Abstract

Recently, linear GCNs have shown competitive performance against non-linear ones with less computation cost, and the key lies in their propagation layers. Spectral analysis has been widely adopted in designing and analyzing existing graph propagations. Nevertheless, we notice that existing spectral analysis fails to explain why existing graph propagations with the same global tendency, such as low-pass or high-pass, still yield very different results. Motivated by this situation, in this paper, we develop a new framework for spectral analysis from a detailed spectral analysis called concentration analysis. In particular, we propose three attributes: concentration centre, maximum response, and bandwidth for our analysis. Through a dissection of the limitations of existing graph propagations via the above analysis, we proposed a new kind of propagation layer, Graph Gaussian Convolution Networks (G2CN), in which the three properties are decoupled and the whole structure becomes more flexible and applicable on different kinds of graphs. Extensive experiments show that we can obtain state-of-the-art performance on various benchmark datasets with our proposed Graph Gaussian Convolution Networks.

1. Introduction

Graph neural networks have achieved excellent performance on various graph-related tasks on social relationship (Li & Goldwasser, 2019; Qiu et al., 2018; Tong et al., 2019), traffic (Bogaerts et al., 2020; Cui et al., 2019; Li et al., 2018b), recommendation (Ying et al., 2018; Feng et al., 2022), medical researches and others (Zhao et al., 2019; Rathi et al., 2020; Jiang et al., 2021; Chen et al., 2022). GCNs mainly consist of the graph propagation part and feature transformation part. The graph propagation part first processes the graph signals to obtain the features with global or local graph properties. Then the signals are fed to a neural network with several or only one layer. Several works (Wu et al., 2019; Zhu & Koniusz, 2020) have shown that even a linear transformation layer can achieve better performance on many benchmark datasets. Therefore, graph propagation is the key part of a GCN model and attracts a lot of attention these days. Among the researches on graph propagation, graph spectral analysis has been widely adopted in designing and analyzing the existing graph propagation (Wu et al., 2019; Balcilar et al., 2020). It treats the graph propagations as graph filters on the spectrum of the graph Laplacian and classifies various graph propagation via the spectral response. For example, a graph filter that responds to amplify eigenvalues and diminish large eigenvalues will be classified as a low-pass graph filter.

However, the former spectral analysis mainly focuses on the global tendency. These analyses cannot completely explain why GCNs with different graph propagation will show different results even if their corresponding graph filter share the same low-pass or high-pass tendency. Motivated by this, we'd analyze the filters’ pass band since it determines the output of the graph propagation. Since the filters’ responses are concentrated on the passband, we called the analysis as spectral concentration analysis to distinguish from the previous study. And we propose three attributes called the concentration attributes for spectral analysis on the graph propagation contrast to the filter’s global tendency: maximum response $R$, bandwidth $BW$ and band concentration centre $b$ in the following papers.

We analyze existing graph propagation with these attributes and find that their different performance can be attributed to their various concentration flexibility. Models with high concentration flexibility can apply different graph filters on various graphs, leading to better performance. With these findings, we also propose our Gaussian Graph Convolution, whose graph filters can be assumed as the combination of several Gaussian filters. Our Gaussian graph filter decouples the above attributes and can be easily adapted to perform as different kinds of filters to suit the graph desire. Then we implement our Gaussian graph filter in a linear graph model.
called Graph Gaussian Convolution Networks (G^2CN). The experiments show that our G^2CN can obtain state-of-art performance on various benchmark datasets with efficient computation cost. We summarize the contributions of our work as below:

1. We extend the former spectral analysis to our concentration spectral analysis with our proposed concentration attributes. Compared with the former study, our proposed one not only focuses on the graph filters global tendency. And we can explain the different performances of various graph propagation with the same global trend.

2. We proposed Gaussian Graph Convolution and its graph propagation formulation from the above analysis. Our proposed filter enjoys sufficient concentration flexibility and can apply to different graphs with better performance, especially the heterophily graphs.

3. We propose the Graph Gaussian Convolution Networks (G^2CN) with our Gaussian Graph Convolution, an efficient linear graph model that can achieve state-of-art performance on the node-classification tasks.

2. Related Works

Graph Neural Networks can be generally classified into spectral-based models (Bruna et al., 2013; Henaff et al., 2015; Li et al., 2018a), which use the eigendecomposition of the graph Laplacian, and spatial based model, which directly uses the graph Laplacian for graph propagation without the decomposition, which is widely used due to its efficiency. In this paper, we only discuss the latter propagation type. And we can divide the graph propagation for the current spatial-based GCNs from the spectral view into two kinds: GCNs with designed graph filters or learnable graph filters.

3.1. Notations and Preliminaries on Graph Propagation

For a graph defined as $G = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V} = \{v_1, ..., v_n\}$ denotes the vertex set of $n$ nodes which contains a subset labeled nodes $\mathcal{V}_l \subset \mathcal{V}$ for training and unlabeled nodes $\mathcal{V}_u = \mathcal{V} \setminus \mathcal{V}_l$ to predict in the node classification task. $\mathcal{A} \in \mathbb{R}^{n \times n}$ is the adjacent matrix where $a_{ij}$ denotes the edge weight between node $v_i$ and $v_j$. The degree matrix $D = \text{Diag}(d_1, ..., d_n)$ of $\mathcal{A}$ is a diagonal matrix with its $i$-th diagonal entry as $d_i = \sum_j a_{ij}$. Each node $v_i$ is represented by a $d$-dimensional feature vector $x_i \in \mathbb{R}^d$ and we use $X \in \mathbb{R}^{n \times d} = [x_1, ..., x_n]^\top$ to denote the feature matrix and use a one-hot vector $y_i \in \{0, 1\}^C$ to denote the label of $i$-th node of $C$ classes. In our paper, we only discuss the normalized graph Laplacian matrix defined as $L = I - D^{-1/2}A D^{-1/2} \in S_+^n$ for convenience. The results can easily be extended to other types of Laplacian. Its eigen-decomposition can be depicted as $U \Lambda U^\top$, where $\Lambda$ is the diagonal matrix consists of $L$'s eigenvalues, and $U \in \mathbb{R}^{n \times n}$ is made by the eigenvectors of the Laplacian matrix.

