Towards Better Graph Representation Learning with Parameterized Decomposition & Filtering

Mingqi Yang [1] **Wenjie Feng** [2] **Yanming Shen** [1] **Bryan Hooi** [2 3]

Abstract

Proposing an effective and flexible matrix to represent a graph is a fundamental challenge that has been explored from multiple perspectives, e.g., filtering in Graph Fourier Transforms. In this work, we develop a novel and general framework which unifies many existing GNN models from the view of parameterized decomposition and filtering, and show how it helps to enhance the flexibility of GNNs while alleviating the smoothness and amplification issues of existing models. Essentially, we show that the extensively studied spectral graph convolutions with learnable polynomial filters are constrained variants of this formulation, and releasing these constraints enables our model to express the desired decomposition and filtering simultaneously. Based on this generalized framework, we develop models that are simple in implementation but achieve significant improvements and computational efficiency on a variety of graph learning tasks. Code is available at https://github.com/qslim/PDF.

1. Introduction

Graph Neural Networks (GNNs) have emerged as a powerful and promising technique for representation learning on graphs and have been widely applied to various applications. A large number of GNN models have been proposed, including spectral graph convolutions, spatial message-passing, and even Graph Transformers, for boosting performance or resolving existing defects, e.g., the oversmoothing issue (Li et al., 2018; Oono & Suzuki, 2020; Huang et al., 2020) or expressive power (Xu et al., 2019b; Morris et al., 2019; Sato, 2020). This raises the natural question of: *how are these dif-*

[1]Dalian University of Technology, China [2]Institute of Data Science, National University of Singapore, Singapore [3]School of Computing, National University of Singapore, Singapore. Correspondence to: Yanming Shen <shen@dlut.edu.cn>.

ferent models related to one another? Consequently, when conducting learning tasks on a graph with these models, the fundamental questions we aim to address are: how to *construct an effective adaptive matrix representation* for the graph topology, and how to *flexibly capture the interactions between multichannel graph signals*.

Under the paradigm of spectral graph convolution theories, the Laplacian and its variants are used as the matrix representation of the graph to maintain theoretical consistency (Chung, 1997; Hammond et al., 2011; Shuman et al., 2013; Defferrard et al., 2016). This induces spectral GNNs, which have become a popular class of GNNs with performance guarantees, including GCN (Kipf & Welling, 2017), SGC (Wu et al., 2019a), S^2GC (Zhu & Koniusz, 2020), and others (Klicpera et al., 2019a;b; Ming Chen et al., 2020). All of them utilize the eigenvectors of the (normalized) Laplacian for Graph Fourier Transform and design different graph signal filters in the frequency domain.

Beyond spectral models, spatial GNNs have gradually become prominent in the research community, which consider graph convolution within the message-passing framework (Gilmer et al., 2017; Xu et al., 2019b; Corso et al., 2020; Yang et al., 2020), and the advantage of flexible implementation makes them favorable for the graph-level prediction task. Interestingly, the aggregation design, which plays a central role in the spatial GNNs, also implicitly corresponds to a specific matrix representation for the graph.

Recently, transformers have shown promising performance on molecular property predictions (Ying et al., 2021; Kreuzer et al., 2021; Bastos et al., 2022; Min et al., 2022; Rampášek et al., 2022). These models apply transformers with positional encoding, structural encoding, and other techniques to graph data, while they actually also refer to specific matrix representations. Besides, several studies in the field of graph signal processing also show that the matrix representation can be flexible as long as it reflects the topology of the underlying graph (Dong et al., 2016; Deri & Moura, 2017; Ortega et al., 2018). Graph Shift Operator (GSO) proposes a group of feasible matrix representations (Sandryhaila & Moura, 2013), and allows GNNs to obtain better performance for graph learning tasks by learning matrix representations from a group of GSOs (Da-

soulas et al., 2021).

Accordingly, designing a flexible and suitable matrix representation for the underlying graph plays an intrinsic role in improving GNNs on various tasks. Although many specific choices exist, each of them has its own limitations and no single matrix representation is suitable for every task (Butler & Chung, 2017). Therefore, it is highly important to systematically explore the differences among these different models and automatically find the most suitable matrix representation, while capturing the complex interaction of multichannel signals for a graph learning task.

In this paper, we show how various graph matrix representations applied to graph learning can be interpreted as different (Decomposition, Filtering) operations over input signals assigned to the graph. Then, based on this understanding, we propose to learn (Decomposition, Filtering) as a whole, fundamentally different from existing spectral graph convolutions with learnable filters. Correspondingly, the objective is extended from learning a suitable filter to learning a suitable graph matrix representation, i.e. (Decomposition, Filtering).

To achieve this, we propose a novel Parameterized-$(\mathcal{D}, \mathcal{F})$ framework which aims to learn a suitable (Decomposition, Filtering) for input signals. It inspires the use of more expressive learnable mappings on graph topology to enlarge the learning space of graph matrix representations. Also, Parameterized-$(\mathcal{D}, \mathcal{F})$ serves as a general framework that unifies existing GNNs ranging from the original spectral graph convolutions to the latest Graph Transformers. For example, spectral graph convolution corresponds to fixed $\mathcal{D}$ and parameterized $\mathcal{F}$ via the lens of Parameterized-$(\mathcal{D}, \mathcal{F})$, and recent Graph Transformers, which improve performance by leveraging positional/structural encodings, potentially result in more effective $(\mathcal{D}, \mathcal{F})$ on input signals. Parameterized-$(\mathcal{D}, \mathcal{F})$ also inspires new insight into the widely-studied smoothing and amplification issues in GNNs, which serves as the motivation of our proposed solution as well. Our main contributions include,

1. We present Parameterized-$(\mathcal{D}, \mathcal{F})$, which addresses the deficiencies of learning filters alone, and also reveals the connections between various existing GNNs;
2. With Parameterized-$(\mathcal{D}, \mathcal{F})$ framework, we develop a model with a simple implementation that achieves superior performance while preserving computational efficiency.

2. Preliminaries

Consider an undirected graph $G = (\mathcal{V}, \mathcal{E})$ with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the adjacency matrix ($\mathbf{W} \in \mathbb{R}^{n \times n}$ if G is weighted) with corresponding degree matrix $\mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1}_n)$, $\mathbf{L} = \mathbf{D} - \mathbf{A}$ be the Laplacian,

$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ and $\tilde{\mathbf{D}} = \mathbf{D} + \mathbf{I}$. Let $\boldsymbol{f} \in \mathbb{R}^n$ be the single-channel graph signal assigned on G, and $\mathbf{H} \in \mathbb{R}^{n \times d}$ be the d-channel graph signal or node feature matrix with d dimensions. We use $[K]$ to denote the set $\{0, 1, 2, \ldots, K\}$.

Graph Representation. A graph topology can be expressed with various matrix representations, e.g., (normalized) adjacency, Laplacian, GSO (Sandryhaila & Moura, 2013), etc. (Dong et al., 2016; Deri & Moura, 2017; Ortega et al., 2018) show that the representation matrix used in graph signal processing can be flexible as long as it reflects the graph topology, and different representations lead to different signal models. In recent GNN studies, the involved graph representations are even more flexible (Ying et al., 2021). Since they are all induced from the same graph topology, we use the graph representation space $\mathcal{M}_G$ to denote the set of all possible representations for a graph G. A summary of existing strategies for building a graph representation $\mathbf{S} \in \mathcal{M}_G$ is provided in Appendix A. In this work, we only consider undirected graphs with $\mathcal{M}_G$ only involving symmetric matrices.

3. Parameterized-$(\mathcal{D}, \mathcal{F})$

3.1. Generalizing Spectral Graph Convolution

Given a graph G with the Laplacian $\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$, $\boldsymbol{\Lambda} = \mathrm{diag}(\boldsymbol{\lambda})$, a filter $\boldsymbol{t}$ and a graph signal $\boldsymbol{f}$ assigned on G, the spectral graph convolution of $\boldsymbol{f}$ with $\boldsymbol{t}$ on G leveraging convolution theorem (Hammond et al., 2011; Defferrard et al., 2016) is defined as

$$
\begin{aligned}
\boldsymbol{f}' = \boldsymbol{t} *_{\mathbf{L}} \boldsymbol{f} &= \mathbf{U}((\mathbf{U}^\top \boldsymbol{t}) \odot (\mathbf{U}^\top \boldsymbol{f})) \\
&\approx \mathbf{U}(g_{\boldsymbol{\theta}}(\boldsymbol{\lambda}) \odot (\mathbf{U}^\top \boldsymbol{f})) = \mathbf{U}(g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})(\mathbf{U}^\top \boldsymbol{f})) \quad (1) \\
&= g_{\boldsymbol{\theta}}(\mathbf{L})\boldsymbol{f},
\end{aligned}
$$

where $\hat{\boldsymbol{f}} = \mathbf{U}^\top \boldsymbol{f}$ is the Graph Fourier Transform, and $\boldsymbol{f} = \mathbf{U}\hat{\boldsymbol{f}}$ is its inverse transform over the graph domain, $g_{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ is the polynomial over $\boldsymbol{\lambda}$ and is used to approximate the transformed filter $\hat{\boldsymbol{t}} = \mathbf{U}^\top \boldsymbol{t}$. Although various graph convolutions have been proposed, they are all under the formulation in Eq. 1, and most of them enhance $g_{\boldsymbol{\theta}}(\boldsymbol{\lambda})$ with sophisticated designed polynomials with learnable coefficients.

As $\mathbf{U}$ is column orthogonal and $g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})$ is diagonal, the convolution $\boldsymbol{t} *_{\mathbf{L}} \boldsymbol{f} = \mathbf{U}(g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})(\mathbf{U}^\top \boldsymbol{f}))$ in Eq. 1 can be divided into three individual steps:

1) $\hat{\boldsymbol{f}} = \mathbf{U}^\top \boldsymbol{f}$: *decomposition* (or *transformation*);
2) $\hat{\boldsymbol{f}}' = g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})\hat{\boldsymbol{f}}$: *filtering* (or *scaling*) with $g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})$;
3) $\boldsymbol{f}' = \mathbf{U}\hat{\boldsymbol{f}}'$: *reverse transformation*.

