A Piece-wise Polynomial Filtering Approach for Graph Neural Networks

⌧Vijay Lingam[⋆], Manan Sharma[⋆], Chanakya Ekbote[⋆], Rahul Ragesh, Arun Iyer, and Sundararajan Sellamanickam

Microsoft Research India
{vijaylingam0810,manan2908}@gmail.com, {t-cekbote, rahulragesh,ariy,ssrajan}@microsoft.com

Abstract. Graph Neural Networks (GNNs) exploit signals from node features and the input graph topology to improve node classification task performance. Recently proposed GNNs work across a variety of homophilic and heterophilic graphs. Among these, models relying on polynomial graph filters have shown promise. We observe that polynomial filter models need to learn a reasonably high degree polynomials without facing any over-smoothing effects. We find that existing methods, due to their designs, either have limited efficacy or can be enhanced further. We present a spectral method to learn a bank of filters using a piece-wise polynomial approach, where each filter acts on a different subsets of the eigen spectrum. The approach requires eigendecomposition only for a few eigenvalues at extremes (i.e., low and high ends of the spectrum) and offers flexibility to learn sharper and complex shaped frequency responses with low-degree polynomials. We theoretically and empirically show that our proposed model learns a better filter, thereby improving classification accuracy. Our model achieves performance gains of up to ∼6% over the state-of-the-art (SOTA) models while being only ∼2x slower than the recent spectral approaches on graphs of sizes up to ∼169K nodes.

Keywords: Graph Neural Networks · Representation Learning · Polynomial Filtering.

1 Introduction

We are interested in the problem of classifying nodes in a graph where a graph with features for all nodes, and labels for a few nodes are made available for learning. Inference is done using the learned model for the remaining nodes (*aka* transductive setting). Graph Neural Networks (GNNs) perform well on such problems [1]. Most GNNs predict a node's label by aggregating information from its neighbours in a certain way, making them dependent on some correlation between the structure and the node labels[1]. For example, in the simplest case,

[⋆] Equal contribution. Work done while author was at Microsoft Research India

[1] Characterizing the correlation between the graph structure and node features/labels is an active area of research. Several metrics have been proposed including edge

GNNs work well when the node and its neighbours share similar labels. However, the performance can be poor if this criterion is not satisfied. Recently, several modeling approaches have been proposed to build/learn robust GNN models. Some modify the aggregation mechanism [4,5,3], while others propose to estimate and leverage the label-label compatibility matrix as a prior [6].

More recent approaches have tackled this problem from a graph filter learning perspective [7,8,38,37,32,39]. With eigenvalues having frequency interpretations [26], one or more filters (i.e., a bank of filters) that selectively accentuates and suppresses various spectral components of graph signals are learned using task-specific available information. The filtering operation enables learning better node representation which translates to improved classification accuracy.

Designing effective graph filters is a challenging problem, and most recent methods [10,8,38,37] suggest interesting ways to learn polynomial filters having finite impulse response (FIR) characteristics. These models are efficient and attractive, as they make use of local neighborhood (i.e., using sparse adjacency matrix repeatedly) and do not require to pre-compute eigendecomposition, which is expensive (when done over the entire spectrum, i.e., for all eigenpairs). Though these models are able to learn better filters and give good performance gains, they are still unable to learn richer and complex frequency responses, which require higher-order polynomials. One key reason for their inability to learn effective high-order polynomials is that they only *mitigate* the over-smoothing problem. This aspect of the problem becomes clear when we analyze a general class of FIR filters (GFIR) and find that the over-smoothing problem exists for the whole class, of which simplified GCN [16], GPR-GNN [8] and several other models are special cases. We also find that while constraining the model space of GFIR (e.g., [8]) helps to mitigate over-smoothing, it is still unable to learn complex-shaped and sharper frequency responses. Considering this background, our interest lies in learning a bank of effective filters in spectral domain to model complex shaped frequency responses, as needed for graphs with diverse label correlations. Our contributions are:

1. We propose a novel piece-wise polynomial filtering approach to learn a filter bank tuned for the task at hand. Since full eigendecomposition is expensive, we present an efficient method that makes use of only a few extremal eigenpairs and leverages GPR-GNN to learn multiple filters. (While computing the extremal eigenpairs does lead to an increased computational cost, we show in A.9 that such a cost is indeed managable, i.e. the model is only $\sim$2x slower than recent spectral SOTA methods.)
2. We analyze, theoretically and experimentally, the shortcomings of a general class of FIR (GFIR) filters. We show that the proposed piece-wise polynomial GNN (PP-GNN) solution is more expressive and is capable of modeling richer and complex frequency responses.

homophily [13,5], node homophily [4], class homophily [27]. All these metrics show that standard GNNs perform well when the graphs and node labels are positively correlated.

3. We conduct a comprehensive experimental study to compare PP-GNN with a wide range of methods ($\sim$20), covering both spatial and spectral convolution based methods on nearly a dozen datasets. Experimental results show that PP-GNN performs significantly better, achieving up to $\sim$6% gains on several datasets.

2 Related Work

Graph Neural Networks (GNNs) have become increasingly popular models for semi-supervised classification with graphs. [11] set the stage for early GNN models, which was then followed by various modifications [12,1,9,2] and improvements along with several different directions such as improved aggregation and attention mechanisms [9,2,3], efficient implementation of spectral convolution [12,16], incorporating random walk information [15,13,14], addressing over-smoothing [15,10,13,14,28,29,30], etc.

Another line of research explored the question of where GNNs help. The key understanding is that the performance of GNN is dependent on the correlation of the graphs with the node labels. Several approaches [13,5,31] considered edge homophily and proposed a robust GNN model by aggregating information from several higher-order hops. [3] also considered edge homophily and mitigated the issue by learning robust attention models. [4] talks about node homophily and proposes to aggregate information from neighbours in the graph and neighbours inferred from the latent space. [6] proposes to estimate label-label compatibility matrix and uses it as a prior to update posterior belief on the labels.

