中國人民大學
RENMIN UNIVERSITY OF CHINA

高瓴人工智能学院
Gaoling School of Artificial Intelligence

Graph Convolutional Networks: Theory and Fundamentals

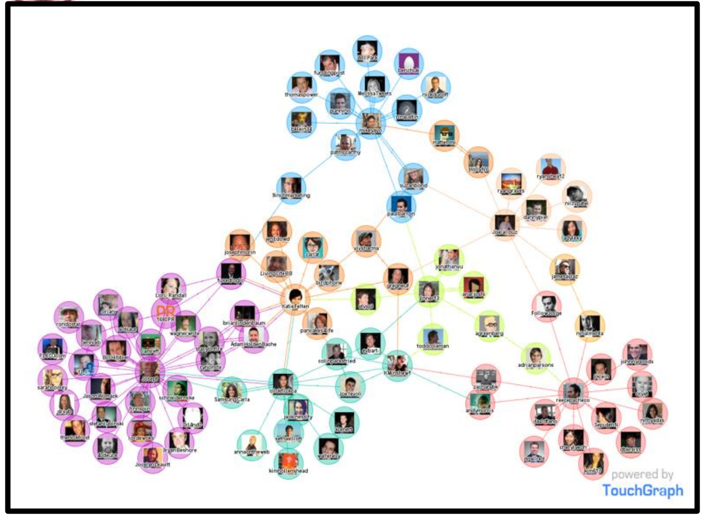
Zhewei Wei

2024-08-06

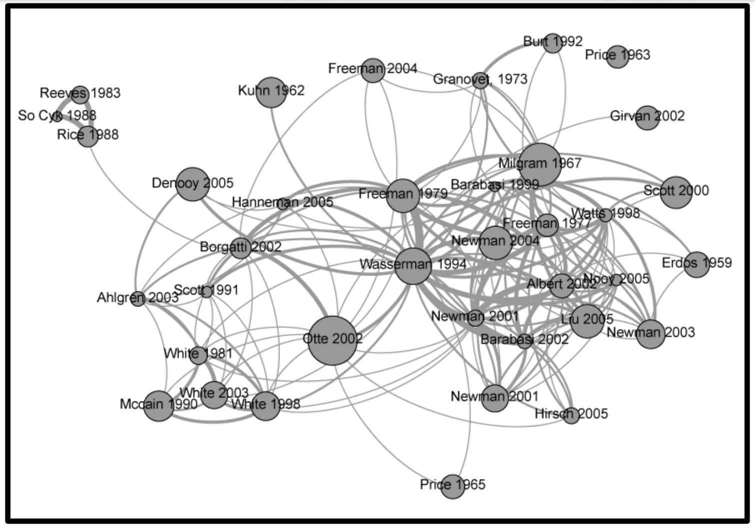
Outline

- Overview of GNNs
- Spectral interpretation of GNNs
- Our works (OptBasisGNN, PolyGCL, PSHGCN)
- Summary & Perspectives

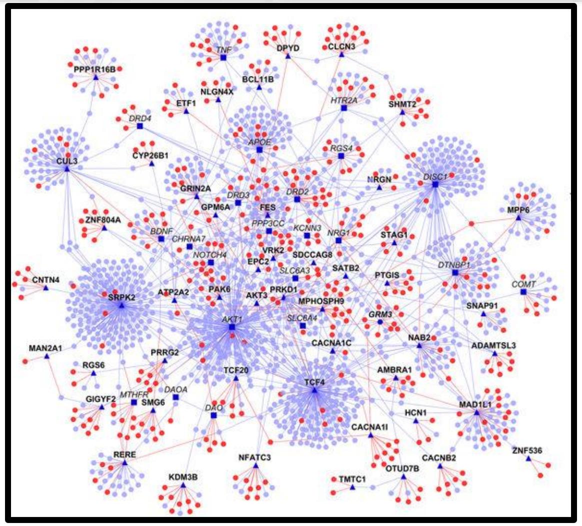
Graphs are ubiquitous



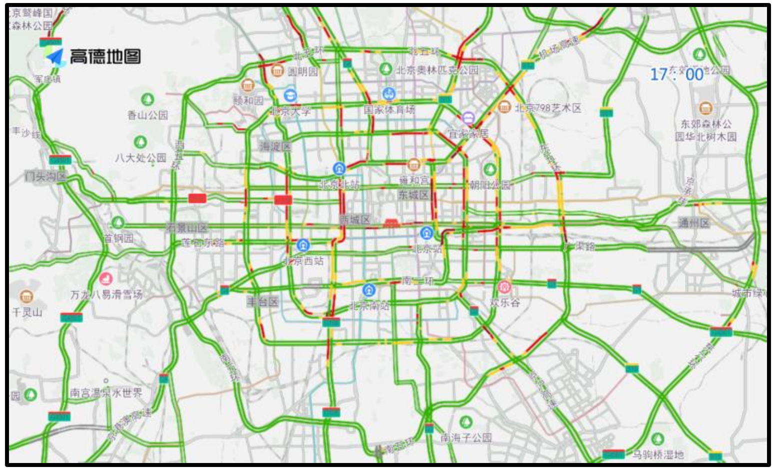
Social Network



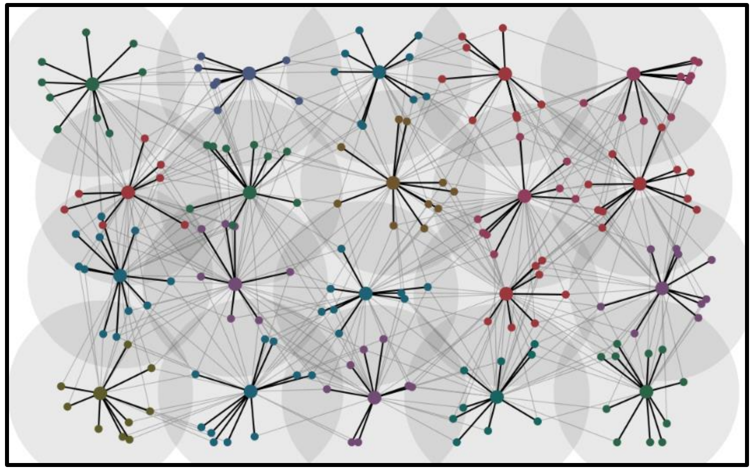
Citation Network



Protein Network



Road Network



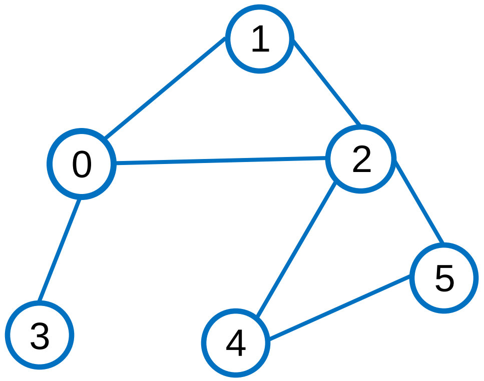
Signal Network

Graph

- $G = (V, E)$

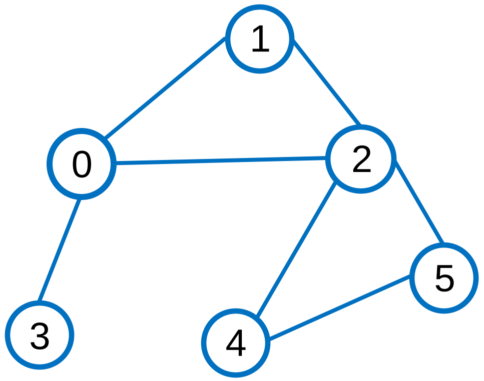
Adjacency matrix $\boldsymbol{A} =$

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$



Graph

- $G = (V, E)$

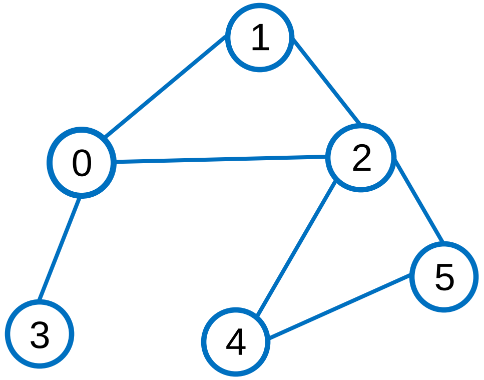


Degree matrix $D = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$

Graph

$$P = D^{-1/2} A D^{-1/2}$$

- $G = (V, E)$



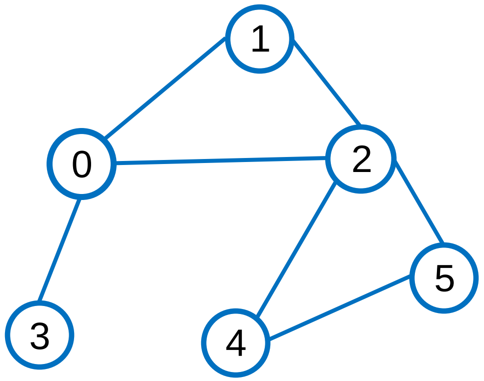
Normalized adjacency matrix

$$\boldsymbol{P} = \begin{bmatrix} 0 & \frac{1}{\sqrt{3}\cdot\sqrt{2}} & \frac{1}{\sqrt{3}\cdot\sqrt{4}} & \frac{1}{\sqrt{3}\cdot\sqrt{1}} & 0 & 0 \\ \frac{1}{\sqrt{3}\cdot\sqrt{2}} & 0 & \frac{1}{\sqrt{2}\cdot\sqrt{4}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{3}\cdot\sqrt{4}} & \frac{1}{\sqrt{2}\cdot\sqrt{4}} & 0 & 0 & \frac{1}{\sqrt{4}\cdot\sqrt{2}} & \frac{1}{\sqrt{4}\cdot\sqrt{2}} \\ \frac{1}{\sqrt{3}\cdot\sqrt{1}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{4}\cdot\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}\cdot\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{4}\cdot\sqrt{2}} & 0 & \frac{1}{\sqrt{2}\cdot\sqrt{2}} & 0 \end{bmatrix}$$

Graph

$$L = I - D^{-1/2} A D^{-1/2}$$

- $G = (V, E)$

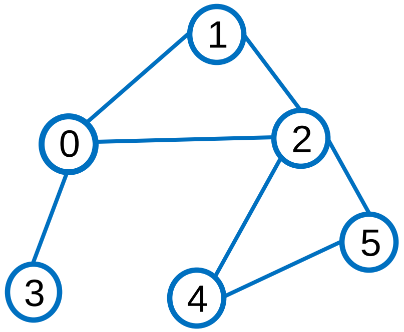


Normalized Laplacian matrix $L =$

$$\begin{bmatrix} 1 & \frac{-1}{\sqrt{3}\cdot\sqrt{2}} & \frac{-1}{\sqrt{3}\cdot\sqrt{4}} & \frac{-1}{\sqrt{3}\cdot\sqrt{1}} & 0 & 0 \\ \frac{-1}{\sqrt{3}\cdot\sqrt{2}} & 1 & \frac{-1}{\sqrt{2}\cdot\sqrt{4}} & 0 & 0 & 0 \\ \frac{-1}{\sqrt{3}\cdot\sqrt{4}} & \frac{-1}{\sqrt{2}\cdot\sqrt{4}} & 1 & 0 & \frac{-1}{\sqrt{4}\cdot\sqrt{2}} & \frac{-1}{\sqrt{4}\cdot\sqrt{2}} \\ \frac{-1}{\sqrt{3}\cdot\sqrt{1}} & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-1}{\sqrt{4}\cdot\sqrt{2}} & 0 & 1 & \frac{-1}{\sqrt{2}\cdot\sqrt{2}} \\ 0 & 0 & \frac{-1}{\sqrt{4}\cdot\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}\cdot\sqrt{2}} & 1 \end{bmatrix}$$

Graph

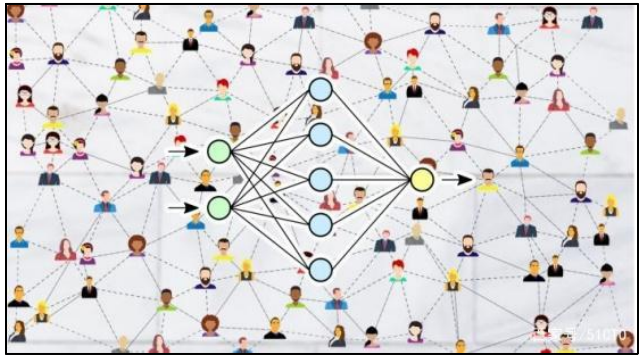
- Node feature matrix $\boldsymbol{X} \in \mathcal{R}^{n \times f}$, f denotes the dimension



Node	Features1	Features2	Features3	Features4	Features5	Features6
x_0	1	0	0	0	0	0
x_1	0	1	0	0	0	0
x_2	0	0	1	0	0	0
x_3	0	0	0	1	0	0
x_4	0	0	0	0	1	0
x_5	0	0	0	0	0	1

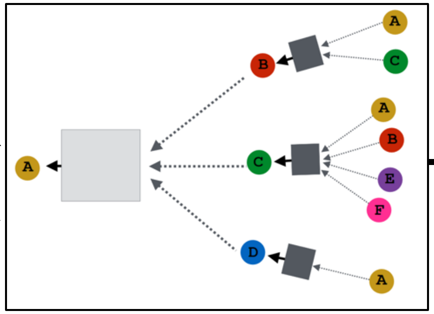
Graph Neural Network

Graph Structure

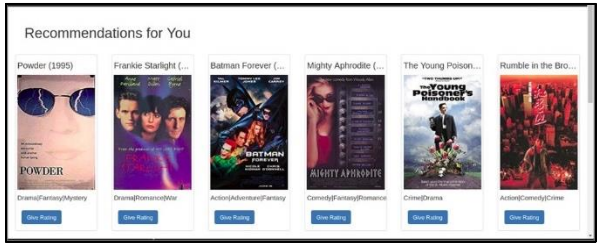


Node Features

Movies	Reviews Given	Rating
Mission Imposible	✓	Good
James Bond	✓	Good
Toy Story	✓	Bad