Consequently, we can interpret the convolution as above that filter $\boldsymbol{f}$ with $g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})$ under the decomposition $\mathbf{U}^\top$. Without loss of generality, we use $\mathcal{D}$ to denote the involved

decomposition and its reversion, $\mathcal{F}$ to denote the involved filtering, and then the above operation is represented as $(\mathcal{D}, \mathcal{F})_{g_{\boldsymbol{\theta}}(\mathbf{L})}(\boldsymbol{f})$ which indicates the applied $(\mathcal{D}, \mathcal{F})$ on $\boldsymbol{f}$ is provided by $g_{\boldsymbol{\theta}}(\mathbf{L})$. The options of $(\mathcal{D}, \mathcal{F})_{g_{\boldsymbol{\theta}}(\mathbf{L})}$ in general polynomial graph filters are only restricted within $\{g_{\boldsymbol{\theta}}(\mathbf{L}) | \boldsymbol{\theta} \in \mathbb{R}^k\} \subset \mathcal{M}_G$ where k is the order of the polynomial and $\boldsymbol{\theta}$ is the polynomial coefficients. Note that any $\mathbf{S} \in \mathcal{M}_G$ is a symmetric matrix with the unique eigendecomposition $\mathbf{S} = \mathbf{U}'\hat{\boldsymbol{\Lambda}}'\mathbf{U}'^{\top}$, where $\mathbf{U}'$ and $\hat{\boldsymbol{\Lambda}}'$ can serve as a feasible $(\mathcal{D}, \mathcal{F})$. It is desirable to extend $(\mathcal{D}, \mathcal{F})$ beyond $\{g_{\boldsymbol{\theta}}(\mathbf{L}) | \boldsymbol{\theta} \in \mathbb{R}^k\}$. Hence we have for any $\mathbf{S} \in \mathcal{M}_G$,

$$\mathbf{S}\boldsymbol{f} = \mathbf{U}'\hat{\boldsymbol{\Lambda}}'(\mathbf{U}'^{\top}\boldsymbol{f}) = (\mathcal{D}, \mathcal{F})_{\mathbf{S}}(\boldsymbol{f}).$$

Still, $\mathbf{S}\boldsymbol{f}$ refers to filtering $\boldsymbol{f}$ with $\hat{\boldsymbol{\Lambda}}'$ under the decomposition $\mathbf{U}'^{\top}$, where $\mathbf{U}'$ is column orthogonal referring to a *rotation* of $\mathbf{U}$. Consequently, applying different $\mathbf{S} \in \mathcal{M}_G$ all correspond to filtering $\boldsymbol{f}$ with related filters under different decomposition $\mathbf{U}^{\top}$ [1]. To build a generalized point of view, we generalize the spectral graph convolution in Eq. 1 as follows.

Definition 3.1 (Parameterized-$(\mathcal{D}, \mathcal{F})$)**.** Given a graph G, and a signal $\boldsymbol{f}$ assigned on G, the Parameterized-$(\mathcal{D}, \mathcal{F})$ of $\boldsymbol{f}$ over G is defined as

$$\boldsymbol{f}' = f_{\boldsymbol{\theta}}(\mathcal{G})\boldsymbol{f} \tag{2}$$

where $\mathcal{G} \subset \mathcal{M}_G$ and $f_{\boldsymbol{\theta}} : \{\mathbb{R}^{n \times n}\} \mapsto \mathbb{R}^{n \times n}$ is the mapping of $\mathcal{G}$ with the parameter $\boldsymbol{\theta}$. $f_{\boldsymbol{\theta}}$ should satisfy the conditions,

- *Closeness*: $\forall \mathcal{G} \subset \mathcal{M}_G$, $f_{\boldsymbol{\theta}}(\mathcal{G}) \in \mathcal{M}_G$, and
- *Permutation-equivariance:* For any permutation matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$, $f_{\boldsymbol{\theta}}(\{\mathbf{M}^{\top}\mathbf{S}\mathbf{M} \,|\, \mathbf{S} \in \mathcal{G}\}) = \mathbf{M}^{\top}f_{\boldsymbol{\theta}}(\mathcal{G})\mathbf{M}$.

Given $\mathcal{G}$, $f_{\boldsymbol{\theta}}(\mathcal{G})$ is parameterized by $\boldsymbol{\theta}$, and the resulting space w.r.t. $\boldsymbol{\theta}$ is $\{f_{\boldsymbol{\theta}}(\mathcal{G}) | \boldsymbol{\theta}\} \subset \mathcal{M}_G$. Any $\mathbf{S} \in \{f_{\boldsymbol{\theta}}(\mathcal{G}) | \boldsymbol{\theta}\}$ corresponds to a specific $(\mathcal{D}, \mathcal{F})_{\mathbf{S}}$. $\{g_{\boldsymbol{\theta}}(\mathbf{L}) | \boldsymbol{\theta} \in \mathbb{R}^k\}$ in Eq.1 based on the theory of spectral graph convolution refers to a constrained Parameterized-$(\mathcal{D}, \mathcal{F})$: (i) $\mathcal{G} = \{\mathbf{L}\}$ as required in Graph Fourier Transform, and (ii) $f_{\boldsymbol{\theta}} = g_{\boldsymbol{\theta}}$ is constrained to be polynomials such that it is equivariant to the decomposition $\mathbf{U}$, i.e., $g_{\boldsymbol{\theta}}(\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\top}) = \mathbf{U}g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})\mathbf{U}^{\top}$. [2] This makes all $\mathbf{S} \in \{g_{\boldsymbol{\theta}}(\mathbf{L}) | \boldsymbol{\theta}\}$ share the same $\mathcal{D}$. In other words, applying learnable $\boldsymbol{\theta}$ will only learn $\mathcal{F}$ but fix $\mathcal{D}$.

By removing such constraints, different $\boldsymbol{\theta}$ results in different $\mathcal{D}$ and $\mathcal{F}$. Therefore, applying learnable $\boldsymbol{\theta}$ makes it capable of learning $(\mathcal{D}, \mathcal{F})$ as a whole. We will systematically investigate the effectiveness of learnable $\mathcal{D}$ by first revisiting existing GNNs via the lens of Parameterized-$(\mathcal{D}, \mathcal{F})$ in Sec. 3.2, and then developing our models leveraging the

---

[1]Furthermore, $\mathbf{S}\boldsymbol{f} = \mathbf{U}'\hat{\boldsymbol{\Lambda}}'(\mathbf{U}'^{\top}\boldsymbol{f}) = \mathbf{U}'(\hat{\boldsymbol{\lambda}}' \odot (\mathbf{U}'^{\top}\boldsymbol{f})) = \mathbf{U}'((\mathbf{U}'^{\top}\boldsymbol{\lambda}') \odot (\mathbf{U}'^{\top}\boldsymbol{f})) = \boldsymbol{\lambda}' *_{\mathbf{S}} \boldsymbol{f}$, which is consistent with the definition of spectral graph convolution.

[2]Or alternatively, $\mathcal{G} = \{\mathbf{L}^k | k \in [K]\}$, and $f_{\boldsymbol{\theta}} = \sum_{k \in [K]} \boldsymbol{\theta}_k \cdot$.
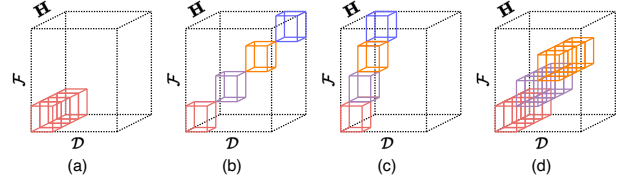

Figure 1. A taxonomy of existing GNN architectures via the lens of $(\mathcal{D}, \mathcal{F})$ for multichannel signal $\mathbf{H}$. Within each of them, the colored box corresponds to a channel processed with an individual $(\mathcal{D}, \mathcal{F})$. We use the same color to denote the same $(\mathcal{D}, \mathcal{F})$. Each entry in the $\mathcal{D}$-$\mathcal{F}$ plane corresponds to a specific $(\mathcal{D}, \mathcal{F}) \in \mathcal{M}_G$. As eigendecomposition is unique, these $(\mathcal{D}, \mathcal{F})$ are different.

learnable $\mathcal{D}$ in Sec. 3.3. In Sec. 4, we will conduct analysis on the limitations of learning $\mathcal{F}$ alone and the necessity of learning $(\mathcal{D}, \mathcal{F})$ as a whole.

3.2. Unifying Existing GNNs with Parameterized-$(\mathcal{D}, \mathcal{F})$

By assigning constraints on $f_{\boldsymbol{\theta}}$ and $\mathcal{G}$ in Eq. 2, we can achieve (fixed $\mathcal{D}$, fixed $\mathcal{F}$), (fixed $\mathcal{D}$, learnable $\mathcal{F}$) or (learnable $\mathcal{D}$, learnable $\mathcal{F}$) respectively, which provides a unification of various GNNs. First, we summarize all possible architectures in multichannel signal scenario as in Fig. 1.

(a) refers that all channels share the same $(\mathcal{D}, \mathcal{F})$;
(b) assigns each channel with an independent $(\mathcal{D}, \mathcal{F})$;
(c) applies independent $\mathcal{F}$ under a shared $\mathcal{D}$ in different channels;
(d) assigns each signal with multiple $(\mathcal{D}, \mathcal{F})$.

Then, by integrating learnable $\mathcal{D}$ or $\mathcal{F}$ into each of these architectures, we can classify all existing models as in Tab. 1. Note that (b) has not been well investigated yet, and to the best of our knowledge, only Correlation-free (Yang et al., 2022a) implicitly results in this architecture due to nonlinear computations in their code implementation, but there is no reflection of this architecture in their paper. (a) acts as the most common form under which many studies correspond to designing sophisticated polynomials to parameterize $\mathcal{F}$, e.g., ChebyNet/CayleyNet/BernNet. JacobiConv differs from them in that it learns $\mathcal{F}$ for each channel independently, therefore it corresponds to (c). Compared with (b), the channel-wise learnable $\mathcal{F}$ in (c) is still under the shared $\mathcal{D}$. (d) assigns each channel with multiple $(\mathcal{D}, \mathcal{F})$. It is generalized from multi-head attention, e.g., Multi-head GAT/Transformer, or multi-aggregator, e.g., PNA/ExpC, where each *head* in multi-head attention, or each *aggregator* in multi-aggregator corresponds to an individual $(\mathcal{D}, \mathcal{F})$. Their effectiveness can be interpreted by filtering task-relevant patterns from multiple $\mathcal{D}$ for each input channel. Some spatial methods with learnable aggregation coefficients implicitly result in learnable $\mathcal{D}$, e.g., GAT/ExpC/PNA, as the resulting $\mathbf{S} \in \mathcal{M}_G$ with different aggregation coefficients generally do not share $\mathcal{D}$.

Table 1. A summary of differences of popular GNNs via the lens of $(\mathcal{D}, \mathcal{F})$. TF denotes Transformer and MGAT denotes multi-head GAT.

	GCN/SGC	BernNet/GPR	JacobiConv	Corr-free	PGSO/GAT	GIN-0	TF/MGAT/ExpC	PNA
Category	Spectral	Spectral	Spectral	?	?	Spatial	Spatial	Spatial
Learnable $\mathcal{F}$	No	Yes	Yes	Yes	Yes	No	Yes	No
Learnable $\mathcal{D}$	No	No	No	Yes	Yes	No	Yes	No
Architecture	(a)	(a)	(c)	(b)*	(a)	(a)	(d)	(d)

Eq. 2 also inspires a new perspective regarding the difference between spectral and spatial methods [3]: if $\mathcal{D}$ is fixed to be the eigenspace of Laplacian, they belong to spectral models, and can be implemented with fixed/learnable or channel-shared/independent $\mathcal{F}$ strategies. Otherwise, they belong to non-spectral or spatial models.

3.3. Developing an Effective Parameterized-$(\mathcal{D}, \mathcal{F})$

We have shown that some competitive graph-level prediction models implicitly include learnable $\mathcal{D}$. But they are introduced as a side effect of various attention, neighborhood aggregation, and transformer designs. And their learning space (the set of all $\mathcal{D}$ that can be learned from the given graph) varies from each other due to different implementations. We believe that the learnable $(\mathcal{D}, \mathcal{F})$ as a whole can potentially contribute to the final performance improvements. To fully leverage learnable $\mathcal{D}$ as well as $\mathcal{F}$ for a graph G, we develop Parameterized-$(\mathcal{D}, \mathcal{F})$ with the objective of learning $(\mathcal{D}, \mathcal{F})$ from a larger subspace of $\mathcal{M}_G$ for input graph signals.

