Bridging the Gap between Spatial and Spectral Domains: A Survey on Graph Neural Networks

Zhiqian Chen[1] , **Fanglan Chen**[1] , **Lei Zhang**[1] , **Taoran Ji**[1] , **Kaiqun Fu**[1] , **Liang Zhao**[2] , **Feng Chen**[3] , **Chang-Tien Lu**[1]
[1]Virginia Tech, [2]George Mason University, [3]University of Taxes at Dallas
{czq, fanglanc, zhanglei, jtr, fukaiqun,ctlu}@vt.edu, feng.chen@utdallas.edu, lzhao9@gmu.edu

Abstract

The success of deep learning has been widely recognized in many machine learning tasks during the last decades, ranging from image classification and speech recognition to natural language understanding. As an extension of deep learning, Graph neural networks (GNNs) are designed to solve the non-Euclidean problems on graph-structured data which can hardly be handled by general deep learning techniques. Existing GNNs under various mechanisms, such as random walk, PageRank, graph convolution, and heat diffusion, are designed for different types of graphs and problems, which makes it difficult to compare them directly. Previous GNN surveys focus on categorizing current models into independent groups, lacking analysis regarding their internal connection. This paper proposes a unified framework and provides a novel perspective that can widely fit existing GNNs into our framework methodologically. Specifically, we survey and categorize existing GNN models into the spatial and spectral domains, and reveal connections among subcategories in each domain. Further analysis establishes a strong link across the spatial and spectral domains.

1 Introduction

The effectiveness of deep learning [28] has been widely recognized in various machine learning tasks [42; 43; 22; 49; 35] during the last decades, achieving remarkable success on Euclidean data. Recent decades has witnessed a great number of emerging applications where effective information analysis generally boils down to the non-Euclidean geometry of the data represented by a graph, such as social networks [27], transportation networks [4], spread of epidemic disease [38], brain's neuronal networks [36], gene data on biological regulatory networks [12], telecommunication networks [14], and knowledge graph [32]. Such non-Euclidean problems on graph-structured data can hardly be handled by general deep learning techniques. Modeling data by the graph is challenging due to that graph data is irregular, i.e., each graph has a variable size of nodes, and each node in a graph has a different number of neighbors, rendering some operations such as convolutions not directly applicable to the graph structure. Recently, there has been increasing interest in extending deep learning for graph data. Inspired by the success of deep learning, ideas are borrowed from deep learning models to handle the intrinsic complexity of the graph. This rising trend attracts increasing interest in the machine learning community, and a large number of GNN models are developed based on various theories [8; 25; 13; 19; 2; 46].

Despite GNNs dominate graph representation learning in recent years, there is still a limited understanding of their representational power and physical meaning. The lack of understanding of GNNs significantly hinders the comparison and improvement of state-of-the-art methods. This gap also makes it challenging to extend GNNs to many domains such as business intelligence or drug development, since black box models may be associated with uncontrollable risks. Therefore, there is a pressing need to demystify GNNs, which motivates researchers to explore a generalized framework for GNNs [53; 16; 55]. However, these works can only explain few GNNs, and the interpretation for the majority of GNNs is still missing.

There exist a large number of different mechanisms among current GNNs, such as random walk, Page Rank, attention model, low-pass filter, message passing, and so on. These methods can be classified into several coarse-grained groups [37; 20; 56; 58; 50] such as spectral [8; 25; 13] and spatial domain [19; 2; 46]. However, the current taxonomies fail to provide an understanding of the connections among different GNN models. Elucidating the underlying mechanisms of GNNs, and understanding connections among all types of GNNs is still at the forefront of GNNs research [53; 48; 31; 30]. This work is not trivial since the mechanisms behind existing GNNs are not inherently consistent, so their internal connection remains unclear. This gap incurs difficulty in understanding GNNs and comparing emerging methods. Previous surveys of GNNs [37; 20; 56; 58; 50] focus on categorizing current models into independent groups and expounding each group separately without analysis regarding their relationship.

The objective of this paper is to provide a unified framework to generalize GNNs, bridging the gap among the existing works in spatial and spectral domains which are currently deemed as independent. The main focus of this work is the connection among GNNs from a theoretical perspective, going beyond existing taxonomies and designing a new scheme for GNNs. Our research is unique in how it links present works of various categories of GNNs. Firstly, we briefly introduce the proposed framework, including spatial and spectral domains, and present their internal connection. Then detailed subcategories from the spatial and spectral domains are provided respectively, and several popular GNN examples are used to illustrate our taxonomies in each subcategory. Detailed contributions of this paper are summarized as follows:

1. **Proposing a taxonomy for summarizing GNN approaches in the spatial domain.** This paper unifies GNN methods in the spatial domain by formulating spatial operation on graph connectivity. Then GNNs can be treated as the same function of graph matrix with different configurations.
2. **Providing a taxonomy for summarizing GNN methods in the spectral domain.** This survey categorizes GNN models by frequency response function in the spectral domain, and applies approximation theory to illuminate their generalization and specialization relationship.
3. **Incorporating spatial and spectral models into a unified framework.** By comparing the analytical forms, the proposed framework links the frequency response function of the spectral domain and node aggregation function in the spatial domain.

The rest of this survey is organized as follows. In Section 2, we introduce problem setup and necessary preliminaries. Then we present details of our proposed taxonomy in Section 3. Section 4 and 5 elaborate the details of the proposed taxonomy. We conclude the survey in Section 6.

2 Problem Setup and Preliminary

This section outlines the background of graph neural networks and problem setup.

