Clenshaw Graph Neural Networks

Yuhe Guo
Renmin University of China
Beijing, China
guoyuhe@ruc.edu.cn

Zhewei Wei[*]
Renmin University of China
Beijing, China
zhewei@ruc.edu.cn

ABSTRACT

Graph Convolutional Networks (GCNs), which use a message-passing paradigm with stacked convolution layers, are foundational **spatial** methods for learning graph representations. Polynomial filters, which have an advantage on heterophilous graphs, are motivated differently from the **spectral** perspective of graph convolutions. Recent spatial GCN models use various residual connection techniques to alleviate the model degradation problem such as over-smoothing and gradient vanishing. However, current residual connections do not effectively harness the full potential of polynomial filters, which are commonly employed in the spectral domain of GNNs.

In this paper, we introduce ClenshawGCN, a GNN model that injects the characteristic of spectral models into spatial models by a simple residual connection submodule: the Clenshaw residual connection, which is essentially a second-order negative residual combined with an initial residual. We show that a ClenshawGCN implicitly simulates an arbitrary polynomial filter under the Chebyshev basis, since the iteration process of stacked (spatial) convolutions equipped with Clenshaw residuals can be interpreted by Clenshaw Summation Algorithm. In addition, we conduct comprehensive experiments to demonstrate the superiority of our model over spatial and spectral GNN models. Our implementation is at https://github.com/yuziGuo/KDDClenshawGNN.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**.

KEYWORDS

Graph Neural Networks, Residual Connection, Graph Polynomial Filter

ACM Reference Format:
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), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599275

[*]Zhewei Wei is the corresponding author. The work was partially done at Gaoling School of Artificial Intelligence, Peng Cheng Laboratory, Beijing Key Laboratory of Big Data Management and Analysis Methods and MOE Key Lab of Data Engineering and Knowledge Engineering.

1 INTRODUCTION

The past few years have witnessed the rise of machine learning on graphs, which considers relations (edges) between elements (nodes) such as interactions among molecules [10, 38], friendship or hostility between users [11, 51], and implicit syntactic or semantic structure in natural language [39, 50]. Existing GNN models can be broadly divided into two categories: spatial GNNs and spectral GNNs.

Spatial GNNs [1, 20, 44, 58] employ a message-passing paradigm to exploit the underlying graph topology by propagating node features iteratively along the edges. Along with the message-passing steps, each node receives information from glowingly expanding neighborhoods. Such a *propagation* entangled with non-linear *transformation* forms a (spatial) *graph convolution layer* in GCN.

Spectral GNNs [13, 41] define the Graph Convolution Operator using the idea of Graph Fourier Transformation from Graph Signal Processing. For computational traceability, most of the practical spectral GNNs [6, 9, 15, 16, 47] use various *polynomials filters* to approximate the Graph Convolution Operator. The **two main features shared by state-of-art polynomial filters** [15, 16, 47] are: (1) the ability to represent *arbitrary* functions; (2) the utilization of *polynomial basis* for better approximation. Spectral GNNs have a natural advantage on *heterophilous graphs*, where the *homophily assumption* (*i.e.*, connected nodes tend to have similar features or same class labels) does not stand. For a more concrete background of Spectral GNNs, please check Section 2.3.

Residual Connection in GNNs. One interesting and widely-used submodule that often appears with graph convolution layers is the residual connections, which is typically used to alleviate the *model degradation* problem that occurs in deep models [14]. The mechanism of residual submodules can be simply described as taking use of (typically by an *add* operation) the representations of the previous layers when iteratively obtaining representations for current layers. As shown Figure 1, we raw categorize residual connections in GNNs into three types: (1) *raw residuals* [20] that are transplanted directly from ResNet [20] denoted roughly as $x^{(\ell+1)} = f(x^{(\ell)}) + x^{(\ell)}$; (2) *initial residuals* [5, 21, 27, 55] adding back to the *first*-layer representation, denoted roughly as $x^{(\ell+1)} = f(x^{(\ell)}) + x^{(0)}$, and (3) *dense residuals* [1, 52] that connect the current layers densely to *all* previous layers. Please check Section 2.4 for a fuller discussion about existing graph residuals.

Motivations. Existing residual connections build a relationship between the spatial (message passing) and spectral (polynomial filtering) domains of GNNs. For example, it has been demonstrated that initial residual connections [5] mimic polynomial filtering in the form of Personalized PageRank [35]. However, raw residuals

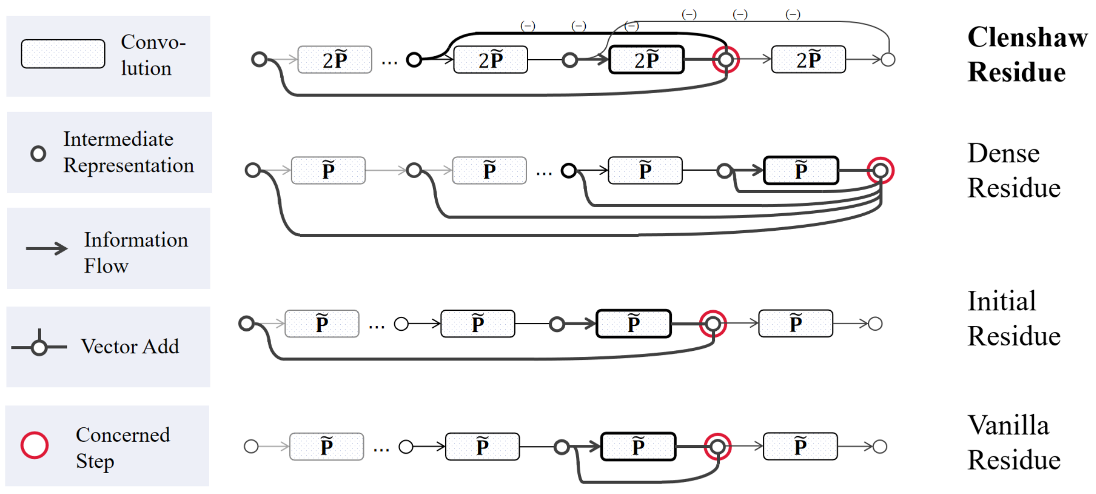

Figure 1: Illustration of different graph residual connections. *Raw Residue* and *Initial Residue* follows a single-line structure, *Dense Residue* connect densely to all the former layers, while our *Clenshaw Residue* are able to approximate arbitrary graph filters via Chebyshev polynomials by a neat *double-line* structure.

and initial residuals are primarily used to mitigate over-smoothing; Their corresponding polynomials belong to the category of *low-pass polynomial filters*, which do not perform well on *heterophilic graphs*.

On the other hand, dense residuals, such as MixHop [1] and JKNet [52], do have the ability to express arbitrary polynomial filters. However, their overly complex structure makes it more difficult to learn the appropriate polynomial filter from the training data, which is a crucial ability possessed by recent spectral models like GPRGNN [6] and ChebNetII [16]. As a result, on heterophilous graphs, these models also do not exhibit comparable performances to spectral models. A question rises: **can we build a *simple* residual connection that can inject the key characteristics of polynomial filters into a spatial model?**

Our Solution: Clenshaw Residual Connections. In this paper, we provide a positive answer to the above question by proposing the Clenshaw Residual Connection:

$$x^{(\ell+1)} = 2f(x^{(\ell)}) - x^{(\ell-1)} + \alpha_\ell x^{(0)}.$$

Equipped with Clenshaw residuals, the iteration process of stacked (spatial) convolutions can be interpreted by *Clenshaw Summation Algorithm*, an iterative method for evaluating the weighted sum over a kind of polynomial bases. Clenshaw algorithm reveals that our proposed spatial model (termed ClenshawGNN in the rest of our paper) can express *arbitrary spectral-domain polynomial filters based on the Chebyshev Basis (the second kind)*.

Contributions. We summarize our contributions as follows:

- **Simple and novel residual connection submodules**. We propose ClenshawGCN, a message-passing GNN which borrows spectral power by adding a simple residual connection module inspired by the Clenshaw algorithm. To the best of our knowledge, we are the first to directly use negative residuals;
- **Ability to approximate arbitrary graph filter via Chebyshev polynomials.** We show that a K-order ClenshawGCN simulates *arbitrary* K-order polynomial function based on the *Chebyshev basis of the Second Kind*. Especially, the *negative second order residue* allows for the implicit use of the Chebyshev basis. We prove this by the Clenshaw Summation Algorithm for the Chebyshev basis (the Second Kind).
- **Superior empirical performances**. We compare ClenshawGCN with both *spatial* models with representative residual connections and state-of-art *spectral* models by extensive empirical studies. We achieve new top performances on two large non-homophilous datasets [26]. The results reveal that our motivation is realized: our model benefits from both sides. Compared to spatial models, our proposed model inherits spectral models' superiority on heterophilous datasets; Compared to spectral models, our model benefits from the layer-wise entangled features transformation as a spatial model.

2 PRELIMINARIES

2.1 Notations

We consider a simple graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$, where $\mathcal{V} = \{1, 2, \cdots, n\}$ is a finite node set with $|\mathcal{V}| = n$, $\mathcal{E}$ is an edge set with $|\mathcal{E}| = m$, and $\mathbf{A}$ is the unnormalized adjacency matrix. $\mathcal{L} = \mathbf{D} - \mathbf{A}$ denotes $\mathcal{G}'s$ unnormalized graph Laplacian, where $\mathbf{D} = \text{diag}\{d_1, \cdots, d_n\}$ is the degree matrix with $d_i = \sum_j \mathbf{A}_{ij}$.

