SoGCN: Second-Order Graph Convolutional Networks Supplementary Material

Anonymous Authors[1]

A. Remark on Definition 1

Let us rewrite the $\mathcal{F}_K$ following Definition 1:

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

We claim that functions $f \in \mathcal{F}_K : \mathcal{G} \times \mathbb{R}^N \to \mathbb{R}^N$ are all Linear Shift-Invariant (LSI) to adjacency matrix.

Proof. Given arbitrary graph $G \in \mathcal{G}$, any filter $\boldsymbol{H}$ associated with it can be written as below:

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

where $\boldsymbol{A}(G) = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$ is the eigendecomposition of $\boldsymbol{A}(G)$. Therefore, $\boldsymbol{H}(G)$ is also diagonalized by the eigenvectors of $\boldsymbol{A}(G)$. By the Lemma 1:

Lemma 1. *Diagonalizable matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ are simultaneously diagonalized if and only if $\boldsymbol{A}_1\boldsymbol{A}_2 = \boldsymbol{A}_2\boldsymbol{A}_1$.*

we say that $\boldsymbol{H}(G)$ commutes with $\boldsymbol{A}(G)$. For any $f \in \mathcal{F}_K$, $\boldsymbol{A}(G) f(G, \boldsymbol{x}) = f(G, \boldsymbol{A}(G)\boldsymbol{x})$. □

B. Ring Isomorphism $\pi : \mathcal{F}_K \to \mathcal{T}_K$

We introduce a mathematical device that bridges the gap between the filter space $\mathcal{F}_K$ and the polynomial space $\mathbb{R}_K[x]$.

Since $\mathcal{G}$ is finite, we can construct a block diagonal matrix $\boldsymbol{T} \in \mathbb{R}^{N|\mathcal{G}| \times N|\mathcal{G}|}$, with adjacency matrix of every graph on the diagonal:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{A}(G_1) & & \\ & \ddots & \\ & & \boldsymbol{A}(G_{|\mathcal{G}|}) \end{bmatrix} \in \mathbb{R}^{N|\mathcal{G}| \times N|\mathcal{G}|}. \quad \text{(S1)}$$

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Remark 1. *The spectrum capacity Γ in Definition 2 represents the number of eigenvalues of $\boldsymbol{T}$ without multiplicity.*

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 provides the upper bound of Γ, so our theories remain their generality.

Now we construct a matrix space $\mathcal{T}_K$ by applying a ring homomorphism $\pi : \mathcal{F}_K \to \mathcal{T}_K$ to every element in $\mathcal{F}_K$:

$$\pi : \sum_{k=0}^{K} \theta_k \boldsymbol{A}(G)^k \mapsto \sum_{k=0}^{K} \theta_k \boldsymbol{T}^k. \quad \text{(S2)}$$

Concretely, we write the matrix space $\mathcal{T}$ as follows:

$$\mathcal{T}_K = \left\{ \boldsymbol{H} : \boldsymbol{H} = \sum_{k=0}^{K} \theta_k \boldsymbol{T}^k, \forall \theta_k \in \mathbb{R} \right\}. \quad \text{(S3)}$$

In the rest section, we prove that π is a ring isomorphism.

Proof. First, we can verify that π is a ring homomorphism because it is invariant to "summation" and "multiplication". Second, we can prove its surjectivity by the definition of $\mathcal{T}_K$ (cf. Equation S3).

Finally, we show its injectivity as follows: Consider any pair of $f_1, f_2 \in \mathcal{F}_K, f_1 \neq f_2$ with parameters $\alpha_k, \beta_k \in \mathbb{R}, k = 0, \cdots, K$, there exists $G_j \in \mathcal{G}$ and $\boldsymbol{x} \in \mathbb{R}^N$ such that $f_1(G_j, \boldsymbol{x}) \neq f_2(G_j, \boldsymbol{x})$. After applying π, we have their images $\boldsymbol{H}_1 = \pi(f_1), \boldsymbol{H}_2 = \pi(f_2)$. Let $\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{0}_{N(j-1)}^T & \boldsymbol{x}^T & \boldsymbol{0}_{N(|\mathcal{G}|-j)}^T \end{bmatrix}^T$, where $\boldsymbol{0}_N$ denote the all-zero vector of length N, then we have:

