SoGCN: Second-Order Graph Convolutional Networks

Anonymous authors
Paper under double-blind review

Abstract

We introduce a second-order graph convolution (SoGC), a maximally localized kernel, that can express a polynomial spectral filter of order K with arbitrary coefficients. We contrast our SoGC with both first-order (one-hop) aggregation and vanilla GCN by analyzing graph convolutional layers via generalized filter space. We argue that the SoGC is a simple design capable of forming the basic building block of graph convolution, playing the same role as 3×3 kernels in CNNs. We build purely topological Second-Order GCN (SoGCN) and demonstrate that SoGCN consistently outperforms the state-of-the-art methods on the latest benchmark.

1 Introduction

Deep localized convolutional filters have achieved great success in the field of deep learning. In image recognition, the effectiveness of 3×3 kernels as the basic building block in Convolutional Neural Networks (CNNs) is shown both experimentally and theoretically (Zhou, 2020). We are inspired to search for the maximally localized Graph Convolution (GC) kernel with full expressiveness power for Graph Convolutional Networks (GCNs).

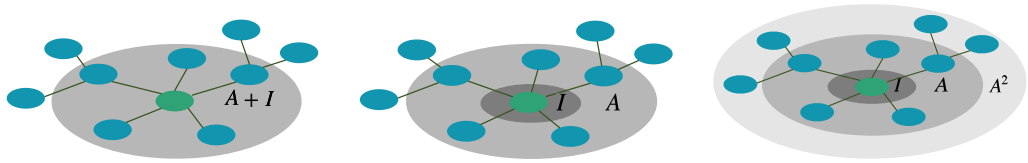
Most existing GCN methods utilize localized GCs based on one-hop aggregation scheme as the basic building block. Extensive works have shown performance limitations of such design due to over-smoothing (Li et al., 2018; Oono & Suzuki, 2019; Cai & Wang, 2020). In vanilla GCNs (Kipf & Welling, 2017) the root cause of its deficiency is the lumping of the graph node self-connection with pairwise neighboring connections. Recent works of Xu et al. (2019); Dehmamy et al. (2019); Ming Chen et al. (2020) disentangle the effect of self-connection by adding an identity mapping (so-called first-order GC). However, its lack of expressive power in filter representation remains (Abu-El-Haija et al., 2019). The work of (Ming Chen et al., 2020) conjectured that the ability to express a polynomial filter with arbitrary coefficients is essential for preventing over-smoothing.

A longer propagation distance in the graph facilitates GCNs to retain its expressive power, as pointed out by (Liao et al., 2019; Luan et al., 2019; Abu-El-Haija et al., 2019). The minimum propagation distance needed to construct our building block of GCN remains the open question. We show that the minimum propagation distance is two: a two-hop graph kernel with the second-order polynomials in adjacency matrices is sufficient. We call our graph kernel Second-Order GC (SoGC).

We introduce a Layer Spanning Space (LSS) framework to quantify the expressive power of multi-layer GCs for modeling a polynomial filter with arbitrary coefficients. By relating low-pass filtering on the graph spectrum (Hoang & Maehara, 2019) with over-smoothing, one can see the lack of filter representation power (Ming Chen et al., 2020) can lead to the performance limitation of GCN.

Using the LSS framework, we show that SoGCs can approximate any linear GCNs in channel-wise filtering Furthermore, higher-order GCs do not contribute more expressiveness, and vanilla GCN or first-order GCs cannot represent all polynomial filters in general. In this sense, SoGC is the maximally localized graph kernel with the full representation power.

To validate our theory, we build Second-Order Graph Convolutional Networks (SoGCN) using SoGC kernels. Our SoGCN using simple graph topological features consistently achieves state-of-the-art performance on the real-world GNN benchmark datasets (Dwivedi et al., 2020), including citation networks, super-pixel classification, and molecule regression.



(a) Vanilla GC (b) First-Order GC (c) Our SoGC

Figure 1: Vertex domain interpretations of vanilla GC, First-Order GC and SoGC. Denote I as the 0-hop aggregator, A the 1-hop aggregator and A^2 the 2-hop aggregator. Nodes in the same colored ring share the same weights. (a) I and A of vanilla GC share the same weights. (b) First-order GC disentangles I from A. (c) SoGC introduces new weights for A^2 in addition.

To our best knowledge, this work is the first study that identifies the importance of the two-hop neighborhood in the context of GCNs' ability to express a polynomial filter with arbitrary coefficients. Our model is a special but non-trivial case of Defferrard et al. (2016). Kipf & Welling (2017) conducted an ablation study with GC kernels of different orders but missed the effectiveness of the second-order relationships. The work of Abu-El-Haija et al. (2019) talked about muti-hop graph kernels; however, they did not identify the critical importance of the second-order form. In contrast, we clarify the prominence of SoGCs in theories and experiments.

Our research on graph convolution using pure topologically relationship is orthogonal to those uses geometric relations (Monti et al., 2017; Fey et al., 2018; Pei et al., 2020), or those with expressive edge features (Li et al., 2016; Gilmer et al., 2017; Corso et al., 2020), and hyper-edges (Morris et al., 2019; Maron et al., 2018; 2019). It is also independent with graph sampling procedures (Rong et al., 2019; Hamilton et al., 2017; Li et al., 2019).

2 PRELIMINARIES

We begin by reformulating spectral GCNs and introducing our notation. We are interested in a finite graph set $\mathcal{G} = \{G_1, \cdots, G_{|\mathcal{G}|}\}$. Assume each graph $G \in \mathcal{G}$ is simple and undirected, associated with a finite vertex set $\mathcal{V}(G)$, an edge set $\mathcal{E}(G) = \{(u, v) : \forall u \leftrightarrow v\}$, and a symmetric normalized adjacency matrix $\boldsymbol{A}(G)$ (Chung & Graham, 1997; Shi & Malik, 2000). Without loss of generality and for simplicity, $|\mathcal{V}(G)| = N$ for every $G \in \mathcal{G}$. Single-channel features $\boldsymbol{x} \in \mathbb{R}^N$ supported in graph $G \in \mathcal{G}$ is a vectorization of function $\mathcal{V}(G) \to \mathbb{R}$.

