Convolutional Neural Networks on Graphs with Chebyshev Approximation, Revisited

Mingguo He  Renmin University of China mingguo@ruc.edu.cn
Zhewei Wei*  Renmin University of China zhewei@ruc.edu.cn
Ji-Rong Wen  Renmin University of China jrwen@ruc.edu.cn

Abstract

Designing spectral convolutional networks is a challenging problem in graph learning. ChebNet, one of the early attempts, approximates the spectral graph convolutions using Chebyshev polynomials. GCN simplifies ChebNet by utilizing only the first two Chebyshev polynomials while still outperforming it on real-world datasets. GPR-GNN and BernNet demonstrate that the Monomial and Bernstein bases also outperform the Chebyshev basis in terms of learning the spectral graph convolutions. Such conclusions are counter-intuitive in the field of approximation theory, where it is established that the Chebyshev polynomial achieves the optimum convergent rate for approximating a function.

In this paper, we revisit the problem of approximating the spectral graph convolutions with Chebyshev polynomials. We show that ChebNet's inferior performance is primarily due to illegal coefficients learnt by ChebNet approximating analytic filter functions, which leads to over-fitting. We then propose ChebNetII, a new GNN model based on Chebyshev interpolation, which enhances the original Chebyshev polynomial approximation while reducing the Runge phenomenon. We conducted an extensive experimental study to demonstrate that ChebNetII can learn arbitrary graph convolutions and achieve superior performance in both full- and semi-supervised node classification tasks. Most notably, we scale ChebNetII to a billion graph ogbn-papers100M, showing that spectral-based GNNs have superior performance. Our code is available at [https://github.com/ivam-he/ChebNetII](https://github.com/ivam-he/ChebNetII).

1 Introduction

Graph neural networks (GNNs) have received considerable attention in recent years due to their remarkable performance on a variety of graph learning tasks, including social analysis [31, 24, 38], drug discovery [49, 19, 32], traffic forecasting [26, 3, 7] and recommendation system [42, 46].

Spatial-based and spectral-based graph neural networks (GNNs) are the two primary categories of GNNs. To learn node representations, spatial-based GNNs [21, 15, 39] often rely on a message propagation and aggregation mechanism between neighboring nodes. Spectral-based methods [8, 12] create spectral graph convolutions or, equivalently, spectral graph filters, in the spectral domain of the graph Laplacian matrix. We can further divide spectral-based GNNs into two categories based on whether or not their graph convolutions can be learned.

- **Predetermined graph convolutions**: GCN [21] employs a simplified first two Chebyshev polynomials as the graph convolution, which is a fixed low-pass filter [1, 41, 43, 54]. APPNP [22] and GDC [12] set the graph convolution with Personalized PageRank (PPR) and also achieve a low-pass filter. [12, 54]. S²GC [52] derives the graph convolution from the Markov Diffusion Kernel, which is a low- and high-pass filter trade-off. GNN-LF/HF [54]

*Zhewei Wei is the corresponding author.

designed the graph convolutions from the perspective of graph optimization that can imitate low- and high-pass filters.

- **Learnable graph convolutions**: ChebNet approximates the graph convolutions using Chebyshev polynomials and, in theory, could learn arbitrary filters. CayleyNet learns the graph convolutions with Cayley polynomials and generates various graph filters. GPR-GNN uses the Monomial basis to approximate graph convolutions, which can derive low- or high-pass filters. ARMA learns the rational graph convolutions through the Auto-Regressive Moving Average filters family. BernNet utilizes the Bernstein basis to approximate the graph convolutions, which can also learn arbitrary graph filters.

Despite the recent developments, two fundamental problems with spectral-based GNNs remain unsolved. First of all, it is well-known that GCN outperforms ChebNet on real-world datasets (e.g., semi-supervised node classification tasks on citation datasets). However, it is also established that GCN is a simplified version of ChebNet with only the first two Chebyshev polynomials and that ChebNet has more expressive capability than GCN in theory. Consequently, a natural question is: Why is ChebNet’s performance inferior to GCN’s despite its better expressiveness?

Secondly, as shown in [17], the real-world performance of ChebNet is also inferior to that of GPR-GNN and BernNet, which use Monomial polynomial basis and Bernstein polynomial basis to approximate the spectral graph convolutions. Such a conclusion is counter-intuitive in the field of approximation theory, where it is established that the Chebyshev polynomial achieves near-optimum error when approximating a function. Therefore, the second question is: Why is ChebNet’s filter inferior to that of GPR-GNN and BernNet, despite the fact that Chebyshev polynomials have a higher approximation ability?

In this paper, we attempt to tackle these problems by revisiting the fundamental problem of approximating the spectral graph convolutions with Chebyshev polynomials. First of all, according to the theory of the Chebyshev approximation, we observe that the coefficients of the Chebyshev expansion for an analytic function need to satisfy an inevitable convergence. Consequently, we prove that ChebNet’s inferior performance is primarily due to illegal coefficients learnt by ChebNet approximating analytic filter functions, which leads to over-fitting. Furthermore, we propose ChebNetII, a new GNN model based on Chebyshev interpolation, which enhances the original Chebyshev polynomial approximation while reducing the Runge phenomenon. Our ChebNetII model has robust scalability and can easily cope with various constraints on the learned filters via simple reparameterization, such as the non-negativity constraints proposed in [17]. Finally, we conduct an extensive experimental study to demonstrate that ChebNetII can achieve superior performance in both full- and semi-supervised node classification tasks and scale to the billion graph ogbn-papers100M.