$$\begin{aligned} \boldsymbol{H}_1\boldsymbol{\xi} &= \begin{bmatrix} \boldsymbol{0}_{N(j-1)}^T & \left(\sum_{k=0}^{K} \alpha_k \boldsymbol{A}(G_j)^k \boldsymbol{x} \right)^T & \boldsymbol{0}_{N(|\mathcal{G}|-j)}^T \end{bmatrix}^T \\ &= \begin{bmatrix} \boldsymbol{0}_{N(j-1)}^T & f_1(G_j, \boldsymbol{x})^T & \boldsymbol{0}_{N(|\mathcal{G}|-j)}^T \end{bmatrix}^T, \end{aligned}$$

$$\begin{aligned} \boldsymbol{H}_2\boldsymbol{\xi} &= \begin{bmatrix} \boldsymbol{0}_{N(j-1)}^T & \left(\sum_{k=0}^{K} \beta_k \boldsymbol{A}(G_j)^k \boldsymbol{x} \right)^T & \boldsymbol{0}_{N(|\mathcal{G}|-j)}^T \end{bmatrix}^T \\ &= \begin{bmatrix} \boldsymbol{0}_{N(j-1)}^T & f_2(G_j, \boldsymbol{x})^T & \boldsymbol{0}_{N(|\mathcal{G}|-j)}^T \end{bmatrix}^T. \end{aligned}$$

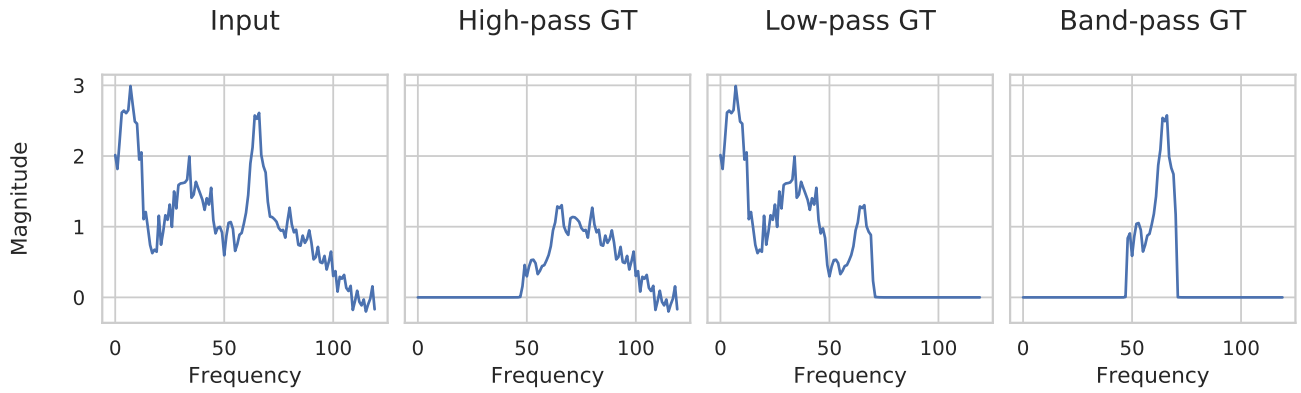


Figure S1. An example of graph spectrum in our SGS dataset and its corresponding high-pass, low-pass and band-pass filtered output using our hand-crafted filters.

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

C. Proof of Lemma 1

Proof. One can show $\mathcal{F}_K$ is a vector space by verifying the linear combination over $\mathcal{F}_K$ is closed (or simply implied from the ring isomorphism π).

Due to isomorphism, $\dim \mathcal{F}_K = \dim \mathcal{T}_K$. Then Lemma 1 follows from Theorem 3 of Sandryhaila & Moura (2013). We briefly conclude the proof as below.

