Spectral Graph Neural Networks

Marius Graml [1]

Abstract

Spectral Graph Neural Networks (GNNs) utilize spectral graph theory to effectively process graph-structured data. This paper focuses on eigenvalue-based spectral GNNs, specifically those employing trainable polynomial filters, and introduces key models within this category. We compare various model types and architectures, analyzing their efficiency and complexity while emphasizing the trade-offs in this context. Furthermore, we explore potential reasons why ChebNet, despite its theoretically superior expressivity, is often outperformed by other models. Finally, we outline promising directions for future research in the development and optimization of spectral GNNs.

1. Introduction

Graph Neural Networks (GNNs) have gained attention for their success in tasks such as node classification (Kipf & Welling, 2017; Veličković et al., 2018), link prediction (Zhang & Chen, 2018), and graph classification (Xu et al., 2019b). GNNs are generally categorized into spatial-based and spectral-based approaches, with spectral GNNs leveraging Graph Signal Processing (GSP) to apply filters in the graph frequency domain (Bo et al., 2023b). This paper focuses on eigenvalue-based spectral GNNs, particularly those employing trainable polynomial filters. First introduced by (Bruna et al., 2014), these methods have been further refined through models like ChebNet (Defferrard et al., 2017), GPR-GNN (Chien et al., 2021), BernNet (He et al., 2022), and ChebNetII (He et al., 2024). We present these models and analyze performance and computational complexity for different architectures. Additionally, we investigate why ChebNet, despite its theoretical expressivity, often exhibits inferior performance in many frameworks. Finally, we point to the potential of eigenvector-based methods as a promising direction for future spectral GNN research.

[1]Department of Computer Science, Technical University Munich, Munich, Germany. Correspondence to: Marius Graml <marius.graml@tum.de>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

2. Preliminaries

We consider an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with a node set $\mathcal{V}$ and a set of edges $\mathcal{E}$. Let $N = |\mathcal{V}|$ and $E = |\mathcal{E}|$ denote the number of nodes and edges of the graph. We use $\boldsymbol{x} \in \mathbb{R}^N$ to represent the graph signal, where $\boldsymbol{x}(i)$ indicates the signal value at node i. In the general case of Graph Neural Networks (GNNs), where the input is a feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, each column of $\boldsymbol{X}$ can be treated as a separate graph signal, and each row as the feature vector of a node i. Let $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ denote the unweighted adjacency matrix, and $\boldsymbol{S} \in \mathbb{R}^{N \times N}$ the weighted adjacency matrix, both encoding connections between pairs of vertices. Furthermore, let $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ be the degree matrix, where $D_{ii} = \sum_{j=1}^{N} A_{ij}$. We define the normalized weighted adjacency matrix as $\underline{\boldsymbol{S}} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{S} \boldsymbol{D}^{-\frac{1}{2}} \in \mathbb{R}^{N \times N}$, and the normalized weighted adjacency matrix with added self - loops as $\hat{\underline{\boldsymbol{S}}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{S}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \in \mathbb{R}^{N \times N}$ where $\tilde{\boldsymbol{S}} = \boldsymbol{S} + \boldsymbol{I}_N$ and $\tilde{\boldsymbol{D}} = \boldsymbol{D} + \boldsymbol{I}_N$, with $\boldsymbol{I}_N$ denoting the identity matrix.

An essential operator in spectral graph analysis is the graph Laplacian. Its combinatorial form is defined as $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{S} \in \mathbb{R}^{N \times N}$, and its symmetric normalized form is $\underline{\boldsymbol{L}} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{L} \boldsymbol{D}^{-\frac{1}{2}} = \boldsymbol{I}_N - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{S} \boldsymbol{D}^{-\frac{1}{2}} \in \mathbb{R}^{N \times N}$. The graph Laplacian can be seen as a discrete analog of the Laplace operator on graphs, providing a measure of the difference between the signal values at a node i and its neighbors $j \in \mathcal{N}(i, 1)$ (Shuman et al., 2013), where $\mathcal{N}(i, K)$ denotes the set of K-hop neighbors of node i.

