Piecewise Constant Spectral Graph Neural Network

Anonymous authors
Paper under double-blind review

Abstract

Graph Neural Networks (GNNs) have achieved significant success across various domains by leveraging graph structures in data. Existing spectral GNNs, which use low-degree polynomial filters to capture graph spectral properties, may not fully identify the graph's spectral characteristics because of the polynomial's small degree. However, increasing the polynomial degree might be computationally infeasible. In this paper, we introduce the **Pie**cewise **Co**nstant Spectral Graph **N**eural Network (PieCoN) to address these challenges. PieCoN combines constant spectral filters with polynomial filters to provide a more flexible way to leverage the graph structure. By adaptively partitioning the spectrum into intervals, our approach increases the range of spectral properties that can be effectively learned. Experiments on seven benchmark datasets, including both homophilic and heterophilic graphs, demonstrate that PieCoN is particularly effective on heterophilic datasets, highlighting its potential for a wide range of applications.

1 Introduction

Graph Neural Networks (GNNs) (Wu et al., 2021; Zhou et al., 2020) have demonstrated remarkable performance across various application domains. They have been successfully applied in areas such as social network analysis (Panagopoulos et al., 2023), recommendation systems (Wu et al., 2019; Ying et al., 2018), drug discovery (Jiang et al., 2021; Bongini et al., 2021), and materials modeling (Coley et al., 2019; Duval et al., 2023), where data can naturally be represented as graphs. GNNs can be broadly classified into two types: spatial and spectral GNNs. Spatial GNNs (Kipf & Welling, 2017; Velickovic et al., 2018; Chien et al., 2021) use a message passing approach to learn node representations by collecting information from neighboring nodes. This approach allows spatial GNNs to capture local structural information and adapt to varying neighborhood sizes. On the other hand, spectral GNNs (Wang & Zhang, 2022; Defferrard et al., 2016; He et al., 2021; Castro-Correa et al., 2024) use the graph's spectral characteristics, such as the graph Laplacian's eigenvalues and eigenvectors, to transform node features. By leveraging the spectral domain, these models can capture global structural patterns and apply graph convolution operations in the frequency domain.

Many existing spectral GNNs use low-degree polynomial filters, which approximate the filtering functions by applying polynomials to the graph's Laplacian matrix or other graph-shift operators (Wang & Zhang, 2022; Defferrard et al., 2016; He et al., 2021). One disadvantage of these low-degree polynomial filters is that they are continuous and, because of the low degree, may not give enough importance to specific eigenvalues, as the change between closely spaced eigenvalues cannot vary significantly. This can be problematic, particularly in real-world graphs where certain eigenvalues like *zero* have large multiplicities (Lu et al., 2024; Lim et al., 2023).

In this paper, we propose the **Pie**cewise **Co**nstant Spectral Graph **N**eural Network (PieCoN) to overcome this limitation. Our approach combines constant spectral filters with polynomial filters to better capture the spectral properties of the graph. The constant filters are defined by setting the values in the diagonal eigenvalue matrix to ones within specific intervals and zeros elsewhere, effectively isolating different frequency bands. By partitioning the spectrum into intervals, PieCoN expands the range of spectral characteristics that can be learned, improving the model's performance.

Figure 1 compares the response of a JacobiConv filter (Wang & Zhang, 2022) versus the response of a PieCoN filter across the spectrum. The figure shows how polynomial filters produce a smooth response, while piecewise constant filters can sharply focus on selected intervals, capturing crucial spectral properties that polynomial filters miss. Our theoretical analysis provides important insights about spectral GNNs by establishing error bounds for polynomial spectral filtering and proving that our model is invariant to eigenvector sign flips and basis changes. We validate our approach on standard benchmark datasets, showing improved performance, particularly when handling graphs with multiple zero eigenvalues.

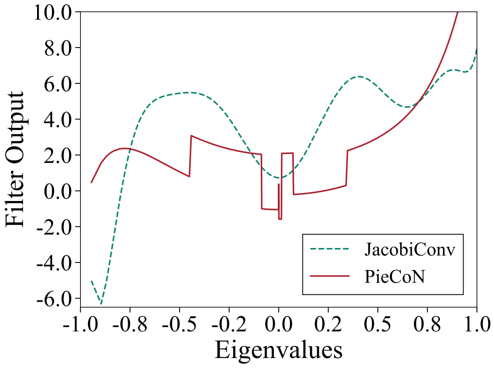


Figure 1: Comparison of JacobiConv and PieCoN trained filters on the Chameleon dataset.

Our main contributions are as follows:

- We introduce a novel spectral GNN model (PieCoN) that uses piecewise constant filters combined with polynomial filters to enhance learning from the spectral properties of graphs.
- We propose a new method to isolate frequency bands in the spectral domain by partitioning the eigenvalue spectrum into intervals, allowing the model to focus on crucial spectral properties.
- We demonstrate, through experiments, that PieCoN outperforms or shows competitive performance against spatial and spectral GNNs on real-world datasets.

2 Related Work

GNNs have evolved significantly since the early work by Bruna et al. (2014), who introduced the first modern spectral-based graph convolution network using the graph Fourier transform. Later, Kipf & Welling (2017) simplified this approach with the Graph Convolutional Network (GCN) model, which applies a first-order approximation of spectral filters to make GNNs more scalable. Other important advancements include Graph Attention Networks (GAT) (Velickovic et al., 2018), which use attention mechanisms to weigh neighboring nodes differently, and GraphSAGE (Hamilton et al., 2017), which focuses on inductive learning by generating node embeddings through sampling and aggregation techniques.

Message passing neural networks (MPNNs) (Gilmer et al., 2017) are a class of GNNs where nodes iteratively exchange and aggregate information with their neighbors. Besides GCN and GAT, several other message-passing approaches have emerged to address specific challenges. For instance, the Graph Isomorphism Networks (GIN) model (Xu et al., 2019) is designed to be as powerful as the 1-WL test for distinguishing non-isomorphic graphs, thus improving expressiveness.

Spectral GNNs leverage the eigenvalues and eigenvectors of the graph Laplacian to perform graph convolution in the spectral domain. ChebNet (Defferrard et al., 2016) introduced the use of Chebyshev polynomials for spectral filtering, improving the scalability of spectral GNNs. JacobiConv (Wang & Zhang, 2022) further enhances this by employing orthogonal Jacobi polynomials to flexibly learn graph filters. Recent work by Maskey et al. (2023) introduces a novel fractional graph Laplacian approach defined in the singular value domain to address over-smoothing, providing theoretical guarantees for both directed and undirected graphs. For heterophilic graphs with label noise, R^2LP (Cheng et al., 2024) demonstrates that increasing graph homophily can help mitigate the impact of noisy labels.

While previous spectral GNNs have leveraged polynomial filters to approximate spectral properties of graphs, they are limited by their lack of flexibility in handling critical eigenvalues because of the low degree of the polynomial filters. DSF (Guo et al., 2023) addresses this by employing a shared network on positional encoding to learn unique polynomial coefficients per node, highlighting the advantages of node-specific filters over node-unified ones. NFGNN (Zheng et al., 2023) proposes a node-oriented spectral filtering approach that learns specific filters for each node, better adapting to local homophily patterns. Lingam et al. (2022), similar to us, explore piecewise spectral filtering by using polynomial filters on two frequency bands. However, their fixed two-part partitioning scheme limits the model's ability to adapt to different graph structures. In contrast, our approach employs adaptive K-way partitioning with constant filters and introduces positive/negative decomposition for finer spectral control.

