How Powerful are Deep Localized Filters for Graph Convolutional Networks?

Peihao Wang[1] Yuehao Wang[1]
[1]ShanghaiTech University
{wangph, wangyh3}@shanghaitech.edu.cn

Abstract

In this project, we dive into the spectral analysis using theories of graph signal processing and propose an efficient yet powerful localized graph kernel using the second-order approximation of polynomials. By stacking such trainable kernels deeply enough, we prove that they can ideally achieve arbitrary linear filtering properties. This also concludes that no more gain on performance of deep graph convolution networks (GCNs) can be achieved unless sacrificing strict localization or shift-invariant property. Meanwhile, we implement our Second-Order approximate GCN (SoGCN) with the proposed graph kernel on the latest benchmark and verify that SoGCN can outperform the state-of-the-art methods in accuracy and efficiency, which suggests potential modification on topology in other work does not improve performance much.

1. Introduction

Graph neural networks (GNNs) have become the standard toolkit for analyzing and learning from data on graphs. They have been successfully applied to a myriad of domains including chemistry, physics, social sciences, knowledge graphs, recommendation, and neuroscience. As a generalization of convolutional neural networks (CNNs) on regular grids, graph convolutional neural networks (GCNs) have become one of the most important techniques to solve machine learning problems on graph-structured data. We note that GNNs are usually referred to more general models (e.g., message passing on graphs), while GCNs borrow the shift-invariant property from CNNs to deal with nodes regardless of their location.

GCNs are associated with graph convolution operator which is not naturally well-defined on graph topology. The conventional methods employ adjacency matrix or Laplacian matrix, which encodes the connectivity among nodes, to perform information aggregation with learnable parameters [13, 27, 8, 28]. The key feature of the aggregation is the localization, which means each node only receives message from its neighbors of several orders, mimicking tiny-size kernels on Euclidean spaces. However, due to the large degree of freedom, variety of the neighbor count and lack of prior correspondence, the localized aggregation is usually isotropic [13].

Therefore, critical voices [10] doubted that the capability of localized isotropic graph kernel proposed in [13] is no more than a low-pass filter. Deep GCNs stacking these filters will cause the *over-smoothing* problem. Intuitively, over-smoothed vertices are no more distinguishable. Sampling based approaches including [16, 15, 2] and anisotropic models which imposes gating [17] attention [24, 26] or structural data [21] on edges break topology and localization to overcome over-smoothing.

However, we believe that localized isotropic kernels can be equivalently powerful as long as filtering goes versatile. To this end, we intend to study the effectiveness of current existing graph filters, and try to answer how to extend them to be more powerful and expressive in this work.

We add that there are mainly two classes of convolution-kind operator: vertex-domain [8, 28, 29, 15] and spectral-domain [1, 9, 4, 13]. The former one is more efficient in time complexity and implementation than the latter one, which requires *eigendecomposition* on the graph Laplacian matrix. We deem that the key to understand and overcome over-smoothing is the spectral analysis on graph signals. As Kipf et al. [13] established the bridge between the vertex and spectral domains, we are inspired to make direct access to the spectrum for specific filtering property and then yield an efficient vertex-domain propagation scheme.

We summarize our contributions and conclusions as below:

1. Empowered by [22], we utilize theories of graph spectral theory [23] and graph signal processing [22, 5, 18] and introduce Theorem 1, 2, 3 and 4. By Theorem 1 and 3, we notice that linear shift-invariant operators on graphs (i.e., graph convolution) can be generally represented by a polynomial of the adjacency matrix. By Theorem 4, we propose that second-order polynomials of the adjacency matrix are more advisable than the first-order, because they are able to fit any linear filters.

2. Following [22, 9, 4], but unlike [13], we disallow the absorption of the constant term and take the second-order approximation for the Chebyshev polynomials. Consequently, we obtain a type of *band-stop* graph kernels that are *linear*, *shift-invariant*, *localized* and *trainable*. Furthermore, cascading and training such kernels can achieve linear filters with arbitrary filtering properties. As long as the layer number goes large enough and the kernels are well-fitted, it can reach the upper-bound of linear shift-invariant filters on graphs.

3. Based on the proposed graph filter, we build our Second-Order approximate GCN (SoGCN), whose layers comprise of the proposed kernel followed by an activation function to realize non-linear transformation. We verify the effectiveness of our novel SoGCN on the latest benchmark [6] and find out that the performance is comparable or even better than the state-of-the-art approaches [13, 24, 8] in various tasks. Hence, we draw the conclusion that these approaches accused of abandoning localization or shift-invariance do not make significant improvements, not more powerful than the cascading localized graph filters. This indicates that aggregation, embedding, or attention mechanisms may not be optimal yet, further explorations on these models are demanding.

2. Related Work

Graph Signal Processing Graph signal processing (GSP) basically borrows ideas from digital signal processing (DSP) to tackle data processing missions on graph structures or networks. Linear shift-invariant operators defined on graphs are so called graph convolution, which implicitly requires a shift operator. Shift operators, however, are not well-defined yet [22, 5, 18]. The basic requirement is that a shift should encode connectivity among nodes. Adjacency matrix or Laplacian matrix are considered as good shift as [22, 18] suggest. The spectral domain and the graph Fourier transform (GFT) are induced by the eigenspaces of the shift. We give the detail explanations in Section 3.