Note that $f_{\boldsymbol{\theta}}(\mathcal{G})$ involves two components: $\mathcal{G}$ and $f_{\boldsymbol{\theta}}$. Building $\mathcal{G}$ can be flexible. For example, in some attention-based methods, e.g., GAT and Graph Transformers, $\mathcal{G}$ is implemented as an attention weight matrix that takes advantage of both graph topology and node features. In PGSO, $\mathcal{G}$ is a group of GSOs. Here, we build $\mathcal{G}$ as $\mathcal{G} = \{(\tilde{\mathbf{D}}^\epsilon \tilde{\mathbf{A}} \tilde{\mathbf{D}}^\epsilon)^k | -0.5 \leq \epsilon \leq 0, k \in [K]\}$ in order that $\tilde{\mathbf{D}}^\epsilon \tilde{\mathbf{A}} \tilde{\mathbf{D}}^\epsilon$ with different ϵ do not share eigenspace. There can be more sophisticated designs of $\mathcal{G}$, and we leave them for future work. Then, we implement $f_{\boldsymbol{\theta}}$ as a multi-layer perceptron (MLP), which satisfies the two conditions in Definition 3.1. For the sake of brevity, we only present one layer perceptron here without loss of generality. Correspondingly, the convolution on a single-channel signal $\boldsymbol{f}$ is

$$\boldsymbol{f}' = f_{\boldsymbol{\theta}}(\mathcal{G})\boldsymbol{f} = \sigma(\sum_{\mathbf{S}_i \in \mathcal{G}} \boldsymbol{\theta}_i \mathbf{S}_i)\boldsymbol{f}, \qquad (3)$$

where $\boldsymbol{\theta} \in \mathbb{R}^{|\mathcal{G}|}$ is the learnable coefficient and σ is a nonlinear function in the MLP. Given G and its associated $\mathcal{G}$, the learning space of $(\mathcal{D}, \mathcal{F})$ w.r.t. $\boldsymbol{\theta}$ is $\{\sigma(\sum_{\mathbf{S}_i \in \mathcal{G}} \boldsymbol{\theta}_i \mathbf{S}_i) | \boldsymbol{\theta}\}$.

---

[3] There are no formal definitions of spectral and spatial-based methods. Generally, the ones introduced from spectral graph convolution as in Eq. 1 are considered as spectral-based methods, and others such as message-passing, Graph Transformers are considered as spatial ones.

Applying more expressive $f_{\boldsymbol{\theta}}$ allows learning $(\mathcal{D}, \mathcal{F})$ from a larger subspace in $\mathcal{M}_G$. In Sec. 5.1, we conduct extensive experiments to evaluate the effects of different $f_{\boldsymbol{\theta}}$ and $\mathcal{G}$.

To extend Eq. 3 to multichannel signal scenario, we design Channel-shared Parameterized-$(\mathcal{D}, \mathcal{F})$ (denoted by "shd-PDF") and Channel-independent Parameterized-$(\mathcal{D}, \mathcal{F})$ (denoted by "idp-PDF") strategies with each of them divided into three steps, differing at Step 2.

1) <u>Pre-transform</u>: $\mathbf{Z} = \sigma(\mathbf{H}\mathbf{W}_1)$
2) Multichannel Parameterized-$(\mathcal{D}, \mathcal{F})$:
   - <u>shd-PDF</u>: $\mathbf{Z}' = \sigma(\sum_{\mathbf{S}_i \in \mathcal{G}} \boldsymbol{\theta}_i \mathbf{S}_i)\mathbf{Z}$
   - <u>idp-PDF</u>: $\mathbf{Z}'_{:j} = \sigma(\sum_{\mathbf{S}_i \in \mathcal{G}} \boldsymbol{\Theta}_{ij} \mathbf{S}_i)\mathbf{Z}_{:j}$
3) <u>Post-transform</u>: $\mathbf{H}' = \sigma(\mathbf{Z}'\mathbf{W}_2)$

Correspondingly, shd-PDF belongs to Fig. 1(a), and idp-PDF belongs to Fig. 1(b) which has not been well investigated by existing studies. $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{d \times d}$ are learnable transformation matrices. In shd-PDF, the channel-shared $(\mathcal{D}, \mathcal{F})$ is parameterized by the learnable $\boldsymbol{\theta} \in \mathbb{R}^{|\mathcal{G}|}$, and in idp-PDF, the channel-independent $(\mathcal{D}, \mathcal{F})$ is parameterized by the learnable $\boldsymbol{\Theta} \in \mathbb{R}^{|\mathcal{G}| \times d}$.

Compared with the latest best-performing methods, shd-PDF and idp-PDF benefit from their simple implementations and computational efficiency. The time and space complexity and the running time are provided in Appendix C.5. In our experiments, the performance improvements under these simple implementations validate the effectiveness of Parameterized-$(\mathcal{D}, \mathcal{F})$.

Scalability. As graph matrices are generally stored in a sparse manner, which only stores non-zero entries to save memory space, the proportion of non-zero entries determines the amount of memory usage. Especially for large graphs, we need to leverage sparse matrix representations, which is a subset of graph matrix representations, to make it tractable under the limited memory resources. There are many ways to express larger scope substructure while preserving the matrix sparsity such as encoding local structures to graph matrix entries. They can all be viewed as a sparse subset of $\mathcal{M}_G$, which is denoted as $\mathcal{M}_G^{\text{sps}}$.

Why is it necessary to learn $\mathcal{D}$ from the input signals? We provide an intuitive example in Fig. 2 here to illustrate the considerations. Suppose all signal channels form two clusters and the downstream task requires identifying these two clusters, then $\mathcal{D}$ in Fig. 2(a) serves as a more suitable

one compared with that in Fig. 2(b), as the projections on that basis better distinguish the two clusters. Therefore, the characteristics/distributions of input signals should be taken into consideration when choosing $\mathcal{D}$, which serves as an empirical understanding of learnable $\mathcal{D}$. This idea is well-adopted in dimension reduction techniques, e.g., PCA (Abdi & Williams, 2010), LDA (Xanthopoulos et al., 2013), etc. The theoretical motivations behind this are given in Sec. 4.
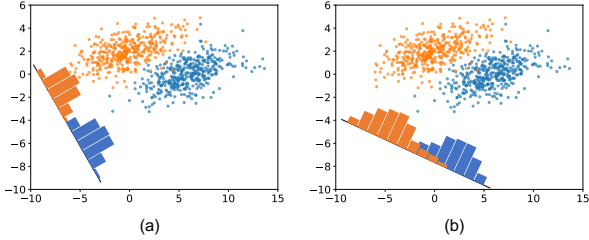

Figure 2. The effects of basis choices when considering within a bunch of signal channels.

4. Motivations on Learning $(\mathcal{D}, \mathcal{F})$ as a Whole

A multi-channel graph signal $\mathbf{H} \in \mathbb{R}^{n \times d}$ involves d channels. Along with the iterative graph convolution operations, the smoothness of each channel dynamically change w.r.t. model depth, and oversmoothed signals account for the performance drop of deep models. In this section, we study the smoothness of signals from two complementary perspectives, i.e., the smoothness within a single channel and that over different channels, showing how Parameterized-$(\mathcal{D}, \mathcal{F})$ as a whole affects both of them.

4.1. Smoothness within a Single Channel

For a weighted undirected graph G, its Laplacian is defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{W}$ is the weighted adjacency matrix with $\mathbf{W}_{ij} > 0$ if vertices i and j are adjacent, and $\mathbf{W}_{ij} = 0$ otherwise. $\mathbf{D} = \mathrm{diag}(\mathbf{W1}_n)$ is the diagonal degree matrix.

Smoothness. The smoothness of a graph signal $\boldsymbol{f} \in \mathbb{R}^n$ w.r.t. G is measured in terms of a quadratic form of the Laplacian (Zhou & Schölkopf, 2004; Shuman et al., 2013):

$$\boldsymbol{f}^\top \mathbf{L} \boldsymbol{f} = \frac{1}{2} \sum_{i,j \in [n]} \mathbf{W}_{ij} (\boldsymbol{f}(i) - \boldsymbol{f}(j))^2, \qquad (4)$$

where $\boldsymbol{f}(i)$ and $\boldsymbol{f}(j)$ are the signal values associated with these two vertices. Intuitively, given that the weights are non-negative, Eq. 4 shows that a graph signal $\boldsymbol{f}$ is considered to be smooth if strongly connected vertices (with a large weight on the edge between them) have similar values. In particular, the smaller the quadratic form in Eq. 4, the smoother the signal on the graph (Dong et al., 2016).

In addition to the traditional smoothness analysis, we provide a new insight based on $(\mathcal{D}, \mathcal{F})$. Let $\hat{\boldsymbol{f}}_i = \mathbf{U}_i^\top \boldsymbol{f} \in \mathbb{R}$

be the i-th component after Graph Fourier Transform, then

$$\begin{aligned} \boldsymbol{f}^\top \mathbf{L} \boldsymbol{f} &= \boldsymbol{f}^\top \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^\top \boldsymbol{f} \\ &= (\mathbf{U}^\top \boldsymbol{f})^\top \boldsymbol{\Lambda} \mathbf{U}^\top \boldsymbol{f} = \hat{\boldsymbol{f}}^\top \boldsymbol{\Lambda} \hat{\boldsymbol{f}} \\ &= \sum_{i=1}^n \hat{\boldsymbol{f}}_i^2 \boldsymbol{\lambda}_i. \end{aligned} \qquad (5)$$

As $\boldsymbol{\lambda}_i \geq 0$, Eq. 5 shows that the smoothness of a graph signal $\boldsymbol{f}$ w.r.t. Laplacian $\mathbf{L}$ in vertex domain is equivalent to the squares of its *weighted norm* in frequency domain with the spectrum of $\mathbf{L}$ serving as weights. Also, Eq. 5 shows that the smoothness w.r.t. $\mathbf{L}$ is decided by both decomposition $\mathbf{U}$ and filtering $\boldsymbol{\Lambda}$. Next, we show what signal smoothness on the underlying graph implies in graph neural networks.

The learnable polynomial filter designs are the most popular approaches in spectral GNN studies, e.g., ChebyNet (Defferrard et al., 2016), CayleyNet (Levie et al., 2019), Bern-Net (He et al., 2021), GPR (Chien et al., 2021), Jacobi-Conv (Wang & Zhang, 2022), Corr-free (Yang et al., 2022a), etc. But applying various polynomials g_θ only introduces modifications on filtering, with decomposition unchanged as $g_\theta(\mathbf{L}) = \mathbf{U} g_\theta(\boldsymbol{\Lambda}) \mathbf{U}^\top$. This leads to the negative effect on signal smoothness as follows.

Proposition 4.1. *For a polynomial graph filter with the polynomial g_θ and the underlying (normalized) Laplacian $\mathbf{L}$ with the spectrum $\boldsymbol{\lambda}_i, i \in [n]$,*
(i) if $|g_\theta(\boldsymbol{\lambda}_i)| < 1$ for all $i \in [n]$, the resulting graph convolution smooths *any input signal w.r.t. $\mathbf{L}$;*
(ii) if $|g_\theta(\boldsymbol{\lambda}_i)| > 1$ for all $i \in [n]$, the resulting graph convolution amplifies *any input signal w.r.t. $\mathbf{L}$.*

We prove Proposition 4.1 in Appendix B.1. Proposition 4.1 shows that if we fix $\mathcal{D}$ and consider $\mathcal{F}$ alone, the smoothness of the resulting signals is sensitive to the range of $g_\theta(\boldsymbol{\lambda}_i)$. A freely learned polynomial coefficient $\boldsymbol{\theta}$ is more likely to result in the smoothness issue. Especially in deep models, the smoothness issue will accumulate. This may explain why learnable polynomial filter design is challenging and usually requires sophisticated polynomial bases. For example, JacobiConv uses Jacobi polynomials and configures the basis $P_k^{a,b}$ by carefully setting a and b (Wang & Zhang, 2022).