Graph Convolutions (GCs) is known as Linear Shift-Invariant (LSI) operators to adjacency matrices (Sandryhaila & Moura, 2013). By this definition, GCs can extract features regardless of where local structures fall. Given parameter space $\Omega \subseteq \mathbb{R}$, we write a single-channel GC (Sandryhaila & Moura, 2013; Defferrard et al., 2016) as a mapping $f_{\boldsymbol{\theta}} : \mathcal{G} \times \mathbb{R}^N \to \mathbb{R}^N$ such that [1]:

$$f_{\boldsymbol{\theta}}(G, \boldsymbol{x}) = \sum_{k=0}^{K} \theta_k \boldsymbol{A}(G)^k \boldsymbol{x}, \tag{1}$$

where $\boldsymbol{\theta} = [\theta_0 \quad \cdots \quad \theta_K]^T \in \Omega^{K+1}$ parameterizes the GC. K reflects the localization of $f_{\boldsymbol{\theta}}$: a linear combination of features aggregated by $\boldsymbol{A}(G)^k$. Moreover, we reformulate two popular models, vanilla GC (Figure 1a) and first-order GC (Figure 1b), as below:

$$f_0(G, \boldsymbol{x}) = \theta \left(\boldsymbol{A}(G) + \boldsymbol{I}\right) \boldsymbol{x}, \quad f_1(G, \boldsymbol{x}) = \left(\theta_1 \boldsymbol{A}(G) + \theta_0 \boldsymbol{I}\right) \boldsymbol{x}. \tag{2}$$

The general spectral GCNs stack L layers of GCs (Equation 1) with nonlinear activations. Let $f^{(l)}$ be GC layers with parameters $\boldsymbol{\theta}^{(l)} \in \Omega^{K+1}, l \in [L]$, the single-channel GCNs can be written as:

$$F(G, \boldsymbol{x}) = g \circ f^{(L)} \circ \sigma \circ f^{(L-1)} \circ \cdots \circ \sigma \circ f^{(1)}(G, \boldsymbol{x}), \tag{3}$$

where σ is an element-wise activation function, the superscripts denote the corresponding layer number, g is a task-specified readout function (e.g., softmax), the inputs are graph $G \in \mathcal{G}$ and signals $\boldsymbol{x} \in \mathbb{R}^N$. The compositionality principle of deep learning suggests L being large, while K being small and localized (LeCun et al., 2015).

[1] We can replace the Laplacian matrix $\boldsymbol{L}$ in Defferrard et al. (2016) with the normalized adjacency matrix $\boldsymbol{A}$ since $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{A}$.

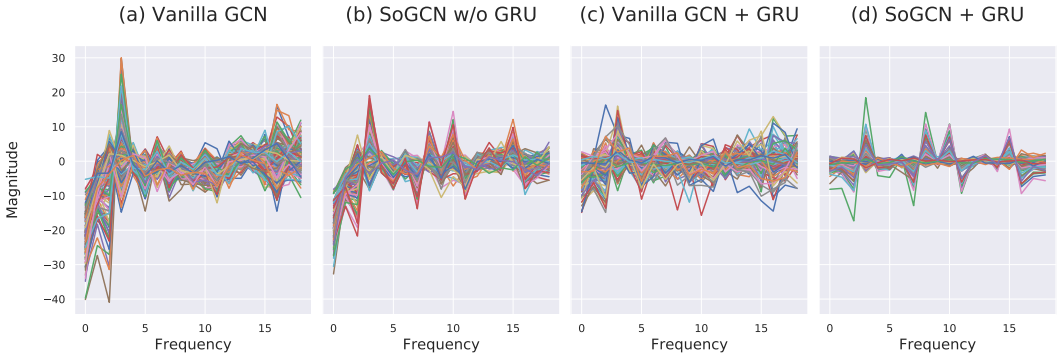


Figure 2: Visualizing output activation in graph spectrum domain for vanilla GCN, SoGCN, and GRU variants. The test is conducted on a graph from the ZINC dataset. The spectrum is defined as a projection of activation functions on the graph eigenvectors. SoGCN preserved higher-order spectrum, while vanilla GCN shows over-smoothing. See Appendix F for more visualizations on the ZINC dataset.)

3 Overview: Second-Order Graph Convolution

We are interested in the overall graph convolution network's representation power of expressing a polynomial filter (Equation 1) with arbitrary coefficients. A multi-layer GCN approximate a K-order polynomial filter $\sum_{k=0}^{K} \theta_k \boldsymbol{A}(G)^k, \theta_k \in \Omega, k = 0, \cdots K$, by stacking basic building blocks of graph convolution (GC) layers (Wu et al., 2019; Ming Chen et al., 2020).

We formally define the second-order GC (SoGC) using the second-order polynomials of adjacency matrices:

$$f_2(G, \boldsymbol{x}) = \left(\theta_2 \boldsymbol{A}(G)^2 + \theta_1 \boldsymbol{A}(G) + \theta_0 \boldsymbol{I}\right) \boldsymbol{x}, \tag{4}$$

where $\theta_i \in \mathbb{R}, i = 0, 1, 2$ are trainable paremeters in the context of machine learning. Its vertex-domain interpretation is illustrated in Figure 1c. At first glance, it seems that we could stack two one-hop graph convolution (GC) kernels to approximate a SoGC. However, as shown in Section 4.2, that is not the case.

The critical insight is that graph filter approximation can be viewed as a polynomial factorization problem. It is known that any univariate polynomial can be factorized into sub-polynomials of degree two. Based on this fact, we show by stacking enough SoGCs (and varying their parameters) can achieve decomposition of any polynomial filters.

In contrast, first-order GCs are not universal approximators; two stacked one-hop GCs cannot model every two-hop GC (by the polynomial approximation argument). Polynomial filter completeness of SoGC leads to better performance of GCNs. As shown in Figure 2, networks built with SoGC can overcome over-smoothing and extract features on high-frequency bands as measured in the graph eigenvector space.

4 Representation Power Analysis

4.1 Layer Spanning Space Framework

To illustrate the representation power of GC layers, we establish a Layer Spanning Space (LSS) framework to study the graph filter space spanned by stacking multiple graph kernels.

First, we present our mathematical devices in Definition 1, 2 with Lemma 1 as below.

Definition 1. *Suppose the parameter space $\Omega = \mathbb{R}$. The Linear Shift-Invariant (LSI) graph filter space of degree $K > 0$ with respect to a finite graph set $\mathcal{G}$ is defined as $\mathcal{A} = \left\{ f_{\boldsymbol{\theta}} : \mathcal{G} \times \mathbb{R}^N \to \mathbb{R}^N, \forall \boldsymbol{\theta} \in \mathbb{R}^{K+1} \right\}$, where $f_{\boldsymbol{\theta}}$ follows the definition in Equation 1.*

Definition 2. *Let spectrum set $\mathcal{S}(\mathcal{G}) = \{\lambda : \lambda \in \mathcal{S}(\boldsymbol{A}(G)), \forall G \in \mathcal{G}\}$, where $\mathcal{S}(\boldsymbol{A})$ denotes the eigenvalues of $\boldsymbol{A}$. Define spectrum capacity $\Gamma = |\mathcal{S}(\mathcal{G})|$. In particular, $\Gamma = (N-1)|\mathcal{G}|$ if every graph adjacency matrix has no common eigenvalues other than* 1.

Lemma 1. *$\mathcal{A}$ of degree $K > 0$ has dimension $\min\{K+1, \Gamma\}$ as a vector space.*

Proof of Lemma 1 follows from Theorem 3 of Sandryhaila & Moura (2013). The complete version can be found in Appendix C. Here, we induce a finite-dimension filter space by Lemma 1. This allows us to make the deterministic analysis on $\mathcal{A}$.