Definition 2.1 (Graph) *A graph is defined as* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$*, where* $\mathcal{V}$ *is a set of* n *nodees,* $\mathcal{E}$ *represents edges. An entry* $v_i \in \mathcal{V}$ *denotes a node,* $e_{i,j} = \{v_i, v_j\} \in \mathcal{E}$ *indicates an edge between node* i *and* j*. The adjacency matrix* $\mathbf{A} \in \mathbb{R}^{N \times N}$ *is defined by* $\mathbf{A}_{i,j} = 1$ *iff there is a link between node* i *and* j*. Graph signal or node attributes* $\mathbf{X} \in \mathbb{R}^{N \times D}$ *is a feature matrix with each entry* $x_i \in \mathbf{X}$ *representing the feature vector on node* i*.*

Learning node-level embeddings can be summarized as

$$\mathbf{Z} = f_\Theta(\mathcal{G}, \mathbf{X}), \tag{1}$$

where Θ indicates the parameters of the model. We aim to find a $f_\Theta(\cdot)$ which can integrate graph structure and original node attributes, outputting a new node embedding $\mathbf{Z}$. Since this review focuses on two major categories, i.e., spectral and spatial methods, two related definitions are listed below for understanding this paper.

Definition 2.2 (Spatial Method) *Treating graph Laplacian* $\mathbf{L}$ *[11] as spatial connectivity among nodes, spatial method integrate* $\mathbf{L}$ *and signal* $\mathbf{X}$*:*

$$\mathbf{Z} = f(\mathcal{G}, \mathbf{X}) = f(\mathbf{L}, \mathbf{X}). \tag{2}$$

Therefore, spatial methods focus on finding a function $f(\cdot)$*.*

Definition 2.3 (Spectral Method) *Graph Laplacian is defined as* $\mathbf{L} = \mathbf{D} - \mathbf{A} \in \mathbb{R}^{N \times N}$ *where* $\mathbf{D}$ *is degree matrix. Due to its generalization ability [7] , the normalized Laplacian is defined as* $\tilde{\mathbf{L}} = \mathbf{D}^{-\frac{1}{2}}\,\mathbf{L}\,\mathbf{D}^{-\frac{1}{2}}$*. The Laplacian* $\mathbf{L}$ *is diagonalized by the Fourier basis* $\mathbf{U}^\intercal$ *(i.e., graph Fourier transform) [44; 60]:* $\tilde{\mathbf{L}} = \mathbf{U}\,\mathbf{\Lambda}\,\mathbf{U}^\intercal$ *where* $\mathbf{\Lambda}$ *is the diagonal matrix whose diagonal elements are the corresponding eigenvalues (i.e.,* $\mathbf{\Lambda}_{ii} = \lambda_i$*), and* $\mathbf{U}$ *is also called eigenvectors. The graph Fourier transform of a signal* $\mathbf{X}$ *is defined as* $\hat{\mathbf{X}} = \mathbf{U}^\intercal\,\mathbf{X} \in \mathbb{R}^{N \times N}$ *and its inverse as* $\mathbf{X} = \mathbf{U}\,\hat{\mathbf{X}}$*. A graph convolution operation is defined in the Fourier domain such that*

$$f_1 * f_2 = \mathbf{U}\left[(\mathbf{U}^\intercal\, f_1) \odot (\mathbf{U}^\intercal\, f_2)\right], \tag{3}$$

where $\odot$ *is the element-wise product, and* f_1/f_2 *are two signals defined on node domain. It follows that a node signal* $f_2 = \mathbf{X}$ *is filtered by spectral signal* $\hat{f}_1 = \mathbf{U}^\intercal\, f_1 = \mathbf{g}$ *as:*

$$\mathbf{Z} = f(\tilde{\mathbf{L}}, \mathbf{X}) = \mathbf{U}\left[\mathbf{g}(\mathbf{\Lambda}) \odot (\mathbf{U}^\intercal\,\mathbf{X})\right] = \mathbf{U}\,\mathbf{g}(\mathbf{\Lambda})\,\mathbf{U}^\intercal\,\mathbf{X} \tag{4}$$

where $\mathbf{g}$ *is known as frequency response function of filter* $\tilde{\mathbf{L}}$*. Therefore, spectral methods is defined as learning* $\mathbf{g}(\cdot)$*.*

3 The proposed taxonomy

As shown in Fig. (1), the proposed framework categorizes GNNs into the spatial (A0) and spectral (B0) groups, each of which is further divided into three subcategories respectively. By transforming the analytical form of these subcategories, we found three relations of equivalence as below:

(A1)*Local Aggregation* $\Longleftrightarrow$ **(B1)*Frequency Aggregation*:** Through graph and matrix theory, *Local Aggregation* adjusts weights on a set of neighbor nodes, which corresponds to adjusting weights on frequency components in *Frequency Aggregation.*

(A2)*Connectivity Order* $\Longleftrightarrow$ **(B2)*Approximation Order*:** Accumulating different orders of neighbors in *Connectivity Order* can be rewritten as the sum of different orders of frequency components, which is exactly the analytical form of *Approximation Order*.

(A3)*Propagation Type* $\Longleftrightarrow$ **(B3)*Approximation Type*:** *Propagation Type* defines a label propagation with or without reverse propagation, while *Approximation Type* adjusts filter function with or without simple denominator (i.e., 1). In this way, they share the same formula after a simple transformation.

Sections 3 and 4 will discuss details of the proposed taxonomy, and illustrate each subcategory with several GNN examples.