## 2 Revisiting ChebNet

### Notations

We consider an undirected graph \( G = (V, E) \) with node set \( V \) and edge set \( E \). Let \( n = |V| \) denote the number of nodes. We use \( x \in \mathbb{R}^n \) to denote the graph signal, where \( x(i) \) denotes the signal at node \( i \). Note that in the general case of GNNs where the input feature is a matrix \( X \), we can treat each column of \( X \) as a graph signal. Let \( A \) denote the adjacency matrix and \( D \) denote the diagonal degree matrix, where \( D_{ii} = \sum_j A_{ij} \). For convenience, we use \( P = D^{-1/2} A D^{-1/2} \) and \( L = I - D^{-1/2} A D^{-1/2} \) to denote the normalized adjacency matrix and the normalized Laplacian matrix of \( G \), respectively. We use \( L = U \Lambda U^T \) to represent the eigendecomposition of \( L \), where \( U \) denotes the matrix of eigenvectors and \( \Lambda = \text{diag}[\lambda_1, ..., \lambda_n] \) is the diagonal matrix of eigenvalues.

### 2.1 Spectral-based GNNs and ChebNet

Spectral-based GNNs create the spectral graph convolutions in the domain of Laplacian spectrum. Recent studies suggest that many popular methods use the polynomial spectral filters to achieve graph convolutions [8, 21, 17]. We can formulate this polynomial filtering operation as

\[
y = U \text{diag}[h(\lambda_1), ..., h(\lambda_n)] U^T x = U h(A) U^T x \approx \sum_{k=0}^{K} w_k L^k x, \tag{1}
\]

where \( y \) denotes the filtering results of \( x \), and \( h(\lambda) \) is called the spectral filter, which is a function of eigenvalues of the Laplacian matrix \( L \). The \( w_k \) denote the polynomial filter weights, and the
where $f$ with explicit coefficients, ChebBase, which simply replaces the Monomial basis of GPR-GNN and

\[ \hat{W} \]

with the trainable weights

\[ \theta \]

of the filtering operation (2) are

\[ y \approx \sum_{k=0}^{K} w_k T_k(\hat{L})x, \]  

where $\hat{L} = 2L/\lambda_{max} - I$ denotes the scaled Laplacian matrix. $\lambda_{max}$ is the largest eigenvalue of $L$ and $w_k$ denote the Chebyshev coefficients. The Chebyshev polynomials can be recursively defined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$. ChebNet's structure is:

\[ Y = \sum_{k=0}^{K} T_k(\hat{L})XW_k, \]  

with the trainable weights $W_k$. The Chebyshev coefficients $w_k$ of the filtering operation (2) are implicitly encoded in the weight matrices $W_k$. We list more spectral-based GNNs’ details in the supplementary materials.

### 2.2 The motivation of revisiting ChebNet

**ChebNet versus GCN.** Even though GCN is a simplified form of ChebNet, it is well known that ChebNet is inferior to GCN for semi-supervised node classification tasks [21]. Table 1 shows the results of ChebNet and GCN for semi-supervised node classification tasks on Cora, Citeseer and Pubmed datasets (see the Appendix for experimental details). We find that ChebNet is inferior to GCN, especially when we increase the polynomial order $K$ from 2 to 10 in Equation (3).

On the other hand, existing research [11] has shown that ChebNet is more expressive than GCN in theory. In particular, ChebNet can approximate arbitrary spectral filters as $K$ increases, while GCN is a fixed low-pass filter. If we set $K = 1$ and $w_0 = w_1$ in the Equation (2), ChebNet corresponds to a high-pass filter; if we set $K = 1$ and $w_0 = -w_1$, ChebNet becomes a low-pass filter which is essentially the same as GCN. Consequently, a natural question is: Why is ChebNet’s performance inferior to GCN’s despite its better expressiveness?

**Chebyshev basis versus other bases.** Chebyshev polynomials are widely used to approximate various functions in the digital signal processing and the graph signal filtering [36, 37]. The truncated Chebyshev expansions are demonstrated to produce a minimax polynomial approximation for the analytic functions [13]. Consequently, the spectral filters can be well-approximated by a truncated expansion in terms of Chebyshev polynomials $T_k(x)$ up to $K$-th order [16].