Recent approaches motivated by the developments in graph signal processing [25], focus on learning graph filters with filter functions that operate on the eigenvalues of the graph directly or indirectly, adapting the frequency response of graph filters for the desired task. [7] models the filter function as an attention mechanism on the edges, which learns the difference in the proportion of low-pass and high-pass frequency signals. [8] proposes a polynomial filter on the eigenvalues that directly adapts the graph for the desired task. [32] decompose the graph into low-pass and high-pass frequencies, and define a framelet based convolutional model. [38] propose to learn graph filters using Bernstein approximation of arbitrary filtering function. [37] suggest to learn adaptive graph filters for different feature channels and frequencies by stacking multiple layers. Our work is closely related to these lines of exploration. All these works still need high-degree polynomials when sharper frequency responses are needed; however, though improved performance is observed and over-smoothing is mitigated, further improvements seem possible. Another class of Infinite Impulse Response (IIR) filters have been proposed to learn complex filter responses. ARMA [39] achieves this by using auto-regressive moving average, but empirically have been found to have limited effectiveness. Implementing precise ARMA filters for graphs is a challenging problem and has high computation costs. [39] proposes several approximations to mitigate the issues, but these come with limited efficacy. In our work, we propose to learn a filter function as a sum of polynomials over different subsets of the eigenvalues (in essence, a bank of filters) by operating directly in the spectral domain, enabling design of effective filters to model task-specific complex frequency responses with compute trade-offs.

3 Problem Setup and Motivation

We focus on the problem of semi-supervised node classification on a simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges. Let $\mathbf{A} \in \{0,1\}^{n \times n}$ be the adjacency matrix associated with $\mathcal{G}$, where $n = |\mathcal{V}|$ is the number of nodes. Let $\mathcal{Y}$ be the set of all possible class labels. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the d-dimensional feature matrix for all the nodes in the graph. Given a training set of nodes $\mathcal{D} \subset \mathcal{V}$ whose labels are known, along with $\mathbf{A}$ and $\mathbf{X}$, our goal is to predict the labels of the remaining nodes. Let $\mathbf{A_I} = \mathbf{A} + \mathbf{I}$ where $\mathbf{I}$ is the identity matrix. Let $\mathbf{D_{A_I}}$ be the degree matrix of $\mathbf{A_I}$ and $\widetilde{\mathbf{A}} = \mathbf{D}_{\mathbf{A_I}}^{-1/2}\mathbf{A_I}\mathbf{D}_{\mathbf{A_I}}^{-1/2}$. Let $\widetilde{\mathbf{A}} = \mathbf{U\Lambda U}^T$ be the eigendecomposition. The spectral convolution of $\mathbf{X}$ on the graph $\mathbf{A}$ can be defined via the reference operator $\widetilde{\mathbf{A}}$ and a general Finite Impulse Response (FIR) filter ([40]), parameterized by $\mathbf{\Theta}$ as:

$$\mathbf{Z} = \sum_{j=1}^{k} \widetilde{\mathbf{A}}^j \mathbf{X} \mathbf{\Theta}_j \tag{1}$$

The term, $\widetilde{\mathbf{A}}^j\mathbf{X}$ uniformly converges to a stationary value as the value of j increases, making the node features indistinguishable (often referred to as the problem of over-smoothing), thereby reducing the importance of the corresponding term for the task at hand. We formalize the argument via commenting on the Dirichlet energy of the higher-order terms [41]. Dirichlet energy reveals the embedding smoothness with the weighted node pair distance. A smaller value is highly related to over-smoothing [42]. Under some conditions, the upper bound of Dirichlet energy of higher terms is theoretically proved to converge to 0 in the limit of infinite layers. In other words, all nodes converge to a trivial fixed point in the embedding space and hence do not contribute to the discriminative signals. This is formalized as follows:
Proposition 3.1: The upper bound of Dirichlet energy for the higher-order terms in the general FIR model exponentially decreases to 0 with the order, k. Formally, with $\mathbf{S}$ as any graph shift operator (in our case, the normalized adjacency), and $\mathbf{\Theta}_k$ be the set of parameters, indexed by k:

$$E(\mathbf{S}^k\mathbf{X}\mathbf{\Theta}_k) \leq (1-\lambda)^{2k} s_{\Theta_k} E(\mathbf{X})$$

where, λ is the positive eigenvalue of the graph Laplacian Δ that is closest to 0; s_{Θ_k} is the largest singular value of $\mathbf{\Theta}_k$. We relegate the proof of the corollary as well as the formal definition of a few terms in section A.3 of the supplementary material.

The family of general FIR filters is ubiquitous and gives rise to various other filter families (eg. polynomial) simply by placing constraint on the form of parameterization. We experiment with placing simple constraints on the bare GFIR model in section A.7 of supplementary and observe that while constraining helps improving the performance, it does not help in learning complex responses. It is not difficult to see that the models of [12],[8], etc. are just instantiations of the GFIR family. Particularly, by restricting $\mathbf{\Theta}_j = \alpha_j \mathbf{I}$, we recover the linear model (without MLP) of [8], which can now be interpreted as the polynomial filter function h operating on the eigenvalues, in the Fourier domain [25,8] as,

$$\mathbf{Z} = \sum_{j=1}^{k} \alpha_j \widetilde{\mathbf{A}}^j \mathbf{X} = \mathbf{U} h(\mathbf{\Lambda}) \mathbf{U}^T \mathbf{X} \tag{2}$$

with $h : \mathbb{R} \to \mathbb{R}$ is defined as $h(\lambda;\ \boldsymbol{\alpha}) = \sum_{i=1}^{k} \alpha_i \lambda^i$ where α_i's are coefficients of the polynomial, k is the order of the polynomial and λ is any eigenvalue from $\mathbf{\Lambda}$. $h()$ is applied element-wise across $\mathbf{\Lambda}$ in Eq.2. In this process, the filter function is essentially adapting the graph for the desired task at hand.