For simplicity, we will model the linear composition of filters to analyze its representation power. The nonlinear activation effects are beyond the scope of this work. Following Definition 1, let $\mathcal{A}$ be the full filter space of degree $\Gamma - 1$ and $\mathcal{B}$ be the low-level filter space as a set of polynomials in adjacency matrices (Equation 1). Denote the GC at l-th layer by $f^{(l)} \in \mathcal{B}$, then we yield the LSS of stacking L layers as below:

$$\mathcal{B}^L = \left\{ f : f(G, \boldsymbol{x}) = f^{(L)} \circ \cdots \circ f^{(1)}(G, \boldsymbol{x}) = \prod_{l=1}^{L} p^{(l)}(\boldsymbol{A}(G))\boldsymbol{x} \right\}, \tag{5}$$

where $p^{(l)}(x)$ varies in a certain class of polynomials. We can assess the expressive capability of GC layers by comparing the LSS with $\mathcal{A}$. Localized kernels in $\mathcal{B}$ have full representation power if $\mathcal{A} \subseteq \mathcal{B}^L$. We are interested in $\mathcal{B}_K$, which denotes all localized filters of degree at most K. The LSS of $\mathcal{B}_K$ is modeled as:

$$\mathcal{B}_K^L = \left\{ f : f(G, \boldsymbol{x}) = \prod_{l=1}^{L} \sum_{k=0}^{K} \theta_k^{(l)} \boldsymbol{A}(G)^k \boldsymbol{x}, \theta_k^{(l)} \in \mathbb{R} \right\}, \tag{6}$$

where the number of layers is bounded by $L \leq \lceil (\Gamma - 1)/K \rceil$, according to Lemma 1.

4.2 Universal Representation Power of SoGC

In this section, we will show the universal representation power of SoGCs using our LSS framework. First, we add superscripts to Equation 4 to indicate the layer number. We present Theorem 1 as follows.

Theorem 1. *For any $f \in \mathcal{A}$, there exists $f_2^{(l)} \in \mathcal{B}_2$ with coefficients $\theta_0^{(l)}, \theta_1^{(l)}, \theta_2^{(l)} \in \mathbb{R}, l = 1, \cdots, L$ such that $f = f_2^{(L)} \circ \cdots \circ f_2^{(1)}$ where $L \leq \lceil (\Gamma - 1)/2 \rceil$.*

We conclude the proof by leveraging a fundamental polynomial factorization theorem (See Appendix D). Theorem 1 can be regarded as the universal approximation theorem of linear GCNs, which implies that SoGCs should be the basic building blocks without loss of expressiveness.

Theorem 1 also demonstrates how GCNs with SoGCs benefit from depth, which coincides with the view of Dehmamy et al. (2019). Figure 3a verifies our SoGCN can overcome over-smoothing and successfully utilize depth to attain performance gain.

4.3 Representation Power of Other Graph Convolution

We show that vanilla and first-order GCs lack expressiveness, while higher-order GCs are too complex to learn.

Vanilla vs. second-order. Extensive works have shown the performance deficiency of vanilla GCNs (Hoang & Maehara, 2019; Luan et al., 2019; Oono & Suzuki, 2019; Cai & Wang, 2020). Based on the LSS framework, we can point out a similar issue but from a novel perspective. Let us write $f_0^{(l)}(G, \boldsymbol{x}) = \theta^{(l)}(\boldsymbol{A}(G) + \boldsymbol{I})\boldsymbol{x} \in \mathcal{B}_0$ as the l-th GC layer[2]. Then L of them can represent a LSS as follows:

$$\mathcal{B}_0^L = \left\{ f : f(G, \boldsymbol{x}) = \theta \sum_{l=0}^{L} \binom{l}{L} \boldsymbol{A}(G)^l \boldsymbol{x} \right\}, \tag{7}$$

[2]Notice that we use $\mathcal{B}_0$ to denote the filter space with lumping of self-connection with pairwise neighbor nodes, since the zero-degree ones are too trivial. We assume the renormalization trick can be merged into θ.

by letting $\theta = \theta^{(L)} \cdots \theta^{(1)}$. No matter how large L is or how optimizers tune the parameters $\theta^{(l)}$, $\dim \mathcal{B}_0^L = 1$ which signifies $\mathcal{B}_0^L$ degenerates to a negligible subspace of $\mathcal{A}$.

First-order vs. second-order. SoGCs, as the quadratic response function on the spectral domain, characterize more diverse filters than first-order GCs. For example, first-order GCs cannot simulate band-pass filters. We denote first-order GCs as $f_1^{(l)}(G, \boldsymbol{x}) = (\theta_1^{(l)} \boldsymbol{A}(G) + \theta_0^{(l)} \boldsymbol{I})\boldsymbol{x} \in \mathcal{B}_1$. In the spirit of Section 4.1, write the LSS as:

$$\mathcal{B}_1^L = \left\{ f : f(G, \boldsymbol{x}) = \prod_{l=1}^{L} \left(\theta_1^{(l)} \boldsymbol{A}(G) + \theta_0^{(l)} \boldsymbol{I} \right) \boldsymbol{x}, \theta_0^{(l)}, \theta_1^{(l)} \in \mathbb{R} \right\}, \tag{8}$$

which is isomorphic to a polynomial space whose elements split over the real domain (following the proof sketch in Appendix D). Compared with $\mathcal{B}_0^L$ (Equation 7), $\mathcal{B}_1^L$ represents a much larger subset of $\mathcal{A}$. This highlights the importance of the first-order term or the identity mapping mentioned in (Xu et al., 2019; Dehmamy et al., 2019; Ming Chen et al., 2020).

The limitations also become obvious since not all polynomials can be factorized into first-order polynomials. These polynomials only occupy a small proportion in the ambient polynomial space (Li, 2011), which indicates first-order GCs are not universal approximators in general.

Higher-order vs. second-order. GCs of degree $K \geq 2$ should have equivalent expressive power as SoGCs are their reduced cases. But their by-product is an uncertain sparsity of coefficients, which is not compatible with gradient-based optimization algorithms. Extensive experiments (Defferrard et al., 2016) have shown the ineffectiveness of learning higher-order kernels, because eigenvalues of graph adjacency matrices diminish when powered. This results in a decreasing numerical rank of $\boldsymbol{A}(G)^k$, which prevent higher-order GCs from aggregating larger-scale information. SoGCs can alleviate this problem by preventing the loss of information due to higher-order powering operation. Finally, higher-order GC lacks nonlinearity. SoGCN can bring a better balance between the expressive power of low-level layers and nonlinearity between them.