Some methods alternatively use pre-defined filters to avoid learning a filter with the above smoothness issue (Wu et al., 2019a; Klicpera et al., 2019a; Ming Chen et al., 2020; Klicpera et al., 2019b; Zhu & Koniusz, 2020). However, for some models, the smoothness issue still exists.

Proposition 4.2. *The GCN layer always smooths any input signal w.r.t. $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}}$.*

We prove Proposition 4.2 in Appendix B.2. Proposition 4.2 shows that the accumulation of signal smoothness may ac-

count for the performance degradation of deep GCN. Similarly, we can prove that SGC (Wu et al., 2019a) suffers from the same issue. Some other fixed filter methods apply residual connections and manually set spectrum shift, which implicitly help to preserve the range of $g_{\boldsymbol{\theta}}(\boldsymbol{\lambda}_i)$, making them tractable for stacking deep architectures (Klicpera et al., 2019a; Ming Chen et al., 2020; Xu et al., 2018).

Comparisons with existing smoothness analysis. First, based on the definition of smoothness in Eq. 4, the smoothness analysis of signals in Proposition 4.1 and Proposition 4.2 is always considered with the underlying graphs, while the convergence analysis in most existing oversmoothing studies is not considered under the graph topology. Also, they only deal with the theoretical infinite depth case and study the final convergence result. Second, Proposition 4.1 shows that in addition to smoothness issue, the amplification issue will also occur complementarilly with inappropriate filter design. The amplification issue is less discussed, in contrast, the numerical instabilities as a reflection of the amplification issue are more well-known. And most methods choose $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ with the bounded spectrum $[-1, 1]$ to avoid numerical instability (Kipf & Welling, 2017). Some other studies find the signal amplification issue from their empirical evaluations and overcome it by proposing various normalization techniques (Zhou et al., 2021; Guo et al., 2022). Our analysis shows that both smoothness and amplification issues can somehow appear complementarily in (fixed $\mathcal{D}$, learnable $\mathcal{F}$) designs for deeper models, which reveals the connections between existing oversmoothing and numerical instability/amplification studies.

4.2. Smoothness over Different Channels

Apart from the smoothness within each channel, the smoothness over different channels in hidden features also indicates information loss and affects the performance. For example, the cosine similarity between signal pairs is usually used as a metric of smoothness over different channels. The larger the cosine similarity is, the smoother they are to each other. In the extreme case, all signal pairs have a cosine similarity equal to one, which means the worst information loss case where each signal is linearly dependent on each other. (Yang et al., 2022a) shows that in a polynomial graph filter with the resulting matrix representation $\mathbf{S} = \mathbf{U}g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})\mathbf{U}^{\top}$,

$$\cos\left(\langle\mathbf{S}\boldsymbol{f}, \mathbf{U}_i\rangle\right) = \frac{(\mathbf{U}_i^{\top}\boldsymbol{f})g_{\boldsymbol{\theta}}(\boldsymbol{\lambda}_i)}{\sqrt{\sum_{j=1}^{n}(\mathbf{U}_j^{\top}\boldsymbol{f})^2 g_{\boldsymbol{\theta}}(\boldsymbol{\lambda}_j)^2}}. \quad (6)$$

As the model goes deeper, the spectrum diversity will accumulate exponentially, i.e. $\mathbf{S}^k = \mathbf{U}g_{\boldsymbol{\theta}}(\boldsymbol{\Lambda})^k\mathbf{U}^{\top}$. As a result, all signals will tend to correlate to the leading eigenvector $\mathbf{U}_0$. Inspired by this, some work explores strategies to restrict the expanding of spectrum diversity in deep models to alleviate this issue, but in return restricting the expressive-

ness of polynomial filters (Yang et al., 2022a).

However, note that $\mathbf{U}_i^{\top}\boldsymbol{f}$ and $g_{\boldsymbol{\theta}}(\boldsymbol{\lambda}_i)$ have equivalent contributions to the smoothness over different channels as reflected in Eq. 6. We can explore a better matrix representation, which improves $(\mathcal{D}, \mathcal{F})$ as a whole, to avoid restrictions imposed on the filters as introduced in other works.

The smoothness of a single channel and that over multi-channels serve as two orthogonal perspectives, having been investigated by many existing works, e.g., the former studied by (Li et al., 2018; Oono & Suzuki, 2020; Rong et al., 2020; Huang et al., 2020), and the latter studied by (Zhao & Akoglu, 2020; Liu et al., 2020; Chien et al., 2021; Yang et al., 2022a; Jin et al., 2022). In comparison to existing work which only consider $\mathcal{F}$ and related spectrum, we further involve $\mathcal{D}$, indicating that by treating $(\mathcal{D}, \mathcal{F})$ as a whole, we can handle the above smoothness issues simultaneously.

5. Experiments

We evaluate our model PDF induced from Parameterized-$(\mathcal{D}, \mathcal{F})$ on the graph-level prediction tasks. Detailed information of the datasets is given in Appendix C.1. We first conduct extensive ablation studies to validate its effectiveness, and then compare its performance with baselines.

5.1. Ablation Studies

We evaluate the effectiveness of PDF as the instantiation of Parameterized-$(\mathcal{D}, \mathcal{F})$ on multichannel signal scenario on ZINC following its default dataset settings. We use "shd" and "idp" to represent the channel-shared and channel-independent architectures, respectively. Both architectures learn the corresponding coefficients $\boldsymbol{\theta} \in \mathbb{R}^{|\mathcal{G}|}$ and $\boldsymbol{\Theta} \in \mathbb{R}^{|\mathcal{G}| \times d}$ from scratch.

Parameterized-$(\mathcal{D}, \mathcal{F})$ involves two components: the construction of $\mathcal{G}$ and the implementation of $f_{\boldsymbol{\theta}}$. For each of them, we design the following variants:

- For the matrix representation $\mathcal{G}$,
  - Lap: $\mathcal{G} = \{(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}})^k | k \in [K]\}$,
  - (ϵ, k): $\mathcal{G} = \{(\tilde{\mathbf{D}}^{\epsilon}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{\epsilon})^k | -0.5 \leq \epsilon \leq 0, k \in [K]\}$;
- For the mapping function $f_{\boldsymbol{\theta}}$,
  - Lin: $f_{\boldsymbol{\theta}}$ is a learnable affine transformation,
  - 1L: $f_{\boldsymbol{\theta}}$ is a 1-layer perceptron,
  - 2L: $f_{\boldsymbol{\theta}}$ is a 2-layer perceptron.

(Lap, Lin) corresponds to general spectral GNNs that apply fixed normalized Laplacian decomposition and learnable filtering. $((\epsilon, k), \text{Lin})$, $((\epsilon, k), 1L)$ and $((\epsilon, k), 2L)$ belong to learnable $(\mathcal{D}, \mathcal{F})$. shd-$((\epsilon, k), \text{Lin})$ is similar to PGSO with the involved $\mathcal{G}$ slightly different. $((\epsilon, k), 2L)^{\text{sps}}$ is used to test the effects of learning within sparse representation subspace $\mathcal{M}_G^{\text{sps}} \subset \mathcal{M}_G$, which is implemented by masking all

Table 2. Ablation studies on ZINC.

	Ablation	valid MAE	test MAE
shd+	(Lap, Lin)	0.227 ± 0.0445	0.219 ± 0.0520
	$((\epsilon, k), \text{Lin})$	0.184 ± 0.0276	0.167 ± 0.0234
	$((\epsilon, k), 2L)^{\text{sps}}$	0.174 ± 0.0125	0.150 ± 0.0141
	$((\epsilon, k), 1L)$	0.172 ± 0.0087	0.160 ± 0.0119
	$((\epsilon, k), 2L)$	0.121 ± 0.0137	0.112 ± 0.0138
idp+	(Lap, Lin)	0.188 ± 0.0048	0.172 ± 0.0041
	$((\epsilon, k), \text{Lin})$	0.168 ± 0.0071	0.150 ± 0.0038
	$((\epsilon, k), 2L)^{\text{sps}}$	0.127 ± 0.0028	0.111 ± 0.0024
	$((\epsilon, k), 1L)$	0.104 ± 0.0028	0.088 ± 0.0031
	$((\epsilon, k), 2L)$	$\mathbf{0.085 \pm 0.0038}$	$\mathbf{0.066 \pm 0.0020}$

neighbors beyond 2-hop in $((\epsilon, k), 2L)$. Detailed model configurations of each test case are provided in Appendix C.6. Tab. 2 presents the ablation study results, and more detailed statistics are in Appendix C.4.

Effectiveness of learnable $(\mathcal{D}, \mathcal{F})$: (Lap, Lin) is analogous to spectral graph convolutions where all matrix representations that can be learned share the same eigenspace, which results in a fixed $\mathcal{D}$. Also, Lap refers to $(-0.5, k) \subset (\epsilon, k)$, and therefore, for a given G and Lin, the learning space w.r.t. Θ has $\{((-0.5, k), \text{Lin}) | \Theta\} \subset \{((\epsilon, k), \text{Lin}) | \Theta\}$. The results show that $((\epsilon, k), \text{Lin})$ with learnable $(\mathcal{D}, \mathcal{F})$ outperforms (Lap, Lin) with learnable $\mathcal{F}$ on both architectures.

Effects of the expressiveness of f_θ: Note that the expressiveness comparisons of f_θ in $((\epsilon, k), \text{Lin})$, $((\epsilon, k), 1L)$ and $((\epsilon, k), 2L)$ is $\text{Lin} \prec 1L \prec 2L$. Thus, for a given graph G and (ϵ, k), the learning space comparison w.r.t. Θ is

$$\{((\epsilon, k), \text{Lin}) | \Theta\} \subset \{((\epsilon, k), 1L) | \Theta\}$$
$$\subset \{((\epsilon, k), 2L) | \Theta\} \subset \mathcal{M}_G.$$

This is exactly reflected on their performance comparisons where the more expressive f_θ is, the better performance the resulting model achieves. This validates our analysis that learning from a larger subspace within $\mathcal{M}_G$ helps to find more effective graph matrix representation, i.e., $(\mathcal{D}, \mathcal{F})$ for input signals. Also, 2L is sufficient to be used to parameterize any desired mapping according to the universal approximation theorem (Hornik et al., 1989), and we can see that $((\epsilon, k), 2L)$ does benefit from this guarantee, making it outperform the other two by a large margin.

Effects of sparse representations: Compared with $((\epsilon, k), 2L)$, the learning space of $((\epsilon, k), 2L)^{\text{sps}}$ is limited within $\mathcal{M}_G^{\text{sps}}$, and the results show that $((\epsilon, k), 2L)^{\text{sps}}$ does not outperform $((\epsilon, k), 2L)$. Sparse representations only correspond to a restricted subspace $\mathcal{M}_G^{\text{sps}} \subset \mathcal{M}_G$. It acts as a trade-off between scalability and prediction performance. Users are suggested to use dense representations if it is tractable on the given graph scales.