Then we define the graph Fourier transform of a graph singal $x$ as $\hat{x} = U^\top x$ and its inverse is $x = U \hat{x}$. Thus the graph
propagation for signal \(x\) with kernel \(g\) can be defined as:
\[
H = g \ast g \cdot x = U ( (U^T g) \odot Ux) = U \hat{G} U^T X,
\]
where \(\hat{G} = diag(\hat{g}_1, ..., \hat{g}_2)\) denotes the spectral kernel coefficients. Exactly computing the graph convolution on a random kernel \(g\) is complicated due to the eigendecomposition of graph’s Laplacian consumes unaffordable computation complexity. For this count, as we illustrate in the following, current works with spatial convolution usually use the polynomial functions \(g(\lambda)\) to approximate different kernels:
\[
H = g(L)x = U g(\Lambda) U^T X,
\]
where \(g(\Lambda) = Diag (g(\lambda_1), ..., g(\lambda_n))\) and we call \(g\) the graph filter function for above graph propagation Eqn. (2).
Then GCNs with such graph propagation can be considered those \(g\) preserve small \(\lambda\) can be regarded as low-pass filters while those who keep large \(\lambda\) are high-pass filters. Apart from the graph filters global tendency, we also analyzed the concentrated properties via the concentration attributes defined as follows.

### 3.2. Concentration Attributes on Graph Propagation

First, we’d like to define the graph filters maximum response and the concentration centre:

**Definition 3.1.** The maximum response \(R_g\) for graph filter \(g\) is denoted as:
\[
R_g = \max_{\lambda \in [0, 2]} |g(\lambda)|,
\]
and the concentration centre can be defined as:

**Definition 3.2.** Then we define \(b\) as the centre for graph filter \(g\) if,
\[
g(b) = R_g,
\]
Then we give the definition for the graph filter’s bandwidth.

**Definition 3.3.** The bandwidth \(BW_g\) for a given graph filter \(g(\lambda)\) can be defined as:
\[
BW_g = \int_0^2 \mathbb{I}(g(\lambda), \frac{R_g}{\sqrt{2}}) d\lambda,
\]
where \(\mathbb{I}(g(\lambda), \frac{R_g}{\sqrt{2}})\) is a indicator function:
\[
\mathbb{I}(\gamma_1, \gamma_2) = \begin{cases} 
1, & \text{if } \gamma_1 \geq \gamma_2 \\
0, & \text{if } \gamma_1 < \gamma_2
\end{cases}
\]
for \(\gamma_1, \gamma_2 \in \mathbb{R}\).

The above attributes tell which spectrum the graph filter will amplify to diminish. And different graph filters enjoy unique concentration attributes and lead to different performances, as we discussed. The bandwidth and concentration centre should be flexible to ensure the proper graph propagation for a different dataset, while the maximum response should be bounded for stable training. Then we try to dissect existing graph propagations via our spectral concentration analysis.

### 3.3. Dissecting Existing Designed Graph Propagations

**Graph Convolutional Networks (GCN).** Firstly, we dissect GCN (Kipf & Welling, 2017) via our concentration spectral analysis. The graph propagation for GCN can be formulated as:
\[
H_{gcn} = (2I - L)^K X
\]
where \(K \in \mathbb{Z}^+\) is the propagation steps for SGC. Its graph filter can be formulated as \(g_{gcn}(\lambda) = (2 - \lambda)^K\), which is also a low-pass filter, whose concentration centre is 0 with \(R_{gcn} = 2^K\). And its bandwidth can be formulated as:
\[
BW_{gcn} = 2 - 2^1 - \frac{\pi}{\lambda}.
\]
One can see that the bandwidth of GCN and its latter works (Wu et al., 2019; Xu et al., 2018) are determined by its propagation steps. Therefore, the bandwidth for GCN’s graph filter is limited and such defects restrict its performance since it cannot flexibly implement low-pass filters with different bandwidths as the graph desires.

**Personalized PageRank (PPR).** Then Klicpera et al. (2019a) proposed the personalized PageRank (PPR) graph propagation, trying to make the graph filter more flexible. Its graph propagation tries to approximate:
\[
H_{ppr} = (I - (1 - \alpha)(I - L))^{-1} X,
\]
with \(\alpha \in (0, 1)\) to ensure the inverse exists. Its graph filter is \(g_{ppr} = \frac{1}{1 - (1 - \alpha)(1 - \lambda)}\) and its concentration centres are both fixed at \(\lambda = 0\) with \(R_{ppr} = \frac{1}{\alpha}\) also a low pass filter. The bandwidth for PPR can be formulated as:
\[
BW_{ppr} = \frac{\sqrt{2} - 1 + \alpha}{1 - \alpha}.
\]
Since the bandwidth for PPR can continuously change with different \(\alpha\), models with PPR propagation are more flexible than original GCN and enjoy better performance. Whereas when \(BW_{ppr} \rightarrow 0, \alpha \rightarrow 0\) and its maximum response will explode \(R_{ppr} = \frac{1}{\alpha} \rightarrow \infty\). Therefore, we cannot choose a narrow PPR filter in practice, although the bandwidth for PPR can vary from 0 to \(\infty\) in theory.

**Frequency Adaptation GCN (FAGCN).** The spectral filter for FAGCN (Bo et al., 2021) is the combination of:
\[
h_{\text{FAGCN}_L}(\lambda) = (1 - \lambda + \epsilon)^2,
\]
\[
h_{\text{FAGCN}_H}(\lambda) = (\lambda - 1 + \epsilon)^2,
\]
with $\epsilon \in [0, 1]$. From the above equation, one can see the center of FAGCN’s base filters are fixed at 0 or 2 like GCN. Their bandwidth are $BW_{\text{FAGCN}} = (2^{1/4} - 1)(1 + \epsilon) \in [0.19, 0.38]$ and maximum responses are $(1 + \epsilon)^2$ which are less flexible than ours.