It is well-known that polynomial filters can approximate any graph filter [26,25]. Since polynomial filters are a class of the GFIR filter family, they inherit the same problem of over-smoothing as the order of the polynomial becomes higher. [8] show that they achieve the diminishing of the contribution of higher-order terms by showing that their coefficients converge to zero during training. While this mitigates the over-smoothing problem, use of lower-order polynomials results in an imprecise approximation when the dataset requires a complex spectral filter for obtaining a superior performance, which we will show is the case for certain datasets (See Figure 1 and supplementary's A.6). Empirical results demonstrating the key points discussed in this section: a) smoothening of the higher-order terms (can be found in Figure 5a of the supplementary material) and b) their effect on the test performance on a few datasets (can be found in Figure 5b and 5c of supplementary material). These problems indicate the need for a method that can approximate arbitrarily complex filters better and at the same time mitigate the effects of over-smoothing.

4 Proposed Approach

We propose to learn a bank of polynomial filters with each filter operating on different parts of the spectrum, taking task-specific requirements into account. We show that our proposed filter design can approximate the latent optimal graph filter better than a single polynomial, and the resultant class of learnable filters is richer.

4.1 Piece-wise Polynomial Filters

We start with the expression (2) for node embedding rewritten with an MLP network transforming input features, $\mathbf{X}$, :

$$\mathbf{Z} = \sum_{i=1}^{n} h(\lambda_i) \mathbf{u}_i \mathbf{u}_i^T \mathbf{Z}_x(\mathbf{X}; \mathbf{\Theta}) \tag{3}$$

where $\mathbf{u}_i$ is the eigenvector corresponding to the eigenvalue, λ_i, and $\mathbf{Z}_x(\mathbf{X};\mathbf{\Theta})$ is an MLP network with parameters $\mathbf{\Theta}$. Our goal is to learn a filtering function, $h(\lambda)$ jointly with MLP network, using which we compute the node embedding, $\mathbf{Z}$. We model $h(\lambda)$ as a piece-wise polynomial or spline function where each polynomial is of a lower degree (e.g., a cubic polynomial). We partition the spectrum in $[-1,1]$ (or $[0,2]$ as needed) into contiguous intervals and approximate the desired frequency response by fitting a low degree polynomial in each interval. This process helps us to learn a more complex shaped frequency response as needed for the task. Let $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ denote a partition of the spectrum, containing m contiguous intervals and $h_{i,k_i}(\lambda;\ \alpha_i)$ denote a k_i-degree polynomial filter function defined over the interval σ_i (and 0 elsewhere) with polynomial coefficients α_i. We define piece-wise polynomial GNN (PP-GNN) filter function as:

$$h(\lambda) = \sum_{\sigma_i \in \mathcal{S}} h_{i,k_i}(\lambda;\ \alpha_i) \tag{4}$$

and learn a smooth filter function by imposing additional constraints to maintain continuity between polynomials of contiguous intervals at different endpoints (*aka* knots). This class of filter functions is rich, and its complexity is controlled by choosing intervals (i.e., endpoints and number of partitions) and polynomial degrees. Given the filter function, we compute the PP-GNN node embedding matrix as:

$$\mathbf{Z} = \sum_{\sigma_i \in \mathcal{S}} \mathbf{U}_i h(\lambda_{\sigma_\mathrm{i}}) \mathbf{U}_i^T \mathbf{Z}_x(\mathbf{X};\mathbf{\Theta}) \tag{5}$$

where $\mathbf{U}_i$ is a matrix with columns as eigenvectors corresponding to eigenvalues that lie in σ_i and $h(\lambda_\sigma)$ is the diagonal matrix with diagonals containing the h_i evaluated at the eigenvalues lying in σ_i. Thus, the node embedding, $\mathbf{Z}$, is computed as a sum of outputs from a bank of polynomial filters with each filter operating over a spectral interval, σ_i.

4.2 Practical and Implementation Considerations

The filter function (5) requires computing full eigendecomposition of $\widetilde{\mathbf{A}}$ and is expensive, therefore, not scalable for very large graphs. We address this problem by performing eigendecomposition only for a few extreme values (i.e., at low and high ends of the spectrum) for sparse matrices, for which efficient algorithms exist [43] with corresponding off-the-shelf implementations. The primary motivation is that many recent works including GPR-GNN investigated the problem of designing robust graph neural networks that work well across homophilic and heterophilic graphs, and, they found that graph filters that amplify or attenuate low and high-frequency components of signals (i.e., low-pass and high-pass filters) are critical to improving performance on several benchmark datasets. However, there is still one question: *how do we extract signals from the remaining (middle) portion of the spectrum, and that too efficiently?* We answer this question as follows. Using the observation that the GPR-GNN method learns a graph filter but operates on the entire spectrum by sharing the filter coefficients across the spectrum, our proposal is to use an efficient variant of (4) as:

$$\tilde{h}(\lambda) = \eta_l \sum_{\sigma_i \in \mathcal{S}^l} h_i^{(l)}(\lambda; \gamma_i^{(l)}) + \eta_h \sum_{\sigma_i \in \mathcal{S}^h} h_i^{(h)}(\lambda; \gamma_i^{(h)}) + \eta_{gpr} h_{gpr}(\lambda; \gamma) \tag{6}$$

where $\mathcal{S}^l$ consists of partitions over low-frequency components, $\mathcal{S}^h$ consists of partitions over high-frequency components, the first and second terms fit piece-wise polynomials[2] in low/high-frequency regions, as indicated through superscripts. We refer PP-GNN models using only filters corresponding to the first and second terms alone in (6) as PP-GNN (Low) and PP-GNN (High), respectively. We extract any useful information from other frequencies in the middle region by adding the GPR-GNN filter function, $h_{gpr}(\lambda; \gamma)$ (the final term in 6), which is computationally efficient. Since $h_{gpr}(\lambda; \gamma)$ is a special case of (4) and the terms in (6) are additive, it is easy to see that (6) is same as (4) with a modified set of polynomial coefficients. Furthermore, we can control the contributions from each term by setting or optimizing over hyperparameters, η_l, η_h and η_{gpr}. Thus, the proposed model offers richer capability and flexibility to learn complex frequency response and balance computation costs over GPR-GNN. Please see Section A.3 for implementation details.