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

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

$$h(x) = q(x)m(x) + r(x), \tag{S4}$$

where $\deg r(x) < \deg m(x) = \Gamma$. We insert $\boldsymbol{T}$ into Equation S4 as below:

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

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

D. Proof of Lemma 2

Proof. Consider a mapping $\varphi : \mathcal{T}_K \to \mathbb{R}_K[x]$:

$$\varphi : \sum_{k=0}^{K} \theta_k \boldsymbol{T}^k \mapsto \sum_{k=0}^{K} \theta_k x^k. \tag{S5}$$

When $K+1 \leq \Gamma$, $\dim \mathcal{T}_K = \dim \mathbb{R}_K[x]$ (as $\deg m(x) = \Gamma$), which implies φ is a ring isomorphism as well. Since function composition preserves isomorphism property, we can conclude the proof by showing that $\tau = \varphi \circ \pi$. □

Remark 2. *The assumption that each graph has the same number of vertices is made only for the sake of simplicity. Lemma 1 and Lemma 2 still hold when the vertex numbers are varying, since the construction of $\boldsymbol{T}$ (cf. Equation S1) is independent of this assumption.*

Remark 3. *The graph set $\mathcal{G}$ need to be finite, otherwise Γ might be uncountable. We leave the discussion on infinite graph sets for future study.*

E. Synthetic Graph Spectrum Dataset

Our Synthetic Graph Spectrum (SGS) dataset is designed for testing the filter fitting power of spectral GCNs. It includes 3 types of graph signal filters: High-Pass (HP), Low-Pass (LP) and Band-Pass (BP) filters. For each type, we generate 1k, 1k and 2k undirected graphs along with graph signals and groundtruth response in training set, validation set and test set, respectively. Each graph has 80˜120 nodes and 80˜350 edges. Models are trained on each dataset to learn the corresponding filter by supervision on the MAE loss.

For each sample, we generate an undirected Erdős-Rényi random graph $G = (\mathcal{V}, \mathcal{E})$ with normalized adjacency matrix $\boldsymbol{A}$, i.e., the existence of the edge between each pair of nodes accords to a Bernoulli distribution $\mathrm{Bern}(p)$. In our experiments, we set $p = 0.02$ to satisfy $|\mathcal{E}(G)| = O(N)$. We also compute $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\boldsymbol{T}}$, where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ with $\lambda_N \geq \cdots \geq \lambda_1$ are eigenvalues, $\boldsymbol{U} = \begin{bmatrix}\boldsymbol{u}_1 & \cdots & \boldsymbol{u}_N\end{bmatrix}$ are corresponding eigenvectors.

Next, we generate input graph signals $\boldsymbol{s}$ on the spectral domain. Independent sampling for each frequency from a distribution tends to generate white noises. Hence, we

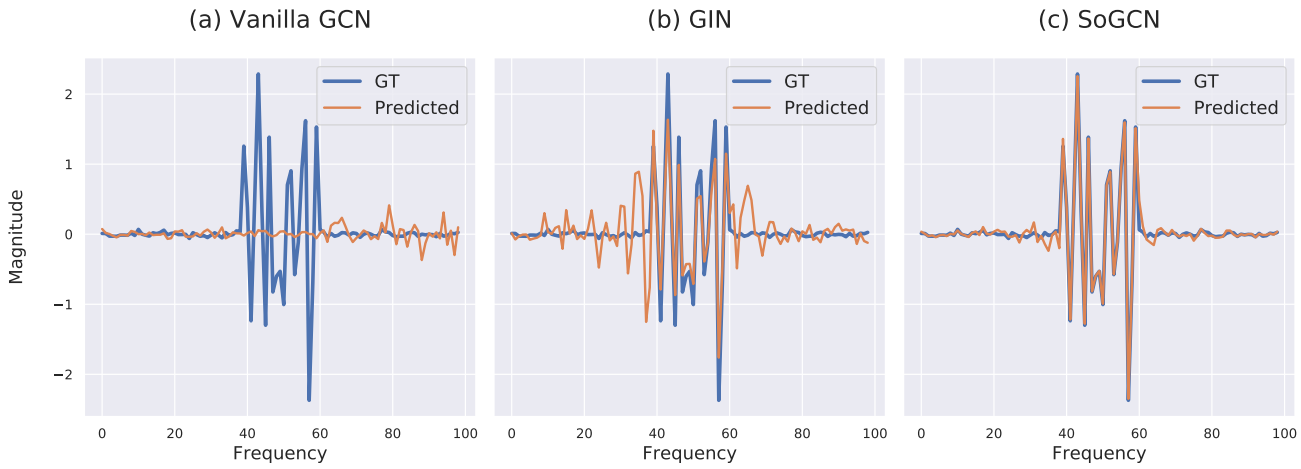


Figure S2. Visualize the spectrum of outputs from vanilla GCN, GIN and SoGCN on the SGS Band-Pass dataset.