**Auto-Regressive Moving Average (ARMA).** The spectral filter for ARMA (Bianchi et al., 2021) is the combination of:

$$h_{\text{ARMA}}(\mu) = \frac{b_K}{1 - a_K \mu}, \quad \text{(10)}$$

with $\mu = (\lambda_{\text{max}} - \lambda_{\text{min}})/2 - \lambda$, $\lambda$ is eigenvalue of graph Laplacian, $|a_K| < 1$ and $b_K$. It is a low-pass filter when $a_K > 0$ with fixed center $\lambda = 0$ and it is a high-pass filter when $a_K < 0$ with fixed center $\lambda = 2$. In most cases, $\lambda_{\text{max}} = 2$ and $\lambda_{\text{min}} = 0$, then maximum response is $\frac{b_K}{1 - |a_K|}$ with bandwidth $BW_{\text{ARMA}} = \frac{\sqrt{2} - 1}{|a_K|}$ which performs like APPNP but less flexible than ours.

**Heat Kernels (HK).** Apart from the above models, another widespread designed graph propagation is the heat kernel propagation (Klicpera et al., 2019b; Wang et al., 2021) which is inspired by the heat diffusion. Such propagation tries to approximate the following graph propagation:

$$H_{\text{HK}} = e^{-TLX}, \quad \text{(11)}$$

where $T \in \mathbb{R}^+$ is the diffusion factor and its graph filter is an exponential function $g_{\text{dac}}(\lambda) = e^{-T \lambda}$. Its maximum response is fixed to be 1 with fixed centre $\lambda = 0$. And the bandwidth for HK is:

$$BW_{\text{HK}} = \frac{\log(\sqrt{2})}{T}, \quad \text{(12)}$$

The above formula shows that the bandwidth of DGC is flexible and can be easily controlled by $T$, and the fixed maximum response can also stabilize the training procedure. Thereby, as illustrated in (Klicpera et al., 2019b; Wang et al., 2021), heat kernels can show better performance against the above graph propagations on most datasets. However, the unflexible concentration centre for the above models also hinders their performance on many datasets since these low-pass models will permanently cut off the graph information corresponding to large eigenvalues of the graph Laplacian.

### 3.4. Dissecting Graph Propagation with Learnable Graph Propagations

To approximate the graph filters with different concentration centres, many graph models combine a series of polynomial bases with learnable weights as their graph propagation. Although these models can theoretically approximate every filter with proper weight for each basis, different properties for the basis will inflect the difficulties of weight training and lead to different performances. Thereby, this section discusses the concentration attributes for a single filter basis.

**Chebyshev Polynomials.** ChebNet (Defferrard et al., 2016) which utilizes the Chebyshev Polynomials to approximate various graph filters, whose $i$-th basis can be formulated as:

$$C^{(i)} = 2C^{(2)}C^{(s-1)} - C^{s-2} \quad \text{(13)}$$

with $C^{(0)} = 0$ and $C^{(1)} = \frac{2b_K}{a_K}$ and its propagation for $i$-th basis is $H_{\text{Cheb}}^i = C^{(2)}X$. Although we cannot give the closed-form formula for the concentration centre and bandwidth for its $i$-th basis, its concentration centre and bandwidth are determined by its index $i$. We left the detailed calculation of these attributes for ChebNet filters in the Appendix E. The bases for ChebNet have more than one concentration centre lies in $[0, 2]$ and relies on the complicated weights to approximate the usually used filters like Comb Filters as experiment shows in (He et al., 2021).

**Bernstein Polynomials.** BernNet (He et al., 2021) which sum the Bernstein bases with weights for their network, the graph propagation for $i$-th basis for an order-$K$ BernNet graph propagation can be formulated as follows:

$$H_{\text{Bern}}^{K,i} = \frac{1}{2^K} \binom{K}{k} (2I - L)^{K-k} L^k, \quad \text{(14)}$$

whose corresponding graph filter can be simplified as $(2 - \lambda)^{K-i} \lambda^i$. Its concentration centre of with order $K$ is $\frac{2i}{K}$, which cannot continuously change since $K \in \mathbb{Z}^+$. But when comparing with ChebNet, each Bernstein filters only have one concentration centre, which makes the weights’ training easier and leads to better performance.

The concentration centre and bandwidth for each bernstein basis is fixed by its index $i$ and order $K$ as we show in the Appendix. When the objective filter is similar to their bases, the approximation is easy. And single concentration centre for each basis make it easier to approximate different filters compared with ChebNet. Since real-world graphs usually needs passbands of low or high frequency with various band-widths, BernNet needs a complicated linear transformation layer to train a complicated weight and then they can obtain a complicated weights for bases during training. Thereby, the BernNet is not efficient and cannot easily adapt on different graphs. We summarized the analysis above as a Table in Appendix.

### 4. The proposed G$^2$CN models.

#### 4.1. Gaussian Graph Filters and their proposed

From the above analysis, one can see that models with better flexibility like Heat Kernel propagation and BernNet perform better than the others since they can easily approximate unique filters adapted to different graphs. However, as we stated above, these models are not flexible enough as the concentration center and bandwidth for bernstein basis are unchangeable and Heat Kernel based propagation always
performs as a low-pass filter. Therefore, we’d like to construct a series of bases with better concentration flexibility and combine them to approximate unique filters for different graphs.

A proper graph filter should have flexible concentration attributes which means that its concentration centre and bandwidth can change smoothly and its maximum response is bounded, then it can easily adapted to various graphs. To build such a graph filter, we use the a series of Gaussian bases as our graph filter. We propose our Gaussian Graph Convolution inspired from the close form for graph heat diffusion. The graph propagation for a single Gaussian basis concentrated at \( b \) can be formulated as follows:

\[
H_G = e^{-T(L - bI)^2}X,
\]

with \( b \in [0, 2] \) and its maximum response 1. Then for its corresponding Gaussian filter, we can obtain its bandwidth as follows:

**Proposition 4.1.** For the Gaussian filter denoted as:

\[
g_{G_{T,b}}(\lambda) = e^{-T(\lambda - b)^2},
\]

with \( b \in [0, 2] \) and \( T \) is the hyper-parameter related the bandwidth. It’s bandwidth is:

\[
BW_{G_{T,b}} = \begin{cases} 
  b, & b < \sqrt{\frac{\log \sqrt{2}}{T}}, \\
  2\sqrt{\frac{\log \sqrt{2}}{T}}, & \sqrt{\frac{\log \sqrt{2}}{T}} \leq b \leq 2 - \sqrt{\frac{\log \sqrt{2}}{T}}, \\
  2 - b, & b > 2 - \sqrt{\frac{\log \sqrt{2}}{T}}
\end{cases}
\]

The proposition can be quickly proved by calculating the \( g_{G_{T,b}}(\lambda) \leq \frac{1}{\sqrt{2}} \) with \( \lambda \in [0, 2] \). From the above results, we can demonstrate the advantages for the Gaussian filter. Like heat kernel filters, Gaussian filters decoupled the three concentration attributes. For this count, we can flexibly choose different Gaussian filters and combine them to achieve our graph filter with the desired pass-band. We can choose \( T \) for different graph propagation to different approximated filters with various bandwidths and choose \( b \) to change the concentration centre smoothly. Empirical results show that such flexibility enables our G^2CN to perform as different graph filters and show satisfying results on real-world datasets.