Graph Convolution Networks GCNs were originated from the messaging propagation models on graphs. Groundbreaking work for GCNs [1, 9, 4, 13] leveraged theories of spectral GSP and created less agnostic learning models on the spectral domain. Kipf and Welling [13] simplified this model into a first-order propagation with isotropic weighting model using approximation and renormalization tricks on Chebyshev polynomials. In [8], the authors further extend this to a aggregator-combiner model on the vertex domain, where aggregator collects features from the neighbor nodes and combiner performs weighting on the features.

The combiners are commonly multi-layer perceptrons, while most recent research conducts discussion on the aggregators since they are hard to parametrize. GraphSage [8] argued that such aggregators should be permutation-invariant; GAT [24] exploits the attention mechanism; GIN [28] discussed which aggregator should be powerful; GatedGNN [17] performs feature propagation with edge gating; GeomGCN [21] explored latent structural neighbors.

We follow definitions in GSP [18, 4]: such anisotropic filters in [17, 24, 21] actually change the edge connectivity, thus do not abide by the adjacency matrix explicitly. We refer to these approaches as using non-localized kernels. We intend to argue that such non-localized kernels do not guarantee the effectiveness. On the contrary, we defend the localized filters in [13, 8, 28]. However, layer units of first-order localization is not sufficient, second-order layers should be preferred.

Over-smoothing and Criticism The earliest GCNs [13] cannot go deep to increase accuracy, as over-smoothing problem will harm the result. Some new isotropic methods [8, 28, 29] demonstrate that more importance on the central node feature can improve performance, while anisotropic models [24, 17, 25] can benefit even more. Nevertheless, these GCNs still result in over-smoothing and cannot go deeper either.

GIN [28] argued that capability of GNNs is as powerful as the Weisfeiler-Lehman graph isomorphism test. [10] gave theoretic analysis and draw the conclusion that GCNs are no more than a low-pass filter. However, [16] explained that GCNs are effective thanks to the Laplacian smoothing; DeepGCNs [15] and FeaStNet [25] managed to adopt very deep networks (over 50 layers) for point cloud processing with too much tricks (e.g. dynamic edges) involved.

The first attempts to directly relieve over-smoothing [26, 2] rarely pay attention to the spectral domain. State-of-the-art models [21, 24, 17] did not clarify their effectiveness on spectral domain.

3. Preliminaries

To address our work, we formulate the problem settings using GSP models. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ contains finite vertex set $\mathcal{V}$ with $|\mathcal{V}| = N$ and edge set $\mathcal{E} = \{(v_i, v_j) : \forall v_i \to v_j\}$, associated with an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ which indicates the connectivity and weight of each pair of vertices. Accordingly, we define degree matrix $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$ and Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$. In this project, we only consider simple undirected graphs without negative or parallel edges. Therefore, $\boldsymbol{A}$ should be symmetric, and $\boldsymbol{L}$ should be symmetric positive semidefinite. We further introduce normalized adjacency matrix $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ and normalized Laplacian matrix $\tilde{\boldsymbol{L}} = \boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$.

Graph-structured data are often organized by a data matrix $\boldsymbol{X} \in \mathbb{R}^{N\times C}$, where C is the number of channels (or dimensionality of features). This can also be viewed as a mapping $\mathcal{V} \to \mathbb{R}^C$. The learning problem is to search for a model (or function) $f_{\boldsymbol{\theta}}$ with trainable parameter $\boldsymbol{\theta}$ to regress specified results, utilizing the graph topology.

3.1. Graph Fourier Transform

In this project, analysis on the spectral domain is vitally important. Hence, we are introducing the mathematical framework borrowed from GSP in this section. We start from the definition of total variation $\Delta(\boldsymbol{x})$ on single-channel graph signal $\boldsymbol{x} \in \mathbb{R}^N$ under graph $\mathcal{G}$ as below [10].

$$\Delta(\boldsymbol{x}) := \sum_{(i,j)\in\mathcal{E}} w_{ij}(x_i - x_j)^2 = \boldsymbol{x}^T\boldsymbol{L}\boldsymbol{x} \tag{1}$$

where w_{ij} are entries in adjacency matrix $\boldsymbol{A}$. This formulation describes the weighted summation over variations between connected vertices, which becomes natural to describe frequencies on the graph. Larger $\Delta(\boldsymbol{x})$ means higher frequency and vice versa.

To induce Fourier basis, we determine the orthonormal signals $\{\boldsymbol{u}_i\}_{i=1}^N$ that signify distinct frequency bands.

$$\boldsymbol{u}_i := \underset{\substack{\boldsymbol{x}\perp\boldsymbol{u}_j,\forall j<i\\ \|\boldsymbol{x}\|_2=1}}{\operatorname{argmax}} \Delta(\boldsymbol{x}) \tag{2}$$

which is equivalent to the Theorem 7.4 in [23]. We provide the proof in the Appendix A. So far, Courant-Fischer Theorem tells us that $\{\boldsymbol{u}_i\}_{i=1}^N$ are the eigenvectors of $\boldsymbol{L}$ and $\{\lambda_i = \Delta(\boldsymbol{u}_i)\}_{i=1}^N$ are eigenvalues of $\boldsymbol{L}$ in descending order.

If the graph signal is decomposed into $\boldsymbol{x} = \sum_{i\in[N]} \xi_i\boldsymbol{u}_i$, then $\Delta(\boldsymbol{x}) = \sum_{i\in[N]} \xi_i^2\lambda_i$. And we call $\hat{\boldsymbol{x}} = [\xi_1 \cdots \xi_N]^T$ the spectral-domain response of vertex-domain signal $\boldsymbol{x}$. And we can compute the corresponding spectrum via Graph Fourier Transform (GFT) as follows.

$$\hat{\boldsymbol{x}} = \mathcal{F}\{\boldsymbol{x}\} = \boldsymbol{U}^T\boldsymbol{x} \qquad \boldsymbol{x} = \mathcal{F}^{-1}\{\hat{\boldsymbol{x}}\} = \boldsymbol{U}\hat{\boldsymbol{x}} \tag{3}$$

where $\boldsymbol{U} = [\boldsymbol{u}_1 \cdots \boldsymbol{u}_N]$ is the eigenspace matrix of $\boldsymbol{L}$.

