A Comprehensive Benchmark on Spectral GNNs: The Impact on Efficiency, Memory, and Effectiveness

NINGYI LIAO, Nanyang Technological University, Singapore
HAOYU LIU, Nanyang Technological University, Singapore
ZULUN ZHU, Nanyang Technological University, Singapore
SIQIANG LUO*, Nanyang Technological University, Singapore
LAKS V.S. LAKSHMANAN, The University of British Columbia, Canada

With recent advancements in graph neural networks (GNNs), spectral GNNs have received increasing popularity by virtue of their ability to retrieve graph signals in the spectral domain. These models feature uniqueness in efficient computation as well as rich expressiveness, which stems from advanced management and profound understanding of graph data. However, few systematic studies have been conducted to assess spectral GNNs, particularly in benchmarking their efficiency, memory consumption, and effectiveness in a unified and fair manner. There is also a pressing need to select spectral models suitable for learning specific graph data and deploying them to massive web-scale graphs, which is currently constrained by the varied model designs and training settings.

In this work, we extensively benchmark spectral GNNs with a focus on the spectral perspective, demystifying them as spectral graph filters. We analyze and categorize 35 GNNs with 27 corresponding filters, spanning diverse formulations and utilizations of the graph data. Then, we implement the filters within a unified spectral-oriented framework with dedicated graph computations and efficient training schemes. In particular, our implementation enables the deployment of spectral GNNs over million-scale graphs and various tasks with comparable performance and less overhead. Thorough experiments are conducted on the graph filters with comprehensive metrics on effectiveness and efficiency, offering novel observations and practical guidelines that are only available from our evaluations across graph scales. Different from the prevailing belief, our benchmark reveals an intricate landscape regarding the effectiveness and efficiency of spectral graph filters, demonstrating the potential to achieve desirable performance through tailored spectral manipulation of graph data. Our code and evaluation is available at: https://github.com/gdmnl/Spectral-GNN-Benchmark.

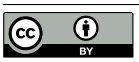
CCS Concepts: • **Mathematics of computing → Spectra of graphs**; *Approximation algorithms*; • **Computing methodologies → Neural networks**; *Spectral methods*; • **General and reference → Surveys and overviews**.

Additional Key Words and Phrases: Graph Neural Networks; Efficiency and Scalability; Graph Spectrum

ACM Reference Format:
Ningyi Liao, Haoyu Liu, Zulun Zhu, Siqiang Luo, and Laks V.S. Lakshmanan. 2025. A Comprehensive Benchmark on Spectral GNNs: The Impact on Efficiency, Memory, and Effectiveness. *Proc. ACM Manag. Data* 3, 4 (SIGMOD), Article 238 (September 2025), 29 pages. https://doi.org/10.1145/3749156

*Corresponding author.

Authors' Contact Information: Ningyi Liao, College of Computing and Data Science, Nanyang Technological University, Singapore, liao0090@e.ntu.edu.sg; Haoyu Liu, College of Computing and Data Science, Nanyang Technological University, Singapore, haoyu.liu@ntu.edu.sg; Zulun Zhu, College of Computing and Data Science, Nanyang Technological University, Singapore, zulun001@ntu.edu.sg; Siqiang Luo, College of Computing and Data Science, Nanyang Technological University, Singapore, siqiang.luo@ntu.edu.sg; Laks V.S. Lakshmanan, Department of Computer Science, The University of British Columbia, Canada, laks@cs.ubc.ca.



ACM 2836-6573/2025/9-ART238
https://doi.org/10.1145/3749156

1 Introduction

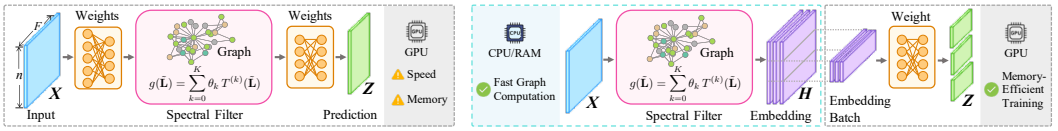
Graph neural networks (GNNs) have emerged by integrating neural network architectures with graph data processing techniques, serving as powerful tools for a wide range of graph understanding tasks [46, 93, 104, 115]. Among them, a notable branch is spectral GNNs, which are characterized by understanding the graph data through leveraging the graph spectrum, i.e., eigenvalues [5, 9, 18, 46]. In the graph, such spectral information is powerful for characterizing various patterns, from local edge connections to global clustering structures [33, 98]. Hence, spectral GNNs have garnered exceptional utility for real-world applications encompassing specialized graph topologies, such as multivariate time-series forecasting [10, 69] and point cloud analysis [43, 56].

Particularly, spectral models are superior in addressing the notorious scalability issue of common GNNs [86, 89, 109]. This merit stems from the unique graph data management as illustrated in Figure 1. Conventionally, learning GNNs is achieved through the full-batch scheme, where graph data and neural network weights are integrated, rendering iterative computation and high GPU overhead. In comparison, the mini-batch setup is uniquely available for spectral GNNs by dividing learning data into small batches, thereby alleviating both efficiency bottlenecks and memory footprints [22, 100]. The graph data and weights can be processed separately with tailored computations, enabling spectral GNNs to excel in learning web-scale graphs with hundreds of millions of nodes [28, 61, 78].

Challenges: Benchmarking Spectral GNNs at Scale. Previous surveys on spectral GNNs [7, 15] primarily focus on deriving spectral models and evaluating their accuracy on small datasets. Although the spectral design has been shown to be promising in scaling up GNNs, there is a lack of comprehensive study, especially on their efficiency and scalability performance. We note that implementing and evaluating these models for large-scale graphs remains a demanding task considering the following major challenges:

- **Under-explored practical performance on large-scale graphs.** For the interest of fast and useful GNNs in a broader scope, it is crucial to understand the performance of advanced solutions, extensively considering their time efficiency, memory overhead, effectiveness, and the relationships among these aspects. However, it remains unknown of such systematic insights regarding the feasibility and performance characteristics of spectral GNNs at various data scales, rendering it difficult to decide the appropriate model.
- **Limited availability of efficient and scalable computation techniques.** Most existing scalable computations based on spectral GNNs, such as graph operation accelerations [13, 25, 60, 95] and mini-batch training techniques [26, 40, 59] are exclusively available for only a few models. The majority of spectral GNNs have never been implemented or evaluated with these techniques, consequently constraining their practical application in large-scale scenarios.
- **Varied implementations and inconsistent evaluations.** Although a broad scope of models can be regarded as spectral GNNs, their computational designs vary greatly in terms of graph managements, network architectures, and training schemes. The distinct settings and implementations lead to varied time and memory overheads and complicate fair comparison among spectral models.

Our Contributions. In this research, we present a comprehensive benchmark on spectral GNNs, especially targeting the above challenges. We extensively review existing GNN designs with spectral interpretations, ranging from traditional to cutting-edge studies, and reach a total of 35 models. To facilitate fair and versatile evaluations, we encapsulate the *filters*, i.e., the specialized graph management process, among these models. A novel taxonomy (Table 1) is proposed to identify



(a) Full-batch: coupled performance bottlenecks **(b) Mini-batch: decoupled computation**

Fig. 1. Model composition and training pipeline of **(a)** full-batch scheme for common GNN training and inference, where model weights and graph processing are integrated; and **(b)** mini-batch learning, which is specialized for spectral GNNs with separated graph computations. Among these two settings, the graph operations remain equivalent and are extracted as spectral filters.

and categorize these filters into three categories, each exhibiting distinct spectral capabilities and performances.

We implement the spectral filters in a unified manner applicable to a variety of settings and tasks. Our framework embraces scalable graph processing techniques, especially the dedicated mini-batch training scheme, and allows for **widely applicable spectral operations on million-scale graphs**, which have not been achieved before. On top of our taxonomy and implementation, we perform evaluation on efficiency, memory consumption, and effectiveness of spectral filters. Our findings aim to provide new assessment criteria, insightful observations, and useful guidelines, which are of pragmatic interest for developing and deploying spectral GNNs.

Our Findings. Our benchmark evaluations cover an inclusive range of spectral GNN components, graph filter designs, and training schemes. The observations feature diverse graph properties and impact factors on the performance of spectral models, and reveal the intricate relationship between efficiency and effectiveness across different scales of graphs, which is **only available by our unified framework highlighting scalability**. In specific, we outline the novel findings from benchmarking GNNs at scale as follows:

- GNN efficiency bottleneck shifts across data scales. Graph operations only dominate time and memory overheads on graphs above the million scale. Additional variable weights further intensify the computation.
- Mini-batch training is capable of enhancing efficiency and scalability. Similarly, the improvement is only significant on larger graphs, where the graph computation overhead is considerable.
- The effectiveness and efficiency of spectral GNNs are not mutually exclusive, challenging the prevailing belief. With adequate configurations, simple filters can excel in high accuracy and fast computation on most graphs.
- The effectiveness of spectral filters primarily roots in their alignment with the graph information; that is, preserving useful patterns while attenuating noise. Contrarily, sophisticated designs usually benefit generality but do not guarantee improved accuracy.
- Our taxonomy can effectively characterize filter efficiency. As the approach to harmonizing effectiveness and efficiency in spectral GNNs, it is recommended to employ simple but suitable filters through a better understanding of the graph data.

2 Preliminaries

In this section, we introduce the concepts in spectral graph processing and GNNs, followed by configurations of filter operations, model architectures and learning pipelines. A comparative analysis with existing studies on GNN evaluations is provided to highlight the novelty of this benchmark.

Table 1. Taxonomy of spectral GNN filters with notable corresponding models included in this paper. "Param." represents learnable filter parameters acquired during model training, while "HP" represents tunable filter hyperparameters acquired through hyperparameter search. "Time" and "Memory" are time and space complexity for filter computation, respectively.

Type	Filter*	Filter Function $g(\tilde{\mathbf{L}})$	Param.	HP	Time	Memory	Model†
Fixed	Identity	$\mathbf{I}$	/	/	$O(KnF)$	$O(nF)$	MLP
	Linear	$2\mathbf{I}-\tilde{\mathbf{L}}$	/	/	$O(KmF)$	$O(nF)$	𝕀: GCN [46]
	Impulse	$(\mathbf{I}-\tilde{\mathbf{L}})^K$	/	/	$O(KmF)$	$O(nF)$	𝔻: SGC [102], gfNN [81], GZoom [19], GRAND+ [26]
	Monomial	$\frac{1}{K+1}\sum_{k=0}^{K}(\mathbf{I}-\tilde{\mathbf{L}})^k$	/	/	$O(KmF)$	$O(nF)$	𝔻: S^2GC [126], AGP [95], GRAND+ [26]
	PPR	$\sum_{k=0}^{K}\alpha(1-\alpha)^k(\mathbf{I}-\tilde{\mathbf{L}})^k$	/	α	$O(KmF)$	$O(nF)$	𝕀: GLP [53], GCNII [80]; 𝔻: APPNP [47], GDC [32], AGP [95], GRAND+ [26]
	HK	$\sum_{k=0}^{K}\frac{e^{-\alpha}\alpha^k}{k!}(\mathbf{I}-\tilde{\mathbf{L}})^k$	/	α	$O(KmF)$	$O(nF)$	𝔻: GDC [32], AGP [95], DGC [99]
	Gaussian	$\sum_{k=0}^{K}\frac{\alpha^k}{k!}(2\mathbf{I}-\tilde{\mathbf{L}})^k$	/	α	$O(KmF)$	$O(nF)$	𝔻: G^2CN [51]
Variable	Linear	$(1+\theta)\mathbf{I}-\tilde{\mathbf{L}}$	θ_k	/	$O(KmF)$	$O(nF)$	𝕀: GIN [104], AKGNN [45]
	Monomial	$\sum_{k=0}^{K}\theta_k(\mathbf{I}-\tilde{\mathbf{L}})^k$	θ_k	/	$O(KmF)$	$O(nF)$	𝔻: DAGNN [67], GPRGNN [17]
	Horner	$\sum_{k=0}^{K}\theta_k(\mathbf{I}-\tilde{\mathbf{L}})^k$	θ_k	/	$O(KmF)$	$O(2nF)$	𝕀: ARMAGNN [6], HornerGCN [35]
	Chebyshev	$\sum_{k=0}^{K}\theta_k T_{\text{Cheb}}^{(k)}(\tilde{\mathbf{L}})$	θ_k	/	$O(KmF)$	$O(2nF)$	𝕀: ChebNet [18]; 𝔻: ChebBase [40]
	ChebInterp	$\frac{2}{K+1}\sum_{k=0}^{K}\sum_{\kappa=0}^{K}\theta_\kappa T_{\text{Cheb}}^{(k)}(x_\kappa)T_{\text{Cheb}}^{(k)}(\tilde{\mathbf{L}})$	θ_k	/	$O(KmF+K^2nF)$	$O(2nF)$	𝔻: ChebNetII [40]
	Clenshaw	$\sum_{k=0}^{K}\theta_k T_{\text{Cheb2}}^{(k)}(\tilde{\mathbf{L}})$	θ_k	/	$O(KmF)$	$O(3nF)$	𝕀: ClenshawGCN [35]
	Bernstein	$\sum_{k=0}^{K}\frac{\theta_k}{2^k}\binom{K}{k}(2\mathbf{I}-\tilde{\mathbf{L}})^{K-k}\tilde{\mathbf{L}}^k$	θ_k	/	$O(K^2mF)$	$O(nF)$	𝔻: BernNet [39]
	Legendre	$\sum_{k=0}^{K}\theta_k\frac{(-1)^k}{k!\binom{2k}{k}}(2\mathbf{I}-\tilde{\mathbf{L}})^k\tilde{\mathbf{L}}^k$	θ_k	/	$O(KmF)$	$O(2nF)$	𝔻: LegendreNet [12]
	Jacobi	$\sum_{k=0}^{K}\theta_k T_{\text{Jacobi}}^{(k)}(\tilde{\mathbf{L}})$	θ_k	α,β	$O(KmF)$	$O(2nF)$	𝔻: JacobiConv [98]
	Favard	$\sum_{k=0}^{K}\theta_k T_{\text{Favard}}^{(k)}(\tilde{\mathbf{L}})$	θ_k	/	$O(KmF+KnF)$	$O(2nF)$	𝔻: FavardGNN [36]
	OptBasis	$\sum_{k=0}^{K}\theta_k T_{\text{OptBasis}}^{(k)}(\tilde{\mathbf{L}})$	θ_k	/	$O(KmF+KnF^2)$	$O(2nF)$	𝔻: OptBasisGNN [36]
Bank	Linear (channel-wise)	$\Vert_{q=1}^{Q}(\mathbf{I}-\gamma_q\tilde{\mathbf{L}})$	γ_q	/	$O(KmF)$	$O(nF)$	𝔻: AdaGNN [21]
	Linear (LP, HP)	$\gamma_1(\mathbf{I}-\tilde{\mathbf{L}})+\gamma_2\tilde{\mathbf{L}}$	γ_q	/	$O(QKmF+QKnF)$	$O(QnF)$	𝕀: FBGCN (I/II) [72]
	Linear (LP, HP, ID)	$\gamma_1(\mathbf{I}-\tilde{\mathbf{L}})+\gamma_2\tilde{\mathbf{L}}+\gamma_3\mathbf{I}$	γ_q	/	$O(QKmF+QKnF)$	$O(QnF)$	𝕀: ACMGNN (I/II) [71]
	Linear (LP, HP)	$\gamma_1((\beta+1)\mathbf{I}-\tilde{\mathbf{L}})+\gamma_2((\beta-1)\mathbf{I}+\tilde{\mathbf{L}})$	γ_q	β	$O(QKmF)$	$O(QnF)$	𝔻: FAGNN [8]
	Gaussian (LP, HP)	$\sum_{q=1}^{Q}\sum_{k=0}^{\lfloor K/2\rfloor}\frac{\alpha_q^k}{k!}((1+\beta_q)\mathbf{I}-\tilde{\mathbf{L}})^{2k}$	γ_q	α_q,β_q	$O(QKmF)$	$O(QnF)$	𝔻: G^2CN [51]
	PPR (LP, HP)	$\sum_{q=1}^{Q}\sum_{k=0}^{K}\alpha_q(1-\alpha_q)^k(\mathbf{I}+\beta_q\tilde{\mathbf{L}})(\mathbf{I}-\tilde{\mathbf{L}})^k$	γ_q	α_q,β_q	$O(QKmF)$	$O(QnF)$	𝔻: GNN-LF/HF [127]
	Mono, Cheb, Bern, ID	$\sum_{q=1}^{Q}\sum_{k=0}^{K}\gamma_q\theta_{q,k}T_q^{(k)}(\tilde{\mathbf{L}})$	$\gamma_q,\theta_{q,k}$	/	$O(QKmF)$	$O(QnF)$	𝔻: FiGURe [24]

* **Filter**: LP = Low-pass, HP = High-pass, ID = Identity. † **Model**: 𝕀 = Iterative architecture, 𝔻 = Decoupled architecture.