3 Background

We consider an undirected graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ with n nodes. Here, $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ is the set of nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ ($|\mathcal{E}| = m$) represents the set of edges, and $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ is the node feature matrix. The adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{n \times n}$ of the graph is defined as $\boldsymbol{A}_{ij} = 1$ if there is an edge between nodes v_i and v_j, and $\boldsymbol{A}_{ij} = 0$ otherwise. The degree matrix $\boldsymbol{D} = \operatorname{diag}(d_1, \ldots, d_n)$ is a diagonal matrix with the i-th diagonal entry as $d_i = \sum_j \boldsymbol{A}_{ij}$, representing the degree of node i. The normalized adjacency matrix $\hat{\boldsymbol{A}}$ is defined as $\hat{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A} \boldsymbol{D}^{-\frac{1}{2}}$. The normalized adjacency matrix is symmetric and can be decomposed as $\hat{\boldsymbol{A}} = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^\top$, where $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues λ_i of $\hat{\boldsymbol{A}}$, and $\boldsymbol{U}$ is an orthogonal matrix whose columns are the corresponding eigenvectors $\boldsymbol{u}_i$. Let s be the number of distinct eigenvalues of $\hat{\boldsymbol{A}}$, denoted by $\lambda'_1, \lambda'_2, \ldots, \lambda'_s$. For an eigenvalue λ, we define $\sigma(\lambda)$ as the *algebraic multiplicity* of λ, which is the number of times λ appears in the eigenvalues. A graph signal is a function that assigns a scalar value to each node in the graph. Formally, a graph signal can be represented as a vector $\boldsymbol{x} \in \mathbb{R}^n$, where each entry x_i corresponds to the signal value at node v_i.

3.1 Graph Fourier Transform and Spectral Filtering

The graph Fourier transform (Ortega et al., 2018) of a graph signal $\boldsymbol{x} \in \mathbb{R}^n$ is defined as $\hat{\boldsymbol{x}} = \boldsymbol{U}^\top \boldsymbol{x}$, and its inverse (Shuman et al., 2013) is given by $\boldsymbol{x} = \boldsymbol{U} \hat{\boldsymbol{x}}$. The spectral filtering of a signal $\boldsymbol{x}$ with a kernel $\boldsymbol{v}$ is defined as:

$$\boldsymbol{z} = \boldsymbol{v} *_G \boldsymbol{x} = \boldsymbol{U} \left(\hat{\boldsymbol{v}} \odot \hat{\boldsymbol{x}} \right) = \boldsymbol{U} \hat{\boldsymbol{V}} \boldsymbol{U}^\top \boldsymbol{x}, \tag{1}$$

where $\hat{\boldsymbol{V}} = \operatorname{diag}(\hat{v}_1, \ldots, \hat{v}_n)$ represents the spectral kernel coefficients and $\odot$ denotes element-wise multiplication.

To avoid the computationally expensive eigen-decomposition, polynomial functions $h(\hat{\boldsymbol{A}})$ are often used to approximate different kernels in spectral GNNs. Specifically, the spectral filter $h(\lambda)$ is parameterized as a polynomial of degree K:

$$h(\lambda) = \sum_{k=0}^{K} \alpha_k \lambda^k, \tag{2}$$

where α_k are learnable coefficients. Consequently, the filtering process can be reformulated as:

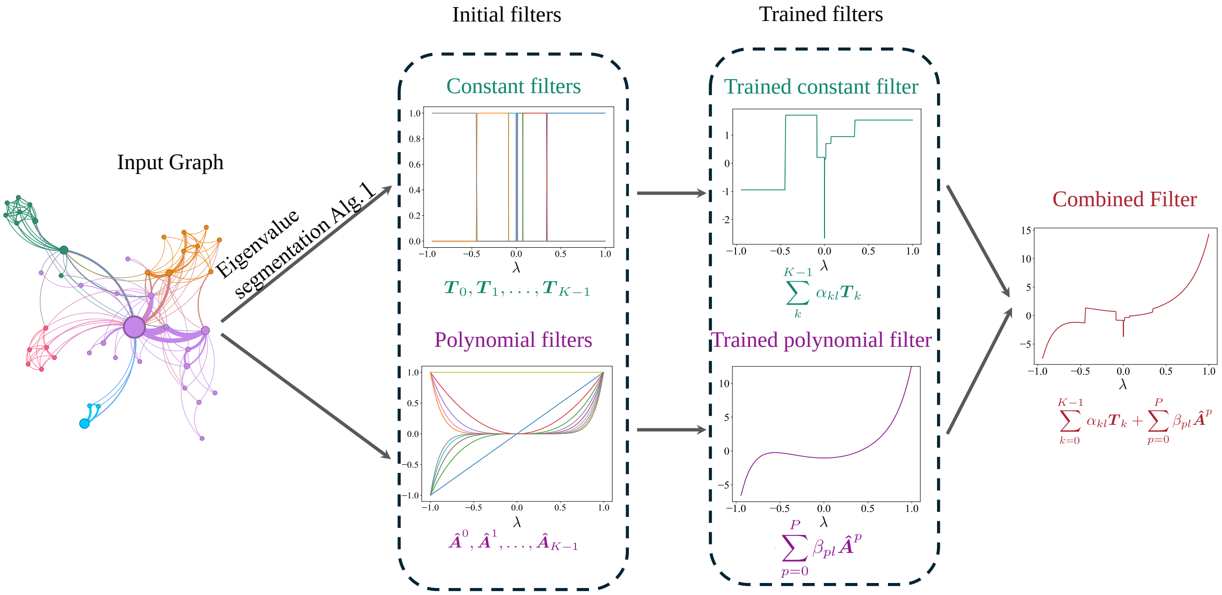
$$h(\hat{\boldsymbol{A}}) \boldsymbol{x} = \sum_{k=0}^{K} \alpha_k \hat{\boldsymbol{A}}^k \boldsymbol{x} = \boldsymbol{U} h(\boldsymbol{\Lambda}) \boldsymbol{U}^\top \boldsymbol{x}, \tag{3}$$

which allows efficient computation of the filtered signal using only matrix multiplications.

4 Piecewise Constant Spectral Graph Neural Network (PieCoN)

Current spectral GNNs often have limited flexibility in how they process graph structures due to their reliance on polynomial filters. We propose PieCoN, a model that combines different types of spectral filters to process graph data in ways that complement the capabilities of polynomial filters. The key steps of our methodology, illustrated in Figure 2, include: (1) partitioning the graph spectrum into intervals based on significant points identified through spectral analysis, (2) constructing constant spectral filters for each interval to capture global and local spectral properties, and (3) combining constant spectral filters with polynomial filters. Below, we describe the methodology in detail.

4.1 Identifying Significant Points in the Spectrum


Input Graph
Eigenvalue segmentation Alg. 1
Initial filters
Constant filters
$\boldsymbol{T}_0, \boldsymbol{T}_1, \ldots, \boldsymbol{T}_{K-1}$
Polynomial filters
$\hat{\boldsymbol{A}}^0, \hat{\boldsymbol{A}}^1, \ldots, \hat{\boldsymbol{A}}_{K-1}$
Trained filters
Trained constant filter
$\sum_{k}^{K-1} \alpha_{kl} \boldsymbol{T}_k$
Trained polynomial filter
$\sum_{p=0}^{P} \beta_{pl} \hat{\boldsymbol{A}}^p$
Combined Filter
$\sum_{k=0}^{K-1} \alpha_{kl} \boldsymbol{T}_k + \sum_{p=0}^{P} \beta_{pl} \hat{\boldsymbol{A}}^p$
λ

$$\boldsymbol{Z}_{:l} = \sum_{k=0}^{K-1} \alpha_{kl} \boldsymbol{T}_k \sigma(\boldsymbol{X}\boldsymbol{W}^1)\boldsymbol{W}^2_{:l} + \sum_{p=0}^{P} \beta_{pl} \hat{\boldsymbol{A}}^p \sigma(\boldsymbol{X}\boldsymbol{W}^1)\boldsymbol{W}^2_{:l} \quad (7)$$

Figure 2: Overview of the PieCoN model. Our method processes an input graph through eigenvalue segmentation (Alg. 1) to create constant filters, while separately applying polynomial filters. These filters are trained and combined to create the final spectral filter.