4 Spatial-based GNNs (A0)

Several important aspects are often discussed in the existing literature of spatial methods, such as self-loop, normalization, high-order neighbors, aggregation, and combination among nodes. Based on these operations, we propose a new taxonomy of graph neural networks, categorizing spatial-based GNNs into three groups:

4.1 Local Aggregation (A1)

A number of works [41; 53; 52; 16; 19; 46] can be treated as learning the aggregation scheme among first order neighbors (i.e., direct neighbors). This aspect focuses on adjusting the weights for node and its neighbors to reveal the pattern regarding the supervision signal. Formally, updated node embeddings, $\mathbf{Z}(v)$, can be written as:

$$\mathbf{Z}(v_i) = \Phi(v_i)\,\mathbf{h}(v_i) + \sum_{u_j \in \mathcal{N}(v_i)} \Psi(u_j)\,\mathbf{h}(u_j), \tag{5}$$

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

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

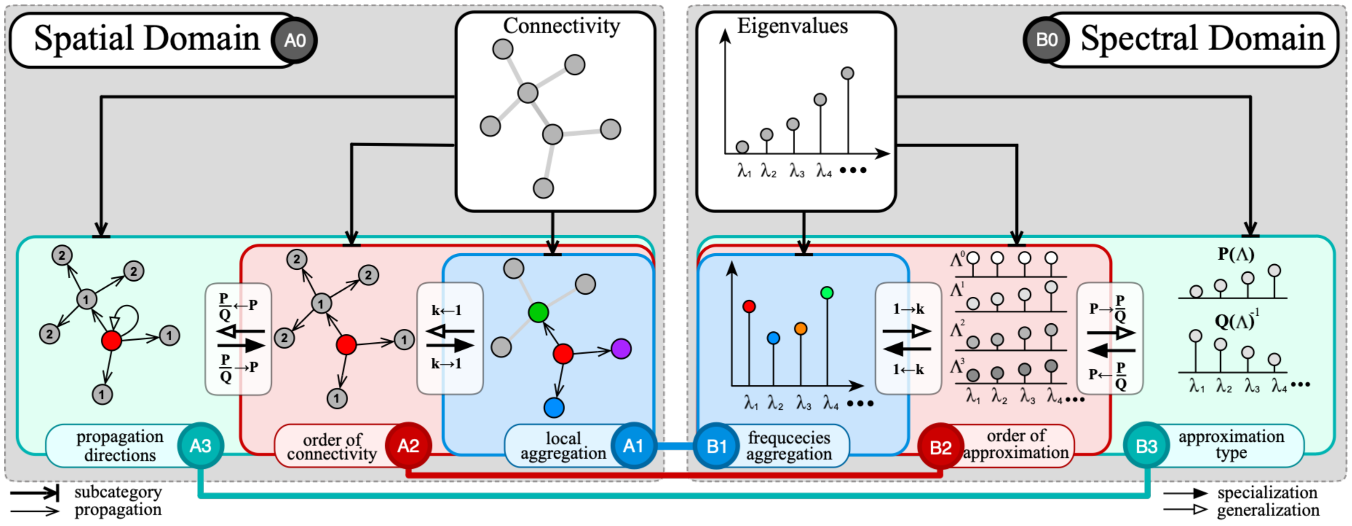

Figure 1: Illustration of major graph neural operations and their relationship. Spatial and spectral methods are divided into three groups, respectively. Group A1, A2, and A3 are strongly-correlated by generalization and specialization, so are group B1, B3, and B3. The equivalence relationship among them is marked in the same color.

or symmetric normalization:

$$\mathbf{Z}(v_i) = \Phi(v_i)\,\mathbf{h}(v_i) + \sum_{u_j \in \mathcal{N}(v_i)} \Psi(u_j)\frac{\mathbf{h}(u_j)}{\sqrt{d_i d_j}}, \tag{7}$$

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 [24]. In a simplified configuration, weights for the neighbors (Ψ) are the same. Therefore, they can be rewritten in matrix form as:

$$\mathbf{Z} = \phi\,\mathbf{X} + \psi\,\mathbf{D}^{-1}\,\mathbf{A}\,\mathbf{X} = (\phi\,\mathbf{I} + \psi\,\mathbf{D}^{-1}\,\mathbf{A})\,\mathbf{X} \tag{8}$$

or

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

where ϕ and ψ are the weights. Eq. (8) and (9) can be generalized as the same form:

$$\mathbf{Z} = (\phi\,\mathbf{I} + \psi\tilde{\mathbf{A}})\,\mathbf{X} \tag{10}$$

where $\tilde{\mathbf{A}}$ denotes normalized $\mathbf{A}$, which could be implemented by random walk or symmetric normalization. Several state-of-the-art methods are selected to illustrate this schema:

(1) Graph Convolutional Network [25] is a simplification of ChebNet [13]. GCN 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:

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

where $\hat{\mathbf{A}} = \mathbf{A}+\mathbf{I}$, and $\tilde{\mathbf{A}}$ is normalized adjacency matrix with self loop. Therefore, Eq. (4.1) is equivalent to Eq. (10) when setting $\phi = 0$ and $\psi = 1$ with the renormalization trick, and the result of GCN is exactly the sum of the current node and average of its neighbors.

(2) GraphSAGE [19] with mean aggregator averages a node with its neighbors by:

$$\mathbf{Z}(v_i) = \mathrm{MEAN}\left(\{\mathbf{h}(v_i)\} \cup \{\mathbf{h}(u_j), \forall u_j \in \mathcal{N}(v_i)\}\right), \tag{11}$$

where $\mathbf{h}$ indicates the representation, and $\mathcal{N}$ denotes the neighbor nodes. Eq. (11) can be written in matrix form:

$$\mathbf{Z} = \hat{\mathbf{D}}^{-1}(\mathbf{I}+\mathbf{A})\,\mathbf{X} = (\hat{\mathbf{D}}^{-1} + \tilde{\mathbf{A}})\,\mathbf{X}, \tag{12}$$

which is equivalent to Eq. (10) with $\phi = 1$ and $\psi = 1$. Note that the key difference between GCN and GraphSAGE is the normalization. The former is symmetric normalization and the latter is random walk normalization.