2.1 Formulation of Spectral GNN Paradigm

Graph Notation. An undirected graph is denoted as $\mathcal{G}=\langle\mathcal{V},\mathcal{E}\rangle$ with $n=|\mathcal{V}|$ nodes and $m=|\mathcal{E}|$ edges. The adjacency matrix without and with self-loops are $\mathbf{A}$ and $\bar{\mathbf{A}}=\mathbf{A}+\mathbf{I}$, respectively. The normalized adjacency matrix is $\tilde{\mathbf{A}}=\bar{\mathbf{D}}^{\rho-1}\bar{\mathbf{A}}\bar{\mathbf{D}}^{-\rho}$, where $\bar{\mathbf{D}}$ is the self-looped diagonal degree matrix and $\rho\in[0,1]$ is the normalization coefficient [95, 107]. The attributed graph is associated with a node attribute matrix $\mathbf{X}\in\mathbb{R}^{n\times F}$, carrying an F-dimensional attribute vector for each node.

Spectral Graph Theory. Spectral graph theory [20] relate the graph with signal processing techniques, which utilizes the graph Laplacian matrix $\mathbf{L}=\mathbf{D}-\mathbf{A}$ or its normalized counterpart $\tilde{\mathbf{L}}=\mathbf{I}-\tilde{\mathbf{A}}$. Graph Laplacian leads to its eigen-decomposition $\tilde{\mathbf{L}}=\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where the graph *spectrum* $\mathbf{\Lambda}=\text{diag}(\lambda_1,\cdots,\lambda_n)$ is the diagonal matrix of eigenvalues $0=\lambda_1\le\lambda_2\le\cdots\le\lambda_n\le 2$, and $\mathbf{U}$ is

the matrix consisting of eigenvectors. The eigenvalues are also known as the signal *frequencies*, where small and large values correspond to low and high frequencies in the graph, respectively.

The spectral graph property is highly related to the heterophily pattern, which depicts the spatial distribution of similar graph nodes and motivates a number of spectral GNN designs [70]. A *homophilous* graph implies that nodes with the same labels tend to connect to each other, while a *heterophilous* graph does not. Graphs of high heterophily usually signify high-frequency spectral signals. An exemplar measure of the heterophily property is the node homophily score [84]: $\mathcal{H} = \frac{1}{n}\sum_{u\in V}|\{v|v\in\mathcal{N}(u), y(v)=y(u)\}|/|\mathcal{N}(u)|$, where $y(v)$ is the label of node v. A score closer to 0 indicates higher homophily with more similar neighbors, and vice versa.

Spatial Graph Neural Network. General GNNs are based on the *spatial* interpretation of the graph structure $\tilde{\mathbf{A}}$, which is exploited to perform the *propagation* operation $f(\tilde{\mathbf{A}})$. The model aims to iteratively update the node-wise representation $\mathbf{H}$ through its j-th layer: $\mathbf{H}^{(j+1)} = \varphi(f^{(j)}(\tilde{\mathbf{A}})\cdot\mathbf{H}^{(j)})$, where φ denotes the *transformation* consisting of learnable weights and non-linear activations. The representation can be initialized by node attributes $\mathbf{H}^{(0)} = \mathbf{X}$.

Typically, one GNN layer corresponds to one hop of propagation through graph edges, and spatial GNNs stack J layers to receive more information from a larger field. The layer combination derives two variants of the GNN architecture, since the two types of operations can be either integrated or separated: An *iterative* model performs propagation f and transformation φ alternatively across its layers, while the *decoupled* architecture executes all propagations together, with weight transformations only applied on the propagation results. Despite the different combinations of operations, these two architectures are considered to posses the same expressiveness in propagating graph information [98, 117].

Spectral Graph Neural Network. Spectral GNNs view the graph information as a matrix to perform signal analysis techniques such as eigen-decomposition. Although it is based on the distinct interpretation with spatial GNNs, these two variants can be derived from each other, as revealed by previous research [15, 46]. To demonstrate, we start from the spatial GNN formulation by focusing on the graph-related components $f^{(j)}(\tilde{\mathbf{A}})$ in GNN and omitting the non-linear transformation, i.e., let $\varphi(\mathbf{x}) = \mathbf{x}$. We also utilize the graph Laplacian $\tilde{\mathbf{L}}$ instead of adjacency to denote graph information, and use the $F = 1$ attribute vector $\mathbf{x}$ as node-wise signal instead of the attribute matrix. In this case, the K-hop graph computation can be characterized by a polynomial $g(\tilde{\mathbf{L}})$:

$$g(\tilde{\mathbf{L}})\cdot\mathbf{x} = \sum_{k=0}^{K}\theta_k\, T^{(k)}(\tilde{\mathbf{L}})\,\mathbf{x}, \tag{1}$$

where $T^{(k)}(\tilde{\mathbf{L}})$ is the k-th term of the polynomial basis, and θ_k is the corresponding parameter. Combination of these two components determines the actual management of graph signals [98].

In spectral domain, a *filter* $\hat{g}:[0,2]\to\mathbb{R}$ processes the input *signal* into *response* by amplifying or attenuating certain frequencies. Denote the filtering operation on signal $\mathbf{x}$ as $\hat{g}*\mathbf{x}$, its relationship with spatial operations is given by the Fourier transform:

$$\hat{g}*\mathbf{x} = \mathbf{U}\hat{g}(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{x} = \sum_{i=0}^{n}\hat{g}(\lambda_i)\mathbf{u}_i\mathbf{u}_i^\top\mathbf{x}. \tag{2}$$

Eq. (2) can be understood by three consecutive stages: the Fourier transform $\mathbf{y}_1 = \mathbf{U}^\top\mathbf{x}$ which transfers the input signal to the spectral domain; the modulation $\mathbf{y}_2 = \hat{g}(\mathbf{\Lambda})\mathbf{y}_1$ which applies graph information through the filter; and lastly the inverse Fourier transform $\mathbf{x}^\star = \mathbf{U}\mathbf{y}_2$ which transfers the output back from the spectral domain.

Eq. (2) implies that the spectral filtering operation can be achieved by iterative spatial multiplications with the signal. Therefore, Eq. (1) provides a feasible approximation by only considering

$K \ll n$ orders of those relatively significant graph spectrum. In practice, this polynomial form dominates the actual implementations in spectral GNNs, bypassing the prohibitive eigen-decomposition through the well-established propagation based on graph adjacency. As a consequence, spectral and spatial GNNs share the same elementary operations, rendering the possibility to derive and evaluate a large family of GNNs under the interpretation of spectral filters.

2.2 GNN Learning Workflow

Complexity Analysis. As indicated by Eq. (2), spectral and spatial GNNs share the common elementary operations. *Propagation* describes the process of applying graph matrix to the node representation. Each application of the basis $T^{(k)}(\tilde{\mathbf{L}})$ to the graph signal in Eq. (1) is regarded as one propagation step. The operation is conducted by the matrix multiplication between the $n \times n$ sparse graph matrix and the $n \times F$ dense representation, resulting in $O(mF)$ time. The memory overhead of graph matrix is $O(m)$. The *transformation* operation updates the representation with learnable weights. Typical transformations include scalar and matrix multiplication, corresponding to time complexity $O(nF)$ and $O(nF^2)$, respectively. Employing a representation matrix to be iteratively updated for all nodes throughout learning consumes $O(nF)$ memory space, while advanced transformation architectures such as residual connection and concatenation cause time and memory expense to rise in accordance.

Learning Schemes. The learning scheme describes how the model and data are deployed to CPU and GPU for training and inference. *Full-batch* (FB) training loads all input data onto the GPU, which suffices the need on small-scale graphs and is the de facto scheme for most GNNs. However, for larger graphs, the critical scalability issue emerges, as the full graph topology exceeds the GPU memory capacity. To mitigate the overhead, a common model-agnostic solution is to employ *graph partition* (GP) algorithms to divide and train the graph data in batches. However, this process impacts the graph topology and undermines GNN expressiveness [22, 75].

In contrast, the decoupled *mini-batch* (MB) scheme is uniquely available for the spectral paradigm. It chooses to perform graph-related operations to generate one or more node-wise representation matrices in a precomputation stage on CPU. Then, only the intermediate representations are loaded onto the GPU in batches during training, which prevents the memory footprint from being coupled with the graph size and enjoys better scalability. As showcased in Figure 1, the spectral filter $g(\tilde{\mathbf{L}})$ remains invariant across different model architectures and training schemes with varied efficiency and scalability scenarios. This invariance allows for a shared formulation and implementation across these scenarios, which can be further enhanced by dedicated graph data management techniques.

2.3 Related Works

GNNs Surveys and Benchmarks. The broad area of graph neural networks has been extensively reviewed from different aspects, while the spectral paradigm is a relatively unprecedented topic. We respectively discuss the related benchmark studies:

- Surveys for *general* GNNs examine the algorithmic designs including architectural selections, convolutional and sequential operations, and overall graph learning pipelines [103, 120, 125]. Software frameworks [27, 97] and datasets [42, 63, 85] have also been developed. Although a few classic spectral models are covered, they are not distinguished from other spatial models.
- The GNN *efficiency* is especially surveyed in [2, 68, 116] and evaluated in [22, 23, 75, 77], focusing on the empirical performance of algorithmic designs in GNN architectures. However, most of these techniques are limited to spatial interpretations, which are orthogonal to the spectral perspective in our work.
- Two recent surveys target *spectral* GNNs: [15] derives the general connection between typical spatial and spectral operators, advancing the unified interpretation. Our definition of spectral

Table 2. Review of GNN frameworks and benchmarks literature with empirical evaluations. "Scheme": available learning schemes. "Eff.": whether efficiency is evaluated. "Scale": largest dataset used in terms of the number of nodes.

Type	Study	Spectral Models	Scheme*	Eff.	Scale
General	PyG [27]	GCN, ChebNet, SGC, APPNP, ARMA	FB & GP	✗	/
	Platonov et al. [85]	GCN, CPGNN, FAGCN, GPRGNN, JacobiConv	FB	✗	49K
	OGB [42]	GCN, SGC	FB	✗	100M
	LINKX [63]	GCN, SGC, APPNP, GPRGNN	FB & MB	✗	3M
Efficiency	Maekawa et al. [77]	GCN, ChebNet, SGC, GPRGNN	FB	✓	15K
	Dwivedi et al. [23]	GCN	FB & GP	✓	235K
	Duan et al. [22]	SGC	FB & MB	✓	3M
	SGL [75]	SGC, S^2GC, APPNP	FB & MB	✓	3M
Spectral	GAMMA-Spec [7]	14	FB	✗	20K
	Ours	35	FB & MB	✓	3M

* **Scheme**: FB = Full-Batch, GP = Graph Partition, MB = Mini-Batch.

GNNs is primarily based on this work. [7] reviews 14 spectral models identified by the eigen-decomposition. However, both frameworks contain computation-intensive operations, which are prohibitive for large graphs. We further elaborate on the derivation between our taxonomy and these works in our technical report [1].

Table 2 presents the literature containing empirical evaluations. Overall, we note a lack of research encompassing a broad range of up-to-date spectral GNNs, particularly in benchmarking graph operations and scalable implementations. This study hence contributes by offering efficiency comparisons, performance on large-scale datasets, and a comprehensive coverage of spectral filters.

Efficient GNN Computation. To mitigate the computational bottleneck, a series of studies [25, 30, 57, 60, 61, 64, 123] accelerate graph propagation in specific existing *spectral* filters using dedicated approximate algorithms. *Spatial* techniques such as sparsification [14, 49, 62, 101], sampling [73, 91, 110], and compression [76, 124] are also widely explored to reduce the overhead of model operations. Although we do not directly evaluate these works on compute optimization, some efficient data processing techniques from them have been incorporated into our pipeline.

GNN Training Systems. Instead of tailoring particular models, system-level research aims to enhance efficiency and scalability through data management techniques on graph topology [3, 55, 94, 114, 128] as well as structured embeddings [44, 79, 83, 90]. Additionally, a line of studies advances GNNs training in distributed environments [29, 34, 65, 118, 122]. These systems typically partition the graph for data parallelism, which differs from our straightforward mini-batch scheme in Section 2.2. We refer readers to [89, 112] for more comprehensive reviews on this topic.

3 Taxonomy of Spectral Filters

In this study, we propose a novel taxonomy for spectral GNNs based on their bases $T(\tilde{\mathbf{L}})$ and parameters θ, and categorize them into three types in Table 1. We respectively introduce the filter types and their representative models, while detailed derivations of all filters within each type listed in Table 1 are provided in our technical report [1].

Selection and Naming of Filters. We extensively survey existing GNN designs that can be represented in the polynomial form Eq. (1). In specific, we focus on works encompassing graph data and propagation operations, which signify the spectral formulations $g(\tilde{\mathbf{L}};\theta)$ of the filter. The

naming of the filters generally follows the cited original works in Table 1, since most of the GNN models are inspired by traditional polynomial functions or signal processing filters. Exceptional cases include the novel filters ChebInterp and OptBasis as well as filter bank models, which are named after the respective works in Table 1.

3.1 Fixed Filter GNN

For the first type of spectral GNNs, the basis and parameters are both constant during learning, resulting in fixed filters $g(\tilde{\mathbf{L}})$. In practice, these filters are usually well-studied graph data processing schemes such as monomial summation or PPR propagation. Below we give an example of the classic APPNP model and its filter:

Personalized PageRank (PPR). APPNP [47] is a pioneering decoupled GNN that iteratively applies a decaying graph propagation $\mathbf{H}^{(j+1)} = \varphi((1-\alpha)\tilde{\mathbf{A}}\mathbf{H}^{(j)} + \alpha\mathbf{H}^{(0)})$, where $\alpha \in [0,1]$ denotes the decay coefficient. It is revealed to be equivalent to the well-studied PPR calculation [66, 82], i.e., iteratively multiplying the signal with $\tilde{\mathbf{A}}^k$, and accumulating with a decaying coefficient $\alpha(1-\alpha)^k$. Recall that $\tilde{\mathbf{L}} = \mathbf{I} - \tilde{\mathbf{A}}$, the spectral basis for each iteration can be written as $T^{(k)}(\tilde{\mathbf{L}}) = (\mathbf{I} - \tilde{\mathbf{L}})^k$, and the corresponding parameter is $\theta_k = \alpha(1-\alpha)^k$. We name it as the PPR filter:

$$g(\tilde{\mathbf{L}}) = \sum_{k=0}^{K} \alpha(1-\alpha)^k (\mathbf{I} - \tilde{\mathbf{L}})^k, \quad \theta_k = \alpha(1-\alpha)^k.$$

Regarding computational complexity, calculating the K-order filter requires $O(KmF)$ time for the graph computation and $O(nF)$ space for storing the representation matrix.

3.2 Variable Filter GNN

The second type in our taxonomy also features a predetermined basis $T^{(k)}(\tilde{\mathbf{L}})$ in Eq. (1). The series of parameters θ_k is variable, usually acquired through gradient descent during model training, leading to a variable filter denoted as $g(\tilde{\mathbf{L}}; \theta)$. Compared to spectral GNNs with fixed filters, these models are claimed to enjoy better capability, especially for fitting high-frequency signals and capturing non-local topology. Below we present the Chebyshev filter:

Chebyshev. ChebNet [18] is a representative spectral GNN that explicitly utilizes the classical Chebyshev polynomial [38] as basis, which possesses orthogonal terms with straightforward computation. Each term $T^{(k)}(\tilde{\mathbf{L}})$ is expressed in the recursive form based on the computation results $T^{(k-1)}(\tilde{\mathbf{L}})$ and $T^{(k-2)}(\tilde{\mathbf{L}})$ of previous hops. The Chebyshev filter can be formulated as:

$$g(\tilde{\mathbf{L}}; \theta) = \sum_{k=0}^{K} \theta_k T^{(k)}(\tilde{\mathbf{L}}), \quad T^{(k)}(\tilde{\mathbf{L}}) = 2\tilde{\mathbf{L}}T^{(k-1)}(\tilde{\mathbf{L}}) - T^{(k-2)}(\tilde{\mathbf{L}}),\ T^{(1)}(\tilde{\mathbf{L}}) = \tilde{\mathbf{L}},\ T^{(0)}(\tilde{\mathbf{L}}) = \mathbf{I}.$$

The computation scheme of ChebNet can be directly inferred from the filter formulation. Initially, it employs $\mathbf{X}$ and $\tilde{\mathbf{L}}\mathbf{X}$ as the 0- and 1-order terms. Then, the recursive equation is followed to apply $T^{(k)}(\tilde{\mathbf{L}})$ and results are accumulated by multiplying θ_k. Its complexity for performing graph operations is $O(KmF)$ We denote its memory footprint as $O(2nF)$ for maintaining two $n \times F$ matrices during the iterative filter computation.

3.3 Filter Bank GNN

Some studies argue that a single filter is limited in leveraging complex graph signals. Hence, spectral GNNs with filter bank arise as an advantageous approach to provide abundant information for learning. In this study, we innovatively incorporate these models into the spectral GNN taxonomy

as a mixture of Q fixed or variable filters $g_q(\tilde{\mathbf{L}};\theta)$, each assigned a filter weight parameter γ_q:

$$\mathcal{g}(\tilde{\mathbf{L}};\gamma,\theta)=\bigoplus_{q=1}^{Q}\gamma_q\cdot g_q(\tilde{\mathbf{L}};\theta),\quad g_q(\tilde{\mathbf{L}};\theta)=\sum_{k=0}^{K}\theta_{q,k}\,T_q^{(k)}(\tilde{\mathbf{L}}),\tag{3}$$

where $\bigoplus$ denotes an arbitrary fusion function such as summation or concatenation. By this means, the filter bank is able to cover different *channels*, i.e., frequency ranges, to generate more comprehensive embeddings of the graph. The filter parameter γ can be either learned separately or along with GNN training, depending on the specific model implementation.