Model Training. Like GPR-GNN, we apply Softmax activation function on (5) and use the standard cross-entropy loss function to learn the sets of polynomial coefficients (γ) and classifier model parameters ($\mathbf{\Theta}$) using labeled data. To ensure smoothness of the learned filter functions, we add a regularization term that penalizes squared differences between the function values of polynomials of contiguous intervals at each other's interval end-points. More details can be found in the supplementary material (A.3).

Discussion. In our model (4), we alleviate the over-smoothing problem using low-order polynomials, and learning complex and sharper frequency responses is feasible as we approximate higher-order polynomial functions effectively using several low-order piece-wise polynomials. However, this comes with eigen-decomposition compute cost for a few (k) extreme eigenvalues, but is controllable by choosing k in an affordable way[3]. We observe this cost is (one time) pre-training cost and can be amortized over multiple rounds of model training required for the optimization of hyperparameters. Also, we need to compute each filter specific embedding with non-local eigen-graphs (via the operations, $\mathbf{U}_i H_i(\gamma_\mathbf{i}) \mathbf{U}_i^T \mathbf{Z}_x(\mathbf{X}; \mathbf{\Theta})$); thus, we lose (spatial) local neighborhood property of conventional methods like GPR-GNN. We compute node embeddings afresh whenever the model parameters are updated, thereby incurring an additional cost (over GPR-GNN) of $O(nkL)$ where k and L denote the number of selected low/high eigenvalues and classes, respectively. We conduct a comprehensive ex-

[2] For brevity, we dropped the polynomial degree dependency.

[3] Most algorithms for this task utilize Lanczos' iteration, convergence bounds of which depends on the input matrix' spectrum [34,35], which although have superlinear convergence, but are observed to be efficient in practice.

perimental study to assess the time taken by our method, compare against other state-of-the-art methods and present our findings in the experiment section.

4.3 Analysis

This section is arranged as follows: (a) Theorem 1 establishes superior capabilities of our model in approximating arbitrary filters than a standard polynomial filter; (b) Theorem 2 demonstrates the new space of filters that our model learns from, each region of which induces a controllable, strong bias towards certain parts of the spectrum while at the same time has dimension of the same order as the corresponding polynomial family.

Theorem 1. *For any frequency response h^*, and an integer $K \in \mathbb{N}$, let $\tilde{h} := h + h_f$, with h_f having a continuous support over a subset of the spectrum, σ_f. Assume that h and h_f are parameterized by independent K and K'-order polynomials, p and p_f, respectively, with $K' \leq K$. Then there exists $\tilde{h}$, such that $\min \|\tilde{h} - h^*\|_2 \leq \min \|h - h^*\|_2$, where the minimum is taken over the polynomial parameterizations. Moreover, for multiple polynomial adaptive filters $h_{f_1}, h_{f_2}, ..., h_{f_m}$ parameterized by independent K'-degree polynomials with $K' \leq K$ but having disjoint, contiguous supports, the same inequality holds for $\tilde{h} = h + \sum_{i=1}^{m} h_{f_i}$.*

For a detailed proof please refer to A.3 of the supplementary. We also conducted an experiment to illustrate the main conclusion of the above theorem in Section A.2 of the supplementary material.

Next, we note that since an actual waveform is not observed in practice and instead, we estimate it by optimizing over the observed labels via learning a graph filter, we theoretically show that the family of filters that we learn is a strict superset of the polynomial filter family. The same result holds for the families of the resulting adapted graphs.

Theorem 2. *Define $\mathbb{H} := \{h(\cdot) \mid \forall$ possible K-degree polynomial parameterizations of $h\}$ to be the set of all K-degree polynomial filters, whose arguments are $n \times n$ diagonal matrices, such that a filter response over some $\mathbf{\Lambda}$ is given by $h(\mathbf{\Lambda})$ for $h(\cdot) \in \mathbb{H}$. Similarly $\mathbb{H}' := \{\tilde{h}(\cdot) \mid \forall$ possible polynomial parameterizations of $\tilde{h}\}$ is set of all filters learnable via* PP-GNN *, with $\tilde{h} = h + h_{f_1} + h_{f_2}$, where h is parameterized by a K-degree polynomial supported over entire spectrum, h_{f_1} and h_{f_2} are localized adaptive filters parameterized by independent K'-degree polynomials which only act on top and bottom t diagonal elements respectively, with $t < n/2$ and $K' \leq K$; then $\mathbb{H}$ and $\mathbb{H}'$ form a vector space, with $\mathbb{H} \subset \mathbb{H}'$. Also, $\frac{dim(\mathbb{H}')}{dim(\mathbb{H})} = \frac{K+2K'+3}{K+1}$.*

Corollary 1. *The corresponding adapted graph families $\mathbb{G} := \{\mathbf{U}h(\cdot)\mathbf{U}^T | \forall h(\cdot) \in \mathbb{H}\}$ and $\mathbb{G}' := \{\mathbf{U}\tilde{h}(\cdot)\mathbf{U}^T | \forall \tilde{h}(\cdot) \in \mathbb{H}'\}$ for any unitary matrix $\mathbf{U}$ form a vector space, with $\mathbb{G} \subset \mathbb{G}'$ and $\frac{dim(\mathbb{G}')}{dim(\mathbb{G})} = \frac{K+2K'+3}{K+1}$.*