(3) Graph Isomorphism Network (GIN) [53] updates node representations as:

$$\mathbf{Z} = (1+\epsilon)\cdot\mathbf{h}(v) + \sum_{u_j \in \mathcal{N}(v_i)} \mathbf{h}_(u_j) = [(1+\epsilon)\,\mathbf{I} + \mathbf{A}]\,\mathbf{X}, \tag{13}$$

which is equivalent to Eq. (10) with $\phi = 1+\epsilon$ and $\psi = 1$. Note that **GIN** dose not perform normalization.

4.2 Order of Connectivity (A2)

To collect richer local structure, several studies [2; 13; 48; 45; 17] involve higher orders of neighbors. Since direct neighbors (i.e., first-order neighbors) are not always sufficient for representing the node. On the other hand, large order usually averages all node representations, causing an over-smoothing issue and losing its focus on the local neighborhood [30]. This motivates many models to tune the aggregation scheme on different orders of neighbors. Therefore, proper constraint and flexibility of orders are critical for node representation. High order of neighbors has been proved to characterize challenging signal such as Gabor-like filters [1]. Formally, this type of work can be written as:

$$\mathbf{Z}(v_i) = \phi\,\mathbf{h}(v) + \overbrace{\sum_{u_j^{(1)} \in \mathcal{N}(v)} \psi_j^{(1)}\,\mathbf{h}(u_j^{(1)})}^{\text{1st order neighbor}} + \overbrace{\sum_{u_j^{(2)} \in \bigcup_j \mathcal{N}(u_j^{(1)})} \psi_j^{(2)}\,\mathbf{h}(u_j^{(2)})}^{\text{2nd order neighbor}}$$

$$\ldots + \overbrace{\sum_{u_j^{(k+1)} \in \bigcup_j \mathcal{N}(u_j^{(k)})} \psi_j^{(k+1)}\,\mathbf{h}(u_j^{(k+1)})}^{\text{k-th order neighbor}} + \ldots, \tag{14}$$

where $u_i^{(n)}$ indicates a n-th order neighbors of node v. Eq. (14) can be rewritten in matrix form:

$$\mathbf{Z} = (\phi\,\mathbf{I} + \sum_{j=1}^{k} \psi_j\, \mathbf{A}^j)\,\mathbf{X} = \mathbf{P}(\mathbf{A})\,\mathbf{X}, \tag{15}$$

where $\mathbf{P}(\cdot)$ is a polynomial function. Applying normalization, Equation (15) can be rewritten in matrix form as:

$$\mathbf{Z} = (\phi\,\mathbf{I} + \sum_{j=1}^{k} \psi_i (\mathbf{D}^{-\frac{1}{2}}\,\mathbf{A}\,\mathbf{D}^{-\frac{1}{2}})^j)\,\mathbf{X} = \mathbf{P}(\tilde{\mathbf{A}})\,\mathbf{X}, \tag{16}$$

where $\phi = \psi_0$, and $\mathbf{A}$ could also be normalized by random walk normalization: $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\,\mathbf{A}$. Several existing works are analyzed below, showing that they are variants of Eq. (15) or (16):

(1) ChebNet [21] first introduced truncated Chebyshev polynomial for estimating wavelet in graph signal processing. Based on this polynomial approximation, Defferrard et al.[13] designed ChebNet which embeds a novel neural network layer for the convolution operator. Specifically, ChebNet is written as:

$$\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}})\,\mathbf{X} = (\tilde{\theta_0}\,\mathbf{I} + \tilde{\theta_1}\tilde{\mathbf{L}} + \tilde{\theta_2}\tilde{\mathbf{L}}^2 + ...)\,\mathbf{X}, \tag{17}$$

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:

$$\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}})\,\mathbf{X} = [\tilde{\theta_0}\,\mathbf{I} + \tilde{\theta_1}(\mathbf{I} - \tilde{\mathbf{A}}) + \tilde{\theta_2}(\mathbf{I} - \tilde{\mathbf{A}})^2 + ...]\,\mathbf{X}, \tag{18}$$

which can be reorganized as:

$$\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}})\,\mathbf{X} = (\phi\,\mathbf{I} + \sum_{i=1}^{k} \psi_i\, \tilde{\mathbf{A}}^i)\,\mathbf{X} = \mathbf{P}(\tilde{\mathbf{A}})\,\mathbf{X}, \tag{19}$$

which is exactly Eq. (16)

(2) DeepWalk [41] is a random walk based model that is integrated with deep learning technique. DeepWalk first draws a group of random paths from graph and applies a skip-gram algorithm to extract node features. Assuming the number of samples is large enough, then the transfer probability of random walk on a graph can be written as:

$$\tilde{\mathbf{A}} = \mathbf{D}^{-1}\,\mathbf{A}, \tag{20}$$

with random walk normalization. Let the window size of skip-gram be $2t + 1$ and the current node is the (t+1)-th one, the farthest neighbor current node can reach is a t-th order one. If the training is sufficient and samples are adequate, the node will converge to its neighbors. Therefore, the updated representation is as follows:

$$\mathbf{Z} = \frac{1}{t+1}(\mathbf{I} + \tilde{\mathbf{A}} + \tilde{\mathbf{A}}^2 + ... + \tilde{\mathbf{A}}^t)\,\mathbf{X} = \frac{1}{t+1}\mathbf{P}(\tilde{\mathbf{A}})\,\mathbf{X} \tag{21}$$

(3) Diffusion convolutional neural networks (DCNN) [2] considers using a degree-normalized transition matrix, i.e., renormalized adjacency matrix: $\tilde{\mathbf{A}} = \tilde{\mathbf{D}}\,\mathbf{A}$:

$$\mathbf{Z} = W \odot \tilde{\mathbf{A}}^*\,\mathbf{X}, \tag{22}$$

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:

$$\mathbf{Z} = (\psi_1\,\tilde{\mathbf{A}} + \psi_2\,\tilde{\mathbf{A}}^2 + \psi_3\,\tilde{\mathbf{A}}^3 + ...)\,\mathbf{X} = \mathbf{P}(\tilde{\mathbf{A}})\,\mathbf{X}, \tag{23}$$