Following GCN, we add a self-connecting edge to each node, and conduct symmetric normalization on $\mathcal{L}$ and $\mathbf{A}$. The resultant *self-looped symmetric-normalized* adjacency matrix and Laplacian are denoted as $\tilde{\mathbf{P}} = (\mathbf{D}+\mathbf{I})^{-1/2}(\mathbf{A}+\mathbf{I})(\mathbf{D}+\mathbf{I})^{-1/2}$ and $\tilde{\mathcal{L}} = \mathbf{I} - \tilde{\mathbf{P}}$, respectively.

Further, we attach each node with an f-dimensional raw feature and denote the feature matrix as $\mathbf{X} \in \mathbb{R}^{n \times f}$. Based on the topology of the underlying graph, GNNs enhance the raw node features to better representations for downstream tasks, such as node classification and link prediction.

2.2 Spatial Background

Layer-wise Message Passing Architecture. From a spatial view, the main body of a Graph Neural Network is a stack of *convolution* layers; they broadcast and aggregate features along the edges. Such a graph neural network is also called a Message Passing Neural Network (MPNN). To be concrete, we consider an MPNN with K graph convolution layers and denote the nodes' representations of the ℓ-th layer as $\mathbf{H}^{(\ell)}$. $\mathbf{H}^{(\ell)}$ is constructed based on $\mathbf{H}^{(\ell-1)}$ by a *propagation* and possibly a *transformation* operation. For example, in vanilla GCN [20], a convolution layer is defined as

$$\mathbf{H}^{(\ell)} = \sigma\Big(\tilde{\mathbf{P}}\mathbf{H}^{(\ell-1)}\mathbf{W}^{(\ell)}\Big), \tag{1}$$

whose *propagation* operator is $f: \mathbf{H}^{(\ell-1)} \rightarrow \tilde{\mathbf{P}}\mathbf{H}^{(\ell-1)}$ and *transformation* operator is $f: \tilde{\mathbf{P}}\mathbf{H}^{(\ell-1)} \rightarrow \sigma(\tilde{\mathbf{P}}\mathbf{H}^{(\ell-1)}\mathbf{W}^{(\ell)}))$.

Extra transformations, probably with poolings or concatenations over $\{\mathbf{H}^{(\ell)}\}_{\ell=0}^{K}$ are applied before and after the stack of convolutions to form a map from $\mathbf{X}$ to the final output $\hat{\mathbf{Y}}$, which is determined by the downstream task, *e.g.*, $\mathbf{H}^{(0)} = \mathrm{MLP}(\mathbf{X}; \mathbf{W}^{(0)})$ and $\hat{\mathbf{Y}} = \mathrm{SoftMax}(\mathrm{MLP}(\mathbf{H}^{(K)}; \mathbf{W}^{(K+1)}))$ for node classification tasks. In this paper, a slight difference in notation lies in that, we denote the input of convolution layers to be $\mathbf{H}^{*}$, instead of $\mathbf{H}^{(0)}$, and introduce notations $\mathbf{H}^{(-1)}$ and $\mathbf{H}^{(-2)}$ as zero matrices for simplicity of later presentation.

Entangled and Disentangled Architectures. The motivations for *propagation* and *transformation* in a convolution layer differ. *Propagations* are related to graph topology, analyzed as an analog of walks [5, 21, 46, 52], diffusion processes [4, 22, 56], *etc.* , while the *entangling* of *transformations* between *propagations* follows the convention of deep learning. A GNN model is of a disentangled architecture if the *transformation* operations are separate from the *propagation* operations, *e.g.*, APPNP [21], GPRGNN [6] and other polynomial filters. Though it is raised in [54] that the entangled architecture might cause model degradations, we observe that GCNII [5] under the entangled architecture does not suffer from model degradation and even benefits from the entangled non-linear transformations. Therefore, in our work, we adopt an entangled architecture, and simply leverage the identity mapping of weight matrices as in GCNII.

2.3 Spectral Background

Spectral Definition of Convolution. Graph spectral domain leverages the geometric structure of underlying graphs in another way [41]. Conducting eigen-decomposition on $\tilde{\mathcal{L}}$, *i.e.*, $\tilde{\mathcal{L}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top}$, with the spectrum $\mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, \cdots, \lambda_n\}$ in non-decreasing order, since $\tilde{\mathcal{L}}$ is real-symmetric, elements in $\mathbf{\Lambda}$ are real, and $\mathbf{U}$ is a complete set of n orthonormal eigenvectors. The spectral domain utilizes $\mathbf{U}$ as a basis of *frequency components* analogously to *classic Fourier transform.*

Now consider a column in $\mathbf{X}$ as a *graph signal* scattered on $\mathcal{V}$, denoted as $\mathbf{x} \in \mathbb{R}^n$. *Graph Fourier transform* is defined as $\hat{\mathbf{x}} := \langle \mathbf{U}, \mathbf{x} \rangle = \mathbf{U}^{\top}\mathbf{x}$, which projects graph signal $\mathbf{x}$ to the *frequency responses* of basis components $\hat{\mathbf{x}}$. It is then followed by *modulation*, which can be presented as $\hat{\mathbf{x}}^{*} := g_{\theta}\hat{\mathbf{x}} = \mathrm{diag}\{\theta_1, \cdots \theta_n\}\hat{\mathbf{x}}$. After modulation, *inverse Fourier transform*: $\mathbf{x}^{*} := \mathbf{U}\hat{\mathbf{x}}^{*}$ transform $\hat{\mathbf{x}}^{*}$ back to the spatial domain. The three operations form a spectral definition of *convolution*:

$$g_{\theta} \star \mathbf{x} = \mathbf{U}g_{\theta}\mathbf{U}^{\top}\mathbf{x} = \mathbf{U}\mathrm{diag}\{\theta_1, \cdots \theta_n\}\mathbf{U}^{\top}\mathbf{x}, \tag{2}$$

which is also called *spectral filtering*. Specifically, when $\theta_i = 1 - \lambda_i$, $\mathbf{U}g_{\theta}\mathbf{U}^{T}\mathbf{x} \equiv \tilde{\mathbf{P}}\mathbf{x}$, giving a spectral explanation of GCN's convolution in Equation (1).

Polynomial Filtering. The calculation of $\mathbf{U}$ in Equation (2) is of prohibitively expensive. To avoid explicit eigen-decomposition of $\mathbf{U}$, θ_i is approximated by a polynomial function of λ_i, that is,

$$\hat{\mathbf{x}}_i^{*} = g_{\theta}(\lambda_i)\langle \mathbf{U}_i, \mathbf{x} \rangle.$$

The spectral filtering process then becomes

$$g_{\theta} \star \mathbf{x} = \mathbf{U}g_{\theta}(\mathbf{\Lambda})\mathbf{U}^{T}\mathbf{x} \equiv g_{\theta}(\tilde{\mathcal{L}})\mathbf{x}, \tag{3}$$

where the calculation of $g_{\theta}(\tilde{\mathcal{L}})\mathbf{x}$ eliminates eigen-decomposition and can be calculated in a localized way in $O(|\mathcal{E}|)$ [9, 20].

Definition 2.1 (Polynomial Filters). Consider a graph whose Laplacian matrix is $\tilde{\mathcal{L}}$. By adopting the set of orthonormal eigenvectors of $\tilde{\mathcal{L}}$ as the frequency basis, a **polynomial filter** is a process or operator that scales each frequency component of the input signal by $g_{\theta}(\lambda)$, where g_{θ} is a polynomial filtering function and λ is the corresponding eigenvalue of the frequency component.

Equivalently, we can define the filtering function on the spectrum of $\tilde{\mathbf{P}}$, instead of $\tilde{\mathcal{L}}$. Since $\tilde{\mathbf{P}} = \mathbf{I} - \tilde{\mathcal{L}}$, $\tilde{\mathbf{P}}$ and $\tilde{\mathcal{L}}$ share the same set of orthonormal eigenvectors $\mathbf{U}$, and the spectrum of $\tilde{\mathbf{P}}$, denoted as $\mathbf{M} = \{\mu_1, \cdots, \mu_n\}$, satisfies $\mu_i = 1 - \lambda_i (i = 1, \cdots, n)$. Thus, the filtering function can be defined as h_{θ}, where $h_{\theta}(\mu) \equiv g_{\theta}(1 - \mu)$. To ensure a concise presentation, we shall employ this equivalent definition in Section 3.

Advantage On Heterophilous Graphs. Spectral graph neural networks have the ability to strengthen desired frequency components in $\mathbf{U}$ by g_{θ}. Since they can strengthen high-frequency components which capture the differences between nodes, favorable performances can be achieved on heterophilous graphs [30, 57]. On the contrary, some spatial GNNs are proven to be fixed low-pass filters [34].