\[ h(\hat{\lambda}) \approx \sum_{k=0}^{K} w_k T_k(\hat{\lambda}), \hat{\lambda} \in [-1, 1], \]  

where $\hat{\lambda}$ is the eigenvalue of the scaled Laplacian matrix $\hat{L}$. ChebNet [8] then defined the graph convolutions using the Chebyshev approximated filters, while recent works were inspired by ChebNet and used Monomial (i.e., GPR-GNN [6]) and Bernstein (i.e., BernNet [17]) bases to approximate filters. In order to evaluate the approximation ability of Chebyshev basis, we propose ChebBase, which simply replaces the Monomial basis of GPR-GNN and Bernstein basis of BernNet with the Chebyshev basis. The expression of ChebBase is

\[ Y = \sum_{k=0}^{K} w_k T_k(\hat{L})f_\theta(X), \]  

where $f_\theta(X)$ denotes Multi-Layer Perceptron (MLP). Table 2 reveals the results of ChebBase, GPR-GNN and BernNet for node classification tasks on three citation graphs. We can observe that

<table>
<thead>
<tr>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChebNet (2)</td>
<td>80.54±0.38</td>
<td>70.35±0.31</td>
<td>75.52±0.72</td>
</tr>
<tr>
<td>ChebNet (10)</td>
<td>74.91±0.52</td>
<td>67.69±0.64</td>
<td>65.91±1.71</td>
</tr>
<tr>
<td>GCN</td>
<td>81.32±0.18</td>
<td>71.77±0.21</td>
<td>79.15±0.18</td>
</tr>
<tr>
<td>ChebBase</td>
<td>79.29±0.36</td>
<td>70.76±0.37</td>
<td>78.07±0.32</td>
</tr>
<tr>
<td>GPR-GNN</td>
<td>83.95±0.22</td>
<td>70.92±0.57</td>
<td>78.97±0.27</td>
</tr>
<tr>
<td>BernNet</td>
<td>83.15±0.32</td>
<td>72.24±0.25</td>
<td>79.65±0.25</td>
</tr>
</tbody>
</table>

Table 1: Comparison of ChebNet and GCN.
ChebBase has the worst performance, which is inconsistent with the fact that the Chebyshev basis can approximate minimax polynomial in theory. Therefore, the second question is: Why is ChebNet’s filter inferior to that of GPR-GNN and BernNet, despite the fact that Chebyshev polynomials have a higher approximation ability?

2.3 Coefficient Constraints

We now demonstrate that ChebNet’s suboptimal performance is due to the illegal coefficients learned, which results in over-fitting. Given an arbitrary continuous function \( f(x) \) in the interval \([-1, 1]\), the Chebyshev expansion is defined as \( f(x) = \sum_{k=0}^{\infty} w_k T_k(x) \) with the Chebyshev coefficients \( w_k \).

The following theorem establishes that in order to approximate an analytic function, the Chebyshev expansion’s coefficients must be constrained.

**Theorem 2.1.** \cite{48} If \( f(x) \) is weakly singular at the boundaries and analytic in the interval \((-1, 1)\), then the Chebyshev coefficients \( w_k \) will asymptotically (as \( k \to \infty \)) decrease proportionally to \( 1/k^q \) for some positive constant \( q \).

Here, "weakly singular" means that the derivative of \( f \) can vanish at the boundaries, and "analytic" means \( f \) can be locally given by a convergent power series in the interval \((-1, 1)\). Intuitively, Chebyshev polynomial \( T_k(x) \) with larger \( k \) corresponds to higher frequency oscillation (see the Appendix for more details). Theorem 2.1 essentially demonstrates that high frequency polynomials should be constrained in the Chebyshev expansion to approximate an analytic function. Figure 1(a) depicts the Chebyshev expansion’s coefficients of the analytic function \( \exp(\hat{\lambda}) \) used as a spectral filter in GDC \cite{12} and shows that the coefficients are convergent.

The ability to approximate an analytic function is crucial in the task of approximating the spectral filters, since non-analytic filters are more difficult to approximate by polynomials and may result in over-fitting. In particular, ChebNet and ChebBase learn the coefficients \( w_k \) by gradient descent without any constraints. The coefficients may not satisfy Theorem 2.1, leading to their poor performance. To validate this conjecture, we conducted an empirical analysis of ChebBase with different coefficient constraints. Inspired by Theorem 2.1, we use the following propagation process for the ChebBase/k.

\[
Y = \sum_{k=0}^{K} \frac{w_k}{k} T_k(\hat{\mathbf{L}}) f_\theta(\mathbf{X}),
\]

where \( w_k/k \) denote the Chebyshev coefficients implemented by reparameterizing the learnable parameters \( w_k \). Table 3 shows the experimental results of the semi-supervised node classification performed on the citation graphs. We can observe that with a simple penalty on \( w_k \), ChebBase/k outperforms ChebNet, ChebBase, and GCN. Figure 1(b) plots the absolute value of the Chebyshev coefficients learnt by ChebBase and ChebBase/k on Cora. We can observe that the coefficients of ChebBase/k could more readily satisfy the convergence constraint. These results validate Theorem 2.1.