(4) Node2Vec [17] defines a 2nd order random walk to control the balance between BFS (breath first search) and DFS (depth first search). Consider a random walk that traversed edge (t, v) and now resides at node v. The transition probabilities to next stop x from node v is defined as:

$$\alpha(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \tag{24}$$

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

$$\mathbf{Z} = (\frac{1}{p}\cdot \overbrace{\mathbf{I}}^{\text{source}} + \overbrace{\tilde{\mathbf{A}}}^{\text{BFS}} + \frac{1}{q}\overbrace{(\tilde{\mathbf{A}}^2 - \tilde{\mathbf{A}})}^{\text{DFS}})\,\mathbf{X}, \tag{25}$$

which can be transformed and reorganized as:

$$\mathbf{Z} = [\frac{1}{p}\,\mathbf{I} + (1 - \frac{1}{q})\,\tilde{\mathbf{A}} + \frac{1}{q}\,\tilde{\mathbf{A}}^2]\,\mathbf{X} = \mathbf{P}(\tilde{\mathbf{A}})\,\mathbf{X}, \tag{26}$$

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

(5) Simple Graph Convolution (SGC) [48] removes nonlinear function between neighboring graph convolution layers, and combine graph propagation in one single layer:

$$\mathbf{Z} = \tilde{\mathbf{A}}^K\,\mathbf{X} \tag{27}$$

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:

$$\mathbf{Z} = (0\cdot\mathbf{I} + 0\cdot\tilde{\mathbf{A}} + 0\cdot\tilde{\mathbf{A}}^2 + ... + 1\cdot\tilde{\mathbf{A}}^K)\,\mathbf{X} = \mathbf{P}(\tilde{\mathbf{A}})\,\mathbf{X} \tag{28}$$

4.3 Propagation Directions (A3)

Most works merely consider label propagation from the node to its neighbors (i.e., gathering information form its neighbors) but ignore propagation in reverse direction. Reverse propagation means that labels or attributes can be propagated back to itself with probabilities, or restart propagating with a certain probability. This reverse behavior can avoid over-smoothing issue [26]. Note that [A2] can also alleviate over-smoothing issue by manually adjusting the order number, while [A3] can automatically fit the proper order number. Several works explicitly or implicitly implement reverse propagation by applying rational function on the adjacency matrix [10; 26; 31; 34; 23; 29; 6]. Since general label propagation is implemented by multiplying graph Laplacian, reverse propagation could be implemented by multiplying inverse graph Laplacian as:

$$\mathbf{Z} = \mathbf{P}(\tilde{\mathbf{A}})\,\mathbf{Q}(\tilde{\mathbf{A}})^{-1}\,\mathbf{X} = \frac{\mathbf{P}(\tilde{\mathbf{A}})}{\mathbf{Q}(\tilde{\mathbf{A}})}\,\mathbf{X}, \tag{29}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are two different polynomial functions, and the bias of $\mathbf{Q}$ is often set to 1.

(1) Auto-Regressive label propagation (LP) [59; 57; 5] 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 label, the other is to become similar to neighboring vertices. The label can be treated as part of node attributes, so we have:

$$\mathbf{Z} = (\mathbf{I} + \alpha\,\tilde{\mathbf{L}})^{-1}\,\mathbf{X} = \frac{\mathbf{I}}{\mathbf{I} + \alpha(\mathbf{I} - \tilde{\mathbf{A}})}\,\mathbf{X} = \frac{\mathbf{I}}{(1+\alpha)\,\mathbf{I} - \alpha\,\tilde{\mathbf{A}}}\,\mathbf{X}, \tag{30}$$

which is equivalent to the form of Eq. (29), i.e., $\mathbf{P} = \mathbf{I}$ and $\mathbf{Q} = (1+\alpha)\,\mathbf{I} - \alpha\,\tilde{\mathbf{A}}$.

(2) Personalized PageRank (PPNP) [26] can obtain node's representation via teleport (restart) probability α which is the ratio of keeping the original representation $\mathbf{X}$, i.e., no propagation. (1-α) is the ratio of performing the normal label propagation:

$$\mathbf{Z} = \alpha\left(\mathbf{I} - (1-\alpha)\,\tilde{\mathbf{A}}\right)^{-1}\mathbf{X} = \frac{\alpha}{\mathbf{I} - (1-\alpha)\,\tilde{\mathbf{A}}}\,\mathbf{X}, \tag{31}$$

where $\tilde{\mathbf{A}} = \mathbf{D}^{-1}\,\mathbf{A}$ is random walk normalized adjacency matrix with self loop. Eq. (44) is with a rational function whose numerator is a constant.

(3) ARMA filter [6] utilize ARMA filter for approximating any desired filter response function, which can be written in the spatial domain as:

$$\mathbf{Z} = \frac{b}{1 - a\,\tilde{\mathbf{A}}}\,\mathbf{X}. \tag{32}$$

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

(4) RationalNet [10] proposes a general rational function and optimized by Remez algorithm, and the analytic form is exactly Eq. (29)

(5) CayleyNets [29] applies rational filtering function in the complex domain.

Remark: The computational cost of [A3] is expensive since it involves the inverse of matrix. Typical solution is to apply iterative algorithms [6; 26; 29].

4.4 Connection among spatial methods

Three groups of spatial methods introduced above (i.e., A1, A2, A3) are strongly connected under *generalization* and *specialization* relationship, as shown in Fig. (1): **(1) Generalization:** *Local Aggregation* can be extended to *Order of Connectivity* by adding more neighbors of higher order. *Order of Connectivity* can be upgraded to *Propagation Direction* by adding reverse propagation; **(2) Specialization:** *Local Aggregation* is a special case of *Order of Connectivity* when setting the the order to 1. *Order of Connectivity* is a special case of *Propagation Direction* if removing reverse propagation.