Polynomial Approximation Using Bases. A line of work approximate g_{θ} over some polynomial basis up to the truncated K-th order, *i.e.*, $g_{\theta}(x) = \sum_{k=0}^{K} \theta_i \phi_i(x)$, where $\vec{\theta} = [\theta_0, \cdots, \theta_K] \in \mathbb{R}^{K+1}$ is the coefficients. In the field of polynomial filtering and spectral GNNs, different bases have been explored for $\{\phi_i(x)\}_{i=0}^{i=K}$, including Chebyshev basis [9, 16], Bernstein basis [15], Jacobi basis [47], *etc.*

Chebyshev Basis. Chebyshev basis, which has been extensively explored [31], has appeared in related literature since early attempts for the approximation of g_θ [9] or kernels in Graph signal Processing [13]. Besides Chebyshev polynomials of the first kind ($\{T_i(x)\}_{i=0}^{\infty}$), the second kind ($\{U_i(x)\}_{i=0}^{\infty}$) is also wildly used. Both of them can be generated by a *recurrence relation*:

$$\begin{aligned}&T_0(x)=1, \quad T_1(x)=x,\\&T_n(x)=2xT_{n-1}(x)-T_{n-2}(x). \quad (n=2,3,\cdots)\end{aligned}$$

$$\begin{aligned}&U_{-1}(x)=0, \quad U_0(x)=1, U_1(x)=2x,\\&U_n(x)=2xU_{n-1}(x)-U_{n-2}(x). \quad (n=1,2,\cdots)\end{aligned} \tag{4}$$

In this paper, we will show the relationship between our negative residual and the Chebyshev polynomials of the *second kind* using this recurrence relation. A useful characteristic we will make use of in the proof in Appendix A is that the recurrence starts from U_{-1}.

2.4 Graph Residual Connections

As shown in Figure 1, we broadly classify the graph residual connections into three types.

Raw Residual Connections. Residual connections were initially introduced to graph neural networks to alleviate the issue of model degradation problem, which refers to the drop of GCN performance as the network depth increases. This degradation hinders GCN from the effective leverage of larger neighborhoods. Kipf and Welling [20] directly transplant the raw residual connection utilized in ResNet [14] to GCN as a variant model, denoted as $\mathbf{H}^{(\ell)}=\sigma(\tilde{\mathbf{P}}\mathbf{H}^{(\ell-1)}\mathbf{W}^{(\ell)})+\mathbf{H}^{(\ell-1)}$. However, despite the raw residual can alleviate model degradation for GCN up to 10 layers, it gradually loses its effectiveness as models become even deeper, as evidenced by follow-up studies [5, 52]. The reason behind this was gradually elucidated through discussions on the concept of <u>over-smoothing</u> [24, 37, 46].

Initial Residual Connections. A line of discussion regarding over-smoothing view GCNs equipped with raw residual connections as *lazy random walks* [5, 46, 52], which still converge to the *stationary vector*. Building upon this understanding, the initial residual connection was proposed to simulate *personalized random walks* [35]. As an example, the **GCNII** model is denoted as [1]:

$$\mathbf{H}^{(\ell)}=\sigma\Big(\Big((1-\alpha)\tilde{\mathbf{P}}\mathbf{H}^{(\ell-1)}+\alpha\mathbf{H}^{*}\Big)\Big((1-\beta_\ell)\mathbf{I}_n+\beta_\ell\mathbf{W}^{(\ell)}\Big)\Big). \tag{5}$$

Zhu et al. [59] interprets the initial residual connection utilized in GCNII from an optimization perspective, suggesting that by disregarding the entangled non-linear transformation, GCNII can iteratively solve the optimization problem:

$$\arg\min_{\mathbf{H}} \alpha\left\|\mathbf{H}-\mathbf{H}^*\right\|_F^2+(1-\alpha)\operatorname{tr}\left(\mathbf{H}^\top\tilde{\mathcal{L}}\mathbf{H}\right). \tag{6}$$

The optimization problem reveals that the initial residual connection in GCNII achieves a *compromise* between Laplacian smoothing (the second term, promoting similarity among neighboring nodes) and preserving individual characteristics (the first term). While this compromise successfully addresses over-smoothing, GCNs with initial residuals still fall within the realm of the *homophily assumption* [32].

AirGNN[27] introduces an extension for the initial residual connection, which is derived from a similar optimization problem as shown in Equation(6), but with the first term replaced by utilizing the ℓ_{21} norm. By employing the proximal gradient technique to solve the optimization problem, AirGNN obtains a suitable value of α for each step. However, due to the inherent limitations imposed by the *predetermined optimization objective*, AirGNN also inherently falls within the scope of the homophily assumption.

Furthermore, the initial residual connection has gained widespread usage as a submodule in various models, such as GloGNN [25] and GAMLP [55]. As these models extend beyond the basic graph convolution framework, we have chosen to omit the discussion of these models in our paper.

Dense Residual Connections. These studies exploit feature maps after different times of convolutions and combine them selectively to make extensive use of multi-scale information [1, 52]. For instance, **JKNet** [52] employs a dense residual connection *at the final layer* and combines all intermediate representations in various ways, such as weighted-sum, max-pooling, concatenation, LSTM, and more. **MixHop** [1] concatenates feature maps from multiple hops *at each layer*, denoted as $\mathbf{H}^{(\ell+1)}=\Big\Vert_{j\in K}\sigma\left(\tilde{\mathbf{P}}^j\mathbf{H}^{(\ell)}\mathbf{W}_j^{(\ell)}\right)$, which can be considered as staking several dense graph residual networks.

Besides, Li et al. [23] explore the connection pattern inspired by DenseNet [18], where each layer is connected to all former layers. However, this approach results in unaffordable space consumption.

3 METHOD

3.1 Clenshaw Convolution

We formally pose the ℓ-th layer's representation of ClenshawGCN as

$$\mathbf{H}^{(\ell)}=\sigma\Bigg(\Big(2\tilde{\mathbf{P}}\mathbf{H}^{(\ell-1)}\underbrace{-\mathbf{H}^{(\ell-2)}}_{\substack{\textbf{Negative Second}\\\textbf{Order Residue}}}+\underbrace{\alpha_\ell\mathbf{H}^*}_{\substack{\text{Initial}\\\text{Residue}}}\Big)\cdot\underbrace{\Big((1-\beta_\ell)\mathbf{I}_n+\beta_\ell\mathbf{W}^{(\ell)}\Big)}_{\substack{\text{Identity Mapping}\\\text{from GCNII [5]}}}\Bigg), \tag{7}$$

for $\ell=0,1,\ldots,K$, $\mathbf{H}^{(-2)}=\mathbf{H}^{(-1)}=\mathbf{O}$, $\mathbf{H}^*=\mathrm{MLP}(\mathbf{X};\mathbf{W}^*)$.

We adopt an entangled architecture for Clenshaw convolution, that is, each convolution layer is composed of a propagation operation and a transformation operation. For the propagation part, we enhance the conventional convolution with the **Clenshaw residue**, which is composed of a **Negative Second Order Residue** and an auxiliary *initial residue*. The detailed mechanism of Clenshaw residue will be elucidated in the next Section.

As for the transformation part, we incorporate the *identity mapping* from GCNII, where the weight matrice for feature transformation are defined as $\{(1-\beta_\ell)\mathbf{I}_n+\beta_\ell\mathbf{W}^{(\ell)}\}_{\ell=0}^{K}$ is used instead of $\{\mathbf{W}^{(\ell)}\}_{\ell=0}^{K}$. Following GCNII, we set $\beta_\ell=\log(\frac{\lambda}{\ell}+1)\approx\lambda/\ell$, where λ is a hyper-parameter selected among $\{0.5,1.0,1.5\}$.

As mentioned in the Introduction, the key feature of the Clenshaw residual connection is the ability to mimic a polynomial filter with arbitrary coefficients above the Chebyshev basis (the second

[1]It's worth noting that besides the initial residual connection *i.e.*, $(1-\alpha)\mathbf{P}\mathbf{H}^{(\ell-1)}+\alpha\mathbf{H}^*$, GCNII introduces an additional submodule in weight matrices, namely identity mapping, which is also incorporated into our approach

kind). In this section, we provide a detailed explanation of how the residual connections establish a connection between our model and arbitrary polynomial representations over the chosen basis.

3.2 A Warm-up: GCNII and Horner's method

In this section, we will first consider an **incomplete** version of the ClenshawGCN termed HornerGCN. This extension is derived from a warm-up discussion on GCNII.

GCNII: Predetermined Personalized PageRank Coefficients. We start with an analysis of GCNII. To simplify the analysis, we consider $(1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}^{(\ell)}$ as $\mathbf{I}$, and take $\mathrm{relu}(x) = x$. Then the iteration (5) is simplified as

$$\mathbf{H}^{(\ell)} = (1-\alpha)\tilde{\mathbf{P}}\mathbf{H}^{(\ell-1)} + \alpha\mathbf{H}^*. \tag{8}$$

For a GCNII model with K stacked layers, expanding (8) yields $\mathbf{H}^{(K)} = \sum_{\ell=0}^{K} \hat{\alpha}_\ell \tilde{\mathbf{P}}^\ell \mathbf{H}^*$, where $\hat{\alpha}_\ell$ represents the effective weight associated with the ℓ-th propagation step. Specifically, it can be calculated that $\hat{\alpha}_\ell = \begin{cases} \alpha(1-\alpha)^\ell, & \ell < K \\ (1-\alpha)^K, & \ell = K \end{cases}$. This formulation demonstrates that the iterative process of GCNII combines the representations from different layers with fixed PageRank coefficients

Free GCNII's Coefficients. However, in the case of heterophilous graphs, a more *flexible* exploitation of different diffusion layers is preferred [52, 56], where arbitrarily learnable $\{\hat{\alpha}_\ell\}_{\ell=0}^{K}$ are used, in contrast to GCNII's fixed PageRank coefficients. To align with these works, an equivalent method to modify GCNII is to replace the fixed α in (5) with *learnable* coefficients. In this way, the convolution we obtain is:

$$\mathbf{H}^{(\ell)} = \sigma\left(\left(\tilde{\mathbf{P}}\mathbf{H}^{(\ell-1)} + \alpha_\ell \mathbf{H}^*\right)\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}^{(\ell)}\right)\right). \tag{9}$$

The unfolded nested expression of Equation (9), when ignoring non-linear transformations, then becomes $\sum_{\ell=0}^{K} \alpha_{K-\ell}\tilde{\mathbf{P}}^\ell \mathbf{H}^*$. This is equivalent to applying spectral filtering over $\mathbf{H}^*$ using a polynomial filtering function with *arbitrary* coefficients over the monomial basis: $h(\mu) = \sum_{\ell=0}^{K} \alpha_{K-\ell}\tilde{\mathbf{P}}^\ell \mu^\ell$.