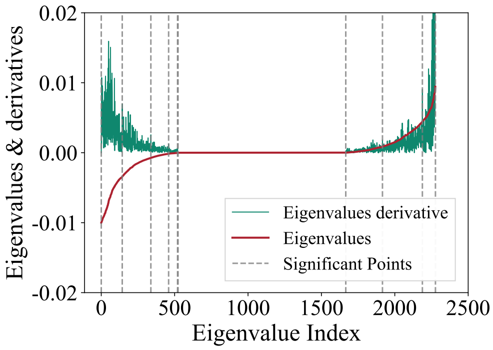


Figure 3: Using the derivative of eigenvalues to identify significant points which show relatively high changes in the spectrum.

Understanding the spectral properties of a graph is crucial for analyzing its structure. A key challenge is identifying significant points in the eigenvalue spectrum, which can reveal important structural insights (Figure 3). Algorithm 1 addresses this by identifying large gaps in the eigenvalue spectrum, which can indicate distinct frequency bands in the graph's spectral representation and highlight structural changes or regions of high spectral variation (Luxburg, 2007; Fiedler, 1973; Chung, 1997). By adaptively partitioning the eigenvalue space around these critical points, PieCoN can capture the most informative spectral features of the graph.

To formalize this process, let us consider the sorted eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ of a graph. Define the discrete derivative of these eigenvalues as $d(\lambda_i) = \lambda_{i+1} - \lambda_i$ for $i = 1, 2, \ldots, n-1$. For identifying significant points, we use Algorithm 1 to rank the peaks by significance. Let $\{a_0, a_1, \ldots, a_{K-1}\}$ denote the indexes of K peaks with highest significance, where each λ_{a_k} corresponds to a significant change in the spectrum. These peaks partition the spectrum into K intervals, such that $[\lambda_{a_k}, \lambda_{a_{k+1}})$ for $k = 0, 1, \ldots, K-1$ are the boundaries of these intervals.

4.2 Construction of Constant Filters

For each interval $[a_k, a_{k+1})$, we construct a spectral filter $\boldsymbol{T}_k$, as shown in Fig. 4). The filter is defined as:

$$\boldsymbol{T}_k = \boldsymbol{U}\boldsymbol{E}_k\boldsymbol{U}^\top = \boldsymbol{U}_{[:,a_k:a_{k+1}]}\boldsymbol{U}^\top_{[:,a_k:a_{k+1}]}, \tag{4}$$

where $\boldsymbol{E}_k$ is a binary diagonal matrix with non-zero entries corresponding to the eigenvalues within the k-th interval and $\boldsymbol{U}_{[:,a_k:a_{k+1}]}$ is the submatrix of $\boldsymbol{U}$ with columns a_k to $a_{k+1} - 1$. Specifically, the matrix $\boldsymbol{E}_k$ for

Algorithm 1 Thresholding Algorithm for Identifying Significant Eigenvalue Gaps

```
function IDENTIFY_SIGNIFICANT_GAPS($\boldsymbol{\lambda}$, $W$, $K$)    ▷ $\boldsymbol{\lambda}$: Sorted eigenvalues, $W$: Window size for averaging, $K$: Number of top indices
    $\epsilon \leftarrow$ small constant    ▷ A very small positive value to avoid division by zero
    $\boldsymbol{s} \leftarrow \mathbf{0}$    ▷ Significance of each index
    for $i \leftarrow W$ to $n - W - 1$ do
        $\mu_p, \sigma_p \leftarrow \text{mean}(\boldsymbol{\lambda}_{i-W:i}), \text{std}(\boldsymbol{\lambda}_{i-W:i})$    ▷ Mean and standard deviation before $i$
        $\mu_n, \sigma_n \leftarrow \text{mean}(\boldsymbol{\lambda}_{i+1:i+W+1}), \text{std}(\boldsymbol{\lambda}_{i+1:i+W+1})$    ▷ Mean and standard deviation after $i$
        $s_i \leftarrow \frac{|\lambda_i - \mu_p|}{\sigma_p + \epsilon} + \frac{|\lambda_i - \mu_n|}{\sigma_n + \epsilon}$    ▷ Sum of normalized distances to adjacent means
        if $\lambda_i = \lambda_{i-1}$ then
            $s_i \leftarrow 0$    ▷ Set to zero if no gap exists
        end if
    end for
    $a_0 = 0$, $a_{K-1} = n + 1$
    $a_1, a_2, \ldots, a_{K-2} \leftarrow$ indices of the largest $K - 2$ values in $\boldsymbol{s}$
    return $a_0, a_1, \ldots, a_{K-1}$
end function
```

an interval $[a_k, a_{k+1})$ is defined as:

$$\boldsymbol{E}_k = \text{diag}(0, \ldots, \underbrace{1}_{a_k}, 1, \ldots, \underbrace{1}_{a_{k+1}-1}, \ldots, 0). \tag{5}$$

This construction ensures that $\boldsymbol{E}_k$ captures the eigenvectors in the specified interval, allowing the filter $\boldsymbol{T}_k$ to encapsulate the corresponding spectral properties. Our approach differs from traditional polynomial-based spectral filters by enabling more flexible and tailored filtering of the graph's spectral components.

4.3 Polynomial Filters

In addition to the spectral filters $\boldsymbol{T}_k$, polynomial filters of the form $\hat{\boldsymbol{A}}^p$ are used (Fig. 5). These filters provide a way to incorporate local neighborhood information into the model. By adjusting the polynomial degree p, we can capture varying scales of locality in the graph.

Using the distinct eigenvalues of $\hat{\boldsymbol{A}}$ we can get:

$$(\hat{\boldsymbol{A}} - \lambda'_1 \boldsymbol{I}) \cdots (\hat{\boldsymbol{A}} - \lambda'_s \boldsymbol{I}) = 0. \tag{6}$$

This implies that polynomial filters have at most s free parameters because any polynomial of degree higher than s can be reduced to a polynomial of degree s. The inclusion of polynomial filters complements the constant spectral filters by providing a smooth interpolation between different spectral components. This combination allows PieCoN to capture both sharp and gradual changes in the graph's spectral properties.