5 Second-Order Graph Convolutional Networks

Promoting SoGCs to the multi-channel version analogous to Kipf & Welling (2017), we establish our Second-Order Graph Convolutional Networks (SoGCN) following the fashion of deep learning. We cascade a feature embedding layer, multiple SoGC layers, and append a readout module. Suppose the multi-channel input is $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ supported in graph $G \in \mathcal{G}$, denote the output of l-th layer as $\boldsymbol{X}^{(l)} \in \mathbb{R}^{N \times E}$, the final node-level output as $\boldsymbol{Y} \in \mathbb{R}^{N \times F}$, or graph-level output as $\boldsymbol{Y} \in \mathbb{R}^{E}$, we formulate our novel deep GCN based on SoGC as follows:

$$\boldsymbol{X}^{(0)} = \rho\left(\boldsymbol{X}; \boldsymbol{\Phi}\right), \tag{9}$$

$$\boldsymbol{X}^{(l+1)} = \sigma\left(\boldsymbol{A}(G)^2 \boldsymbol{X}^{(l)} \boldsymbol{\Theta}_2^{(l+1)} + \boldsymbol{A}(G) \boldsymbol{X}^{(l)} \boldsymbol{\Theta}_1^{(l+1)} + \boldsymbol{X}^{(l)} \boldsymbol{\Theta}_0^{(l+1)} \right), \tag{10}$$

$$\boldsymbol{Y} = \tau\left(\boldsymbol{X}^{(L)}; \boldsymbol{\Psi} \right), \tag{11}$$

where $\boldsymbol{\Theta}_i^{(l)} \in \mathbb{R}^{E \times E}, i = 0, 1, 2$ are trainable weights for linear filters. $\rho : \mathbb{R}^{N \times D} \to \mathbb{R}^{N \times E}$ is an equivariant (Maron et al., 2018) embedder with parameters $\boldsymbol{\Phi}$. $\sigma : \mathbb{R}^{N \times E} \to \mathbb{R}^{N \times E}$ is a generalized nonlinear activation. For node-level readout, $\tau : \mathbb{R}^{N \times E} \to \mathbb{R}^{N \times F}$ can be a decoder (with parameters $\boldsymbol{\Psi}$) or a nonlinear activation (e.g. softmax) in place of the prior layer. For graph-level output, $\tau : \mathbb{R}^{N \times E} \to \mathbb{R}^{E}$ should be an invariant readout function, e.g., channel-wise sum, mean or max (Hamilton et al., 2017). In practice, we adopt a multi-layer perceptron (MLP) as the embedding function ρ, another MLP for node regression readout, and sum (Xu et al., 2019) for graph classification readout. We defer the discussion of nonlinear activation to the next subsection.

5.1 Nonlinear Activation

The nonlinear activations include but not limited to element-wise functions (e.g., ReLU). ReLU breaks shift-invariance and performs low-pass filtering (Oono & Suzuki, 2019; Cai & Wang, 2020),

but benefits deep networks in approximation capability. Hence, we combine ReLUs with a gated recurrent unit (GRU) (Li et al., 2016) shared among each layer (i.e., $\sigma = \mathrm{GRU} \circ \mathrm{ReLU}$).

GRU retains information from previous layers effectively. From our perspectives, each GC layer is performing filtering on the spectrum. Due to the limited capability of low-level GC units, enhancing features means suppressing their counterparts. GRU plays the role of a selective skip connection, which allows GC layers to filter out useless features without balancing the information loss.

We study the effects of GRU with visualization and experiments. Figure 2 illustrates the behavior of GRU on the spectral domain. It acts as a memorable filter that selects useful frequency bands. However, we argue that GRU can only facilitate GCNs to acquire more explicit patterns on the spectrum, whether significant features can be extracted depends on the strength of GC layers. We can whiteness the ineffectiveness of GRU combined with vanilla GCs in Figure 2.

5.2 Related Work

Spectral GCNs. (Bruna et al., 2014) first propose spectral-domain graph convolutional networks. In the follow-up works, ChebyNets (Defferrard et al., 2016) approximate localized graph filters with Chebyshev polynomials. (Kipf & Welling, 2017) simplifies these models by taking the first-order approximation and propose a renormalization trick. Wu et al. (2019) conducts an empirical study in the nonlinear activation, and simulates a GCN as polynomial filters with fixed coefficients. Spectral-domain GCs are all related to the polynomial in the adjacency matrix. Our SoGC identifies the prominence of the second-order polynomials.

Multi-Scale GCNs. Multi-scale GCNs (Liao et al., 2019; Luan et al., 2019; Abu-El-Haija et al., 2019) aggregate information from multiple neighborhoods instead of concentrating on local structures. Liao et al. (2019); Abu-El-Haija et al. (2019) apply higher-order polynomials in adjacency matrices to explore farther neighborhoods. Luan et al. (2019); Li et al. (2019) add dense connections to retain the local features. Luan et al. (2019) further proves the equivalence between these two models. In our settings, these multi-scale GCs can be modeled as higher-order polynomial filters. Our SoGC possesses equivalent expressiveness, which indicates the unnecessity of their longer-distance propagation.

Over-smoothing and Representation Power. Over-smoothing has become a major problem that hinders GCNs from going deeper, which makes node features indistinguishable. Li et al. (2018); Oono & Suzuki (2019); Cai & Wang (2020) provide multi-perspective analysis on this problem. (Hoang & Maehara, 2019) concludes these results as the constant low-pass filtering on the spectrum. Recent works tackle over-smoothing by edge sampling (Rong et al., 2019), jumping connections (Xu et al., 2018), and enhancing polynomial expressiveness (Chen et al., 2020). Although (Chen et al., 2020) notices the importance of filter completeness, their arguments stop at talking about over-smoothing. Filter expressive power should be a more general problem of GCNs.

6 Experiments

Experiments are conducted on the synthetic dataset in Section 6.1 and on the real-world benchmarks (Dwivedi et al. (2020)) in Section 6.2.

6.1 Synthetic Graph Spectrum Dataset for Node Regression

To validate the expressiveness of SoGCN, and its power to preserve higher-order graph signal, we build a Synthetic Graph Spectrum (SGS) dataset for the node signal filtering regression task. We construct SGS dataset with random graphs. The learning task is to mimic three types of hand-crafted filtering functions: high-pass, low-pass, and band-pass on the graph spectral space (defined over the graph eigenvectors). There are 1k training graphs, 1k validation graphs, and 2k testing graphs for each filtering function. Each graph is undirected and comprises 80 ˜120 nodes. Appendix E covers more details of our SGS dataset. We choose Mean Absolute Error (MAE) as the evaluation metric.

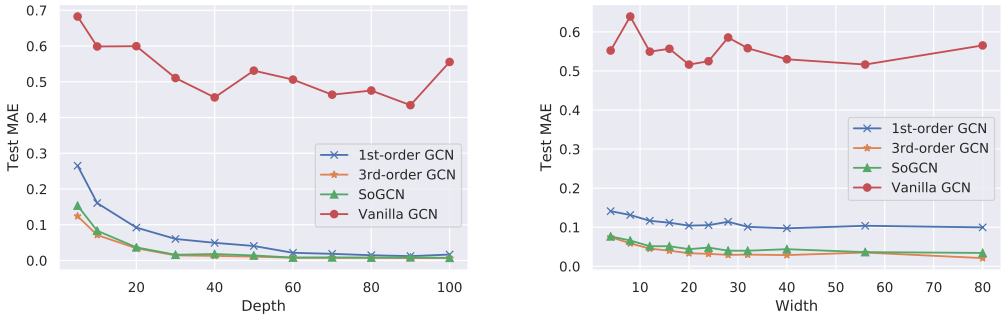
Experimental Setup. We compare SoGCN with vanilla GCN (Kipf & Welling (2017)), first-order GCN, and higher-order GCNs on the synthetic dataset. To evaluate each model's expressiveness

purely on the graph kernel design, we remove ReLU activations for all tested models. We adopt the Adam optimizer (Kingma & Ba (2015)) in our training process, with a batch size of 128. The learning rate begins with 0.01 and decays by half once the validation loss stagnates for more than 10 training epochs.