Since $\boldsymbol{L}$ is a real symmetric positive semidefinite matrix, it has a complete set of orthonormal eigenvectors $\{\boldsymbol{v}_l\}_{l=1}^{N} \in \mathbb{R}^N$, known as the graph Fourier modes, spanning the frequency domain of the graph. These eigenvectors have associated ordered, nonnegative eigenvalues $\{\lambda_l\}_{l=1}^{N}$, which correspond to the graph's frequencies. The eigendecomposition of $\boldsymbol{L}$ is given by $\boldsymbol{L} = \boldsymbol{V} \boldsymbol{\Lambda} \boldsymbol{V}^T$, where $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N] \in \mathbb{R}^{N \times N}$ is the matrix of eigenvectors and $\boldsymbol{\Lambda} = \text{diag}([\lambda_1, \ldots, \lambda_N]) \in \mathbb{R}^{N \times N}$ is the diagonal matrix of eigenvalues, where $\lambda_i \in [0, 2]\ \forall i = 1, ..., N$ (von Luxburg, 2007; Bo et al., 2023a;b).

The graph Fourier transform (GFT) of a signal $\boldsymbol{x} \in \mathbb{R}^N$ is defined as $\hat{\boldsymbol{x}} = \boldsymbol{V}^T \boldsymbol{x} \in \mathbb{R}^N$, with the inverse transform given by $\boldsymbol{x} = \boldsymbol{V} \hat{\boldsymbol{x}} = \sum_{l=1}^{N} \hat{\boldsymbol{x}}(\lambda_l) \boldsymbol{v}_l$. This generalizes the classical Fourier transform (FT) from time-domain signals or images to graph structures (Shuman et al., 2013). Using the GFT, we can perform filtering on graph signals

analogously to classical signal processing, where a filter modifies a signal in the frequency domain. In general, each spectral graph filter is represented by a diagonal matrix, where each diagonal element corresponds to a scaling factor for the respective frequency component. Thus, spectral filtering can be expressed as $\hat{\boldsymbol{y}}(\boldsymbol{\lambda}) = \hat{g}(\boldsymbol{\lambda})\hat{\boldsymbol{x}}(\boldsymbol{\lambda})$, where $\hat{g}(\boldsymbol{\lambda}) = \mathrm{diag}([g(\lambda_1), \dots, g(\lambda_N)]) \in \mathbb{R}^{N\times N}$ represents the spectral filter and $\hat{\boldsymbol{y}} \in \mathbb{R}^N$ the filtered graph signal in the frequency domain. Applying the inverse GFT provides the filtering equation in the vertex domain as $\boldsymbol{y} = \boldsymbol{V}\hat{g}(\boldsymbol{\lambda})\boldsymbol{V}^T\boldsymbol{x} = \sum_{l=1}^{N} \hat{g}(\lambda_l)\boldsymbol{v}_l\boldsymbol{v}_l^T\boldsymbol{x} = (\boldsymbol{g} * \boldsymbol{x})$ representing a convolution between the graph signal and the filter function (Shuman et al., 2013).

3. Spectral GNNs

The primary objective of eigenvalue-based spectral GNNs is to design a filter function that effectively extracts meaningful information from graph structures and their associated signals. Filters are generally classified into three categories: advanced filters, polynomial filters, and linear filters (Bo et al., 2023b). Polynomial filter approaches can be further divided into fixed filter, variable (trainable) filter, and filter bank GNNs (Liao et al., 2024). Due to the scalability issues of advanced filters, which depend on time-intensive matrix factorization (Bo et al., 2023b), polynomial filters have garnered significant attention in recent years for their ability to efficiently approximate optimal filter functions. The polynomial approach is rooted in classical polynomial approximation theory and can be motivated by the Weierstraß Approximation Theorem, which states that any continuous function defined on a compact interval can be approximated arbitrarily closely by a polynomial function (Hammond et al., 2009; Mason & Handscomb, 2002). Thus, polynomial filters typically use a predetermined polynomial basis and learn the coefficients from the training data. However, in practice, the polynomial degree K cannot be chosen arbitrarily large due to computational efficiency and numerical stability concerns. Consequently, the main challenge lies in identifying an effective parameterization for the polynomial filter that approximates the optimal filter well, while remaining efficient and numerical stable to train.