Horner's Method. What captivates us is that the recurring *unfolding process* can be effectively characterized as an implementation of Horner's Method. To illustrate this, we present and compare (1) Horner's Methods and (2) the unfolding process of Equation (9).

Horner's Method [17] is a technique for computing the value of $p(x_0)$, where $p(x)$ is defined via $p(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_n x^n$. The essence of Horner's Method lies in the rearrangement of the evaluation process for $p(x_0)$, as follows:

$$\begin{aligned} b_n &:= a_n, \\ b_{n-1} &:= a_{n-1} + b_n x_0, \\ &\vdots \\ b_0 &:= a_0 + b_1 x_0, \\ \Rightarrow p(x_0) &:= b_0. \end{aligned}$$

The unfolding Process of Equation (9) is:

$$\begin{aligned} \mathbf{H}^{(0)} &= \alpha_0 \mathbf{H}^*, \\ \mathbf{H}^{(1)} &= \tilde{\mathbf{P}}\left(\alpha_0\mathbf{H}^*\right) + \alpha_1\mathbf{H}^*, \\ \mathbf{H}^{(2)} &= \tilde{\mathbf{P}}\left(\tilde{\mathbf{P}}\left(\alpha_0\mathbf{H}^*\right) + \alpha_1\mathbf{H}^*\right) + \alpha_2\mathbf{H}^*, \\ &\vdots \\ \mathbf{H}^{(K)} &= \tilde{\mathbf{P}}\left(\cdots\left(\tilde{\mathbf{P}}\left(\tilde{\mathbf{P}}\left(\alpha_0\mathbf{H}^*\right) + \alpha_1\mathbf{H}^*\right) + \alpha_2\mathbf{H}^*\right)\cdots\right) + \alpha_K\mathbf{H}^* \\ \Rightarrow \mathbf{H}^{(K)} &= \alpha_K\mathbf{H}^* + \alpha_{K-1}\tilde{\mathbf{P}}\mathbf{H}^* + \cdots + \alpha_0\tilde{\mathbf{P}}^K \\ &= \sum_{\ell=0}^{K} \alpha_{K-\ell}\tilde{\mathbf{P}}^\ell\mathbf{H}^*. \end{aligned}$$

This process aligns in parallel with the recurring of Horners' method to expand a polynomial. Therefore, we term the model stacked by convolutions in Equation (9) as **HornerGCN**.

Towards Clenshaw's Method. In this warm-up section, we eliminate the homophilic assumption of GCNII by simply substituting the predetermined PageRank coefficients with arbitrary learnable coefficients. The resultant model, named HornerGCN, is able to represent *any possible polynomial filter*. Notably, the unfolding process of the stacked convolutional layers within HornerGCN corresponds precisely to the well-known classical technique called Horner's Method.

One of the main characteristic shared by state-of-art polynomial filters [15, 16, 47] is their utilization of the *polynomial basis* for better approximation. Given that Horner's Method is designed specifically for the weighted sum of the Monomial basis, a natural question arises, *Can we incorporate polynomial basis into such convolutions?* In fact, Horner's Method is a special case for **Clenshaw** summation algorithm [7, 48, 49]. The Clenshaw Summation Algorithm indicates that, to introduce the utilization of a diverse range of polynomial basis, all we need is to establish *a negative connection back towards the second last layer*. Based on the warm-ups, we proceed to the subsequent section where we present our primary methodology: ClenshawGCN.

3.3 Polynomial Filter behinds ClenshawGCN

Section Overview. We have proposed the formulation of Clenshaw convolution in Equation (7). In this section, the role of the **negative residual** will be revealed.

Theorem 3.1 concretely states the existence and exact form of the underlying polynomial filtering function within ClenshawGCN. For the proof of this theorem, we first introduce the *Clenshaw Summation Algorithm*, an iterative algorithm for evaluating summations over a kind of polynomial bases, which inspires our Clenshaw convolution. Using a customized version of the Clenshaw algorithm specifically tailored for the second kind of Chebyshev basis, we demonstrate that the iteration of graph convolutions, enhanced by our residuals, effectively yields a series of underlying polynomial filtering functions. Notably, the form of these functions bears an obvious resemblance to the structure of the Clenshaw summation algorithm. Based on the previous foundations, the spectral nature of Theorem 3.1 is obviously revealed.

Spectral Nature. Given the definition of *polynomial filters* (Definition 2.1), we pose Theorem 3.1 on ClenshawGCN's corresponding polynomial filter.

Theorem 3.1 (ClenshawGCN's corresponding polynomial filter). *A K-order ClenshawGCN defined in Equation* (7)*, when consider $(1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}^{(\ell)}$ for each ℓ to be $\mathbf{I}$, and $relu(x)$ to be x, is a polynomial filter*

$$h(\mu) = \sum_{\ell=0}^{K} \alpha_{K-\ell} U_\ell(\mu),$$

where $\{U_\ell\}_{\ell=0}^{K}$ is the truncated K-order second-kind Chebyshev basis, and $\{\alpha_\ell\}_{\ell=0}^{K}$ is the set of initial residue coefficients.

In the subsequent sections, we will proceed to prove this theorem comprehensively.

Lemma: Clenshaw Algorithm. Clenshaw summation algorithm was first proposed by Clenshaw [7] to evaluate the linear combination of the *shifted Chebyshev polynomials (the first kind)* by a *recurrence*. It is pointed out in Clenshaw's original note [7] that, this algorithm can be easily adapted to any polynomial series that satisfy a three-term recurrence relation[2] among the neighboring polynomials. Horner's method is a special case for the Clenshaw algorithm.

In preparation for the proof of Theorem 3.1 in the upcoming section, we introduce Clenshaw's Algorithm for Chebyshev Polynomials (the second Kind) in Lemma 3.2. Compared to the more general Clenshaw's algorithm, we have made certain simplifications in the calculation of $S(x)$ within Lemma 3.2. This distinction leads us to present our proof in Appendix A. Interested readers can compare our proof with similar proofs for the first kind of Chebyshev basis as found in Mason and Handscomb [31].

Lemma 3.2 (Clenshaw Summation Algorithm for Chebyshev polynomials (Second Kind)). *For the Second Kind of Chebyshev Polynomials $\{U_k(x)\}_{k=0}^{\infty}$, the weighted sum of the truncated series with order n:*

$$S(x) = \sum_{k=0}^{n} a_k U_k(x),$$

can be computed by a recurrence formula:

$$\begin{aligned} b_{n+2}(x) &:= 0, \\ b_{n+1}(x) &:= 0, \\ b_k(x) &:= a_k + 2x b_{k+1}(x) - b_{k+2}(x). \\ & \quad (k = n, n-1, \cdots, 0) \end{aligned} \tag{10}$$

Then $S(x) \equiv b_0(x)$.

Proof of Theorem 3.1. Theorem 3.1 is immediately clear after the proposition below is proved.

[2]For three-term recurrence relation, we mean that the k-th polynomial ϕ_k in a polynomial series $\{\phi_i\}_{i=0}^{\infty}$ can be generated from ϕ_{k-1} and ϕ_{k-2}. Check Equation (13) for an example.

Proposition 3.3. *Consider $(1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}^{(\ell)}$ for each ℓ to be $\mathbf{I}$, and $relu(x)$ to be x, then each $\mathbf{H}^{(\ell)}$ obtained through Equation* (7) *can be equivalently derived from a polynomial filtering function, denoted as $h^{(\ell)}(\mu)$, operating on the input features $\mathbf{H}^*$. Moreover, for $\ell = 0, 1, \cdots, K$, $h^{(\ell)}(\mu)$ satisfies :*

$$h^{(\ell)}(\mu) = \alpha_\ell + 2\mu h^{(\ell-1)}(\mu) - h^{(\ell-2)}(\mu).$$

Proof. We establish the proof of Proposition 3.3 using the inductive method.