![Chebyshev coefficients of \( \exp(\hat{\lambda}) \).](image)

![Chebyshev coefficients learnt on Cora.](image)

**Figure 1:** Illustrations of the Chebyshev expansion’s coefficients of \( \exp(\hat{\lambda}) \) and the Chebyshev coefficients learnt by ChebBase and ChebBase/k on Cora.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChebNet</td>
<td>80.54±0.38</td>
<td>70.35±0.33</td>
<td>75.52±0.75</td>
</tr>
<tr>
<td>GCN</td>
<td>81.32±0.18</td>
<td>71.77±0.21</td>
<td>79.15±0.18</td>
</tr>
<tr>
<td>ChebBase</td>
<td>79.29±0.36</td>
<td>70.76±0.37</td>
<td>78.07±0.32</td>
</tr>
<tr>
<td>ChebBase/k</td>
<td><strong>82.66±0.28</strong></td>
<td><strong>72.52±0.29</strong></td>
<td><strong>79.25±0.31</strong></td>
</tr>
</tbody>
</table>

Table 3: The performance of ChebBase.
3 ChebNetII model

Although ChebBase/$k$ appears to be a promising approach, it still has some drawbacks: 1) Imposing the penalty on the coefficients is not mathematically elegant, as Theorem 2.1 only provides a necessary condition for the coefficients; 2) It is hard to impose further constraints on the learned spectral filters. For example, it is unclear how we can modify Equation (6) to obtain non-negative filters, a requirement proposed in [17]. In this section, we describe ChebNetII, a GNN model based on Chebyshev interpolation that resolves the above two issues. We also discuss the advantages and disadvantages of various polynomial interpolations as well as the Runge phenomenon.

3.1 Chebyshev interpolation

Consider a real filter function $h(\hat{\lambda})$ that is continuous in the interval $[-1, 1]$. When the values of this filter are known at a finite number of points $\hat{\lambda}_k$, one can consider the approximation by a polynomial $P_K$ with $K$ degree such that $h(\hat{\lambda}_k) = P_K(\hat{\lambda}_k)$, which is the general polynomial interpolation. We give an explicit expression of the general polynomial interpolation in the supplementary materials.

We generally sample the $K + 1$ points $\hat{\lambda}_0 < \hat{\lambda}_1 < \ldots < \hat{\lambda}_K$ uniformly from $[-1, 1]$ to construct the interpolating polynomial $P_K(\hat{\lambda})$. Intuitively, increasing $K$ should improve the approximation quality. However, this is not always the case due to the Runge Phenomenon [10] (The details are discussed in section 3.3). The popular approach to this problem in the literature [14] is Chebyshev interpolation, having superior approximation ability and faster convergence. Instead of sampling the interpolation points uniformly, ChebNet interpolation uses Chebyshev nodes as the interpolation points, which are essentially the zeros of the $(K + 1)$-th Chebyshev polynomial.

Definition 3.1. (Chebyshev Nodes) The Chebyshev polynomial $T_k(x)$ satisfies the closed form expression $T_k(x) = \cos (k \arccos(x))$. The Chebyshev Nodes for $T_k(x)$ are defined as $x_j = \cos \left(\frac{2j+1}{2K} \pi\right), j = 0, 1, \ldots, k - 1$, which lie in the interval $(-1, 1)$ and are the zeros of $T_k(x)$.

Definition 3.1 suggests that each Chebyshev polynomial $T_k(x)$ has $k$ zeros, and we can define Chebyshev interpolation by replacing the equispaced points with Chebyshev nodes in the general polynomial interpolation (see the Appendix for details). More eloquently, definition 3.2 efficiently defines the Chebyshev interpolation via their orthogonality properties.

Definition 3.2. (Chebyshev Interpolation) [14] Given a continuous filter function $h(\hat{\lambda})$, let $x_j = \cos \left(\frac{j+1/2}{K+1} \pi\right), j = 0, \ldots, K$ denote the Chebyshev nodes for $T_{K+1}$ and $h(x_j)$ denotes the function value at node $x_j$. The Chebyshev interpolation of $h(\hat{\lambda})$ is defined to be

$$P_K(\hat{\lambda}) = \sum_{k=0}^{K} c_k T_k(\hat{\lambda}), c_k = \frac{2}{K+1} \sum_{j=0}^{K} h(x_j) T_k(x_j), \quad (7)$$

where the prime indicates the first term is to be halved, that is, $c'_0 = c_0/2, c'_1 = c_1, \ldots, c'_K = c_K$.

3.2 ChebNetII via Chebyshev Interpolation

Inspired by Chebyshev interpolation, we propose ChebNetII, a graph convolutional network that approximates an arbitrary spectral filter $h(\hat{\lambda})$ with an optimal convergence rate. ChebNetII simply reparameterizes the filter value $h(x_j)$ in Equation (7) as a learnable parameter $\gamma_j$, which allows the model to learn an arbitrary spectral filter via gradient descent. More precisely, the ChebNetII model can be formulated as

$$Y = \frac{2}{K+1} \sum_{k=0}^{K} \sum_{j=0}^{K} \gamma_j T_k(x_j) T_k(\hat{\lambda}) f_\theta(X), \quad (8)$$

where $x_j = \cos ((j + 1/2) \pi/(K + 1))$ are the Chebyshev nodes of $T_{K+1}$, $f_\theta(X)$ denotes an MLP on the node feature matrix $X$, and $\gamma_j$ for $j = 0, 1, \ldots, K$ are the learnable parameters. Note that similar to APPNP [22], we decouple feature propagation and transformation.
Consequently, the filtering operation of ChebNetII can be expressed as

\[ y \approx \frac{2}{K+1} \sum_{k=0}^{K} \sum_{j=0}^{K} \gamma_j T_k(x_j) T_k(\hat{L}) x. \]  