3.1. Polynomial Filter

Approximating a filter function in the frequency domain using a monomial basis of degree K yields the general polynomial filter of order K with θ_k being the learnable filter weights and $\boldsymbol{\Lambda} = diag([\lambda_1, ..., \lambda_N])$ the diagonal matrix of eigenvalues of the graph Laplacian:

$$\hat{g}_\theta(\boldsymbol{\lambda}) = \sum_{k=0}^{\infty} \theta_k \boldsymbol{\Lambda}^k \approx \sum_{k=0}^{K} \theta_k \boldsymbol{\Lambda}^k \tag{1}$$

Transforming the filter function from the spectral domain to the vertex domain, the filter function is approximated by a polynomial of order K, expressed in terms of the monomial basis applied to the graph Laplacian (see A.1, Eq. 10):

$$\begin{aligned} \boldsymbol{y} = \boldsymbol{V}\hat{g}_\theta(\boldsymbol{\lambda})\boldsymbol{V}^T\boldsymbol{x} &= \sum_{k=0}^{\infty} \theta_k \boldsymbol{L}^k\boldsymbol{x} = g_\theta(\boldsymbol{L})\boldsymbol{x} \\ &\approx \sum_{k=0}^{K} \theta_k \boldsymbol{L}^k\boldsymbol{x} = g_\theta(\boldsymbol{L}^K)\boldsymbol{x} = \boldsymbol{Q}\boldsymbol{\theta} \end{aligned} \tag{2}$$

where $g_\theta(\boldsymbol{L}^K)$ denotes the polynomial function up to degree K. For a single element of the graph signal, this can be written as (see A.1, Eq. 11)

$$\boldsymbol{y}(i) = b_{i,i}\boldsymbol{x}(i) + \sum_{j\in\mathcal{N}(i,K)} b_{i,j}\boldsymbol{x}(j). \tag{3}$$

This shows that the polynomial filter is a K-localized filter, i.e., it generates the new signal value at node i by only considering the K-hop neighbors $\mathcal{N}(i, K)$. Thus, the polynomial filter can be viewed as an extension of a classical CNN kernel to graphs, with a receptive field of K and learnable weights $\boldsymbol{b} \in \mathbb{R}^{K+1}$ or $\boldsymbol{\theta} \in \mathbb{R}^{K+1}$, respectively. Importantly, the filter function in the vertex domain is not restricted to the monomial basis (as in Eq. 2). Instead, any polynomial basis applied to the graph Laplacian can be utilized (see 3.3). Each choice of basis comes with specific trade-offs in terms of numerical stability, approximation accuracy, and computational complexity.

3.2. Complexity

The closed-form expression of the polynomial filter in the vertex domain eliminates the need to transform the graph signal into the frequency domain and back. This transformation would involve a complexity of $\mathcal{O}(N^2)$ due to the double multiplication with the eigenvector matrix $\boldsymbol{V}$, combined with the computationally expensive eigendecomposition of the Laplacian $\boldsymbol{L}$ for large graphs, which scales as $\mathcal{O}(N^3)$ (Bo et al., 2023b; Chen et al., 2024; Susnjara et al., 2015). Instead, the output signal can be computed directly in the vertex domain with linear complexity in E and independent of N. In particular, for many prominent spectral GNN layers such as ChebNet (Defferrard et al., 2017), GCN (Kipf & Welling, 2017), or GPR-GNN (Chien et al., 2021) the computation achieves a complexity of $\mathcal{O}(KE) \ll \mathcal{O}(N^2)$. This efficiency arises from the ability to recursively compute the powers of the graph Laplacian with a complexity of $\mathcal{O}(KE)$ (Defferrard et al., 2017; Hammond et al., 2009), as

$$\boldsymbol{q}_{k+1} = \boldsymbol{L}^{k+1}\boldsymbol{x} = \boldsymbol{L}(\boldsymbol{L}^k\boldsymbol{x}) = \boldsymbol{L}\boldsymbol{q}_k. \tag{4}$$