synthesize spectrum by summing random functions. We notice the mixture of beta function $\mathrm{Beta}(a, b)$ and Gaussian function $\mathrm{Norm}(\mu, \sigma)$ is a powerful model to construct diverse curves by tuning shape parameters (a, b) and (μ, σ). We sum two discretized beta functions and four discretized Gaussian functions with random parameters to generate signal spectrums. Equation S6 elaborates the generation process and hyper-parameter chosen in our experiments, where $g[x; a, b]$ is the PDF of $\mathrm{Beta}(a, b)$ distribution, $f[x; \mu, \sigma]$ denotes the PDF of $\mathrm{Norm}(\mu, \sigma)$ distribution.

$$
\begin{aligned}
&\boldsymbol{s}_t = \sum_{i=1}^{2} g[t/N; a_i, b_i] + \sum_{j=1}^{4} c_j f[t; \mu_j, \sigma_j], t \in [N] \\
&a_i, b_i \sim \mathrm{Unif}\{0.1, 5\}, \quad \mu_j \sim \mathrm{Unif}\{0, N\}, \\
&\sigma_j \sim \mathrm{Unif}\left\{\frac{N}{(j+1)}, \frac{N}{j}\right\} \Big/ 9 \,, \\
&c_j \sim \frac{\mathrm{Unif}\{0.5, 2\}}{\max_{x \in [N]} f[x; \mu_j, \sigma_j]},
\end{aligned}
\tag{S6}
$$

We can retrieve the vertex-domain signals via inverse graph Fourier transformation: $\hat{\boldsymbol{x}} = \boldsymbol{U}\boldsymbol{s}$. Then Gaussian noise is added to the vertex-domain signals to simulate observation errors: $\boldsymbol{x} = \hat{\boldsymbol{x}} + \epsilon, \epsilon \sim \mathrm{Norm}(0, \mathrm{c}), c \sim \mathrm{Unif}(0.05, 0.35)$.

We design three filters $\boldsymbol{F}^*_{HP}$, $\boldsymbol{F}^*_{LP}$, $\boldsymbol{F}^*_{BP}$ in Equation S7:

$$
\begin{aligned}
&f^*_{HP}(\boldsymbol{s}) = \frac{1}{1 + \zeta(\boldsymbol{s}; 50, 1)}, \\
&f^*_{LP}(\boldsymbol{s}) = 1 - \frac{1}{1 + \zeta(\boldsymbol{s}; 50, 1)} \\
&f^*_{BP}(\boldsymbol{s}) = \frac{-1}{1 + \zeta(\boldsymbol{s}; 100, 1.05)} + \frac{1}{1 + \zeta(\boldsymbol{s}; 100, 0.95)}, \\
&\\
&\boldsymbol{F}^*_k = \boldsymbol{U} f^*_k(\boldsymbol{\Lambda}) \boldsymbol{U}^{\boldsymbol{T}}, k \in \{HP, LP, BP\},
\end{aligned}
\tag{S7}
$$

where $\zeta(\boldsymbol{s}; \alpha, \beta) = \exp\{-\alpha(\boldsymbol{s} - \beta)\}$. For supervising purpose, we applying each filter to synthetic inputs to generate the groundtruth output: $\boldsymbol{y} = \boldsymbol{F}^*_k \boldsymbol{x}, k \in \{HP, LP, BP\}$. Figure S1 illustrates an example of the generated spectral signals and the groundtruth responses of three filters.

F. More Visualizations of Spectrum

For multi-channel node signals $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, where N is the number of nodes and D is the number of signal channels, the spectrum of $\boldsymbol{X}$ is computed by $\boldsymbol{S} = \boldsymbol{U}^T \boldsymbol{X}$. More information about the graph spectrum and graph Fourier transformation can be found in Sandryhaila & Moura (2013).