**Analysis via the function approximation perspective.** As the following lemma shows, all square-integrable functions can approximate \( f \) with a series of Gaussian kernels.

**Lemma 4.2.** (Calcaterra & Boldt, 2008) For any \( f \in L^2\mathbb{R} \) and any \( \epsilon > 0 \) there exists \( T > 0 \) and \( b > 0 \) and \( N \in \mathbb{N} \) and \( \theta_i \in \mathbb{R} \) such that:

\[
\left\| f(x) - \sum_{i=0}^{N} \theta_i e^{-T(x - nb)^2} \right\| \leq \epsilon.
\]

We can obtain the above conclusion based on the findings proposed in (Calcaterra & Boldt, 2008). Thereby, we can approximate the most practical filters with proper series Gaussian filter. The above results demonstrate the expressive power of our proposed Gaussian Graph Convolution.

**4.2. Formulation of the Gaussian Filter in Graph and our G^2CN**

To achieve the Gaussian filter in the graph propagation, we propose two formulations for the approximated graph propagation as listed below:

**G^2CN-Taylor** Firstly, we can obtain dissecting the Gaussian filter as graph propagation by \( K \)-order Taylor expansion:

\[
H = g_{G_{T,b}}(L)X \approx \sum_{k=0}^{K} \frac{T^k}{k!}(L - bI)^{2k}X, \tag{16}
\]

where \( K \) here denotes the propagation times, which only influences the accuracy of the approximation.

**G^2CN-Euler** Since the equivalent graph filter \( g_{G_{T,b}} \) can also be regarded as the close form solution for the following differential system at \( t = 1 \):

\[
\frac{\partial X(t)}{\partial t} = -T(L - bI)^2X(t), \tag{17}
\]

with \( X(0) = X \). Then we can use the forward Euler method to solve the graph propagation with the step size equal to \( 1/K \).

\[
X \left( \frac{i}{K} \right) = X \left( \frac{i - 1}{K} \right) - \frac{T}{K}(L - bI)^2X \left( \frac{i - 1}{K} \right),
\]

with \( i = 0, ..., K \) to obtain the final state \( H \approx X(1) \). We only need to store the state \( i/K \) after the \( i \)-th iteration with such a scheme. And since the matrix-matrix multiplication is too expansive, we iteratively use matrix-vector multiplication in practice.

Using the above methods, we can obtain the equivalent graph propagation for the graph filter concentrated at \( b \). Since one Gaussian filter can only respond to the eigenvalues around the basis, we can combine features obtained by Gaussian filters with different concentration centres and bandwidth.

**The whole structure of G^2CN.** After combining our Gaussian Graph Convolution, we can obtain the graph information filtered by the graph spectral filter we desired. As
shown in (He et al., 2021), the node classification mainly needs graph information centred on the high-frequency or low-frequency part in practice, we only adopt two Gaussian filters for our G²CN with fixed centre \( b = 0 \) and \( b = 2 \) and different \( T \) for the following equations. Since the high flexibility makes the weights training much easier, we only use a linear layer for classification after the propagation for computational efficiency. Then we finally get our G²CN.

### 4.3. Numerical Analysis on above two G²CN type

In this section, we’d like to compare to type of G²CN from the perspective of their approximated error from their closed-form solution which can be defined as:

\[
e_{T,K,b} = \|H_{T,K,b} - e^{-T(bI-L)^2}X\|,
\]

with \( M \) is “Eu” or “Ta” to denote G²CN-Euler and G²CN-Taylor and \( \| \cdot \| \) denotes the operator norm. Then the numerical error for the Taylor method is:

**Theorem 4.3.** For a single G²CN-Taylor’s graph filter with order \( K \) with width factor \( T \) and bias \( b \), the approximated error \( e_{T,K,b}^{Ta} \) compared with its closed-from \( e^{-T(bI-L)^2}X \) is:

\[
e_{T,K,b}^{Ta} \leq \frac{(T\|bI-L\|_2^2)^{K+1}}{(K+1)!} \|X\|.
\]

And the numerical error for Euler method is:

**Theorem 4.4.** For a single G²CN-Euler’s graph filter with order \( K \) with width factor \( T \) and bias \( b \), if \( 2T < K \) the approximated error \( e_{T,K,b}^{Eu} \) compared with its closed-from \( e^{-T(bI-L)^2}X \) can be bounded by:

\[
e_{T,K,b}^{Eu} \leq \frac{T^2\|bI-L\|_2^4}{K} \|X\|.
\]

From the above results, one can see that the increase of \( K \) can reduce the numerical error for the Euler method. However, when \( K < T\|bI-L\|_2^2 \) the error for the Taylor expansion will increase first as \( K \) increases and then decrease when \( K \) is large enough and will return a smaller error than the Euler method when \( K \) is large enough. Therefore, we recommend the G²CN Euler when \( K \) is small due to the limited computation complexity. And as the results demonstrate, we only recommend the G²CN-Taylor when \( K \) is significantly large compared with \( T \), G²CN-Taylor will return a better approximation in this scenario.