(9)

It is easy to see that compared to the filtering operation of the original ChebNet [2], we only make one simple change: reparameterizing the coefficient \( w_k \) by \( w_k = \frac{2}{K+1} \sum_{j=0}^{K} \gamma_j T_k(x_j) \). However, this simple modification allows us to have more control on shaping the resulting filter, as Chebyshev interpolation suggests that \( \gamma_j \) directly corresponds to the filter value \( h(x_j) \) at the Chebyshev node \( x_j \). The coefficients \( w_k = \frac{2}{K+1} \sum_{j=0}^{K} h(x_j) T_k(x_j) \) are fundamentally guaranteed to satisfy the constraints of Theorem 2.1 since we directly approximate the filter \( h \). Furthermore, Chebyshev interpolation also provides the ChebNetII with several beneficial mathematical properties.

### 3.3 Analysis of ChebNetII

ChebNetII has several advantages over existing GNN models due to the unique nature of Chebyshev interpolation. From the standpoint of polynomial approximation and computational complexity, we compare ChebNetII with current related approaches such as GPR-GNN [6] and BernNet [17].

**Near-minimax approximation.** First of all, we examine ChebNetII’s capabilities in terms of filter function approximation. Theorem 3.1 exhibits that ChebNetII provides an approximation that is close to the best polynomial approximation for a spectral filter \( h \).

**Theorem 3.1.** [27] A polynomial approximation \( P^*_K(x) \) for a function \( f(x) \) is said to be near-best/minimax approximation with a relative distance \( \rho \) if

\[ ||f(x) - P^*_K(x)|| \leq (1 + \rho)||f(x) - P^K(x)||, \]

(10)

where \( \rho \) is the Lebesgue constant, \( P^K(x) \) is a best polynomial approximation, and \( || \cdot || \) represents the uniform norm (i.e., \( ||g|| = \max_{x \in [-1, 1]} |g(x)| \)). Then, we have \( \rho \sim 2^K \) as \( K \to \infty \) for the general polynomial interpolation, and \( \rho \sim \log(K) \) as \( K \to \infty \) for the Chebyshev interpolation.

**Convergence.** In comparison to BernNet [17], which uses the Bernstein basis, ChebNetII has a faster convergence rate for approximating a filter function. Specifically, we have the following Theorem:

**Theorem 3.2.** [14][29] Let \( P_K(x) \) be the polynomial approximation for a function \( f(x) \). Then the error is given as \( ||f(x) - P_K(x)|| \leq E(K) \). If \( P_K(x) \) is obtained by Bernstein approximation, then \( E(K) \sim (1 + (2K)^{-2})\omega(K^{-1/2}) \). If \( P_K(x) \) is obtained by Chebyshev Interpolation, then \( E(K) \sim C\omega(K^{-1}) \log(K) \) with a constant \( C \), where \( \omega \) is the modulus of continuity.

**Runge phenomenon.** In comparison to GPR-GNN [6], which uses the Monomial basis, ChebNetII has the advantage of reducing the Runge phenomenon [10]. In particular, when we use the general polynomial interpolation to approximate a Runge filter \( h(\hat{\lambda}) \) with a high degree over a set of equispaced interpolation points, it will cause oscillation along the edges of an interval. Consequently, as the degree of the polynomial increases, the interpolation error increases. Following [14], we define the error of polynomial interpolation as

\[ R_K(\hat{\lambda}) = h(\hat{\lambda}) - P_K(\hat{\lambda}) = \frac{h^{K+1}(\zeta)}{(K+1)!} \pi_{K+1}(\hat{\lambda}), \]

(11)

where \( \pi_{K+1}(\hat{\lambda}) = \prod_{k=0}^{K}(\hat{\lambda} - \hat{\lambda}_k) \) denotes the nodal polynomial and \( \zeta \) is the value depending on \( \hat{\lambda} \). The terrible Runge phenomenon is caused by the values of this nodal polynomial, which have very
Table 5: Dataset statistics of large heterophilic graphs.

<table>
<thead>
<tr>
<th></th>
<th>Penn94</th>
<th>pokc</th>
<th>arXiv-year</th>
<th>genius</th>
<th>twitch-gamers</th>
<th>wiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>41,554</td>
<td>1,632,803</td>
<td>169,343</td>
<td>421,961</td>
<td>168,114</td>
<td>1,925,342</td>
</tr>
<tr>
<td>Edges</td>
<td>1,362,229</td>
<td>30,622,564</td>
<td>1,166,243</td>
<td>984,979</td>
<td>6,797,557</td>
<td>303,434,860</td>
</tr>
<tr>
<td>Features</td>
<td>5</td>
<td>65</td>
<td>128</td>
<td>12</td>
<td>7</td>
<td>600</td>
</tr>
<tr>
<td>Classes</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>$\mathcal{H}(G)$</td>
<td>0.47</td>
<td>0.46</td>
<td>0.22</td>
<td>0.62</td>
<td>0.55</td>
<td>0.39</td>
</tr>
</tbody>
</table>

On the other hand, we have the following Theorem 3.3 that explains that Chebyshev nodes can minimize and quantify this error caused by the nodal polynomial, meaning Chebyshev interpolation minimizes the problem of the Runge phenomenon.