(1) Given Conditions.

$$\text{Iteration:} \qquad \mathbf{H}^{(\ell)} = 2\tilde{\mathbf{P}}\mathbf{H}^{(\ell-1)} - \mathbf{H}^{(\ell-2)} + \alpha_\ell \mathbf{H}^*, \tag{11}$$

$$\text{Initialization:} \qquad \mathbf{H}^{(-1)} = \mathbf{H}^{(-2)} = \mathbf{0}.$$

(2) Basis Case. Note that $\mathbf{H}^{(-2)} = \mathbf{H}^{(-1)} = \mathbf{0}$. We can soonly write the corresponding polynomial filtering functions for generating $\mathbf{H}^{(-2)}$ and $\mathbf{H}^{(-1)}$ that are:

$$h^{(-1)}(\mu) = 0, \quad h^{(-2)}(\mu) = 0.$$

(3) Inductive Hypothesis. Assume that when the convolution (Equation (11)) progresses to the ℓ-th layer, we have already established that $\mathbf{H}^{(\ell-1)}$ and $\mathbf{H}^{(\ell-2)}$ are polynomial filtered results of $\mathbf{H}^*$, with the corresponding filtering functions $h^{(\ell-1)}$ and $h^{(\ell-2)}$, respectively. Specifically,

$$\mathbf{H}^{(\ell-1)} = \mathbf{U}h^{\ell-1}(\mathbf{M})\mathbf{U}^T\mathbf{H}^*, \ \mathbf{H}^{(\ell-2)} = \mathbf{U}h^{\ell-2}(\mathbf{M})\mathbf{U}^T\mathbf{H}^*.$$

(4) Inductive Step. Plugging the inductive hypothesis into Equation (11), we soonly get:

$$\begin{aligned} \mathbf{H}^{(\ell)} &= 2\underline{\mathbf{U}\mathbf{M}\mathbf{U}^T}\underline{\mathbf{U}h^{(\ell-1)}(\mathbf{M})\mathbf{U}^T\mathbf{H}^*} - \underline{\mathbf{U}h^{(\ell-2)}(\mathbf{M})\mathbf{U}^T\mathbf{H}^*} + \alpha_\ell \mathbf{H}^* \\ &= \mathbf{U}\left(2\mathbf{M}h^{(\ell-1)}(\mathbf{M}) - h^{(\ell-2)}(\mathbf{M}) + \alpha_\ell \mathbf{I}\right)\mathbf{U}^T\mathbf{H}^*. \end{aligned}$$

We can conclude that $\mathbf{H}^{(\ell)}$ is also a polynomial filtered result of $\mathbf{H}^*$, with the corresponding filtering function given by:

$$h^{(\ell)}(\mu) = \alpha_\ell + 2\mu h^{(\ell-1)}(\mu) - h^{(\ell-2)}(\mu).$$

Thus, the filtering function for the ℓ-th layer, denoted as $h^{(\ell)}(\mu)$, exhibits the same form as stated above.

By combining the basis case, the inductive hypothesis, and the inductive step, we have successfully established the proof of Proposition 3.3. □

Stacking the inductive relations from Proposition 3.3 for $\ell = 0, 1, \cdots, K$, we obtain the following series:

$$\begin{aligned} h^{(-2)}(\mu) &= 0, \\ h^{(-1)}(\mu) &= 0, \\ h^{(\ell)}(\mu) &= \alpha_\ell + 2\mu h^{(\ell-1)}(\mu) - h^{(\ell-2)}(\mu), \\ & \quad (k = 0, 1, \cdots, K), \end{aligned}$$

Note that the progressive access of the underlying polynomial filtering functions is in a **totally parallel** way with the recurrence in Equation (10). From Theorem 3.1, we soonly come to the point: the final output of ClenshawGCN is corresponding to a polynomial filter, which is: $\sum_{\ell=0}^{K} \alpha_{K-\ell} U_\ell(\mu) \equiv h^{(K)}(\mu)$. Thus, we have finished the proof of Theorem 3.1.

4 EXPERIMENTS

Overview. In this section, we conduct three sets of experiments.

- In Section 4.1 to 4.3, we verify that our method, aimed at injecting *spectral* power into a *spatial* model, **benefits from both sides**. Specifically, we perform a comparative analysis between ClenshawGCN and both spatial methods enhanced with residuals (Section 4.1) and state-of-the-art spectral methods (Section 4.2). Furthermore, we demonstrate the state-of-the-art performances achieved by our approach on two large LINKX datasets (Section 4.3).
- In Section 4.4 to 4.5, we carry out **ablation studies** to validate the effectiveness of the submodules in our model. The first ablation investigates ClenshawGCN's advantage as a spatial model; the second ablation verifies the effectiveness of the two components in Clenshaw residual connections. The first ablation study examines the advantages of ClenshawGCN as a spatial model; while the second ablation study verifies the effectiveness of the two components in Clenshaw residual connections.
- In Appendix C, we include two experiments conducted during the rebuttal phase that merit further discussion. The first experiment provides evidence that ClenshawGCN successfully addresses **model degradation**, which is a primary focus of early graph residuals. The second experiment demonstrates the potential of Clenshaw residuals to be employed in **other spatial backbones**, *i.e.*, GAT [45], despite lacking a direct spectral interpretation.

Overall Experimental Setup. The ten datasets used for our experiments are outlined in Table 1. Additional details regarding the experimental setup, including further information about the used datasets, data splitting, ClenshawGCN's configuration, and hyperparameter tuning, can be found in Appendix B.

Table 1: Statistics for all node classification datasets we use. Datasets of different homophily degrees and different sizes are used.

Dataset	#Nodes	#Edges	#Classes	$\mathcal{H}(G)$
Cora	2,709	5,429	7	.83
PubMed	19,717	44,338	3	.71
Citeseer	3,327	4,732	6	.79
Squirrel	5,201	217,073	5	.22
Chameleon	7,600	33,544	5	.23
Texas	183	309	5	.11
Cornell	183	295	5	.30
Penn94	41,554	1,362,229	2	.47
Genius	**421,961**	984,979	2	.62
Twitch-Gamers	168,114	**6,797,557**	2	.55

4.1 Comparing ClenshawGCN with Other Residual-Based Methods.

In this subsection, we demonstrate the effectiveness of ClenshawGCN's residual connections by comparing it with other spatial models, namely GCNII [5], H_2GCN [58], MixHop [1], and JKNet [52]. Among these models, GCNII incorporates initial residual connections, while the remaining models employ dense residual connections. Additionally, the methods used to combine multi-scale representations in H_2GCN and JKNet are more intricate than the weighted sum.

Results. As presented in Table 2a, our ClenshawGCN surpasses all the baseline models. In accordance with our expectations, ClenshawGCN exhibits notably superior performance on the *heterophilous datasets*, including the specialized *H_2GCN* model designed specifically for heterophilic graphs. This observation highlights the effectiveness of leveraging spectral characteristics in our approach.

On the other hand, ClenshawGCN demonstrates a competitive edge even on the *homophilic datasets*, despite the presence of strong baseline models such as GCNII and JKNet, which are known for their effectiveness on such datasets. Particularly noteworthy is the performance of ClenshawGCN on the PubMed dataset, where it achieves state-of-the-art results.

4.2 Comparing ClenshawGCN with Spectral Baselines

In Section 3.3, we have demonstrated that ClenshawGCN functions as a spectral model and has the capability to emulate any K-order polynomial filter by utilizing $\{U_\ell\}_{\ell=0}^{K}$. In this subsection, we compare ClenshawGCN with strong spectral GNNs, including ChebNet [9], APPNP [21], ARMA [3], GPRGNN [6], BernNet [15], and ChebNetII [16]. Among these models, APPNP utilizes *fixed* parameters to simulate polynomial filters, ARMA GNN simulates ARMA filters [33], and the remaining models employ *learnable* polynomial filters based on the Chebyshev basis, Monomial basis, or Bernstein basis.

Results. As indicated in Table 2b, ClenshawGCN demonstrates superior performance compared to most of the baseline models on each dataset, with the exception of Chameleon and Citeseer. Notably, ClenshawGCN achieves a large margin of improvement of 7.66% over other models on the Squirrel dataset.

It is worth noting the comparison between ClenshawGCN and ChebNetII. ChebNetII benefits from the utilization of *Chebyshev nodes*, which play a vital role in polynomial interpolation and provide it with additional power. However, even without the use of Chebyshev nodes, ClenshawGCN exhibits comparable performance to ChebNetII. The additional power of ClenshawGCN may stem from the entangled non-linear transformations it employs. We will further investigate and re-examine this aspect in Section 4.4.

4.3 ClenshawGCN on Large non-Homophilous Datasets

Additionally, we conduct a comparison of ClenshawGCN on three large non-homophilous datasets obtained from Lim et al. [26]: Penn94, Genius, and Twitch-gamer. The selected baselines for comparison include spatial models with representative residual blocks (GCNII, MixHop), state-of-the-art spectral models (GPRGNN, ChebNet, BernNet, ChebNetII), as well as the best-performing methods on these datasets, namely LINKX [26], glognn+ [25], and ACM-GCN [29].

Table 2: Comparison with (a) spatial models with representative residuals submodules and (b) state-of-art spectral models. Results of ClenshawGCN are reported after repeated experiments on twenty random 60%/20%/20% splits.

(a) Comparison with other models equipped with different kinds of residual connections. Mean classification accuracies *(± standard derivations)* of twenty random splits are displayed. Besides the ClenshawGCN, all the results are taken directly from Luan et al. [28] and Lim et al. [26].

Datasets	Chameleon	Squirrel	Actor	Texas	Cornell	Cora	Citeseer	PubMed
$\lvert\mathcal{V}\rvert$	2,277	5,201	7,600	183	183	2,708	3,327	19,717
MLP	46.59±1.84	31.01±1.18	40.18±0.55	86.81±2.24	84.15±3.05	76.89±0.97	76.52±0.89	86.14±0.25
GCN	60.81±2.95	45.87±0.88	33.26±1.15	76.97±3.97	65.78±4.16	87.18±1.12	79.85±0.78	86.79±0.31
GCNII	63.44±0.85	41.96±1.02	36.89±0.95	80.46±5.91	84.26±2.13	88.46±0.82	79.97±0.65	89.94±0.31
H_2GCN	52.30±0.48	30.39±1.22	38.85±1.17	85.90±3.53	86.23±4.71	87.52±0.61	79.97±0.69	87.78±0.28
MixHop	36.28±10.22	24.55±2.60	33.13±2.40	76.39±7.66	60.33±28.53	65.65±11.31	49.52±13.35	87.04±4.10
GCN+JK	64.68±2.85	53.40±1.90	32.72±2.62	80.66±1.91	66.56±13.82	86.90±1.51	73.77±1.85	90.09±0.68
ClenshawGCN	**69.44±2.06**	**62.14±1.65**	**42.08±1.99**	**93.36±2.35**	**92.46±3.72**	**88.90±1.26**	**80.34±1.26**	**91.99±0.41**

(b) Comparison with spectral models. Mean classification accuracies *(±95% confidence intervals)* on twenty random 60%/20%/20% train/validation/test splits are displayed. Besides the ClenshawGCN, all the results are taken directly from [15].