FiGURe. FiGURe [24] is an exemplar filter bank GNN following exactly Eq. (3) with summation fusion $\mathcal{g}(\tilde{\mathbf{L}};\gamma,\theta)=\sum_{q=1}^{Q}\gamma_q\cdot g_q(\tilde{\mathbf{L}};\theta)$, where filter-level parameters γ_q are learned to control the channel strength. Common variable filters including Monomial, Chebyshev, and Bernstein bases are utilized to compose the filter bank. With a straightforward implementation, the time and memory complexity of the combination are Q times that of an individual filter.

4 Benchmark Design

Framework Implementation. Based on the paradigm of spectral GNNs, we develop a unified framework oriented toward the spectral graph filters in particular, which is applicable to multiple GNN learning schemes. Our implementation embraces the modular design principle, offering similar framework arrangement and plug-and-play filter modules that can be seamlessly integrated into the popular graph learning toolkit PyG [27]. It is also easily extendable to new filter designs, as only the spectral formulation in the form of Eq. (1) needs to be implemented, while other model architectures and learning configurations can be effectively reused. Additionally, we include our reproducible benchmark pipeline for training and evaluating the models, with a particular focus on spectral analysis as well as scalable mini-batch training.

Table 3. Dataset statistics. $\mathcal{H}$, F_i, and F_o are homophily score, input attribute dimension, and number of class labels, respectively.

Category		Dataset	Nodes n	Edges m	$\mathcal{H}$	F_i	F_o	Metric
Small	Homophily	CORA [88]	2,708	10,556	0.83	1,433	7	Accuracy
		CITESEER [88]	3,327	9,104	0.72	3,703	6	Accuracy
		PUBMED [88]	19,717	88,648	0.79	500	3	Accuracy
		MINESWEEPER [85]	10,000	78,804	0.68	7	2	ROC AUC
		QUESTIONS [85]	48,921	307,080	0.90	301	2	ROC AUC
		TOLOKERS [85]	11,758	1,038,000	0.63	10	2	ROC AUC
	Heterophily	CHAMELEON [84]	890	17,708	0.24	2,325	5	Accuracy
		SQUIRREL [84]	2,223	93,996	0.19	2,089	5	Accuracy
		ACTOR [84]	7,600	30,019	0.22	932	5	Accuracy
		ROMAN [85]	22,662	65,854	0.05	300	18	Accuracy
		RATINGS [85]	24,492	186,100	0.38	300	5	Accuracy
Medium	Homophily	FLICKR [113]	89,250	899,756	0.32	500	7	Accuracy
		OGBN-ARXIV [42]	169,343	1,166,243	0.63	128	40	Accuracy
	Heterophily	ARXIV-YEAR [63]	169,343	1,166,243	0.31	128	5	Accuracy
		PENN94 [63]	41,554	2,724,458	0.48	4,814	2	Accuracy
		GENIUS [63]	421,961	984,979	0.08	12	2	ROC AUC
		TWITCH-GAMER [63]	168,114	6,797,557	0.10	7	2	Accuracy
Large	Homophily	OGBN-MAG [42]	736,389	5,416,271	0.31	128	349	Accuracy
		OGBN-PRODUCTS [42]	2,449,029	123,718,280	0.83	100	47	Accuracy
	Heterophily	POKEC [63]	1,632,803	30,622,564	0.43	65	2	Accuracy
		SNAP-PATENTS [63]	2,923,922	13,972,555	0.22	269	5	Accuracy
		WIKI [63]	1,925,342	303,434,860	0.28	600	5	Accuracy

Table 4. Configuration search scheme. Hyperparameters are explored based on the combination of listed ranges, and underlined values are the universal settings used across main experiments.

Scheme	Hyperparameter	Full-batch	Mini-batch
Universal	Propagation hop K	$\{2, 4, 6, 8, \underline{10}, 12, \cdots, 30\}$	$\{2, 4, 6, 8, \underline{10}, 12, \cdots, 30\}$
	Hidden width F	$\{16, 32, 64, \underline{128}, 256\}$	$\{16, 32, 64, \underline{128}, 256\}$
	Layer of linear φ_0	$\{\underline{1}, 2, 3\}$	$\{\underline{0}\}$
	Layer of linear φ_1	$\{\underline{1}, 2, 3\}$	$\{1, \underline{2}, 3\}$
Individual	Graph normalization ρ	$[0, 1]$	$[0, 1]$
	Learning rate of φ_0, φ_1	$[10^{-5}, 0.5]$	$[10^{-5}, 0.5]$
	Learning rate of θ, γ	$[10^{-5}, 0.5]$	$[10^{-5}, 0.5]$
	Weight decay of φ_0, φ_1	$[10^{-7}, 10^{-3}]$	$[10^{-7}, 10^{-3}]$
	Weight decay of θ, γ	$[10^{-7}, 10^{-3}]$	$[10^{-7}, 10^{-3}]$

Tasks and Metrics. In the main experiment, we focus on semi-supervised learning for node-classification task on a single graph, while other tasks are also analyzed in Section 6. Particularly, the evaluation features some of the largest datasets available for graph learning, rendering it the mainstream task for spectral GNNs and is ideal for assessing scalability. We follow the dataset settings for using *efficacy* metrics including accuracy and ROC AUC. *Efficiency* with respect to both running speed and CPU/GPU memory footprint is separately measured for each existing learning stage, including precomputation, training, inference. We conduct 10 runs with different random seeds by default and report the average metrics with standard deviation. Evaluations are conducted on a single machine with 32 Intel Xeon CPUs (2.4GHz), an Nvidia A30 GPU (24GB memory), and 512GB RAM.

Datasets. We comprehensively involve 22 datasets that are widely used in GNN node classification detailed in Table 3. In the table, we incorporate self-loop edges and count undirected edges twice to better reflect the graph computation overhead. We mainly categorize the datasets from two aspects. In efficacy evaluations, we separately investigate *homophilous* and *heterophilous* datasets considering they exhibit different patterns of graph signals. Based on the graph scale, we alternatively distinguish them into *small*, *medium*, and *large* categories. We focus on medium and large datasets for efficiency analysis, with large graphs identified by empirical evaluation that at least some models are prohibitive due to the critical scalability issue. We follow the unified dataset processing protocol for common node classification task including edge direction, degree normalization, and feature transformation in line with the original literature. Random 60%/20%/20% splits for train/validation/test sets are used for graphs without predefined splits.

Model Architectures and Hyperparameters. Our selection of model architectures and hyperparameters intends to provide fair and comparable results of both effectiveness and efficiency while minimizing the effect of non-spectral aspects. Hence, we focus on the decoupled architecture with a same number of transformation layers. Referring to Table 1, the number of hops K and hidden dimension F are the critical model-level hyperparameters affecting computational complexity. A *universal* choice is therefore searched and then fixed for experiments across all models and datasets: the number of training epochs is kept constant at 500, while batch sizes are set to 4,096 and 200,000 for mini-batch learning on small/medium- and large-scale datasets, respectively. For the other hyperparameters, including filter-level ones in Table 1, we perform *individual* tuning for each model and dataset to pursue satisfying accuracy. The ranges and used values of hyperparameter search are listed in Table 4.

5 Results: Efficiency and Effectiveness

In this section, we present primary research questions (RQs) and analysis on the performance comparison among a total of 27 filters across three types, benchmarking their effectiveness, efficiency, and memory overhead under different datasets and training schemes. Full experiments and results can be found in our technical report [1].

5.1 Efficiency

Thanks to our unified implementation framework, the time and memory efficiency of spectral GNNs can be compared across different graph scales under fair criteria. Figure 2 provides a detailed breakdown of full- and mini-batch schemes, separating learning stages and devices for the time and memory evaluations, respectively. We respectively assess the performance from multiple perspectives of filter operations and learning schemes.

5.1.1 Model Operations.

RQ1: *What are the efficiency bottlenecks of filters under different graph scales?*

As analyzed in Section 2.2, the key operations of graph propagation and weight transformation exhibit diverse computational overheads. Overall, it can be observed from Figure 2 that filter efficiency can be effectively characterized by our taxonomy Table 1. The three types of filters exhibit distinct performances, and the empirical time and memory efficiency are in line with our complexity analysis.

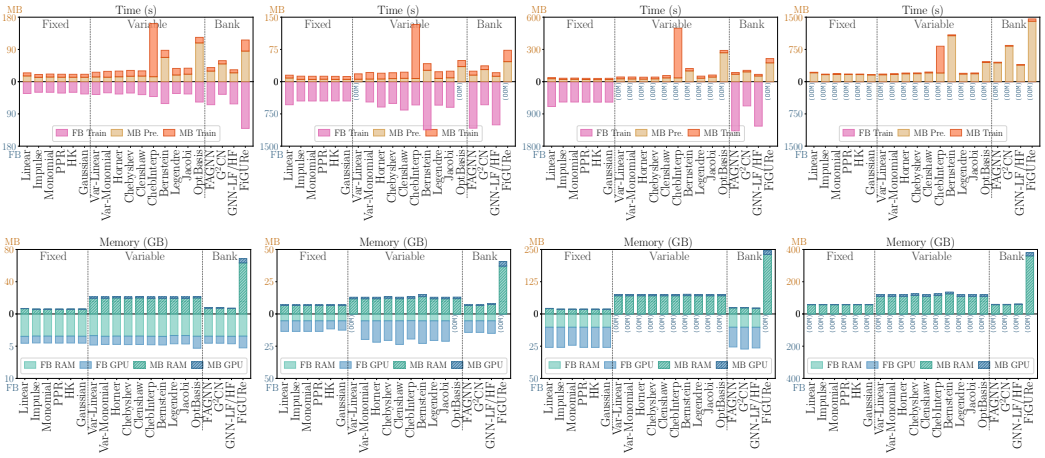
Regarding specific operations, **variable *transformations* come at the cost of additional computational resources**. In the FB scheme, this leads to reduced speed and increased GPU memory footprint. While some simple variable filters (Monomial, Chebyshev) and filter banks consisting of fixed filters (G^2CN, GNN-LF/HF) attain favorable performance when the graph scale is relatively small, models with complex transformations (Favard, OptBasis) experience prolonged training times and poor GPU memory scalability, and even result in out-of-memory (OOM) errors on large graphs. For MB training, the cost is reflected in increased RAM usage. Compared to fixed filters, memory footprint of variable designs is K times larger for separately storing results in each hop. Similarly, a filter bank design with Q variable filters demands up to Q times more memory.

On the other hand, ***propagation* becomes more computationally intensive on larger graphs**. In Figure 2, network transformation represented by MB training entails a large portion of time overhead on medium-scale graphs (PENN94), while propagation in precomputation dominates the MB training on larger graphs (POKEC, SNAP). Additional propagation further increases learning time but barely affects memory scalability, which is evident from variable filters with more graph-related propagations (ChebInterp, Bernstein). Nonetheless, thanks to our scalable implementation, their memory footprints are on par with other variable filters and are applicable to large-scale data even in memory-constrained environments.

5.1.2 Learning Schemes.

RQ2: *What are the efficiency and scalability improvements brought by mini-batch training?*

Figure 2 implies that filter *scalability*, i.e., whether filters can be applied to larger graphs, is mainly constrained by GPU memory capacity. With FB, models typically encounter the OOM issue on datasets larger than OGBN-MAG and POKEC due to the representation storage proportional to node size n. In comparison, **MB training benefits scalability by shifting the memory overhead from GPU to RAM**, enabling the employment of memory-intensive filters (Favard, OptBasis, FiGURe) on POKEC and even larger graphs. To the best of our knowledge, our implementation is the first attempt



(a) PENN94 **(b) POKEC** **(c) SNAP** **(d) WIKI**

Fig. 2. Comparison of filter time and memory efficiency under mini-batch (MB, upper axis) and full-batch (FB, lower axis) training on four medium- to large-scale datasets. Note that FB and MB axes may be on different scales. Empty bars are models with OOM error.

to successfully scale up an array of advanced spectral GNNs to these million-scale datasets and comprehensively characterize the performance.

Another advantage of MB lie in *time efficiency*. Even though FB for some filters is applicable to some million-scale graphs (POKEC, SNAP), the training time is excessively long. **MB achieves a significant** $10-50\times$ **speedup** by decoupling the computation-intensive propagation operations and simplifying the iterative training. Moreover, **the speedup is mainly observable on large-scale datasets**, while FB and MB present the same level of overall time on medium-sized graphs (PENN94). This can be explained by RQ1, as the increase in efficiency stems from faster propagation and is only effective when propagation is the bottleneck.

By summarizing RQ1&2, we conclude our **guidelines for employing mini-batch training**: Firstly, MB is particularly suitable for filters without variable parameters, showcasing faster computation and a smaller memory footprint thanks to better device utilization. Another ideal use case arises when FB for complex filters is prohibitive due to memory constraints. MB is capable of conducting graph filtering operations on CPU and storing full-scale representations in RAM at the cost of increased CPU/RAM usage. Even so, since GPU memory often has stricter constraints on practical platforms, MB with GPU memory complexity independent of the graph scale is favorable for deploying spectral GNNs across a wider range of environments.

5.2 Effectiveness

Our benchmark specifically features the filter effectiveness from the varied ancillary settings, aiming to demystify their capability in processing graph signals with fair comparison. Table 5 displays the filter accuracy with decoupled architecture and full-batch training. In particular, Identity represents the baseline without graph information. Our results align with previous evaluations involving spectral GNNs such as [7, 63, 85] and are mostly comparable to those in the original papers listed in Table 1.

5.2.1 Filter-level Comparison. As an overview, our results suggest that filter efficacy depends on graph patterns, and no single filter consistently achieves dominant accuracy across all scenarios.

Table 5. Effectiveness results (%) and standard deviations of spectral filters with full-batch training on available datasets. For each dataset, results are highlighted based on the relative effectiveness among filters, where green results are better.