4.4 Combining Constant and Polynomial Filters

The final embedding matrix of the nodes is computed by combining the constant spectral filters and polynomial filters. As in Jacobi convolutions (Wang & Zhang, 2022), we consider independent filtering on each of the h channels in $\boldsymbol{X}$ simultaneously, the multichannel filtering can be denoted as:

$$\boldsymbol{Z}_{:l} = \underbrace{\sum_{k=0}^{K-1} \alpha_{kl} \boldsymbol{T}_k \sigma(\boldsymbol{X}\boldsymbol{W}^1)\boldsymbol{W}^2_{:l}}_{\text{Constant Filters}} + \underbrace{\sum_{p=0}^{P} \beta_{pl} \hat{\boldsymbol{A}}^p \sigma(\boldsymbol{X}\boldsymbol{W}^1)\boldsymbol{W}^2_{:l}}_{\text{Polynomial Filters}}, \tag{7}$$

where:

- **Matrix**$_{:l}$ is the l^{th} column of the **Matrix**;

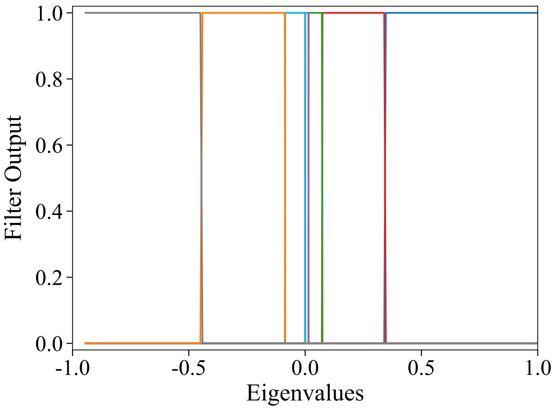


Figure 4: Constant Filters.

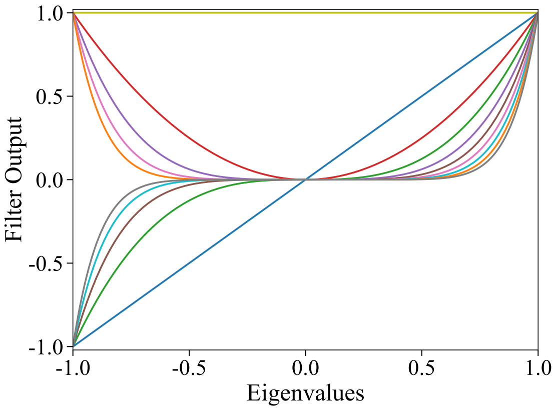


Figure 5: Polynomial Filters.

- $\boldsymbol{X}$ is the input feature matrix, representing the initial features of the nodes;
- $\boldsymbol{W}^1$ and $\boldsymbol{W}^2$ are weight matrices to be learned during training. $\boldsymbol{W}^1$ maps the input features to an intermediate space, and $\boldsymbol{W}^2$ maps the intermediate representations to the embedding space;
- $\sigma(\cdot)$ is a non-linear activation function applied element-wise, introducing non-linearity into the model;
- α_{kl} are learnable coefficients associated with the spectral filters $\boldsymbol{T}_k$ for each dimension l;
- β_{pl} are learnable coefficients associated with the polynomial filters $\hat{\boldsymbol{A}}^p$ for each dimension l.

Each element of $\boldsymbol{T}_k$ represents a similarity between two nodes in some range of frequencies. When we perform the matrix multiplication $\boldsymbol{M}\boldsymbol{X}$ for some similarity matrix $\boldsymbol{M} = \boldsymbol{T}_k$, each entry $\boldsymbol{M}_{ij}$ in the similarity matrix $\boldsymbol{M}$ represents the weight or importance of node j in contributing to the feature vector of node i. If $\boldsymbol{M}$ is non-negative, it means each node contributes either positively or not at all to the feature aggregation. Negative values, on the other hand, would imply subtracting features from neighbors, which is typically not meaningful in most graph-based learning contexts, where the goal is to aggregate features to enhance node representations. Therefore, we separate $\boldsymbol{T}_k$ into positive and negative parts, as follows:

$$\boldsymbol{Z}_{:l} = \underbrace{\sum_{k=0}^{K-1} \alpha_{kl}^{+} (\boldsymbol{T}_k)^{+} \sigma(\boldsymbol{X}\boldsymbol{W}^1)\boldsymbol{W}^2_{:l}}_{\text{Positive Part}} + \underbrace{\sum_{k=0}^{K-1} \alpha_{kl}^{-} (\boldsymbol{T}_k)^{-} \sigma(\boldsymbol{X}\boldsymbol{W}^1)\boldsymbol{W}^2_{:l}}_{\text{Negative Part}} + \underbrace{\sum_{p=0}^{P} \beta_{pl} \hat{\boldsymbol{A}}^p \sigma(\boldsymbol{X}\boldsymbol{W}^1)\boldsymbol{W}^2_{:l}}_{\text{Polynomial Filters}} \tag{8}$$

where α_{kl}^{+} and α_{kl}^{-} are learnable coefficients associated with the spectral filters $\boldsymbol{T}_k^{+}$ and $\boldsymbol{T}_k^{-}$ for each dimension l. The superscripts + and - indicate coefficients for the positive and negative parts of the spectral filters, respectively.

4.5 Computational Complexity

The computational complexity of our method is broken down as follows:

1. *Eigendecomposition (precomputation):* $\mathcal{O}(n^3)$ for computing spectral components.
2. *Filter construction and sparsification (precomputation):* $\mathcal{O}(n^3 + 2Kn^2\log(m))$ for constructing and sparsifying T_K filters, where K is the number of spectral intervals, and m is the edge count.
3. *Model propagation:* $\mathcal{O}((2K+P)md)$ during training and inference, where P is the polynomial degree, and d is the feature dimension. This matches the theoretical time complexity of JacobiConv (Wang & Zhang, 2022) and GPRGNN (Chien et al., 2021), while being more efficient than BernNet's (He et al., 2021) $\mathcal{O}(P^2md)$.

5 Empirical and Theoretical Analysis

This section provides theoretical and empirical analyses to establish the advantages of piecewise constant spectral filters over polynomial filters and validate the design choices of PieCoN. For the theoretical part, we explain an error bound from polynomial approximation theory, which shows the limits of polynomial spectral filtering when dealing with sharp changes in functions. We also discuss the challenges with eigenvector representations, such as their invariance under sign flips and basis shifts, which can affect the generalization of graph learning models. Additionally, in Appendix A, we analyze how specific graph structures influence the eigenvalue spectrum, with a focus on the eigenvalue 0, and explain why separating constant filters into negative and positive parts helps improve model performance and reduces approximation errors.

5.1 Error Analysis for Polynomial Approximation

To analyze the fundamental limitations of polynomial approximations in spectral filtering, we establish a theorem characterizing the error bounds.

Theorem 1 (Error bound for spectral polynomial approximation)**.** *Let* $\hat{\boldsymbol{A}} \in \mathbb{R}^{n\times n}$ *be a normalized adjacency matrix with spectrum* $\{\lambda_i\}_{i=1}^n$ *where* $-1 \leq \lambda_1 \leq \cdots \leq \lambda_n \leq 1$*. Let* $f : [-1,1] \to \mathbb{R}$ *be some filter function that the model needs to find and suppose it has a jump discontinuity of magnitude* $h > 0$ *between consecutive eigenvalues* λ_R *and* λ_{R+1}*. For any polynomial* $p \in \mathcal{P}_d$*(the space of polynomials of degree at most* d*), satisfying* $\|p\|_\infty = \sup_{x\in[-1,1]} |p(x)| = 1$*, the approximation error is:*

$$\mathcal{E}(p,f) = \sum_{i=1}^{n} |p(\lambda_i) - f(\lambda_i)| \geq h - |\lambda_{R+1} - \lambda_R| \cdot d^2. \tag{9}$$

Proof. By Markov's polynomial inequality(Sahoo & Riedel, 1998):

$$\|p'\|_\infty \leq d^2 \|p\|_\infty = d^2, \quad \forall p \in \mathcal{P}_d. \tag{10}$$

Next using the mean value theorem(Achiezer, 1992), we find that $\exists\ \xi_{xy} \in [x,y]$ such that:

$$|p(x) - p(y)| = |p'(\xi_{xy})| \cdot |x-y| \leq d^2|x-y|, \quad \forall\ x,y \in [-1,1]. \tag{11}$$