3.2. Linear Shift-Invariant Operator

Linear shift-invariant operators in DSP are namely convolution. In GSP, it is desirable to inherit the same property for graph convolution. The reason is that we hope to distinguish nodes according to their features regardless of their locations. A shift operator $\boldsymbol{S} \in \mathbb{R}^{N\times N}$, thus, should be defined, which propagates data along the specified topology $\mathcal{G}$ [18]. An intuitive example is the cycle matrix for 1D Euclidean space. We add that good shifts should encode topology, maintain localization, and preserve energy. Adjacency matrices are desirable but without energy preserving.

Afterward, we define convolutional operators on the graph $\mathcal{G}$, which are also called filters and kernels in this report. Each such filter can be represented by $\boldsymbol{H} \in \mathbb{R}^{N\times N}$, which satisfies the following relationship with shift $\boldsymbol{S}$.

$$\boldsymbol{H}\boldsymbol{S}\boldsymbol{x} = \boldsymbol{S}\boldsymbol{H}\boldsymbol{x} \tag{4}$$

given arbitrary graph signal $\boldsymbol{x} \in \mathbb{R}^N$. This equation suggests that filtering after shifting is equivalent to shifting followed by filtering. [22, 5] demonstrate the comparison with convolution for DSP.

4. Our Approach

In this section, we will leverage graph signal processing theories [22, 5, 18] to yield our differentiable graph kernels thought to be powerful. The kernels maintain atomic, linear, shift-invariant, and **localization**, enabling us to further build deep neural networks on it. We emphasize localization because we think it should be an intrinsic and inherent property of graph kernels. State-of-the-art approaches [17, 24, 21] incline to abandon constrains of fixed edge connections for better performance, while we exploit localized convolution layers to achieve similar results making compelling argument.

4.1. Commutative Algebra $\mathcal{A}$ of Filter Space $\mathcal{H}$

Suppose we have an ideal but unavailable shift $\boldsymbol{S}$ and tractable shift e.g., adjacency matrix $\boldsymbol{A}$ for graph $\mathcal{G}$. We notice that $\boldsymbol{A}$ has incoherent filtering property while ideal shifts are ill-defined as [7] and [3] suggest.

Now we intend to tackle the definition of graph kernels. Let $\mathcal{H} = \{\boldsymbol{H} : \boldsymbol{H}\boldsymbol{S} = \boldsymbol{S}\boldsymbol{H}\}$ be the filter space satisfying Equation 4. To study $\mathcal{H}$ of linear shift-invariant operators on graphs, we introduce Theorem 1 from [22].

Theorem 1. *Suppose the characteristic and minimal polynomials of shift $\boldsymbol{S}$ are equal: $m_{\mathbf{S}}(x) = p_{\mathbf{S}}(x)$. Then a linear transformation $\boldsymbol{H}$ is linear shift-invariant if and only if $\boldsymbol{H}$ is a polynomial in the graph shift $\boldsymbol{S}$.*

Theorem 1 has been proved in [22] over complex domain. We provide the real-domain version in Appendix B. Then we prove Theorem 2 that shows any graph filter has a equivalent filter on the same graph with at most N taps.

Theorem 2. *With the same assumption of Theorem 1, for any filter $\boldsymbol{H} \in \mathcal{H}$, there exists polynomial $h(x)$ such that $\boldsymbol{H} = h(\boldsymbol{S})$ with $\deg h(x) \le N-1$.*

The proof can be found in Appendix C. By Theorem 1 and 2, we convert $\mathcal{H}$ into polynomial algebra $\mathcal{A} = \{p(\boldsymbol{S}) : \text{polynomial in } \boldsymbol{S}\}$ with $\dim \boldsymbol{A} = N$ as vector space [22, 5].

However, acquiring such graph shift $\boldsymbol{S}$ is infeasible, the most available one is the adjacency matrix $\boldsymbol{A}$, which may

not have $m_{\mathbf{A}}(x) = p_{\mathbf{A}}(x)$. Theorem 3 guarantees that using adjacency matrix $\boldsymbol{A}$ as the shift can be effective as well.

Theorem 3. *For any matrix $\boldsymbol{A}$, there exists a latent shift $\boldsymbol{S}$ and polynomial $g(x)$ such that $\boldsymbol{A} = g(\boldsymbol{S})$ and $m_{\mathbf{S}}(x) = p_{\mathbf{S}}(x)$. Meanwhile, $\boldsymbol{A}$ and $\boldsymbol{S}$ share the same eigenspace.*

Please check Appendix D for the proof. As a consequence of Theorem 3, any filter on the graph $\mathcal{G}$ as a polynomial in $\boldsymbol{A}$ is equivalent to another polynomial in an ideal graph shift $\boldsymbol{S}$. Formally, if we have $\boldsymbol{H} = h(\boldsymbol{A})$, then $\boldsymbol{H} = h \circ g(\boldsymbol{S})$ turns out to be a composition polynomial in $\boldsymbol{S}$. Moreover, we raise a notion on isometric shift operator for further research [7, 3].