Datasets	Chameleon	Squirrel	Actor	Texas	Cornell	Cora	Citeseer	PubMed
$\lvert\mathcal{V}\rvert$	2,277	5,201	7,600	183	183	2,708	3,327	19,717
ChebNet	59.51±1.25	40.81±0.42	37.42±0.58	86.28±2.62	83.91±2.17	87.32±0.92	79.33±0.57	87.82±0.24
ARMA	60.21±1.00	36.27±0.62	37.67±0.54	83.97±3.77	85.62±2.13	87.13±0.80	80.04±0.55	86.93±0.24
APPNP	52.15±1.79	35.71±0.78	39.76±0.49	90.64±1.70	91.52±1.81	88.16±0.74	80.47±0.73	88.13±0.33
GPRGNN	67.49±1.38	50.43±1.89	39.91±0.62	92.91±1.32	91.57±1.96	88.54±0.67	80.13±0.84	88.46±0.31
BernNet	68.53±1.68	51.39±0.92	41.71±1.12	92.62±1.37	92.13±1.64	88.51±0.92	80.08±0.75	88.51±0.39
ChebNetII	**71.37±1.01**	57.72±0.59	41.75±1.07	93.28±1.47	92.30±1.48	88.71±0.93	**80.53±0.79**	88.93±0.29
ClenshawGCN	69.44±0.92	**62.14±0.70**	**42.08±0.86**	**93.36±0.99**	**92.46±1.64**	**88.90±0.59**	80.34±0.57	**91.99±0.17**

Table 3: Comparison with state-of-art models on three LINKX datasets. we use the five random splits given in LINKX [26] with a 50%/25%/25% proportion to align with reported results. Results other than our methods are mainly taken from Lim et al. [26], He et al. [16] and Li et al. [25]. Results of ACM-GCN++ are taken from the leaderboard of the paperswithcode website.

Datasets	Penn94	genius	twitch-gamers
$\lvert\mathcal{V}\rvert$	41,554	421,961	168,114
MLP	73.61 ± 0.40	86.68 ± 0.09	60.92 ± 0.07
GCN	82.47 ± 0.27	87.42 ± 0.35	62.18 ± 0.26
GAT	81.53 ± 0.55	55.80 ± 0.87	59.89 ± 4.12
MixHop	83.47 ± 0.71	90.58 ± 0.16	65.64 ± 0.27
GCNII	82.92 ± 0.59	90.24 ± 0.09	63.39 ± 0.61
GPR-GNN	83.54 ± 0.32	90.15 ± 0.30	62.59 ± 0.38
ChebNet	82.59 ± 0.31	89.36 ± 0.31	62.31 ± 0.37
BernNet	83.26 ± 0.29	90.47 ± 0.33	64.27 ± 0.31
ChebNetII	84.86 ± 0.33	90.85 ± 0.32	65.03 ± 0.27
LINKX	84.71 ± 0.52	90.77 ± 0.27	66.06 ± 0.19
ACM-GCN	82.52 ± 0.96	80.33 ± 3.91	62.01 ± 0.73
ACM-GCN++	**86.08 ± 0.43**	91.40 ± 0.07	65.94 ± 0.28
GloGNN	85.57 ± 0.35	90.66 ± 0.11	66.19 ± 0.29
GloGNN++	85.74 ± 0.42	90.91 ± 0.13	66.34 ± 0.29
ClenshawGCN	85.38 ± 0.25	**91.69 ± 0.25**	**66.56 ± 0.28**

Results. As depicted in Table 3, the performance of ClenshawGCN on the non-homophilous datasets is exceptional. It outperforms state-of-the-art *spectral* methods on all three datasets. Particularly noteworthy is that ClenshawGCN achieves new best performances, to the best of our knowledge, on the genius and twitch-gamer datasets.

4.4 Ablation Study I: Advantage over Pure Polynomial Filters.

We have shown in the methodology section that, beyond general spatial models, ClenshawGCN incorporates a polynomial filter backbone based on the Chebyshev basis (the second kind). On the other side, beyond general polynomial filters, ClenshawGCN shares entangled non-linear transformations with 'deep' models, as opposed to 'shallow' models that typically have one or two MLP layers disentangled from graph propagations. In this ablation study, we re-examine the effectiveness of these non-linear transformations.

Ablation Models. We use two ablation models. The first ablation model, ClenshawGCN(-Act), represents a variant of ClenshawGCN where the activation function in Equation (7) is omitted. The second ablation model, ClenshawGCN(-Act-W), represents another variant of ClenshawGCN where both the activation function and the weight matrix in Equation (7) are removed. Note that ClenshawGCN(-Act-W) can be seen as a pure polynomial filter based on the second kind of Chebyshev basis, with the order of calculations rearranged.

Results. We conduct this ablation study on four datasets and exhibit the results in Table 4. On Squirrel, PubMed and Penn94, the performances of ClenshawGCN surpass ClenshawGCN(-Act) and ClenshawGCN(-Act-W), which demonstrates that the entangling of non-linear transformation further improves the performances based on pure polynomial filters.

Table 4: Results for ablation study I. The reported results are based on the same experimental setup in Table 2.

Models	ClenshawGCN	ClenshawGCN(-Act)	Clenshaw(-Act-W)
Squirrel	**62.14 ± 1.65**	61.55 ± 1.42	56.92 ± 2.13
Chameleon	69.45 ± 2.12	67.29 ± 2.35	**70.08 ± 2.43**
PubMed	**91.99 ± 0.41**	91.56 ± 0.46	91.27 ± 0.53
Penn94	**85.38 ± 0.25**	84.68 ± 0.56	84.39 ± 0.26

4.5 Ablation Study II: Submodules of Clenshaw Residual Connection

Recall Equation (7). Compared with vanilla GCN convolution, ClenshawGCN is composed of two submodules: an initial residual and a second-order negative residual. In this section, we conduct ablation analyses on these two modules to verify their contributions.

Ablation Model: HornerGCN. We use HornerGCN as an ablation model to verify the contribution of *negative residues*. Recall Section 3.2, the corresponding polynomial filter of HornerGCN is $h_{Horner}(\mu) = \sum_{\ell=0}^{K} \alpha_{K-\ell}\mu^{\ell}$ which uses the Monomial basis. While the complete form of our ClenshawGCN borrows the use of Chebyshev basis by negative residues.

Ablation Model: FixedParamClenshawGCN. By the FixedParamClenshawGCN model, we verify the contribution of *flexible* initial residue in Equation (7). We keep the use of negative residuals, but fix $\vec{\alpha} = [\hat{\alpha}_0, \hat{\alpha}_1, \cdots, \hat{\alpha}_K]$ to be $\hat{\alpha}_\ell = \alpha(1-\alpha)^{K-\ell}$ and $\hat{\alpha}_0 = (1-\alpha)^K$ following APPNP [21] and the initialization setting in GPRGNN [6], where $\alpha \in [0,1]$ is a hyperparameter. The corresponding polynomial filter of FixedParamClenshawGCN is $h_{Fix}(\mu) = \sum_{\ell=0}^{K} \hat{\alpha}_{K-\ell} U_\ell(\mu)$.

Results. We compare the performance of HornerGCN and FixedParamClenshawGCN with GCN, GCNII and ClenshawGCN on two median-sized datasets: Chameleon and Squirrel. As shown in Figure 2, either removing the negative second-order residue or fixing the initial residue causes an obvious drop in Test Accuracy.

Notably, even with only one residual module, HornerGCN and FixedParamClenshawGCN still outperform homophilic models such as GCNII. The reason behind HornerGCN's superiority is evident: it can simulate any polynomial filter, which is advantageous for heterophilic graphs. As for FixedParamClenshawGCN, although the coefficients of $U_\ell(\mu)$ are fixed, the contribution of each fused level (*i.e.*, $\tilde{\mathbf{P}}^{\ell}\mathbf{H}^{*}$) is no longer restricted to be solely positive, as observed in GCNII. This is because each Chebyshev polynomial comprises terms with *alternating signs*, such as $U_4(\tilde{\mathbf{P}}) = 16\tilde{\mathbf{P}}^4 - 12\tilde{\mathbf{P}}^2 + 1$. This deviation from the underlying homophily assumption allows

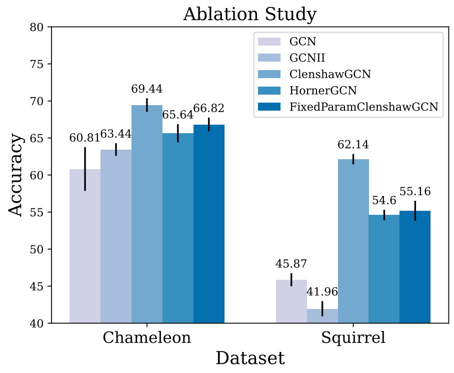

Figure 2: Results of the ablation study II. HornerGCN and FixedParamClenshawGCN are weakened versions of ClenshawGCN. HornerGCN is equipped solely with adaptive initial residue, while FixedParamClenshawGCN is equipped only with negative second-order residue. Although the performances of these two ablation models are inferior to that of the complete ClenshawGCN, they still outperform GCN and GCNII.