Type	Filter	CORA	CITESEER	PUBMED	MINE	QUESTIONS	TOLOKERS	FLICKR	ARXIV	MAG	CHAM.	SQUIRREL	ACTOR	ROMAN	RATINGS	YEAR	PENN94	GENIUS	TWITCH	POKEC
Fixed	Identity	72.37±2.02	72.24±1.51	87.74±0.62	50.52±2.42	71.29±1.81	72.30±1.09	46.83±0.30	53.46±0.49	25.43±1.21	31.60±2.28	29.18±6.07	35.32±1.31	64.64±0.72	41.87±0.54	35.14±0.03	74.22±0.73	86.21±0.11	97.73±1.45	62.21±0.07
	Linear	86.14±1.15	74.07±1.91	85.15±0.75	68.50±1.49	71.47±0.70	72.74±1.55	50.55±0.46	65.56±1.86	31.52±0.97	40.66±2.70	36.43±2.42	26.88±1.43	37.60±0.68	42.17±0.42	44.52±2.04	63.57±9.48	85.71±3.76	96.93±0.06	51.59±1.54
	Impulse	84.43±0.82	74.24±2.19	83.38±0.54	64.06±1.61	69.24±1.37	56.75±2.13	49.92±0.23	63.84±1.58	29.32±1.57	37.57±3.71	35.48±2.12	25.67±1.13	27.37±3.27	31.67±3.94	40.68±0.46	51.00±2.85	88.02±0.15	96.93±0.06	51.40±0.81
	Monomial	86.28±1.72	75.21±1.62	89.07±0.49	70.21±1.46	66.61±6.57	76.51±1.36	50.27±0.73	59.41±0.55	32.41±0.57	39.82±3.38	38.31±2.64	37.84±1.65	63.44±1.48	44.57±1.12	44.74±2.55	75.27±0.29	88.30±0.07	96.99±0.10	60.06±0.22
	PPR	86.58±1.96	76.09±2.30	89.12±0.39	67.97±1.95	64.19±3.85	76.82±2.37	49.45±0.72	65.14±1.06	26.18±0.96	38.20±3.82	35.73±1.45	36.77±1.29	63.78±1.72	39.38±0.49	41.21±0.48	74.14±0.24	88.31±0.16	97.15±0.35	59.91±0.22
	HK	86.02±1.53	74.62±1.96	89.15±0.66	72.95±1.15	70.71±1.64	76.72±1.91	47.15±0.06	69.26±0.37	23.53±1.04	40.10±4.77	37.89±2.27	37.57±1.04	65.08±1.08	41.40±0.91	44.62±0.98	74.59±0.34	88.31±0.21	96.94±0.08	62.10±1.09
	Gaussian	86.44±1.02	75.36±1.64	88.97±0.73	72.61±1.09	66.25±3.71	78.32±1.69	46.67±0.30	63.72±0.83	27.87±3.45	36.38±4.57	36.35±1.36	37.35±1.22	64.04±0.90	41.68±0.40	46.11±1.94	74.64±0.17	87.93±0.21	97.63±0.06	59.95±0.06
Variable	Linear	85.72±1.46	75.17±1.95	85.50±0.69	72.41±1.21	68.97±3.02	72.04±6.40	50.53±0.10	63.69±1.22	31.11±0.60	40.66±3.57	37.61±2.27	26.81±1.71	31.16±0.86	43.83±1.36	46.58±0.63	70.81±0.09	88.42±0.11	97.18±0.49	(OOM)
	Monomial	86.85±0.65	76.18±1.86	89.05±0.77	89.26±0.80	65.29±4.63	78.95±0.72	52.23±1.30	65.47±1.37	32.03±1.38	40.10±3.84	36.94±1.97	34.95±1.64	64.16±1.68	39.50±1.38	46.19±0.50	75.23±0.99	88.42±0.06	96.91±0.07	59.10±1.49
	Horner	86.39±2.06	75.44±1.83	89.71±0.48	88.53±1.06	64.01±1.32	77.40±0.69	52.94±0.46	62.94±2.40	29.71±0.55	40.59±3.09	33.74±1.24	37.67±1.43	58.27±2.38	41.82±1.81	48.43±0.48	74.56±0.39	88.43±0.99	96.93±0.06	72.02±0.82
	Chebyshev	85.95±1.13	75.38±2.13	89.14±0.61	89.92±0.78	62.65±4.21	78.85±0.83	55.11±0.14	66.31±0.61	30.90±1.42	33.71±3.30	37.11±2.39	35.76±1.03	67.92±2.22	39.70±1.17	42.91±1.42	84.31±0.27	87.47±0.03	96.93±0.06	54.26±0.43
	Clenshaw	83.90±1.98	73.63±1.83	88.49±2.85	88.70±1.33	62.49±5.24	79.66±0.54	52.46±0.38	47.25±10.56	24.06±5.66	30.62±2.91	34.66±0.56	36.11±1.95	70.37±3.27	40.44±3.20	43.72±2.00	82.55±0.33	87.40±0.12	96.98±0.04	53.02±0.29
	ChebInterp	83.52±5.59	66.96±6.76	89.72±0.87	89.43±0.86	68.15±5.93	80.27±0.91	54.97±2.55	62.81±1.84	23.25±3.75	30.41±4.08	35.59±1.74	35.78±2.29	69.38±1.98	36.70±0.14	46.50±0.06	82.26±1.37	79.65±7.86	96.93±0.07	69.15±0.10
	Bernstein	71.03±2.49	62.21±2.39	84.24±0.97	87.38±1.12	62.51±2.89	76.36±1.24	47.95±0.56	48.86±0.73	13.83±3.31	34.97±3.41	35.84±1.81	34.90±0.97	46.77±2.18	27.56±2.78	43.08±0.21	81.36±0.56	87.32±0.10	96.92±0.06	72.03±0.21
	Legendre	87.68±0.96	74.75±2.13	89.72±0.36	89.86±0.76	66.36±2.00	79.04±2.09	48.86±0.92	59.04±1.84	25.00±2.69	36.17±2.48	36.94±2.39	37.69±1.43	59.00±6.53	42.90±0.22	44.64±3.46	84.43±0.51	87.79±0.71	96.95±0.04	57.76±0.21
	Jacobi	87.02±0.87	75.25±2.28	89.35±0.95	89.48±0.80	68.08±1.57	78.13±2.62	49.09±0.55	38.92±2.73	28.58±1.78	37.43±4.48	34.86±1.74	30.28±1.91	64.77±1.74	43.58±0.98	48.38±0.07	84.77±0.48	87.67±0.27	97.16±0.35	56.15±3.20
	Favard	86.14±1.60	73.29±2.77	89.18±1.00	89.74±0.79	58.86±4.25	74.51±4.90	42.26±0.00	64.53±0.35	31.29±0.95	30.83±5.11	36.54±2.69	35.97±1.58	70.82±0.86	40.98±3.44	48.05±0.20	79.59±9.46	86.82±0.24	97.67±0.66	(OOM)
	OptBasis	80.74±2.03	69.65±2.84	89.28±0.75	88.73±0.51	56.31±7.84	79.93±1.25	44.81±1.21	58.67±1.92	(OOM)	35.53±2.54	34.13±1.09	36.88±1.13	72.76±1.06	41.65±0.73	48.98±0.08	80.72±4.70	87.80±0.69	97.39±0.44	(OOM)
Bank	AdaGNN	85.45±1.25	74.52±2.09	89.82±0.65	85.83±4.37	67.48±1.86	79.65±1.29	53.55±0.75	54.62±2.20	28.06±5.25	36.10±4.05	38.62±2.19	36.77±1.23	64.62±0.89	39.39±1.48	46.41±0.58	77.02±1.94	87.73±0.36	97.17±0.42	63.99±1.00
	FBGNNI	81.43±5.82	73.56±1.36	87.06±0.85	89.60±3.13	61.14±3.56	76.63±2.09	51.08±0.66	63.73±3.91	(OOM)	35.88±3.40	33.20±2.22	34.66±1.72	60.35±1.52	40.36±1.71	47.70±0.48	68.67±2.35	87.62±0.31	97.63±0.14	(OOM)
	FBGNNII	81.18±4.04	73.08±2.08	88.12±0.91	84.37±3.70	62.72±5.60	77.31±1.48	52.84±1.12	66.76±3.93	(OOM)	39.97±4.30	34.05±1.66	36.02±1.18	58.82±2.97	42.91±0.41	45.14±2.53	72.22±0.35	82.11±0.36	97.77±0.24	(OOM)
	ACMGNNI	85.12±1.98	74.62±1.90	88.70±0.94	86.21±4.93	56.29±7.46	80.30±1.85	51.04±1.07	69.41±0.51	(OOM)	38.83±4.15	22.78±6.72	35.73±0.81	55.79±2.70	42.13±0.25	43.46±2.03	72.50±0.27	85.73±1.86	96.93±0.06	(OOM)
	ACMGNNII	84.16±4.09	73.95±2.11	89.16±0.54	86.23±3.50	52.93±5.17	84.51±1.09	52.83±1.65	54.49±2.39	(OOM)	35.04±4.34	34.69±1.42	32.50±3.91	55.20±2.08	40.04±2.83	46.94±0.37	73.96±0.30	87.44±0.37	97.08±0.25	(OOM)
	FAGNN	85.45±0.85	75.53±1.58	87.19±1.36	73.98±2.10	66.95±4.62	74.99±1.20	50.06±1.75	67.51±0.40	30.78±1.89	37.71±2.79	35.34±1.62	26.50±1.77	39.50±0.36	43.28±0.77	48.25±0.28	67.15±2.28	88.13±0.16	97.06±0.21	53.55±0.28
	G^2CN	85.12±3.32	75.50±1.44	88.67±0.69	67.98±1.24	68.95±2.30	77.91±1.95	51.83±0.21	68.96±0.92	29.72±2.17	40.45±2.75	34.63±1.12	35.76±1.22	61.45±1.89	44.35±1.35	48.29±0.46	77.26±0.65	88.37±0.27	97.43±0.15	54.13±0.04
	GNN-LF/HF	86.83±0.98	75.53±1.67	89.03±0.66	87.80±0.77	60.29±3.55	78.27±2.02	52.34±0.19	60.97±0.32	27.63±0.48	36.80±2.40	36.04±2.04	34.29±2.07	62.68±3.63	38.72±0.04	41.28±0.26	74.54±0.68	87.81±0.11	97.02±0.14	63.06±0.23
	FiGURe	87.04±0.99	74.49±1.71	88.22±0.95	89.71±0.56	65.38±8.16	79.76±1.39	53.02±0.65	67.57±0.40	29.43±2.18	34.90±1.63	39.16±1.24	33.45±1.09	62.84±1.45	41.69±0.34	44.14±3.99	83.30±0.31	88.19±0.83	97.15±0.38	(OOM)

In this section, we analyze the effectiveness among the filter variants, particularly focusing on two aspects: filters under varying degrees of heterophily and filters across different categories.

RQ3: *How do the graph pattern of homophily/heterophily affect filter effectiveness?*

As introduced in Section 2.1, under *homophily*, simple filters such as Linear can leverage graph inductive bias in low-frequency components and benefit effectiveness over Identity [31]. On conventional homophilous graphs (CORA and PUBMED), it is common that a large part of filters can achieve top-tier accuracy within the error interval. This observation indicates that the graph signal is relatively easy to learn, and is in line with other recent GNN evaluations [74, 87]. With proper settings, **even conventional fixed filters such as PPR and HK can deliver satisfying performance**. Contrarily, variable filters and filter banks (AdaGNN and FiGURe) are superior on TOLOKERS and MINESWEEPER. Although these graphs are also classified as homophilous, we deduce their graph patterns are more complex than simple low-frequency clustering, and thus demands more adaptive filters.

The scenario of *heterophily* is more challenging, as the local graph structure does not assist graph learning and entails advanced filters featuring high-frequency spectral signals, i.e., filters emphasizing θ_k for larger k-s under our polynomial formulation Eq. (1). Empirical results in Table 5 support the design intuitive, as low-pass filters (Linear, Impulse) fail on heterophilous graphs, sometimes performing even worse than Identity. **Filters incorporating high-pass components are effective in learning under heterophily**. This guideline applies to all filter categories: for fixed filters, this can be achieved by increasing the hop number K (Monomial) or reducing the decay factor α (PPR, Gaussian). For variable filters, it is usually beneficial to select bases that facilitate learning for high-frequency parameters (Horner, Bernstein) [70]. Meanwhile, there are also critiques that some variable bases, including Monomial and Bernstein, prioritize heterophilous accuracy and sacrifice their capability under homophily [70].

RQ4: *What is the impact of variable and filter bank designs on filter effectiveness?*

In our taxonomy Table 1, we identify the different designs of filter parameterization, typically discriminating fixed, variable, and filter bank designs. Our observation in Table 5 suggests that, **different designs do not guarantee improved accuracy on *certain* graphs, but may benefit**

Table 6. Effectiveness results (%) and standard deviations of spectral filters with *mini-batch* training on all datasets. For each dataset, results are highlighted based on the relative effectiveness among filters, where green results are better.

Filter	CORA	CITESEER	PUBMED	MINE	QUESTIONS	TOLOKERS	ARXIV	MAG	PRODUCTS	CHAM.	SQUIRREL	ACTOR	ROMAN	RATINGS	FLICKR	YEAR	PENN94	GENIUS	TWITCH	POKEC	SNAP	WIKI
Identity	66.54±2.35	67.10±1.79	87.48±0.56	50.18±2.13	66.39±1.49	74.39±0.31	47.05±0.12	55.28±0.31	10.61±nan	26.62±0.04	29.10±5.13	24.98±2.41	34.86±1.46	63.64±0.47	44.49±1.27	35.69±0.25	72.27±0.49	85.31±1.29	99.98±0.01	61.71±0.08	30.39±0.03	35.21±0.13
Linear	85.29±1.27	74.66±1.11	84.96±0.50	65.64±2.16	63.13±2.83	78.84±1.44	52.58±0.63	68.24±0.21	13.06±0.92	26.59±0.01	38.09±3.13	38.33±2.46	25.99±1.07	31.28±1.14	34.69±2.49	49.07±0.26	72.38±0.34	88.37±0.15	72.77±2.62	54.98±0.38	45.33±0.19	37.69±0.42
Impulse	86.04±1.76	74.14±1.04	84.03±0.51	62.99±0.79	67.53±1.31	66.54±5.00	50.55±0.62	70.75±0.15	22.42±1.69	26.16±0.75	39.83±2.21	38.94±1.78	24.32±1.60	28.63±0.87	35.81±2.37	45.23±0.15	60.52±0.49	88.38±0.34	77.50±0.67	57.84±0.11	53.56±0.18	32.88±1.47
Monomial	86.30±1.54	74.66±1.88	89.46±0.47	58.46±5.64	73.70±2.08	81.61±1.30	50.80±0.23	70.66±0.11	32.79±0.84	26.59±0.00	37.70±2.16	34.91±0.89	32.64±1.41	64.14±0.51	43.83±0.89	48.93±0.31	75.03±0.32	88.15±0.25	94.36±6.64	63.82±0.10	42.83±1.12	23.07±2.46
PPR	87.19±1.72	75.53±1.50	88.73±0.46	81.53±2.86	65.36±1.91	80.90±1.19	50.31±0.11	70.02±0.40	17.02±3.99	26.91±0.44	40.73±2.78	34.89±1.29	32.49±1.22	69.81±0.59	39.72±1.00	48.56±0.31	75.03±0.36	89.06±0.49	97.34±0.32	62.18±0.04	47.83±3.67	22.40±2.00
HK	86.60±1.39	74.76±0.96	89.39±0.41	68.48±1.02	71.99±1.90	76.79±0.89	50.69±0.27	58.91±0.43	25.72±0.67	26.61±0.76	41.63±3.14	36.37±1.94	32.88±1.39	67.82±0.49	36.66±1.16	47.50±0.30	72.77±0.76	89.10±0.14	97.79±1.10	62.04±0.08	50.99±0.16	43.06±0.30
Gaussian	67.86±1.81	74.44±1.23	88.70±0.55	82.16±1.65	57.80±1.95	67.78±4.30	51.45±0.24	68.61±0.29	25.70±0.70	26.59±0.00	39.08±2.93	35.32±1.60	32.84±1.28	65.38±1.55	34.76±2.36	48.03±0.21	73.93±0.45	88.44±0.40	99.32±0.10	62.20±0.05	52.58±0.04	42.81±0.45
Linear	76.36±0.79	71.79±1.40	87.60±0.47	50.57±1.89	65.34±4.59	71.50±0.35	46.63±0.18	54.85±0.32	26.44±0.84	26.49±0.23	37.36±3.17	34.93±1.94	35.72±0.88	63.34±0.79	36.27±0.83	35.03±0.50	74.63±0.67	86.10±0.15	89.90±5.42	62.21±0.16	30.47±0.34	38.98±0.51
Monomial	87.41±1.24	76.26±1.35	89.83±0.53	67.66±3.43	63.67±4.81	75.35±2.85	53.62±0.41	67.05±0.47	33.11±0.37	26.82±0.42	33.99±3.79	35.07±2.04	34.81±1.34	70.63±1.80	30.84±4.82	47.92±0.41	82.63±0.54	89.51±0.47	81.87±6.42	78.11±0.70	50.35±0.07	31.34±0.48
Horner	86.64±1.63	75.33±0.80	84.17±0.46	62.14±1.32	63.35±2.36	71.22±1.21	52.47±0.30	65.53±0.63	30.04±0.47	26.59±0.00	36.40±2.74	36.62±1.69	25.49±1.47	28.40±1.69	30.91±4.71	44.84±0.19	47.09±0.65	88.38±0.14	77.82±0.67	62.87±0.25	38.73±1.71	31.42±0.37
Chebyshev	87.89±0.87	75.71±1.63	90.06±0.33	88.36±4.03	68.64±4.79	80.33±1.35	47.99±1.14	71.05±0.21	19.54±5.50	26.58±0.01	34.27±3.39	34.23±1.21	29.89±1.82	35.06±2.65	37.15±2.22	45.54±0.49	83.40±0.40	86.70±0.19	99.92±0.05	78.95±0.13	42.83±0.96	24.26±2.39
Clenshaw	80.83±2.22	72.60±0.96	75.75±1.07	54.17±2.07	63.99±1.80	75.03±1.64	41.35±0.99	54.34±0.15	29.74±0.94	26.59±0.00	35.06±2.57	35.18±0.97	23.93±0.55	21.42±0.60	45.63±1.05	44.64±2.77	48.39±2.59	86.98±0.07	74.34±4.28	52.79±2.66	47.18±0.20	31.25±0.10
ChebInterp	84.70±3.59	74.73±1.89	89.28±0.43	89.48±0.95	58.05±1.94	68.83±0.97	53.79±0.64	65.73±0.89	28.26±0.84	28.11±1.17	28.37±3.09	34.62±1.98	30.36±1.22	70.95±0.69	35.58±2.22	44.28±0.29	63.59±0.70	87.65±0.38	99.71±0.45	78.60±0.68	45.77±0.56	23.85±1.84
Bernstein	82.05±1.46	69.18±1.83	85.46±0.45	77.98±1.77	62.56±3.79	74.01±1.19	44.58±2.06	39.92±0.71	16.07±0.32	26.59±0.17	39.61±3.21	35.68±2.35	30.68±1.05	53.87±1.09	31.24±2.14	37.34±1.11	79.72±0.34	88.12±0.47	97.43±0.55	73.61±0.02	48.67±0.28	24.54±1.95
Legendre	87.43±1.00	76.17±1.02	89.84±0.29	63.32±1.81	64.14±2.24	75.78±1.02	50.53±0.20	71.65±0.13	34.74±0.45	27.10±0.36	38.70±2.49	36.15±2.02	33.05±1.45	61.08±2.63	39.94±1.27	47.73±0.30	76.88±0.41	89.15±0.42	97.32±4.52	72.60±1.49	42.94±0.54	31.16±0.40
Jacobi	87.17±1.50	75.34±1.26	89.75±0.33	89.84±0.91	71.42±3.01	76.80±0.72	53.29±0.17	71.41±0.09	32.96±0.23	24.37±3.14	38.54±3.73	35.86±1.80	33.32±2.14	71.05±0.67	37.06±1.47	50.29±0.23	84.06±0.43	89.79±0.10	98.61±2.03	73.05±2.02	53.37±2.23	33.25±3.10
OptBasis	82.20±2.46	73.11±1.23	88.23±1.10	88.56±1.13	57.70±5.70	79.62±2.38	50.81±0.68	70.46±0.24	31.29±1.68	26.56±0.06	38.09±2.54	35.00±1.62	29.34±1.61	56.85±2.90	42.15±1.08	40.87±0.89	81.81±0.99	88.00±0.57	86.13±0.50	78.67±0.11	57.49±2.42	30.87±3.11
FAGNN	86.95±1.13	75.07±1.05	85.25±0.28	73.82±0.93	62.01±2.39	77.91±0.59	53.94±0.21	70.11±0.33	35.55±0.37	26.59±0.01	39.66±2.47	37.91±2.05	28.23±1.78	46.59±1.97	41.44±0.75	45.85±0.96	68.49±0.88	88.14±0.17	79.92±1.70	62.60±0.12	39.29±0.66	31.23±0.55
G^2CN	86.51±1.79	75.05±1.11	84.42±0.51	73.11±1.09	72.65±2.74	69.70±6.19	51.04±0.24	67.34±0.41	29.68±0.27	26.65±0.13	40.79±3.47	39.08±2.93	21.47±0.88	65.53±0.74	33.84±1.18	49.85±0.36	80.55±0.45	89.50±0.46	78.72±1.21	50.69±0.05	25.89±0.16	38.66±0.10
GNN-LF/HF	87.36±1.02	75.95±1.35	89.76±0.39	76.77±7.55	73.70±1.52	79.35±0.95	47.65±0.45	70.51±0.06	29.41±0.22	25.80±0.91	40.66±2.14	34.75±1.98	33.91±1.61	45.11±5.00	38.04±1.84	49.47±0.38	75.72±0.44	88.79±0.15	99.91±0.07	62.95±0.65	44.82±1.81	35.97±1.65
FiGURe	87.21±1.13	76.67±1.33	89.73±0.49	57.93±3.89	70.88±1.87	77.36±3.03	52.51±0.36	71.87±0.18	34.43±1.13	26.24±1.03	39.33±4.26	38.21±2.33	33.45±2.07	63.48±6.27	39.09±1.17	50.26±0.33	83.70±0.38	83.21±0.18	71.26±3.64	73.58±2.53	41.76±0.66	38.57±0.76