Table 1: The graph node signal regression performances with *High-pass*, *Low-pass*, and *Band-pass* filters (over graph spectral space) as learning target. Each model has 16 GC layers and 16 channels of hidden layers.

Model	**#Param**	Test MAE $\pm$ s.d.		
		High-pass	**Low-pass**	**Band-pass**
Vanilla GCN	4611	0.308$\pm$0.006	0.317$\pm$0.011	0.559$\pm$0.071
Vanilla GCN + ReLUs [3]	4611	0.466$\pm$0.002	0.457$\pm$0.002	0.299$\pm$0.000
1st-order GCN	8467	0.036$\pm$0.004	0.032$\pm$0.002	0.115$\pm$0.008
3rd-order GCN	16179	0.021$\pm$0.003	0.022$\pm$0.001	0.045$\pm$0.008
4th-order GCN	20035	0.021$\pm$0.003	0.022$\pm$0.002	0.049$\pm$0.006
SoGCN	12323	0.021$\pm$0.003	0.023$\pm$0.002	0.050$\pm$0.004

Results and Discussion. Table 1 summarizes the quantitative comparisons. SoGCN achieves the superior performance on all of the 3 tasks outperforming vanilla GCN and 1st-order GCN, which implies that SoGC graph convolutional kernel does benefit from explicit disentangling of the $\theta_0^{(l)}\boldsymbol{I}$ (0-hop) and $\theta_2^{(l)}\boldsymbol{A}^2$ (2-hop) terms. Our results also show that higher-order (3rd-order and 4th-order) GCNs do not improve the performance further, even though they have many more parameters. SoGCN possesses a more expressive representation ability and a good trade-off between performance and model size.



(a) Relations between test MAE and depth.

(b) Relations between test MAE and width.

Figure 3: Relations between test MAE and depth or width. Experiments of both (a) and (b) are conducted on synthetic *Band-pass* dataset. Each model in (a) has 16 channels per hidden layer with varying depth. Each model in (b) consists of 16 GC layers with varying width.

Figure 3 plots MAE results as we vary the depth of GC layers and channel width. Vanilla GCN can benefit from neither depth nor width. First-order GC, SoGC, and higher-order GC can leverage depth to span larger LSS. Figure 3a illustrates the corresponding performance for each graph kernel types. SoGC and higher-order GC both outperform first-order GC as depth increases. Figure 3b shows the benefits of SoGC remain as we move to multi-channel construction. Comparing Figure 3a and 3b, we find that depth has larger effect on GCNs. As the depth increases, the performance gap between first-order GC and SoGC reduces for a fixed feature channel width of 16 in this experiment.

[3]This experimental group shows that ReLUs are not always such beneficial on our synthetic dataset.

6.2 REAL-WORLD BENCHMARKS

We follow the benchmarks outlined in Dwivedi et al. (2020) for evaluating GNNs on many datasets across a variety of artificial and real-world tasks. We choose to evaluate our SoGCN on a chemistry dataset (ZINC molecules) for the graph regression task and two computer vision datasets (CIFAR10 and MNIST superpixels) for the graph classification task.

Experimental Setup. We created a variant of our model, called SoGCN-GRU, which replaces the ReLU activations with $\mathrm{GRU} \circ \mathrm{ReLU}$. We compare our proposed SoGCN and SoGCN-GRU with state-of-the-art GNNs: vanilla GCN (Kipf & Welling (2017)), GAT (Veličković et al. (2018)), GIN (Xu et al. (2019)), GraphSage (Hamilton et al. (2017)), GatedGCN(Bresson & Laurent (2017)) and 3WL-GNN (Maron et al. (2019)). To ensure fair comparisons, we follow the same training and evaluation pipelines (including optimizer settings) and data splits of benchmarks. Furthermore, we adjusted our model depth to ensure it satisfies parameter budgets as specified in the benchmark. Note that we do not use any geometrical information to encode rich graph edge relationship, as in models such as GatedGCN-E-PE. We use only binary graph connectivity information.

Table 2: Comparision with GNN models on CIFAR10, MNIST and ZINC datasets. **Red**: the best model, **Green**: good models.

Model	**# Param**	Test ACC ± s.d.	Test ACC ± s.d.	**# Param**	Test MAE ± s.d.
		CIFAR10	**MNIST**		**ZINC**
Vanilla GCN	100k	55.710±0.381	90.705±0.218	500k	0.367±0.011
GAT	100k	64.223±0.455	95.535±0.205	500k	0.384±0.007
MoNet	100k	65.911±2.515	90.805±0.032	500k	0.292±0.006
GraphSage	100k	65.767±0.308	**97.312±0.097**	500k	0.398±0.002
GIN	100k	55.255±1.527	96.485±0.252	500k	0.387±0.015
GatedGCN	100k	**67.312±0.311**	**97.340±0.143**	500k	0.350±0.020
3WLGNN	100k	59.175±1.593	95.075±0.961	100k	0.407±0.028
SoGCN	100k	**66.338±0.155**	**96.785±0.113**	500k	**0.238±0.017**
SoGCN-GRU	100k	**68.208±0.271**	**97.729±0.159**	500k	**0.201±0.006**

Results and Discussion Table 2 reports the benchmark results. Our model SoGCN makes small computational changes to GCN by adopting 2-hop and 0-hop neighborhoods, and it outperforms models with complicated graph edge encoding mechanisms. With GRU nonlinear activation, SoGCN-GRU tops almost all state-of-the-art GNNs on the three datasets we tested. Furthermore, we visualize the network activation on the graph spectrum in Figure 2 and Figure 5 to illustrate SoGCN's and GRU's impacts on preventing over-smoothing.

7 CONCLUSION

What is the most localized graph convolution kernel (GC) with full expressiveness for GCNs? To answer this question, we generalize the filter space to a finite graph set and establish our LSS framework to assess GC layers functioning on different hops. We show our second-order model, called SoGC, possesses full representation power than one-hop GCs. Thus, it is the most localized GC that we adopt as the basic building block to establish our SoGCN. Both synthetic and real-world experiments exhibit the prominence of our theoretic design.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *ICML*, 2019.

Xavier Bresson and Thomas Laurent. Residual gated graph convnets. *arXiv:1711.07553*, 2017.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. 2014.