Now, we define the local error at consecutive eigenvalues as:

$$\mathcal{E}_2 = |p(\lambda_R) - f(\lambda_R)| + |p(\lambda_{R+1}) - f(\lambda_{R+1})|. \tag{12}$$

Applying the triangle inequality, we obtain:

$$\begin{aligned} \mathcal{E}_2 &\geq |f(\lambda_{R+1}) - f(\lambda_R)| - |p(\lambda_{R+1}) - p(\lambda_R)| \\ &\geq h - |\lambda_{R+1} - \lambda_R| \cdot d^2. \end{aligned} \tag{13}$$

Since it holds that $\mathcal{E}(p,f) \geq \mathcal{E}_2$, the result follows. □

This result demonstrates a key limitation: For small d when $|\lambda_{R+1} - \lambda_R| \ll \frac{h}{d^2}$, the approximation error $\mathcal{E}(p,f) \geq \mathcal{E}_2 \approx h$. This is particularly problematic for spectral filtering where: (1) sharp transitions in the filter response are often desired (h is large), (2) some eigenvalues may be very close together ($|\lambda_{R+1} - \lambda_R|$ is small), and (3) using high-degree polynomials is computationally expensive.

In contrast, by using piecewise constant filters, we can add a constant filter only at point λ_{R+1} with the value h and eliminate the jump entirely. The ablation study in Table 3 demonstrates that adding constant filters to polynomial filters improves performance. This theoretical result justifies combining polynomial filters with constant filters in our approach.

5.2 Sign and Basis Invariance

Eigenvectors corresponding to a given eigenvalue can have multiple representations. For example, if $\lambda = 0$ is an eigenvalue of the normalized adjacency matrix $\hat{\boldsymbol{A}}$, any eigenvector $\boldsymbol{u}_i$ associated with $\lambda = 0$ can be replaced with its opposite $-\boldsymbol{u}_i$ or any linear combination of eigenvectors for the same eigenvalue. This variability introduces sign and basis ambiguity problems, which can lead to inconsistent or unpredictable results in learning tasks.

Proposition 1. *Polynomial filters are invariant to sign changes and basis choices.*

Proof. The key property we use is that the product $\boldsymbol{U}_\mu \boldsymbol{U}_\mu^\top$, where $\boldsymbol{U}_\mu$ is a matrix of eigenvectors associated with eigenvalue μ, remains invariant under sign changes and different choices of basis (Lim et al., 2023). This implies that regardless of which orthonormal basis is chosen for a given eigenspace, the product $\boldsymbol{U}_\mu \boldsymbol{U}_\mu^\top$ stays the same.

Consider two orthonormal bases $\boldsymbol{U}_\mu$ and $\boldsymbol{V}_\mu$ for the eigenspace corresponding to eigenvalue μ. There exists an orthogonal matrix $\boldsymbol{Q}$ such that:

$$\boldsymbol{V}_\mu = \boldsymbol{U}_\mu \boldsymbol{Q}. \tag{14}$$

Using this, we show the invariance:

$$\boldsymbol{V}_\mu \boldsymbol{V}_\mu^\top = (\boldsymbol{U}_\mu \boldsymbol{Q})(\boldsymbol{U}_\mu \boldsymbol{Q})^\top = \boldsymbol{U}_\mu \boldsymbol{Q}\boldsymbol{Q}^\top \boldsymbol{U}_\mu^\top = \boldsymbol{U}_\mu \boldsymbol{U}_\mu^\top. \tag{15}$$

where we used the fact that $\boldsymbol{Q}\boldsymbol{Q}^\top = \boldsymbol{I}$ since $\boldsymbol{Q}$ is orthogonal. This confirms that $\boldsymbol{U}_\mu \boldsymbol{U}_\mu^\top$ is invariant under the change of basis.

Now, consider the matrix power $\hat{\boldsymbol{A}}^p$, which can be expressed as:

$$\hat{\boldsymbol{A}}^p = \sum_{i=1}^{n} \lambda_i^p \boldsymbol{u}_i \boldsymbol{u}_i^\top = \sum_{i=1}^{s} (\lambda_i')^p \boldsymbol{U}_{\lambda_i'} \boldsymbol{U}_{\lambda_i'}^\top, \tag{16}$$

where λ_i' are the distinct eigenvalues, and $\boldsymbol{U}_{\lambda_i'}$ are the corresponding eigenvector matrices.

Since each term $\boldsymbol{U}_{\lambda_i'} \boldsymbol{U}_{\lambda_i'}^\top$ is invariant to basis choices and each term $\boldsymbol{u}_i \boldsymbol{u}_i^\top = (-\boldsymbol{u}_i)(-\boldsymbol{u}_i)^\top$ is invariant to sign changes, it follows that $\hat{\boldsymbol{A}}^p$ is also invariant to both. ☐

Proposition 2. *Constant filters are invariant to sign changes and basis choices.*

Proof. Consider the constant filter $\boldsymbol{T}_k$ defined over the interval $[a_k, a_{k+1})$ for some k. Let t_k be the index such that $\lambda'_{t_k} = \lambda_{a_k}$, and thus $\lambda'_{t_{k+1}-1} = \lambda_{a_{k+1}-1}$.

Suppose l, r are indices such that $\lambda_{l-1} \neq \lambda_l = \lambda_{l+1} = \cdots = \lambda_{r-1} \neq \lambda_r$. According to Algorithm 1, no significant point i will be selected with $l < i < r$, ensuring that constant eigenvalue intervals are not split. Consequently, $\lambda'_{t_k} \neq \lambda_{a_k-1}$ when $a_k > 1$, and $\lambda'_{t_{k+1}-1} \neq \lambda_{a_{k+1}}$ when $a_{k+1} \leq n$.

The distinct eigenvalues of $\hat{\boldsymbol{A}}$ within the interval $[\lambda_{a_k}, \lambda_{a_{k+1}-1}]$ are:

$$\lambda'_{t_k}, \lambda'_{t_k+1}, \ldots, \lambda'_{t_{k+1}-1}. \tag{17}$$

Using Equation 4, we express $\boldsymbol{T}_k$ as:

$$\boldsymbol{T}_k = \boldsymbol{U}_{[:,a_k:a_{k+1}]} \boldsymbol{U}_{[:,a_k:a_{k+1}]}^\top = \sum_{i=a_k}^{a_{k+1}-1} \boldsymbol{u}_{\lambda_i} \boldsymbol{u}_{\lambda_i}^\top = \sum_{i=t_k}^{t_{k+1}-1} \boldsymbol{U}_{\lambda_i'} \boldsymbol{U}_{\lambda_i'}^\top. \tag{18}$$

Since each term $\boldsymbol{U}_{\lambda_i'} \boldsymbol{U}_{\lambda_i'}^\top$ is invariant to basis choices and each term $\boldsymbol{u}_i \boldsymbol{u}_i^\top = (-\boldsymbol{u}_i)(-\boldsymbol{u}_i)^\top$ is invariant to sign changes, it follows that $\boldsymbol{T}_k$ is also invariant to both. ☐

Both polynomial and constant filters are robust to different eigenvector representations, ensuring that learned representations are not affected by arbitrary sign changes or basis choices, thus improving model stability and generalization.

Table 1: Statistics of the datasets used for node classification.