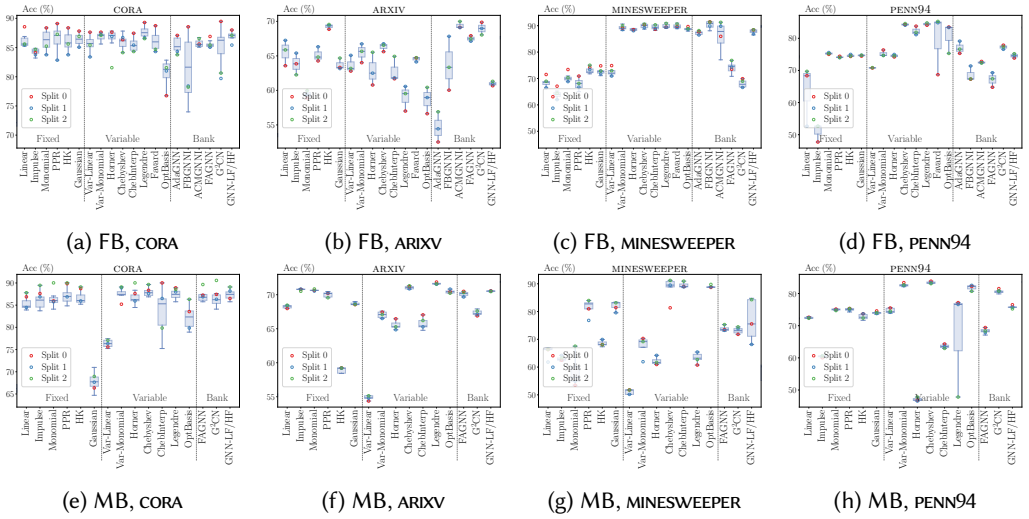
generality across *various* datasets. In specific, the simple and fast *fixed* filters sufficiently achieve top-tier accuracy on a number of datasets, as demonstrated in RQ3. *Variable* filters offer a more flexible approach for approximating a wider range of the graph spectrum. This allows for dynamically learning weight parameters from input signals, which is advantageous for capturing and leveraging richer information in the frequency domain. In cases where graph information is useful, these filters empirically produce better effectiveness, although the level of improvement varies with different data distributions. The *filter bank* design alternatively enhances model capacity by adopting multiple filters, usually spanning diverse frequency ranges. This approach is effective in mitigating the failure of a single filter and ensures reasonable accuracy in broad scenarios. On the other hand, this design does not necessarily outperform a single filter since there is no inherent improvement in filter expressiveness.

5.2.2 Relation to Efficiency. Then, we relate filter effectiveness and efficiency to deliver a comprehensive discussion on choosing proper filters. We first investigate the impact of learning schemes in RQ5, then discuss the relationship between effectiveness and efficiency in our core research question RQ6.

RQ5: *How do full-batch and mini-batch training schemes affect model efficacy?*

Theoretically, mini-batch computation is identical to the full-batch scheme from the aspect of spectral operations, and the only difference roots in the feature transformation procedure, as MB models do not perform the pre-transformation on input attributes before applying graph filters. *Empirically*, the majority results in Table 6 confirm that **MB training delivers comparable accuracy to the corresponding FB results** in Table 5, and our key observations RQ3&4 on filter effectiveness still hold. It also supports our motivation for extracting and benchmarking spectral filters, as they are generally applicable to different training schemes without affecting the capability of GNN learning.

Meanwhile, accuracy drops of MB can be observed on graphs with a small attribute dimension F_i (MINESWEEPER, TOLOKERS, RATINGS). This defect can be explained by the over-squashing phenomenon [4], which stems from the information loss when encoding the comprehensive graph topology into a small F_i during the separate filtering process. Variable filters are relatively prone to this issue since their filter parameters are not sufficiently trained under these circumstances. In contrast, MB on heterophilous datasets such as CHAMELEON and ROMAN achieves higher accuracy than full-batch models. This is precisely the opposite outcome of over-squashing, where malignant graph information is implicitly alleviated by the spectral filtering process on raw attributes.



(a) FB, CORA (b) FB, ARXIV (c) FB, MINESWEEPER (d) FB, PENN94

(e) MB, CORA (f) MB, ARXIV (g) MB, MINESWEEPER (h) MB, PENN94

Fig. 3. Statistical significance of full-batch (upper row) and mini-batch (lower row) filter effectiveness. Parameter initialization and dataset split for all filters follow the same set of random seeds. Results from three seeds are marked in the figure.

> **RQ6:** *What is the relationship between filter effectiveness and efficiency? Moreover, how to choose spectral filters that are both effective and efficient?*

A prevailing trend in spectral GNN studies favors more sophisticated filter designs for better effectiveness, which, as indicated by RQ1, compromise time and memory efficiency. However, RQ3 demonstrates that these two techniques lead to distinctive outcomes compared with fixed filters: Improving filter variability through *learnable parameters* can expand model capability in some cases, though the impact largely depends on the usefulness of graph signals. The design also incurs additional efficiency overhead related to the feature transformation. The *filter bank* approach usually contributes to overall filter adaptability across various graph signals, rather than improving accuracy of existing filters. This comes at the cost of multiplying the computation time and memory by the number of filters involved.

In other words, different from the common view of a straightforward trade-off between efficacy and efficiency, **our benchmark study uncovers a more intricate nature: these two aspects are not mutually exclusive**. The filter effectiveness is determined by its *inherent frequency response*. Even with the same variability and complexity, different filters yield varied accuracy, and filters appropriate to the graph signals results in better accuracy. Alternatively, time and memory efficiency are relevant to the *external designs* of graph computation and feature transformations, which can be explicitly inferred from their complexity. For instance, when processing heterophilous graph topology, a fixed high-order filter is more likely to achieve superior performance compared to a filter with a large parameter size focusing on low-frequency components.

Thus, balancing effectiveness and efficiency preferably requires a comprehensive consideration of graph knowledge and available environments. It is more important to find a suitable spectral expression by examining the particular graph input, instead of simply exploiting sophisticated designs. Significant factors affecting learning efficacy and spectral expressiveness are thus explored more specifically in Section 6. **We suggest the following practice as an attempt toward effective and efficient spectral GNNs**: When learning on simple and homophilous graphs, one can stick to fixed filters for the best efficiency and comparably superb precision. For tasks with

complex graph signals, or when fixed filters are inadequate, it is recommended to carefully choose an appropriate model that fits the graph spectrum and producing satisfactory performance, while ensuring that the time and memory overheads remain feasible in the given environment.

5.3 Result Stability

In this section, we especially investigate the statistical significance of our evaluation on filter effectiveness and efficiency, ensuring that our conclusions are widely applicable to general circumstances.

5.3.1 Efficacy Variance and Divergence. We further visualize the confidence intervals of filter accuracy in Tables 5 and 6 by box plot in Figure 3 with selected seeds. Particularly, CORA and ARXIV represent the datasets with random and attribute-based splits, respectively, and all filters learn on the same split for the same seed. Figure 3(a) implies that the variance in datasets like CORA is largely caused by the split difference, as some seeds lead to high accuracy for most filters while others greatly impede efficacy. This is a common phenomenon in semi-supervised learning, where random splits may not be representative of containing sufficient information, such as lacking minor label groups, which results in large accuracy deviation. On ARXIV, where the splits are more stable, the filter accuracy is more concentrated. Nonetheless, in both cases, **the relative effectiveness among filters can be effectively depicted by average accuracy** while certain outliers are caused by less representative data splits, which supports our RQ3&4 based on Table 5. Results between FB and MB are also supports the observation in RQ5 that MB largely maintains accuracy, while being more unstable on graphs with low-dimensional attributes such as MINESWEEPER.

Our scalable implementation offer an unprecedented opportunity to study filter efficacy across graphs with different scales, which is showcased in Figure 4 with representatives in three categories. Together with Figure 3, we observe that **the relative difference among filters is more significant on larger graphs**. On small-scale graphs (Figure 3(a)), the accuracy difference among filters are marginal, and a broad range of filters can achieve high accuracy. However, filter effectiveness becomes more divergent on larger graphs where $n > 10^5$ (Figure 3(b)). While filters with suitable frequency responses, e.g. the fixed Monomial, can still achieve leading performance on corresponding graphs, inappropriate ones exhibit larger accuracy gaps from the best filters. The finding further underscores the importance of our conclusion RQ3 especially on large-scale graphs, that choosing filters that fit the graph spectrum is critical for effectiveness.

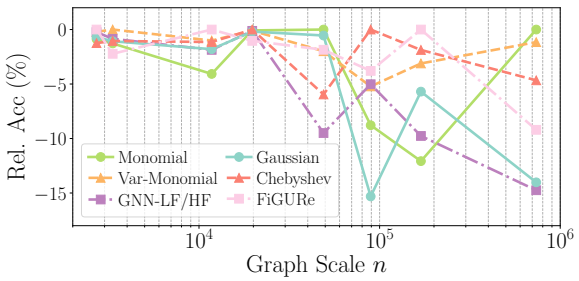
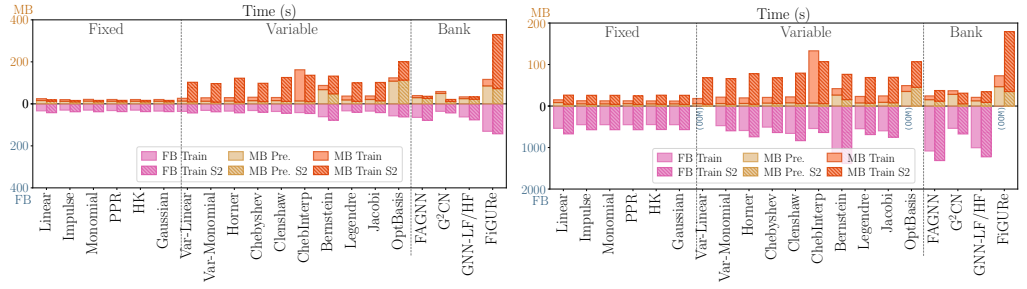


Fig. 4. Shift of filter effectiveness on homophilous datasets across different scales. Effectiveness (y-axis) is presented by the relative accuracy to the highest filter in each dataset. The graph scale (x-axis) is presented by node size n in log scale.

5.3.2 Efficiency on Different Hardware. To validate our efficiency observations on diverse hardware platforms, we evaluate the filter efficiency on another server marked as S2, which is with slower CPUs (Intel Xeon, 2.2GHz) and a faster GPU (NVIDIA RTX A5000). The time breakdown on the typical dataset PENN94 is in Figure 5. Notably, the figure demonstrates **varied bottlenecks for different filter types** due to the hardware difference. For MB fixed filters with transformation being the bottleneck, the overall learning time on S2 is shorter thanks to faster GPU computation.



(a) PENN94 (b) POKEC

Fig. 5. Time efficiency comparison of FB (lower axis) and MB (upper axis) training on PENN94 and POKEC with different hardware.

In comparison, as graph propagation dominates the efficiency of FB training and MB variable filters, the empirical speed is relatively slower. The slowdown is more significant on datasets larger than PENN94. The observation aligns with RQ1&2 regarding the efficiency of model operations and learning schemes, verifying that they are generally applicable irrespective of hardware settings.

5.4 Key Conclusions

C1. Model time and memory efficiency are respectively dominated by graph propagation and weight transformation operations as indicated by our taxonomy. On graphs above million-scale, propagation turns into the bottleneck of GNN training. (RQ1)

C2. Mini-batch training is a unique learning scheme of spectral GNN models, offering comparable efficacy and better scalability, especially on large graphs. In turn, it falls short in flexibility and is more sensitive to raw graph attributes. (RQ2&5)

C3. Model effectiveness mainly stems from the inherit filter property of frequency response, i.e., the emphasis on low- and high-frequency signals. With proper configurations, simple filters can also excel on suitable graphs. (RQ3)

C4. While sophisticated filter designs are favored in common consensus, they are less related to accuracy improvement, but potentially benefit generality across datasets. Variable filters increase the capacity to deal complex input signals, while filter banks can assemble the performance of individual filters. (RQ4)

C5. Contrary to the prevailing belief, we reveal that efficacy and efficiency are not mutually exclusive. We provide our practical guidelines for analyzing the graph spectrum and choosing appropriate filters to achieve both efficacy and efficiency. (RQ6)

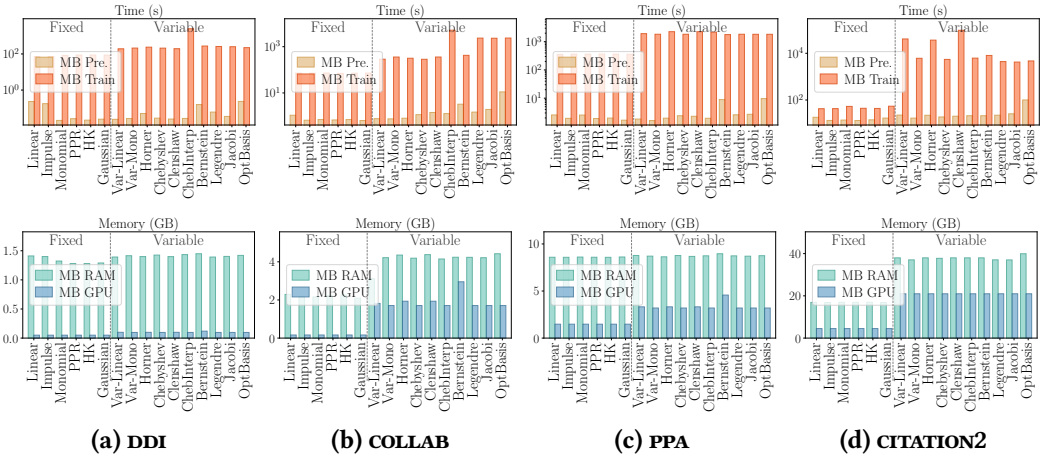
6 Results: Specific Evaluations

In this section, we conduct in-depth evaluations regarding specific properties of individual spectral filters, offering ancillary view on obtaining desired filter effectiveness and efficiency. Further research questions are raised for evaluations leading to new findings.

6.1 Extended Tasks

In this section, we highlight the generalizability of our implementation and evaluation by comparing to other graph processing methods and performing tasks other than node classification.