Chen Cai and Yusu Wang. A note on over-smoothing for graph neural networks. In *ICML*, 2020.

Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *AAAI*, 2020.

Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. American Mathematical Soc., 1997.

Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Lio, and Petar Velickovic. Principal neighbourhood aggregation for graph nets. In *ICML*, 2020.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016.

Nima Dehmamy, Albert-László Barabási, and Rose Yu. Understanding the representation power of graph neural networks in learning graph topology. In *NeurIPS*, 2019.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv:2003.00982*, 2020.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. In *CVPR*, 2018.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

NT Hoang and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv:1905.09550*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 2015.

Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *ICCV*, 2019.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, 2018.

Wembo Li. Probability of all real zeros for random polynomial with the exponential ensemble. *Preprint*, 2011.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *ICLR*, 2016.

Renjie Liao, Zhizhen Zhao, Raquel Urtasun, and Richard S Zemel. Lanczosnet: Multi-scale deep graph convolutional networks. In *ICLR*, 2019.

Sitao Luan, Mingde Zhao, Xiao-Wen Chang, and Doina Precup. Break the ceiling: Stronger multi-scale deep graph convolutional networks. In *NeurIPS*, 2019.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. In *ICLR*, 2018.

Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. Provably powerful graph networks. In *NeurIPS*, 2019.

Zhewei Wei Ming Chen, Bolin Ding Zengfeng Huang, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, 2020.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, 2017.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, 2019.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *ICLR*, 2019.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *ICLR*, 2020.

Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2019.

Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE Trans. Signal Process*, 2013.

Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 2000.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr, Christopher Fifty, Tao Yu, and Kilian Q Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, 2018.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

Ding-Xuan Zhou. Universality of deep convolutional neural networks. *ACHA*, 2020.

A SHIFT-INVARIANCE OF DEFINITION 1

Let us rewrite the $\mathcal{A}$ of degree K in Definition 1:

$$\mathcal{A} = \left\{ f : f(G, \boldsymbol{x}) = \sum_{k=0}^{K} \theta_k \boldsymbol{A}(G)^k \boldsymbol{x}, \forall \theta_k \in \mathbb{R} \right\}, \tag{12}$$

which contains all LSI functions $f : \mathcal{G} \times \mathbb{R}^N \to \mathbb{R}^N$ with adjacency matrix as the graph shift. We show this in the following way. First, all linear transformations $\boldsymbol{H}$ invariant to the shift $\boldsymbol{S}$ should have $\boldsymbol{HS} = \boldsymbol{SH}$. Second, specifying arbitrary graph $G \in \mathcal{G}$, any filter associated with it can be written as below:

$$\boldsymbol{H}(G) = \sum_{k=0}^{K} \theta_k \boldsymbol{A}(G)^k = \boldsymbol{U} \left(\sum_{k=0}^{K} \theta_k \boldsymbol{\Lambda}^k \right) \boldsymbol{U}^T, \tag{13}$$

where $\boldsymbol{A}(G) = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ is the eigendecomposition of $\boldsymbol{A}(G)$. Then we can conclude the shift-invariance property by the following Lemma 2.

Lemma 2. *Diagonalizable matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are simultaneously diagonalized if and only if $\boldsymbol{A}_1\boldsymbol{A}_2 = \boldsymbol{A}_2\boldsymbol{A}_1$.*

B RING ISOMORPHISM: $\mathcal{A} \to \mathcal{T}$

We derive an equivalent form of $\mathcal{A}$, namely construct a tractable space for $\mathcal{A}$. Notice that, this construction is essential to the proof of Lemma 1 and Theorem 1.

Since $\mathcal{G}$ is finite, we can construct a block diagonal matrix $\boldsymbol{T} \in \mathbb{R}^{N|\mathcal{G}| \times N|\mathcal{G}|}$, consisting of all adjacency matrices on the diagonal.

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{A}(G_1) & & \\ & \ddots & \\ & & \boldsymbol{A}(G_{|\mathcal{G}|}) \end{bmatrix} \in \mathbb{R}^{N|\mathcal{G}| \times N|\mathcal{G}|}. \tag{14}$$

Hereby, we stop to explain the big picture of Definition 2 with Equation 14. Obviously, the spectrum capacity Γ represents the number of eigenvalues of $\boldsymbol{T}$ without multiplicity. Note that, eigenvalues of adjacency matrices signify graph similarity. The spectrum capacity Γ identifies a set of graphs by enumerating the structural patterns. Even if the graph set goes extremely large (to guarantee the generalization capability), the distribution of spectrum provide the upper bound of Γ, so our theories will not lose generality.

Now we get back to construct a matrix space $\mathcal{T}$ from $\mathcal{A}$ via a ring homomorphism $\pi : \mathcal{A} \to \mathcal{T}$:

$$\pi : \sum_{k=0}^{K} \theta_k \boldsymbol{A}(G)^k \mapsto \sum_{k=0}^{K} \theta_k \boldsymbol{T}^k. \tag{15}$$

Recall that a ring homomorphism preserves the "summation" and "multiplication". Concretely, we write the matrix space $\mathcal{T}$ as follows:

$$\mathcal{T} = \left\{ \boldsymbol{H} : \boldsymbol{H} = \sum_{k=0}^{K} \theta_k \boldsymbol{T}^k, \forall \theta_k \in \mathbb{R} \right\}. \tag{16}$$

In the following part, we prove that π is an isomorphism.

Proof. First, the definition of $\mathcal{T}$ (Equation 16) basically conclude the surjectivity. Second, for any $f_1 \neq f_2 \in \mathcal{A}$ with parameter $\alpha_k, \beta_k \in \mathbb{R}$ where $k = 0, \cdots, K$, there exists $G_j \in \mathcal{G}, \boldsymbol{x} \in \mathbb{R}^N$ such that $f_1(G_j, \boldsymbol{x}) \neq f_2(G_j, \boldsymbol{x})$. And we have their images $\boldsymbol{H}_1 = \pi(f_1), \boldsymbol{H}_2 = \pi(f_2)$. By padding $\boldsymbol{x}$ with zeros like:

$$\boldsymbol{x'} = \begin{bmatrix} \boldsymbol{0}_{N(j-1)}^T & \boldsymbol{x}^T & \boldsymbol{0}_{N(|\mathcal{G}|-j)}^T \end{bmatrix}^T, \tag{17}$$

where $\mathbf{0}_N$ denote the all-zero vector of length N. We apply $\boldsymbol{H}_1, \boldsymbol{H}_2$ to $\boldsymbol{x'}$:

$$\boldsymbol{H}_1\boldsymbol{x'} = \begin{bmatrix} \mathbf{0}_{N(j-1)}^T & \left(\sum_{k=0}^{K} \alpha_k \boldsymbol{A}(G_j)^k \boldsymbol{x}\right)^T & \mathbf{0}_{N(|\mathcal{G}|-j)}^T \end{bmatrix}^T = \begin{bmatrix} \mathbf{0}_{N(j-1)}^T & f_1(G_j, \boldsymbol{x})^T & \mathbf{0}_{N(|\mathcal{G}|-j)}^T \end{bmatrix}^T, \tag{18}$$

$$\boldsymbol{H}_2\boldsymbol{x'} = \begin{bmatrix} \mathbf{0}_{N(j-1)}^T & \left(\sum_{k=0}^{K} \beta_k \boldsymbol{A}(G_j)^k \boldsymbol{x}\right)^T & \mathbf{0}_{N(|\mathcal{G}|-j)}^T \end{bmatrix}^T = \begin{bmatrix} \mathbf{0}_{N(j-1)}^T & f_2(G_j, \boldsymbol{x})^T & \mathbf{0}_{N(|\mathcal{G}|-j)}^T \end{bmatrix}^T. \tag{19}$$