5 Spectral-based GNNs (B0)

Spectral-based GNN models are built on spectral graph theory which applies eigen-decomposition and analyzes the weight-adjusting function (i.e., filter function) on eigenvalues of graph matrices. The weights yielded by filter function are assigned to frequency components (eigenvectors) for reconstructing the target signal. Based on spectral operation, we propose a new taxonomy of graph neural networks, categorizing spectral-based GNNs into three subgroups:

5.1 Frequency Aggregation (B1)

There exist numerous works that can be boiled down to adjusting weights of frequency components in the spectral domain. The goal of filter function is to adjust eigenvalues (i.e., the weights of eigenvectors) to fit the target output. Many of them are proving to be low-pass filters [31], which means that only low-frequency components are emphasized, i.e., the first few eigenvalues are enlarged, and the others are reduced. There exist a large number of works that can be understood as adjusting weights of frequency component during aggregation. Specifically, a linear function of $\mathbf{g}$ is employed:

$$\mathbf{Z} = (\sum_{i=0}^{l} \theta_i \lambda_i\, \mathbf{u}_i\, \mathbf{u}^\intercal{}_i)\,\mathbf{X} = \mathbf{U}\,\mathbf{g}_\theta(\mathbf{\Lambda})\,\mathbf{U}^\intercal\,\mathbf{X}, \tag{33}$$

where $\mathbf{u}_i$ is the i-th eigenvector, $\mathbf{g}$ is *frequency filter function* controlled by parameters θ, and selected l lowest frequency components. The goal of $\mathbf{g}$ is to change the weights of eigenvalues to fit the target output. Several state-of-the-art methods introduced in the last section are analyzed to illustrate this scheme:

(1) Graph Convolutional Network [25] can be rewritten in spectral domain as:

$$\mathbf{Z} = \tilde{\mathbf{A}}\,\mathbf{X} = (\mathbf{I} - \tilde{\mathbf{L}})\,\mathbf{X} = \mathbf{U}(1 - \mathbf{\Lambda})\,\mathbf{U}^\intercal\,\mathbf{X} \tag{34}$$

Therefore, the frequency response function is $\mathbf{g}(\mathbf{\Lambda}) = 1 - \mathbf{\Lambda}$ which is a low-pass filter, i.e.,smaller eigenvalue will be adjusted to a large value, in which small eigenvalue corresponds to low frequency component.

(2) GraphSAGE [19] can be written in matrix form:

$$\mathbf{Z} = \hat{\mathbf{D}}^{-1}(\mathbf{I} + \mathbf{A})\,\mathbf{X} = (2\,\mathbf{I} - \tilde{\mathbf{L}})\,\mathbf{X} = \mathbf{U}(2 - \mathbf{\Lambda})\,\mathbf{U}^\intercal\,\mathbf{X}, \tag{35}$$

so the frequency response function is $\mathbf{g}(\mathbf{\Lambda}) = 2 - \mathbf{\Lambda}$.

(3) Graph Isomorphism Network (GIN) [53] can be rewritten as:

$$\mathbf{Z} = [(1+\epsilon)\,\mathbf{I} + \mathbf{A}]\,\mathbf{X} = \mathbf{U}[(1+\epsilon) + \mathbf{\Lambda}]\,\mathbf{U}^\intercal\,\mathbf{X}. \tag{36}$$

GIN can be seen as a generalization of GCN or GraphSAGE without normalized adjacency matrix $\mathbf{A}$. The frequency response function is $\mathbf{g}(\mathbf{\Lambda}) = 1 + \epsilon + \mathbf{\Lambda}$

5.2 Order of Approximation (B2)

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:

$$\mathbf{Z} = (\sum_{i=0}^{l}\sum_{j=0}^{k} \theta_j \lambda_i^j\, \mathbf{u}_i\, \mathbf{u}^\intercal{}_i)\,\mathbf{X} = \mathbf{U}\,\mathbf{P}_\theta(\mathbf{\Lambda})\,\mathbf{U}^\intercal\,\mathbf{X}, \tag{37}$$

where $\mathbf{g}(\cdot) = \mathbf{P}_\theta(\cdot)$ is a polynomial function.

(1) ChebNet [21] can be rewritten as:

$$\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}})\,\mathbf{X} = (\tilde{\theta_0}\,\mathbf{I} + \tilde{\theta_1}\tilde{\mathbf{L}} + \tilde{\theta_2}\tilde{\mathbf{L}}^2 + \ldots)\,\mathbf{X}, \tag{38}$$

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

$$\sum_{k=0}^{K-1} \theta_k T_k(\tilde{\mathbf{L}})\,\mathbf{X} = \mathbf{U}(\tilde{\theta_0}\cdot 1 + \tilde{\theta_1}\,\mathbf{\Lambda} + \tilde{\theta_2}\mathbf{\Lambda}^2 + \ldots)\,\mathbf{U}^\intercal\,\mathbf{X}, \quad (39)$$

where $\mathbf{g}(\mathbf{\Lambda}) = \tilde{\theta_0}\cdot 1 + \tilde{\theta_1}\,\mathbf{\Lambda} + \tilde{\theta_2}\mathbf{\Lambda}^2 + \ldots = \mathbf{P}(\mathbf{\Lambda})$.