6.1.1 Implementation Comparison. To demonstrate the performance of our implementation, especially regarding GNN efficiency and scalability, Table 7 presents the additional evaluation for typical GNN models deployed in other popular frameworks. The baselines include spatial message-passing GNNs (GraphSAGE [37]), spectral message-passing GNNs (GCN [46], ChebNet [18]), as



(a) DDI (b) COLLAB (c) PPA (d) CITATION2

Fig. 6. Filter time and memory efficiency of MB link prediction on four datasets. Note that the time axis is on log scale.

well as scalability-oriented Graph Transformers (NAGphormer [11], ANS-GT [121]). The models are implemented with available graph backends from the PyG framework [27]. Especially, the EI backend is the most common backend used in PyG by default entailing $O(mF)$ space footprint for graph propagation. The SP backend is more efficient and used as the default backend in our main experiments, while only a handful of PyG models are compatible with it.

It can be observed that most accuracy are in line with our results in Table 5. Regarding efficiency and scalability, message-passing models with the SP backend is more superior than EI for faster training and less GPU memory footprint, while the most common EI backend encounters OOM on large datasets. Graph Transformers are more computational intensive for demanding significantly long precomputation, slower training speed, and more memory overhead due to the complicated model architecture.

6.1.2 Link Prediction. Link prediction is another popular task that presents substantial challenges regarding GNN scalability [41, 50]. Compared to the node classification task in Section 5, link prediction focuses on transformation operations, as identifying node-pair correlation through neural networks is critical for prediction accuracy. Graph information retrieved by the filters only serve as fundamental knowledge for the downstream transformation. Different from Section 2.2, the typical complexity for transformation is $O(\kappa mF^2)$. This is because for a graph with m ground-truth edges, the model needs to processes a total of κm positive and negative samples, where κ is usually

Table 7. Effectiveness and efficiency of models outside our framework on representative datasets. "Train" and "Infer" respectively refer to average training time per epoch and inference time (s), while precomputation time is separated in applicable models.

Model (Backend*)	ARXIV				MAG				PENN94				POKEC			
	Acc	Train	Infer	GPU	Acc	Train	Infer	GPU	Acc	Train	Infer	GPU	Acc	Train	Infer	GPU
GCN (SP)	53.2	0.05	0.03	1.1	27.4	0.31	0.11	7.8	73.1	0.04	0.03	1.3	60.4	663.3	0.35	0.01
GraphSAGE (SP)	50.3	0.04	0.005	1.2	24.8	0.25	0.01	7.7	74.2	0.08	0.003	2.3	63.4	0.45	0.009	9.6
GCN (EI)	53.0	0.06	0.06	3.2		(OOM)			67.6	0.06	0.06	3.7		(OOM)		
GraphSAGE (EI)	54.1	0.09	0.03	2.7	28.2	0.33	0.18	10.8		(OOM)				(OOM)		
ChebNet (EI)	53.4	0.11	0.05	3.0		(OOM)				(OOM)				(OOM)		
NAGphormer (EI)	67.8	280+3.2	2.1	2.3	33.2	89+10.3	2.2	3.8	74.4	237+6.1	2.1	2.3	73.1	70+16.1	3.1	8.9
ANS-GT (EI)	71.1	16205+109	2.7	11.3		(OOM)			67.8	34092+37	5.0	8.7		(OOM)		

* **Backend**: SP = `torch.sparse`, EI = `torch_geometric.EdgeIndex` [27].

$2-10$. Hence, performing the **link prediction task inevitably entails mini-batch training**, as the full-scale memory overhead is prohibitive.

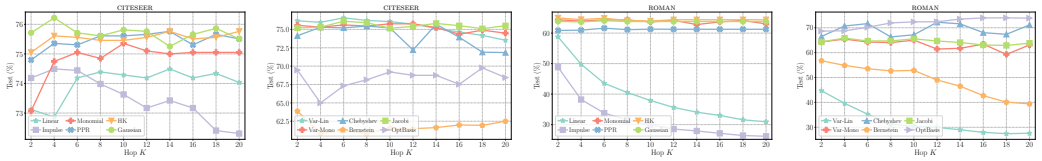
As the effectiveness of link prediction is mainly determined by transformation networks which are orthogonal to this study, we mainly stick to a simple MLP network and focus on efficiency evaluations shown in Figure 6, while more sophisticated downstream modules can be integrated in a plug-and-play manner as introduced in Section 4. Compared to RQ1, link prediction **efficiency is dominated by the transformation operation on larger graphs** due to the considerable amount of iterative edge-wise computations, while GPU memory can be controlled by the batch size. Nonetheless, such computational bottleneck is less related to the topic of graph processing, and a range of dedicated accelerations are available [108, 111, 115].

6.1.3 Signal Regression. Spectral filtering also offers a regression task for fitting a given signal encoded in the graph, which is intuitive in characterizing the frequency response to signals spanning diverse frequency components. We define the fully-supervised graph regression task [39] by learning from the predefined pair of input $\mathbf{x}$ and the objective $\mathbf{z} = g^\star * \mathbf{x}$, as an approach to approximate the given simple spectral filter function $g^\star$. Table 8 presents the average R^2 score of five typical signal functions, where larger scores indicate better regression precision.

The observation implies that most filters still highly focus on low-frequency components, demonstrating higher precision on LOW and BAND REJECT. This is in line with our findings in RQ4 that introducing variable bases does not necessarily strengthen filter capability, as the performance is principally determined by its spectral formulation. On the contrary, **filters enforcing higher attention on high-frequency domains**, including Monomial, Horner, and OptBasis, **achieve strong performance on high-frequency signals** (BAND, COMBINE, and HIGH), although Monomial and Horner sacrifice low-frequency ability to certain degrees. OptBasis outperforms on all signals thanks to its flexible parameter acquisition. However, it should be noted that the regression precision of simple signals does not guarantee node classification effectiveness in our main experiment, as better utilization of graph signals is not always beneficial due to the complex grounding of the realistic task.

Table 8. Average R^2 scores of graph regression on five signal functions. For each dataset, results are highlighted based on the relative effectiveness among filters, where green results are better.

Type	Dataset / Signal	BAND $e^{-10(\lambda-1)^2}$	COMBINE $\lvert\sin(\pi\lambda)\rvert$	HIGH $1-e^{-10\lambda^2}$	LOW $e^{-10\lambda^2}$	REJECT $1-e^{-10(\lambda-1)^2}$
Fixed	PPR	21.40	32.13	35.42	77.58	91.24
	Linear	6.74	10.12	8.87	94.90	83.22
	Impulse	6.73	9.99	8.79	93.27	82.25
	Monomial	21.40	32.13	35.42	77.58	91.24
	HK	7.57	11.77	10.23	96.93	86.94
	Gaussian	7.62	11.96	10.26	96.92	86.93
Variable	Monomial	6.87	10.21	8.87	94.80	79.75
	Horner	48.98	69.10	78.87	89.14	78.96
	Chebyshev	6.74	8.40	6.87	91.15	81.12
	Clenshaw	6.77	8.06	8.14	76.68	81.49
	ChebInterp	6.72	9.58	8.88	90.42	80.95
	Bernstein	13.03	17.79	22.10	97.56	86.08
	Legendre	6.68	9.82	8.59	94.00	72.86
	Jacobi	6.69	9.73	8.62	93.99	82.14
	Favard	5.23	8.89	7.46	67.57	67.68
	OptBasis	82.88	79.73	93.69	99.19	99.06



(a) Fixed, CITESEER (b) Variable, CITESEER (c) Fixed, ROMAN (d) Variable, ROMAN

Fig. 7. Effect of propagation hops K on representative full-batch fixed and variable filters.

6.2 Spectral Capabilities

As analyzed in RQ3, the effectiveness of graph learning with different filters is related to the compatibility between their spectral expressions and the graph data. Therefore, we delve into the spectral properties to provide a thorough understanding of the spectral capabilities concerning different filter formulations. These experiments orient the following question:

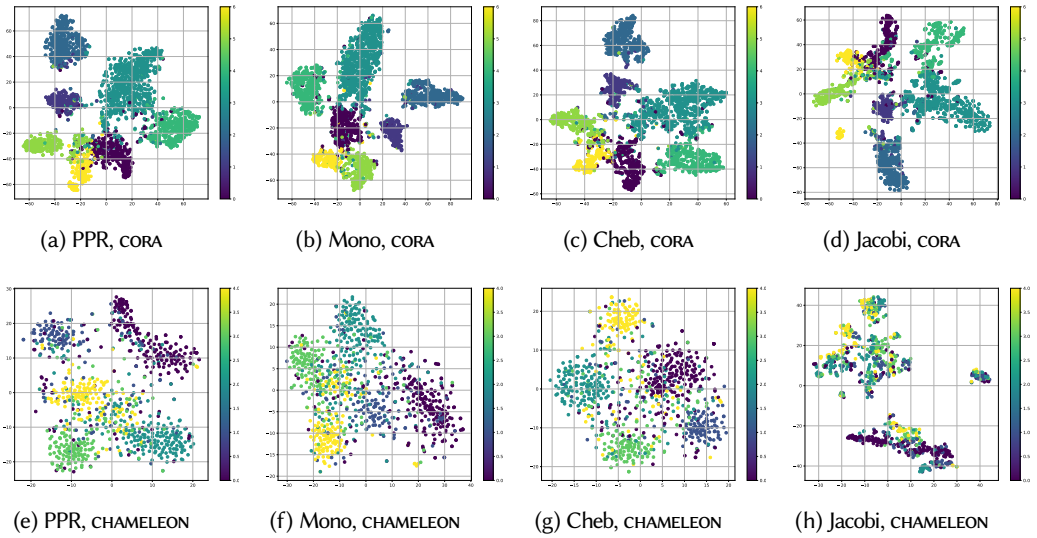
RQ7: *What inherent spectral properties are impactful in determining filter effectiveness?*

6.2.1 Effect of Propagation Hop. The number of propagation hops K is a critical hyperparameter governing both spectral expressiveness and empirical efficiency. Deciding the value of K requires careful consideration in the spectral domain. A small K indicates a limited number of polynomial terms in Eq. (1), leading to restricted filter capability. Conversely, a large number of propagations may introduce excessive graph information, which also implies linearly increased computational overhead.

The value of K in existing literature varies significantly due to non-spectral designs, complicating the assessment of the filter capability. Our framework offers a unified architecture for spectral GNNs, enabling a more thorough examination of the impact of propagation hops on graph filters. Figure 7 illustrates the model accuracy patterns when varying the number of hops in common the range $K \in [2, 20]$ on homophilous and heterophilous graphs. **Generally, a limited K is more favorable, especially for homophilous graphs.** Overall, our selection of $K = 10$ in the main experiments is reasonable for maintaining the performance of most models across various datasets.

For low-pass filters (Linear and Impulse), the efficacy gradually decreases when the hop number increases for both homophilous and heterophilous graphs. This corresponds to the over-smoothing issue [52, 105, 106], where the signal is overwhelm by non-local noise and loses useful information. Filters such as PPR and Gaussian can alleviate this issue by tuning the decaying factor, while filters with high-frequency components, especially those with orthogonal basis, are also more stable by manipulating those signals.

6.2.2 Clustering Visualization. The t-SNE method provides an intuitive way to understand the decision boundary learned by spectral GNNs [96], which is essentially related to their prediction performance in Table 5. Figure 8 showcases visualizations of representative filters. Generally, embeddings for most filters on the homophilous CORA are well-clustered with sharp boundaries, explaining the similar classification accuracy. On the heterophilous CHAMELEON, a few filters are able to maintain the ability to form clear clusters and consequently satisfy prediction performance (Monomial, Chebyshev). In comparison, the dispersed and overlapping clusters of PPR and ChebInterp correspond to the noise introduced by heterophilous graph topology, which undermines their effectiveness. Filters such as Impulse and Jacobi generate scattered clusters with notable outliers, rendering difficulties in fitting graph signals and producing meaningful representations. In summary, the evaluation also implies that **spectral expressiveness outweighs variable designs** in terms of filter efficacy when conducting classification tasks under different graph patterns.



(a) PPR, CORA (b) Mono, CORA (c) Cheb, CORA (d) Jacobi, CORA

(e) PPR, CHAMELEON (f) Mono, CHAMELEON (g) Cheb, CHAMELEON (h) Jacobi, CHAMELEON

Fig. 8. t-SNE clusters of learning results for selected filters and datasets. Nodes are colored by ground truth labels.

6.3 Degree-specific Effectiveness

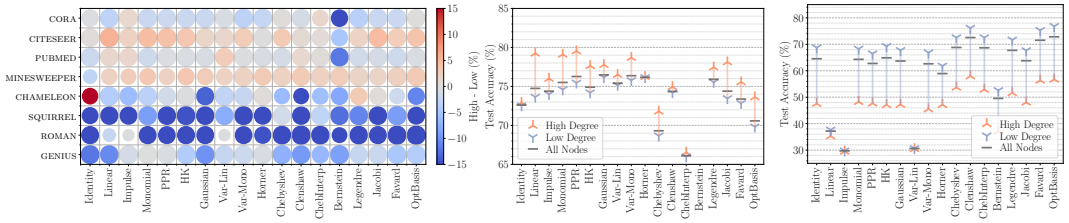
A recent trend examines node-wise performance and discovers that the learning effectiveness of vanilla GNNs varies with respect to nodes of differing degree levels [16, 48, 54, 58, 92, 105]. Unlike previous investigations assuming homophily, we extend the degree-wise evaluation to heterophilous graphs. Figure 9 displays the difference between prediction accuracy on high- and low-degree nodes across representative datasets. The extended experimental evaluation enables us to relate the degree-specific performance to overall model accuracy in Section 5.2.

6.3.1 Relation with Filter Effectiveness.

> **RQ8:** *What is the degree-wise effectiveness under homophily and heterophily, and how does it affect the filter efficacy?*

It can be observed in Figure 9 that most filters behave distinctively on homophilous and heterophilous datasets. On *homophilous* graphs (CITESEER in Figure 9(b)), the performance of high-degree nodes is generally on par with or higher than low-degree ones, which echoes earlier studies. From the spectral domain view, high-degree nodes are commonly located in clusters, which is more related to low frequency in the graph spectrum and is preferable for homophilous GNNs throughout learning. Contrarily, the accuracy of high-degree nodes is usually significantly lower under *heterophily*, as shown in Figure 9(c) for ROMAN. In this case, the hypothesis that higher degrees are naturally favored by GNNs no longer holds true. Although these nodes aggregate more information from the neighborhood, it is not necessarily beneficial, and heterophilous connections may carry destructive bias and hinder the prediction. As a more precise amendment to the conclusion in prior works, we state that the **degree-wise bias is more *sensitive*, but not necessarily beneficial, to high-degree nodes** with regard to varying graph conditions.

It is also notable in Figure 9 that the degree-specific difference is correlated with the test accuracy of models within each dataset, especially under heterophily. For example, in Figure 9(c) ROMAN, filters with greater bias achieve relatively better overall accuracy. In comparison, filters such as Linear and Impulse fail to distinguish between high- and low-degree nodes, resulting in lower



(a) Homophilous/heterophilous gap (b) CITESEER (c) ROMAN

Fig. 9. **(a)** Accuracy gap between high- and low-degree nodes on homophilous (upper four) and heterophilous (bottom four) graphs. Color of each data point indicates the difference value, while size of the circle represents the relative overall accuracy among all filters in each dataset. **(b)-(c)** Respective accuracy of high- and low-degree nodes on CITESEER and ROMAN.

test accuracy. From the observation, we deduce that **GNNs tend to recognize and adapt to the majority of low-degree nodes** in order to achieve higher performance. This preference is potentially exaggerated by the heterophily-oriented filter design, which pays more attention to the high-frequency components correlating to low-degree nodes while compromising the performance of high-degree nodes. It is thus advisable to explore filter formulations balancing different frequency ranges, which are promising for addressing the current drawback and further advancing overall model accuracy.

6.3.2 *Effect of Graph Normalization.*

> **RQ9:** *How to utilize the graph information to control degree-wise effectiveness, and therefore affect overall accuracy?*

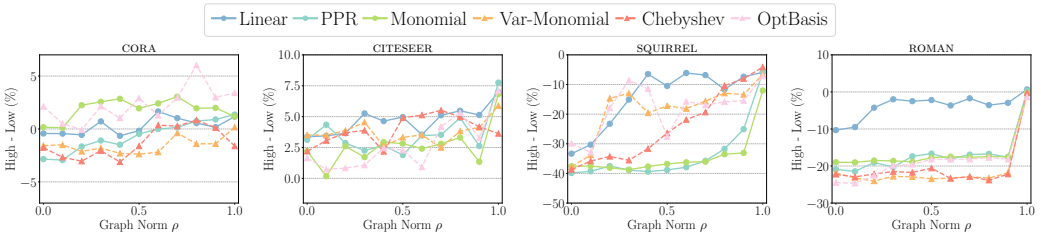
Recall that the normalized adjacency $\tilde{\mathbf{A}} = \bar{\mathbf{D}}^{\rho-1}\bar{\mathbf{A}}\bar{\mathbf{D}}^{-\rho}$ explicitly encodes degree information during propagation, we are thence motivated to use it to impact filter effectiveness. In previous studies, it is revealed that the normalization in form of $\tilde{\mathbf{A}} = \bar{\mathbf{D}}^{-1}\bar{\mathbf{A}}$ affects the node-wise performance on homophilous graphs [54, 119], while we extend to the continuous range of $\rho \in [0, 1]$. Specially, $\rho = 1/2$ is the symmetric normalization with equal contributions from both in- and out-edges.