Dataset	**Nodes**	**Edges**	**Classes**	**Homophily Ratio**	$\sigma(0)$/**Nodes**
Chameleon	2,277	31,396	5	0.23	0.52
Squirrel	5,201	198,423	5	0.22	0.37
Actor	7,600	26,705	5	0.22	0.15
Amazon-Ratings	24,492	93,050	5	0.38	0.17
Cora	2,708	5,278	7	0.81	0.11
Citeseer	3,327	4,614	6	0.74	0.14
Amazon-Photo	7,650	71,831	8	0.83	0.02

6 Experimental Evaluation

6.1 Datasets

We evaluate PieCoN on seven diverse node classification datasets with varying graph structures and homophily ratios (Table 1). Cora and Citeseer are citation networks where nodes are research papers and edges represent citations. Photo is a product co-occurrence graph with nodes as products and edges representing co-purchase relationships. Actor is a graph where nodes are actors and edges denote co-occurrence in films. Chameleon and Squirrel are graphs derived from Wikipedia pages. Nodes represent web pages, and edges denote mutual links. Amazon-Ratings is a product co-purchasing network where nodes are products and edges indicate frequent co-purchases, with the task of predicting product rating classes.

All datasets were randomly split into 60% training, 20% validation, and 20% test sets for 10 different seeds. For each dataset, we report the average performance along with the 95% confidence interval. Details about hyperparameter optimization and the running environment are provided in Appendix B.

6.2 Baselines

We compare PieCoN against several baseline models categorized into different groups based on their underlying graph learning methodologies:

- **Spatial-based GNNs**: Graph Convolutional Network (GCN) (Kipf & Welling, 2017), Graph Attention Network (GAT) (Velickovic et al., 2018), Higher-order GCN (H_2GCN) (Zhu et al., 2020), and GCNII (Chen et al., 2020).
- **Spectral-based GNNs**: UniFilter (Huang et al., 2024), LanczosNet (Liao et al., 2019), ChebyNet (Defferrard et al., 2016), Generalized PageRank GNN (GPR-GNN) (Chien et al., 2021), BernNet (He et al., 2021), ChebNetII (He et al., 2022), JacobiConv (Wang & Zhang, 2022), OptBasisGNN (Guo & Wei, 2023), and Specformer (Bo et al., 2023).
- **Free eigenvalues**: A graph neural network that learns a spectral filter by directly parameterizing the eigenspectrum $\hat{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, where $\boldsymbol{\Lambda}$ contains trainable eigenvalues.

6.3 Results

Table 2 shows the node classification accuracy of PieCoN compared to baseline models across various datasets. We observe that PieCoN achieves the highest performance on five datasets. Notably, the largest improvements are observed on the heterophilic datasets Chameleon, Squirrel, and Amazon-Ratings, with gains of 1.1%, 2.2% and 4.1%, respectively. This may be linked to the high multiplicity of the eigenvalue 0 in the normalized adjacency matrix of these graphs (see Table 1), to which our method gives more importance. For homophilic datasets, such as Cora and Amazon-Photo, PieCoN also demonstrates competitive performance, achieving slight improvements over existing methods. The smaller gains suggest that traditional GNNs already perform well in these settings, as they inherently align with homophilic assumptions. Nonetheless, PieCoN remains

Table 2: Results on real-world node classification tasks.

Model	**Heterophilic**				**Homophilic**		
	Chameleon	**Squirrel**	**Actor**	**Amazon-Ratings**	**Cora**	**Citeseer**	**Amazon-Photo**
	Spatial-based GNNs						
GCN	$68.10_{\pm 1.20}$	$50.11_{\pm 1.21}$	$34.65_{\pm 0.68}$	$48.80_{\pm 0.22}$	$87.18_{\pm 0.87}$	$81.04_{\pm 0.67}$	$85.87_{\pm 0.83}$
GAT	$63.13_{\pm 1.93}$	$44.49_{\pm 0.88}$	$33.93_{\pm 2.47}$	$50.28_{\pm 0.55}$	$88.03_{\pm 0.79}$	$80.52_{\pm 0.71}$	$90.94_{\pm 0.68}$
H_2GCN	$57.11_{\pm 1.58}$	$36.42_{\pm 1.89}$	$35.86_{\pm 1.03}$	$48.17_{\pm 0.52}$	$86.92_{\pm 1.37}$	$77.07_{\pm 1.64}$	$93.02_{\pm 0.91}$
GCNII	$63.44_{\pm 0.85}$	$41.96_{\pm 1.02}$	$36.89_{\pm 0.95}$	$46.60_{\pm 1.20}$	$88.46_{\pm 0.82}$	$79.97_{\pm 0.65}$	$89.94_{\pm 0.31}$
	Spectral-based GNNs						
Free eigenvalues	$69.58_{\pm 1.31}$	$59.76_{\pm 1.01}$	$41.61_{\pm 0.63}$	$44.28_{\pm 1.13}$	$84.91_{\pm 0.89}$	$77.39_{\pm 0.82}$	$86.08_{\pm 0.81}$
LanczosNet	$64.81_{\pm 1.56}$	$48.64_{\pm 1.77}$	$38.16_{\pm 0.91}$	$48.35_{\pm 0.40}$	$87.77_{\pm 1.45}$	$80.05_{\pm 1.65}$	$93.21_{\pm 0.85}$
ChebyNet	$59.28_{\pm 1.25}$	$40.55_{\pm 0.42}$	$37.61_{\pm 0.89}$	$50.20_{\pm 0.52}$	$86.67_{\pm 0.82}$	$79.11_{\pm 0.75}$	$93.77_{\pm 0.32}$
GPR-GNN	$67.28_{\pm 1.09}$	$50.15_{\pm 1.92}$	$39.92_{\pm 0.67}$	$49.37_{\pm 0.71}$	$88.57_{\pm 0.69}$	$80.12_{\pm 0.83}$	$93.85_{\pm 0.28}$
BernNet	$68.29_{\pm 1.58}$	$51.35_{\pm 0.73}$	$41.79_{\pm 1.01}$	$48.82_{\pm 0.20}$	$88.52_{\pm 0.95}$	$80.09_{\pm 0.79}$	$93.63_{\pm 0.35}$
ChebNetII	$71.37_{\pm 1.01}$	$57.72_{\pm 0.59}$	$41.75_{\pm 1.07}$	$48.79_{\pm 0.21}$	$88.71_{\pm 0.93}$	$80.53_{\pm 0.79}$	$94.92_{\pm 0.33}$
JacobiConv	$73.92_{\pm 1.07}$	$57.38_{\pm 0.60}$	$40.43_{\pm 0.81}$	$48.53_{\pm 0.96}$	$88.69_{\pm 1.03}$	$\mathbf{81.65}_{\mathbf{0.46}}$	$95.36_{\pm 0.24}$
OptBasisGNN	$74.40_{\pm 0.90}$	$63.98_{\pm 1.12}$	$\mathbf{42.39}_{\pm \mathbf{0.52}}$	$48.80_{\pm 0.21}$	$87.96_{\pm 0.71}$	$80.79_{\pm 1.35}$	$94.71_{\pm 0.33}$
Specformer	$74.92_{\pm 0.98}$	$64.26_{\pm 1.18}$	$41.56_{\pm 1.25}$	OOM	$87.55_{\pm 0.87}$	$80.98_{\pm 0.79}$	$95.29_{\pm 0.30}$
UniFilter	$74.11_{\pm 1.68}$	$63.52_{\pm 1.30}$	$40.11_{\pm 1.31}$	$50.02_{\pm 0.70}$	$89.10_{\pm 1.07}$	$81.21_{\pm 1.66}$	$94.96_{\pm 0.74}$
PieCoN	$\mathbf{75.75}_{\pm \mathbf{0.96}}$	$\mathbf{65.67}_{\pm \mathbf{0.82}}$	$39.79_{\pm 0.56}$	$\mathbf{52.37}_{\pm \mathbf{0.50}}$	$\mathbf{89.16}_{\pm \mathbf{0.64}}$	$80.98_{\pm 0.57}$	$\mathbf{95.65}_{\pm \mathbf{0.34}}$

robust, indicating that its spectral filtering approach does not hinder its ability to learn from homophilic graphs. These results indicate that PieCoN is effective in both heterophilic and homophilic settings.