The above theorem can be trivially extended to an arbitrary number of adaptive filters with arbitrary support. The presence of each adaptive filter induces a bias in the model towards learning a bank of filters that operate only on the corresponding support. Since the number of filters and their support sizes are hyperparameters, tuning them offers control and flexibility to model richer frequency responses over the entire spectrum. Thus, our model learns from a more diverse space of filters and the corresponding adapted graphs. The result also implies that our model learns from a space of filters that is only $O(1)$-fold greater than that of polynomial filters[4]. Note that learning from this diverse region is feasible. This observation comes from the proofs of Theorem 4.2 and Corollary 4.2.1 (A.3 and A.3 in supplementary). Using the localized adaptive filters without any filter with the entire spectrum as support results in learning a set of adapted graphs, $\hat{\mathbb{G}}$. This set is disjoint from $\mathbb{G}$, with $\mathbb{G}' = \mathbb{G} \oplus \hat{\mathbb{G}}$. We conduct various ablative studies where we demonstrate the effectiveness of learning from $\hat{\mathbb{G}}$ and $\mathbb{G}'$.

Our model formulation is a generalization of the formulation by [8], and we show in Section A.3 of the supplementary material by extending their analysis to our model that it still inherits their property of mitigating oversmoothing effects when high degree polynomial is used. Our experiments show that we are able to obtain superior performance without needing the higher-order polynomials.

4.4 Comparison against other Filtering Methods

General FIR filter are a generalization of the polynomial filter family and thus a precursor to the models based on the latter. As per the study conducted in section A.7 of the supplementary, constraining the model is required to obtain better performance. Restricting to polynomial filters can be seen as having an implicit regularization on the same and we also empirically observe that such a restriction (restricting to polynomial filters) gives much better performance than constraining GFIR (see 5.1 and A.7) by simpler regularization methods such as L2 and/or dropout. We have also shown in theorem 2 that PP-GNN increases the space of graph filters (over GPR-GNN) and we observe in 5.1 that this increase in graph space results in an increased performance, over other polynomial filter methods. Thus, it requires a careful balance of the constraints imposed on the filter family, while also appropriately increasing the graph space to obtain better performance. A comprehensive study of this balance is beyond the scope of this work and we leave that as future work. Below, we first show the different ways of constraining the space (via polynomial filters) and compare them against PP-GNN.

Polynomial filters are a class of filters constructed and evaluated from polynomials. These filters can be constructed via multiple bases (for instance monomial, Bernstein) in the polynomial vector space. APPNP, GPR-GNN, and BernNet are all instances of polynomial graph filters defined in different bases

[4] We leave the formal bias-variance analysis for adapted graph families as future work.

and with different constraints. Below, we illustrate the differences between these three methods and also discuss the shortcomings of each of them.

APPNP: One of the early works, APPNP [10], can be interpreted as a fixed polynomial graph filter that works with monomial basis. The polynomial coefficients correspond to Personalised PageRank (PPR) [44]. The node embeddings are learnt by APPNP as described in A.8. The main shortcoming of this method is the assumption that the optimal coefficients for the polynomial filter (for all tasks) are PPR coefficients, which need not necessarily be the case.

GPR-GNN: GPR-GNN builds on APPNP by overcoming this shortcoming by making the coefficients γ_k (see A.8) learnable. [8] identified that negative coefficients allows the model to exploit high frequency signals required for better performance on heterophilic graphs. GPR-GNN, like APPNP, uses the monomial basis. The node embeddings are learnt by GPR-GNN as described in A.8. While this method is an improvement over APPNP, adapting an arbitrary filter response which requires a high-order polynomial is difficult due to the oversmoothing problem. GPR-GNN mitigates oversmoothing by showing that the higher order terms' coefficients uniformly converge to zero during training. Mitigating the oversmoothing problem limits the complexity of the filter learnt, and therefore making GPR-GNN ineffective at learning complex frequency responses.

BernNet: While oversmoothing is one shortcoming of GPR-GNN, BernNet identified another shortcoming that GPR-GNN and other polynomial filtering based methods can result in ill-posed solutions and face optimization issues (converging to saddle points) by not constraining the filter response to non-negative values. [38] proposed a model that learns a non-negative frequency response, a constraint that can be easily enforced by modifying the learning problem from learning the coefficients of the monomial basis functions to learning the coefficients of the Bernstein basis functions, since the latter are non-negative in their standard domain. [38] argue that constraining coefficients to take on non-negative values is required for stability and interpretability of the learned filters and is the main reason for performance improvements. The node embeddings are learnt as described in A.8. Note that in the expression referenced, $\theta_k(\forall k)$ are learnable coefficients and are constrained to non-negative values. We first replace $\frac{1}{2^K}\binom{K}{r}\sum_{p=0}^{q}\binom{K-r}{q-p}\binom{r}{p}(-1)^p$ with α_{rq} and then subsequently replace $\sum_{r=0}^{K}\theta_r\alpha_{rq}$ with w_q. Such an exercise was done to show that the filter defined by BernNet does indeed fall into the class of polynomial filters. We tabulate the important attributes of each of the polynomial filters described above in Table 11 of the supplementary material.

All of these approaches run into the oversmoothing issue with an increase in the degree of the polynomial filter (A.1 of supplmentary). PP-GNN, owing to its piece-wise definition, can model more complex shaped responses better without the need to increase the degree. Our proposed model only requires extremal eigendecomposition (i.e. computing only the extreme eigenpairs), for which there exists efficient algorithms to compute [45,46]. Further, as mentioned earlier, this is a one time pre-training cost, that can be amortized over training multiple models for hyper-parameter tuning. We illustrate this through a comprehensive empirical study in section A.9 of the supplementary material. In the next section, we experimentally show the benefits of PP-GNN.