4.2. Graph Filters with Normalization

As adjacent matrix has been shown effective, we begin with choosing normalized adjacency matrix $\tilde{\boldsymbol{A}}$ of $\mathcal{G}$ as the shift and substituting $\tilde{\boldsymbol{L}}$ for $\boldsymbol{L}$ in Equation 1 [13]. We still have the meaningful definition of total variation.

$$\tilde{\Delta}(\boldsymbol{x}) := \sum_{(i,j)\in\mathcal{E}} w_{ij}\left(\frac{x_i}{\deg(i)} - \frac{x_j}{\deg(j)}\right)^2 = \boldsymbol{x}^T\tilde{\boldsymbol{L}}\boldsymbol{x} \quad (5)$$

where $\deg(i)$ is the degree of i-th node in graph $\mathcal{G}$. This revised variation resembles the objective of normalized cut [11]. We can derive GFT on $\tilde{\boldsymbol{L}}$ following Equation 2 and 3.

$$\hat{\boldsymbol{x}} = \tilde{\mathcal{F}}\{\boldsymbol{x}\} = \tilde{\boldsymbol{U}}^T\boldsymbol{x} \qquad \boldsymbol{x} = \tilde{\mathcal{F}}^{-1}\{\hat{\boldsymbol{x}}\} = \tilde{\boldsymbol{U}}\hat{\boldsymbol{x}} \quad (6)$$

where $\tilde{\boldsymbol{U}} = [\tilde{\boldsymbol{u}}_1 \cdots \tilde{\boldsymbol{u}}_N]$ consists of eigenvectors of $\tilde{\boldsymbol{L}}$. Let $\tilde{\boldsymbol{\Lambda}} = \mathrm{diag}(\tilde{\lambda}_1, \cdots, \tilde{\lambda}_N)$ are corresponding eigenvalues, then $\tilde{\boldsymbol{L}} = \tilde{\boldsymbol{U}}\tilde{\boldsymbol{\Lambda}}\tilde{\boldsymbol{U}}^T$. Note that, $\tilde{\lambda}_i$ ranges in $[0, 2]$ with $\tilde{\lambda}_N = 0$. Hence, eigenvalues of $\tilde{\boldsymbol{A}}$ fall within $[-1, 1]$ with achievable maximal value 1.

Suppose a graph filter $\boldsymbol{h}$ runs convolution on the signal $\boldsymbol{x}$, it is equivalent to considering performing multiplication on spectral domain, analogous to DSP.

$$\boldsymbol{x} * \boldsymbol{h} := \boldsymbol{H}\boldsymbol{x} = \tilde{\mathcal{F}}^{-1}\left\{\tilde{\mathcal{F}}\{\boldsymbol{x}\} \odot \tilde{\mathcal{F}}\{\boldsymbol{h}\}\right\} \quad (7)$$

$$= \tilde{\boldsymbol{U}}\left(\tilde{\boldsymbol{U}}^T\boldsymbol{h}\right) \odot \left(\tilde{\boldsymbol{U}}^T\boldsymbol{x}\right) \quad (8)$$

$$= \tilde{\boldsymbol{U}}\boldsymbol{\Lambda}_{\boldsymbol{h}}\tilde{\boldsymbol{U}}^T\boldsymbol{x} \quad (9)$$

where $*$ is defined as the convolutional operator, $\odot$ denotes element-wise multiplication and $\boldsymbol{\Lambda}_{\boldsymbol{h}}$ is the diagonal matrix. This yields that $\boldsymbol{H} = \tilde{\boldsymbol{U}}\boldsymbol{\Lambda}_{\boldsymbol{h}}\tilde{\boldsymbol{U}}^T$ is simultaneously diagonalized with $\tilde{\boldsymbol{A}}$ consistent with Theorem 1. Therefore, we can compute $\boldsymbol{H}$ from $\tilde{\boldsymbol{A}}$ by fitting a polynomial with an order up to $N-1$ indicated by Theorem 2.

$$\boldsymbol{H}(\boldsymbol{\theta}) = h_{\boldsymbol{\theta}}\left(\tilde{\boldsymbol{A}}\right) = \sum_{k=0}^{N-1}\theta_k\tilde{\boldsymbol{A}}^k \quad (10)$$

where $\boldsymbol{\theta}$ are the coefficients to fit this polynomial. In context of neural networks, $\boldsymbol{\theta}$ are so called training parameters.

4.3. Second-Order Approximation

However, dealing with $(N-1)$-th order polynomial is computationally prohibitive [4, 13]. We borrow the K-th order approximation from [4].

$$\boldsymbol{H}_K(\boldsymbol{\theta}) = \sum_{k=0}^{K}\theta_k\tilde{\boldsymbol{A}}^k \quad (11)$$

which reduces dimensionality of $\boldsymbol{\theta}$ to $K+1$. We no more refer to Chevbyshev polynomials here [4, 13].

An interpretation of Equation 11 is that: $\tilde{\boldsymbol{A}}^k$ encodes neighbor nodes within k hops. The summation means weighted aggregation of their features within such neighborhood. To impose localization of a single kernel, K should be small. [13] selects $K = 1$ and shares the coefficients jointly. The kernel they can acquire is of the form $\boldsymbol{H} = \frac{\theta}{2}\left(\boldsymbol{I} + \tilde{\boldsymbol{A}}\right)$ [1], whose spectral response ranges within $[0, \theta]$. Applying this filter will shrink the high-frequency component while preserving the low-frequency component as if a low-pass filter [10].