GNNs

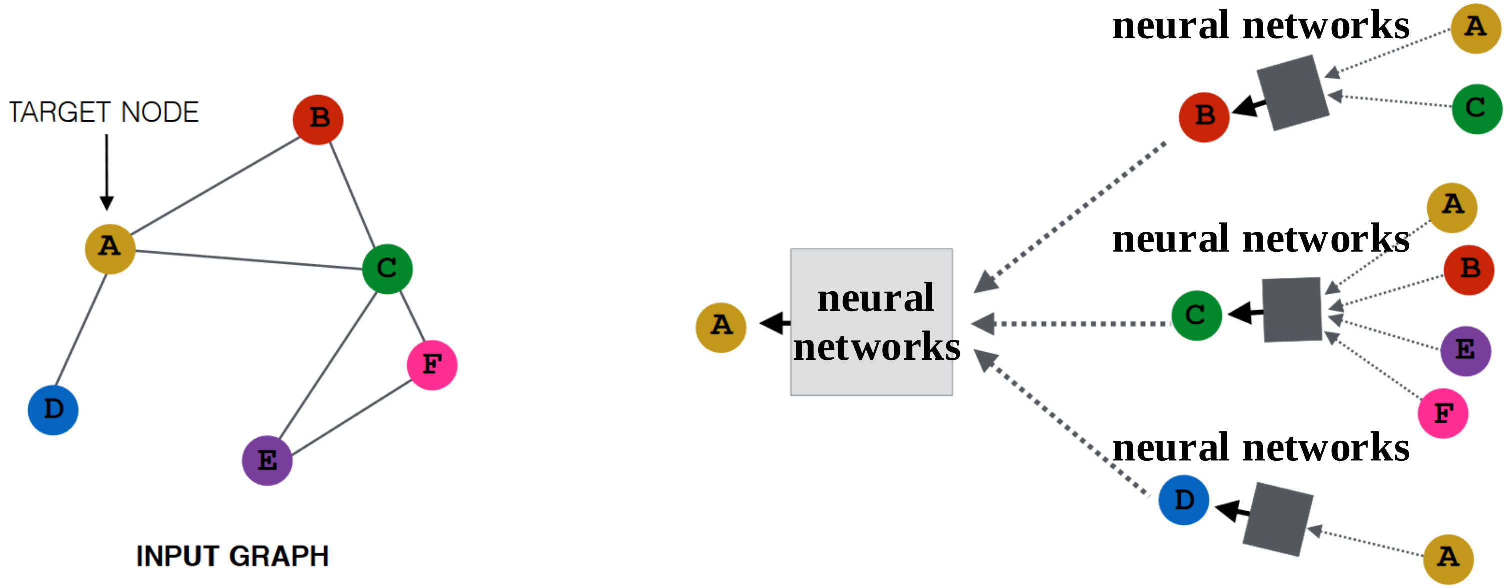


Prediction, Recommendation, Classification…

Graph Neural Network

- Graph Convolution Neural Network(GCN) [Kipf et al.,2017]
  - Aggregating the neighbors' node features,
  - Training the weights with Message-Passing Scheme
  - Architecture:

$$H^{(\ell+1)} = \sigma\left(\widetilde{P} H^{(\ell)} W^{(\ell)}\right)$$



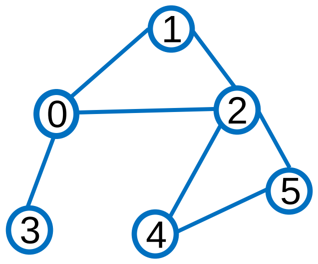
Message passing scheme

$$H^{(\ell+1)} = \sigma\left(\boxed{\tilde{P}} \cdot \boxed{H^{(\ell)}} \cdot \boxed{W^{(\ell)}}\right)$$

$\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}$

representation of previous layer

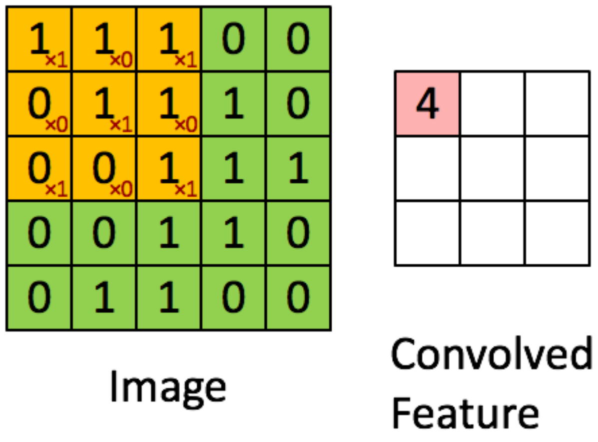
weight matrix

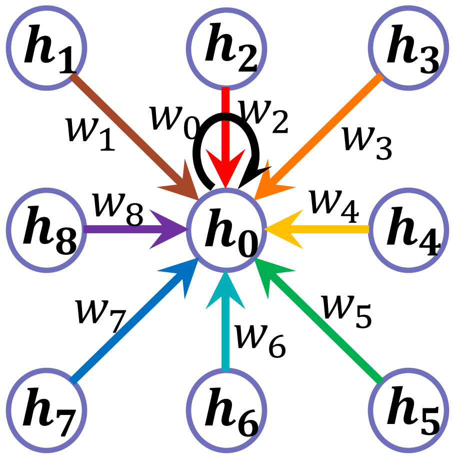


Node	Features1	Features2	Features3	Features4	Features5	Features6
x_0	1	0	0	0	0	0
x_1	0	1	0	0	0	0
x_2	0	0	1	0	0	0
x_3	0	0	0	1	0	0
x_4	0	0	0	0	1	0
x_5	0	0	0	0	0	1
Sum	**0**	**1**	**1**	**1**	**0**	**0**
Self-loop	**1**	**1**	**1**	**1**	**0**	**0**
Symmetry	**1/**$\sqrt{4}\sqrt{4}$	**1/** $\sqrt{4}\sqrt{3}$	**1/** $\sqrt{4}\sqrt{5}$	**1/** $\sqrt{4}\sqrt{2}$	**0**	**0**

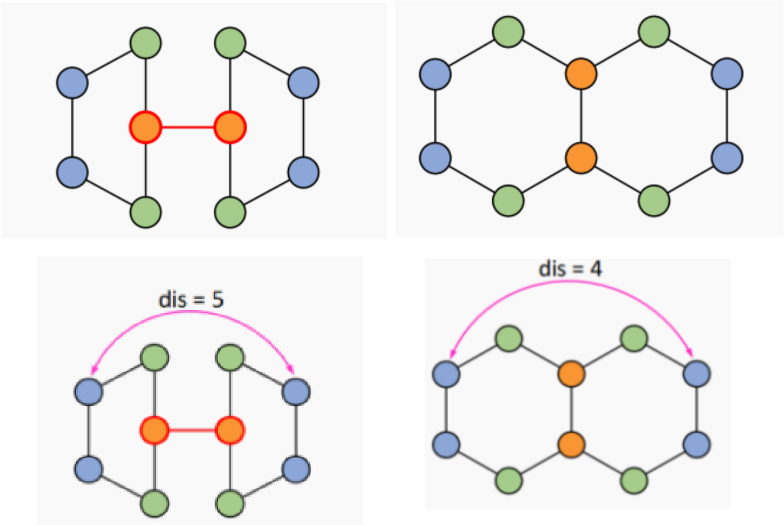
GCN and CNN

- CNN is also a (Message-Passing) GNN
  - Aggregating the eight neighbors‘ and its own features



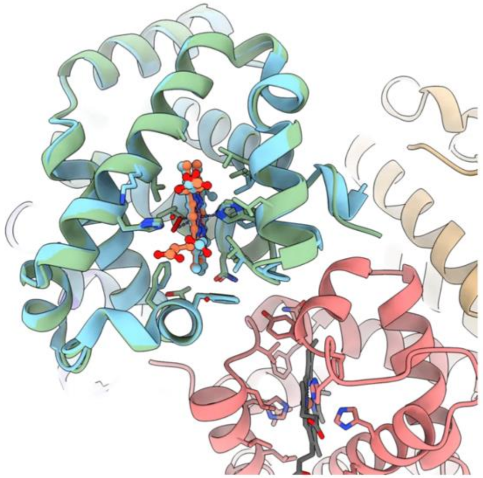


Applications of graph machine learning



GNN+Graph Algorithm

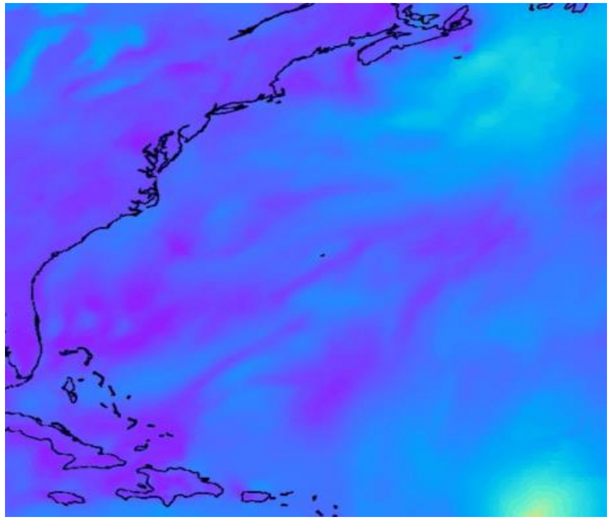
GNN can be used for classic graph algorithms, such as the graph biconnectivity problem.
[ICLR’2023 Best Paper]



实验结构（蓝色）与预测结构（绿色橙色）的对比

GNN+Protein Analysis

DeepMind released its third-generation protein analysis AI tool AlphaFold3 in Nature.
[Nature’2024]



GNN+Weather Forecasting

GraphCast, a weather model developed by DeepMind, has been published in Science.
[Science’2023].

Outline

- Overview of GNNs
- Spectral interpretation of GNNs
- Our works (OptBasisGNN, PolyGCL, PSHGCN)
- Summary & Perspectives

GCN and Graph Signal Processing

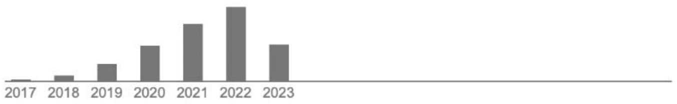
Semi-Supervised Classification with Graph Convolutional Networks

Thomas N. Kipf
University of Amsterdam
T.N.Kipf@uva.nl

Max Welling
University of Amsterdam
Canadian Institute for Advanced Research (CIFAR)
M.Welling@uva.nl

Abstract

We present a scalable approach for semi-supervised learning on graph-structured data that is based on an efficient variant of convolutional neural networks which operate directly on graphs. We motivate the choice of our convolutional architecture via a localized first-order approximation of spectral graph convolutions. Our model scales linearly in the number of graph edges and learns hidden layer representations that encode both local graph structure and features of nodes. In a number of experiments on citation networks and on a knowledge graph dataset we demonstrate that our approach outperforms related methods by a significant margin.



2 Fast Approximate Convolutions on Graphs

In this section, we provide theoretical motivation for a specific graph-based neural network model $f(X, A)$ that we will use in the rest of this paper. We consider a multi-layer Graph Convolutional Network (GCN) with the following layer-wise propagation rule:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) . \tag{2}$$

Here, $\tilde{A} = A + I_N$ is the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connections. I_N is the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$ and $W^{(l)}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ denotes an activation function, such as the $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$. $H^{(l)} \in \mathbb{R}^{N\times D}$ is the matrix of activations in the l^{th} layer; $H^{(0)} = X$. In the following, we show that the form of this propagation rule can be motivated[1] via a first-order approximation of localized spectral filters on graphs (Hammond et al., 2011; Defferrard et al., 2016).

2.1 Spectral Graph Convolutions

We consider spectral convolutions on graphs defined as the multiplication of a signal $x \in \mathbb{R}^N$ (a scalar for every node) with a filter $g_\theta = \mathrm{diag}(\theta)$ parameterized by $\theta \in \mathbb{R}^N$ in the Fourier domain, i.e.:

$$g_\theta \star x = U g_\theta U^\top x , \tag{3}$$

where U is the matrix of eigenvectors of the normalized graph Laplacian $L = I_N - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U\Lambda U^\top$, with a diagonal matrix of its eigenvalues Λ and $U^\top x$ being the graph Fourier transform of x. We can understand g_θ as a function of the eigenvalues of L, i.e. $g_\theta(\Lambda)$. Evaluating Eq. 3 is computationally expensive, as multiplication with the eigenvector matrix U is $\mathcal{O}(N^2)$. Furthermore, computing the eigendecomposition of L in the first place might be prohibitively expensive for large graphs. To circumvent this problem, it was suggested in Hammond et al. (2011) that $g_\theta(\Lambda)$ can be well-approximated by a truncated expansion in terms of Chebyshev polynomials $T_k(x)$ up to K^{th} order:

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda}) , \tag{4}$$

with a rescaled $\tilde{\Lambda} = \frac{2}{\lambda_{\max}}\Lambda - I_N$. $\lambda_{\max}$ denotes the largest eigenvalue of L. $\theta' \in \mathbb{R}^K$ is now a vector of Chebyshev coefficients. The Chebyshev polynomials are recursively defined as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0(x) = 1$ and $T_1(x) = x$. The reader is referred to Hammond et al. (2011) for an in-depth discussion of this approximation.

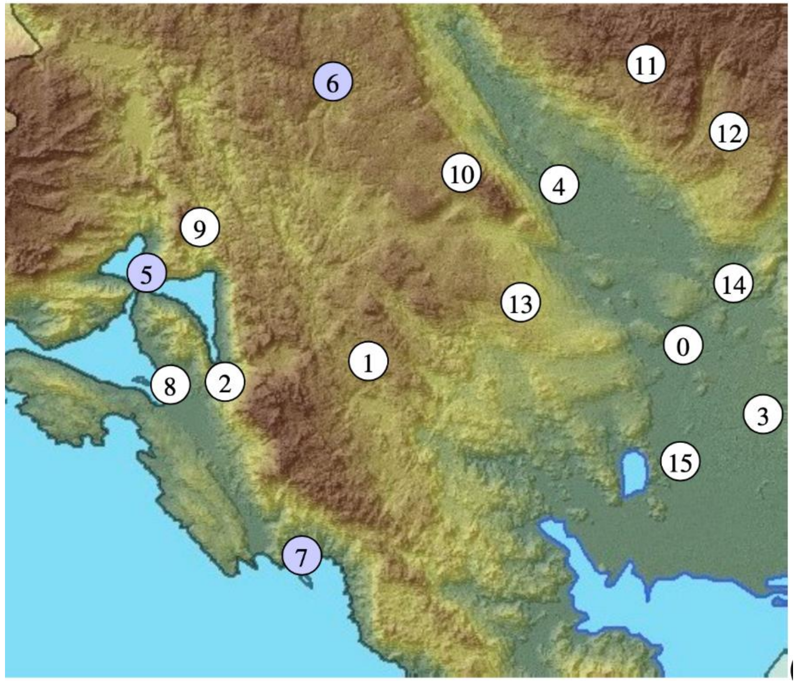
Going back to our definition of a convolution of a signal x with a filter $g_{\theta'}$, we now have:

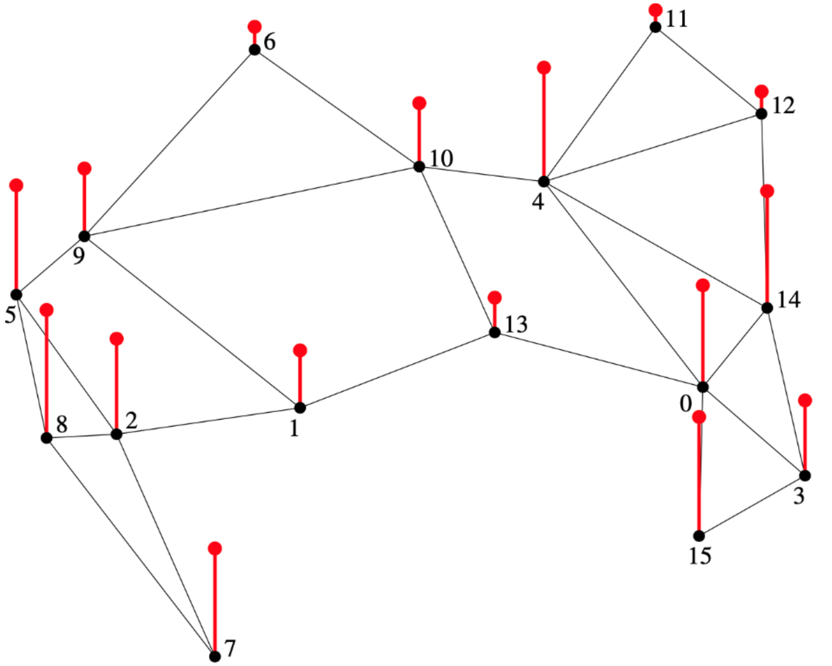
$$g_{\theta'} \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x , \tag{5}$$

with $\tilde{L} = \frac{2}{\lambda_{\max}}L - I_N$; as can easily be verified by noticing that $(U\Lambda U^\top)^k = U\Lambda^k U^\top$. Note that this expression is now K-localized since it is a K^{th}-order polynomial in the Laplacian, i.e. it depends only on nodes that are at maximum K steps away from the central node (K^{th}-order neighborhood). The complexity of evaluating Eq. 5 is $\mathcal{O}(|\mathcal{E}|)$, i.e. linear in the number of edges. Defferrard et al. (2016) use this K-localized convolution to define a convolutional neural network on graphs.