The final output signal can then be computed with a complexity of $\mathcal{O}(KN)$, expressed as

$$\boldsymbol{Q}\boldsymbol{\theta} = [\boldsymbol{q}_1 ... \boldsymbol{q}_K]\boldsymbol{\theta}. \tag{5}$$

For many real world graphs, where $E = dN$ with $d > 1$ (Defferrard et al., 2017; Susnjara et al., 2015), $\mathcal{O}(KE)$ dominates $\mathcal{O}(KN)$. When replacing the graph signal $\boldsymbol{x}$ with a feature matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, the computational complexity simply extends to $\mathcal{O}(KED)$ (He et al., 2024; Liao et al., 2024).

3.3. Selected Spectral GNNs

Polynomial-based spectral GNNs comprise various models, primarily distinguished by the choice of polynomial basis for defining the filter function. These variations affect the resulting polynomial filter, offering different trade-offs in numerical stability, approximation accuracy, and computational complexity. In general, we distinguish between iterative and decoupled architectures. In the former, feature transformation is applied after each K-hop message passing step, whereas in the latter, feature transformation is only applied before or after all message passing steps (Liao et al., 2024). In the following, GNNs with trainable filters are introduced, all supporting an iterative and decoupled architecture.

ChebNet. (Defferrard et al., 2017) introduced a spectral GNN that approximates the filter function using Chebyshev polynomials $T_k(x)$ of order k. The main motivation for Chebyshev polynomials lies in their strong mathematical properties. The truncated Chebyshev expansion provides a near-minimax polynomial over the interval $[-1, 1]$, i.e., it closely approximates any arbitrary filter function (Hammond et al., 2009). Additionally, the recurrence relation of Chebyshev polynomials supports efficient recursive computation, preserving the advantageous computational complexity discussed in 3.2. Furthermore, since $T_k(x) \in [-1, 1]\ \forall k$, using Chebyshev polynomials ensures numerical stability (Susnjara et al., 2015) by avoiding the amplification effects associated with the graph Laplacian's eigenvalues in $[0, 2]$ (Kipf & Welling, 2017). With c_k as the learnable Chebyshev coefficients, the filter is defined as

$$\boldsymbol{y} = \sum_{k=0}^{K} c_k T_k(\tilde{\boldsymbol{L}})\boldsymbol{x} = g_c(\tilde{\boldsymbol{L}}^K)\boldsymbol{x}, \tag{6}$$

where the Chebyshev polynomial of order k can recursively be determined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$ and the graph Laplacian is mapped to the interval $[-1, 1]$ via $\tilde{\boldsymbol{L}} = \frac{2\boldsymbol{L}}{\lambda_{max}} - \boldsymbol{I}_n$.

GPR-GNN. The GPR-GNN architecture (Chien et al., 2021) defines the polynomial filter using a monomial basis applied to the normalized adjacency matrix with self-loops $\hat{\underline{\boldsymbol{S}}}$ instead of the graph Laplacian. This design choice can be motivated by the renormalization trick introduced in (Kipf & Welling, 2017), which helps mitigate numerical instabilities. The filter operation is formally defined as:

$$\boldsymbol{y} = \sum_{k=0}^{K} \theta_k \hat{\underline{\boldsymbol{S}}}^k \boldsymbol{x} = g_\theta(\hat{\underline{\boldsymbol{S}}}^K)\boldsymbol{x} \tag{7}$$