GCNs cannot be powerful by only cascading the low-pass filters above. Therefore, we propose Theorem 4 which suggests what kind of localized kernel should we stack to implement arbitrary filtering. Before presenting Theorem 4, we give the formulation of linear shift-invariant filters by stacking kernel units of K-hop localization.

$$\boldsymbol{H} = \prod_{l=1}^{L}\boldsymbol{H}_K(\boldsymbol{\theta}_l) \quad (12)$$

where L is the number of layers, $\boldsymbol{\theta}_l \in \mathbb{R}^{K+1}$ is the parameters for l-th layer. Then we give Theorem 4 as below.

Theorem 4. *Any filter $\boldsymbol{H}$ can be decomposed to the product of second-order approximate filter ($K = 2$). Formally, there exists $\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_L$ such that $\boldsymbol{H} = \prod_{l=1}^{L}\boldsymbol{H}_2(\boldsymbol{\theta}_l)$, where $\boldsymbol{\theta}_l \in \mathbb{R}^3$ and $L \leq N-1$.*

We prove this theorem in Appendix E. Theorem 4 reveals that the first-order approximation and absorption of the constant term is defective. First, the constant term draws importance on the central node, which has been shown more effective in [28]. Second, we propose that to reach the upper-bound of the linear shift-invariant kernels on graph signals by deeply stacking localized filtering layers, the second-order units $\boldsymbol{H}_2$ are more advantageous. Higher-order unit kernels are inefficient in memory and may lose localization, lower-order ones are, however, too weak.

Algebraically, stacked kernels from Kipf et al [13] only span a subspace of $\mathcal{A}$: $\boldsymbol{H} = \theta\sum_{k=0}^{L}\binom{L}{k}\tilde{\boldsymbol{A}}^k$, while the second-order approximate kernels span the whole space.

[1] We extract an $1/2$ as the normalization term.

With this, the formulation of our layer can be yielded from Equation 11.

$$\boldsymbol{X}^{(l+1)} = \sigma\left(\tilde{\boldsymbol{A}}^2\boldsymbol{X}^{(l)}\boldsymbol{\Theta}_2^{(l)} + \tilde{\boldsymbol{A}}\boldsymbol{X}^{(l)}\boldsymbol{\Theta}_1^{(l)} + \boldsymbol{X}^{(l)}\boldsymbol{\Theta}_0^{(l)}\right) \tag{13}$$

where $\boldsymbol{X}^{(l)} \in \mathbb{R}^{N\times C}$ are multi-channel graph signals as the input of l-th layer, $\boldsymbol{\Theta}_i^{(l)} \in \mathbb{R}^{D\times C}, \forall i = 0, 1, 2$ are trainable parameters for constant, first-order, and second-order terms in l-th layer, $\sigma(*)$ is an activation function. We also hide the normalization into the parameters.

Although this layer is yielded by Theorem 4, we ignore the influence of non-linear activation functions following [27, 10]. Non-linear activation combined with our theory is open for further research.

4.4. Cross-Channel Analysis

Equation 13 shows the version of our layer in multi-channel cases. It can be derived from the single-channel formula easily. First, we consider for each channel of the output, a set of graph convolutional kernels will run on the corresponding input channel and produce $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_C$. Then they are summed up to fuse cross-channel information. Second, we address the perspective for every input channel. Every input channel will be convolved at D times, with different spectral bands filtered out. By filtering cross-channel output in the next layer, multiple frequency bands can be involved. This implies that as long as our kernels are proved to be effective on single-channel signals, multi-channel filtering can even benefit their performance.

4.5. Comparison with Other Work

The work most related to our approach is [4, 13]. In [4], the authors proposed the approximation methods using Chebyshev polynomials. However, they did not notice in which order of approximation the small localized kernels is optimal. Moreover, we consider empirically effective normalized Laplacian matrices alternatively. Kipf et al. use first-order approximation and share the parameter between the first-order term and the constant term. We point out that these trick is actually harmful for deep GCNs. Although they experimented with second-order and third-order approximation at a very small dataset, they missed the observation of their effectiveness on relatively larger benchmark and their potentials in deep architectures. This work can be regarded as a recalling for the conventional low-order Chebyshev approximate graph kernels.

Compared with [28], we agree with them on raising the importance on the central nodes by introducing another set of parameters for them. But we add that the weighting on central nodes should be varying among channels. Most importantly, our novel layer exploits second-order localization, which is unprecedented in the literature [28, 8]. The isotropic localized filter proposed by us differs from those anisotropic kernels [17, 24, 17] in essence.

5. Experiments

We have conducted multiple experiments on various datasets to evaluate our model. In addition, we compared results of our SoGCN with other state-of-the-art graph convolutional models including GCN [13], GraphSage [8], GIN [28] as well as GAT [24]. We implement our novel graph convolutional layer and SoGCN using PyTorch [19, 20] based on the latest benchmark [6]. Our source code is available on Github[2]. All experiments are run on NVIDIA GeForce RTX 2080 Ti®.

5.1. Datasets

We chose five typical datasets in graph signal processing domain, which have different scale and work on diverse tasks. Relatively small CORA, CITESEER and PUBMED datasets contain citation networks. Each document in the networks has sparse bag-of-words features along with a label. Our task on the three datasets is to predict labels for every document by given minority of them. A medium-scale ZINC molecular graphs dataset targets at regressing a constrained solubility property of a molecule [12]. A relatively large-scale CIFAR10 [14] dataset converted to graph by so called superpixels are used for graph classification task.

5.2. Model Tuning

We polished our proposed model by tuning hyper-parameters including channel dimensions per layer and normalization activation in order to make our SoGCN reach considerable performance on multiple datasets. Other models are evaluated by adopting parameters provided in the benchmark [6].

