association for the Advancement of Artificial Intelligence (AAAI)*. 7007–7014.
- [243] 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.
- [244] Xiaofan Zhu and Michael Rabbat. 2012. Approximating signals supported on graphs. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 3921–3924.
- [245] Chenyi Zhuang and Qiang Ma. 2018. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference*. 499–508.
- [246] Eric Ziegel. 1987. Numerical recipes: the art of scientific computing.

A B-1

A.0.1 Graph Convolutional Network (GCN). Rewriting GCN [114] in spectral domain, we have:

$$\mathbf{Z} = \tilde{\mathbf{A}} \mathbf{X} = \mathbf{D}^{-\frac{1}{2}} (\mathbf{A} + \mathbf{I}) \mathbf{D}^{-\frac{1}{2}} \mathbf{X} = \mathbf{D}^{-\frac{1}{2}} (\mathbf{D} - \mathbf{L} + \mathbf{I}) \mathbf{D}^{-\frac{1}{2}} \mathbf{X} = (\mathbf{I} - \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}} + \mathbf{I}) \mathbf{X} = \mathbf{U}(2 - \Lambda) \mathbf{U}^\top \mathbf{X}, \quad (55)$$

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 $\mathbf{g}(\Lambda) = 1 - \Lambda$ which is a low-pass filter, i.e., smaller eigenvalues which correspond to low frequency components are assigned with a larger value.

A.0.2 GraphSAGE. Considering the MEAN aggregation as example, we can rewrite GraphSAGE [87] in matrix form:

$$\mathbf{Z} = \mathbf{D}^{-\frac{1}{2}} (\mathbf{I} + \mathbf{A}) \mathbf{D}^{-\frac{1}{2}} \mathbf{X} = (\mathbf{I} + \tilde{\mathbf{A}}) \mathbf{X} = (2\mathbf{I} - \tilde{\mathbf{L}}) \mathbf{X} = \mathbf{U}(2 - \Lambda) \mathbf{U}^\top \mathbf{X}. \quad (56)$$

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

$$(I + D^{-1}A) \mathbf{X} \quad (57) \quad \mathbf{U} (1 + \Lambda_P) \mathbf{U}^\top \mathbf{X} \quad (58)$$

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

A.0.3 Graph Isomorphism Network (GIN). Multi-Layer neural network is capable of fit the scale (i.e., normalization) [114], so GIN [209] can be rewritten as:

$$\mathbf{Z} = \mathbf{D}^{-\frac{1}{2}} [(1 + \epsilon) \mathbf{I} + \mathbf{A}] \mathbf{D}^{-\frac{1}{2}} \mathbf{X} = \mathbf{D}^{-\frac{1}{2}} [(2 + \epsilon) \mathbf{I} - \tilde{\mathbf{L}}] \mathbf{D}^{-\frac{1}{2}} \mathbf{X} = \mathbf{U}(2 + \epsilon - \Lambda) \mathbf{U}^\top \mathbf{X}. \quad (59)$$

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

B B-2

B.0.1 ChebNet. As analyzed in Equation 21, ChebNet [89] can be written as:

$$\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}}) \mathbf{X} = (\tilde{\theta}_0 \mathbf{I} + \tilde{\theta}_1 \tilde{\mathbf{L}} + \tilde{\theta}_2 \tilde{\mathbf{L}}^2 + \dots) \mathbf{X}, \quad (60)$$

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:

$$\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}}) \mathbf{X} = \mathbf{U}(\tilde{\theta}_0 \cdot 1 + \tilde{\theta}_1 \Lambda + \tilde{\theta}_2 \Lambda^2 + \dots) \mathbf{U}^\top \mathbf{X}, \quad (61)$$

where spectral response function is $\mathbf{g}(\Lambda) = \tilde{\theta}_0 \cdot 1 + \tilde{\theta}_1 \Lambda + \tilde{\theta}_2 \Lambda^2 + \dots = \mathbf{P}(\Lambda)$.

B.0.2 DeepWalk. Starting from Equation 23, DeepWalk [164] can be rewritten as:

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

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

B.0.3 Scalable Inception Graph Neural Networks (SIGN). Substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$ in Equation 28, it can be rewritten as:

$$\hat{\mathbf{Z}} = \sum_r \omega_r \widehat{(\mathbf{I} - \tilde{\mathbf{L}})^r} \mathbf{X} = \mathbf{U} \widehat{\mathbf{P}_{k=r}(\Lambda)} \mathbf{X} \mathbf{U}^\top. \quad (62)$$

B.0.4 Graph diffusion convolution (GDC). Substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$, general case in Equation 30 can be written as:

$$\mathbf{Z} = \sum_{k=0}^{\infty} \theta_k (\mathbf{I} - \tilde{\mathbf{L}}) = \mathbf{U} \sum_{k=0}^{\infty} \theta_k (1 - \Lambda)^k \mathbf{U}^\top = \mathbf{U} \mathbf{P}(\Lambda) \mathbf{X} \mathbf{U}^\top. \quad (63)$$