We present experimental result of the accuracy gap between high- and low-degree nodes when varying the graph normalization hyperparameter ρ in Figure 10. It can be observed that **a larger ρ increases the difference value**, i.e., improves the relative accuracy of high-degree nodes for both fixed and variable filters on CITESEER and ROMAN. This suggests that inbound information is preferred for model inference on high-degree nodes, and adjusting the aggregation through ρ could be an practical attempt to alleviate degree-wise differences and achieve favorable performance. On graphs such as CHAMELEON and ACTOR, where connection utility is hindered by heterophily, this pattern is weaker, and the performance gap becomes more unstable due to the complexity of graph conditions.

6.4 Key Conclusions

C6. Filter effectiveness can be explained by their spectral properties, especially the response on different frequencies. Sophisticated designs are only effective when their frequency components match the graph signal. (RQ7)

C7. Spectral models exhibit dissimilar effectiveness on nodes of high and low degrees under different graph conditions, challenging the assumptions in previous studies. The efficacy on high-degree nodes degrades under heterophily, potentially indicating greater sensitivity to the graph learning process on these nodes. (RQ8)



(a) CORA (b) CITESEER (c) SQUIRREL (d) ROMAN

Fig. 10. Effect of graph normalization ρ on the accuracy difference between high- and low-degree nodes.

C8. The degree-wise bias is also relevant to the overall effectiveness of filters and can be controlled by altering the influence of graph topology. Dedicated filtering designs that benefit both low- and high-degree performance remain underexplored. (RQ9)

7 Conclusion and Open Questions

In this study, we conduct extensive evaluations of spectral GNNs concerning both effectiveness and efficiency. We first provide a thorough analysis of the graph kernels in existing GNNs and categorize them by spectral designs. These filters are then implemented under a unified framework with highly efficient learning schemes. Comprehensive experiments are conducted to assess the model performance, with discussions covering heterophily, graph scales, spectral properties, and degree-wise bias. Our benchmark observations also identify several open questions worthy of further research toward better spectral GNNs:

- *How to properly obtain effective and efficient spectral filters?* As analyzed in RQ6, it is possible to achieve both effectiveness and efficiency for some graphs, while finding the suitable filter is largely empirical. Our observation suggests that despite heterophily, latent patterns of graph data distribution also affect the spectral performance. Identifying and characterizing these factors is beneficial for designing more powerful filters based on graph knowledge.
- *How to further improve filter efficiency and scalability?* In RQ1, the spectral GNN design outlines a solution for performing learning on large graphs, although existing variable and composed filter computations are far from optimized. The GNN scalability issue calls for a persistent pursuit of more efficient designs, which is of great practical interest. Our study is helpful in uncovering impact factors and bottlenecks of large-scale deployment, and underscores the potential of spectral GNNs for achieving efficiency without compromising efficacy.
- *How to address the degree-specific bias to enhance overall effectiveness?* In RQ8, we discover that the degree-wise difference is related to graph heterophily and affects model precision. As current endeavors to mitigate GNN bias largely assume homophily, it is promising to search for specialized schemes that improve the accuracy of both low- and high-degree nodes, considering various graph conditions, which also benefits overall model performance.

Acknowledgments

Research by the first four authors was supported by Singapore MOE AcRF Tier-2 Grant (T2EP20122-0003) and NTU-NAP startup grant (022029-00001). Ningyi Liao is supported by the Joint NTU-WeBank Research Centre on FinTech, Nanyang Technological University, Singapore. The last author's research was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada (Grant Number RGPIN-2020-05408).

References

[1] 2025. A Comprehensive Benchmark on Spectral GNNs: The Impact on Efficiency, Memory, and Effectiveness [Technical Report]. arXiv:2406.09675 [cs] https://github.com/gdmnl/Spectral-GNN-Benchmark/blob/main/Report.pdf

[2] Sergi Abadal, Akshay Jain, Robert Guirado, Jorge López-Alonso, and Eduard Alarcón. 2022. Computing Graph Neural Networks: A Survey from Algorithms to Accelerators. *Comput. Surveys* 54, 9 (Dec. 2022), 1–38.

[3] Xin Ai, Qiange Wang, Chunyu Cao, Yanfeng Zhang, Chaoyi Chen, Hao Yuan, Yu Gu, and Ge Yu. 2024. NeutronOrch: Rethinking Sample-Based GNN Training under CPU-GPU Heterogeneous Environments. *Proceedings of the VLDB Endowment* 17, 8 (April 2024), 1995–2008. doi:10.14778/3659437.3659453

[4] Uri Alon and Eran Yahav. 2021. On the Bottleneck of Graph Neural Networks and Its Practical Implications. In *9th International Conference on Learning Representations*. arXiv:2006.05205 [cs, stat]

[5] James Atwood and Don Towsley. 2016. Diffusion-Convolutional Neural Networks. In *29th Advances in Neural Information Processing Systems*. 2001–2009.

[6] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2021. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3496–3507.

[7] Deyu Bo, Xiao Wang, Yang Liu, Yuan Fang, Yawen Li, and Chuan Shi. 2023. A Survey on Spectral Graph Neural Networks. (Feb. 2023). arXiv:2302.05631 [cs]

[8] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond Low-Frequency Information in Graph Convolutional Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 5 (May 2021), 3950–3957. doi:10.1609/aaai.v35i5.16514

[9] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *2nd International Conference on Learning Representations*.

[10] Defu Cao, Yujing Wang, Juanyong Duan, Ce Zhang, Xia Zhu, Congrui Huang, Yunhai Tong, Bixiong Xu, Jing Bai, Jie Tong, et al. 2020. Spectral temporal graph neural network for multivariate time-series forecasting. In *Advances in neural information processing systems*, Vol. 33. 17766–17778.

[11] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. 2023. NAGphormer: A Tokenized Graph Transformer for Node Classification in Large Graphs. In *11th International Conference on Learning Representations*.

[12] Jiali Chen and Liwen Xu. 2023. Improved Modeling and Generalization Capabilities of Graph Neural Networks With Legendre Polynomials. *IEEE Access* 11 (2023), 63442–63450. doi:10.1109/ACCESS.2023.3289002

[13] Ming Chen, Zhewei Wei, Bolin Ding, Yaliang Li, Ye Yuan, Xiaoyong Du, and Ji Rong Wen. 2020. Scalable graph neural networks via bidirectional propagation. *33rd Advances in Neural Information Processing Systems* (2020).

[14] Yuhan Chen, Haojie Ye, Sanketh Vedula, Alex Bronstein, Ronald Dreslinski, Trevor Mudge, and Nishil Talati. 2023. Demystifying Graph Sparsification Algorithms in Graph Properties Preservation. *Proceedings of the VLDB Endowment* 17, 3 (Nov. 2023), 427–440. doi:10.14778/3632093.3632106

[15] Zhiqian Chen, Fanglan Chen, Lei Zhang, Taoran Ji, Kaiqun Fu, Liang Zhao, Feng Chen, Lingfei Wu, Charu Aggarwal, and Chang-Tien Lu. 2023. Bridging the Gap between Spatial and Spectral Domains: A Unified Framework for Graph Neural Networks. *Comput. Surveys* (Oct. 2023), 3627816. doi:10.1145/3627816

[16] Zhengyu Chen, Teng Xiao, and Kun Kuang. 2022. BA-GNN: On Learning Bias-Aware Graph Neural Network. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 3012–3024.

[17] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *9th International Conference on Learning Representations*. arXiv:2006.07988 [cs, stat]

[18] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).

[19] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. 2020. GraphZoom: A Multi-Level Spectral Approach for Accurate and Scalable Graph Embedding. In *8th International Conference on Learning Representations*.

[20] Xiaowen Dong, Dorina Thanou, Laura Toni, Michael Bronstein, and Pascal Frossard. 2020. Graph signal processing for machine learning: A review and new perspectives. *IEEE Signal processing magazine* 37, 6 (2020), 117–127.

[21] Yushun Dong, Kaize Ding, Brian Jalaian, Shuiwang Ji, and Jundong Li. 2021. AdaGNN: Graph Neural Networks with Adaptive Frequency Response Filter. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, Virtual Event Queensland Australia, 392–401. doi:10.1145/3459637.3482226

[22] Keyu Duan, Zirui Liu, Peihao Wang, Wenqing Zheng, Kaixiong Zhou, Tianlong Chen, Xia Hu, and Zhangyang Wang. 2022. A comprehensive study on large-scale graph training: Benchmarking and rethinking. *Advances in Neural Information Processing Systems* 35 (2022), 5376–5389.

[23] Vijay Prakash Dwivedi, Chaitanya K Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2022. Benchmarking Graph Neural Networks. *Journal of Machine Learning Research* 23 (Dec. 2022), 1–48.

[24] Chanakya Ekbote, Ajinkya Pankaj Deshpande, Arun Iyer, Ramakrishna Bairi, and Sundararajan Sellamanickam. 2023. FiGURe: Simple and Efficient Unsupervised Node Representations with Filter Augmentations. In *36th Advances in Neural Information Processing Systems*.

[25] Chenchen Feng, Yu He, Shiyang Wen, Guojun Liu, Liang Wang, Jian Xu, and Bo Zheng. 2022. DC-GNN: Decoupled Graph Neural Networks for Improving and Accelerating Large-Scale E-Commerce Retrieval. In *Companion Proceedings of the ACM Web Conference 2022*. doi:10.1145/3487553.3524203

[26] Wenzheng Feng, Yuxiao Dong, Tinglin Huang, Ziqi Yin, Xu Cheng, Evgeny Kharlamov, and Jie Tang. 2022. GRAND+: Scalable Graph Random Neural Networks. In *Proceedings of the ACM Web Conference 2022*. ACM, Virtual Event, Lyon France, 3248–3258. doi:10.1145/3485447.3512044 arXiv:2203.06389

[27] Matthias Fey and Jan E. Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[28] Fabrizio Frasca, Emanuele Rossi, Davide Eynard, Ben Chamberlain, Michael Bronstein, and Federico Monti. 2020. SIGN: Scalable Inception Graph Neural Networks. In *ICML 2020 Workshop on Graph Representation Learning and Beyond*.

[29] Swapnil Gandhi and Anand Padmanabha Iyer. 2021. P3 : Distributed Deep Graph Learning at Scale. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*. 551–568.

[30] Xinyi Gao, Wentao Zhang, Junliang Yu, Yingxia Shao, Quoc Viet Hung Nguyen, Bin Cui, and Hongzhi Yin. 2024. Accelerating Scalable Graph Neural Network Inference with Node-Adaptive Propagation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*.

[31] Yuan Gao, Xiang Wang, Xiangnan He, Zhenguang Liu, Huamin Feng, and Yongdong Zhang. 2023. Addressing Heterophily in Graph Anomaly Detection: A Perspective of Graph Spectrum. In *Proceedings of the ACM Web Conference 2023*. ACM, Austin TX USA, 1528–1538. doi:10.1145/3543507.3583268

[32] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion Improves Graph Learning. In *32nd Advances in Neural Information Processing Systems*. arXiv:1911.05485

[33] Chenghua Gong, Yao Cheng, Xiang Li, Caihua Shan, Siqiang Luo, and Chuan Shi. 2024. Towards Learning from Graphs with Heterophily: Progress and Future. (Jan. 2024). arXiv:2401.09769 [cs]

[34] Rustam Guliyev, Aparajita Haldar, and Hakan Ferhatosmanoglu. 2024. D3-GNN: Dynamic Distributed Dataflow for Streaming Graph Neural Networks. *Proceedings of the VLDB Endowment* 17, 11 (July 2024), 2764–2777. doi:10.14778/3681954.3681961

[35] Yuhe Guo and Zhewei Wei. 2023. Clenshaw Graph Neural Networks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*. Association for Computing Machinery, New York, NY, USA, 614–625. doi:10.1145/3580305.3599275

[36] Yuhe Guo and Zhewei Wei. 2023. Graph Neural Networks with Learnable and Optimal Polynomial Bases. In *40th International Conference on Machine Learning*, Vol. 202. PMLR, Honolulu, Hawaii, USA. arXiv:2302.12432 [cs]

[37] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

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

[39] Mingguo He, Zhewei Wei, Zengfeng Huang, and Hongteng Xu. 2021. BernNet: Learning Arbitrary Graph Spectral Filters via Bernstein Approximation. In *34th Advances in Neural Information Processing Systems*.

[40] Mingguo He, Zhewei Wei, and Ji-Rong Wen. 2022. Convolutional Neural Networks on Graphs with Chebyshev Approximation, Revisited. In *35th Advances in Neural Information Processing Systems*.

[41] Weihua Hu, Matthias Fey, Hongyu Ren, Maho Nakata, Yuxiao Dong, and Jure Leskovec. 2021. OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs. *arXiv preprint arXiv:2103.09430* (2021).

[42] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, Jure Leskovec, Regina Barzilay, Peter Battaglia, Yoshua Bengio, Michael Bronstein, Stephan Günnemann, Will Hamilton, Tommi Jaakkola, Stefanie Jegelka, Maximilian Nickel, Chris Re, Le Song, Jian Tang, Max Welling, and Rich Zemel. 2020. Open Graph Benchmark: Datasets for Machine Learning on Graphs. *33rd Advances in Neural Information Processing Systems* (2020).

[43] Wei Hu, Jiahao Pang, Xianming Liu, Dong Tian, Chia-Wen Lin, and Anthony Vetro. 2021. Graph signal processing for geometric data and beyond: Theory and applications. *IEEE Transactions on Multimedia* 24 (2021), 3961–3977.

[44] Kezhao Huang, Haitian Jiang, Minjie Wang, Guangxuan Xiao, David Wipf, Xiang Song, Quan Gan, Zengfeng Huang, Jidong Zhai, and Zheng Zhang. 2024. FreshGNN: Reducing Memory Access via Stable Historical Embeddings for Graph Neural Network Training. *Proceedings of the VLDB Endowment* 17, 6 (Feb. 2024), 1473–1486. doi:10.14778/3648160.3648184

[45] Mingxuan Ju, Shifu Hou, Yujie Fan, Jianan Zhao, Yanfang Ye, and Liang Zhao. 2022. Adaptive kernel graph neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 7051–7058.

[46] Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

[47] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized PageRank. *7th International Conference on Learning Representations* (2019), 1–15.

[48] Yurui Lai, Xiaoyang Lin, Renchi Yang, and Hongtao Wang. 2024. Efficient Topology-aware Data Augmentation for High-Degree Graph Neural Networks. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. arXiv:2406.05482 [cs]

[49] Dongyue Li, Tao Yang, Lun Du, Zhezhi He, and Li Jiang. 2021. AdaptiveGCN: Efficient GCN Through Adaptively Sparsifying Graphs. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. ACM, Virtual Event Queensland Australia, 3206–3210. doi:10.1145/3459637.3482049

[50] Juanhui Li, Harry Shomer, Haitao Mao, Shenglai Zeng, Yao Ma, Neil Shah, Jiliang Tang, and Dawei Yin. 2023. Evaluating Graph Neural Networks for Link Prediction: Current Pitfalls and New Benchmarking. In *36th Advances in Neural Information Processing Systems*.

[51] Mingjie Li, Xiaojun Guo, Yifei Wang, Yisen Wang, and Zhouchen Lin. 2022. G^2CN: Graph Gaussian Convolution Networks with Concentrated Graph Filters. In *39th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (Eds.). PMLR, 12782–12796.

[52] Qimai Li, Zhichao Han, and Xiao Ming Wu. 2018. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (2018). arXiv:1801.07606

[53] Qimai Li, Xiao-Ming Wu, Han Liu, Xiaotong Zhang, and Zhichao Guan. 2019. Label Efficient Semi-Supervised Learning via Graph Filtering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Long Beach, CA, USA, 9574–9583. doi:10.1109/CVPR.2019.00981

[54] Xunkai Li, Jingyuan Ma, Zhengyu Wu, Daohan Su, Wentao Zhang, Rong-Hua Li, and Guoren Wang. 2024. Rethinking Node-Wise Propagation for Large-Scale Graph Learning. In *Proceedings of the ACM Web Conference 2024*.

[55] Xunkai Li, Zhengyu Wu, Wentao Zhang, Yinlin Zhu, Rong-Hua Li, and Guoren Wang. 2023. FedGTA: Topology-Aware Averaging for Federated Graph Learning. *Proceedings of the VLDB Endowment* 17, 1 (Sept. 2023), 41–50. doi:10.14778/3617838.3617842

[56] Yawei Li, He Chen, Zhaopeng Cui, Radu Timofte, Marc Pollefeys, Gregory S Chirikjian, and Luc Van Gool. 2021. Towards efficient graph convolutional networks for point cloud handling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3752–3762.

[57] Yiming Li, Yanyan Shen, Lei Chen, and Mingxuan Yuan. 2023. Zebra: When Temporal Graph Neural Networks Meet Temporal Personalized PageRank. *Proceedings of the VLDB Endowment* 16, 6 (2023), 1332–1345. doi:10.14778/3583140.3583150

[58] Jie Liao, Jintang Li, Liang Chen, Bingzhe Wu, Yatao Bian, and Zibin Zheng. 2023. SAILOR: Structural Augmentation Based Tail Node Representation Learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1389–1399.