Graph Signal

- The temperature measured by sensors is considered as the Graph Signal, denoted by a vector $\boldsymbol{x} \in \mathcal{R}^n$.

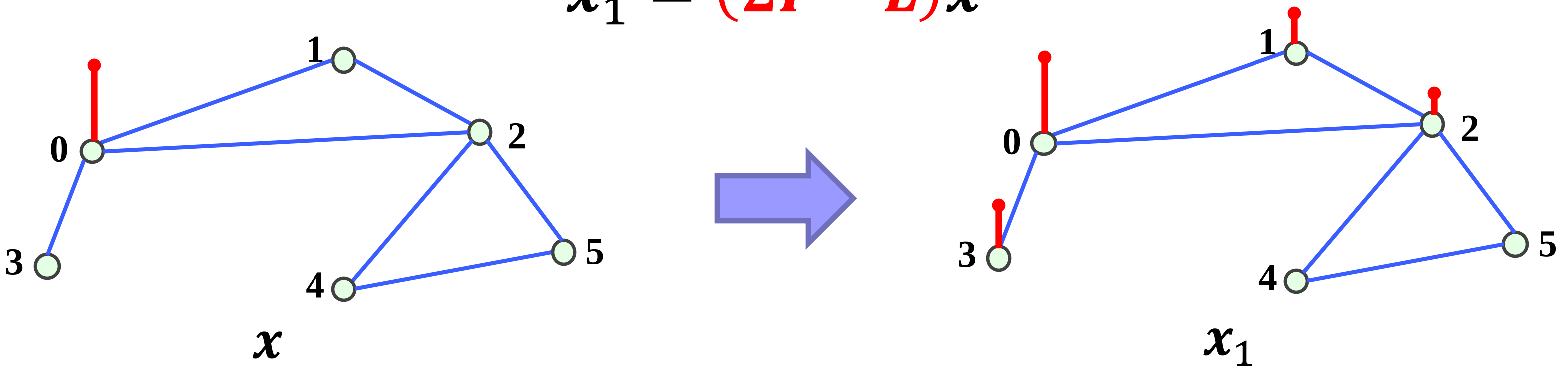




Graph Signal

- Operation on graph signal by Laplacian matrix $\boldsymbol{L}$

$$\boldsymbol{x}_1 = (2\boldsymbol{I} - \boldsymbol{L})\boldsymbol{x}$$

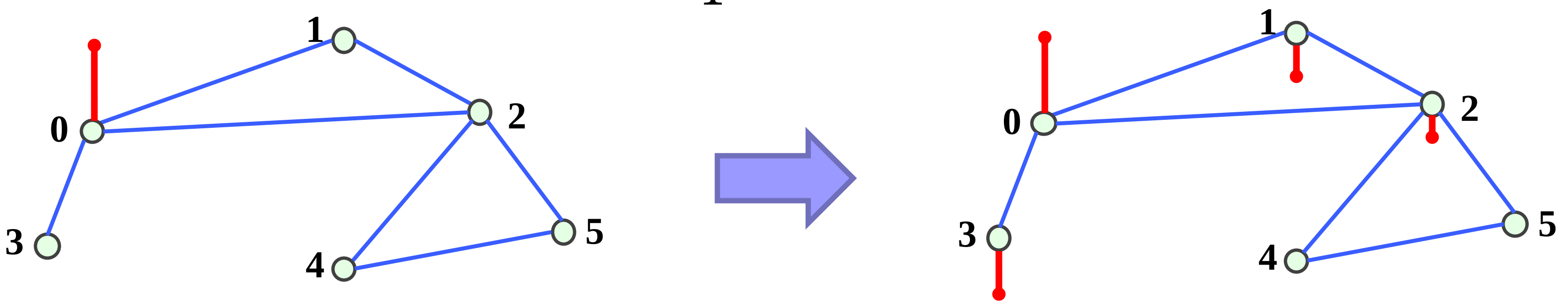


$$\begin{bmatrix} 1 & \frac{1}{\sqrt{3}\cdot\sqrt{2}} & \frac{1}{\sqrt{3}\cdot\sqrt{4}} & \frac{1}{\sqrt{3}\cdot\sqrt{1}} & 0 & 0 \\ \frac{1}{\sqrt{3}\cdot\sqrt{2}} & 1 & \frac{1}{\sqrt{2}\cdot\sqrt{4}} & 0 & 0 & 0 \\ \frac{1}{\sqrt{3}\cdot\sqrt{4}} & \frac{1}{\sqrt{2}\cdot\sqrt{4}} & 1 & 0 & \frac{1}{\sqrt{4}\cdot\sqrt{2}} & \frac{1}{\sqrt{4}\cdot\sqrt{2}} \\ \frac{1}{\sqrt{3}\cdot\sqrt{1}} & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{4}\cdot\sqrt{2}} & 0 & 1 & \frac{1}{\sqrt{2}\cdot\sqrt{2}} \\ 0 & 0 & \frac{1}{\sqrt{4}\cdot\sqrt{2}} & 0 & \frac{1}{\sqrt{2}\cdot\sqrt{2}} & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{\sqrt{3}\cdot\sqrt{2}} \\ \frac{1}{\sqrt{3}\cdot\sqrt{4}} \\ \frac{1}{\sqrt{3}\cdot\sqrt{1}} \\ 0 \\ 0 \end{bmatrix}$$

Graph Signal

- Operation on graph signal by Laplacian matrix $\boldsymbol{L}$

$$\boldsymbol{x}_1 = \boldsymbol{L}\boldsymbol{x}$$



$\boldsymbol{x}$ $\qquad$ $\boldsymbol{x}_1$

$$\begin{bmatrix} 1 & \frac{-1}{\sqrt{3}\cdot\sqrt{2}} & \frac{-1}{\sqrt{3}\cdot\sqrt{4}} & \frac{-1}{\sqrt{3}\cdot\sqrt{1}} & 0 & 0 \\ \frac{-1}{\sqrt{3}\cdot\sqrt{2}} & 1 & \frac{-1}{\sqrt{2}\cdot\sqrt{4}} & 0 & 0 & 0 \\ \frac{-1}{\sqrt{3}\cdot\sqrt{4}} & \frac{-1}{\sqrt{2}\cdot\sqrt{4}} & 1 & 0 & \frac{-1}{\sqrt{4}\cdot\sqrt{2}} & \frac{-1}{\sqrt{4}\cdot\sqrt{2}} \\ \frac{-1}{\sqrt{3}\cdot\sqrt{1}} & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-1}{\sqrt{4}\cdot\sqrt{2}} & 0 & 1 & \frac{-1}{\sqrt{2}\cdot\sqrt{2}} \\ 0 & 0 & \frac{-1}{\sqrt{4}\cdot\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}\cdot\sqrt{2}} & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{-1}{\sqrt{3}\cdot\sqrt{2}} \\ \frac{-1}{\sqrt{3}\cdot\sqrt{4}} \\ \frac{-1}{\sqrt{3}\cdot\sqrt{1}} \\ 0 \\ 0 \end{bmatrix}$$

Graph Fourier Transform

- The eigendecomposition of Laplacian matrix

$$\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T = \boldsymbol{U}\begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{pmatrix}\boldsymbol{U}^T,$$

where $\boldsymbol{U} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_n]$, $\boldsymbol{\Lambda} = \mathrm{diag}([\lambda_1, \dots, \lambda_n])$, $\boldsymbol{u}_i$ and λ_i for $i \in \{1,2, \dots, n\}$ denote the eigenvectors and eigenvalues, respectively, and $\lambda_i \in [0,2]$.

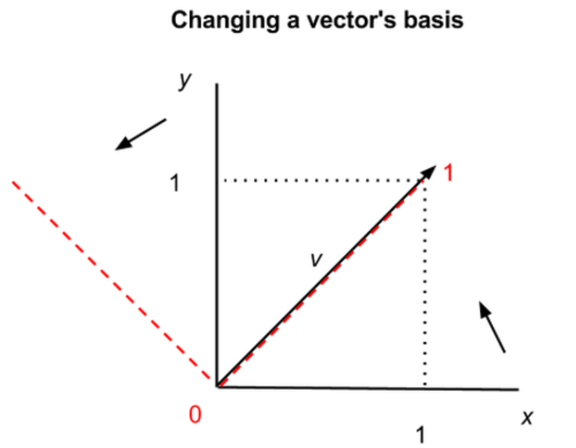
  - Orthonormal basis: $\boldsymbol{U} \cdot \boldsymbol{U}^T = \boldsymbol{I}$,

- Graph Fourier Transform of a signal: $\hat{\boldsymbol{x}} = \boldsymbol{U}^T\boldsymbol{x}$
- Inverse Graph Fourier Transform of a signal: $\boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}}$

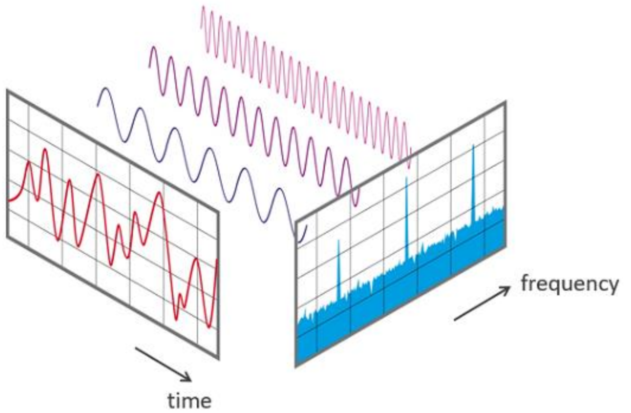
Graph Fourier Transform

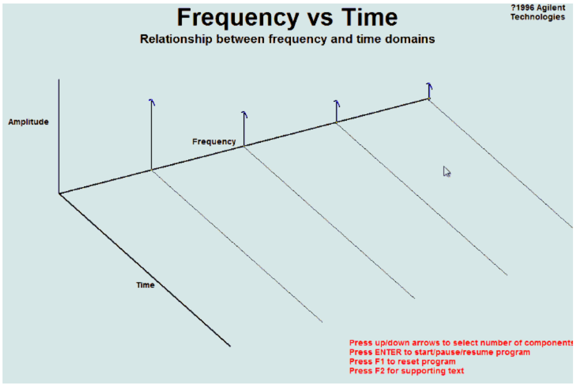
- $\boldsymbol{U}^T\boldsymbol{x}$ projects $\boldsymbol{x}$ to the orthogonal basis consisting of $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n$

$$\begin{bmatrix} \hat{\boldsymbol{x}}(1) \\ \hat{\boldsymbol{x}}(2) \\ \vdots \\ \hat{\boldsymbol{x}}(n) \end{bmatrix} = \begin{bmatrix} \boldsymbol{u}_1(1) & \boldsymbol{u}_1(2) & \dots & \boldsymbol{u}_1(n) \\ \boldsymbol{u}_2(1) & \boldsymbol{u}_2(2) & \dots & \boldsymbol{u}_2(n) \\ \vdots & \vdots & \dots & \dots \\ \boldsymbol{u}_n(1) & \boldsymbol{u}_n(2) & \dots & \boldsymbol{u}_n(n) \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}(1) \\ \boldsymbol{x}(2) \\ \vdots \\ \boldsymbol{x}(n) \end{bmatrix}$$

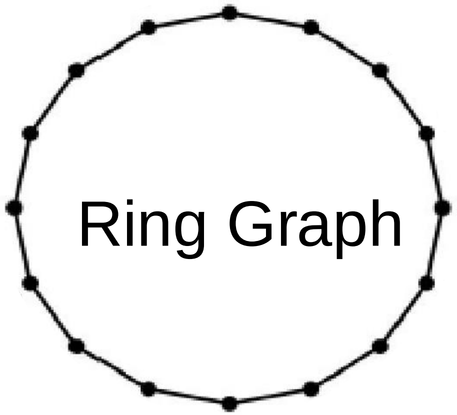


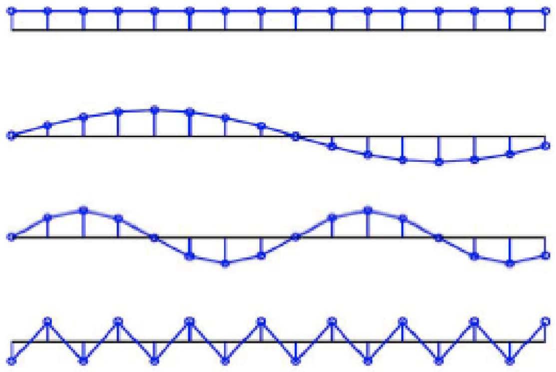
- Fourier Transform





- Graph Fourier Transform





Eigenvectors

Graph Convolution

- (Convolution theorem): the Fourier transform of a convolution of two signals is the pointwise product of their Fourier transforms.

$$\boldsymbol{x} *_G \boldsymbol{g} = \boldsymbol{U}\left((\boldsymbol{U}^T\boldsymbol{x})\odot(\boldsymbol{U}^T\boldsymbol{g})\right)$$

  - where $\odot$ denotes Hadamard products, $\boldsymbol{U}^T\boldsymbol{g}$ is the convolution filter. Reparametrize $\boldsymbol{U}^T\boldsymbol{g}$ as $\mathbf{diag}[\theta_1, \ldots, \theta_n]$:

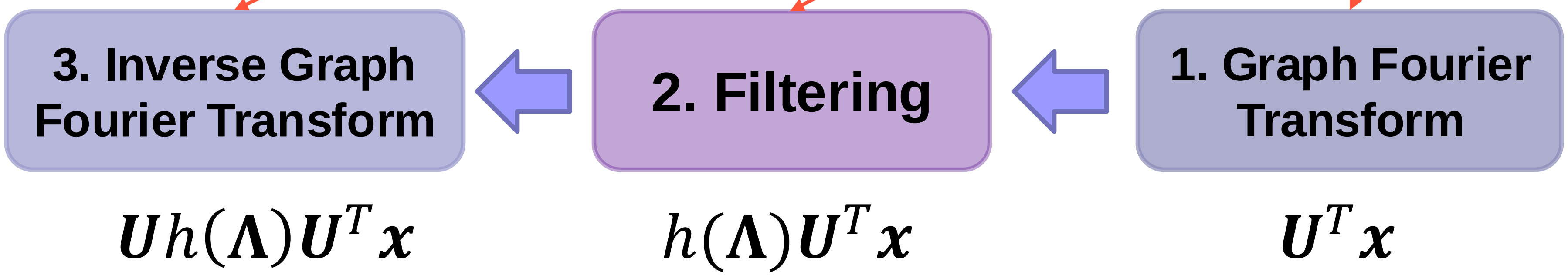
$$\boldsymbol{x} *_G \boldsymbol{g} = \boxed{\boldsymbol{U}}\begin{pmatrix} \theta_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \theta_n \end{pmatrix}\boxed{\boldsymbol{U}^T}\boldsymbol{x}$$