5.3. Results

We will mark the best models with red and mark good models with blue. Table 1 exhibits our results on CORA dataset while Table 2 shows results on CITESEER and PUBMED datasets. Since the three datasets are rather small datasets, we merely trained graph neural networks with 1 hidden layer on them.

For ZINC dataset, we firstly trained tiny models with ˜100k parameters and then trained larger models with ˜500k parameters (see Table 3).

Experimental results on CIFAR10 are present in Table 4. In this experiment, we additionally recorded the total training time for each model. Moreover, we evaluated models by increasing their depths of networks.

Another experimental results on measuring smoothness are displayed in Table 5. We utilize MAD (Mean Average

[2] https://github.com/yuehaowang/SoGCN

Table 1. Performance on CORA dataset. We can observe our model beats GCN and GraphSage in terms of test accuracy and depends on less parameters but yields a very approaching result compared with GAT.

Model	#Param	#L	Test Acc	Train Acc
GCN	23109	2	0.814	0.993
GAT	92332	2	0.840	0.979
GraphSage	46126	2	0.822	1.000
SoGCN	69235	2	0.839	1.000

Table 2. Performance on CITESEER and PUBMED datasets. Our proposed model fails to yield impressive results on the two datasets. Instead, the most basic GCN model works surpassingly among its improved successors.

Model	Test Acc	
	CITESEER	PUBMED
GCN	0.709	0.796
GAT	0.702	0.774
GraphSage	0.707	0.771
SoGCN	0.686	0.782

Table 3. Performance on ZINC dataset. We adopt Mean Absolute Error as evaluation metric. Our SoGCN outperforms under both 100k parameters and 500k parameters circumstances.

Model	100k		500k	
	#Param	MAE	#Param	MAE
GCN	103077	0.476	505079	0.367
GAT	102385	0.453	531345	0.384
GraphSage	105031	0.450	505341	0.398
GIN	103959	0.474	509549	0.526
SoGCN	144257	0.448	497151	0.364

Distance) metric proposed in [2] for quantifying smoothness of output features. The experiment was conducted on CORA dataset.

6. Conclusion

In this project, we establish a novel graph convolutional layer to achieve linear shift-invariant kernels by cascading such layers deeply. Our graph filters are second-order localized and very efficient using propagation algorithms on graphs. Compared with existing localized isotropic kernels, we prove that assembling our second-order approximate filters are able to reach the upper-bound of such kernels. We also provide convincing experiments with SoGCN based on our proposed layer, and verify that SoGCN can outperform state-of-the-art approaches with anisotropic aggregation.

Our techniques are mostly based on Theorem 4, however, it does not pay sufficient attention on the non-linear activation. Moreover, increasing the depth of our SoGCN cannot significantly benefit the accuracy, we blame this issues to the problematic features on graphs and overfitting.

Theorem 3 implies adjacency matrices can be good shifts in GSP. But if their multiplicity of eigenvalues is high [5], we cannot cover the whole polynomial algebra $\mathcal{A}$ and reach the upper-bound. We leave these problems for future work.

Table 4. Performance on CIFAR10 dataset. By trade-off between training time and test accuracy, SoGCN with 10 hidden layers is more feasible than other models. Most of these models' performance does not climb up by increasing depths of the networks. This problem might stem from excessive trainable parameters brought about by increasing depths, resulting in a poorly learnable models. On the other hand, our proposed model has a promotion in test accuracy when depth raised from 4 to 10, which suggests SoGCN is able to represent more powerful kernels by stacking more second order graph convolutional layers.

Model	#Param	#L	Test Acc	Time (hr)
GCN	101657	5	53.890	3.405
	232181	11	55.510	7.928
	449721	21	55.050	8.679
GAT	110704	5	65.490	8.669
	252976	11	63.910	27.930
	490096	21	63.690	38.305
GraphSage	102907	5	66.040	2.455
	244819	11	65.400	5.370
	481339	21	65.590	8.510
GIN	106534	5	49.330	1.220
	263680	11	46.520	2.797
	525590	21	51.090	3.652
SoGCN	273353	5	65.720	4.582
	661421	11	67.080	8.961
	1308201	21	66.810	14.969

Table 5. Smoothness on CORA dataset. Lower MAD indicates smoother output features. The MAD of input features is 0.681. We can observe that with increasing of layers, output features of each model become smoother gradually. When #L=2, models with severer smoothing problem produce better test accuracy (see Table 1). Nevertheless, over-smoothing models like GraphSage with #L=10 has very poor performance. We can conclude this observation that predicting labels on CORA actually rely a lot on low-frequency features but fail when features are over-smoothing.

Model	MAD / Test Acc			
	#L = 2	#L = 3	#L = 5	#L = 10
GCN	0.085/0.814	0.051/0.787	0.053/0.490	0.176/0.263
GAT	0.049/0.840	0.045/0.840	0.038/0.798	0.044/0.675
GraphSage	0.074/0.822	0.057/0.809	0.064/0.654	0.000/0.309
SoGCN	0.050/0.839	0.041/0.803	0.039/0.662	0.000/0.309

References

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[2] 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 *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3438–3445, 2020.

[3] Bruno Scalzo Dees, Ljubisa Stankovic, Milos Dakovic, Anthony G Constantinides, and Danilo P Mandic. Unitary shift operators on a graph. *arXiv preprint arXiv:1909.05767*, 2019.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in neural information processing systems*, pages 3844–3852, 2016.

[5] Joya A Deri and José MF Moura. Spectral projector-based graph fourier transforms. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):785–795, 2017.