6.4 Ablation Study

We have performed an ablation study using Eq. (8) to evaluate the contribution of each component on model performance. The results in Table 3 reveal several key findings. First, the full model incorporating all three components (positive part, negative part, and polynomial filters) achieves the best performance on 5 out of 7 datasets, with notable improvements on Chameleon (75.75%), Squirrel (65.67%), and Amazon-Ratings (52.37%). The combination of positive and negative parts without polynomial filters also shows strong performance, suggesting that these components capture complementary spectral information. For instance, on Squirrel, adding the negative part to the positive part improves accuracy from 60.50% to 65.00%. Interestingly, on the Actor dataset, using only polynomial filters yields the best performance (40.31%), while on Citeseer, the positive part alone achieves optimal results (81.72%).

In a separate experiment, we also create a simple spectral method with the eigenvalues as parameters. The results of this experiment are presented in Table 2 with the model name "Free eigenvalues". However, this approach may be less effective because the method does not receive any explicit structural information associated with the eigenvalues.

7 Limitations

Our work has some limitations. The computational complexity of $O(n^3)$ for eigendecomposition presents scalability challenges for large-scale graphs. Furthermore, the model's performance is highly dependent on how we partition the eigenvalue intervals, and our current approach using hard thresholding to identify significant spectral changes may not be optimal. A more sophisticated approach using soft thresholding could

Table 3: Ablation study results.

Pos. Part	Neg. Part	Poly.	Chameleon	Squirrel	Actor	Amazon-Ratings	Cora	Citeseer	Amazon-Photo
✗	✗	✓	66.35$_{\pm 0.88}$	48.39$_{\pm 0.78}$	**40.31$_{\pm 0.94}$**	50.21$_{\pm 0.87}$	88.74$_{\pm 0.77}$	80.33$_{\pm 0.85}$	95.57$_{\pm 0.35}$
✗	✓	✗	67.26$_{\pm 0.50}$	54.65$_{\pm 0.72}$	32.07$_{\pm 1.17}$	49.66$_{\pm 0.71}$	84.30$_{\pm 1.06}$	75.45$_{\pm 0.68}$	92.70$_{\pm 0.34}$
✓	✗	✗	73.61$_{\pm 0.81}$	60.50$_{\pm 1.03}$	38.98$_{\pm 0.78}$	47.35$_{\pm 0.76}$	86.80$_{\pm 1.16}$	**81.72$_{\pm 0.58}$**	94.95$_{\pm 0.33}$
✓	✓	✗	74.77$_{\pm 1.01}$	65.00$_{\pm 1.12}$	39.02$_{\pm 0.54}$	49.28$_{\pm 0.62}$	87.22$_{\pm 1.12}$	81.54$_{\pm 0.63}$	94.76$_{\pm 0.39}$
✓	✓	✓	**75.75$_{\pm 0.96}$**	**65.67$_{\pm 0.82}$**	39.79$_{\pm 0.56}$	**52.37$_{\pm 0.50}$**	**89.16$_{\pm 0.64}$**	80.98$_{\pm 0.57}$	**95.65$_{\pm 0.34}$**

provide smoother transitions between intervals and potentially better capture the continuous nature of the spectrum.

8 Conclusion

In this paper, we presented the Piecewise Constant Spectral Graph Neural Network (PieCoN), a new approach to graph prediction tasks. Our method aims to address some limitations of existing spectral GNNs by combining constant spectral filters with polynomial filters to capture a broader range of spectral characteristics in real-world graphs. We introduced an adaptive spectral partitioning technique that analyzes the derivative of sorted eigenvalues to identify significant spectral changes. This helps focus on the most informative regions of the spectrum. PieCoN expands the search space of possible eigenvalue filters beyond traditional polynomial-based filters, allowing for a more tailored capture of graph spectral properties. This is particularly useful when dealing with graphs that have high eigenvalue multiplicity. By integrating spectral filters with polynomial filters, our approach attempts to model both global graph structure and local neighborhood information. Our experiments on six benchmark datasets, covering both homophilic and heterophilic graph structures, suggest that PieCoN performs well on both types of datasets.

References

Naum Achiezer. *Theory of approximation.* Dover Publications, Inc., New York, 1992.

Anirban Banerjee. *The Spectrum of the Graph Laplacian as a Tool for Analyzing Structure and Evolution of Networks.* Ph.d. thesis, Universität Leipzig, Leipzig, Germany, 2008.

James Bergstra et al. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2011.

Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. Specformer: Spectral graph neural networks meet transformers. In *International Conference on Learning Representations (ICLR)*, 2023.

Pietro Bongini et al. Molecular generative graph neural networks for drug discovery. *Neurocomputing*, 450: 242–252, 2021.

Joan Bruna et al. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR)*, 2014.

Jhon A. Castro-Correa, Jhony H. Giraldo, Mohsen Badiey, and Fragkiskos D. Malliaros. Gegenbauer graph neural networks for time-varying signal reconstruction. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9):11734–11745, 2024.

Ming Chen et al. Simple and deep graph convolutional networks. In *International Conference on Machine Learning (ICML)*, 2020.

Yao Cheng et al. Resurrecting label propagation for graphs with heterophily and label noise. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2024.

Eli Chien et al. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations (ICLR)*, 2021.

Fan R. K. Chung. *Spectral Graph Theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

Connor W Coley et al. A graph-convolutional neural network model for the prediction of chemical reactivity. *Chemical Science*, 10(2):370–377, 2019.

Michaël Defferrard et al. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Alexandre Duval, Victor Schmidt, Alex Hernández-García, Santiago Miret, Fragkiskos D. Malliaros, Yoshua Bengio, and David Rolnick. FAENet: Frame averaging equivariant GNN for materials modeling. In *International Conference on Machine Learning (ICML)*, 2023.

Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.

Justin Gilmer et al. Neural message passing for quantum chemistry. In *International Conference on Machine Learning (ICML)*, 2017.

Jingwei Guo et al. Graph neural networks with diverse spectral filtering. In *ACM Web Conference (WWW)*, 2023.

Yuhe Guo and Zhewei Wei. Graph neural networks with learnable and optimal polynomial bases. In *International Conference on Machine Learning (ICML)*, 2023.

Will Hamilton et al. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Mingguo He et al. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Mingguo He et al. Convolutional neural networks on graphs with chebyshev approximation, revisited. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Keke Huang et al. How universal polynomial bases enhance spectral graph neural networks: Heterophily, over-smoothing, and over-squashing. In *International Conference on Machine Learning (ICML)*, 2024.

Dejun Jiang et al. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. *Journal of Cheminformatics*, 13(1):1–23, 2021.

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

Renjie Liao et al. Lanczosnet: Multi-scale deep graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Derek Lim et al. Sign and basis invariant networks for spectral graph representation learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Vijay Lingam et al. A piece-wise polynomial filtering approach for graph neural networks. In *ICLR Workshop on Geometrical and Topological Representation Learning*, 2022.