## 5. Experiments

### 5.1. Homophily and Heterophily Datasets

The graph node classification datasets can be divided into homophily graphs and heterophily graphs with the homophily principle (McPherson et al., 2001). Homophily principle states that contact between similar nodes occurs at a higher rate than among dis-similar nodes. In recent work, Pei et al. (2020) designs an index denoted by \( H \) to measure the homophily in a graph:

\[
H(G) = \frac{1}{|V|} \sum_{v \in V} \frac{\text{Number of neighbors of } v \in V \text{ that have the same label as } v}{\text{Number of neighbors of } v}.
\]

Lower \( H \) indicates a strong heterophilic graph, where nodes with distinct labels are more likely to link together. As stated, the homophilic graphs need low-pass filters for a satisfactory performance while heterophilic graphs need other passbands in graph filters for generalization.

In our work, we conduct experiments on five widely used benchmark homophilic graphs used in (Pei et al., 2020), including Wikipedia graphs Chameleon and Squirrel (Rozemberczki et al., 2021), the Actor co-occurrence graph and the WebKB graphs Texas and Cornell (Pei et al., 2020). Experiments on heterophilic graphs demonstrate the superiority of our G²CN models compared with other linear or non-linear graph models due to the flexibility of the Gaussian Graph Filters we used. More rigorously, we also conduct the comparison on a large scale node classification dataset, OGB-Arxiv (?). An overview summary of characteristics of the datasets is given in Table 1.

### 5.2. Experimental Settings

Firstly, we experiment on the semi-supervised node classification task for standard citation networks with the standard data split as in (Wang et al., 2021; Kipf & Welling, 2017; Velickovic et al., 2018; Xu et al., 2019a;b; Poli et al., 2019) with the same setting as DGC. Furthermore, we also evaluate the performance of the homophilic Amazon co-purchase graph Computers and Photo for fully supervised node classification tasks with the same setting as BernNet, who randomly split the node sets into train/validation/test set ratio 60%/20%/20%.

Then for the heterophilic graphs, we use the same setting as BernNet and finish the experiments on the fully-supervised node classification tasks for these graphs. We compare G²CN with three baseline models: GCN (Kipf & Welling, 2017) , GAT (Velickovic et al., 2018), APPNP (Klicpera et al., 2019a), BernNet (He et al., 2021), SGC (Wu et al., 2019), DGC (Wang et al., 2021). We directly tuned the
Graph Gaussian Convolution Networks with Concentrated Graph Filters

Table 1. Datasets statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
<th>Computers</th>
<th>Photo</th>
<th>Chameleon</th>
<th>Squirrel</th>
<th>Actor</th>
<th>Texas</th>
<th>Cornell</th>
<th>OGB-Arxiv</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Nodes</td>
<td>2708</td>
<td>3327</td>
<td>19717</td>
<td>13752</td>
<td>7650</td>
<td>2277</td>
<td>5201</td>
<td>7600</td>
<td>183</td>
<td>183</td>
<td>169343</td>
</tr>
<tr>
<td>#Edges</td>
<td>5278</td>
<td>4552</td>
<td>44324</td>
<td>245861</td>
<td>119081</td>
<td>36101</td>
<td>217073</td>
<td>26659</td>
<td>295</td>
<td>309</td>
<td>1166243</td>
</tr>
<tr>
<td>#Features</td>
<td>1433</td>
<td>3703</td>
<td>500</td>
<td>767</td>
<td>745</td>
<td>2325</td>
<td>2089</td>
<td>932</td>
<td>1703</td>
<td>1703</td>
<td>128</td>
</tr>
<tr>
<td>#Classes</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>H</td>
<td>0.83</td>
<td>0.71</td>
<td>0.79</td>
<td>0.47</td>
<td>0.59</td>
<td>0.25</td>
<td>0.22</td>
<td>0.24</td>
<td>0.06</td>
<td>0.11</td>
<td></td>
</tr>
</tbody>
</table>

hyper-parameters for non-linear models follows the same steps as He et al. (2021) stated and the channel numbers hidden layers for non-linear models are 64. As for linear GCNs, we performed a hyper-parameter search for linear models’ hyper-parameter (T and weight decay) on the validation set as used in DGC for 1000 trials. We choose $K = 200$ for DGC and $K = 100$ for our $G^2$CN to ensure the same propagation steps with graph Laplacian for fairness.

We use Adam optimizer (Kingma & Ba, 2014) for training with 100 epochs for semi-supervised tasks and 1000 epochs for fully-supervised tasks as BernNet for fair with lr=0.5 for our model and report results averaged over ten random runs.

5.3. Experimental Results on Homophily Datasets.

The test results for homophily datasets are shown in Table 2. The results show that our $G^2$CN can show comparable performance on the semi-supervised citation graphs. Furthermore, one can see that our $G^2$CN enjoys an evident advantage on the fully-supervised graph tasks for large graphs. The empirical results demonstrate that our $G^2$CN can return better graph signals output by our graph filters on homophilic graphs, which mainly depend on the graph’s low-frequent composition.

We notice that BernNet perform worse than our $G^2$CN on the homophilic graphs. The proper passband for these graphs shown in Table 3 can explain such phenomenon. The table shows that The bandwidth for each bernstein basis in BernNet is not flexible. For example, the pass band for the last three filters of a BernNet ($K = 10$) is $[1.3, 1.92]$, $[1.61, 1.92]$, $[1.93, 2.0]$. Due to such reason, BernNet needs to rely on complicated weighted parameter to approximate the narrow high-pass filter stated in $G^2$CN and it will make the learning task much harder. Thereby, it cannot utilize proper high-frequency graph information as ours which leads to worse results than ours.

5.4. Experimental Results on Heterophily Datasets

Apart from the homophilic graphs, we also conduct experiments on the heterophilic graphs. The results are shown in Table 4. Compared with the homophilic graphs, the advantages for our $G^2$CN over the low-pass models is more evident since the heterophilic graphs need the graph signals with different frequency for their tasks. Compared with BernNet, our $G^2$CN can still show better performance on these models except Actor. The above results can also demonstrate the superiority of our proposed model.

Understandings via our Concentration Analysis From the experimental results, one can see that our $G^2$CN or BernNet shows significant performance on Chameleon and

Table 2. Test accuracy (%) comparison on common used citation graphs with semi-supervised scheme (above) and Amazon co-purchase graphs with fully-supervised scheme (blow). The results are averaged over 10 runs.