3. Inverse Graph Fourier Transform ⇐ **2. Convolution operation** ⇐ **1. Graph Fourier Transform**

Graph Filter

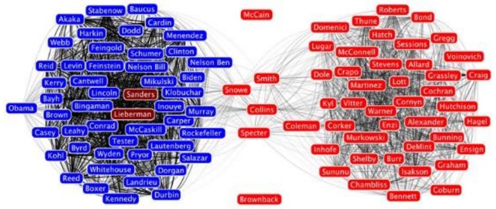
- Further reparametrize $\theta_i = h(\lambda_i)$

$$\boldsymbol{y} = h(\boldsymbol{L})\boldsymbol{x} = \boldsymbol{U}h(\boldsymbol{\Lambda})\boldsymbol{U}^T\boldsymbol{x} = \boxed{\boldsymbol{U}}\begin{pmatrix} h(\lambda_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & h(\lambda_n) \end{pmatrix}\boxed{\boldsymbol{U}^T}\boldsymbol{x}$$

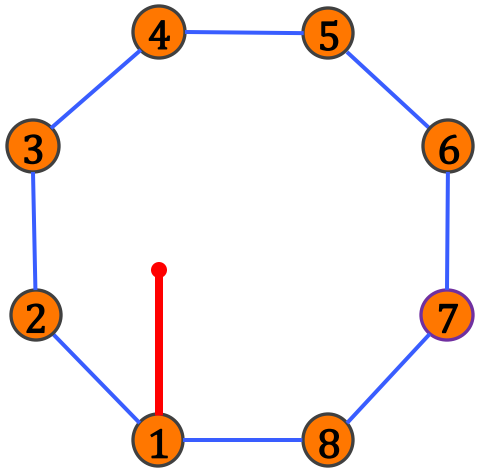


- We call $h(\boldsymbol{\Lambda})/h(\lambda)$ (graph) filter.

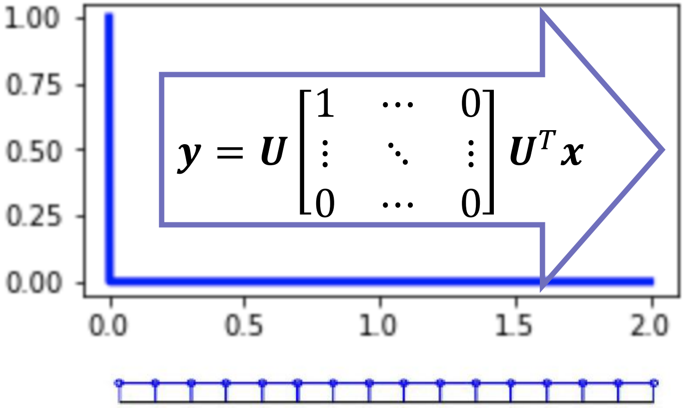
Homo./Heterophilic Graph & Filter



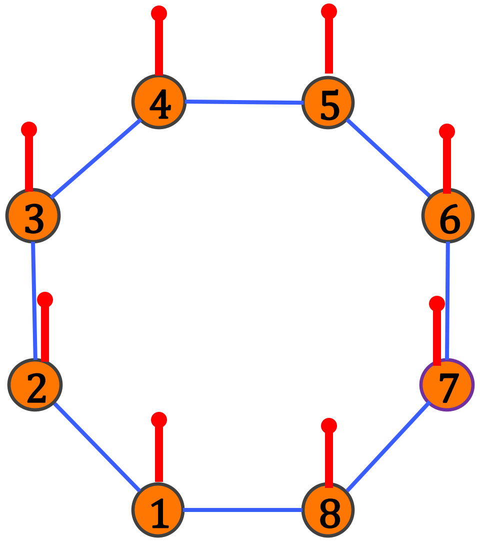
Homophilic graph

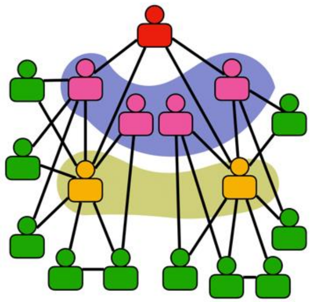


Impulse low-pass

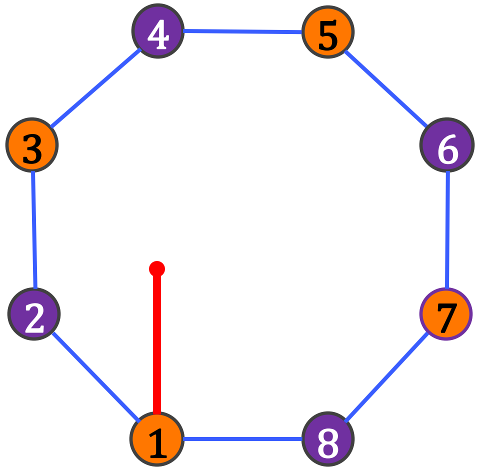


$$y = U \begin{bmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} U^T x$$

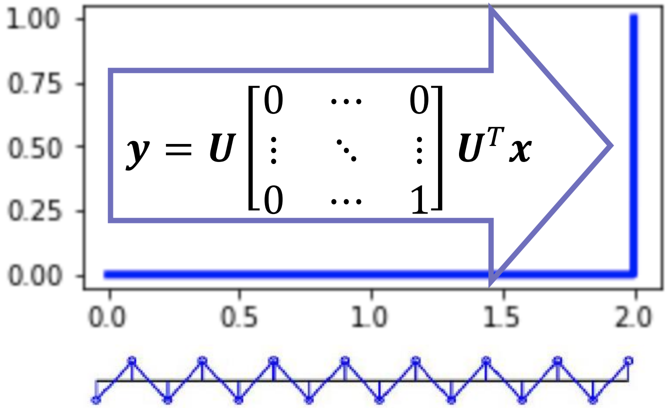




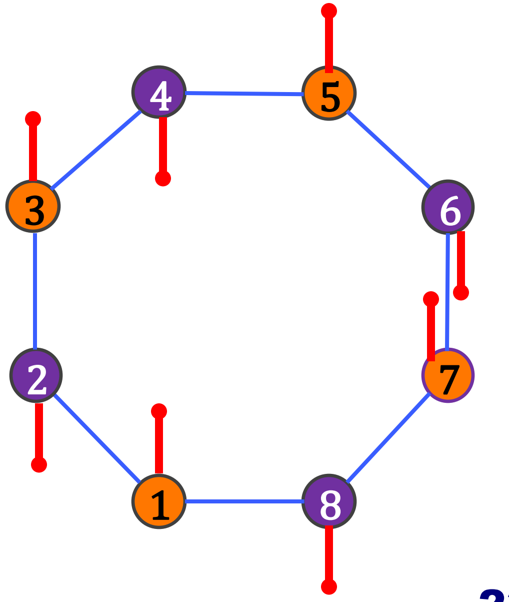
Heterophilic graph



Impulse high-pass



$$y = U \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{bmatrix} U^T x$$



Graph Filter

- How to design arbitrary filters?

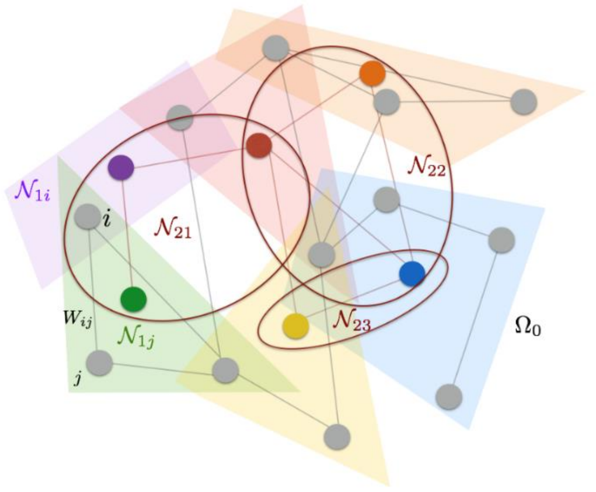
$$\boldsymbol{y} = \boldsymbol{U}\begin{pmatrix} h(\lambda_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & h(\lambda_n) \end{pmatrix}\boldsymbol{U}^T\boldsymbol{x}$$

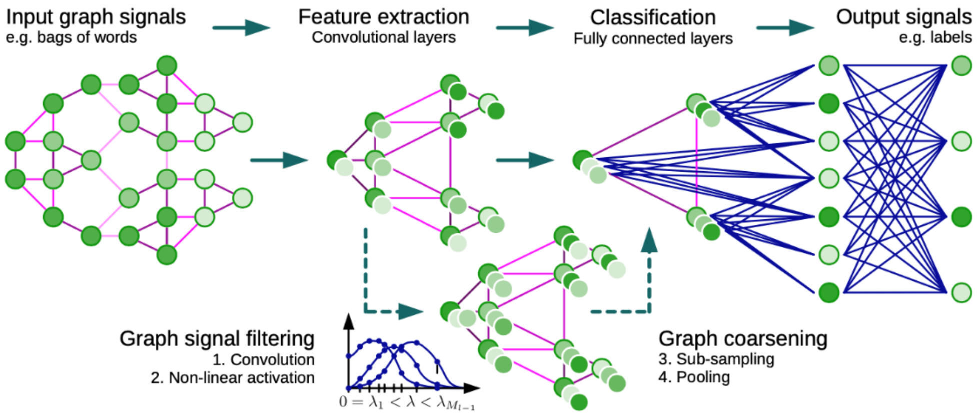
  - The $\boldsymbol{O}(\boldsymbol{n}^2)$ complexity of eigendecomposition is too high.

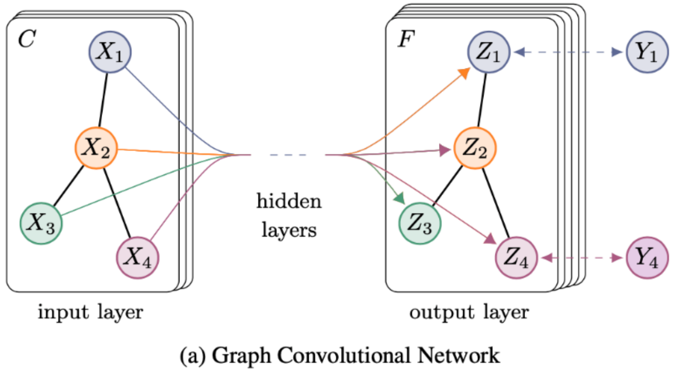
- Approximating filters by polynomials, complexity drops to $\boldsymbol{O}(\boldsymbol{m})$.

$$\boldsymbol{y} \approx \boldsymbol{U}\begin{pmatrix} \sum_{k=0}^{K} w_k {\lambda_1}^k & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sum_{k=0}^{K} w_k {\lambda_n}^k \end{pmatrix}\boldsymbol{U}^T\boldsymbol{x} = \sum_{k=0}^{K} w_k \boldsymbol{L}^k \boldsymbol{x}$$

Pioneering work





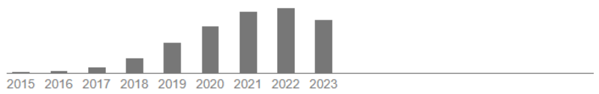


(a) Graph Convolutional Network

Spectral CNN [Bruna et al., ICLR'14]

$$\mathbf{h}_j^{(l+1)} = \sigma(\mathbf{U}\sum_{i=1}^{d_{in}} \mathbf{G}_{ij}^{(l)} \mathbf{U}^\top \mathbf{h}_i^{(l)}) \quad (j = 1 \cdots d_{out})$$

Total citations Cited by 5408

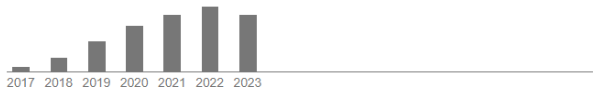


Scholar articles Spectral networks and locally connected networks on graphs
J Bruna, W Zaremba, A Szlam, Y LeCun - arXiv preprint arXiv:1312.6203, 2013
Cited by 5408 Related articles All 14 versions

ChebNet [Defferrard et al., NeurIPS'16]

$$\boldsymbol{H}^{(\ell+1)} = \sigma\left(\sum_{k=0}^{K} T_k(\hat{\boldsymbol{L}})\, \boldsymbol{H}^{(\ell)} \boldsymbol{W}^{(\ell,k)}\right)$$

Total citations Cited by 8121

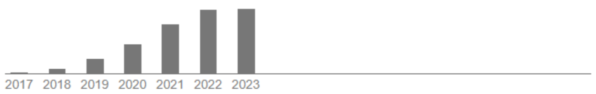


Scholar articles Convolutional neural networks on graphs with fast localized spectral filtering
M Defferrard, X Bresson, P Vandergheynst - Advances in neural information processing systems, 2016
Cited by 8092 Related articles All 10 versions

GCN [Kipf et al., ICLR'17]

$$\boldsymbol{H}^{(\ell+1)} = \sigma\left(\widetilde{\boldsymbol{P}} \cdot \boldsymbol{H}^{(\ell)} \cdot \boldsymbol{W}^{(\ell)}\right)$$

Total citations Cited by 29076



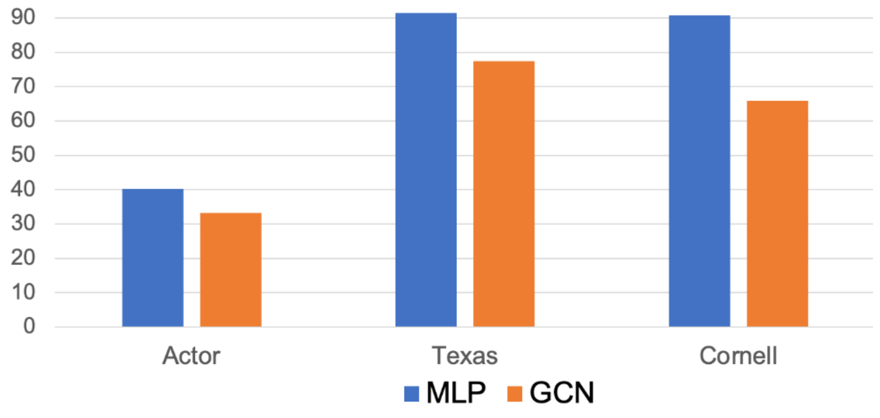
Scholar articles Semi-supervised classification with graph convolutional networks
TN Kipf, M Welling - arXiv preprint arXiv:1609.02907, 2016
Cited by 28944 Related articles All 23 versions

Spectral-based GNNs

- GCN[Kipf et al.,2017] uses a simplified first-order Chebyshev polynomial.
  - Filtering operation: (set $w_0 = -w_1 = \theta$ in $\sum_{k=0}^{K=1} w_k T_k(\lambda)$)

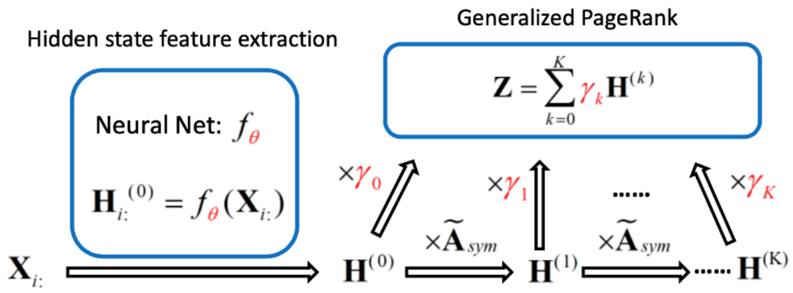
$$\boldsymbol{y} = (\theta \boldsymbol{I} - \theta(\boldsymbol{L} - \boldsymbol{I}))\boldsymbol{x}$$
$$= \theta(2\boldsymbol{I} - \boldsymbol{L})\boldsymbol{x}$$
$$= \theta\left(\boxed{\boldsymbol{I} + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}}\right)\boldsymbol{x} \xrightarrow{\text{Renormalization trick}} \theta\left(\boxed{\widetilde{\boldsymbol{D}}^{-1/2}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-1/2}}\right)\boldsymbol{x}$$

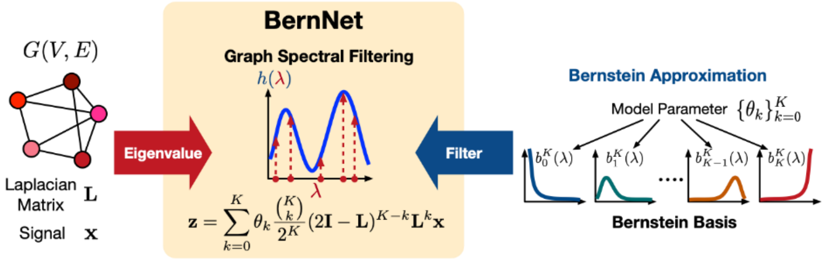
  - The filter of K layer GCN: $h(\tilde{\lambda}) = (1 - \tilde{\lambda})^K$, a fixed low-pass filter.