**Theorem 3.3.** Consider the Chebyshev nodes $x_j = \cos ((j + 1/2) \pi/(K + 1))$, $j = 0, 1, ..., K$. Then the nodal polynomial $\hat{T}_{K+1}(x) = \prod_{k=0}^{K} (x - x_j)$ has the smallest possible uniform norm, i.e., $\|\hat{T}_{K+1}(x)\| = 2^{-K}$.

**Computational complexity.** Compared to BernNet [17], which has a time complexity quadratic to the order $K$ in the forward process, ChebNetII can be computed in time linear to $K$. Specifically, we first compute the ChebNetII’s coefficients $2^{K+1} \sum_{j=0}^{K} \gamma_j T_k(x_j)$ in time linear to $K$ as we can precompute $T_k(x_j)$, and then plug the coefficients into Equation (8) for propagation, which also takes the time linear to $K$, the same as that of ChebNet [8] and GPR-GNN [6].

4 Experiments

In this section, we conduct experiments to evaluate the performance of ChebNetII against the state-of-the-art graph neural networks on a wide variety of open graph datasets.

**Dataset and Experimental setup.** We evaluate ChebNetII on several real-world graphs for the Semi- and Full-supervised node classification tasks. The datasets include three homophilic citation graphs: Cora, Citeseer, and Pubmed [35, 45], five heterophilic graphs: the Wikipedia graphs Chameleon and Squirrel [34], the Actor co-occurrence graph, and webpage graphs Texas and Cornell from WebKB [30], two large citation graphs: ogbn-arxiv and ogbn-papers100M [18], as well as six large heterophilic graphs: Penn94, pokec, arXiv-year, genius, twitch-gamers and wiki [25]. We measure the level of homophily in a graph using the edge homophily ratio $\mathcal{H}(G) = \frac{|\{(u,v) : (u,v) \in E \wedge y_u = y_v\}|}{|E|}$, where $y_v$ denotes the label of node $v$. We summarize the dataset statistics in Tables 4 and 5. All the experiments are carried out on a machine with an NVIDIA RTX8000 GPU (48GB memory), Intel Xeon CPU (2.20 GHz) with 40 cores, and 512 GB of RAM.

4.1 Semi-supervised node classification with polynomial-based methods

**Setting and baselines.** For the semi-supervised node classification task, we compare ChebNetII to 7 polynomial approximation filter methods, including MLP, GCN [21], ARMA [2], APPNP [22], ChebNet [8], GPR-GNN [6] and BernNet [17]. For dataset splitting, we employ both random and fixed splits and report the results on random splits. The results of fixed splits will be discussed in the Appendix. Specifically, we apply the standard training/validation/testing split [45] on the three homophilic citation datasets (i.e., Cora, Citeseer, and Pubmed), with 20 nodes per class for training, 500 nodes for validation, and 1,000 nodes for testing. Since this standard split cannot be used for very small graphs (e.g., Texas), we use the sparse splitting [6] with the training/validation/test sets accounting for 2.5%/2.5%/95%, respectively, on the five heterophilic datasets.

For ChebNetII, we use Equation (8) as the propagation process and use the ReLu function to reparametrize $\gamma_j$, maintaining the non-negativity of the filters [17]. We set the hidden units as 64...
and $K = 10$ for all the datasets as the same as GPR-GNN [6] and BernNet [17]. We employ the Adam SGD optimizer [20] with an early stopping of 200 and a maximum of 1000 epochs to train ChebNetII. We use the officially released code for GPR-GNN and BernNet and use the Pytorch Geometric library implementations [11] for other models (i.e., MLP, GCN, APPNP, ARMA, and ChebNet). More details of hyper-parameters and baselines’ settings are listed in the Appendix.

**Results.** We utilize accuracy (the micro-F1 score) with a 95% confidence interval as the evaluation metric. Table 6 reports the relevant results on 10 random splits. Boldface letters indicate the best result for the given confidence interval, and underlinings denote the next best result. We first observe that ChebNet is inferior to GCN even on heterophilic graphs, which concurs with our theoretical analysis that the illegal coefficients learned by ChebNet lead to over-fitting. ChebNetII, on the other hand, outperforms other methods on all datasets excluding Pubmed, where it achieves top-2 classification accuracy. This quality is due to the fact that the learnable parameters $\gamma_j$ of ChebNetII directly correspond to the filter value $h(x_j)$ at the Chebyshev node $x_j$, effectively preventing it from learning an illegal filter.