Effects of architecture designs: Channel-independent learn-

Table 3. Results on ZINC, ogbg-molpcba and ogbg-ppa with the number of parameters used, where the best results are in bold, and second-best are underlined.

Method	ZINC MAE $\downarrow_{\#\text{para}}$	ogbg-molpcba AP(%) $\uparrow_{\#\text{para}}$	ogbg-ppa ACC(%) $\uparrow_{\#\text{para}}$
GCN	$0.367 \pm 0.011 \ _{505k}$	$24.24 \pm 0.34 \ _{2.0m}$	$68.39 \pm 0.84 \ _{0.5m}$
GIN	$0.526 \pm 0.051 \ _{510k}$	$27.03 \pm 0.23 \ _{3.4m}$	$68.92 \pm 1.00 \ _{1.9m}$
GAT	$0.384 \pm 0.007 \ _{531k}$	-	-
GraphS	$0.398 \pm 0.002 \ _{505k}$	-	-
GatedG	$0.214 \pm 0.006 \ _{505k}$	-	-
MPNN	$0.145 \pm 0.007 \ _{481k}$	-	-
DeeperG	-	$28.42 \pm 0.43 \ _{5.6m}$	$77.12 \pm 0.71 \ _{2.3m}$
PNA	$0.142 \pm 0.010 \ _{387k}$	$28.38 \pm 0.35 \ _{6.6m}$	-
DGN	$0.168 \pm 0.003 \ _{NA}$	$28.85 \pm 0.30 \ _{6.7m}$	-
GSN	$0.101 \pm 0.010 \ _{523k}$	-	-
GINE-AP	-	$\underline{29.79 \pm 0.30} \ _{6.2m}$	-
PHC-GN	-	$29.47 \pm 0.26 \ _{1.7m}$	-
ExpC	-	$23.42 \pm 0.29 \ _{NA}$	$79.76 \pm 0.72 \ _{1.4m}$
GT	$0.226 \pm 0.014 \ _{NA}$	-	-
SAN	$0.139 \pm 0.006 \ _{509k}$	$27.65 \pm 0.42 \ _{NA}$	-
Graphor	$0.122 \pm 0.006 \ _{489k}$	-	-
KS-SAT	$0.094 \pm 0.008 \ _{NA}$	-	$75.22 \pm 0.56 \ _{NA}$
GPS	$\underline{0.070 \pm 0.004} \ _{424k}$	$29.07 \pm 0.28 \ _{9.7m}$	$\mathbf{80.15 \pm 0.33} \ _{3.4m}$
GM-Mix	$0.075 \pm 0.001 \ _{NA}$	-	-
PDF (our)	$\mathbf{0.066 \pm 0.002} \ _{500k}$	$\mathbf{30.31 \pm 0.26} \ _{3.8m}$	$\underline{80.10 \pm 0.52}^* \ _{2.0m}$

able $(\mathcal{D}, \mathcal{F})$ outperforms the channel-shared one in all cases. Meanwhile, channel-independent one is much more stable, as reflected by smaller variations (STD) in different runs in Tab. 2, as well as the training curves in Appendix C.4. These results empirically show that assigning each channel with an individual $(\mathcal{D}, \mathcal{F})$ is more appropriate.

5.2. Performance Comparison

All baseline results of ZINC, ogbg-molpcba and pgbg-ppa in Tab. 3 are quoted from their leaderboards[4] or the original papers. And all baseline results of TUDatasets in Tab. 4 are quoted from their original papers. All baseline models in this section are summarized in Appendix C.2. Ogbg-ppa and RDT-B are large scale graphs with more than 200 vertices. We apply sparse matrix representations subspace $\mathcal{M}_G^{\text{sps}}$ on them with the results marked with $*$. All detailed hyperparameter configurations can be found in Appendix C.3. PDF appearing in the baseline comparisons refers to the implementation idp-$((\epsilon, k), 2L)$.

ZINC. We use the default dataset splits for ZINC, and following baselines settings on leaderboard, set the number of parameters around 500K. PDF achieves the superior performance on ZINC compared with all existing models, including both the lowest MAE and STD in multiple runs.

OGB. We use the default dataset splits provided in OGB. The results in Tab. 4 involve molecular benchmark ogbg-

---

[4]https://paperswithcode.com/sota/graph-regression-on-zinc-500k and https://ogb.stanford.edu/docs/leader_graphprop/

Table 4. Results on TUDataset (Higher is better).

Method	MUTAG	NCI1	NCI109	ENZYMES	PTC_MR	PROTEINS	IMDB-B	RDT-B
GK	81.52±2.11	62.49±0.27	62.35±0.3	32.70±1.20	55.65±0.5	71.39±0.3	-	77.34±0.18
RW	79.11±2.1	-	-	24.16±1.64	55.91±0.3	59.57±0.1	-	-
PK	76.0±2.7	82.54±0.5	-	-	59.5±2.4	73.68±0.7	-	-
FGSD	**92.12**	79.80	78.84	-	62.8	73.42	73.62	-
AWE	87.87±9.76	-	-	35.77±5.93	-	-	74.45±5.80	87.89±2.53
DGCNN	85.83±1.66	74.44±0.47	-	51.0±7.29	58.59±2.5	75.54±0.9	70.03±0.90	-
PSCN	88.95±4.4	74.44±0.5	-	-	62.29±5.7	75±2.5	71±2.3	86.30±1.58
DCNN	-	56.61±1.04	-	-	-	61.29±1.6	49.06±1.4	-
ECC	76.11	76.82	75.03	45.67	-	-	-	-
DGK	87.44±2.72	80.31±0.46	80.32±0.3	53.43±0.91	60.08±2.6	75.68±0.5	66.96±0.6	78.04±0.39
GraphSAGE	85.1±7.6	76.0±1.8	-	58.2±6.0	-	-	72.3±5.3	-
CapsGNN	88.67±6.88	78.35±1.55	-	54.67±5.67	-	76.2±3.6	73.1±4.8	-
DiffPool	-	76.9±1.9	-	<u>62.53</u>	-	**78.1**	-	-
GIN	89.4±5.6	82.7±1.7	-	-	64.6±7.0	76.2±2.8	<u>75.1±5.1</u>	92.4±2.5
k-GNN	86.1	76.2	-	-	60.9	75.5	74.2	-
IGN	83.89±12.95	74.33±2.71	72.82±1.45	-	58.53±6.86	76.58±5.49	72.0±5.54	-
PPGNN	<u>90.55±8.7</u>	<u>83.19±1.11</u>	82.23±1.42	-	66.17±6.54	<u>77.20±4.73</u>	73.0±5.77	-
GCN[2]	89.39±1.60	82.74±1.35	<u>83.00±1.89</u>	-	<u>66.84±1.79</u>	71.71±1.04	74.80±2.01	-
PDF (ours)	89.91±4.35	**85.47±1.38**	**83.62±1.38**	**73.50±6.39**	**68.36±8.38**	76.28±5.1	**75.60±2.69**	**93.40±1.30***

molpcba with small and sparse connected graphs, and protein-protein interaction benchmark ogbg-ppa with large and densely connected graphs. PDF achieves the best AP with relatively fewer parameters on ogbg-molpcba. On ogbg-ppa, PDF explores within $\mathcal{M}_G^{\text{sps}}$ to balance computational efficiency, and is still comparable to SOTA.

TUDataset. We test our model on 8 TUDataset datasets involving both bioinformatics datasets (MUTAG, NCI1, NCI109, ENZYMES, PTC_MR and PROTEINS), and social network datasets (IMDB-B and RDT-B). To ensure a fair comparison with baselines, we follow the standard 10-fold cross-validation and dataset splits in (Zhang et al., 2018), and then report our results according to the protocol described in (Xu et al., 2019b; Ying et al., 2018). The results are presented in Tab.4. PDF achieves the highest classification accuracies on 6 out of 8 datasets, among which PDF outperforms existing models by a large margin on NCI1 and ENZYMES respectively.

Also, PDF benefits from its simple architecture and is more computational efficient than other SOTA models. In Appendix C.5, we show that PDF shares a similar time complexity with GIN and GCN. Also, in our tests, the practical training and evaluation time on each epoch is similar to GIN, which can be more than $2\times$ faster than some latest SOTA models that improve their performance by leveraging high expressive power or transformer architectures.

6. Related Work

GWNN (Xu et al., 2019a) replaces the Fourier basis with wavelet basis, which refers to a different decomposition on input signals. But similar to Fourier basis, it still uses fixed decomposition and cannot adopt relevant bases for input signals.

JacobiConv (Wang & Zhang, 2022) and Corr-free (Yang et al., 2022a) learn individual filtering for each channel, which is similar to our channel-independent architecture, but share the decomposition over these channels since they are induced by the spectral graph convolution theories.

Parameterized Graph Shift Operator (PGSO) (Dasoulas et al., 2021) is motivated by spanning the space of commonly used GSOs as a replacement of the (normalized) Laplacian/Adjacency. As different GSOs generally do not share the eigenspace, PGSO can learn both $\mathcal{D}$ and $\mathcal{F}$. However, the linear combination of several GSOs can only leverage limited subspace within $\mathcal{M}_G$. From the perspective of Parameterized-$(\mathcal{D}, \mathcal{F})$, PGSO is somehow a restricted implementation of our shd-PDF, where k is fixed to 1 and $f_{\boldsymbol{\theta}}$ is a learnable linear transformation. The limited learning space may not be sufficient to obtain a suitable $(\mathcal{D}, \mathcal{F})$ for input signals. As PGSO is analogous to one test case in our ablation studies, we can see that our model outperforms it, as shown in Sec. 5.1.

Multi-graph convolutions (Geng et al., 2019; Khan & Blumenstock, 2019) learn from multiple graphs, with each one having its own semantic meaning. These methods are more application-oriented, e.g., traffic forecasting, and usually need domain expertise to define multigraphs, while our method has no such restriction.

7. Conclusion

In this work, we propose Parameterized-$(\mathcal{D}, \mathcal{F})$, which aims to learn a (decomposition, filtering) as a whole, i.e., a graph matrix representation, for input graph signals. It well unifies existing GNN models and the inspired new model achieves superior performance while preserving computational efficiency.

Acknowledgements

References

Abdi, H. and Williams, L. J. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

Atwood, J. and Towsley, D. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1993–2001, 2016.

Bastos, A., Nadgeri, A., Singh, K., Kanezashi, H., Suzumura, T., and Mulang', I. O. How expressive are transformers in spectral domain for graphs? *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=aRsLetumx1.

Beani, D., Passaro, S., Létourneau, V., Hamilton, W., Corso, G., and Liò, P. Directional graph networks. In *International Conference on Machine Learning*, pp. 748–758. PMLR, 2021.

Bouritsas, G., Frasca, F., Zafeiriou, S., and Bronstein, M. M. Improving graph neural network expressivity via subgraph isomorphism counting. *arXiv preprint arXiv:2006.09252*, 2020.

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2017.

Brossard, R., Frigo, O., and Dehaene, D. Graph convolutions that can finally model local structure. *arXiv preprint arXiv:2011.15069*, 2020.

Butler, S. and Chung, F. Spectral graph theory. handbook of linear algebra (2nd edition, l. hogben, ed.). *Discrete Mathematics and its Applications. CRC Press, Boca Raton*, pp. 47/1—47/14, 2017.