(2) DeepWalk [41] updates representation as:

$$\begin{aligned}\mathbf{Z} &= \frac{1}{t+1}(\mathbf{I} + \tilde{\mathbf{A}} + \tilde{\mathbf{A}}^2 + \ldots + \tilde{\mathbf{A}}^t)\,\mathbf{X}\\ &= \frac{1}{t+1}(\mathbf{I} + (\mathbf{I}-\tilde{\mathbf{L}}) + (\mathbf{I}-\tilde{\mathbf{L}})^2 + \ldots + (\mathbf{I}-\tilde{\mathbf{L}})^t)\,\mathbf{X}\\ &= (\theta_0\,\mathbf{I} + \theta_1\,\tilde{\mathbf{L}} + \theta_2\,\tilde{\mathbf{L}}^2 + \ldots + \theta_t\,\tilde{\mathbf{L}}^t)\,\mathbf{X}\\ &= \mathbf{U}(\theta_0 + \theta_1\,\mathbf{\Lambda} + \theta_2\,\mathbf{\Lambda}^2 + \ldots + \theta_t\,\mathbf{\Lambda}^t)\,\mathbf{U}^\intercal\,\mathbf{X},\end{aligned}$$

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

(3) Diffusion convolutional neural networks (DCNN) [2] can be rewritten as:

$$\mathbf{Z} = \mathbf{P}(\tilde{\mathbf{A}})\,\mathbf{X} = \mathbf{P}(\mathbf{I}-\tilde{\mathbf{L}})\,\mathbf{X} = \mathbf{U}\,\mathbf{P}(\mathbf{\Lambda})\,\mathbf{U}^\intercal\,\mathbf{X}, \quad (40)$$

(4) Node2Vec [17] can be transformed and reorganized after substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$ as:

$$\mathbf{Z} = \mathbf{U}[(1+\frac{1}{p}) - (1+\frac{1}{q})\,\mathbf{\Lambda} + \frac{1}{q}\,\mathbf{\Lambda}^2]\,\mathbf{U}^\intercal\,\mathbf{X}\,. \quad (41)$$

Therefore, Node2Vec's frequency response function, $\mathbf{g}(\mathbf{\Lambda}) = (1+\frac{1}{p}) - (1+\frac{1}{q})\,\mathbf{\Lambda} + \frac{1}{q}\,\mathbf{\Lambda}^2$, is a second order function of $\mathbf{\Lambda}$ with predefined parameters, i.e., p and q.

(5) Simple Graph Convolution (SGC) [48] can be easily rewritten after substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$ as:

$$\begin{aligned}\mathbf{Z} &= [\binom{K}{0}\mathbf{I} + \binom{K}{1}\mathbf{A}^1 + \binom{K}{2}\mathbf{A}^2 + \cdots + \mathbf{A}^n]\,\mathbf{X}\\ &= \mathbf{U}[\binom{K}{0} + \binom{K}{1}\mathbf{\Lambda}^1 + \binom{K}{2}\mathbf{\Lambda}^2 + \cdots + \mathbf{\Lambda}^n]\,\mathbf{U}^\intercal\,\mathbf{X},\end{aligned}$$

where $\mathbf{g}(\mathbf{\Lambda})$ is a polynomial function of $\mathbf{\Lambda}$.

5.3 Approximation Type (B3)

Although polynomial approximation is widely used and empirically effective, it only works when applying on a smooth signal in the spectral domain. However, there is no guarantee that any real world signal is smooth. Therefore, the rational approximation is introduced to improve the accuracy of non-smooth signal modelling. Rational kernel based method can be written as:

$$\mathbf{Z} = (\sum_i^l \frac{\sum_{j=0}^{k}\theta_j\lambda_i^j}{\sum_{m=1}^{n}\phi_m\lambda_i^m + 1}\,\mathbf{u}_i\,\mathbf{u}^\intercal{}_i)\,\mathbf{X} = \mathbf{U}\,\frac{\mathbf{P}_\theta(\mathbf{\Lambda})}{\mathbf{Q}_\phi(\mathbf{\Lambda})}\,\mathbf{U}^\intercal\,\mathbf{X}, \quad (42)$$

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.

(1) Auto-Regressive filter [59; 57; 5] can be rewritten as:

$$\mathbf{Z} = (\mathbf{I} + \alpha\,\tilde{\mathbf{L}})^{-1}\,\mathbf{X} = \mathbf{U}\,\frac{1}{1+\alpha(1-\mathbf{\Lambda})}\,\mathbf{U}^\intercal\,\mathbf{X}, \quad (43)$$

where $\mathbf{g}(\mathbf{\Lambda}) = \frac{1}{1+\alpha(1-\mathbf{\Lambda})}$

(2) Personalized PageRank (PPNP) [26] can be rewritten in the spectral domain with substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$ as:

$$\mathbf{Z} = \frac{\alpha}{\mathbf{I} - (1-\alpha)(\mathbf{I}-\tilde{\mathbf{L}})}\,\mathbf{X} = \mathbf{U}\,\frac{\alpha}{\alpha + (1-\alpha)\,\mathbf{\Lambda}}\,\mathbf{U}^\intercal\,\mathbf{X}, \quad (44)$$

where $\mathbf{g}(\mathbf{\Lambda}) = \frac{\alpha}{1-(1-\alpha)\,\mathbf{\Lambda}}$

(3) ARMA filter [6] can be rewritten in the spectral domain with substituting $\tilde{\mathbf{A}} = \mathbf{I} - \tilde{\mathbf{L}}$ as:

$$\mathbf{Z} = \frac{b}{\mathbf{I} - a(\mathbf{I}-\tilde{\mathbf{L}})}\,\mathbf{X} = \mathbf{U}\,\frac{b}{1 - a + a\,\mathbf{\Lambda}}\,\mathbf{U}^\intercal\,\mathbf{X}\,. \quad (45)$$

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

5.4 Connection among spectral methods

There is a strong-tie among the above-mentioned three groups of spectral methods in the perspective of *generalization* and *specialization*, as shown in Fig. (1): **(1) Generalization:** *Frequency Aggregation* can be extended to *Order of Connectivity* by adding more higher order of eigenvalues, i.e., 1←k. *Order of Connectivity* can be upgraded to *Approximation Type* if the denominator of filter function is not 1; **(2) Specialization:** *Frequency Aggregation* is a special case of *Order of Connectivity* by setting the highest order to 1. *Order of Connectivity* is a special case of *Approximation Type* by setting the denominator of filter function to 1.