### 4.2 Full-supervised node classification

**Setting and baselines.** For full-supervised node classification, we compare ChebNetII to the baselines in the prior semi-supervised node classification. We also include GCNII [5], TWIRLS [44], EGNN [51] and PDE-GCN [9] four competitive baselines for full-supervised node classification. For all datasets, we randomly split the nodes into 60%, 20%, and 20% for training, validation and testing, and all methods share the same 10 random splits for a fair comparison, as suggested in [30] [6] [17].

For ChebNetII, we also set the hidden units to be 64 and $K = 10$ for all datasets, and employ the same training manner as in the semi-supervised node classification task. For GCNII, TWIRLS, EGNN and PDE-GCN we use the officially released code. More details of hyper-parameters and baselines’ settings are listed in the Appendix.

**Results.** Table 7 reports the mean classification accuracy of each model. We first observe that, given more training data, ChebNetII outperforms GCNII on both homophilic and heterophilic datasets, which demonstrates the effectiveness of the Chebyshev approximation. However, we also observe that ChebNetII achieves new state-of-the-art results on 7 out of 8 datasets and competitive results on Pubmed. Notably, ChebNetII outperforms GPR-GNN and BernNet by over 10% on the
works for GPR-GNN [6] and BernNet [17], so we report their results in this manner when ChebNetII fails to finish the preprocessing in 24 hours. We can observe that ChebNetII outperforms both datasets, BernNet on ogbn-papers100M as BernNet has a time complexity quadratic to the order K and fails to finish the preprocessing in 24 hours. We attribute that ChebNetII can precompute $\tilde{L}^kX$ for $k \in 0, \cdots, K$ in the preprocessing, we can scale it to large graphs. Specifically, we use the below propagation expression.

$$Y = f_\theta(Z), \quad Z = \frac{2}{K + 1} \sum_{k=0}^{K} \sum_{j=0}^{K} \gamma_j T_k(x_j) T_k(\tilde{L}) X.$$

The pre-computed $\tilde{L}^kX$ allow us to train $\gamma_j$ and $f_\theta(\cdot)$ in a mini-batch manner. This approach also works for GPR-GNN [6] and BernNet [17], so we report their results in this manner when ChebNetII does. We evaluate the scalability of ChebNetII on the large heterophilic graphs [25] and the widely used OGB datasets [18].

On the large heterophilic graphs, we compare ChebNetII to eight competitive baselines, including MLP, LINK [50], LINKX [25], GCN [21], GCNII [5], ChebNet [3], GPR-GNN [6] and BernNet [17]. For ChebNetII, we use Equation (12) as the propagation process on pokec and wiki and Equation (8) on the four remaining datasets. We establish the experimental setting following [25] and use the published baselines’ results, excluding GPR-GNN. More details are listed in the Appendix. Table 8 reports the mean results of each method over 5 runs. ChebNetII outperforms all other methods on 4 out of 6 datasets and the polynomial-based methods on arXiv-year and twitch-gamers. Notably, LINK and LINKX outperform ChebNetII on arXiv-year because they use a directed graph on this dataset. Using the directed graphs to the spectral-based GNNs is a future meaningful work because the current spectral graph theory only applies to undirected graphs. For the largest heterophilic graph wiki, ChebNetII obtains a new state-of-the-art result. We attribute that ChebNetII can precompute $\tilde{L}^kX$ without the graph sampling used in LINK and LINKX and has a strong filter approximation ability.

<table>
<thead>
<tr>
<th>Method</th>
<th>Penn94</th>
<th>pokec</th>
<th>arXiv-year</th>
<th>genius</th>
<th>twitch-gamers</th>
<th>wiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>MLP</td>
<td>73.61±0.04</td>
<td>62.37±0.02</td>
<td>36.70±0.21</td>
<td>86.68±0.09</td>
<td>60.92±0.07</td>
<td>37.38±0.21</td>
</tr>
<tr>
<td>LINK</td>
<td>80.79±0.49</td>
<td>80.54±0.03</td>
<td>53.97±0.18</td>
<td>73.56±0.14</td>
<td>64.85±0.21</td>
<td>57.11±0.26</td>
</tr>
<tr>
<td>LINKX</td>
<td>84.71±0.52</td>
<td>82.04±0.07</td>
<td>56.00±1.34</td>
<td>90.77±0.27</td>
<td>66.06±0.19</td>
<td>59.80±0.41</td>
</tr>
<tr>
<td>GCN</td>
<td>82.47±0.27</td>
<td>75.45±0.17</td>
<td>46.02±0.26</td>
<td>87.42±0.37</td>
<td>62.18±0.26</td>
<td>OOM</td>
</tr>
<tr>
<td>GCNII</td>
<td>82.92±0.59</td>
<td>78.94±0.11</td>
<td>47.21±0.28</td>
<td>90.24±0.09</td>
<td>63.39±0.61</td>
<td>OOM</td>
</tr>
<tr>
<td>ChebNet</td>
<td>82.59±0.31</td>
<td>72.71±0.66</td>
<td>46.76±0.24</td>
<td>89.36±0.31</td>
<td>62.31±0.37</td>
<td>OOM</td>
</tr>
<tr>
<td>GPR-GNN</td>
<td>83.54±0.32</td>
<td>80.74±0.22</td>
<td>45.97±0.26</td>
<td>90.15±0.30</td>
<td>62.59±0.38</td>
<td>58.73±0.34</td>
</tr>
<tr>
<td>BernNet</td>
<td>83.26±0.29</td>
<td>81.67±0.17</td>
<td>46.34±0.32</td>
<td>90.47±0.33</td>
<td>64.27±0.31</td>
<td>59.02±0.29</td>
</tr>
<tr>
<td>ChebNetII</td>
<td>84.86±0.33</td>
<td>82.33±0.28</td>
<td>48.53±0.31</td>
<td>90.85±0.32</td>
<td>65.03±0.27</td>
<td>60.95±0.39</td>
</tr>
</tbody>
</table>