Figure S2 shows the output spectrum of vanilla GCN, GIN and SoGCN on the synthetic Band-Pass dataset. The visualizations are consistent with the results in Table 2 and Figure 3 in the main text. Vanilla GCN almost loses all the band-pass frequency, resulting in very poor performance. GIN learns to pass a part of middle-frequency band but still has a distance from the groundtruth. SoGCN's filtering response is close to the groundtruth response, showing its strong ability to represent graph signal filters.

We arbitrarily sample graph data from the ZINC dataset as input and visualize the output spectrum of vanilla GCN, SoGCN and their GRU variants in Figure S3. Each curve in the visualization figure represents the spectrum of each output channel, i.e., each column of $\boldsymbol{S}$ is plotted as a curve.

G. More Experiments

G.1. Additional Experiments on SGS Dataset

We supplement two experiments to compare vanilla GCN (Kipf & Welling, 2017), GIN (Xu et al., 2019), SoGCN and 4th-order GCN on synthetic High-Pass and Low-Pass

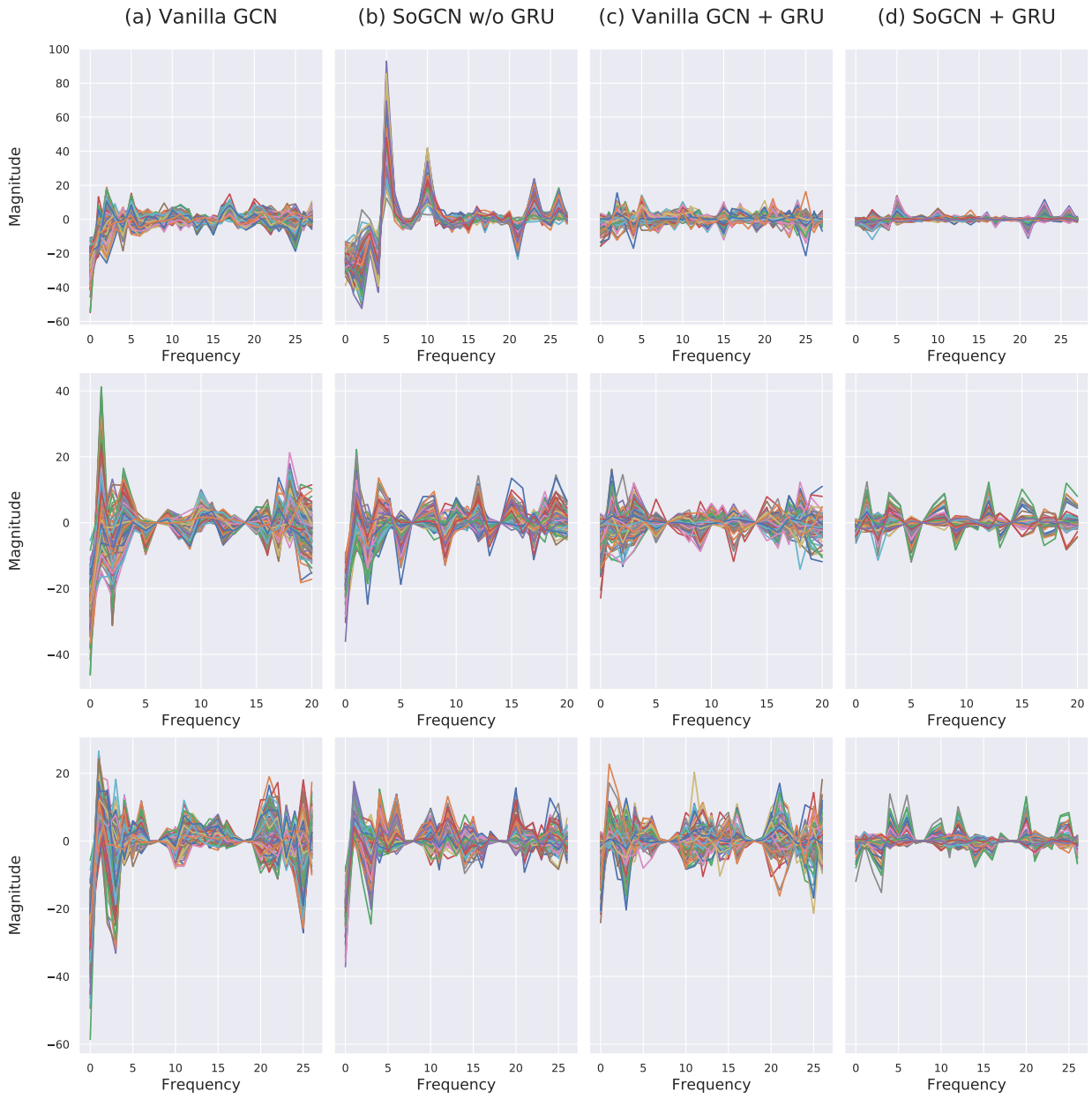


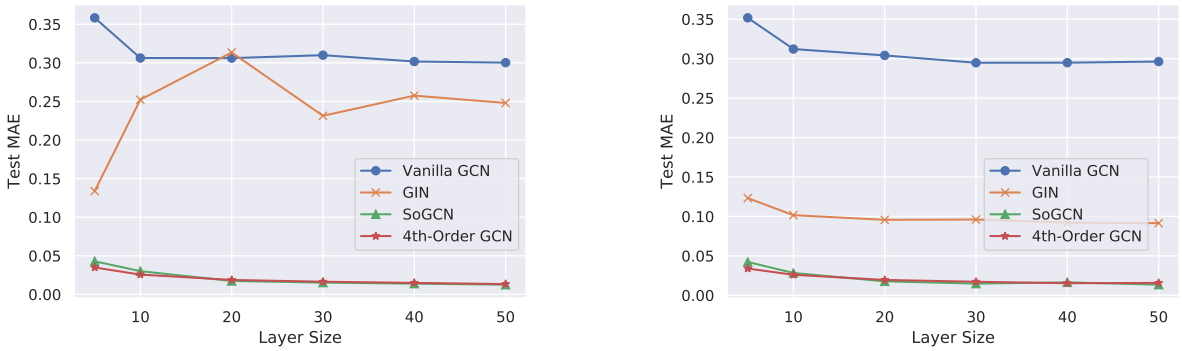
Figure S3. More visualizations of output spectrum on the ZINC dataset.

datasets, respectively. With Figure S4, we conclude that SoGCN and high-order GCNs perform closely on High-Pass and Low-Pass datasets and achieve remarkable filtering capability, while vanilla GCN and GIN cannot converge to considerable results by increasing the layer size. This conclusion is consistent with the previous results on Band-Pass dataset presented in the main text.

G.2. Additional Experiments on OGB Benchmark

In the main text, we have demonstrated our results on ogb-protein dataset (Hu et al., 2020) for node-level tasks. In this subsection, we also show our SoGCN's effectiveness on ogb-molhiv dataset (Hu et al., 2020) for graph-level tasks. As the same with experiments on ogb-protein, we evaluate different GCN models in terms of their total parameter numbers, training time per epoch, and test ROC-AUC.

Experiment Setup Again, we choose vanilla GCN, GIN, GraphSage(Hamilton et al., 2017), APPNP (Klicpera et al., 2019), GCNII (Ming Chen et al., 2020), ARMA (Bianchi et al., 2019), our SoGCN, and two high-order GCNs for performance comparison. We adopt the example code of vanilla GCN and GIN provided in OGB. We reimplemented



(a) Relation on High-Pass dataset

(b) Relation on Low-Pass dataset

Figure S4. Relations between test MAE and layer size. Each model has 16 channels per hidden layer with varying layer size.

Table S1. The performance of graph-level multi-label classification on ogb-molhiv dataset. We compare each model condering the following dimensions: the number of parameters, training time (in seconds) per epoch (ep.), and final test ROC-AUC (%).