Accuracy of node classification on heterophilic graphs with GCN.

Polynomial Based Methods





GCNII [Chen et al., ICML'20] (Ours)

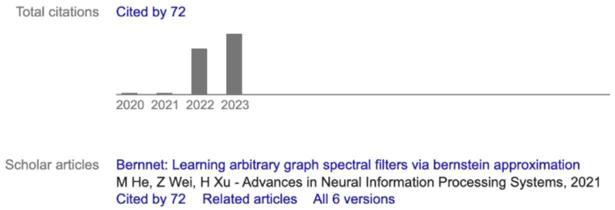
GPRGNN [Chien et al., ICLR'21]

BernNet [He et al., NeurIPS'21] (Ours)

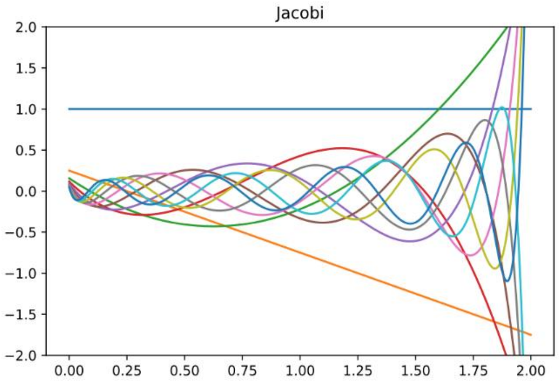
$$H^{(\ell+1)} = \sigma\left(\left((1-\alpha_\ell)\tilde{P}H^{(\ell)} + \alpha_\ell H^{(0)}\right)\left((1-\beta_\ell)I_n + \beta_\ell W^{(\ell)}\right)\right)$$

$$Y = \sum_{k=0}^{K} \gamma_k H^{(k)}, H^{(k)} = \tilde{P}H^{(k-1)}$$

$$Y = \left(\sum_{k=0}^{K} \theta_k \frac{1}{2^k}\binom{K}{k}(2I-L)^{K-k}L^k\right)X$$



Polynomial Based Methods

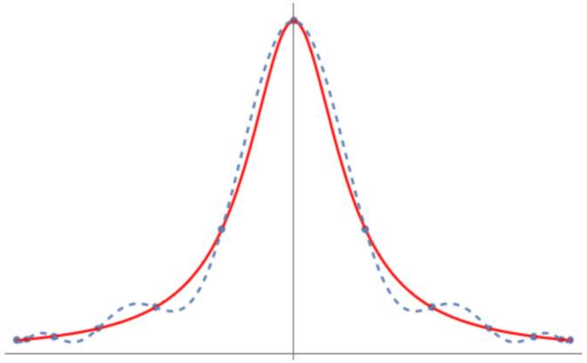


JacobiConv [Wang et al., ICML'22]

$$\boldsymbol{Y} = \sum_{k=0}^{K} \alpha_k P_k^{a,b}(\widetilde{\boldsymbol{P}})\boldsymbol{X}$$

Total citations Cited by 73

Scholar articles How powerful are spectral graph neural networks
X Wang, M Zhang - International Conference on Machine Learning, 2022
Cited by 73 Related articles All 5 versions



ChebNetII [He et al., NeurIPS'22] (Ours)

$$\mathbf{Y} = \frac{2}{K+1}\sum_{k=0}^{K}\sum_{j=0}^{K} \gamma_j T_k(x_j)\, T_k(\hat{\boldsymbol{L}})\boldsymbol{X}$$

Total citations Cited by 18

Scholar articles Convolutional neural networks on graphs with chebyshev approximation, revisited
M He, Z Wei, JR Wen - Advances in Neural Information Processing Systems, 2022
Cited by 18 Related articles All 5 versions

- Learn the basis?
- Optimal Basis?

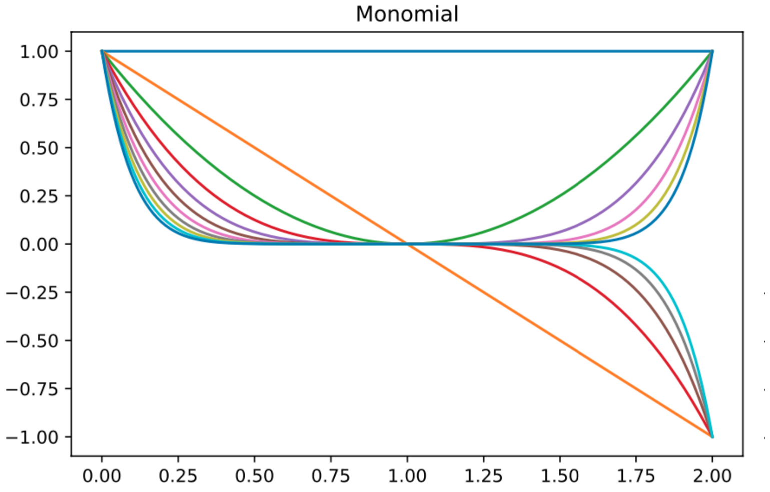
OPTBasis [Guo et al., ICML'23] (Ours)

Outline

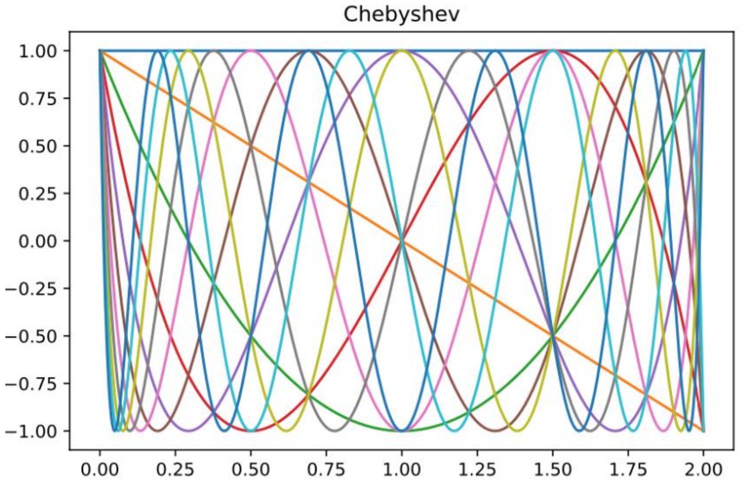
- Overview of GNNs
- Spectral interpretation of GNNs
- Our works (OptBasisGNN, PolyGCL, PSHGCN)
- Summary & Perspectives

Motivations

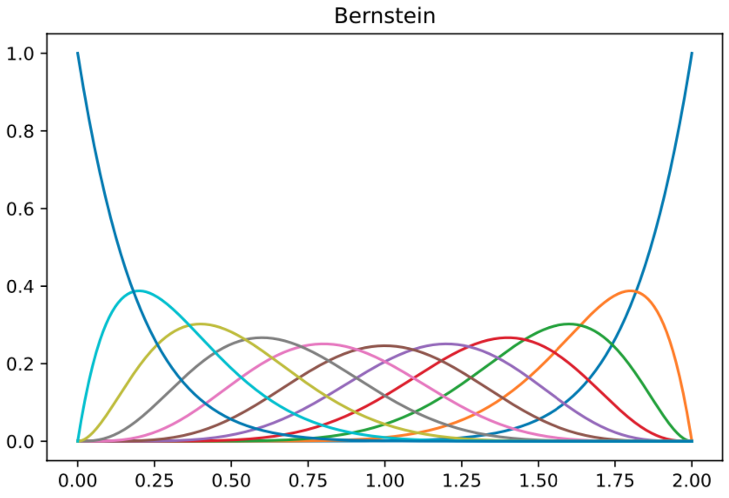
GPR-GNN [Chien et al., 2021]



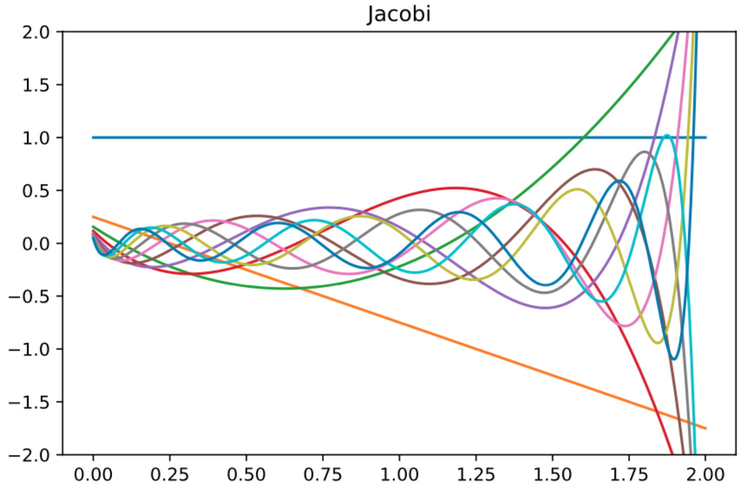
ChebNet [Defferrard et al., 2016]



BernNet [He et al., 2021]



JacobiConv [Wang et al., 2022]



- Learn the bases?
- Optimal bases?

Yuhe G, Wei Z. Graph Neural Networks with Learnable and Optimal Polynomial Bases [ICML'23] (**Ours**)

FavardGNN

- [Three-term Recurrence] *Any* orthonormal polynomial series satisfies the *three-term recurrence formula*:

$$\boxed{\sqrt{\beta_{k+1}}\, p_{k+1}(x) = (x-\gamma_k)p_k(x) - \sqrt{\beta_k}\, p_{k-1}(x),}$$

$$\boxed{p_{-1}(x) := 0,}\ \boxed{p_0(x) = 1/\sqrt{\beta_0},}$$

$$\gamma_k \in \mathbb{R},\ \sqrt{\beta_k} \in \mathbb{R}^+,\ k \geq 0$$

- [Favard's Theorem] Conversly, *any* polynomial series of such a recurrence is deemed to be orthonormal *w.r.t.* some weight function!

Algorithm 1: FAVARDFILTERING

Input: Input signals X with d channels; Normalized graph adjacency $\hat{P}$; Truncated polynomial order K

Learnable Parameters : β, γ, α

Output: Filtered Signals Z

$\boxed{x_{-1} \leftarrow 0}$

for $l = 0$ **to** $d-1$ **do**

  $x \leftarrow X_{:,l}$, $\boxed{x_0 \leftarrow x/\sqrt{\beta_{0,l}}}$, $z \leftarrow \alpha_{0,l} x_0$

  for $k = 0$ **to** K **do**

    $\boxed{x_{k+1} \leftarrow (\hat{P}x_k - \gamma_{k,l}x_k - \sqrt{\beta_{k,l}}x_{k-1})/\sqrt{\beta_{k+1,l}}}$

    $z \leftarrow z + \alpha_{k+1,l} x_{k+1}$

  $Z_{:,l} \leftarrow z$

return Z

Yuhe G, Wei Z. Graph Neural Networks with Learnable and Optimal Polynomial Bases [ICML'23] (**Ours**)

OptBasisGNN

- **Q2:** Is there an standard for "optimal bases"? Can we achieve them?
  - JacobiConv [ICML'22] proposed a definition of the optimal basis from the perspective of convergence;
  - But JacobiConv believe *habitually* that intractable eigen-decomposition is unavoidable. Thus the optimal basis ***cannot be utilized***.
  - We **solve** this optimal polynomial basis exactly in an implicit way
    - We solve the accompanying optimal vector basis.
      - The *next basis* depends *on the last and second last* solved basis.
    - Eigen-decomposition is avoided!

Algorithm 5: OBTAINNEXTBASISVECTOR
(In comment, we write the the $(k+1)$-th optimal basis polynomial $g_{k+1}(\cdot)$ based on $g_k(\cdot)$ and $g_{-1}(\cdot)$ that is implicitly used, but never solved explicitly.)