5 Experiments

We conduct extensive experiments to demonstrate the effectiveness and competitiveness of the proposed method over standard baselines and state-of-the-art (SOTA) GNN methods. We conduct ablative studies to demonstrate the usefulness of different filters and the number of eigenpairs required in PP-GNN. We also compare the quality of the embeddings learned and the time to train different models. We first describe our experimental setup along with baselines and information on hyper-parameter tuning.

We evaluate our model on several real-world heterophilic and homophilic datasets. We resort detailed descriptions of dataset statistics, preprocessing steps, and baselines to the Appendix (A.4). We report the mean and standard deviation of test accuracy over splits to compare model performance.

5.1 PP-GNN versus SOTA Models

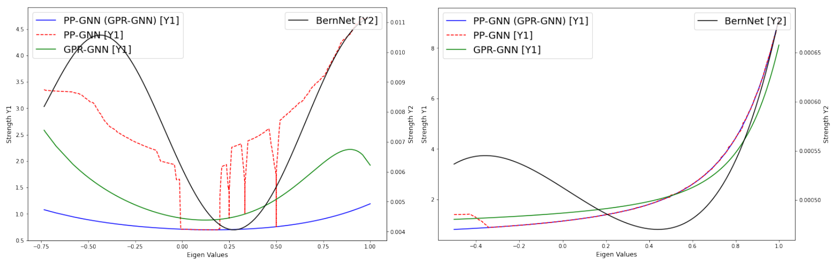
Table 1: Results on a few heterophilic and homophilic datasets. GFIR-1 corresponds to unconstrained setting. GFIR-2 corresponds to constrained setting. For a more detailed comparison and description please refer to Appendix A.5

	Squirrel	**Chameleon**	**Cora**	**Computer**	**Photos**
GFIR-1	36.50±1.12	51.71±3.11	87.93±0.90	78.39±1.09	89.26±1.00
GFIR-2	41.12±1.17	61.27±2.42	87.46±1.26	79.57±2.12	89.38±1.03
FAGCN [7]	42.59±0.79	55.22±3.19	88.21±1.37	82.16±1.48	90.91±1.11
APPNP [10]	39.15±1.88	47.79±2.35	88.13±1.53	82.03±2.04	91.68±0.62
LGC [22]	44.26±1.49	61.14±2.07	88.02±1.44	83.44±1.77	91.56±0.74
GPRGNN [8]	46.31±2.46	62.59±2.04	87.77±1.31	82.38±1.60	91.43±0.89
AdaGNN [37]	53.50±0.96	65.45±1.17	86.72±1.29	81.27±2.10	89.93±1.22
BernNET [38]	52.56±1.69	62.02±2.28	88.13±1.41	83.69±1.99	91.61±0.51
ARMA [39]	47.37±1.63	60.24±2.19	87.37±1.14	78.55±2.62	90.26±0.48
UFG [32]	42.06±1.55	56.29±1.58	87.93±1.52	80.01±1.78	90.20±1.41
PP-GNN	**59.15±1.91**	**69.10±1.37**	**89.52±0.85**	**85.23±1.36**	**92.89±0.37**

Heterophilic Datasets. We perform comprehensive experiments to show the effectiveness of PP-GNN on several Heterophilic graphs and tabulate the results in Table 5 in the Appendix (A.5). Datasets like Texas, Wisconsin, and Cornell contain graphs with high levels of Heterophily and *rich node features*. Standard non-graph baselines like LR and MLP perform competitively or better on these datasets compared to many spatial and spectral-based methods. PP-GNN offers significant lifts in performance with gains of up to $\sim$6%. The node features in datasets like Chameleon and Squirrel are not adequately discriminative, and significant improvements are possible via convolutions, as we compare non-graph and graph-based methods in Table 5. Spatial GNN methods, in general, offer improvements over non-graph counterparts. In specific, methods like GCN, which also have a spectral connotation, show better performance on these datasets. We observe from the Table that Spectral methods offer additional improvements over models like GCN. The difference in performance among spectral methods majorly comes from their ability to learn better frequency responses of graph filters. Our proposed model shows significant lifts over all the baselines with gains up to ∼6% and ∼4% on the Squirrel and Chameleon datasets. These improvements empirically support the efficacy of PP-GNN's filter design.

Homophilic Datasets. The input graphs for these datasets contain informative signals, and one can expect competitive task performance from even basic spatial-convolution based methods as observed in Table 7 present in Appendix (A.5). We can see that spatial models are among the top performers for several Homophilic datasets. Existing spectral methods marginally improve over spatial methods on a few datasets. Not surprisingly, our PP-GNN model with effective filter design can exploit additional discriminatory signals from an already rich informative source of signals. PP-GNN offers additional gains up to 1.3% over other baselines.

Due to space constraints we have shown a small subset of our results in Table 1. For a more detailed comparison please refer to the Appendix A.5 section, where we compare against more SOTA methods and on other datasets as well.



(a) Squirrel (b) Citeseer

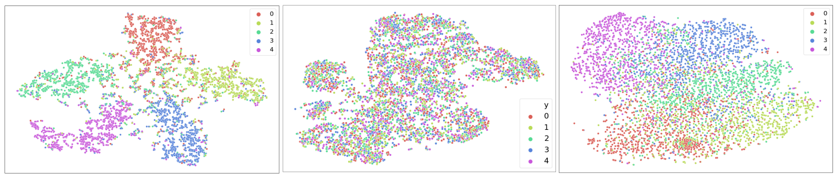
Fig. 1: Learned filter responses of PP-GNN,GPR-GNN, and BernNet.