<table>
<thead>
<tr>
<th>Type</th>
<th>Dataset</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonlinear</td>
<td>GCN</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>83.0</td>
<td>72.5</td>
<td>79.0</td>
</tr>
<tr>
<td></td>
<td>APPNP</td>
<td>83.3</td>
<td>71.8</td>
<td>80.1</td>
</tr>
<tr>
<td>Linear</td>
<td>SGC</td>
<td>81.1</td>
<td>71.9</td>
<td>78.9</td>
</tr>
<tr>
<td></td>
<td>DGC</td>
<td><strong>83.3</strong></td>
<td>73.3</td>
<td>80.3</td>
</tr>
<tr>
<td></td>
<td>$G^2$CN</td>
<td>82.7</td>
<td><strong>73.8</strong></td>
<td><strong>80.4</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Dataset</th>
<th>Computers</th>
<th>Photo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nonlinear</td>
<td>GCN</td>
<td>83.32 ± 0.33</td>
<td>88.26 ± 0.73</td>
</tr>
<tr>
<td></td>
<td>GAT</td>
<td>83.32 ± 0.39</td>
<td>90.94 ± 0.68</td>
</tr>
<tr>
<td></td>
<td>APPNP</td>
<td>85.32 ± 0.32</td>
<td>88.51 ± 0.31</td>
</tr>
<tr>
<td></td>
<td>BernNet</td>
<td>87.64 ± 0.44</td>
<td>94.50 ± 0.40</td>
</tr>
<tr>
<td>Linear</td>
<td>SGC</td>
<td>84.74 ± 0.27</td>
<td>86.48 ± 0.41</td>
</tr>
<tr>
<td></td>
<td>DGC</td>
<td>89.54 ± 0.64</td>
<td>94.89 ± 0.50</td>
</tr>
<tr>
<td></td>
<td>$G^2$CN</td>
<td><strong>91.46 ± 0.35</strong></td>
<td><strong>95.29 ± 0.48</strong></td>
</tr>
</tbody>
</table>

Table 3. Bandwidth for DGC and $G^2$CN (Bandwidth for two Gaussian filters with $b = 0, 2$) Computers and Photo.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>DGC</th>
<th>$G^2$CN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computers</td>
<td>0.21 (0.35, 0.15)</td>
<td>0.23 (0.37, 0.12)</td>
</tr>
<tr>
<td>Photo</td>
<td>0.23 (0.37, 0.12)</td>
<td>0.23 (0.37, 0.12)</td>
</tr>
</tbody>
</table>
Apart from the above datasets, we also finish experiments on the large scale dataset. The results on the large graph OGB-Arxiv demonstrate the effectiveness of our $G^2CN$. Table 5 shows that the Texas, Actor, and Cornell mainly lack high-frequent graph signals. As for BernNet, as illustrated in the former, BernNet needs complicated weight approximation to approximate graph filters, making its training procedure much harder. Empirical results demonstrate the effectiveness of our $G^2CN$ and the rationality of our concentration spectral analysis since we can use it to understand different models’ performance on various datasets.

### 5.5. Performance on Large Scale Datasets

Apart from the above datasets, we also finish experiments for our $G^2CN$ on OGB-Arxiv. Since the OGB-Arxiv dataset is large and hard, we replace the linear layer with 2-layer MLP after the output of our Gaussian graph filters. The results are listed in Table 6. From the table, one can see that our $G^2CN+MLP$ can outperform the DGC+MLP and SGC on the large scale dataset. The results on the large graph OGB-Arxiv demonstrate the effectiveness of our $G^2CN$.

### 5.6. Computation Time for Different Models

We list the computation time for different models trained on large graphs for 1000 epochs in Table 7. The models are evaluated on a single RTX-3070 and Intel I5-10400 (2.90GHz). As a linear GCN, DGC and $G^2CN$ is much faster than non-linear GCNs like GCN and BernNet. Since our $G^2CN$ needs an additional weighted summation for the outputs of two filters, our model is slightly slower than DGC.

Table 4. Test accuracy (%) comparison on heterophily datasets. Reported results are averaged over 10 runs.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>GCN</th>
<th>GAT</th>
<th>APPNP</th>
<th>FAGCN</th>
<th>ARMA</th>
<th>DGC</th>
<th>BernNet</th>
<th>$G^2CN$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chameleon</td>
<td>42.98 ± 1.41</td>
<td>47.62 ± 0.83</td>
<td>43.07 ± 1.55</td>
<td>63.3 ± 1.41</td>
<td>55.1 ± 1.32</td>
<td>67.08 ± 1.58</td>
<td>54.11 ± 1.73</td>
<td>73.61 ± 0.89</td>
</tr>
<tr>
<td>Squirrel</td>
<td>28.41 ± 0.57</td>
<td>27.33 ± 0.81</td>
<td>31.71 ± 0.47</td>
<td>39.7 ± 0.67</td>
<td>35.5 ± 0.81</td>
<td>50.51 ± 0.81</td>
<td>41.35 ± 0.81</td>
<td>66.91 ± 1.13</td>
</tr>
<tr>
<td>Actor</td>
<td>33.23 ± 1.16</td>
<td>33.93 ± 2.47</td>
<td>39.66 ± 0.55</td>
<td>40.11 ± 0.77</td>
<td>40.79 ± 0.89</td>
<td>40.35 ± 0.73</td>
<td>41.79 ± 1.01</td>
<td>41.44 ± 0.76</td>
</tr>
<tr>
<td>Texas</td>
<td>77.38 ± 3.28</td>
<td>80.82 ± 2.13</td>
<td>90.98 ± 1.64</td>
<td>96.5 ± 0.47</td>
<td>92.3 ± 0.66</td>
<td>94.74 ± 0.33</td>
<td>96.22 ± 0.79</td>
<td>96.72 ± 0.73</td>
</tr>
<tr>
<td>Cornell</td>
<td>65.90 ± 4.43</td>
<td>78.21 ± 2.95</td>
<td>91.81 ± 1.96</td>
<td>93.3 ± 1.21</td>
<td>93.4 ± 1.13</td>
<td>92.45 ± 1.21</td>
<td>92.29 ± 2.74</td>
<td>94.11 ± 1.81</td>
</tr>
</tbody>
</table>