Since Generalized PageRank (GPR) is defined as $\boldsymbol{r} = \sum_{k=0}^{\infty} \theta_k \hat{\underline{\boldsymbol{S}}}^k \boldsymbol{\pi}$, the output graph signal $\boldsymbol{y}$ of the polynomial filter $g_\theta(\hat{\underline{\boldsymbol{S}}}^K)$ corresponds to the GPR vector after K iterations, with each weight θ_k representing the teleport probability for step k. Unlike setups based on Personalized PageRank (PPR), such as APPNP (Gasteiger et al., 2022), which restrict filter weights to fixed, non-negative values, this approach allows each θ_k to take any value (see A.2). Thus, while such PPR-based approaches lead to low-pass filters that suppress high-frequency signals, potentially leading to information loss in heterophilic graphs (Chien et al., 2021), the GPR-GNN approach offers greater flexibility, effectively capturing the spectral properties of both homophilic and heterophilic graphs. Furthermore, GPR-GNN's architecture avoids oversmoothing. This is achieved by setting the weights θ_k to zero for sufficiently large k whose corresponding matrix $\hat{\underline{\boldsymbol{S}}}^k$ would lead to significant oversmoothing effects (Chien et al., 2021).

BernNet. BernNet (He et al., 2022) leverages the Bernstein basis applied to the graph Laplacian to approximate the filter function. The polynomial filter in the vertex domain is defined as:

$$\boldsymbol{y} = \sum_{k=0}^{K} \theta_k b_k^K(\boldsymbol{L}) = \sum_{k=0}^{K} \theta_k \frac{1}{2^K} \binom{K}{k} (2\boldsymbol{I} - \boldsymbol{L})^{K-k} \boldsymbol{L}^k \boldsymbol{x} \tag{8}$$

where θ_k are the learnable filter weights and $b_k^K(\boldsymbol{L})$ represents the k-th Bernstein polynomial.
A key strength of BernNet is the interpretability of its designable filters. Moreover, the authors demonstrate that any valid filter (i.e., those with non-negative spectral responses) can be expressed using the Bernstein basis (He et al., 2022).

ChebNetII. In ChebNetII (He et al., 2024), Chebyshev interpolation is proposed as an alternative to Chebyshev expansion for approximating the filter function leading to a filter defined by:

$$\boldsymbol{y} = \sum_{k=0}^{K} w_k T_k(\tilde{\boldsymbol{L}})\boldsymbol{x} = \frac{2}{K+1} \sum_{k=0}^{K} \sum_{j=0}^{K} \gamma_j T_k(x_j) T_k(\tilde{\boldsymbol{L}})\boldsymbol{x} \tag{9}$$

where $w_k = \frac{2}{K+1} \sum_{j=0}^{K} \gamma_j T_k(x_j)$ denote the Chebyshev coefficients and $x_j = cos(\frac{j+\frac{1}{2}}{K+1})\pi$ for $j = 0, 1, ..., K$ represent the Chebyshev nodes of the polynomial $T_{K+1}(x)$. The learnable parameters $\gamma_j = h(x_j)$ for $j = 0, 1, ..., K$ represent the values of the filter function at these nodes. Thus, unlike polynomial approximation methods, ChebNetII directly approximates the filter function.

Chebyshev interpolation offers significant advantages over other polynomial interpolation methods and spectral GNNs. Its key benefit lies in its ability to minimize the Runge phenomenon, which is achieved through the optimal placement of the Chebyshev nodes (He et al., 2024). Due to this special property, Chebyshev interpolation provides a near-minimax approximation of the filter function, resulting in lower approximation errors than other polynomial interpolation techniques, and achieves a theoretically faster convergence rate compared to methods such as BernNet (He et al., 2024).