Hence, $\boldsymbol{H}_1 \neq \boldsymbol{H}_2$ concludes the injectivity. □

C Proof of Lemma 1

We first show $\mathcal{A}$ is a vector space, then we leverage the isomorphism to have equality $\dim \mathcal{A} = \dim \mathcal{T}$. Figuring out the dimension of $\mathcal{T}$ is much more tractable.

Proof. By verifying the linear combination is closed or simply implied from the ring isomorphism π, $\mathcal{A}$ is at least a vector space

Then Lemma 1 follows from the Theorem 3 of Sandryhaila & Moura (2013). We briefly conclude the proof in the following way. Let $m(x)$ denote the minimal polynomial of $\boldsymbol{T}$. We have $\Gamma = \deg m(x)$. Due to the isomorphism, $\dim \mathcal{A} = \dim \mathcal{T}$.

Suppose $K + 1 < \Gamma$. First, $\dim \mathcal{T}$ cannot be larger than $K + 1$, because $\left\{\boldsymbol{I}, \boldsymbol{T}, \cdots, \boldsymbol{T}^K\right\}$ is a spanning set. If $\dim \mathcal{T} < K + 1$, then there exists some polynomial $p(x)$ with $\deg p(x) \leq K$, such that $p(\boldsymbol{A}) = \mathbf{0}$. This contradicts the minimality of $m(x)$. $\dim \mathcal{T}$ can only be $K + 1$.

Suppose $K + 1 \geq \Gamma$. For any $\boldsymbol{H} = h(\boldsymbol{T})$ for some polynomial $h(x)$ with $\deg h(x) \leq K$. By polynomial division, there exists unique polynomials $q(x)$ and $r(x)$ such that

$$h(x) = q(x)m(x) + r(x), \tag{20}$$

where $\deg r(x) < \deg m(x) = \Gamma$. Insert $\boldsymbol{T}$ into Equation 20:

$$h(\boldsymbol{T}) = q(\boldsymbol{T})m(\boldsymbol{T}) + r(\boldsymbol{T}) = q(\boldsymbol{T})\mathbf{0} + r(\boldsymbol{T}) = r(\boldsymbol{T}). \tag{21}$$

Therefore, $\left\{\boldsymbol{I}, \boldsymbol{T}, \cdots, \boldsymbol{T}^{\Gamma-1}\right\}$ form a basis of $\mathcal{T}$, i.e., $\dim \mathcal{T} = \Gamma$. □

Remark that, we assume each graph contains the same number of vertices only for the sake of simplicity. Lemma 1 still holds when the vertex numbers are varying, since the construction of Equation 14 is independent of this assumption. However, we need the graph set to be finite, otherwise Γ might be uncountable. We leave the discussion on infinite graph sets for future study.

D Proof of Theorem 1

First, we borrow the concept of $\mathcal{T}$ (Equation 16) in place of $\mathcal{A}$. Then we leverage the following basic yet powerful Lemma 3 to conclude the proof of Theorem 1 straightforwardly.

Lemma 3. *Over the field of reals, the degree of an irreducible non-trivial univariate polynomial is either one or two.*

Proof. For any $f \in \mathcal{A}$, let us map it to $\boldsymbol{H} = h(\boldsymbol{A})$ through the isomorphism π (Equation 15) for some polynomial $h(x)$ with $\deg h(x) \leq \Gamma - 1$ (By Lemma 1).

By Lemma 3, factorize $h(x)$ into series of polynomials with the degree at most two, and then merge first-order polynomials into second-order ones until no paired first-order polynomials remaining. Finally, we obtain the following equation:

$$h(x) = \prod_{l=1}^{\lceil D/2 \rceil} h^{(l)}(x), \tag{22}$$

where $D = \deg h(x)$. If D is even, $\deg h^{(l)} = 2$ for $l = 1, \cdots, \lceil D/2 \rceil$. Otherwise, there exists some $j \in [L]$ such that $\deg h^{(j)}(x) = 1$. Notice that, the remaining first-order polynomials is at most 1, which indicates the sparsity of coefficients is very low.

Now we obtain filters $\boldsymbol{H}^{(l)} = h^{(l)}(\boldsymbol{T})$ for $l = 1, \cdots, \lceil D/2 \rceil$. The last step is applying the inverse of isomorphism π^{-1} to map $\boldsymbol{H}^{(l)} \in \mathcal{T}$ back to $f^{(l)} \in \mathcal{A}$ as below:

$$\pi^{-1} : \sum_{k=0}^{K} \theta_k \boldsymbol{T}^k \mapsto \sum_{k=0}^{K} \theta_k \boldsymbol{A}(G)^k. \tag{23}$$

Recalling Section 4.1, $f^{(l)} \in \mathcal{B}_2$ for $l = 1, \cdots, \lceil D/2 \rceil$. Since π^{-1} is also a ring isomorphism, $\boldsymbol{H} = \boldsymbol{H}^{(\lceil D/2 \rceil)} \circ \cdots \circ \boldsymbol{H}^{(1)}$ implies $f = f^{(\lceil D/2 \rceil)} \circ \cdots \circ f^{(1)}$. □

Whenever discussing the LSS in the spirit of Theorem 1, we refer to applying the ring isomorphism π and mapping the whole filter space to the matrix space $\mathcal{T}$. Using polynomials to analyze $\mathcal{T}$ is more straightforward.

E Synthetic Graph Spectrum (SGS) Dataset

Our SGS dataset works for 3 types of node signal filtering regression tasks: high-pass, low-pass and band-pass filters, given in Equation 24 25 26, respectively. Models trained on our dataset are expected to learn the filtering response of the 3 types of filters. For each type, we generate 1000, 1000 and 2000 undirected graphs with graph signals in training set, validation set and test set, repectively. Each graph approximately has 80 ~120 nodes and 80 ~350 edges.

$$F_{HP}(x) = \frac{1}{1 + \exp\{-50(x-1)\}} \tag{24}$$

$$F_{LP}(x) = 1 - \frac{1}{1 + \exp\{-50(x-1)\}} \tag{25}$$

$$F_{BP}(x) = \frac{-1}{1 + \exp\{-100(x-1.05)\}} + \frac{1}{1 + \exp\{-100(x-0.95)\}} \tag{26}$$

Undirected graphs are randomly sampled through rejection sampling of edges from complete graphs. In detail, we randomly draw an integer N from $[80, 120]$ as the number of nodes, and then generate a $N \times N$ random matrix $\boldsymbol{B}$ with each element independently sampled from $\mathrm{Unif}(0, 1)$. Set a threshold ϵ and we can construct an adjacency matrix $\boldsymbol{A}$ by letting $a_{i,j} = 1$ if $b_{i,j} > \epsilon$, otherwise $a_{i,j} = 0$, where $a_{i,j}$ is the element located at the i-th row and j-th column of $\boldsymbol{A}$.