Kangkang Lu et al. Improving expressive power of spectral graph neural networks with eigenvalue correction. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

Sohir Maskey et al. A fractional graph laplacian approach to oversmoothing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Antonio Ortega et al. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

George Panagopoulos, Nikolaos Tziortziotis, Michalis Vazirgiannis, and Fragkiskos Malliaros. Maximizing influence with graph neural networks. In *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2023.

Prasanna Sahoo and Thomas Riedel. *Mean value theorems and functional equations.* World Scientific, Singapore, 1998.

David I. Shuman et al. 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.

Petar Velickovic et al. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Xiyuan Wang and Muhan Zhang. How powerful are spectral graph neural networks. In *International Conference on Machine Learning (ICML)*, 2022.

Shu Wu et al. Session-based recommendation with graph neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1): 4–24, 2021.

Keyulu Xu et al. How powerful are graph neural networks? In *International Conference on Learning Representations (ICLR)*, 2019.

Rex Ying et al. Graph convolutional neural networks for web-scale recommender systems. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

Shuai Zheng et al. Node-oriented spectral filtering for graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(1):388–402, 2023.

Jie Zhou et al. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

Jiong Zhu et al. Beyond homophily in graph neural networks: Current limitations and effective designs. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

A Analysis of Graph Structure and Eigenvalue Zero

The presence of eigenvalue 0 in graph spectra reveals important structural properties that many polynomial-based GNN methods overlook. Real-world datasets often exhibit high multiplicity of eigenvalue 0 (Table 1), yet methods like Jacobiconv(Wang & Zhang, 2022), Bernnet (He et al., 2021), and Chebynet (Defferrard et al., 2016), which use low-degree polynomials of the normalized adjacency matrix $\hat{\boldsymbol{A}}$, do not adequately capture these properties.

Theorem 2 (Banerjee (2008))**.** *Let J^H be the graph obtained from some graph J by adding a subgraph H with eigenvalue 0. A subgraph H is a graph consisting of a subset of nodes $\{q_1, q_2, \ldots, q_m\}$ of J and the corresponding edges between them. In this construction, each node $q_i \in V(H)$ is connected to every node $r \in V(J) \setminus V(H)$ that is a neighbor of some node $r_i \in V(J)$.*

Then, the graph J^H has an eigenvalue 0 with an associated eigenvector(u) that is nonzero only at the nodes r_i and q_i. Furthermore $u_{r_i} = -u_{q_i}$.

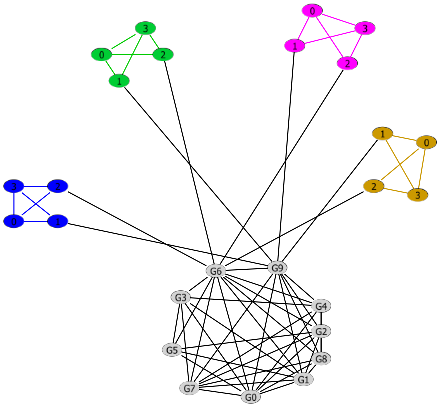


Figure 6: Simple graph with duplicate subgraphs. After adding the duplicates, the multiplicity of eigenvalue 0 increases from 0 to 3.

Theorem 2 reveals that when a graph contains duplicate substructures, it leads to eigenvalue 0 with eigenvector localized to specific node sets. This localization property is particularly relevant for node classification and community detection tasks, as nodes with similar structural roles often share the same labels (Figure 6).

In analyzing how the negative and positive parts of $\boldsymbol{T}_k$ affect and change the structure of the graph, we consider a simple graph with duplicate subgraphs, as illustrated in Figure 6. In this graph, nodes with the same labels are duplicates. According to Theorem 2, these duplicates create eigenvalues equal to 0 in the eigendecomposition of the normalized Adjacency matrix. Let $\boldsymbol{U}_0$ denote the eigenvectors corresponding to eigenvalue 0. We can decompose $\boldsymbol{R}_0 = \boldsymbol{U}_0\boldsymbol{U}0^\top$ into its negative ($\boldsymbol{R}_0^-$) and positive parts ($\boldsymbol{R}_0^+$). Using these parts, we construct two graphs by choosing the edges with the highest score in these matrices. The graphs resulting from this process, including original and added edges, are shown in Figures 7 and 8. From these graphs, we observe that both negative and positive edges identify connections between duplicate motifs. The negative edges also reveal connections between duplicate nodes within these duplicate motifs, highlighting their role in capturing structural similarities. Another intuition to split $\boldsymbol{T}_k$ is that for example the second eigenvector provides a direction that best separates the graph into two groups while minimizing

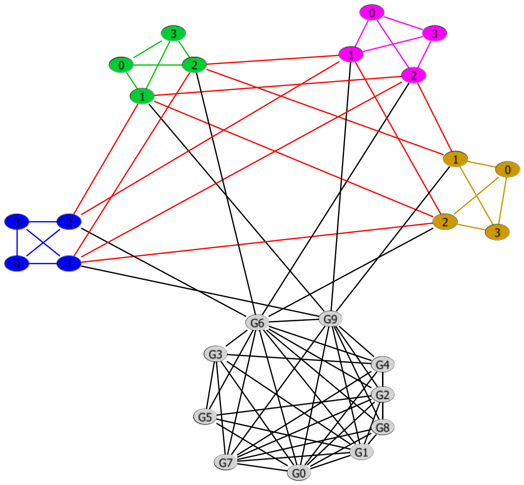


Figure 7: Simple graph with added positive edges.

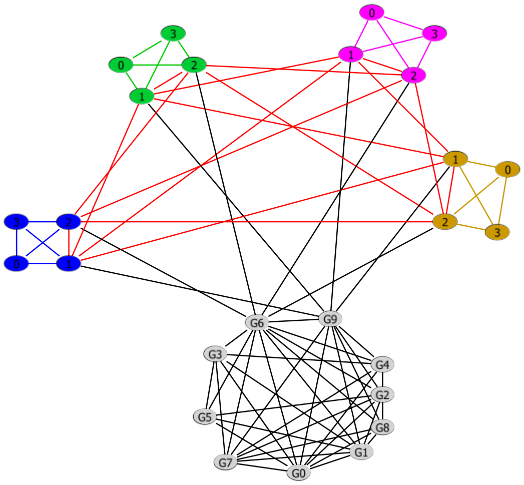


Figure 8: Simple graph with added negative edges.

Table 4: Hyperparameter ranges used for optimization.

Hyperparameter	**Values**
Learning Rate (`lr`)	0.0005, 0.001, 0.005, 0.01, 0.05
Weight Decay (`weight_decay`)	0.0, 5e-5, 1e-4, 5e-4, 1e-3
Feature Dropout (`feat_dropout`)	0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
Number of Layers (`nlayer`)	1, 2, 3, 4, 5
Hidden Dimension (`hidden_dim`)	16, 32, 64
Average Length for Algorithm 1(`average_length`)	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
Number of limits (`num_limits`)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

the connections cut between them. The positive and negative values show two clusters that are internally connected but separated from each other (Fiedler, 1973; Luxburg, 2007).

B Hyperparameter Optimization and Running Environment

All experiments were carried out on a Linux machine with an NVIDIA A100 GPU, Intel Xeon Gold 6230 CPU (20 cores @ 2.1GHz), and 24GB RAM. Hyperparameter tuning was performed using the Hyperopt Tree of Parzen Estimators (TPE) algorithm (Bergstra et al., 2011) with the hyperparameter ranges shown in Table 4.

The Adam optimizer was used for training with 2000 epochs. Hyperparameters were selected to achieve the best performance on a validation set.