6 Conclusion

In this paper, we propose a unified framework that summarizes the state-of-the-art GNNs, providing a new perspective for understanding GNNs of different mechanisms. By analytically categorizing current GNNs into the spatial and spectral domains and further dividing them into subcategories, our analysis reveals that the subcategories are not only strongly connected by generalization and specialization relations within their domain, but also by equivalence relation across the domains. We demonstrate the generalization power of our proposed framework by reformulating numerous existing GNN models. The above survey of the state-of-the-art graph neural networks, showing that GNNs is still a young research area. Increasing number of emerging GNN models [15; 54; 9; 47; 51; 40] makes the theoretical understanding [33; 39] a urgent need. Therefore, the next-generation GNNs are expected to be more interpretable and transparent to the application [18; 3; 55].

References

[1] S. Abu-El-Haija, B. Perozzi, A. Kapoor, H. Harutyunyan, N. Alipourfard, K. Lerman, G. V. Steeg, and A. Galstyan. Mixhop: Higher-order graph convolution architectures via sparsified neighborhood mixing. *International Conference on Machine Learning*, 2019.

[2] J. Atwood and D. Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001, 2016.

[3] F. Baldassarre and H. Azizpour. Explainability techniques for graph convolutional networks. *arXiv preprint arXiv:1905.13686*, 2019.

[4] M. G. Bell, Y. Iida, et al. Transportation network analysis. 1997.

[5] Y. Bengio, O. Delalleau, and N. Le Roux. 11 label propagation and quadratic criterion. 2006.

[6] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi. Graph neural networks with convolutional arma filters. *CoRR*, 2019.

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

[8] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*, pages http–openreview, 2014.

[9] Y. Chen, L. Wu, and M. J. Zaki. Reinforcement learning based graph-to-sequence model for natural question generation. *arXiv preprint arXiv:1908.04942*, 2019.

[10] Z. Chen, F. Chen, R. Lai, X. Zhang, and C.-T. Lu. Rational neural networks for approximating jump discontinuities of graph convolution operator. *ICDM*, 2018.

[11] F. R. Chung. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[12] E. H. Davidson, J. P. Rast, P. Oliveri, A. Ransick, C. Calestani, C.-H. Yuh, T. Minokawa, G. Amore, V. Hinman, C. Arenas-Mena, et al. A genomic regulatory network for development. *science*, 295(5560):1669–1678, 2002.

[13] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[14] J. H. Drew and H. Liu. Diagnosing fault patterns in telecommunication networks, Sept. 23 2008. US Patent 7,428,300.

[15] F. Errica, M. Podda, D. Bacciu, and A. Micheli. A fair comparison of graph neural networks for graph classification. *arXiv preprint arXiv:1912.09893*, 2019.

[16] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1263–1272. JMLR. org, 2017.

[17] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[18] R. Guo, J. Li, and H. Liu. Learning individual treatment effects from networked observational data. *ACM International Conference on Web Search and Data Mining*, 2020.

[19] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

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

[21] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[22] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.

[23] E. Isufi, A. Loukas, A. Simonetto, and G. Leus. Autoregressive moving average graph filtering. *IEEE Transactions on Signal Processing*, 65(2):274–288, Jan 2017.

[24] R. Johnson and T. Zhang. On the effectiveness of laplacian normalization for graph semi-supervised learning. *Journal of Machine Learning Research*, 8(Jul):1489–1517, 2007.

[25] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017.

[26] J. Klicpera, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. 2018.

[27] D. Lazer, A. S. Pentland, L. Adamic, S. Aral, A. L. Barabasi, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, et al. Life in the network: the coming age of computational social science. *Science (New York, NY)*, 323(5915):721, 2009.

[28] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[29] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[30] Q. Li, Z. Han, and X.-M. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[31] Q. Li, X.-M. Wu, H. Liu, X. Zhang, and Z. Guan. Label efficient semi-supervised learning via graph filtering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[32] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, volume 15, pages 2181–2187, 2015.

[33] A. Loukas. What graph neural networks cannot learn: depth vs width. *arXiv preprint arXiv:1907.03199*, 2019.

[34] A. Loukas, A. Simonetto, and G. Leus. Distributed autoregressive moving average graph filters. *IEEE Signal Processing Letters*, 22(11):1931–1935, Nov 2015.

[35] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[36] V. Marx. High-throughput anatomy: charting the brain's networks. *Nature*, 490(7419):293, 2012.

[37] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[38] M. E. Newman. Spread of epidemic disease on networks. *Physical review E*, 66(1):016128, 2002.

[39] K. Oono and T. Suzuki. Graph neural networks exponentially lose expressive power for node classification.

[40] C. W. Park and C. Wolverton. Developing an improved crystal graph convolutional neural network framework for accelerated materials discovery, 2019.

[41] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[42] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[43] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[44] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[45] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

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

[47] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*, 2019.

[48] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. In *International Conference on Machine Learning*, pages 6861–6871, 2019.

[49] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[50] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

[51] T. Xie and J. C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.*, 120:145301, Apr 2018.

[52] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

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

[54] F. Yang, Z. Yang, and W. W. Cohen. Differentiable learning of logical rules for knowledge base reasoning. In *Advances in Neural Information Processing Systems*, pages 2319–2328, 2017.

[55] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec. Gnn explainer: A tool for post-hoc explanation of graph neural networks. 2019.

[56] Z. Zhang, P. Cui, and W. Zhu. Deep learning on graphs: A survey. *arXiv preprint arXiv:1812.04202*, 2018.

[57] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328, 2004.

[58] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[59] X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[60] X. Zhu and M. Rabbat. Approximating signals supported on graphs. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 3921–3924. IEEE, 2012.