Table 8: Experimental results on large heterophilic graphs. OOM denotes "out of memory".

4.3 Scalability of ChebNetII

For ChebNetII, if we calculate and save $T_k(\tilde{L})X$ for $k \in 0, \cdots, K$ in the preprocessing, we can scale it to large graphs. Specifically, we use the below propagation expression.

$$Y = f_\theta(Z), \quad Z = \frac{2}{K + 1} \sum_{k=0}^{K} \sum_{j=0}^{K} \gamma_j T_k(x_j) T_k(\tilde{L}) X.$$  (12)

The mean classification accuracy on large graphs. OOM denotes "out of memory" and "-" means failing to finish preprocessing in 24h.

<table>
<thead>
<tr>
<th>Method</th>
<th>ogbn-arxiv</th>
<th>ogbn-papers100M</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>71.74±0.29</td>
<td>OOM</td>
</tr>
<tr>
<td>ChebNet</td>
<td>71.12±0.22</td>
<td>OOM</td>
</tr>
<tr>
<td>ARMA</td>
<td>71.47±0.25</td>
<td>OOM</td>
</tr>
<tr>
<td>GPR-GNN</td>
<td>71.78±0.18</td>
<td>65.89±0.35</td>
</tr>
<tr>
<td>BernNet</td>
<td>71.96±0.27</td>
<td>-</td>
</tr>
<tr>
<td>SIGN</td>
<td>71.95±0.12</td>
<td>65.68±0.16</td>
</tr>
<tr>
<td>GBP</td>
<td>72.21±0.17</td>
<td>65.23±0.31</td>
</tr>
<tr>
<td>NDLS*</td>
<td>72.24±0.21</td>
<td>65.61±0.29</td>
</tr>
<tr>
<td>ChebNetII</td>
<td>72.32±0.23</td>
<td>67.18±0.32</td>
</tr>
</tbody>
</table>

Table 9: Mean classification accuracy on large graphs. OOM denotes "out of memory" and "-" means failing to finish preprocessing in 24h.
which we attribute to Chebyshev Interpolation’s superior approximation quality. These results also show that ChebNetII has lesser complexity and greater scalability than BernNet.

4.4 Comparison of Different Polynomial Bases

We perform numerical studies comparing the Chebyshev basis to the Monomial and Bernstein bases to demonstrate ChebNetII’s approximation power. Considering a Runge filter \( h(\hat{\lambda}) = \frac{1}{1 + 25\hat{\lambda}^2}, \hat{\lambda} \in [-1, 1] \), Figures 2(a) and 2(b) depict the approximation results and errors for several polynomial bases, with the polynomial degree \( K \) denoted by the numbers in brackets. We find that the Chebyshev basis has a faster convergence rate than the Bernstein basis and does not exhibit the Runge phenomenon compared to the Monomial basis. Notably, JacobiConv \[40\] investigated different polynomial bases at the same period and discovered that orthogonal polynomial bases could learn graph filters more effectively. These findings provide empirical motivations for designing GNNs with Chebyshev interpolation.

5 Conclusion

This paper revisits the problem of approximating spectral graph convolutions with Chebyshev polynomials. We show that ChebNet’s inferior performance is primarily due to illegal coefficients learned by approximating analytic filter functions, which leads to over-fitting. Moreover, we propose ChebNetII, a new GNN model based on Chebyshev interpolation, enhancing the original Chebyshev polynomial approximation while reducing the Runge phenomenon. Experiments show that ChebNetII outperforms SOTA methods in terms of effectiveness on real-world both homophilic and heterophilic datasets. For future work, a promising direction is further to improve the performance of ChebNetII on large graphs and investigate the scalability of spectral-based GNNs.

Acknowledgments and Disclosure of Funding

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. This research was supported in part by the major key project of PCL (PCL2021A12), by National Natural Science Foundation of China (No. 61972401, No. 61932001, No. 61832017, No. U2001212), by Beijing Natural Science Foundation (No. 4222028), by Beijing Outstanding Young Scientist Program No. BJWZYJH01201910002098, by Alibaba Group through Alibaba Innovative Research Program, by CCF-Baidu Open Fund (NO.2021PP15002000) and by Huawei-Remin University joint program on Information Retrieval. We wish to acknowledge the discussion with Professor Zhouchen Lin from Peking University. 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