5.2 PP-GNN Model Investigation

We conducted several experimental studies to understand and illustrate how the PP-GNN model works. Our studies include: (a) how does the frequency response of PP-GNN look like?, (b) what happens when we learn only individual sub-filter banks (e.g., PP-GNN (Low), PP-GNN(Low + GPR-GNN)? and (c) does PP-GNN learn better embeddings?

Frequency Response. In Figure 1a and 1b, we show the learned frequency responses (i.e., $h(\lambda)$) of the overall PP-GNN model, GPR-GNN component of PP-GNN (PP-GNN (GPR-GNN)), stand-alone (GPR-GNN) model and BernNet model on the Squirrel and Citeseer datasets. For Squirrel (a heterophilic dataset), we can observe that while GPR-GNN and BernNet learns the importance of low and high-frequency signals, it is unable to capture their relative strengths/importance adequately, and this happens due to the restriction of learning a single polynomial globally. PP-GNN learns sharper and richer responses at different parts of the spectrum, thereby improving classification accuracy. For Citeseer (a homophilic dataset) we can observe that all the models in comparison learn a smooth polynomial, GPR-GNN is not able to capture the complex transition that can be seen at the lower end of the spectrum, while BernNet is doing it some degree. This inability to capture the complex transition leads to a lower classification accuracy. A similar trend can be found on two other datasets in A.6.

Quality of learned embeddings: We qualitatively assess the difference in the learned embedding of PP-GNN, GPR-GNN and BernNet. Towards this, we generated t-SNE plots of the learned node embeddings and visually inspected them. From Figure 2a, 2b and 2c, we observe that PP-GNN discovers more discriminative features resulting in discernible clusters on the Squirrel dataset compared to GPR-GNN and BernNet, enabling PP-GNN to achieve significantly improved performance.



(a) PP-GNN (b) GPRGNN (c) BernNet

Fig. 2: t-SNE plots of learned embeddings on the Squirrel dataset

5.3 Additional Experiments

We summarize a list of experiments that can be found in the supplementary material. Studies on varying the number of eigenvectors used by PP-GNN and the importance of MLP can be found in Section A.6. Analysis on the effect of varying the order of GPR-GNN's polynomial on performance is presented in A.1. Experimental details for PP-GNN with boundary regularization is in A.3.

6 Conclusion

Several recently proposed methods attempt to build robust models for diverse graphs exhibiting different correlations between graph and node labels. We build on the filter-based approach of GPR-GNN which can be extended further with Generalized FIR models. This work proposed an effective polynomial filter bank design using a piece-wise polynomial filtering approach. We combine GPR-GNN with additional polynomials resulting in a bank of filters that adapt to low and high-end spectrums using multiple polynomial filters. While our method makes an unconventional choice of extremal eigendecomposition, it does help to get improved performance, albeit with some additional but manageable cost. Our experiments demonstrate that the proposed approach can learn effective filter functions that improve node classification accuracy significantly across diverse graphs. While our work shows merit, it is still founded upon the polynomial formulation, and even though piecewise polynomial filters are more expressive than conventional polynomial filters, they still retain the properties of the polynomial filters locally. Hence, there is still room for even more expressive filter formulations that are well motivated, and we leave their exploration as future work.

References

1. Kipf, T. & Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *International Conference On Learning Representations (ICLR).* (2017)
2. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. & Bengio, Y. Graph Attention Networks. *International Conference On Learning Representations (ICLR).* (2018)
3. Kim, D. & Oh, A. How to Find Your Friendly Neighborhood: Graph Attention Design with Self-Supervision. *International Conference On Learning Representations (ICLR).* (2021)
4. Pei, H., Wei, B., Chang, K., Lei, Y. & Yang, B. Geom-GCN: Geometric Graph Convolutional Networks. *International Conference On Learning Representations (ICLR).* (2020)
5. Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L. & Koutra, D. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. *Neural Information Processing Systems (NeurIPS).* (2020)
6. Zhu, J., Rossi, R., Rao, A., Mai, T., Lipka, N., Ahmed, N. & Koutra, D. Graph Neural Networks with Heterophily. *Association For The Advancement Of Artificial Intelligence (AAAI).* (2021)
7. Bo, D., Wang, X., Shi, C. & Shen, H. Beyond Low-frequency Information in Graph Convolutional Networks. *Association For The Advancement Of Artificial Intelligence (AAAI).* (2021)
8. Chien, E., Peng, J., Li, P. & Milenkovic, O. Adaptive Universal Generalized PageRank Graph Neural Network. *International Conference On Learning Representations (ICLR).* (2021)
9. Hamilton, W., Ying, R. & Leskovec, J. Inductive Representation Learning on Large Graphs. *Neural Information Processing Systems (NeurIPS).* (2017)