B.0.5 Diffusion convolutional neural networks (DCNN). As analyzed in Equation 24, DCNN [14] can be transformed with $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$ as:

$$\mathbf{Z} = \mathbf{P}(\mathbf{I} - \tilde{\mathbf{L}}) \mathbf{X} = \mathbf{U} \mathbf{P}(\Lambda) \mathbf{X} \mathbf{U}^\top, \quad (64)$$

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

B.0.6 Node2Vec. Node2Vec [85] can be rewritten in matrix form as Equation 32. Then it can be transformed and reorganized after substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$:

$$\mathbf{Z} = \left[\left(1 + \frac{1}{p}\right) \mathbf{I} - \left(1 + \frac{1}{q}\right) \tilde{\mathbf{L}} + \frac{1}{q} \tilde{\mathbf{L}}^2 \right] \mathbf{X} = \mathbf{U} \left[\left(1 + \frac{1}{p}\right) - \left(1 + \frac{1}{q}\right) \Lambda + \frac{1}{q} \Lambda^2 \right] \mathbf{U}^\top \mathbf{X}. \quad (65)$$

Therefore, Node2Vec's frequency response function is:

$$\mathbf{g}(\Lambda) = \left(1 + \frac{1}{p}\right) - \left(1 + \frac{1}{q}\right) \Lambda + \frac{1}{q} \Lambda^2, \quad (66)$$

which integrates a second order function of Λ with predefined parameters, i.e., p and q .

B.0.7 LINE/SDNE. As described in Equation 36, LINE [137] and SDNE [192] can be rewritten as:

$$\mathbf{Z} = \tilde{\mathbf{A}} \mathbf{X} + \alpha \tilde{\mathbf{A}}^2 \mathbf{X} = (\mathbf{I} - \tilde{\mathbf{L}}) \mathbf{X} + \alpha (\mathbf{I} - \tilde{\mathbf{L}})^2 \mathbf{X} = \mathbf{U}[(\mathbf{I} - \Lambda) + \alpha(1 - \Lambda)^2] \mathbf{X} \mathbf{U}^\top = \mathbf{U} \mathbf{P}_{k=2}(\Lambda) \mathbf{X} \mathbf{U}^\top, \quad (67)$$

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

B.0.8 Simple Graph Convolution (SGC). As analyzed in Equation 38, SGC can be transformed as:

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

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

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

B.0.9 Improved GCN (IGCN). By stacking multiple layers IGCN [130] is proposed as:

$$\mathbf{Z} = \tilde{\mathbf{L}}^k \mathbf{X} = \mathbf{U} \Lambda^k \mathbf{U}^\top \mathbf{X}, \quad (68)$$

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

C B-3

C.0.1 Auto-Regressive filter. Label propagation (LP) [22, 233, 243] is a prevail methodology for graph-based learning. The objective of LP is two-fold: one is to extract embeddings that matches 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:

$$\mathbf{Z} = (\mathbf{I} + \alpha \tilde{\mathbf{L}})^{-1} \mathbf{X} = \mathbf{U} \frac{1}{1 + \alpha(1 - \Lambda)} \mathbf{U}^\top \mathbf{X}. \quad (69)$$

C.0.2 PPNP. Personalized PageRank (PPNP) [115] can obtain node's representation via teleport (restart) probability α which indicates the ratio of keeping the original representation:

$$\mathbf{Z} = \frac{\alpha}{\mathbf{I} - (1 - \alpha)(\mathbf{I} - \tilde{\mathbf{L}})} \mathbf{X} = \mathbf{U} \frac{\alpha}{\alpha + (1 - \alpha) \Lambda} \mathbf{U}^\top \mathbf{X}, \quad (70)$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-1} \mathbf{A}$ is random-walk normalized adjacency matrix with self-loop. Equation 70 is with a rational function whose numerator is a constant.

C.0.3 ARMA filter. Substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$, Equation 44 can be rewritten as:

$$\mathbf{Z} = \frac{b}{\mathbf{I} - a(\mathbf{I} - \tilde{\mathbf{L}})} \mathbf{X} = \mathbf{U} \frac{b}{(1 - a) + a \Lambda} \mathbf{U}^\top \mathbf{X}. \quad (71)$$

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.0.4 ParWalks. [129, 201] Decomposing graph Laplacian, ParWalks can be written as:

$$\mathbf{Z} = \mathbf{U} \frac{\beta}{\beta + \Lambda} \mathbf{X} \mathbf{U}^\top, \quad (72)$$

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

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

C.0.5 RationalNet. Substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$, Equation 39 can be transform to Equation 53. The frequency response function is a generalized rational function.