4. Review

Performance. The examined models generally perform well on both homophilic and heterophilic graphs (Liao et al., 2024; Wang & Zhang, 2022). While they do not show a clear advantage over simpler fixed filter models like APPNP on homophilic graphs, they excel in heterophilic settings, where fixed filters often struggle due to their limitation to low-pass filtering. Additionally, advanced filter models tend to outperform such polynomial-based approaches (Bo et al., 2023b), likely because they are not constrained to approximating the filter function through polynomial approximations. Furthermore, by comparing the experimental results presented in (He et al., 2022; 2024) and (Liao et al., 2024), we hypothesize that a decoupled architecture may offer significant advantages over an iterative architecture in improving the performance of spectral GNNs, particularly for ChebNet. In the former two frameworks, ChebNet employed an iterative architecture, whereas a decoupled architecture was applied to all other models. In this settings, ChebNet was outperformed by GCN, GPR-GNN, BernNet, and ChebNetII. In contrast, the latter framework exclusively utilized decoupled architectures. Here, ChebNet demonstrated comparable or superior performance, which is intuitive from an approximation theory perspective, as the monomial and Bernstein bases do not provide a near-minimax approximation and GCN is essentially a simplified ChebNet with $K = 1$. This finding challenges the assumption made in (He et al., 2024), which attributes ChebNet's lower performance to unconstrained stochastic gradient descent (SGD) optimization of its Chebyshev coefficients. The argument suggests that such unconstrained optimization introduces higher-order terms corresponding to high-frequency components, leading to overfitting by capturing excessive details in the training data or approximating non-analytical filter functions. However, in addition to the contradicting experimental results, this assumption lacks formal mathematical justification.

Complexity. While ChebNet and GPR-GNN exhibit computational complexity as outlined in 3.2, BernNet demonstrates a higher computational complexity of $\mathcal{O}(K^2ED)$ (He et al., 2022; Wang & Zhang, 2022; He et al., 2024). This increased complexity results in longer training times per epoch and overall, particularly for large graphs (Liao et al., 2024; Wang & Zhang, 2022; He et al., 2024). Furthermore, as BernNet uses multiple diffusion matrices for each message propagation step, it requires additional memory, resulting in higher RAM and GPU memory usage during training (Liao et al., 2024). The complexity of ChebNetII is generally given by $\mathcal{O}(K^2 + KED)$, where the additional term K^2 arises from the computation of the components $T_k(x_j)$ of the coefficients w_k (He et al., 2024). Since these components are theoretically precomputable, the associated complexity can be shifted either to the precomputation phase or to the training phase. Experiments reported in (Liao et al., 2024; He et al., 2024) evaluate a decoupled setting in which the components $T_k(x_j)$ are not precomputed, while the terms $T_k(\tilde{\boldsymbol{L}})\boldsymbol{x}$ are precomputed. Under this configuration, the precomputation time for ChebNetII is comparable to that of ChebNet and GPR-GNN, yet significantly lower than that of BernNet. On the other hand, the training time for ChebNetII is noticeably higher than that of the other models, including BernNet.

Trade-off. In conclusion, compared to fixed-parameter models, the trainable approaches are less efficient in terms of runtime and memory consumption but achieve overall better performance (Liao et al., 2024). In comparison to advanced filtering approaches, polynomial methods are less effective but more efficient. These observations highlight a trade-off between effectiveness and efficiency, where polynomial approaches with trainable parameters strike a well-balanced compromise between both.

5. Future Research

Spectral GNNs could further benefit from advancements in eigenvector-based methods, which can be viewed as extending advanced filter approaches to more sophisticated frequency-domain bases. One promising direction involves utilizing a wavelet basis, where the eigenvector matrix $\boldsymbol{V}$ is replaced by a wavelet matrix $\boldsymbol{\Psi}_s$. This substitution enables efficient frequency-domain computations by exploiting the sparsity of $\boldsymbol{\Psi}_s$ in real-world graphs. Additionally, this approach inherently supports localized convolutions, as graph wavelets are naturally vertex-localized, unlike the global eigenvectors of the Laplacian (Xu et al., 2019a). Thus, since advanced filters demonstrate superior performance compared to polynomial-based methods but are often impractical due to their high computational complexity (see 4), the wavelet-based approach holds the potential to outperform polynomial filters while retaining similar efficiency and vertex-localization properties.

References

Bo, D., Shi, C., Wang, L., and Liao, R. Specformer: Spectral graph neural networks meet transformers. In *The Eleventh International Conference on Learning Representations*, 2023a. URL https://openreview.net/forum?id=0pdSt3oyJa1.