FixedParamClenshawGCN to capture more diverse and complex relationships within the graph structure.

5 CONCLUSION

In this paper, we propose ClenshawGCN, a GNN model equipped with a novel and neat residual connection module that is able to mimic a spectral polynomial filter. The construction of this residual connection inherently uses Clenshaw Summation Algorithm. We prove that our model implicitly simulates *any* polynomial filter based on the *second-kind Chebyshev basis*.

For **future work**, a promising direction is to further investigate the mechanism of such spectral-domain-inspired spatial models or spectral models entangled with non-linearity. We expect such models to incorporate the strengths of both sides according to our preliminary results. The entangled structure also poses a challenge in scaling up such models to large graphs.

ACKNOWLEDGMENTS

This research was supported in part by National Natural Science Foundation of China (No. U2241212, No. 61972401, No. 61932001, No. 61832017), by the major key project of PCL (PCL2021A12), by Beijing Natural Science Foundation (No. 4222028), by Beijing Outstanding Young Scientist Program No.BJJWZYJH012019100020098, by Alibaba Group through Alibaba Innovative Research Program, and by Huawei-Renmin University joint program on Information Retrieval. We also wish to acknowledge the support provided by Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education. Additionally, we acknowledge the support from Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Public Policy and Decision-making Research Lab, Public Computing Cloud, Renmin University of China.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*. PMLR, 21–29.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.

[3] 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* (2021).

[4] Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. 2021. Grand: Graph neural diffusion. In *International Conference on Machine Learning*. PMLR, 1407–1418.

[5] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. 2020. Simple and deep graph convolutional networks. In *ICML*. PMLR, 1725–1735.

[6] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*.

[7] Charles W Clenshaw. 1955. A note on the summation of Chebyshev series. *Math. Comp.* 9, 51 (1955), 118–120.

[8] Mark Craven, Andrew McCallum, Dan PiPasquo, Tom Mitchell, and Dayne Freitag. 1998. *Learning to extract symbolic knowledge from the World Wide Web*. Technical Report. Carnegie-mellon univ pittsburgh pa school of computer Science.

[9] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. (6 2016). http://arxiv.org/abs/1606.09375

[10] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. *Advances in neural information processing systems* 28 (2015).

[11] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

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

[13] David K. Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2009. Wavelets on Graphs via Spectral Graph Theory. *CoRR* abs/0912.3848 (2009). arXiv:0912.3848 http://arxiv.org/abs/0912.3848

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[15] Mingguo He, Zhewei Wei, Zengfeng Huang, and Hongteng Xu. 2021. BernNet: Learning Arbitrary Graph Spectral Filters via Bernstein Approximation. *Advances in Neural Information Processing Systems* 34 (6 2021), 14239–14251. http://arxiv.org/abs/2106.10994

[16] Mingguo He, Zhewei Wei, and Ji-Rong Wen. 2022. Convolutional Neural Networks on Graphs with Chebyshev Approximation, Revisited. *arXiv preprint arXiv:2202.03580* (2022).

[17] William George Horner. 1819. XXI. A new method of solving numerical equations of all orders, by continuous approximation. *Philosophical Transactions of the Royal Society of London* 109 (1819), 308–335.

[18] Forrest Iandola, Matt Moskewicz, Sergey Karayev, Ross Girshick, Trevor Darrell, and Kurt Keutzer. 2014. Densenet: Implementing efficient convnet descriptor pyramids. *arXiv preprint arXiv:1404.1869* (2014).

[19] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

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

[21] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *ICLR*.

[22] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. *arXiv preprint arXiv:1911.05485* (2019).

[23] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. 2019. DeepGCNs: Can GCNs Go as Deep as CNNs? (4 2019). http://arxiv.org/abs/1904.03751

[24] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.

[25] Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. 2022. Finding Global Homophily in Graph Neural Networks When Meeting Heterophily. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162)*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (Eds.). PMLR, 13242–13256.

[26] 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. *Advances in Neural Information Processing Systems* 34 (10 2021), 20887–20902. http://arxiv.org/abs/2110.14446

[27] Xiaorui Liu, Jiayuan Ding, Wei Jin, Han Xu, Yao Ma, Zitao Liu, and Jiliang Tang. 2021. Graph Neural Networks with Adaptive Residual. *NIPS* (2021). https://github.com/lxiaorui/AirGNN.

[28] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2021. Is Heterophily A Real Nightmare For Graph Neural Networks To Do Node Classification? (9 2021). http://arxiv.org/abs/2109.05641

[29] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2021. Is Heterophily A Real Nightmare For Graph Neural Networks To Do Node Classification? *arXiv preprint arXiv:2109.05641* (2021).

[30] 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. *arXiv preprint arXiv:2210.07606* (2022).

[31] John C Mason and David C Handscomb. 2002. *Chebyshev polynomials*. Chapman and Hall/CRC.

[32] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* (2001), 415–444.

[33] Sunil K Narang, Akshay Gadde, and Antonio Ortega. 2013. Signal processing techniques for interpolation in graph structured data. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 5445–5449.

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

[35] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. *The pagerank citation ranking: Bringing order to the web*. Technical Report. Stanford InfoLab.

[36] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In *ICLR*.

[37] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. 2020. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*.

[38] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. 2021. E (n) equivariant graph neural networks. In *International conference on machine learning*. PMLR, 9323–9332.

[39] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. 2020. Interpreting graph neural networks for nlp with differentiable edge masking. *arXiv preprint arXiv:2010.00577* (2020).

[40] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[41] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. 2013. Vertex-Frequency Analysis on Graphs. (7 2013). http://arxiv.org/abs/1307.5708

[42] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. 2013. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*. PMLR, 1139–1147.

[43] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. 2009. Social influence analysis in large-scale networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 807–816.

[44] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[45] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. *stat* 1050, 20 (2017), 10–48550.

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

[47] Xiyuan Wang and Muhan Zhang. 2022. How Powerful are Spectral Graph Neural Networks. (5 2022). http://arxiv.org/abs/2205.11172

[48] Wikipedia. 2023. Clenshaw algorithm — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Clenshaw%20algorithm&oldid=1089015914. [Online; accessed 03-February-2023].

[49] Wikipedia. 2023. Horner's method — Wikipedia, The Free Encyclopedia. http://en.wikipedia.org/w/index.php?title=Horner's%20method&oldid=1135871092. [Online; accessed 03-February-2023].

[50] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. 2021. Graph neural networks for natural language processing: A survey. *arXiv preprint arXiv:2106.06090* (2021).

[51] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys (CSUR)* (2020).

[52] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. 2018. Representation Learning on Graphs with Jumping Knowledge Networks. In *ICML*.

[53] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.
[54] Wentao Zhang, Zeang Sheng, Ziqi Yin, Yuezihan Jiang, Yikuan Xia, Jun Gao, Zhi Yang, and Bin Cui. 2022. Model Degradation Hinders Deep Graph Neural Networks. *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2493–2503. https://doi.org/10.1145/3534678.3539374
[55] Wentao Zhang, Ziqi Yin, Zeang Sheng, Wen Ouyang, Xiaosen Li, Yangyu Tao, Zhi Yang, and Bin Cui. 2021. Graph Attention Multi-Layer Perceptron. *CoRR* abs/2108.10097 (2021). arXiv:2108.10097 https://arxiv.org/abs/2108.10097
[56] Jialin Zhao, Yuxiao Dong, Ming Ding, Evgeny Kharlamov, and Jie Tang. 2021. Adaptive Diffusion in Graph Neural Networks. *Advances in Neural Information Processing Systems* 34 (2021), 23321–23333.
[57] Xin Zheng, Yixin Liu, Shirui Pan, Miao Zhang, Di Jin, and Philip S Yu. 2022. Graph neural networks for graphs with heterophily: A survey. *arXiv preprint arXiv:2202.07082* (2022).
[58] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems* 33 (2020), 7793–7804.
[59] 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*. 1215–1226.

A PROOF FOR LEMMA 3.2

Proof. Denote

$$A = \begin{bmatrix} 1 & & & & \\ -2x & 1 & & & \\ 1 & -2x & 1 & & \\ & & \cdots & & \\ & & 1 & -2x & 1 \end{bmatrix}, \quad \vec{u} = \begin{bmatrix} U_{-1}(x) \\ U_0(x) \\ U_1(x) \\ \cdots \\ U_n(x) \end{bmatrix}, \quad \vec{a} = \begin{bmatrix} 0 \\ a_0 \\ a_1 \\ \cdots \\ a_n \end{bmatrix},$$

with $A \in \mathbb{R}^{(n+2)\times(n+2)}$, $\vec{u} \in \mathbb{R}^{n+2}$, then

$$S(x) = \vec{a}^\top \vec{u}. \tag{12}$$

Indexing a vector from -1, we denote $\mathbf{1}_i \in \mathbb{R}^{(n+2)}$ as the one-hot vector with the i-th element being 1 (i starts from -1). Note that, from the recurrence relation for the Chebyshev polynomials given in Equation (4), we soonly get:

$$A\vec{u} = \begin{bmatrix} U_{-1}(x) \\ -2xU_{-1}(x) + U_0(x) \\ 0 \\ \cdots \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ U_0(x) \\ 0 \\ \cdots \\ 0 \end{bmatrix} = \mathbf{1}_0. \tag{13}$$

Now suppose that there is a vector $\vec{b} = [b_{-1}, b_0, \cdots, b_n]$ satisfying

$$\vec{a}^\top = \vec{b}^\top A, \tag{14}$$

then

$$S(x) \overset{(12)}{=} \vec{a}^\top \vec{u} \overset{(14)}{=} \vec{b}^\top A\vec{u} \overset{(13)}{=} \vec{b}\mathbf{1}_0 = b_0.$$

On the other hand, notice that the recurrence defined in (10) is exactly the Gaussian Elimination process of solving $\vec{a}^\top = \vec{b}^\top A$ from b_n down to b_0, which means that $\{b_n, \cdots, b_0\}$ calculated by (10) satisfies (14). Proof for Lemma 3.2 is finished. □

B OVERALL EXPERIMENTAL SETUP

Datasets and Splits. We use both homophilic graphs and heterophilic graphs in our experiments following former works, especially GCN [20], Geom-GCN [36] and LINKX [26].