Chen, D., O'Bray, L., and Borgwardt, K. Structure-aware transformer for graph representation learning. In *International Conference on Machine Learning*, pp. 3469–3489. PMLR, 2022.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C.-J. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 257–266, 2019.

Chien, E., Peng, J., Li, P., and Milenkovic, O. Adaptive universal generalized pagerank graph neural network. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=n6jl7fLxrP.

Chung, F. R. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets. In *Advances in Neural Information Processing Systems*, 2020.

Dasoulas, G., Lutzeyer, J. F., and Vazirgiannis, M. Learning parametrised graph shift operators. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0OlrLvrsHwQ.

de Haan, P., Cohen, T. S., and Welling, M. Natural graph networks. *Advances in Neural Information Processing Systems*, 33:3636–3646, 2020.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852, 2016.

Deri, J. A. and Moura, J. M. Spectral projector-based graph fourier transforms. *IEEE Journal of Selected Topics in Signal Processing*, 11(6):785–795, 2017.

Dong, X., Thanou, D., Frossard, P., and Vandergheynst, P. Learning laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23):6160–6173, 2016.

Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Geng, X., Li, Y., Wang, L., Zhang, L., Yang, Q., Ye, J., and Liu, Y. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 3656–3663, 2019.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Guo, K., Zhou, K., Hu, X., Li, Y., Chang, Y., and Wang, X. Orthogonal graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 3996–4004, 2022.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017.

Hammond, D. K., Vandergheynst, P., and Gribonval, R. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

He, M., Wei, Z., Huang, Z., and Xu, H. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. In *NeurIPS*, 2021.

He, X., Hooi, B., Laurent, T., Perold, A., LeCun, Y., and Bresson, X. A generalization of vit/mlp-mixer to graphs, 2022.

Hornik, K., Stinchcombe, M., and White, H. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Huang, W., Rong, Y., Xu, T., Sun, F., and Huang, J. Tackling over-smoothing for general graph convolutional networks. *arXiv preprint arXiv:2008.09864*, 2020.

Ivanov, S. and Burnaev, E. Anonymous walk embeddings. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2191–2200, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/ivanov18a.html.

Jiang, X., Yang, Z., Wen, P., Su, L., and Huang, Q. A sparse-motif ensemble graph convolutional network against over-smoothing. In *IJCAI*, 2022.

Jin, W., Liu, X., Ma, Y., Aggarwal, C., and Tang, J. Feature overcorrelation in deep graph neural networks: A new perspective. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.

Khan, M. R. and Blumenstock, J. E. Multi-gcn: Graph convolutional networks for multi-view networks, with applications to global poverty. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 606–613, 2019.

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

Klicpera, J., Bojchevski, A., and Günnemann, S. Predict then propagate: Graph neural networks meet personalized pagerank. In *International Conference on Learning Representations (ICLR)*, 2019a.

Klicpera, J., Weißenberger, S., and Günnemann, S. Diffusion improves graph learning. *Advances in Neural Information Processing Systems*, 32:13354–13366, 2019b.

Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., and Tossou, P. Rethinking graph transformers with spectral attention. *arXiv preprint arXiv:2106.03893*, 2021.

Le, T., Bertolini, M., Noé, F., and Clevert, D.-A. Parameterized hypercomplex graph neural networks for graph classification. *arXiv preprint arXiv:2103.16584*, 2021.

Lee, J. B., Rossi, R. A., Kong, X., Kim, S., Koh, E., and Rao, A. Graph convolutional networks with motif-based attention. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pp. 499–508, 2019.

Levie, R., Monti, F., Bresson, X., and Bronstein, M. M. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2019. doi: 10.1109/TSP.2018.2879624.

Li, G., Xiong, C., Thabet, A., and Ghanem, B. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:2006.07739*, 2020.

Li, Q., Han, Z., and Wu, X.-M. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Liu, M., Gao, H., and Ji, S. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 338–348, 2020.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. Provably powerful graph networks. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pp. 2156–2167, 2019.

Min, E., Chen, R., Bian, Y., Xu, T., Zhao, K., Huang, W., Zhao, P., Huang, J., Ananiadou, S., and Rong, Y. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022.

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

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Neumann, M., Garnett, R., Bauckhage, C., and Kersting, K. Propagation kernels: efficient graph kernels from propagated information. *Machine Learning*, 102(2):209–245, 2016.

Niepert, M., Ahmed, M., and Kutzkov, K. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pp. 2014–2023, 2016.

Oono, K. and Suzuki, T. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1ldO2EFPr.

Ortega, A., Frossard, P., Kovačević, J., Moura, J. M., and Vandergheynst, P. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a General, Powerful, Scalable Graph Transformer. *arXiv:2205.12454*, 2022.

Rong, Y., Huang, W., Xu, T., and Huang, J. Dropedge: Towards deep graph convolutional networks on node classification. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=Hkx1qkrKPr.

Sandryhaila, A. and Moura, J. M. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61 (7):1644–1656, 2013.

Sato, R. A survey on the expressive power of graph neural networks. *arXiv preprint arXiv:2003.04078*, 2020.

Shervashidze, N., Vishwanathan, S., Petri, T., Mehlhorn, K., and Borgwardt, K. Efficient graphlet kernels for large

graph comparison. In *Artificial Intelligence and Statistics*, pp. 488–495, 2009.

Shuman, D. I., Narang, S. K., Frossard, P., Ortega, A., and Vandergheynst, P. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3693–3702, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Verma, S. and Zhang, Z.-L. Hunt for the unique, stable, sparse and fast feature learning on graphs. In *Advances in Neural Information Processing Systems*, pp. 88–98, 2017.

Vishwanathan, S. V. N., Schraudolph, N. N., Kondor, R., and Borgwardt, K. M. Graph kernels. *Journal of Machine Learning Research*, 11(Apr):1201–1242, 2010.

Wang, X. and Zhang, M. How powerful are spectral graph neural networks. *ICML*, 2022.

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. Simplifying graph convolutional networks. In *Proceedings of the 36th International Conference on Machine Learning*, pp. 6861–6871. PMLR, 2019a.

Wu, M., Pan, S., Zhou, C., Chang, X., and Zhu, X. Unsupervised domain adaptive graph convolutional networks. In *Proceedings of The Web Conference 2020*, pp. 1457–1467, 2020.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019b.

Xanthopoulos, P., Pardalos, P. M., Trafalis, T. B., Xanthopoulos, P., Pardalos, P. M., and Trafalis, T. B. Linear discriminant analysis. *Robust data mining*, pp. 27–33, 2013.

Xinyi, Z. and Chen, L. Capsule graph neural network. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Byl8BnRcYm.

Xu, B., Shen, H., Cao, Q., Qiu, Y., and Cheng, X. Graph wavelet neural network. In *International Conference*

on Learning Representations, 2019a. URL https://openreview.net/forum?id=H1ewdiR5tQ.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K.-i., and Jegelka, S. Representation learning on graphs with jumping knowledge networks. In *International Conference on Machine Learning*, pp. 5453–5462. PMLR, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=ryGs6iA5Km.

Yanardag, P. and Vishwanathan, S. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1365–1374. ACM, 2015.

Yang, M., Shen, Y., Qi, H., and Yin, B. Breaking the expressive bottlenecks of graph neural networks. *arXiv preprint arXiv:2012.07219*, 2020.

Yang, M., Shen, Y., Li, R., Qi, H., Zhang, Q., and Yin, B. A new perspective on the effects of spectrum in graph neural networks. In *Proceedings of the 39th International Conference on Machine Learning*, 2022a.

Yang, M., Wang, R., Shen, Y., Qi, H., and Yin, B. Breaking the expression bottleneck of graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2022b. doi: 10.1109/TKDE.2022.3168070.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do transformers really perform bad for graph representation? *arXiv preprint arXiv:2106.05234*, 2021.

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling. In *Advances in Neural Information Processing Systems*, pp. 4800–4810, 2018.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Zhao, L. and Akoglu, L. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkecl1rtwB.

Zhou, D. and Schölkopf, B. A regularization framework for learning from graph data. In *ICML 2004 Workshop on Statistical Relational Learning and Its Connections to Other Fields (SRL 2004)*, pp. 132–137, 2004.

Zhou, K., Dong, Y., Wang, K., Lee, W. S., Hooi, B., Xu, H., and Feng, J. Understanding and resolving performance degradation in deep graph convolutional networks. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 2728–2737, 2021.

Zhu, H. and Koniusz, P. Simple spectral graph convolution. In *International Conference on Learning Representations*, 2020.

A. Graph Representation.

Considering the various possible representations for the topology of a graph, we denote the matrix representation space of an undirected graph G by $\mathcal{M}_G$. Admittedly, providing the formal unified definition for $\mathcal{M}_G$ or enumerating all its elements is indeed hard, but it is still possible to give some feasible instances. For example, $\mathcal{M}_G$ can include

- Graph shift operators (Sandryhaila & Moura, 2013): including the adjacency matrix, Laplacian matrix, and their various normalization versions, as well as the mean aggregation operator of GNNs and Parametrized graph shift operator (PGSO) (Dasoulas et al., 2021);
- Structure derived matrices: k path-length counting matrix $\mathbf{A}^k$, shortest path distance matrix (SPD), motif adjacency matrix (Lee et al., 2019; Jiang et al., 2022), and point-wise mutual information matrix (Wu et al., 2020), etc.
- Feature-engineering-based matrices, like the neural graph fingerprint (Duvenaud et al., 2015).

Different matrices emphasize the graph structure or topology from different angles, like local or global view; and each of them has its own limitations in that there are some properties that the matrix cannot always determine (Butler & Chung, 2017). Under the graph spectral formulation, here we only account for $\mathcal{M}_G$ which is composed of some symmetric matrices; for any $S \in \mathcal{M}_G$, it has the unique eigendecomposition as $S = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, according to the spectral theorem.