Table 5. Bandwidth for DGC and $G^2CN$ (Bandwidth for two Gaussian filters with $b = 0, 2$) for heterophilic Squirrel and Cornell.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DGC</th>
<th>$G^2CN$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas</td>
<td>2</td>
<td>(1.13, 0.16)</td>
</tr>
<tr>
<td>Actor</td>
<td>2</td>
<td>(2.0, 0.27)</td>
</tr>
<tr>
<td>Cornell</td>
<td>2</td>
<td>(2.0, 0.08)</td>
</tr>
<tr>
<td>Squirrel</td>
<td>0.09</td>
<td>(0.12, 0.37)</td>
</tr>
<tr>
<td>Chameleon</td>
<td>0.07</td>
<td>(0.11, 0.27)</td>
</tr>
</tbody>
</table>

The table shows that the Texas, Actor, and Cornell mainly need to apply an all-pass filter. For this count, APPNP and DGC can also perform better than GCN and GAT since the bandwidth for APPNP and DGC is more flexible than GCN and GAT. As for Chameleon and Squirrel datasets, the best bandwidth adapted for DGC and $G^2CN$ demonstrates that they need a comb graph filter with narrow pass-bands. Therefore, graph models cannot perform well since they lack high-frequent graph signals. As for BernNet, as illustrated in the former, BernNet needs complicated weight approximation to approximate graph filters, making its training procedure much harder. Empirical results demonstrate the effectiveness of our $G^2CN$ and the rationality of our concentration spectral analysis since we can use it to understand different models’ performance on various datasets.

### 5.7. Ablation Studies on Spectral Response Curve For Selected Dataset

In this section, we draw the approximated spectral response curve of our $G^2CN$’s Gaussian graph filter for selected datasets as shown in Figure 1. The curve depicts whether our Gaussian graph filter amplifies or diminishes the signals corresponding to eigenvalue $\lambda$. From the above figures, one
can see that same as our former analysis. Pubmed needs a low-pass filter while Computers and Squirrel needs comb filters with narrow bands and filters. And Squirrel mostly rely on its graph signals with high frequency for classification which is the same as Chameleon listed in the Appendix. Since our $G^2CN$ can easily approximate such filter, our outperforms other datasets with significant advantages. Cornell is more likely to be an all-pass filter. We left the curves for other graphs in the Appendix F.

![Graphs](image)

(a) Pubmed  
(b) Computers  
(c) Squirrel  
(d) Cornell

*Figure 1. $G^2CN$'s equivalent graph filters for selected real-world datasets.*

### 6. Conclusions

In this work, we first proposed three attributes called concentration attributes to analyze the spectral properties for different graph models called concentration spectral analysis. Our analysis can explain why various graph models will show different results even if their global tendency is the same. Furthermore, we analyzed the graph filters’ weaknesses and advantages for different models. We notice that these models’ concentration attributes are not flexible, making it hard to apply proper graph filters for different graphs. We then proposed our Gaussian graph filters and our $G^2CN$ using this filter. The concentration attributes for our $G^2CN$ is much more flexible and can easily apply proper filters for different graphs. Empirical results also demonstrate the superiority of our model.

### ACKNOWLEDGMENTS

Zhouchen Lin is supported by the NSF China (No. 61731018), NSFC Tianyuan Fund for Mathematics (No. 12026606), and Project 2020BD006 supported by PKU-Baidu Fund.

Yisen Wang is partially supported by the NSF China (No. 62006153), Project 2020BD006 supported by PKU-Baidu Fund, and Open Research Projects of Zhejiang Lab (No. 2022RC0AB05).
References


A. Proof for Theorem 4.3

The proof for Theorem 4.3 is as follows:

Proof. From the definition, the approximated error for $G^2$CN-Taylor is:

$$e_{T,K,b}^{T,K,b} = \left\| T_a H_a e^{T(aL)^2} X \right\|_2$$

$$= \left\| \sum_{i=0}^{K} \frac{(-T(bI-L)^2)^i}{i!} X - \sum_{i=0}^{\infty} \frac{(-T(bI-L)^2)^i}{i!} X \right\|_2$$

$$= \left\| \frac{(-T(bI-L)^2)^{K+1}}{(K+1)!} \sum_{i=0}^{\infty} \frac{(-T(bI-L)^2)^i}{i!} X \right\|_2$$

Then Theorem 4.3 is proved.

B. Proof for Theorem 4.4

The proof for Theorem 4.4 is as follows:

Proof. The Euler Scheme is to solve the following differential equation at $t = 1$:

$$\begin{cases} \frac{dX}{dt} = -(bI-L)^2 X, \\ X_0 = X, \end{cases}$$

with step size $1/K$ for $K$ steps. Then consider a general Euler forward scheme for our problem:

$$\hat{X}(k+1) = X^{(k)} - hL_{b,T}^2 \hat{X}(k), \quad k = 0, \ldots, K - 1, \quad X^{(0)} = X,$$

with $\hat{X}^{(k)}$ denotes approximated $X(k/K)$ by the forward Euler, $h = \frac{1}{K}$ and $L_{b,T} = \sqrt{T}(bI-L)$. Then we denote the error at step $k$ as:

$$e_k = X^{(k)} - \hat{X}^{(k)},$$

then the approximated error is $e_{T,K,b}^{T,K,b} = e_K$, and the truncation error of the Euler forward finite difference (Eqn (31)) at step $k$ as:

$$T^{(k)} = \frac{X^{(k+1)} - X^{(k)}}{h} + L_{b,T}^2 X^{(k)},$$

which can be reformulated as follows:

$$X^{(k+1)} = X^{(k)} + h(T^{(k)} - L_{b,T} X^{(k)}).$$

Then subtract Eqn 34 by Eqn 31, we can obtain:

$$e^{(k+1)} = (I - hL_{b,T}^2)e^{(k)} + hT^{(k)},
whose norm can be upper bounded as:

$$\left\| e^{(k+1)} \right\|_2 \leq \left\| I - hL^2 \right\|_2 \left\| e^{(k)} \right\|_2 + h \left\| T^{(k)} \right\|.$$  (36)