- *Citation Graphs.* Cora, PubMed, and CiteSeer [40] are citation datasets processed by Planetoid [53]. In these graphs, nodes are scientific publications, edges are citation links processed to be bidirectional, and node features are bag-of-words representations of the documents. These graphs show strong homophily.
- *Wikipedia Graphs.* The Chameleon dataset and Squirrel dataset are page-page networks on topics in Wikipedia, where nodes are entries, and edges are mutual links.
- *Webpage Graphs.* Texas dataset and Cornell dataset collect web pages from computer science departments of different universities. The nodes in the graphs are web pages of students, projects, courses, staff, or faculties [8], the edges are hyperlinks between them, and node features are the bag-of-words representations of these web pages.
- *Co-occurrence Network.* The Actor network represents the co-occurrence of actors on a Wikipedia page [43]. The node features are filtered keywords in the Wikipedia pages. The categorization of the nodes is done by [36].
- *Mutual following Networks.* Graphs in Twitch-Gamers, Penn94 and genius dataset represent mutual following relationships between accounts from (subsets of) three platforms respectively, that are: streaming platform Twitch, facebook 100 networks and genius.com. The nodes are labeled by states of the channels, genders of the users and tagged states of spam users, respectively.

We list the messages of these networks in Table 1, where $\mathcal{H}(G)$ is the measure of homophily in a graph proposed by Geom-GCN [36]. Larger $\mathcal{H}(G)$ implies stronger homophily.

For all datasets except for the Twitch-gamers dataset, we take a 60%/20%/20% train/validation/test split proportion following former works [6, 15, 16, 36]. We run these datasets twenty times over random splits with random initialization seeds. For the Twitch-gamers dataset, we use the five random splits given in LINKX [26] with a 50%/25%/25% proportion to align with reported results.

ClenshawGCN Setup. Before and after the stack of Clenshaw convolution layers, there are two non-linear transformations to link with the dimensions of the raw features and final class numbers. All intermediate transformation layers are set with 64 hidden units. For initialization of $\vec{\alpha} = [\alpha_0, \cdots, \alpha_K]$, we simply set α_K to be 1 and all the other coefficients to be 0, which corresponds to initializing the polynomial filter to be $g(\lambda) = 1$ (or equivalently, $h(\mu) = 1$).

Optimization and Tuning. For the optimization process on the training sets, we tune $\vec{\alpha}$ with SGD optimizer with momentum [42] and all the other parameters with Adam SGD [19]. We use early stopping with a patience of 300 epochs.

We tune all the hyperparameters on validation sets. To accelerate hyperparameter searching, we use Optuna [2] and run 100 completed trials [3] for each dataset. Below is the search space:

- Orders of convolutions: $K \in \{8, 12, \cdots, 32\}$;
- Learning rates: $\{0.001, 0.005, 0.1, 0.2, 0.3, 0.4, 0.5\}$;
- Weight decays: $\{1e{-}8, 1e{-}7, \cdots, 1e{-}3\}$;
- Dropout rates: $\{0, 0.1, \cdots, 0.7\}$.

The selected hyperparameters can be found in our public repository.

[3] In Optuna, a *trial* means a run with hyperparameter combination; the term 'complete' refers to that, some trials of bad expectations would be pruned before completion.

C SUPPLEMENTARY EXPERIMENTS

This appendix section involves two experiments we conducted during the rebuttal period that are valuable to share.

C.1 Supplementary Experiment I: Overcoming Model Degradation

Stacking more convolution layers while overcoming model degradation is the shared attempt of early graph residuals. Pioneer works often demonstrate their models' capability to mitigate degradation by presenting performance results with varying numbers of layers as control groups, *e.g.*, Li et al. [23, Figure 1] and Chen et al. [5, Table 3 & Figure 2].

In line with these studies, we conduct experiments to evaluate ClenshawGCN's performance with different truncated orders on the PubMed and Squirrel datasets. The experimental setups follow those described in Appendix B, with the exception that the value of K is fixed.

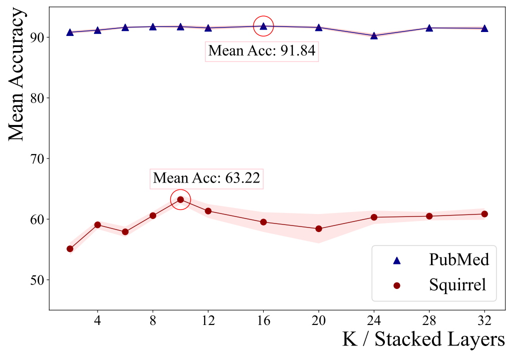

Figure 3: ClenshawGCN's performance with different fixed values of K. The shaded areas indicate the 95% confidence intervals over twenty random splits. The circled data points highlight the best average accuracies achieved. Model degradations have been overcome in that both curves *do not* show decreasing trends.

Results. As shown in Figure 3, ClenshawGCN does not encounter model degradation. The *best performances* are obtained on relatively large layer numbers, *i.e.*, $K = 10$ and $K = 16$, respectively.

Notably, even on relatively shallow ClenshawGCNs with $K = 2$, the performances remain highly competitive with other compared models. This is particularly evident in the significant performance improvement of ClenshawGCN(K=2) compared to GCN, despite both models utilizing two hops of propagation. Detailed performance comparisons with other models can be found in Section 4.1 and Section 4.2.

C.2 Supplementary Experiment II: Effectiveness in Equipping GAT model

In the main body of the paper, we concentrate on a GCN backbone, since GCN, when equipped with Clenshaw residual connections, can be interpreted through a polynomial filter, while for other spatial backbones, *i.e.*, Graph Attention Networks (GAT) [45], such an interpretation is less obvious.

However, Clenshaw residuals potentially benefit general spatial models. From the spectral perspective, when used in GCN convolution layers, negative residuals play a role in leveraging the Chebyshev polynomials. From a spatial perspective, the negative residuals are also well-motivated as they explicitly exploit *negative relations*. Similar ideas can be found in other approaches, such as 'sharpening', or the 'delta operator' introduced in Abu-El-Haija et al. [1] in the context of MixHop.

GAT with Clenshaw Residuals. We propose the formulation of GAT enhanced with Clenshaw residuals. GATConv follows the definition in Velickovic et al. [45]:

$$\mathbf{H}^{(\ell)} = 2\text{GATConv}(\mathbf{H}^{(\ell-1)}) - \mathbf{H}^{(\ell-2)} + \alpha_\ell \mathbf{H}, \quad \ell \in [0, \cdots, K],$$
$$\mathbf{H}^{(-2)} = \mathbf{H}^{(-1)} = \mathbf{O}, \mathbf{H}^* = \text{MLP}(\mathbf{X}, \mathbf{W}^*).$$

We also evaluate the effectiveness of the individual *negative residual submodule*:

$$\mathbf{H}^{(\ell)} = 2\text{GATConv}(\mathbf{H}^{(\ell-1)}) - \mathbf{H}^{(\ell-2)}, \quad \ell \in [1, \cdots, K],$$
$$\mathbf{H}^{(-1)} = \mathbf{O}, \mathbf{H}^{(0)} = \text{MLP}(\mathbf{X}, \mathbf{W}^*).$$

Setup. We adopt the implementation of GATConv From PyG [12]. Both the vanilla GAT model and GAT equipped with Jumping Knowledge [52] are used as baselines. In all cases, the GAT convolution layers use eight attention heads, each with eight hidden channels. For Jumping Knowledge, we constrain the pooling method to either concatenation or max pooling. We conduct repeating experiments over 20 splits on the Squirrel, Chameleon, and PubMed datasets. Hyperparameter tuning and data splits follow the same setup as described in Section B, except that the number of stacked GAT layers are chosen from $[2, 4, \cdots, 12]$.

Table 5: The performance of GAT improves on all three datasets when equipped with Clenshaw residual connections. This suggests that Clenshaw residual connections have the potential to be an effective submodule for general spatial models, extending beyond the GCN backbone.

Method / Variants	Squirrel	Chameleon	PubMed
GAT	51.13 ± 1.70	64.45 ± 1.14	88.74 ± 0.23
GAT+JK	51.61 ± 4.31	63.63 ± 0.95	88.60 ± 0.26
GAT+Negative Residual	**58.89 ± 0.90**	65.85 ± 0.89	91.27 ± 0.19
GAT+Clenshaw Residual	51.56 ± 1.29	**66.84 ± 1.07**	**91.34 ± 0.17**

Results. Table 5 demonstrates the results. When GAT is equipped with Clenshaw residual connections, it achieves significantly improved performance on the Chameleon and PubMed datasets compared to the baseline. The use of negative residuals alone also yields notable benefits, particularly on the Squirrel dataset.