[59] Ningyi Liao, Siqiang Luo, Xiang Li, and Jieming Shi. 2023. LD2: Scalable Heterophilous Graph Neural Network with Decoupled Embeddings. In *Advances in neural information processing systems,*, Vol. 36. 10197–10209.

[60] Ningyi Liao, Dingheng Mo, Siqiang Luo, Xiang Li, and Pengcheng Yin. 2022. SCARA: Scalable Graph Neural Networks with Feature-Oriented Optimization. *Proceedings of the VLDB Endowment* 15, 11 (2022), 3240–3248. doi:10.14778/3551793.3551866 arXiv:2207.09179

[61] Ningyi Liao, Dingheng Mo, Siqiang Luo, Xiang Li, and Pengcheng Yin. 2024. Scalable Decoupling Graph Neural Networks with Feature-Oriented Optimization. *The VLDB Journal* 33 (2024).

[62] Ningyi Liao, Zihao Yu, Ruixiao Zeng, and Siqiang Luo. 2025. Unifews: You Need Fewer Operations for Efficient Graph Neural Networks. In *42nd International Conference on Machine Learning* (Vancouver, Canada), Vol. 267. PMLR.

[63] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. 2021. Large Scale Learning on Non-Homophilous Graphs: New Benchmarks and Strong Simple Methods. In *34th Advances in Neural Information Processing Systems*.

[64] Haoyu Liu, Ningyi Liao, and Siqiang Luo. 2025. SIGMA: An Efficient Heterophilous Graph Neural Network with Fast Global Aggregation. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 1924–1937.

[65] Husong Liu, Shengliang Lu, Xinyu Chen, and Bingsheng He. 2020. G3: When Graph Neural Networks Meet Parallel Graph Processing Systems on GPUs. *Proceedings of the VLDB Endowment* 13, 12 (Aug. 2020), 2813–2816. doi:10.14778/3415478.3415482

[66] Haoyu Liu and Siqiang Luo. 2024. BIRD: Efficient Approximation of Bidirectional Hidden Personalized PageRank. *Proceedings of the VLDB Endowment* 17, 9 (2024), 2255–2268.

[67] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards Deeper Graph Neural Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, Virtual Event CA USA, 338–348. doi:10.1145/3394486.3403076

[68] Xin Liu, Mingyu Yan, Lei Deng, Guoqi Li, Xiaochun Ye, Dongrui Fan, Shirui Pan, and Yuan Xie. 2022. Survey on Graph Neural Network Acceleration: An Algorithmic Perspective. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*. arXiv:2202.04822 [cs]

[69] Yijing Liu, Qinxian Liu, Jian-Wei Zhang, Haozhe Feng, Zhongwei Wang, Zihan Zhou, and Wei Chen. 2022. Multivariate time-series forecasting with temporal polynomial graph neural networks. In *Advances in neural information processing systems*, Vol. 35. 19414–19426.

[70] Sitao Luan, Chenqing Hua, Qincheng Lu, Liheng Ma, Lirong Wu, Xinyu Wang, Minkai Xu, Xiao-Wen Chang, Doina Precup, Rex Ying, Stan Z. Li, Jian Tang, Guy Wolf, and Stefanie Jegelka. 2024. The Heterophilic Graph Learning Handbook: Benchmarks, Models, Theoretical Analysis, Applications and Challenges. arXiv:2407.09618 [cs]

[71] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2022. Revisiting Heterophily For Graph Neural Networks. In *36th Advances in Neural Information Processing Systems*.

[72] Sitao Luan, Mingde Zhao, Chenqing Hua, Xiao-Wen Chang, and Doina Precup. 2022. Complete the Missing Half: Augmenting Aggregation Filtering with Diversification for Graph Convolutional Networks. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*.

[73] Siqiang Luo, Ben Kao, Guoliang Li, Jiafeng Hu, Reynold Cheng, and Yudian Zheng. 2018. TOAIN: a throughput optimizing adaptive index for answering dynamic kNN queries on road networks. *Proceedings of the VLDB Endowment* 11, 5 (2018), 594–606. doi:10.1145/3177732.3177736

[74] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. 2024. Classic GNNs are Strong Baselines: Reassessing GNNs for Node Classification. In *37th Advances in Neural Information Processing Systems*. arXiv:2406.08993

[75] Lu Ma, Zeang Sheng, Xunkai Li, Xinyi Gao, Zhezheng Hao, Ling Yang, Wentao Zhang, and Bin Cui. 2024. Acceleration Algorithms in GNNs: A Survey. (May 2024). arXiv:2405.04114 [cs]

[76] Yuxin Ma, Ping Gong, Tianming Wu, Jiawei Yi, Chengru Yang, Cheng Li, Qirong Peng, Guiming Xie, Yongcheng Bao, Haifeng Liu, and Yinlong Xu. 2024. Eliminating Data Processing Bottlenecks in GNN Training over Large Graphs via Two-level Feature Compression. *Proceedings of the VLDB Endowment* 17, 11 (July 2024), 2854–2866. doi:10.14778/3681954.3681968

[77] Seiji Maekawa, Koki Noda, Yuya Sasaki, et al. 2022. Beyond real-world benchmark datasets: An empirical study of node classification with GNNs. *Advances in Neural Information Processing Systems* 35 (2022), 5562–5574.

[78] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. 2022. Simplifying Approach to Node Classification in Graph Neural Networks. *Journal of Computational Science* 62 (July 2022), 101695. doi:10.1016/j.jocs.2022.101695

[79] Xupeng Miao, Hailin Zhang, Yining Shi, Xiaonan Nie, Zhi Yang, Yangyu Tao, and Bin Cui. 2021. HET: Scaling out Huge Embedding Model Training via Cache-enabled Distributed Framework. *Proceedings of the VLDB Endowment* 15, 2 (2021), 312–320. doi:10.14778/3489496.3489511 arXiv:2112.07221

[80] Chen Ming, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and Deep Graph Convolutional Networks. In *37th International Conference on Machine Learning*, Vol. 119. PMLR, 1703–1713. arXiv:2007.02133

[81] Hoang Nt and Takanori Maehara. 2019. Revisiting graph neural networks: All we have is low-pass filters. (2019). arXiv:1905.09550 [cs]

[82] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report.

[83] Yeonhong Park, Sunhong Min, and Jae W. Lee. 2022. Ginex: SSD-Enabled Billion-Scale Graph Neural Network Training on a Single Machine via Provably Optimal in-Memory Caching. *Proceedings of the VLDB Endowment* 15, 11 (July 2022), 2626–2639. doi:10.14778/3551793.3551819

[84] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. *International Conference on Learning Representations* (2020).

[85] Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. 2023. A Critical Look at Evaluation of GNNs under Heterophily: Are We Really Making Progress?. In *11th International Conference on Learning Representations*.

[86] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M. Tamer O¨zsu. 2018. The Ubiquity of Large Graphs and Surprising Challenges of Graph Processing. *Proceedings of the VLDB Endowment* 11, 4 (2018), 420–431. doi:10.1145/3164135.3164139

[87] Kaan Sancak, Muhammed Fatih Balin, and Umit Catalyurek. 2024. Do We Really Need Complicated Graph Learning Models? – A Simple but Effective Baseline. In *The 3rd Learning on Graphs Conference*.

[88] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Magazine* 29, 3 (Sep. 2008), 93.

[89] Yanyan Shen, Lei Chen, Jingzhi Fang, Xin Zhang, Shihong Gao, and Hongbo Yin. 2024. Efficient Training of Graph Neural Networks on Large Graphs. *Proceedings of the VLDB Endowment* 17, 12 (2024), 4237–4240. doi:10.14778/3685800.3685844

[90] Zeang Sheng, Wentao Zhang, Yangyu Tao, and Bin Cui. 2024. OUTRE: An OUT-of-Core De-REdundancy GNN Training Framework for Massive Graphs within A Single Machine. *Proceedings of the VLDB Endowment* 17, 11 (July 2024), 2960–2973. doi:10.14778/3681954.3681976

[91] Zhen Song, Yu Gu, Tianyi Li, Qing Sun, Yanfeng Zhang, Christian S. Jensen, and Ge Yu. 2023. ADGNN: Towards Scalable GNN Training with Aggregation-Difference Aware Sampling. *Proceedings of the ACM on Management of Data* 1, 4 (Dec. 2023), 1–26. doi:10.1145/3626716

[92] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Yiqi Wang, Jiliang Tang, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. 2020. Investigating and Mitigating Degree-Related Biases in Graph Convoltuional Networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1435–1444.

[93] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. In *8th International Conference on Learning Representations*.

[94] Xinchen Wan, Kaiqiang Xu, Xudong Liao, Yilun Jin, Kai Chen, and Xin Jin. 2023. Scalable and Efficient Full-Graph GNN Training for Large Graphs. *Proceedings of the ACM on Management of Data* 1, 2 (June 2023), 1–23. doi:10.1145/3589288

[95] Hanzhi Wang, Mingguo He, Zhewei Wei, Sibo Wang, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. 2021. Approximate Graph Propagation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. ACM, 1686–1696. doi:10.1145/3447548.3467243

[96] Lu Wang, Wenchao Yu, Wei Wang, Wei Cheng, Wei Zhang, Hongyuan Zha, Xiaofeng He, and Haifeng Chen. 2019. Learning Robust Representations with Graph Denoising Policy Network. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, Beijing, China, 1378–1383. doi:10.1109/ICDM.2019.00177

[97] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. 2019. Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks. (2019). arXiv:1909.01315

[98] Xiyuan Wang and Muhan Zhang. 2022. How Powerful are Spectral Graph Neural Networks. In *Proceedings of the 39th International Conference on Machine Learning*, Vol. 162. PMLR, 23341–23362.

[99] Yifei Wang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. 2021. Dissecting the Diffusion Process in Linear Graph Convolutional Networks. In *Advances in Neural Information Processing Systems*, Vol. 34. 5758–5769.

[100] Isaac Ronald Ward, Jack Joyner, Casey Lickfold, Yulan Guo, and Mohammed Bennamoun. 2022. A Practical Tutorial on Graph Neural Networks. *Comput. Surveys* 54, 10s (Jan. 2022), 1–35. doi:10.1145/3503043

[101] Ryan Wickman, Xiaofei Zhang, and Weizi Li. 2022. A Generic Graph Sparsification Framework using Deep Reinforcement Learning. In *2022 IEEE International Conference on Data Mining (ICDM)*. 1221–1226. doi:10.1109/ICDM54844.2022.00158

[102] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying Graph Convolutional Networks. In *Proceedings of the 36th International Conference on Machine Learning*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.), Vol. 97. 6861–6871.

[103] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2021. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems* 32, 1 (Jan. 2021).

[104] Keyulu Xu, Stefanie Jegelka, Weihua Hu, and Jure Leskovec. 2019. How Powerful Are Graph Neural Networks?. In *7th International Conference on Learning Representations*. 1–17. arXiv:1810.00826

[105] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. 2022. Two Sides of the Same Coin: Heterophily and Oversmoothing in Graph Convolutional Neural Networks. In *22nd IEEE International Conference on Data Mining*. arXiv:2102.06462 [cs]

[106] Liang Yang, Chuan Wang, Junhua Gu, Xiaochun Cao, and Bingxin Niu. 2021. Why Do Attributes Propagate in Graph Convolutional Neural Networks? *Proceedings of the AAAI Conference on Artificial Intelligence* 35, 5 (May 2021), 4590–4598. doi:10.1609/aaai.v35i5.16588

[107] Mingqi Yang, Yanming Shen, Rui Li, Heng Qi, Qiang Zhang, and Baocai Yin. 2022. A New Perspective on the Effects of Spectrum in Graph Neural Networks. In *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 25261–25279.

[108] Haoteng Yin, Muhan Zhang, Yanbang Wang, Jianguo Wang, and Pan Li. 2022. Algorithm and System Co-design for Efficient Subgraph-based Graph Representation Learning. *Proceedings of the VLDB Endowment* 15, 11 (2022), 2788–2796.

[109] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom). 974–983.

[110] Minji Yoon, Théophile Gervet, Baoxu Shi, Sufeng Niu, Qi He, and Jaewon Yang. 2021. Performance-Adaptive Sampling Strategy Towards Fast and Accurate Graph Neural Networks. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. ACM, Virtual Event Singapore, 2046–2056. doi:10.1145/3447548.3467284

[111] Zihao Yu, Ningyi Liao, and Siqiang Luo. 2024. GENTI: GPU-Powered Walk-Based Subgraph Extraction for Scalable Representation Learning on Dynamic Graphs. *Proceedings of the VLDB Endowment* 17, 9 (May 2024), 2269–2278.

[112] Hao Yuan, Yajiong Liu, Yanfeng Zhang, Xin Ai, Qiange Wang, Chaoyi Chen, Yu Gu, and Ge Yu. 2024. Comprehensive Evaluation of GNN Training Systems: A Data Management Perspective. *Proceedings of the VLDB Endowment* 17, 6 (Feb. 2024), 1241–1254. doi:10.14778/3648160.3648167

[113] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. 2019. GraphSAINT: Graph Sampling Based Learning Method. In *International Conference on Learning Representations*.

[114] Dalong Zhang, Xin Huang, Ziqi Liu, Jun Zhou, Zhiyang Hu, Xianzheng Song, Zhibang Ge, Lin Wang, Zhiqiang Zhang, and Yuan Qi. 2020. AGL: A Scalable System for Industrial-purpose Graph Machine Learning. *Proceedings of the VLDB Endowment* 13, 12 (2020), 3125–3137. doi:10.14778/3415478.3415539

[115] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems* 31 (2018), 5165–5175.

[116] Shichang Zhang, Atefeh Sohrabizadeh, Cheng Wan, Zijie Huang, Ziniu Hu, Yewen Wang, Yingyan, Lin, Jason Cong, and Yizhou Sun. 2023. A Survey on Graph Neural Network Acceleration: Algorithms, Systems, and Customized Hardware. (June 2023). arXiv:2306.14052 [cs]

[117] Wentao Zhang, Zhi Yang, Yexin Wang, Yu Shen, Yang Li, Liang Wang, and Bin Cui. 2021. Grain: Improving Data Efficiency of Graph Neural Networks via Diversified Influence Maximization. *Proceedings of the VLDB Endowment* 14, 11 (July 2021), 2473–2482. doi:10.14778/3476249.3476295 arXiv:2108.00219

[118] Xin Zhang, Yanyan Shen, Yingxia Shao, and Lei Chen. 2023. DUCATI: A Dual-Cache Training System for Graph Neural Networks on Giant Graphs with the GPU. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–24.

[119] Ziwei Zhang, Peng Cui, Jian Pei, Xin Wang, and Wenwu Zhu. 2021. Eigen-GNN: a Graph Structure Preserving Plug-in for GNNs. *IEEE Transactions on Knowledge and Data Engineering* (2021), 1–1. doi:10.1109/TKDE.2021.3112746

[120] Ziwei Zhang, Peng Cui, and Wenwu Zhu. 2020. Deep Learning on Graphs: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 14, 8 (2020).

[121] Zaixi Zhang, Qi Liu, Qingyong Hu, and Chee-Kong Lee. 2022. Hierarchical graph transformer with adaptive node sampling. *Advances in Neural Information Processing Systems* 35 (2022), 21171–21183.

[122] Chenguang Zheng, Hongzhi Chen, Yuxuan Cheng, Zhezheng Song, Yifan Wu, Changji Li, James Cheng, Hao Yang, and Shuai Zhang. 2022. ByteGNN: Efficient Graph Neural Network Training at Large Scale. *Proceedings of the VLDB Endowment* 15, 6 (Feb. 2022), 1228–1242. doi:10.14778/3514061.3514069

[123] Yanping Zheng, Zhewei Wei, and Jiajun Liu. 2023. Decoupled Graph Neural Networks for Large Dynamic Graphs. *Proceedings of the VLDB Endowment* 16, 9 (May 2023), 2239–2247. doi:10.14778/3598581.3598595

[124] Hongkuan Zhou, Ajitesh Srivastava, Hanqing Zeng, Rajgopal Kannan, and Viktor Prasanna. 2021. Accelerating Large Scale Real-Time GNN Inference Using Channel Pruning. *Proceedings of the VLDB Endowment* 14, 9 (2021), 1597–1605. doi:10.14778/3461535.3461547 arXiv:2105.04528

[125] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph Neural Networks: A Review of Methods and Applications. *AI Open* 1 (2020), 57–81.

[126] Hao Zhu and Piotr Koniusz. 2021. Simple Spectral Graph Convolution. In *9th International Conference on Learning Representations*.

[127] Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. 2021. Interpreting and Unifying Graph Neural Networks with An Optimization Framework. In *Proceedings of the Web Conference 2021*. ACM, 1215–1226.

[128] Rong Zhu, Kun Zhao, Hongxia Yang, Wei Lin, Chang Zhou, Baole Ai, Yong Li, and Jingren Zhou. 2019. Aligraph: A comprehensive graph neural network platform. *Proceedings of the VLDB Endowment* 12, 12 (Aug. 2019), 2094–2105. doi:10.14778/3352063.3352127

Received January 2025; revised April 2025; accepted May 2025