Model	**ogb-molhiv**		
	#Param	Time / Ep.	ROC-AUC ± s.d.
Vanilla GCN	527,701	25.57 ± 1.37	76.06 ± 0.97
GIN	980,706	29.01 ± 1.24	75.58 ± 1.40
GCNII	524,701	24.19 ± 1.26	77.04 ± 1.03
APPNP	327,001	13.56 ± 1.32	68.00 ± 1.36
GraphSage	976,201	24.43 ± 1.39	76.90 ± 1.36
ARMA	8,188,201	43.14 ± 0.99	76.91 ± 1.75
SoGCN	1,426,201	27.02 ± 1.28	**77.26 ± 0.85**
4th-Order GCN	2,326,201	32.24 ± 1.10	77.24 ± 1.21
6th-Order GCN	3,226,201	37.64 ± 1.15	77.10 ± 0.72

GCNII, GraphSage, APPNP, ARMA based on the official code in PyTorch Geometric (Fey & Lenssen, 2019). According to the benchmark's guideline, we add edge features to fan-out node features while propagation. Every model has the same depth and width, as well as other modules. The timing, training and evaluation procedures conform with the descriptions in our main text. We train vanilla GCN, GIN, APPNP, GraphSage for ˜100 epochs, and SoGCN, higher-order GCNs, GCNII, ARMA for ˜500 epochs.

Results and Discussion. Table S1 demonstrates the ROC-AUC score for each model on ogb-molhiv dataset. We reach the same conclusion with our main text. On ogb-molhiv dataset, we notice that GCNII is another lightweight yet effective model. However, GCNII only allows inputs whose channel number equals to output dimension. One needs to add additional blocks (e.g., linear modules) to support varying hidden dimensions, which incorporates more parameters and higher complexity (e.g., on ogb-protein dataset).

H. Implementation Details

We open source our implementation of SoGCN at https://github.com/yuehaowang/SoGCN. All of our code, datasets, hyper-parameters, and runtime configurations can be found there.

H.1. Second-Order Graph Convolution

Our SoGC can be implemented using a message-passing scheme (Hamilton et al., 2017) (cf. Equation S8). We regard the normalized adjacency matrix $\boldsymbol{A}(G)$ as a one-hop aggregator (message propagator). When we compute the power of $\boldsymbol{A}(G)$, we invoke the propagator multiple times. After passing the messages twice, we transform and mix up aggregated information from two hops via a linear block.

$$\begin{aligned} \boldsymbol{h}_v^{(1)} &= \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_v d_u}} \boldsymbol{x}_u, \\ \boldsymbol{h}_v^{(2)} &= \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{d_v d_u}} \boldsymbol{h}_u^{(1)}, \\ \boldsymbol{y}_v &= \boldsymbol{\Theta}_2 \boldsymbol{h}_v^{(2)} + \boldsymbol{\Theta}_1 \boldsymbol{h}_v^{(1)} + \boldsymbol{\Theta}_0 \boldsymbol{x}_v, \end{aligned} \tag{S8}$$

where $\boldsymbol{x}_v \in \mathbb{R}^E$ is the input feature vector for node $v \in \mathcal{V}(G)$, $\boldsymbol{y}_v \in \mathbb{R}^F$ denotes the output for node v. d_v is the degree for vertex v, $\mathcal{N}(v)$ is the set of v's neighbor vertices. $\boldsymbol{h}_v^{(1)}$ is the feature representation of v's first-hop neighborhood. It can be computed by aggregating information once from the directly neighboring nodes. $\boldsymbol{h}_v^{(2)}$ is the feature representation of v's second-hop neighborhood. It can be computed by feature aggregation upon neighbors' $\boldsymbol{h}_u^{(1)}, u \in \mathcal{N}(v)$. $\boldsymbol{\Theta}_i \in \mathbb{R}^{F \times E}, i = 0, 1, 2$ are the weight matrices (a.k.a. layer parameters).

Our design can reduce computational time by reusing previously aggregated information and preventing power operations on $\boldsymbol{A}(G)$. In practice, our SoGC is easy to implement.

Our message-passing design conforms to mainstream graph learning frameworks, such as Deep Graph Library (Wang et al., 2019) and PyTorch Geometric (Fey & Lenssen, 2019). One can simply add another group of parameters and invoke the "propagation" method of vanilla GC (Kipf & Welling, 2017) twice to simulate our SoGC. For the sake of clarity, we provide the pseudo-code for general K-order GCs in Algorithm 1. Our SoGC can be called by passing $K = 2$.