B. Proofs.

B.1. Proof of Proposition 4.1

Proof. Let $\boldsymbol{f}' = g_{\boldsymbol{\theta}}(\mathbf{L})\boldsymbol{f}$ denote the graph signal after graph convolution. If $|g_{\boldsymbol{\theta}}(\boldsymbol{\lambda}_i)| < 1, i \in [n]$, then

$$
\begin{aligned}
\boldsymbol{f}'^\top \mathbf{L} \boldsymbol{f}' &= (g_{\boldsymbol{\theta}}(\mathbf{L})\boldsymbol{f})^\top \mathbf{L} g_{\boldsymbol{\theta}}(\mathbf{L})\boldsymbol{f} \\
&= \boldsymbol{f}^\top g_{\boldsymbol{\theta}}(\mathbf{L}) \mathbf{L} g_{\boldsymbol{\theta}}(\mathbf{L})\boldsymbol{f} \\
&= \boldsymbol{f}^\top \mathbf{U} g_{\boldsymbol{\theta}}(\mathbf{\Lambda}) \mathbf{\Lambda} g_{\boldsymbol{\theta}}(\mathbf{\Lambda}) \mathbf{U}^\top \boldsymbol{f} \\
&= \boldsymbol{f}^\top \mathbf{U} \mathrm{diag}(g_{\boldsymbol{\theta}}(\boldsymbol{\lambda}_i)^2 \boldsymbol{\lambda}_i) \mathbf{U}^\top \boldsymbol{f} \\
&= \hat{\boldsymbol{f}}^\top \mathrm{diag}(g_{\boldsymbol{\theta}}(\boldsymbol{\lambda}_i)^2 \boldsymbol{\lambda}_i) \hat{\boldsymbol{f}} \\
&= \sum_{i=1}^n \hat{\boldsymbol{f}}_i^2 \boldsymbol{\lambda}_i g_{\boldsymbol{\theta}}(\boldsymbol{\lambda}_i)^2 \\
&< \sum_{i=1}^n \hat{\boldsymbol{f}}_i^2 \boldsymbol{\lambda}_i \qquad / * \boldsymbol{\lambda}_i \geq 0, i \in [n] * / \\
&= \boldsymbol{f}^\top \mathbf{L} \boldsymbol{f} \qquad / * \text{As in Eq. 5} * /
\end{aligned}
$$

Hence, $\boldsymbol{f}'$ is smoother than $\boldsymbol{f}$ w.r.t. $\mathbf{L}$. Similarly, we can prove $\boldsymbol{f}'^\top \mathbf{L} \boldsymbol{f}' > \boldsymbol{f}^\top \mathbf{L} \boldsymbol{f}$ $\boldsymbol{f}'$ when $|g_{\boldsymbol{\theta}}(\boldsymbol{\lambda}_i)| > 1, i \in [n]$. □

B.2. Proof of Proposition 4.2

Proof. In GCN, we have $\boldsymbol{f}' = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\boldsymbol{f}$. Let $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$. Then $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}} = \mathbf{I} - \tilde{\mathbf{L}} = g_{\boldsymbol{\theta}}(\tilde{\mathbf{L}})$ is the polynomial of the Laplacian $\tilde{\mathbf{L}}$. Let $\tilde{\boldsymbol{\lambda}}_i, i \in [n]$ denote the spectrum of $\tilde{\mathbf{L}}$. Then, $0 \leq \tilde{\boldsymbol{\lambda}}_i < 2$ and $g_{\boldsymbol{\theta}}(\tilde{\boldsymbol{\lambda}}_i) = 1 - \tilde{\boldsymbol{\lambda}}_i \in (-1, 1)$. According to Proposition 4.1, we have $\boldsymbol{f}'^\top \tilde{\mathbf{L}} \boldsymbol{f}' \leq \boldsymbol{f}^\top \tilde{\mathbf{L}} \boldsymbol{f}$, □

C. Experimental Details.

C.1. Datasets Statistics.

All detailed statistics of the datasets used in our experiments are presented in Tab. 5. The corresponding tasks involve graph regression task and graph classification task collecting from real-world molecules, social networks and protein-protein interactions. The scale of datasets ranges from hundreds of graphs (e.g. MUTAG, PTC_MR) to hundreds of thousands of graphs (e.g., ogbg-molpcba, pgbg-ppa). The scale of graphs involved in each dataset ranges from 10-20 (e.g., MUTAG, PTC_MR, IMDB-B) to 400-500 (e.g., RDT-B). Also, the density of connectivity e.g., $\frac{2 \times \text{Avg \# edges}}{\text{Avg \# nodes}}$ ranges from 2.x (most molecular datasets) to 18.x (e.g., ogbg-ppa).

13

Table 5. Statistics of the datasets used in our experiments.

Dataset	# Graphs	Avg # nodes	Avg # edges	Node attr	Edge attr	Task type
ZINC	12,000	23.2	24.9	Y	Y	Regression
ogbg-molpcba	437,929	26.0	28.1	Y	Y	Binary classi.
ogbg-ppa	158,100	243.4	2,266.1	N	Y	37-way classi.
MUTAG	188	17.93	19.79	N	N	Binary classi.
NCI1	4110	29.87	32.39	N	N	Binary classi.
NCI109	4127	29.68	32.13	N	N	Binary classi.
ENZYMES	600	32.63	62.14	Y	N	6-way classi
PTC_MR	344	14.29	14.69	N	N	Binary classi.
PROTEINS	1113	39.06	72.82	Y	N	Binary classi.
IMDB-B	1000	19.77	96.53	N	N	Binary classi.
RDT-B	2000	429.63	497.75	N	N	Binary classi.

C.2. Baselines.

The baseline models used for comparisons include: GK (Shervashidze et al., 2009), RW (Vishwanathan et al., 2010), PK (Neumann et al., 2016), FGSD (Verma & Zhang, 2017), AWE (Ivanov & Burnaev, 2018), DGCNN (Zhang et al., 2018), PSCN (Niepert et al., 2016), DCNN (Atwood & Towsley, 2016), ECC (Simonovsky & Komodakis, 2017), DGK (Yanardag & Vishwanathan, 2015), CapsGNN (Xinyi & Chen, 2019), DiffPool (Ying et al., 2018), GIN (Xu et al., 2019b), k-GNN (Morris et al., 2019), IGN (Maron et al., 2018), PPGNN (Maron et al., 2019), GCN^2 (de Haan et al., 2020) GraphSage (Hamilton et al., 2017), GAT (Veličković et al., 2018), GatedGCN-PE (Bresson & Laurent, 2017), MPNN (sum) (Gilmer et al., 2017), DeeperG (Li et al., 2020), PNA (Corso et al., 2020), DGN (Beani et al., 2021), GSN (Bouritsas et al., 2020), GINE-APPNP (Brossard et al., 2020), PHC-GNN (Le et al., 2021), ExpC (Yang et al., 2022b), GT (Dwivedi et al., 2020), SAN (Kreuzer et al., 2021), Graphormer (Ying et al., 2021), KS-SAT (Chen et al., 2022), GPS (Rampášek et al., 2022), GM-Mix (He et al., 2022).

C.3. Experimental Setup.

Tab. 6 and Tab. 7 present all hyperparameter configurations used in baseline comparisons in Sec. 5.2. On ZINC, we keep the number of learnable coefficients used on our model close to 500K as configured by other baseline methods.

C.4. Additional Experimental Details.

We present the learning curves of training, valid and test sets on ZINC in Fig. 3 and Fig. 4, which are exactly the same runs as that in Tab. 2. But the curves give more prominent results: on all test cases, channel-independent Parameterized-$(\mathcal{D}, \mathcal{F})$ architecture which assigns each channel with an independent $(\mathcal{D}, \mathcal{F})$ is much more robust than channel-shared Parameterized-$(\mathcal{D}, \mathcal{F})$ architecture in different runs.

C.5. Complexity Analysis and Computational Efficiency.

We analyze the time and space complexities of our PDF[5]. The results are presented in Tab. 8, where n, m, l, d refer to the number of vertices, edges, layers, and hidden dimensions respectively; g denotes the number of attention heads used in multi-head GAT or the number of aggregators used in PNA and ExpC, and $k = |\mathcal{G}|$ in PDF. Generally, $k \ll d$, hence, the additional computations of both channel-shared and channel independent PDF are minor compared with GCN or GIN. Also, in the channel-independent architecture, each individual channel's computation is fully independent and can be parallelized.

We tested the practical running time on a shared computing cluster environment running on Nvidia A100 (40GiB) GPU server. All test codes are built upon Deep Graph Library (DGL). Tab. 9 presents the running time on ZINC dataset which involves small molecular graphs. And we test both dense graph matrix representations and 1-hop sparse graph matrix representations. Tab. 10 presents the running time on RDT-B dataset, which involves large social network graphs. And we

---

[5]Many GNN studies discuss the complexity of their models, but the results vary from each other. Here, we follow the widely-adopted one used in (Wu et al., 2019b; Veličković et al., 2018; Chiang et al., 2019).

Table 6. Hyperparameter settings on ZINC and OGB datasets.

Hyperparameter	ZINC	ogbg-molpcba	ogbg-ppa
Hidden Dim.	160	384	384
Num. Layers	6	8	4
Drop. Rate	0	0	0.5
Readout	Mean	Max	Sum
Batch Size	64	64	16
Initial LR	0.001	0.0005	0.001
LR Dec. Steps	35	5	30
LR Dec. Rate	0.6	0.2	0.65
# Warm. Steps	5	5	5
Weight Dec.	5e-5	1e-2	0
$\mathcal{G}$	Dense	Dense	Sparse
σ in $f_{\boldsymbol{\theta}}$	GELU	GELU	GELU
$\{(\epsilon, k)\}$	(-0.1, 4), (-0.2, 4), (-0.3, 4), (-0.4, 4), (-0.5, 4)	(-0.2, 1), (-0.2, 2), (-0.2, 3), (-0.2, 4), (-0.2, 5), (-0.25, 1), (-0.25, 2), (-0.25, 3), (-0.25, 4), (-0.25, 5), (-0.3, 1), (-0.3, 2), (-0.3, 3), (-0.3, 4), (-0.3, 5), (-0.35, 1), (-0.35, 2), (-0.35, 3), (-0.35, 4), (-0.35, 5)	(0, 1), (-0.05, 1), (-0.1, 1), (-0.15, 1), (-0.2, 1), (-0.2, 2), (-0.25, 1), (-0.25, 2), (-0.3, 1), (-0.3, 2), (-0.35, 1), (-0.35, 2), (-0.4, 1), (-0.4, 2), (-0.4, 3), (-0.45, 1), (-0.45, 2), (-0.45, 3), (-0.5, 1), (-0.5, 2), (-0.5, 3)

only test 1-hop sparse graph matrix representations. The running time results show that all our four test cases on ZINC and all our two test cases on RDT-B have minor differences in computational efficiency. And their running time is analogical to GIN. Channel-shared architectures almost have the same efficiency compared with channel-independent architecture on both PDF and PDF [1-hop]. On both architectures, PDF takes slightly longer time than PDF [1-hop] due to processing densely vertex connections. But the differences are also very small. In conclusion, the best performing test case idp-PDF, which is used in baseline comparisons in Sec. 5.2, has similar training and evaluation speed compared with GIN.