Bo, D., Wang, X., Liu, Y., Fang, Y., Li, Y., and Shi, C. A survey on spectral graph neural networks, 2023b. URL https://arxiv.org/abs/2302.05631.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs, 2014. URL https://arxiv.org/abs/1312.6203.

Chen, J., Lei, R., and Wei, Z. PolyGCL: GRAPH CONTRASTIVE LEARNING via learnable spectral polynomial filters. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=y21ZO6M86t.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized pagerank graph neural network, 2021. URL https://arxiv.org/abs/2006.07988.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering, 2017. URL https://arxiv.org/abs/1606.09375.

Gasteiger, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank, 2022. URL https://arxiv.org/abs/1810.05997.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory, 2009. URL https://arxiv.org/abs/0912.3848.

He, M., Wei, Z., Huang, Z., and Xu, H. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation, 2022. URL https://arxiv.org/abs/2106.10994.

He, M., Wei, Z., and Wen, J.-R. Convolutional neural networks on graphs with chebyshev approximation, revisited, 2024. URL https://arxiv.org/abs/2202.03580.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks, 2017. URL https://arxiv.org/abs/1609.02907.

Liao, N., Liu, H., Zhu, Z., Luo, S., and Lakshmanan, L. V. S. Benchmarking spectral graph neural networks: A comprehensive study on effectiveness and efficiency, 2024. URL https://arxiv.org/abs/2406.09675.

Mason, J. and Handscomb, D. C. *Chebyshev Polynomials*. Chapman and Hall/CRC, New York, 2002. ISBN 9780849303555. doi: 10.1201/9781420036114.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. 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, May 2013. ISSN 1053-5888. doi: 10.1109/msp.2012.2235192. URL http://dx.doi.org/10.1109/MSP.2012.2235192.

Susnjara, A., Perraudin, N., Kressner, D., and Vandergheynst, P. Accelerated filtering on graphs using lanczos method, 2015. URL https://arxiv.org/abs/1509.04537.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks, 2018. URL https://arxiv.org/abs/1710.10903.

von Luxburg, U. A tutorial on spectral clustering, 2007. URL https://arxiv.org/abs/0711.0189.

Wang, X. and Zhang, M. How powerful are spectral graph neural networks, 2022. URL https://arxiv.org/abs/2205.11172.

Xu, B., Shen, H., Cao, Q., Qiu, Y., and Cheng, X. Graph wavelet neural network. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=H1ewdiR5tQ.

Xu, W., Li, Q., Zhu, Z., and Wu, X. A novel graph wavelet model for brain multi-scale activational-connectional feature fusion. In Shen, D., Liu, T., Peters, T. M., Staib, L. H., Essert, C., Zhou, S., Yap, P.-T., and Khan, A. (eds.), *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, pp. 763–771, Cham, 2019b. Springer International Publishing. ISBN 978-3-030-32248-9.

Zhang, M. and Chen, Y. Link prediction based on graph neural networks, 2018. URL https://arxiv.org/abs/1802.09691.

A. Mathematical Considerations

A.1. Derivation of Filter Operation with Standard Polynomial Filter

The filter operation with a standard polynomial filter is derived as follows:

$$\begin{aligned}\boldsymbol{y} &= \boldsymbol{V} g_\theta(\boldsymbol{\lambda}) \boldsymbol{V}^T \boldsymbol{x} = \boldsymbol{V} \sum_{k=0}^{K} \theta_k \boldsymbol{\Lambda}^k \boldsymbol{V}^T \boldsymbol{x} = \sum_{k=0}^{K} \theta_k \boldsymbol{V} \boldsymbol{\Lambda}^k \boldsymbol{V}^T \boldsymbol{x} \\ &= \sum_{k=0}^{K} \theta_k \boldsymbol{L}^k \boldsymbol{x} = \sum_{k=0}^{K} \theta_k \boldsymbol{q}_k = g_\theta(\boldsymbol{L}^K)\boldsymbol{x} = \boldsymbol{Q}\boldsymbol{\theta}\end{aligned} \tag{10}$$