Input: Normalized graph $\hat{P}$; **Two** solved basis vectors v_{k-1}, v_k $(k \geq 0)$
Output: v_{k+1}

Step 1: $v^*_{k+1} \leftarrow \hat{P} v_k$ // $g^*_{k+1}(\mu) := \mu g_k(\mu)$
Step 2:
$\boxed{v^\perp_{k+1} \leftarrow v^*_{k+1} - \langle v^*_{k+1}, v_k \rangle v_k - \langle v^*_{k+1}, v_{k-1} \rangle v_{k-1}}$
// $g^\perp_{k+1}(\mu) :=$
$g^*_{k+1}(\mu) - \langle v^*_{k+1}, v_k \rangle g_k(\mu) - \langle v^*_{k+1}, v_{k-1} \rangle g_{k-1}(\mu)$
Step 3: $v_{k+1} \leftarrow v^\perp_{k+1} / \|v^\perp_{k+1}\|$
// $g_{k+1}(\mu) := g^\perp_{k+1}(\mu) / \|v^\perp_{k+1}$
return v_{k+1}

Yuhe G, Wei Z. Graph Neural Networks with Learnable and Optimal Polynomial Bases [ICML'23] (**Ours**)

Experiments

1. Node classification (1)

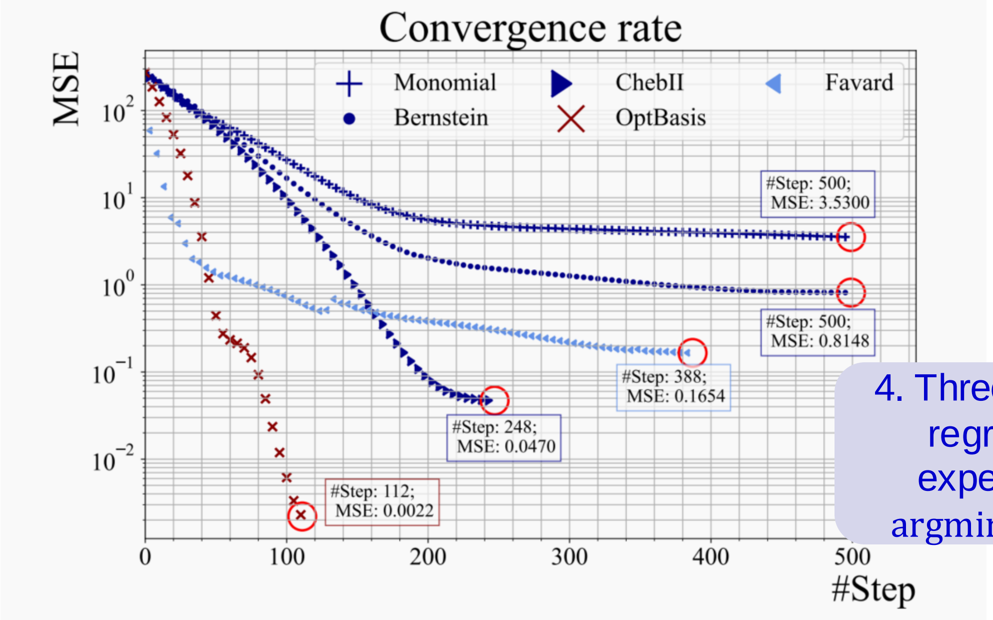
Dataset	Chameleon	Squirrel	Actor	Citeseer	Pubmed
$\|V\|$	2,277	5,201	7,600	3,327	19,717
$\mathcal{H}(G)$	.23	.22	.22	.74	.80
MLP	46.59 ± 1.84	31.01 ± 1.18	40.18 ± 0.55	76.52 ± 0.89	86.14 ± 0.25
GCN	60.81 ± 2.95	45.87 ± 0.8	33.26 ± 1.15	79.85 ± 0.78	86.79 ± 0.31
ChebNet	59.51 ± 1.25	40.81 ± 0.42	37.42 ± 0.58	79.33 ± 0.57	87.82 ± 0.24
ARMA	60.21 ± 1.00	36.27 ± 0.62	37.67 ± 0.54	80.04 ± 0.55	86.93 ± 0.24
APPNP	52.15 ± 1.79	35.71 ± 0.78	39.76 ± 0.49	80.47 ± 0.73	88.13 ± 0.33
GPRGNN	67.49 ± 1.38	50.43 ± 1.89	39.91 ± 0.62	80.13 ± 0.84	88.46 ± 0.31
BernNet	68.53 ± 1.68	51.39 ± 0.92	41.71 ± 1.12	80.08 ± 0.75	88.51 ± 0.39
ChebNetII	71.37 ± 1.01	57.72 ± 0.59	41.75 ± 1.07	80.53 ± 0.79	88.93 ± 0.29
JacobiConv	74.20 ± 1.03	57.38 ± 1.25	41.17 ± 0.64	80.78 ± 0.79	89.62 ± 0.41
FavardGNN	72.32 ± 1.90	63.49 ± 1.47	43.05 ± 0.53	81.89 ± 0.63	90.90 ± 0.27
OptBasisGNN	74.26 ± 0.74	63.62 ± 0.76	42.39 ± 0.52	80.58 ± 0.82	90.30 ± 0.19

3. Scalability experiments

Dataset	ogbn-arxiv	ogbn-papers100M
$\|V\|$	169,343	111,059,956
$\|E\|$	1,166,243	1,615,685,872
$\mathcal{H}(G)$	0.66	-
GCN	71.74 ± 0.29	OOM
ChebNet	71.12 ± 0.22	OOM
ARMA	71.47 ± 0.25	OOM
GPR-GNN	71.78 ± 0.18	65.89 ± 0.35
BernNet	71.96 ± 0.27	–
SIGN	71.95 ± 0.12	65.68 ± 0.16
GBP	71.21 ± 0.17	65.23 ± 0.31
NDLS*	72.24 ± 0.21	65.61 ± 0.29
ChebNetII	72.32 ± 0.23	67.18±0.32
OptBasisGNN	72.27 ± 0.15	67.22 ± 0.15

Dataset	Penn94	Genius	Twitch-Gamers	Pokec	Wiki
$\|V\|$	41,554	421,961	168,114	1,632,803	1,925,342
$\|E\|$	1,362,229	984,979	6,797,557	30,622,564	303,434,860
$\mathcal{H}(G)$	.470	.618	.545	.445	.389
MLP	73.61 ± 0.40	86.68 ± 0.09	60.92 ± 0.07	62.37 ± 0.02	37.38 ± 0.21
GCN	82.47 ± 0.27	87.42 ± 0.31	62.18 ± 0.26	75.45 ± 0.17	OOM
GCNII	82.92 ± 0.59	90.24 ± 0.09	63.39 ± 0.61	78.94 ± 0.11	OOM
MixHop	83.47 ± 0.71	90.58 ± 0.16	65.64 ± 0.27	81.07 ± 0.16	49.15 ± 0.26
LINK	80.79 ± 0.49	73.56 ± 0.14	64.85 ± 0.21	80.54 ± 0.03	57.11 ± 0.26
LINKX	84.71 ± 0.52	90.77 ± 0.27	66.06 ± 0.19	82.04 ± 0.07	59.80 ± 0.41
GPRGNN	83.54 ± 0.32	90.15 ± 0.30	62.59 ± 0.38	80.74 ± 0.22	58.73 ± 0.34
BernNet	83.26 ± 0.29	90.47 ± 0.33	64.27 ± 0.31	81.67 ± 0.17	59.02 ± 0.29
ChebNetII	84.86 ± 0.33	90.85 ± 0.32	65.03 ± 0.27	82.33 ± 0.28	60.95 ± 0.39
FavardGNN	84.92 ± 0.41	90.29 ± 0.14	64.26 ± 0.12	-	-
OptBasisGNN	84.85 ± 0.39	90.83 ± 0.11	65.17 ± 0.16	82.83 ± 0.04	61.85 ± 0.03

2. Node Classification (Continued)



4. Three-channel regression experiment: $\arg\min |\boldsymbol{Z} - \boldsymbol{Y}|^2$

PolyGCL: GRAPH CONTRASTIVE LEARNING via Learnable Spectral Polynomial Filters

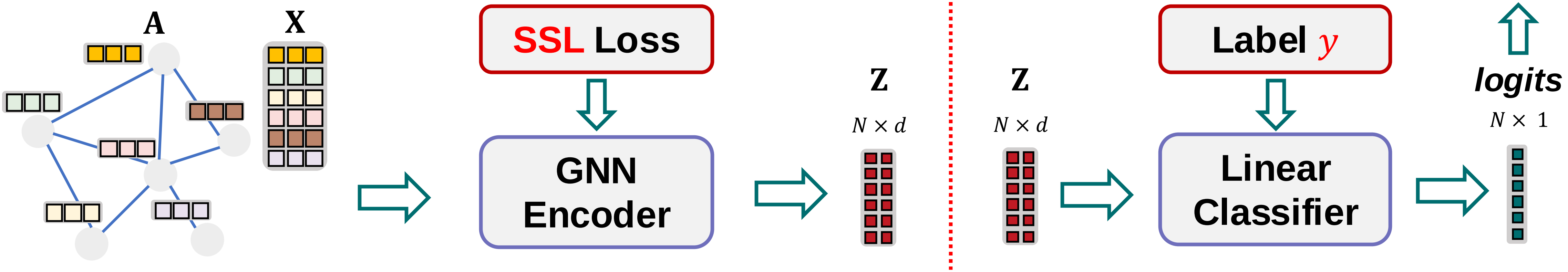
(ICLR 2024, spotlight)

Jingyu Chen, Runlin Lei, Zhewei Wei*

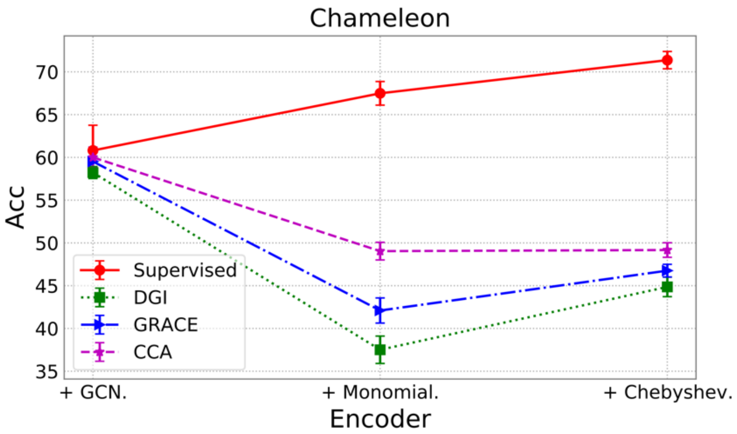
Motivations

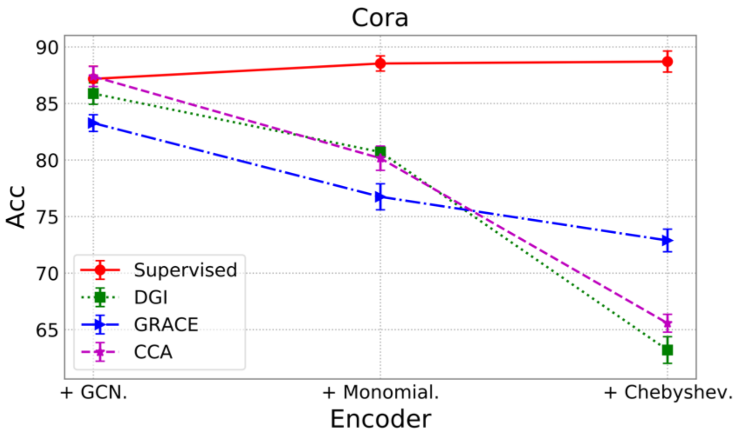
- Graph self-supervised learning (SSL)

Stage 2: Downstream tasks



- A natural idea: *Can we incorporate the excellent properties of spectral polynomial filters into graph contrastive learning?*





Spectral GNNs in supervised settings. ☺

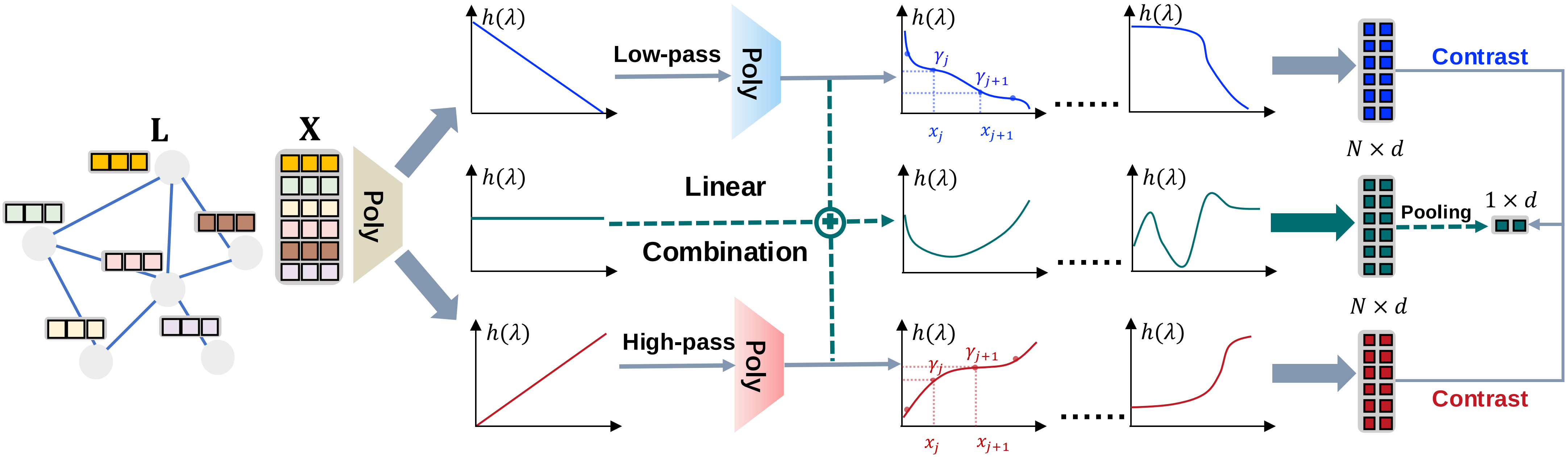
Plug in GCL setting directly? ☹

Chen J, Lei R, Wei Z. PolyGCL: Graph Contrastive Learning via Learnable Spectral Polynomial Filters. [ICLR'24] (**Ours**)

Model: PolyGCL

- **PolyGCL**
  - Encoder: ChebNetII [He et al., 2022]
  - Decoupling low-pass and high-pass:

$$\mathbf{Y} = \frac{2}{K+1}\sum_{k=0}^{K}\sum_{j=0}^{K}\gamma_j T_k(x_j) T_k(\hat{\mathbf{L}})\mathbf{X} \qquad \mathbf{Y} = \frac{2}{K+1}\sum_{k=0}^{K}\sum_{j=0}^{K}\gamma_j T_k(x_j) T_k(\hat{\mathbf{L}})\mathbf{X}$$



Chen J, Lei R, Wei Z. PolyGCL: Graph Contrastive Learning via Learnable Spectral Polynomial Filters. [ICLR'24] (Ours)

Experiments

- Downstream task
  - Node classification
  - Split: 60%/20%/20%
- Datasets
  - Synthetic:
    - cSBM [Chien et al., 2021]
    - Parameter $\phi \in [-1,1]$
  - Real-world:
    - Homophilic & Heterophilic