Algorithm 1 K-Order Graph Convolution

Input: Graph $G = (\mathcal{V}, \mathcal{E})$, node degrees $\{d_v \in \mathbb{N}, \forall v \in \mathcal{V}\}$, input features $\{\boldsymbol{x}_v \in \mathbb{R}^E, \forall v \in \mathcal{V}\}$, GC order K, weight matrices $\boldsymbol{\Theta}_i \in \mathbb{R}^{F \times E}, i = 0, \cdots K$.
Output: Feature representation $\boldsymbol{y}_v \in \mathbb{R}^F, \forall v \in \mathcal{V}$.
$\boldsymbol{h}_v^{(0)} \leftarrow \boldsymbol{x}_v, \forall v \in \mathcal{V}$
$\boldsymbol{y}_v \leftarrow \boldsymbol{\Theta}_0 \boldsymbol{h}_v^{(0)}, \forall v \in \mathcal{V}$
for $t = 1$ **to** K **do**
  for $v \in \mathcal{V}$ **do**
    $\boldsymbol{h}_v^{(t)} \leftarrow \sum_{u \in \mathcal{N}(v)} \boldsymbol{h}_u^{(t-1)} \Big/ \sqrt{d_v d_u}$
  end for
  $\boldsymbol{y}_v \leftarrow \boldsymbol{y}_v + \boldsymbol{\Theta}_t \boldsymbol{h}_v^{(t)}, \forall v \in \mathcal{V}$
end for
Return $\{\boldsymbol{y}_v \in \mathbb{R}^F, \forall v \in \mathcal{V}\}$

H.2. Gated Recurrent Unit

We supplement two motivations behind using Gated Recurrent Unit (GRU) (Cho et al., 2014): 1) GRU has been served as a basic building block in message-passing GNN architectures (Li et al., 2016; Gilmer et al., 2017). We make an explorative attempt to first introduce them into spectral GCNs. 2) By selectively maintaining information from previous layer and canceling the dominance of DC components (Figure S3), GRU can also relieve the side-effect of ReLU, which is proved to be a special low-pass filter (Oono & Suzuki, 2019; Cai & Wang, 2020).

Similar to Li et al. (2016); Gilmer et al. (2017), we appends a shared GRU module after each GC layer, which takes the signals before the GC layers as the hidden state, after the GC layers as the current input. We formulate its implementation by replacing Equation 10 with Equation S9 as below.

$$\begin{aligned} \boldsymbol{X}_{conv}^{(l)} &= f_2^{(l)} \left(\boldsymbol{X}^{(l-1)}; \boldsymbol{\Theta}^{(l)} \right) \\ \boldsymbol{X}^{(l+1)} &= \mathrm{GRU} \left(\mathrm{ReLU} \left(\boldsymbol{X}_{conv}^{(l+1)} \right), \boldsymbol{X}^{(l)}; \boldsymbol{\Omega} \right), \end{aligned} \tag{S9}$$

where $\boldsymbol{X}_{conv}^{(l+1)}$ is the input, $\boldsymbol{X}^{(l)}$ represents the hidden state, $\boldsymbol{\Omega}$ denotes parameters of the GRU. Figure S3 illustrates the spectrum outputs of vanilla GCN + GRU and SoGCN + GRU. One can see, without filtering power of SoGCN, vanilla GCN + GRU fails to extract sharp patterns on the spectrum. Thereby, we suggest that it is SoGC that mainly contributes to the higher expressiveness.

References

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

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

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv:1406.1078*, 2014.

Fey, M. and Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *ICML*, 2017.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. *arXiv:2005.00687*, 2020.

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

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*, 2019.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. In *ICLR*, 2016.

Ming Chen, Z. W., Zengfeng Huang, B. D., and Li, Y. Simple and deep graph convolutional networks. In *ICML*, 2020.

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

Sandryhaila, A. and Moura, J. M. Discrete signal processing on graphs. *IEEE Trans. Signal Process*, 2013.

Wang, M., Zheng, D., Ye, Z., Gan, Q., Li, M., Song, X., Zhou, J., Ma, C., Yu, L., Gai, Y., Xiao, T., He, T., Karypis, G., Li, J., and Zhang, Z. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv:1909.01315*, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019.