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

[7] Benjamin Girault, Paulo Gonçalves, and Éric Fleury. Translation on graphs: An isometric shift operator. *IEEE Signal Processing Letters*, 22(12):2416–2420, 2015.

[8] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[9] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

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

[11] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[12] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

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

[14] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[15] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[16] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[17] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[18] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[19] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017.

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

[21] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[22] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE Transactions on Signal Processing*, 61(7):1644–1656, 2013.

[23] Manolis Tsakiris. *Lecture Notes on Linear Algebra and Applications*. ShanghaiTech University, 2020.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[25] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2598–2606, 2018.

[26] Guangtao Wang, Rex Ying, Jing Huang, and Jure Leskovec. Improving graph attention networks with large margin-based constraints. *arXiv preprint arXiv:1910.11945*, 2019.

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

[28] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[29] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

A. Proof of Equation 2

Proposition 1. *The solutions $\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_N\}$ to the following optimization objective are the eigenvectors of Laplacian matrix $\boldsymbol{L}$*

$$\boldsymbol{u}_i = \underset{\substack{\boldsymbol{x} \perp \boldsymbol{u}_j, \forall j < i \\ \|\boldsymbol{x}\|_2 = 1}}{\operatorname{argmax}} \boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} \tag{14}$$

Proof: Prove by converting this problem into Courant-Fischer Theorem in [23]. By induction: assume $\boldsymbol{u}_{i-1}$ with $i > 1$ is also the solution to the following objective:

$$\boldsymbol{u}_i = \arg \min_{\dim \mathcal{V} = n-i+1} \max_{\substack{\boldsymbol{x} \in \mathcal{V} \\ \|\boldsymbol{x}\|_2 = 1}} \mathbf{x}^T \boldsymbol{L} \boldsymbol{x} \tag{15}$$

Let $\mathcal{S}_i = \operatorname{span}\{\boldsymbol{u}_1, \cdots, \boldsymbol{u}_{i-1}\} \subset \mathcal{S}_{i-1} \subset \cdots \subset \mathcal{S}_2$, we have $\dim \mathcal{S}_i^{\perp} = n - i + 1$. For every subspace $\mathcal{T} \neq \mathcal{S}_i^{\perp}$ such that $\dim \mathcal{T} = n - i + 1$, there exists $\boldsymbol{y} \in \mathcal{S}_i \cap \mathcal{T}$ such that $\|\boldsymbol{y}\|_2 = 1$.

Let $\boldsymbol{y} = \sum_{j \in [i-1]} \alpha_j \boldsymbol{u}_j$ and $\|\boldsymbol{y}\|_2^2 = \sum_{j \in [i-1]} \alpha_j^2 = 1$.

$$\max_{\substack{\boldsymbol{x} \in \mathcal{T} \\ \|\boldsymbol{x}\|_2 = 1}} \boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} \geq \mathbf{y}^T \boldsymbol{L} \boldsymbol{y} = \sum_{j=1}^{i-1} \lambda_j \alpha_j^2 \tag{16}$$

$$\geq \sum_{j=1}^{i-1} \lambda_{i-1} \alpha_j^2 = \lambda_{i-1} \tag{17}$$

$$= \max_{\boldsymbol{x} \in \mathcal{S}_{i-1}^{\perp}} \boldsymbol{x}^T \boldsymbol{L} \boldsymbol{x} \geq \boldsymbol{u}_i^T \boldsymbol{L} \boldsymbol{u}_i \tag{18}$$

Therefore, the min-max search will fall into $\mathcal{S}_i^{\perp}$.

$$\boldsymbol{u}_i = \arg \min_{\dim \mathcal{V} = n-i+1} \max_{\substack{\boldsymbol{x} \in \mathcal{V} \\ \|\boldsymbol{x}\|_2 = 1}} \mathbf{x}^T \boldsymbol{L} \boldsymbol{x} \tag{19}$$

Since we have $\boldsymbol{u}_1 = \operatorname{argmax}_{\|\boldsymbol{x}\|_2 = 1} \mathbf{x}^T \boldsymbol{L} \boldsymbol{x}$, we can conclude the proof. □

B. Proof of Theorem 1

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

Theorem 1. *Suppose the characteristic and minimal polynomials of shift $\boldsymbol{S}$ are equal: $m_{\mathbf{S}}(x) = p_{\mathbf{S}}(x)$. Then a linear transformation $\boldsymbol{H}$ is linear shift-invariant if and only if $\boldsymbol{H}$ is a polynomial in the graph shift $\boldsymbol{S}$.*

Proof: Notice that $\boldsymbol{H}$ is shift-invariant only if $\boldsymbol{H}\boldsymbol{S} = \boldsymbol{S}\boldsymbol{H}$. By Lemma 1, there exists $\boldsymbol{U}$ such that $\boldsymbol{H} = \boldsymbol{U}^T \boldsymbol{M} \boldsymbol{U}$ and $\boldsymbol{S} = \boldsymbol{U}^T \boldsymbol{\Lambda} \boldsymbol{U}$, where $\boldsymbol{M} = \operatorname{diag}(M_1, \cdots, M_N)$ and $\boldsymbol{\Lambda} = \operatorname{diag}(\Lambda_1, \cdots, \Lambda_N)$ are diagonal.