Methods	$\phi=-1$	$\phi=-0.75$	$\phi=-0.5$	$\phi=-0.25$	$\phi=0$	$\phi=0.25$	$\phi=0.5$	$\phi=0.75$	$\phi=1$
DGI	83.04 ± 0.92	93.24 ± 0.54	85.75 ± 0.49	68.41 ± 0.94	59.95 ± 0.78	68.70 ± 0.60	84.04 ± 0.61	91.53 ± 0.42	82.68 ± 0.72
MVGRL	68.80 ± 1.00	84.35 ± 0.78	78.81 ± 0.63	64.14 ± 1.05	59.09 ± 1.15	70.74 ± 0.73	89.91 ± 0.58	95.95 ± 0.37	89.13 ± 0.55
GGD	82.90 ± 0.83	92.76 ± 0.63	85.56 ± 0.58	66.63 ± 0.66	56.00 ± 0.51	67.06 ± 1.06	84.22 ± 0.61	91.75 ± 0.45	83.84 ± 0.76
GMI	54.47 ± 0.94	54.38 ± 0.71	50.70 ± 0.91	50.41 ± 0.64	51.79 ± 0.39	59.57 ± 0.93	82.28 ± 0.76	93.74 ± 0.46	96.01 ± 0.48
CCA-SSG	50.55 ± 0.75	52.71 ± 1.08	51.21 ± 0.98	50.88 ± 0.85	51.16 ± 0.67	56.33 ± 0.90	72.41 ± 1.20	90.83 ± 0.62	62.03 ± 0.91
BGRL	49.86 ± 0.77	49.47 ±0.74	49.95 ± 0.90	50.21 ± 0.87	54.58 ± 0.99	60.80 ± 0.56	70.79 ± 1.01	74.46 ± 0.79	68.69 ± 0.96
GBT	57.41 ± 1.43	64.99 ± 0.53	58.84 ± 0.80	51.80 ± 0.87	57.55 ± 0.69	**72.62 ± 0.63**	**91.09 ± 0.37**	97.80 ± 0.25	96.03 ± 0.38
GRACE	98.74 ± 0.28	**97.55 ± 0.17**	90.06 ± 0.50	68.74 ± 1.01	56.85 ± 1.12	66.70 ± 0.91	89.50 ± 0.60	97.41 ± 0.25	98.78 ± 0.28
GCA	76.56 ± 0.92	85.56 ± 0.40	78.96 ± 0.43	62.32 ± 0.89	58.01 ± 1.07	65.30 ± 1.15	77.16 ± 1.03	81.38 ± 0.59	75.54 ± 0.76
GraphCL	58.82 ± 1.06	57.89 ± 0.68	52.91 ± 0.70	50.18 ± 0.59	51.25 ± 0.76	55.11 ± 0.56	62.54 ± 1.13	65.57 ± 1.17	71.31 ± 1.01
GREET	50.82 ± 0.67	58.79 ± 0.52	59.91 ± 1.09	63.57 ± 0.76	65.99 ± 0.64	71.04 ± 0.67	80.17 ± 0.50	83.11 ± 0.53	75.93 ± 1.19
POLYGCL	**98.84 ± 0.17**	94.23 ± 0.31	**90.82 ± 0.50**	**75.43 ± 0.68**	**66.51 ± 0.69**	69.43 ± 0.65	88.22 ± 0.72	**98.09 ± 0.29**	**99.29 ± 0.23**

Methods	Cora	Citeseer	Pubmed	Cornell	Texas	Wisconsin	Actor	Chameleon	Squirrel
DGI	85.88 ± 0.95	76.44 ± 0.80	82.13 ± 0.24	70.82 ± 7.21	81.48 ± 2.79	75.00 ± 2.00	32.09 ± 1.18	58.23 ± 0.70	38.80 ± 0.76
MVGRL	87.36 ± 0.64	78.70 ± 0.64	86.30 ± 0.23	67.70 ± 4.75	73.11 ± 4.75	74.25 ± 4.13	32.98 ± 0.53	57.75 ± 1.20	40.25 ± 1.14
GGD	87.21 ± 1.08	79.25 ± 0.72	85.38 ± 0.25	80.33 ± 1.80	82.62 ± 3.11	73.25 ± 2.25	32.27 ± 1.11	57.64 ± 1.16	40.87 ± 0.66
GMI	85.09 ± 1.13	76.38 ± 0.70	83.06 ± 0.24	62.79 ± 7.54	68.03 ± 4.10	62.13 ± 2.88	32.37 ± 1.01	62.47 ± 1.55	39.82 ± 0.93
CCA-SSG	87.39 ± 0.89	79.60 ± 0.71	84.96 ± 0.20	78.69 ± 3.44	87.87 ± 1.64	82.88 ± 1.50	34.86 ± 0.56	60.00 ± 1.20	41.50 ± 0.72
BGRL	84.45 ± 0.66	74.84 ± 1.04	83.06 ± 0.29	59.84 ± 2.95	69.84 ± 3.61	62.88 ± 4.13	32.48 ± 0.67	64.09 ± 1.27	47.02 ± 0.88
GBT	84.89 ± 1.13	76.59 ± 0.68	86.10 ± 0.23	59.18 ± 9.34	72.79 ± 6.56	62.38 ± 3.00	34.34 ± 0.67	68.77 ± 1.25	48.86 ± 0.80
GRACE	83.27 ± 0.74	73.79 ± 0.60	81.71 ± 0.16	60.66 ± 11.32	75.74 ± 2.95	72.13 ± 2.75	31.97 ± 1.15	59.52 ± 1.49	42.68 ± 0.90
GCA	84.09 ± 0.85	75.23 ± 0.75	82.01 ± 0.31	53.11 ± 9.34	81.97 ± 2.30	73.50 ± 3.00	31.13 ± 0.71	65.54 ± 1.07	47.13 ± 0.61
GraphCL	86.54 ± 0.54	78.99 ± 0.50	85.16 ± 0.21	61.48 ± 5.74	66.07 ± 6.07	60.63 ± 3.50	32.45 ± 1.22	58.49 ± 1.31	42.92 ± 0.62
GREET	85.16 ± 0.77	79.06 ± 0.44	85.64 ± 0.24	78.36 ± 3.77	78.03 ± 3.94	84.63 ± 3.88	38.26 ± 0.87	60.57 ± 1.03	39.76 ± 0.74
POLYGCL	**87.57 ± 0.62**	**79.81 ± 0.85**	**87.15 ± 0.27**	**82.62 ± 3.11**	**88.03 ± 1.80**	**85.50 ± 1.88**	**41.15 ± 0.88**	**71.62 ± 0.96**	**56.49 ± 0.72**

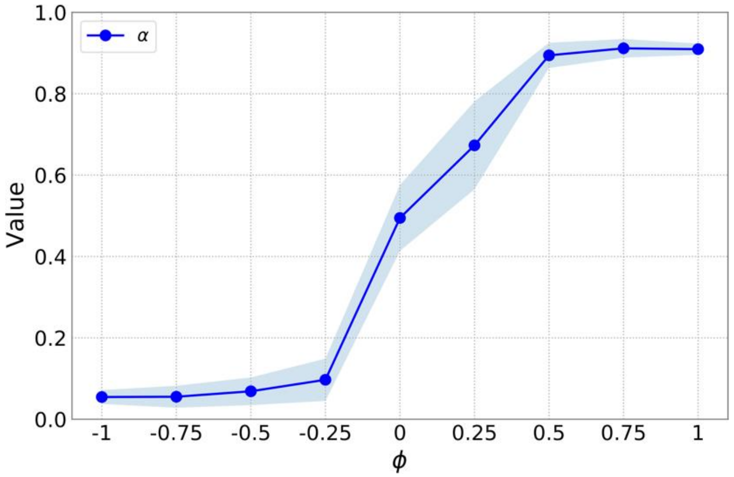
Methods	Roman-empire	Amazon-ratings	Minesweeper	Tolokers	Questions
DGI	58.57 ± 0.26	42.72 ± 0.42	68.36 ± 0.60	76.29 ± 0.66	74.44 ± 0.63
MVGRL	70.02 ± 0.25	42.18 ± 0.29	**90.07 ± 0.36**	80.86 ± 0.63	OOM
GGD	58.04 ± 0.40	43.15 ± 0.34	78.15 ± 0.48	76.43 ± 0.63	74.63 ± 0.66
GMI	32.33 ± 0.27	40.98 ± 0.30	72.38 ± 0.63	79.89 ± 0.62	OOM
CCA-SSG	42.82 ± 0.24	41.23 ± 0.25	72.42 ± 0.60	75.46 ± 0.75	74.64 ± 0.57
BGRL	39.34 ± 0.32	41.17 ± 0.25	72.82 ± 0.60	79.73 ± 0.61	72.27 ± 0.55
GBT	45.96 ± 0.34	43.58 ± 0.28	72.39 ± 0.56	75.74 ± 0.78	**75.98 ± 0.88**
GRACE	59.57 ± 0.39	43.79 ± 0.28	68.10 ± 0.70	76.31 ± 0.71	74.34 ± 0.71
GCA	59.77 ± 0.40	42.57 ± 0.17	68.11 ± 0.66	77.26 ± 0.61	75.09 ± 0.57
GraphCL	29.92 ± 0.30	37.81 ± 0.14	82.15 ± 0.46	76.88 ± 0.60	60.51 ± 1.45
GREET	72.68 ± 0.31	41.19 ± 0.25	82.71 ± 0.51	80.60 ± 0.56	OOM
POLYGCL	**72.97 ± 0.25**	**44.29 ± 0.43**	86.11 ± 0.43	**83.73 ± 0.53**	75.33 ± 0.67

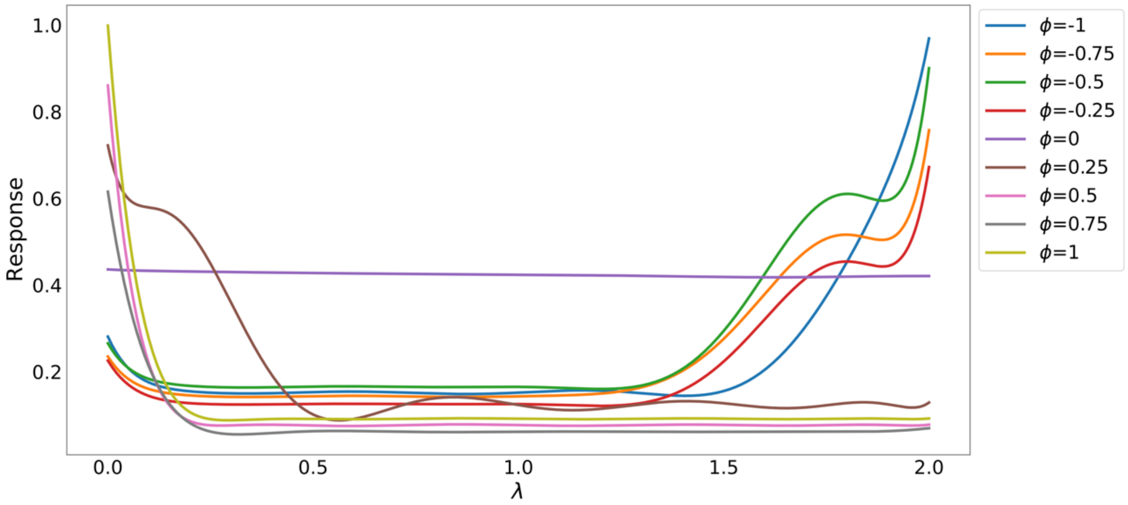
Results:

Baselines: homophily assumption.

PolyGCL: works in all settings.

Experiments





	arXiv-year
DGI	40.60 $\pm$ 0.21
GGD	40.86 $\pm$ 0.22
MVGRL	-
BGRL	OOM
GBT	41.90 $\pm$ 0.26
CCA-SSG	40.76 $\pm$ 0.25
GRACE	OOM
POLYGCL	**43.07** $\pm$ **0.23**

Linear coefficients: $\alpha + \beta = 1$

Normalized learned filters

Time complexity: $O(KE + N)$

$\phi < 0$: low-pass information accounts for less.

$\phi < 0$: increasing fuctions;
$\phi = 0$: all-pass property.

PolyGCL: SOTA performance with satisfactory efficiency.

Chen J, Lei R, Wei Z. PolyGCL: Graph Contrastive Learning via Learnable Spectral Polynomial Filters. [ICLR'24] (**Ours**)

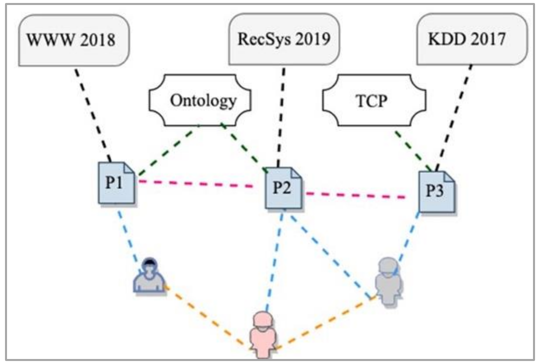
Spectral Heterogeneous Graph Convolutions via Positive Noncommutative Polynomials

(TheWebConf 2024, Oral)

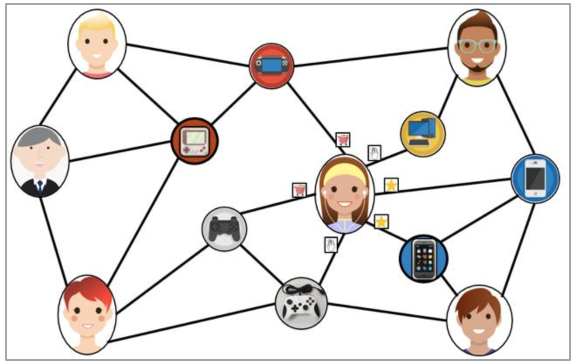
Mingguo He, Zhewei Wie, Shikun Feng,
Zhengjie Huang, Weibin Li, Yu Sun, Dianhai Yu

Motivation

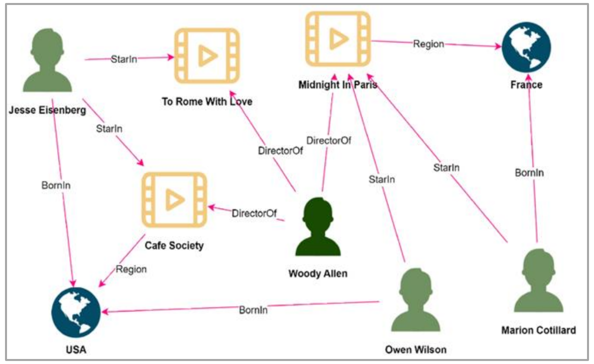
- Heterogeneous graphs are ubiquitous in our lives



Academic network

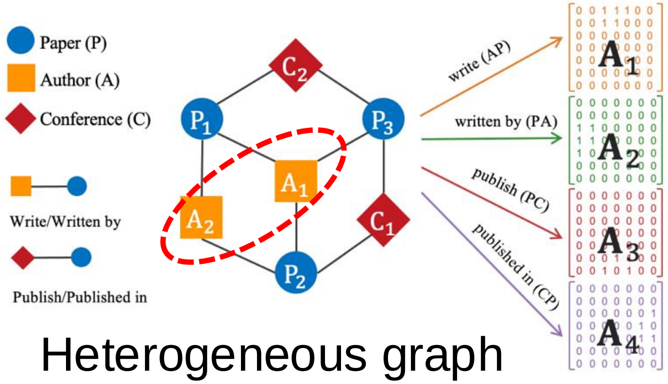


Social network



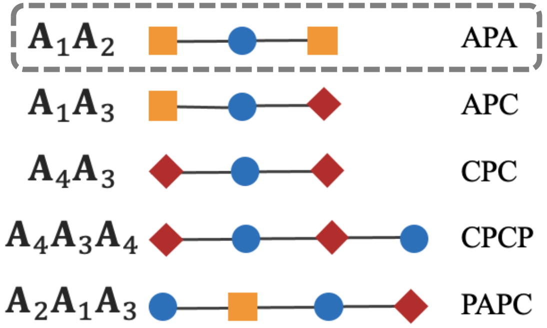
Movie network

- Traditional meta-path methods manually define meta-paths or learn them



Heterogeneous graph

$$\mathbf{A}_1\mathbf{A}_2 = \begin{array}{c|ccccccc} & A_1 & A_2 & P_1 & P_2 & P_3 & C_1 & C_2 \\ \hline A_1 & 3 & \boxed{2} & 0 & 0 & 0 & 0 & 0 \\ A_2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ P_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ P_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ C_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ C_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$



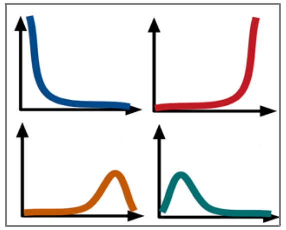
Meta-paths

He M, Wei Z, et al. Spectral Heterogeneous Graph Convolutions via Positive Noncommutative Polynomials. [TheWebConf'24] (Ours)

Motivation

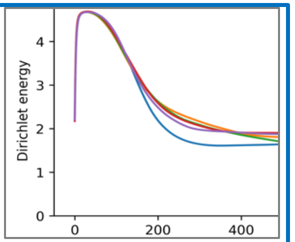
- How can we define a valid heterogeneous graph convolution on the spectral domain?