For a single element $\boldsymbol{y}(i)$, the operation can be expressed as:

$$\begin{aligned}\boldsymbol{y}(i) &= \sum_{k=0}^{K} \theta_k (\boldsymbol{L}^k)_i \boldsymbol{x} = \sum_{k=0}^{K} \sum_{j=1}^{N} \theta_k (\boldsymbol{L}^k)_{i,j} \boldsymbol{x}(j) = \sum_{k=0}^{K} \theta_k (\boldsymbol{L}^k)_{i,i} \boldsymbol{x}(i) + \sum_{j \in \mathcal{N}(i,K)} \sum_{k=0}^{K} \theta_k (\boldsymbol{L}^k)_{i,j} \boldsymbol{x}(j) \\ &= b_{i,i} \boldsymbol{x}(i) + \sum_{j \in \mathcal{N}(i,K)} b_{i,j} \boldsymbol{x}(j)\end{aligned} \tag{11}$$

where $(\boldsymbol{L})_i$ denotes the i-th row of $\boldsymbol{L}$ and $(\boldsymbol{L})_{i,j}$ the element in the i-th row and j-th column.
Furthermore, it applies:

$$\begin{aligned}b_{i,j} &= \sum_{k=0}^{K} \theta_k (\boldsymbol{L}^k)_{i,j} \\ \boldsymbol{Q} &= [\boldsymbol{q}_1 \dots \boldsymbol{q}_K] \in \mathbb{R}^{N \times K}\end{aligned} \tag{12}$$

and, according to Lemma 5.2 in (Hammond et al., 2009), it holds that

$$(\boldsymbol{L}^k)_{i,j} = 0 \text{ if } j \notin \mathcal{N}(i,K)$$

A.2. PageRank Formulation for Fixed and Learnable Parameters

A well-known definition of the Generalized PageRank (GPR) formulation with fixed teleport probabilities is given as:

$$\boldsymbol{r} = \beta \hat{\underline{\boldsymbol{S}}} \boldsymbol{r} + (1-\beta)\boldsymbol{\pi} \tag{13}$$

Reformulating the closed-form solution of the PageRank vector illustrates the relationship between the formulation with fixed parameters and the formulation with trainable parameters:

$$\boldsymbol{r} = (1-\beta)(I_n - \beta \hat{\underline{\boldsymbol{S}}})^{-1} \boldsymbol{\pi} \overset{(1)}{=} (1-\beta) \sum_{k=0}^{\infty} (\beta \hat{\underline{\boldsymbol{S}}})^k \boldsymbol{\pi} \quad = \sum_{k=0}^{\infty} (1-\beta) \beta^k \hat{\underline{\boldsymbol{S}}}^k \boldsymbol{\pi} = \sum_{k=0}^{\infty} \theta^k \hat{\underline{\boldsymbol{S}}}^k \boldsymbol{\pi} \tag{14}$$

In step (1), the geometric series expansion for matrices is applied to express the inverse $(I_n - \beta \hat{\underline{\boldsymbol{S}}})^{-1}$ as an infinite sum. This expansion is valid because the matrix $\beta \hat{\underline{\boldsymbol{S}}}$ satisfies the convergence criterion for the series, i.e., its spectral norm is strictly less than 1. This is because the eigenvalues of $\hat{\underline{\boldsymbol{S}}}$, denoted by μ_i, are constrained to the interval $[-1, 1]$, as shown in (von Luxburg, 2007) and the parameter $\beta \in (0, 1)$ ensures that the eigenvalues of $\beta \hat{\underline{\boldsymbol{S}}}$ remain within $(-1, 1)$.

Thus, by introducing the parameters $\theta_i = (1-\beta)\beta^i \in \mathbb{R}$ as trainable weights in the spectral filter, the requirement of a fixed, non-negative teleport probability $(1-\beta)$ is relaxed. This modification allows for greater flexibility, as θ_i can now adapt to the specific structural properties of the graph, enabling models to capture complex dynamics that are not possible with fixed parameters.