There exists a polynomial $h(x) = \sum_{k=0}^{N-1} \alpha_k x^k$ such that $\boldsymbol{H} = h(\boldsymbol{S}) = \boldsymbol{U}^T h(\boldsymbol{\Lambda}) \boldsymbol{U}$ if the following linear equation is well-conditioned.

$$\alpha_0 \boldsymbol{I} + \alpha_1 \boldsymbol{\Lambda} + \cdots + \alpha_{N-1} \boldsymbol{\Lambda}^{N-1} = \boldsymbol{M} \tag{20}$$

$$\begin{bmatrix} 1 & \Lambda_1 & \Lambda_1^2 & \cdots & \Lambda_1^{N-1} \\ 1 & \Lambda_2 & \Lambda_2^2 & \cdots & \Lambda_2^{N-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \Lambda_N & \Lambda_N^2 & \cdots & \Lambda_N^{N-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{bmatrix} = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_N \end{bmatrix} \tag{21}$$

where the left-side square matrix is called Vandermonde matrix $\boldsymbol{V}$. If $p_{\boldsymbol{S}}(x) = m_{\boldsymbol{S}}(x)$, $\boldsymbol{S}$ has distinct eigenvalues, which means $\det(\boldsymbol{V}) = \prod_{1 \leq j < j \leq N} (\Lambda_j - \Lambda_i) \neq 0$.

Conversely, if $\boldsymbol{H} = h(\boldsymbol{S}) = \boldsymbol{U}^T h(\boldsymbol{\Lambda}) \boldsymbol{U}$, by Lemma 1, $\boldsymbol{H}$ and $\boldsymbol{S}$ commute. □

C. Proof of Theorem 2

Theorem 2. *With the same assumption of Theorem 1, for any filter $\boldsymbol{H} \in \mathcal{H}$, there exists polynomial $h(x)$ such that $\boldsymbol{H} = h(\boldsymbol{S})$ with $\deg h(x) \leq N - 1$.*

Proof: According to Theorem 1, let $\boldsymbol{H} = h(\boldsymbol{S})$. By polynomial division, there exists unique polynomials $q(x)$ and $r(x)$ such that

$$h(x) = q(x) m_{\boldsymbol{S}}(x) + r(x) \tag{22}$$

where $\deg r(x) < \deg m_{\boldsymbol{S}}(x)$. Insert $\boldsymbol{S}$ into Equation 22:

$$h(\boldsymbol{S}) = q(\boldsymbol{S}) m_{\boldsymbol{S}}(\boldsymbol{S}) + r(\boldsymbol{S}) = q(\boldsymbol{S}) \mathbf{0} + r(\boldsymbol{S}) \tag{23}$$

Thus, $h(\boldsymbol{A}) = r(\boldsymbol{A})$ and $\deg h(x) < \deg m_{\boldsymbol{A}}(x)$.

Notice that $\deg m_{\boldsymbol{S}}(x) = \deg p_{\boldsymbol{S}}(x) = N$. Hence, $\deg h(x) \leq N - 1$. □

D. Proof of Theorem 3

Theorem 3. *For any matrix $\boldsymbol{A}$, there exists a latent shift $\boldsymbol{S}$ and polynomial $g(x)$ such that $\boldsymbol{A} = g(\boldsymbol{S})$ and $m_{\mathbf{S}}(x) = p_{\mathbf{S}}(x)$. Meanwhile, $\boldsymbol{A}$ and $\boldsymbol{S}$ share the same eigenspace.*

Proof: Since we only requires that $m_{\mathbf{S}}(x) = p_{\mathbf{S}}(x)$, we can let $\boldsymbol{S}$ and $\boldsymbol{A}$ be simultaneously diagonalized by orthonormal matrix $\boldsymbol{U}$ comprises of eigenvectors from $\boldsymbol{A}$. The difference is that $\boldsymbol{S} = \boldsymbol{U}^T \boldsymbol{\Lambda} \boldsymbol{U}$ has the diagonal matrix $\boldsymbol{\Lambda}$ containing distinct eigenvalues. Now $\boldsymbol{A}$ can be regarded as a shift-invariant filter associated with shift $\boldsymbol{S}$. By Theorem 1, there exists $g(x)$ such that $\boldsymbol{A} = g(\boldsymbol{S})$. □

E. Proof of Theorem 4

Lemma 2. *Over the field of reals, the degree of an irreducible non-trivial univariate polynomial is either one or two.*

Theorem 4. *Any filter $\boldsymbol{H}$ can be decomposed to the product of second-order approximate filter ($K = 2$). Formally, there exists $\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta}_L$ such that $\boldsymbol{H} = \prod_{l=1}^{L} \boldsymbol{H}_2(\boldsymbol{\theta}_l)$, where $\boldsymbol{\theta}_l \in \mathbb{R}^3$ and $L \leq N - 1$.*

Proof: Let us write $\boldsymbol{H}$ in terms of shift $\boldsymbol{A}$ (the same with $\tilde{\boldsymbol{A}}$) as $\boldsymbol{H} = h(\boldsymbol{A})$ with $\deg h(x) \leq N - 1$, by Theorem 1 and 2.

By Lemma 2, if $h(x)$ can be decomposed into series of polynomials with the order up to two. And the count of these factor polynomials are up to $\deg h(x)$.

$$h(x) = \prod_{i=1}^{\deg h(x)} (u_i x^2 + v_i x + c_i) \tag{24}$$

Therefore, we apply this property to our polynomial algebra. Let $\boldsymbol{\theta}_l = \begin{bmatrix} u_l & v_l & c_l \end{bmatrix}^T \in \mathbb{R}^3$, we have $\boldsymbol{H} = \prod_{l=1}^{L} u_l \boldsymbol{A}^2 + v_l \boldsymbol{A} + c_l \boldsymbol{I} = \prod_{l=1}^{L} \boldsymbol{H}_2(\boldsymbol{\theta}_l)$, where $L \leq N - 1$. □