Since $L_{b,T} = (bI - L)$ with $b \in [0, 2]$ and the eigenvalue of the symmetric matrix $L$ also lies in $[0, 2]$, then the eigenvalue of $L_{b,T}^2$ lies in $[0, 4]$ and eigenvalue of $hL_{b,T}^2$ lies in $[0, 2h]$ since $h = T/K < 0.5$. Therefore, the spectral norm of $I - hL_{b,T}^2$ is upper bounded by 1. Then with $M = \max_{0 \leq k \leq K-1} \left\| T^{(k)} \right\|_2$ be the upper bound on the truncation error, we have:

$$\left\| e^{(k+1)} \right\|_2 \leq \left\| e^{(k)} \right\|_2 + hM.$$  (37)

Since $e^{(0)} = X^{(0)} - \hat{X}^{(0)} = 0$, we have the following formula by induction:

$$\left\| e^{(k)} \right\|_2 \leq KhM = M.$$  (38)

Then since $\frac{dX^{(k)}}{dt} = -LX^{(k)}$ and applying Taylor’s theorem, there exists $\delta \in [kh, (k+1)h]$ such that the truncation error $T^{(k)}$ in Eqn 33 follows:

$$T^{(k)} = \frac{h}{2} L^4_{b,T} X_\delta.$$  (39)

Furthermore, since

$$\left\| X_\delta \right\|_2 = \left\| e^{-T3L^4_{b,T}} X_0 \right\|_2 \leq \left\| X \right\|_2.$$  (40)

Thereby, we can bound the truncated error:

$$\left\| T^{(k)} \right\|_2 = \frac{h}{2} \left\| L_0 \right\|^4 \left\| X_\delta \right\|_2 \leq \frac{h}{2} \left\| L_0 \right\|^4 \left\| X \right\|_2,$$  (41)

Finally, we have

$$\left\| e_{K,b,T}^{(K)} \right\|_2 = \left\| e^{(K)} \right\|_2 \leq M \leq \frac{T^2}{2K} \left\| hI - L \right\|^4 \left\| X \right\|_2.$$  (42)

Then we complete the proof.

**C. Bandwidth and Concentration Centers for Graph filters approximated by Polynomial Bases.**

We list the bandwidth of each Bernstein basis for BernNet with $K = 10$ listed in Table 8.

<table>
<thead>
<tr>
<th>Index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concentration Center</td>
<td>0.0</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.8</td>
<td>1.0</td>
<td>1.2</td>
<td>1.4</td>
<td>1.6</td>
<td>1.8</td>
<td>2.0</td>
</tr>
<tr>
<td>PassBand</td>
<td>[0.0, 0.069]</td>
<td>[0.070, 0.209]</td>
<td>[0.219, 0.455]</td>
<td>[0.481, 0.655]</td>
<td>[0.74, 1.258]</td>
<td>[0.938, 1.445]</td>
<td>[1.145, 1.619]</td>
<td>[1.365, 1.781]</td>
<td>[1.610, 1.921]</td>
<td>[1.931, 2.0]</td>
<td></td>
</tr>
<tr>
<td>Maximum Response</td>
<td>1</td>
<td>0.387</td>
<td>0.301</td>
<td>0.267</td>
<td>0.251</td>
<td>0.246</td>
<td>0.251</td>
<td>0.267</td>
<td>0.301</td>
<td>0.387</td>
<td>1</td>
</tr>
</tbody>
</table>

**D. Ablation Study on the Propagation Stpes $K$.**

We also finish the experiments on Photo with different propagation times $K$ for our $G^2$CN Euler to explore $K$’s influence. The results are shown in Figure 2. Same to our analysis stated in Theorem 4.3, as $K$ increases, the approximated error will decrease and the performance will increase and finally converge.

**E. Summary for our concentration analysis on different graph propagation.**

We summarize the concentration attributes for different graph propagation as follows:

**F. Response Curve for Other Graphs**

We list the response curve for the rest Graphs as follows:
Figure 2. Test Accuracy with respect to different propagation times for our $G^2$CN on Photo.

Table 9. Concentration attributes for different graph propagation.

<table>
<thead>
<tr>
<th>Graph Propagation Kernel (Or Basis)</th>
<th>$\mathcal{R}$</th>
<th>$b$</th>
<th>$BW$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN $(2I - L)^K$</td>
<td>$2^K$</td>
<td>0</td>
<td>$2 - 2^1 - \frac{2}{\pi}$</td>
</tr>
<tr>
<td>PPR $(I - (1 - \alpha)(1 - L))^{-1}$</td>
<td>$\frac{1}{\alpha}$</td>
<td>0</td>
<td>$\frac{(\sqrt{2}-1)\alpha}{1 -</td>
</tr>
<tr>
<td>ARMA $\frac{1}{1 - a_K\mu}b_K$</td>
<td>$\frac{b_K}{1 -</td>
<td>a_K</td>
<td>}$</td>
</tr>
<tr>
<td>FAGCN $(1 - \lambda + \epsilon)^2$</td>
<td>$(1 + \epsilon)^2$</td>
<td>0 or 2</td>
<td>$(2^{1/4} - 1)(1 + \epsilon) \in [0.19, 0.38]$</td>
</tr>
<tr>
<td>Heat Kernel $e^{-TL}$</td>
<td>1</td>
<td>0</td>
<td>$\frac{\log(\sqrt{2})}{\mu}$</td>
</tr>
<tr>
<td>ChebNet $C^{(0)} = I$</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>$C^{(1)} = 2L/\lambda_{max} - I$</td>
<td>1</td>
<td>1, -1</td>
<td>$2 - \sqrt{2}$</td>
</tr>
<tr>
<td>$C^{(k)} = 2C^{(2)}C^{(s-1)} - C^{(s-2)}$</td>
<td>....</td>
<td>....</td>
<td>....</td>
</tr>
<tr>
<td>BernNet $\frac{1}{2\pi} \binom{K}{k} (2I - L)^{K-k}L^k$</td>
<td>Appendix C</td>
<td>Appendix C</td>
<td>Appendix C</td>
</tr>
<tr>
<td>Graph Gaussian $e^{-T(b-L)^2}$</td>
<td>1</td>
<td>$b$</td>
<td>Proposition 4.1</td>
</tr>
</tbody>
</table>
Figure 3. $\text{G}^2\text{CN}$’s equivalent graph filters for selected real-world datasets.