Spectral-based GNNs learn arbitrary filters on homogeneous graphs



Can learn arbitrary heterogenous graph filters

A generalized heterogeneous graph optimization problem
$$\min_{\mathbf{y}} f(\mathbf{y}) = \min_{\mathbf{y}} (1-\alpha)\mathbf{y}^T \gamma(\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_R)\mathbf{y} + \alpha \|\mathbf{y}-\mathbf{x}\|_2^2$$



Heterogenous filters have to be positive semidefinite

- Existing HGNNs do not meet these

Method	Shift $\mathbf{P}_r$	Graph Convolution
GTN [41]	$\mathbf{A}_r$	$\mathbf{D}^{-1}\prod_{k=0}^{K}\sum_{r=0}^{R}\alpha_r^{(k)}\mathbf{A}_r\mathbf{x}$
EMRGNN [33]	$\tilde{\mathbf{A}}_r$	$\sum_{k=0}^{K}\alpha(1-\alpha)^k\left(\sum_{r=1}^{R}\mu_r\tilde{\mathbf{A}}_r\right)^k\mathbf{x}$
MHGCN [40]	$\mathbf{A}_r$	$\left(\sum_{r=1}^{R}\beta_r\mathbf{A}_r\right)^K\mathbf{x}$

He M, Wei Z, et al. Spectral Heterogeneous Graph Convolutions via Positive Noncommutative Polynomials. [TheWebConf'24] (Ours)

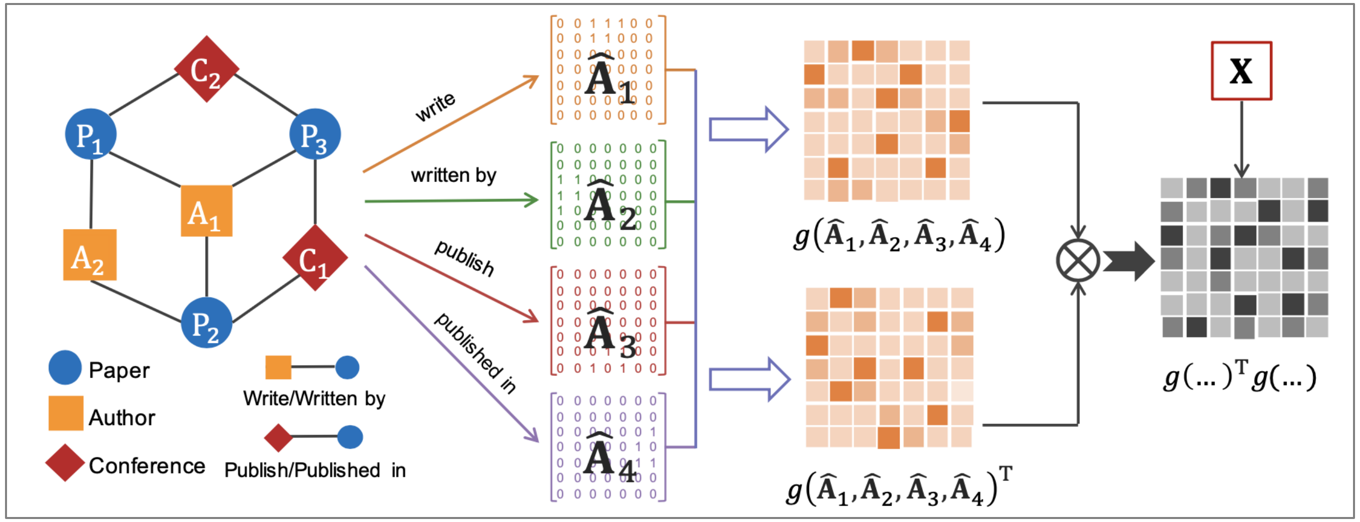
Method：PSHGCN

- Use positive noncommutative polynomials to approximate valid heterogeneous graph filters

$$h(\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_R) = \boxed{\sum_i g_i(\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_R)^{\mathrm{T}} g_i(\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_R)}$$

Sum of Squares[1]
- guarantees the filter h is positive semidefinite

- **P**ositive **S**pectral **H**eterogeneous **G**raph **C**onvolutional **N**etwork



[1] J William Helton. " positive" noncommutative polynomials are sums of squares. Annals of Mathematics, pages 675–694, 2002.

He M, Wei Z, et al. Spectral Heterogeneous Graph Convolutions via Positive Noncommutative Polynomials. [TheWebConf'24] (Ours)

Experiment

- Node classification

Table 2: Node classification performance (Mean F1 scores ± standard errors) comparison of different methods on four datasets. Tabular results are presented in percentages, with the best result highlighted in bold and the runner-up underlined.

	DBLP		ACM		IMDB		AMiner	
	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
GCN	$90.84_{\pm 0.32}$	$91.47_{\pm 0.34}$	$92.17_{\pm 0.24}$	$92.12_{\pm 0.23}$	$62.37_{\pm 1.35}$	$68.13_{\pm 0.83}$	$75.63_{\pm 1.08}$	$85.77_{\pm 0.43}$
GAT	$93.83_{\pm 0.27}$	$93.39_{\pm 0.30}$	$92.26_{\pm 0.94}$	$92.19_{\pm 0.93}$	$62.45_{\pm 1.36}$	$68.08_{\pm 0.49}$	$75.23_{\pm 0.60}$	$85.56_{\pm 0.65}$
GPRGNN	$91.66_{\pm 1.01}$	$92.45_{\pm 0.76}$	$92.36_{\pm 0.28}$	$92.28_{\pm 0.27}$	$63.02_{\pm 1.48}$	$68.83_{\pm 0.95}$	$75.32_{\pm 0.67}$	$86.13_{\pm 0.58}$
ChebNetII	$92.05_{\pm 0.53}$	$92.97_{\pm 0.48}$	$92.45_{\pm 0.37}$	$92.33_{\pm 0.38}$	$62.54_{\pm 1.29}$	$68.33_{\pm 0.92}$	$75.59_{\pm 0.73}$	$85.82_{\pm 0.52}$
RGCN	$91.52_{\pm 0.50}$	$92.07_{\pm 0.50}$	$91.55_{\pm 0.74}$	$91.41_{\pm 0.75}$	$63.24_{\pm 0.57}$	$66.51_{\pm 0.28}$	$63.03_{\pm 2.27}$	$82.79_{\pm 1.12}$
HAN	$91.67_{\pm 0.49}$	$92.05_{\pm 0.62}$	$90.89_{\pm 0.43}$	$90.79_{\pm 0.43}$	$62.05_{\pm 0.93}$	$67.69_{\pm 0.64}$	$63.86_{\pm 2.15}$	$82.95_{\pm 1.33}$
GTN	$93.52_{\pm 0.55}$	$93.97_{\pm 0.54}$	$91.31_{\pm 0.70}$	$91.20_{\pm 0.71}$	$64.59_{\pm 1.03}$	$68.27_{\pm 0.65}$	$72.39_{\pm 1.79}$	$84.74_{\pm 1.24}$
MAGNN	$93.28_{\pm 0.51}$	$93.76_{\pm 0.45}$	$90.88_{\pm 0.64}$	$90.77_{\pm 0.65}$	$61.36_{\pm 2.85}$	$67.82_{\pm 1.54}$	$71.56_{\pm 1.63}$	$83.48_{\pm 1.37}$
EMRGNN	$92.19_{\pm 0.38}$	$92.57_{\pm 0.37}$	$92.93_{\pm 0.34}$	$93.85_{\pm 0.33}$	$65.63_{\pm 1.97}$	$68.76_{\pm 0.78}$	$73.74_{\pm 1.25}$	$85.46_{\pm 0.74}$
MHGCN	$93.56_{\pm 0.41}$	$94.03_{\pm 0.43}$	$92.12_{\pm 0.66}$	$91.97_{\pm 0.68}$	$67.59_{\pm 1.25}$	$70.28_{\pm 0.71}$	$73.56_{\pm 1.75}$	$85.18_{\pm 1.28}$
SimpleHGN	$94.01_{\pm 0.24}$	$94.46_{\pm 0.22}$	$93.42_{\pm 0.44}$	$93.35_{\pm 0.45}$	$68.72_{\pm 1.54}$	$70.83_{\pm 1.07}$	$75.43_{\pm 0.88}$	$86.52_{\pm 0.73}$
HALO	$92.37_{\pm 0.32}$	$92.84_{\pm 0.34}$	$93.05_{\pm 0.31}$	$92.96_{\pm 0.33}$	$71.63_{\pm 0.77}$	$\underline{73.81_{\pm 0.72}}$	$74.91_{\pm 1.23}$	$\underline{87.25_{\pm 0.89}}$
SeHGNN	$\underline{95.06_{\pm 0.17}}$	$\underline{95.42_{\pm 0.17}}$	$\underline{94.05_{\pm 0.35}}$	$\underline{93.98_{\pm 0.36}}$	$\underline{71.71_{\pm 0.62}}$	$73.42_{\pm 0.47}$	$\underline{76.83_{\pm 0.57}}$	$86.96_{\pm 0.64}$
PSHGCN	$\mathbf{95.27_{\pm 0.13}}$	$\mathbf{95.61_{\pm 0.12}}$	$\mathbf{94.35_{\pm 0.23}}$	$\mathbf{94.27_{\pm 0.23}}$	$\mathbf{72.33_{\pm 0.57}}$	$\mathbf{74.46_{\pm 0.32}}$	$\mathbf{77.26_{\pm 0.75}}$	$\mathbf{88.21_{\pm 0.31}}$

He M, Wei Z, et al. Spectral Heterogeneous Graph Convolutions via Positive Noncommutative Polynomials. [TheWebConf'24] (Ours)

Experiment

- Link prediction

Table 3: Link prediction performance (ROC-AUC/MRR ± standard errors). Results are presented in percent, with the best result highlighted in bold and the runner-up underlined.

	Amazon		LastFM	
	ROC-AUC	MRR	ROC-AUC	MRR
GCN	$92.84_{\pm 0.34}$	$\underline{97.05_{\pm 0.12}}$	$59.17_{\pm 0.31}$	$79.38_{\pm 0.65}$
GAT	$91.65_{\pm 0.80}$	$96.58_{\pm 0.26}$	$58.56_{\pm 0.66}$	$77.04_{\pm 2.11}$
RGCN	$86.32_{\pm 0.28}$	$93.92_{\pm 0.16}$	$57.21_{\pm 0.09}$	$77.68_{\pm 0.17}$
GATNE	$77.39_{\pm 0.50}$	$92.04_{\pm 0.36}$	$66.87_{\pm 0.16}$	$85.93_{\pm 0.63}$
HetGNN	$77.74_{\pm 0.24}$	$91.79_{\pm 0.03}$	$62.09_{\pm 0.01}$	$83.56_{\pm 0.14}$
HGT	$88.26_{\pm 2.06}$	$93.87_{\pm 0.65}$	$54.99_{\pm 0.28}$	$74.96_{\pm 1.46}$
SeHGNN	$91.67_{\pm 0.94}$	$95.83_{\pm 0.58}$	$66.59_{\pm 0.62}$	$88.61_{\pm 1.25}$
SimpleHGN	$\underline{93.40_{\pm 0.62}}$	$96.94_{\pm 0.29}$	$\underline{67.59_{\pm 0.23}}$	$\underline{90.81_{\pm 0.32}}$
PSHGCN	$\mathbf{94.12_{\pm 0.58}}$	$\mathbf{97.93_{\pm 0.46}}$	$\mathbf{69.25_{\pm 0.63}}$	$\mathbf{91.19_{\pm 0.51}}$

- Node classification on ogbn-mag

(1.9M nodes and 21.1M edges)

Table 4: Node classification performance (Mean accuracies ± standard errors) on ogbn-mag, where the symbol "*" denotes the usage of extra embeddings and multi-stage training. The best results are highlighted in bold.

Methods	Validation accuracy	Test accuracy
RGCN	48.35±0.36	47.37±0.48
HGT	49.89±0.47	49.27±0.61
NARS	51.85±0.08	50.88±0.12
SAGN	52.25±0.30	51.17±0.32
GAMLP	53.23±0.41	51.63±0.22
SeHGNN	55.95±0.11	53.99±0.18
PSHGCN	**56.16±0.21**	**54.57±0.16**
SAGN*	55.91±0.17	54.40±0.15
GAMLP*	57.02±0.41	55.90± 0.27
SeHGNN*	59.17±0.09	57.19±0.12
PSHGCN*	**59.43±0.15**	**57.52±0.11**

He M, Wei Z, et al. Spectral Heterogeneous Graph Convolutions via Positive Noncommutative Polynomials. [TheWebConf'24] (Ours)

Outline

- Overview of GNNs
- Spectral interpretation of GNNs
- Our works (OptBasisGNN, PolyGCL, PSHGCN)
- Summary & Perspectives

Summary

- The theoretical foundation of GCN is the graph signal theory.
  - GCN is a fixed linear low-pass filter that is inapplicable to heterophilic graphs. FavardGNN can learn arbitrary filters, and OptBasisGNN achieves an optimal convergence rate.
  - Using polynomial filters with graph contrastive learning, PolyGCL can enhance performance on both homophilic and heterophilic graphs.
  - PSHGCN can learn arbitrary heterogeneous graph filters using positive noncommutative polynomials.

- Perspectives
  - Theoretical assumptions of graph machine learning.
  - Efficient computation of spectral-based GNN.

Team Members & Collaborators

Team Members


Zhe Yuan


Hanzhi Wang


Yanping Zheng

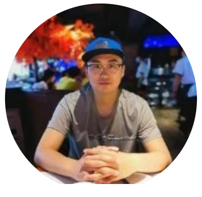

Mingguo He

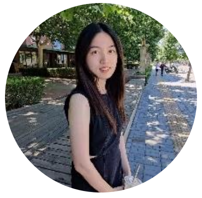

Jiajun Li


Tianjing Zeng


Fangrui Lv


Zhewei Wei


Gengmo Zhou


Yuhe Guo


Jinjia Feng


Yang Zhang


Ruoqi Zhang


Xu Liu


Mingji Yang


Haipeng Ding


Lu Yi


Runlin Lei


Guanyu Cui


Jingyu Chen


Jiahong Ma


Xiang Li

Collaborators


Sibo Wang


Jiajun Liu


Zengfeng Huang


Hongteng Xu


Bolin Ding


Yaliang Li


Zhen Wang

Thanks!

Q&A