Next, we need to generate spectral signals $\boldsymbol{s}$ for the graph. Independent sampling for each spectrum from a probabilistic distribution will always generate noises. Hence, we synthesize spectrum by summing random functions. We notice that beta function $\mathrm{Beta}(a, b)$ is a powerful tool to yield diverse curves by tuning a and b. Also, Gaussian function $\mathrm{Norm}(\mu, \sigma)$ is able to yield diverse bell-shaped curves by tuning μ and σ. On basis of our findings, we sum 2 discretized beta functions, 4 discretized Gaussian functions with random parameters (uniformly sampled from their domains), and some Gaussian noises to generate spectral signals.

In most real-world cases, graph signals are usually represented in vertex-domain. With a generated graph and its spectral signals, we can retrieve the vertex-domain signals by inverse graph Fourier transformation: perform eigen-decomposition on the normalized adjacency matrix $\tilde{\boldsymbol{A}}$ of the graph to retrieve its graph Fourier basis $\tilde{\boldsymbol{U}}$, then we can find vertex-domain signals by $\tilde{\boldsymbol{U}}\boldsymbol{s}$.

For supervising purpose, we retrieve the groundtruth of each filter's response by applying the filter to the generated spectral signals. Figure 4 illustrates an example of the generated spectral signals and its filtering response of the 3 filters.

F More Visualizations of Spectrum on ZINC Dataset

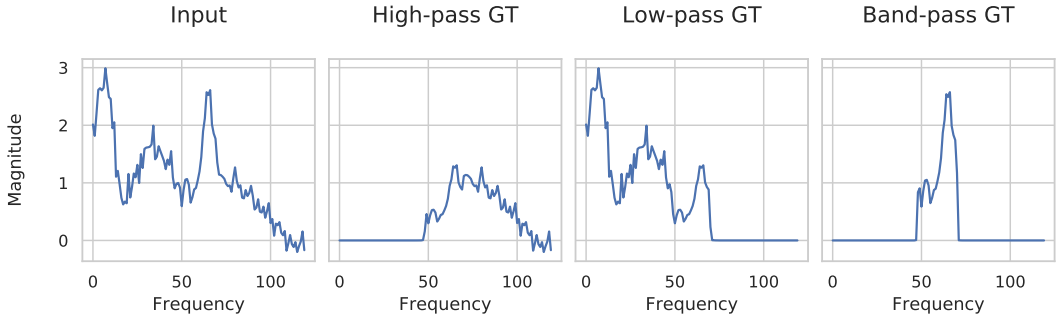


Figure 4: An example of graph spectrum in our SGS dataset and its corresponding high-pass, low-pass and band-pass filtering response using our hand-crafted filters.

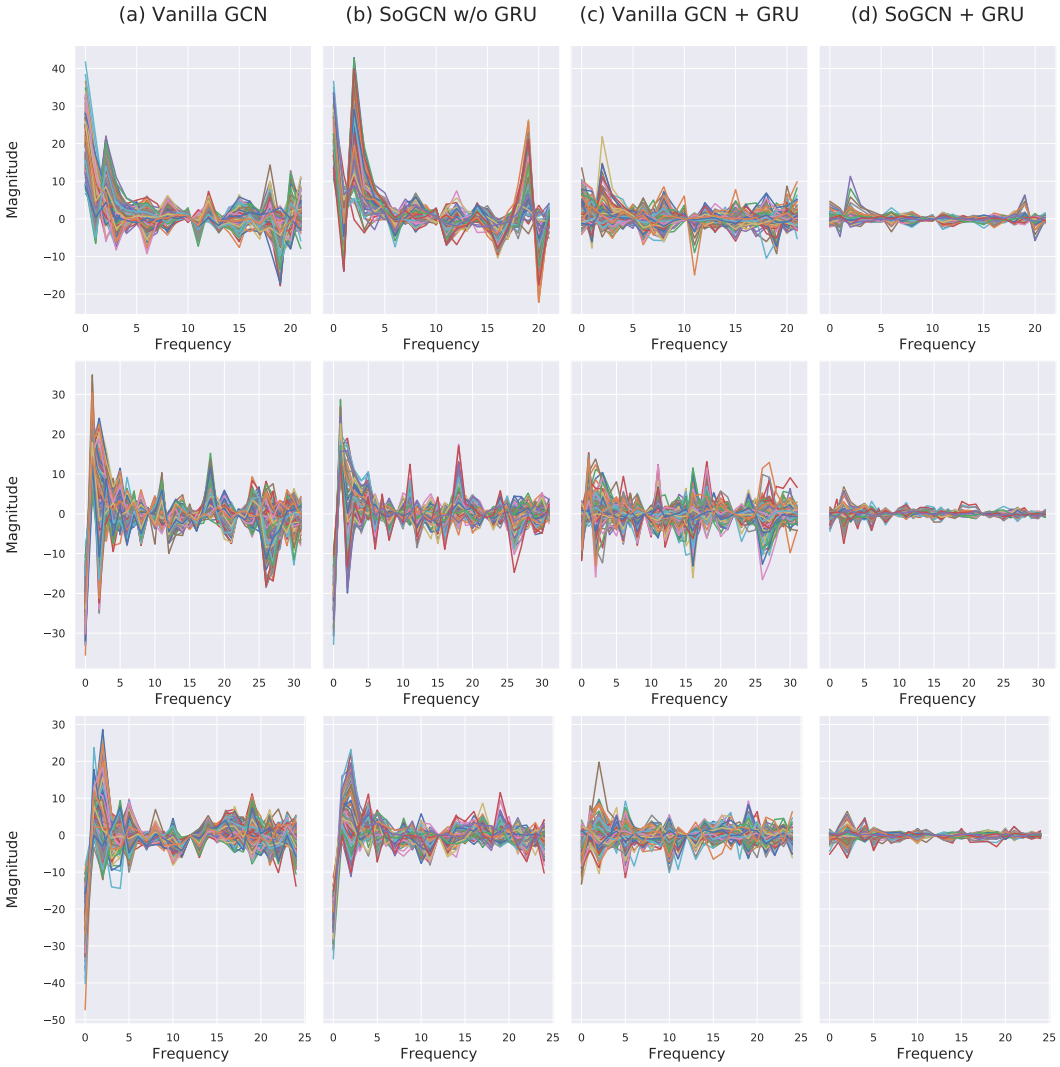


Figure 5: More visualizations of spectrum on the ZINC dataset.