10. Klicpera, J., Bojchevski, A. & Günnemann, S. Combining Neural Networks with Personalized PageRank for Classification on Graphs. *International Conference On Learning Representations (ICLR).* (2019)
11. Bruna, J., Zaremba, W., Szlam, A. & LeCun, Y. Spectral Networks and Locally Connected Networks on Graphs. *International Conference On Learning Representations (ICLR).* (2014)
12. Defferrard, M., Bresson, X. & Vandergheynst, P. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. *Neural Information Processing Systems (NeurIPS).* (2016)
13. Galstyan, S. MixHop: Higher-Order Graph Convolution Architectures via Sparsified Neighborhood Mixing. *International Conference On Machine Learning (ICML).* (2019)
14. Lee, S. N-GCN: Multi-scale Graph Convolution for Semi-supervised Node Classification. *Conference On Uncertainty In Artificial Intelligence (UAI).* (2019)
15. Li, Q., Han, Z. & Wu, X. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. *Association For The Advancement Of Artificial Intelligence (AAAI).* (2018)
16. Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T. & Weinberger, K. Simplifying Graph Convolutional Networks. *International Conference On Machine Learning (ICML).* (2019)
17. Tang, J., Sun, J., Wang, C. & Yang, Z. Social Influence Analysis in Large-Scale Networks. *ACM SIGKDD International Conference On Knowledge Discovery And Data Mining (KDD).* (2009)
18. Rozemberczki, B., Allen, C. & Sarkar, R. Multi-Scale attributed node embedding. *Journal Of Complex Networks.* (2021)
19. Kingma, D. & Ba, J. Adam: A Method for Stochastic Optimization. *International Conference On Learning Representations (ICLR).* (2015)
20. Chua, T., Tang, J., Hong, R., Li, H., Luo, Z. & Zheng, Y. NUS-WIDE: A Real-World Web Image Database from National University of Singapore. *Proc. Of ACM Conf. On Image And Video Retrieval (CIVR'09).*
21. Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M. & Leskovec, J. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *ArXiv Preprint ArXiv:2005.00687.* (2020)
22. Navarin, N., Erb, W., Pasa, L. & Sperduti, A. Linear Graph Convolutional Networks. *28th European Symposium On Artificial Neural Networks, Computational Intelligence And Machine Learning, ESANN 2020, Bruges, Belgium, October 2-4, 2020.* pp. 151-156 (2020)
23. Akiba, T., Sano, S., Yanase, T., Ohta, T. & Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. *ArXiv.* **abs/1907.10902** (2019)
24. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. & Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances In Neural Information Processing Systems 32.* pp. 8024-8035 (2019)
25. Tremblay, N., Gonçalves, P. & Borgnat, P. Design of graph filters and filterbanks. (2017)
26. Shuman, D., Narang, S., Frossard, P., Ortega, A. & 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**, 83-98 (2013)

27. Lim, D., Li, X., Hohne, F. & Lim, S. New Benchmarks for Learning on Non-Homophilous Graphs. *The WebConf Workshop On Graph Learning Benchmarks (GLB-WWW).* (2021)
28. Lukovnikov, D. & Fischer, A. Improving Breadth-Wise Backpropagation in Graph Neural Networks Helps Learning Long-Range Dependencies.. *Proceedings Of The 38th International Conference On Machine Learning.*
29. Chamberlain, B., Rowbottom, J., Gorinova, M., Bronstein, M., Webb, S. & Rossi, E. GRAND: Graph Neural Diffusion. *Proceedings Of The 38th International Conference On Machine Learning.* **139** pp. 1407-1418 (2021,7,18)
30. Yang, Y., Liu, T., Wang, Y., Zhou, J., Gan, Q., Wei, Z., Zhang, Z., Huang, Z. & Wipf, D. Graph Neural Networks Inspired by Classical Iterative Algorithms. *Proceedings Of The 38th International Conference On Machine Learning.*
31. Wang, Y. & Derr, T. Tree Decomposed Graph Neural Network. *Conference On Information And Knowledge Management.* (2021)
32. Zheng, X., Zhou, B., Gao, J., Wang, Y., Lió, P., Li, M. & Montufar, G. How Framelets Enhance Graph Neural Networks. *Proceedings Of The 38th International Conference On Machine Learning.*
33. Navarin, N., Erb, W., Pasa, L. & Sperduti, A. Linear Graph Convolutional Networks. *ESANN.* pp. 151-156 (2020)
34. Saad, Y. On the Rates of Convergence of the Lanczos and the Block-Lanczos Methods. *SIAM Journal On Numerical Analysis.* **17**, 687-706 (1980)
35. LI, R. SHARPNESS IN RATES OF CONVERGENCE FOR THE SYMMETRIC LANCZOS METHOD. *Mathematics Of Computation.* **79**, 419-435 (2010)
36. Cullum, J. & Willoughby, R. Lanczos Algorithms for Large Symmetric Eigenvalue Computations. (Society for Industrial,2002)
37. Dong, Y., Ding, K., Jalaian, B., Ji, S. & Li, J. Graph Neural Networks with Adaptive Frequency Response Filter. (2021)
38. He, M., Wei, Z., Huang, Z. & Xu, H. BernNet: Learning Arbitrary Graph Spectral Filters via Bernstein Approximation. (2021)
39. Bianchi, F., Grattarola, D., Livi, L. & Alippi, C. Graph Neural Networks with Convolutional ARMA Filters. *IEEE Transactions On Pattern Analysis And Machine Intelligence.* pp. 1-1 (2021)
40. Gama, F., Marques, A., Leus, G. & Ribeiro, A. Convolutional Neural Network Architectures for Signals Supported on Graphs. *IEEE Transactions On Signal Processing.* **67**, 1034-1049 (2019,2)
41. Cai, C. & Wang, Y. A Note on Over-Smoothing for Graph Neural Networks. (2020)
42. Zhou, K., Huang, X., Zha, D., Chen, R., Li, L., Choi, S. & Hu, X. Dirichlet Energy Constrained Learning for Deep Graph Neural Networks. (2021)
43. Davidson, E. & Thompson, W. Monster Matrices: Their Eigenvalues and Eigenvectors. *Computers In Physics.* **7**, 519-522 (1993)
44. Wang, H., Wei, Z., Gan, J., Wang, S. & Huang, Z. Personalized PageRank to a Target Node, Revisited. *CoRR.* **abs/2006.11876** (2020)
45. Stewart, G. A Krylov–Schur Algorithm for Large Eigenproblems. *SIAM Journal On Matrix Analysis And Applications.* **23**, 601-614 (2002)
46. Lehoucq, R., Sorensen, D. & Yang, C. ARPACK Users Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods.. (1997)
47. Shchur, O., Mumme, M., Bojchevski, A. & Günnemann, S. Pitfalls of graph neural network evaluation. *ArXiv Preprint ArXiv:1811.05868.* (2018)