C.6. Ablation Settings

Tab. 11 summaries the model setting details in our ablation studies. To ensure a fair comparison, all five cases share the same hyperparameter setting, which is also the same as that used in baseline comparisons in Tab. 6. Also, they apply the same number of input graph matrix representations, i.e. $|\mathcal{G}| = |\{(\epsilon, k)\}|$. In $((\epsilon, k), \text{Lin})$ case, the applied $\{(\epsilon, k)\}$ is slightly different with that in the case $((\epsilon, k), 2L)$ because in our test, $((\epsilon, k), \text{Lin})$ did not get the best performance when sharing the same $\{(\epsilon, k)\}$ with $((\epsilon, k), 2L)$. Based on the above settings, all five implementation cases share the same amount of learnable coefficients, 499681.
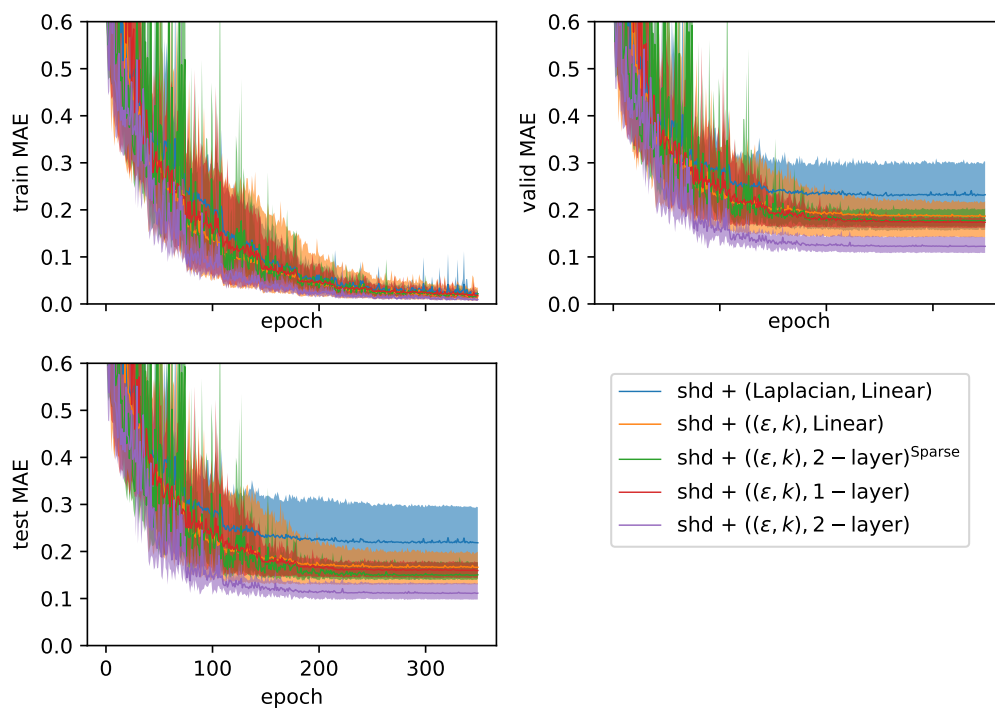
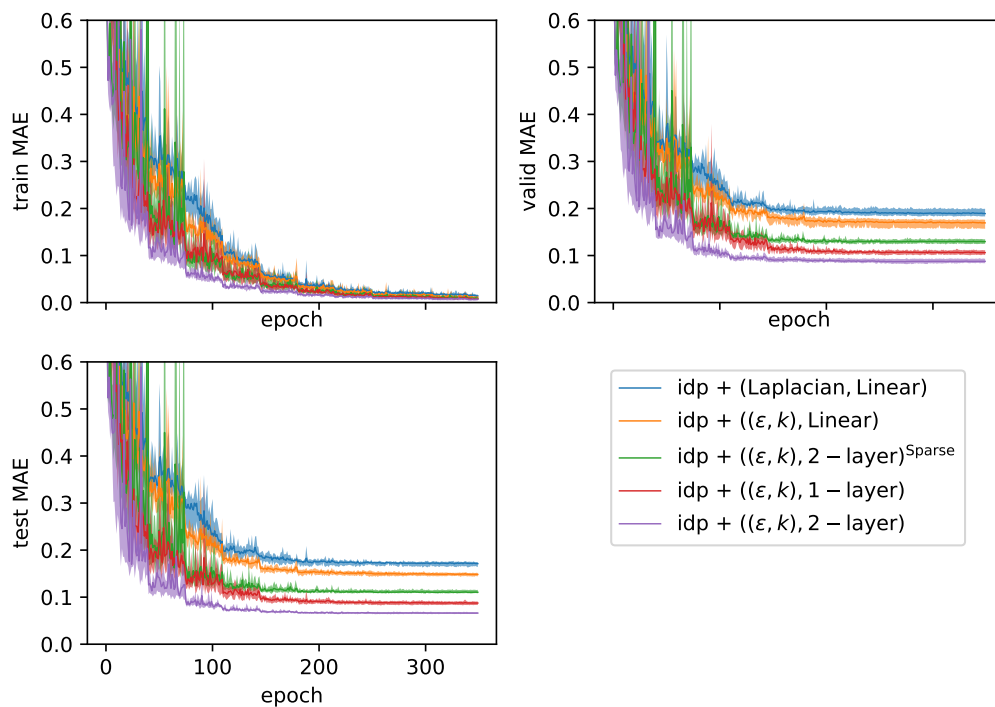*Figure 3.* Ablation studies with channel-shared architecture on ZINC.

Figure 4. Ablation studies with channel-independent architecture on ZINC.

Table 7. Hyperparameter settings on TUDataset.

Hyperparameter	MUTAG	NCI1	NCI109	ENZYMES	PTC_MR	PROTEINS	IMDB-B	RDT-B
Hidden Dim.	256	256	256	256	128	128	256	256
Num. Layers	4	6	6	6	6	6	3	4
Drop. Rate	0	0	0	0.2	0	0	0	0
Readout	Max	Max	Max	Max	Max	Mean	Max	Max
Batch Size	64	64	64	64	64	64	16	64
Initial LR	0.001	0.001	0.001	0.001	0.001	0.001	0.001	0.001
LR Dec. Steps	50	50	50	40	30	50	40	50
LR Dec. Rate	0.6	0.6	0.6	0.6	0.65	0.65	0.6	0.6
# Warm. Steps	0	0	0	0	0	0	0	0
Weight Dec.	0	0	0	0	0	0	0	0
$\mathcal{G}$	Dense	Dense	Dense	Dense	Dense	Dense	Dense	Sparse
σ in f_{θ}	GELU	GELU	GELU	GELU	GELU	GELU	GELU	GELU
$\{(\epsilon, k)\}$	(-0.2, 1), (-0.2, 2), (-0.2, 3), (-0.2, 4), (-0.2, 5), (-0.25, 1), (-0.25, 2), (-0.25, 3), (-0.25, 4), (-0.25, 5), (-0.3, 1), (-0.3, 2), (-0.3, 3), (-0.3, 4), (-0.3, 5)	(-0.2, 1), (-0.2, 2), (-0.2, 3), (-0.2, 4), (-0.2, 5), (-0.25, 1), (-0.25, 2), (-0.25, 3), (-0.25, 4), (-0.25, 5)	(-0.2, 1), (-0.2, 2), (-0.2, 3), (-0.2, 4), (-0.2, 5), (-0.25, 1), (-0.25, 2), (-0.25, 3), (-0.25, 4), (-0.25, 5)	(-0.2, 1), (-0.2, 2), (-0.2, 3), (-0.2, 4), (-0.2, 5), (-0.2, 6), (-0.25, 1), (-0.25, 2), (-0.25, 3), (-0.25, 4), (-0.25, 5), (-0.25, 6), (-0.3, 1), (-0.3, 2), (-0.3, 3), (-0.3, 4), (-0.3, 5), (-0.3, 6), (-0.35, 1), (-0.35, 2), (-0.35, 3), (-0.35, 4), (-0.35, 5), (-0.35, 6)	(-0.3, 1), (-0.3, 2), (-0.3, 3), (-0.3, 4), (-0.3, 5), (-0.3, 6), (-0.3, 7), (-0.3, 8), (-0.35, 1), (-0.35, 2), (-0.35, 3), (-0.35, 4), (-0.35, 5), (-0.35, 6), (-0.35, 7), (-0.35, 8), (-0.4, 1), (-0.4, 2), (-0.4, 3), (-0.4, 4), (-0.4, 5), (-0.4, 6), (-0.4, 7), (-0.4, 8)	(-0.3, 1), (-0.3, 2), (-0.3, 3), (-0.3, 4), (-0.35, 1), (-0.35, 2), (-0.35, 3), (-0.35, 4), (-0.4, 1), (-0.4, 2), (-0.4, 3), (-0.4, 4)	(-0.2, 1), (-0.2, 2), (-0.2, 3), (-0.25, 1), (-0.25, 2), (-0.25, 3), (-0.3, 1), (-0.3, 2), (-0.3, 3), (-0.3, 4), (-0.3, 5), (-0.35, 1), (-0.35, 2), (-0.35, 3), (-0.35, 4), (-0.35, 5), (-0.4, 1), (-0.4, 2), (-0.4, 3), (-0.4, 4), (-0.4, 5)	(-0.3, 1), (-0.3, 2), (-0.35, 1), (-0.35, 2), (-0.4, 1), (-0.4, 2

Table 8. Time and space complexities of existing methods and ours.

Method	Time Complexity	Memory Complexity
GCN (Kipf & Welling, 2017)	$O(l(md + nd^2))$	$O(m + l(nd + d^2))$
GIN (Xu et al., 2019b)	$O(l(md + nd^2))$	$O(m + l(nd + d^2))$
GAT (Veličković et al., 2018)	$O(lg(md + nd^2))$	$O(m + lg(nd + d^2))$
PNA (Corso et al., 2020)	$O(lg(md + nd^2))$	$O(m + lg(nd + d^2))$
ExpC (Yang et al., 2022b)	$O(lg(md + nd^2))$	$O(m + lg(nd + d^2))$
shd-PDF	$O(mk + l(md + nd^2))$	$O(km + l(nd + d^2))$
idp-PDF	$O(mkd + l(md + nd^2))$	$O(k(m + d) + l(nd + d^2))$

Table 9. Training and evaluation time on ZINC. mean $\pm$ std over 50 epochs (seconds).

	Model	Training (Train set)	Eval (Train set)	Eval (Val set)	Eval (Test set)
	GIN	3.016 ± 0.198	1.459 ± 0.083	0.531 ± 0.081	0.533 ± 0.077
shd-	PDF [1-hop]	3.539 ± 0.218	1.503 ± 0.078	0.426 ± 0.064	0.430 ± 0.056
	PDF	4.024 ± 0.211	1.769 ± 0.115	0.588 ± 0.086	0.583 ± 0.078
idp-	PDF [1-hop]	3.638 ± 0.184	1.599 ± 0.089	0.572 ± 0.076	0.574 ± 0.089
	PDF	4.050 ± 0.174	1.744 ± 0.086	0.622 ± 0.091	0.613 ± 0.071

Table 10. Training and evaluation time on RDT-B. mean $\pm$ std over 50 epochs (seconds).

Model	Training (Train set)	Eval (Train set)	Eval (Val set)
GIN	0.792 ± 0.009	0.335 ± 0.004	0.040 ± 0.002
shd-PDF [1-hop]	0.723 ± 0.005	0.301 ± 0.003	0.035 ± 0.001
idp-PDF [1-hop]	1.035 ± 0.006	0.407 ± 0.006	0.049 ± 0.002

Table 11. Ablation study settings on ZINC.

Hyperparameter	(Lap, Lin)	$((\epsilon, k), \text{Lin})$	$((\epsilon, k), 2\text{L})^{\text{sps}}$	$((\epsilon, k), 1\text{L})$	$((\epsilon, k), 2\text{L})$
Hidden Dim.			160		
Num. Layers			6		
Drop. Rate			0		
Readout			Mean		
Batch Size			64		
Initial LR			0.001		
LR Dec. Steps			35		
LR Dec. Rate			0.6		
# Warm. Steps			5		
Weight Dec.			5e-5		
$\mathcal{G}$	Dense	Dense	Sparse	Dense	Dense
σ in $f_{\boldsymbol{\theta}}$	Linear	Linear	GELU	GELU	GELU
$\{(\epsilon, k)\}$	(-0.5, 1), (-0.5, 2), (-0.5, 3), (-0.5, 4), (-0.5, 5)	(-0.1, 3), (-0.2, 3), (-0.3, 4), (-0.4, 4), (-0.5, 4)	(-0.1, 4), (-0.2, 4), (-0.3, 4), (-0.4, 4), (-0.5, 4)	(-0.1, 4), (-0.2, 4), (-0.3, 4), (-0.4, 4), (-0.5, 4)	(-0.1, 4), (-0.2, 4), (-0.3, 4), (-0.4, 4), (-0.5, 4)