SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS

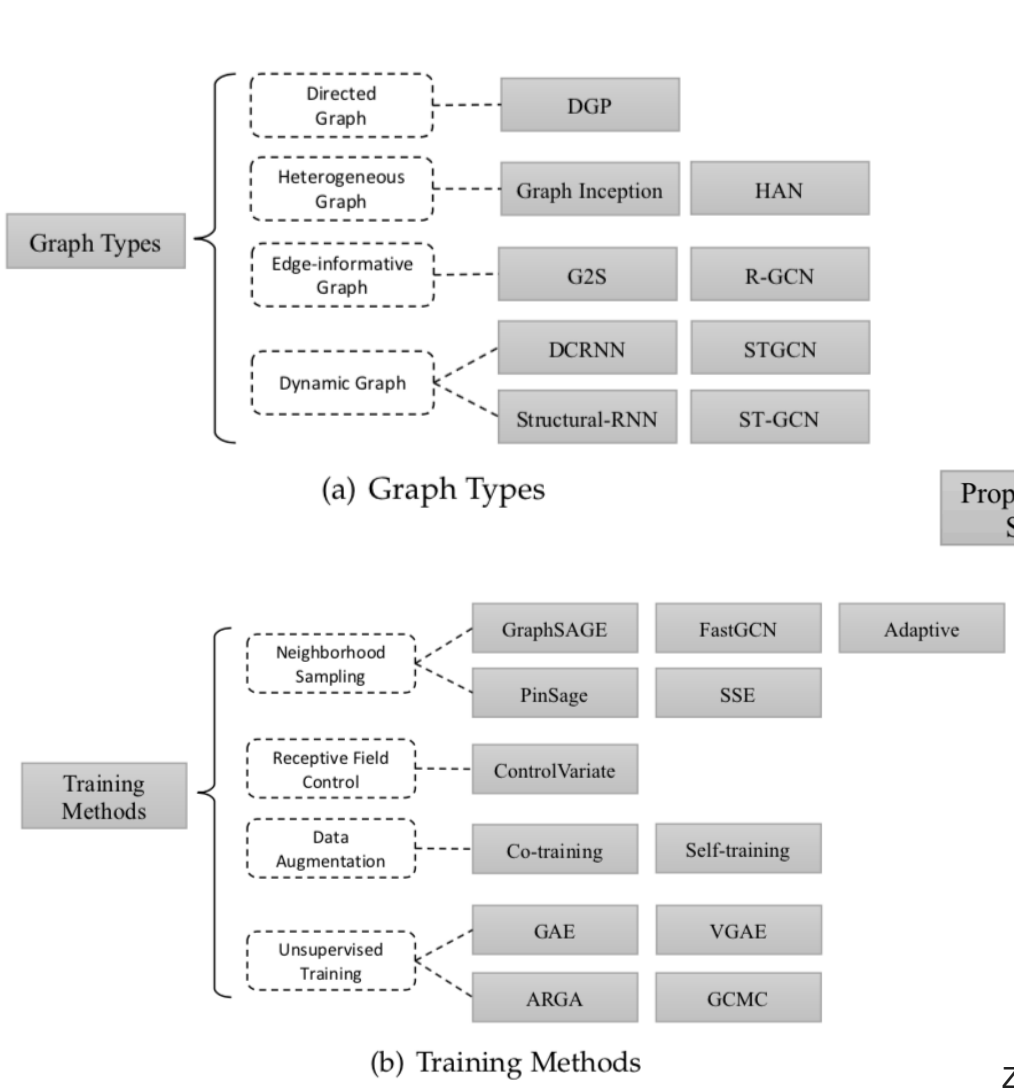
by

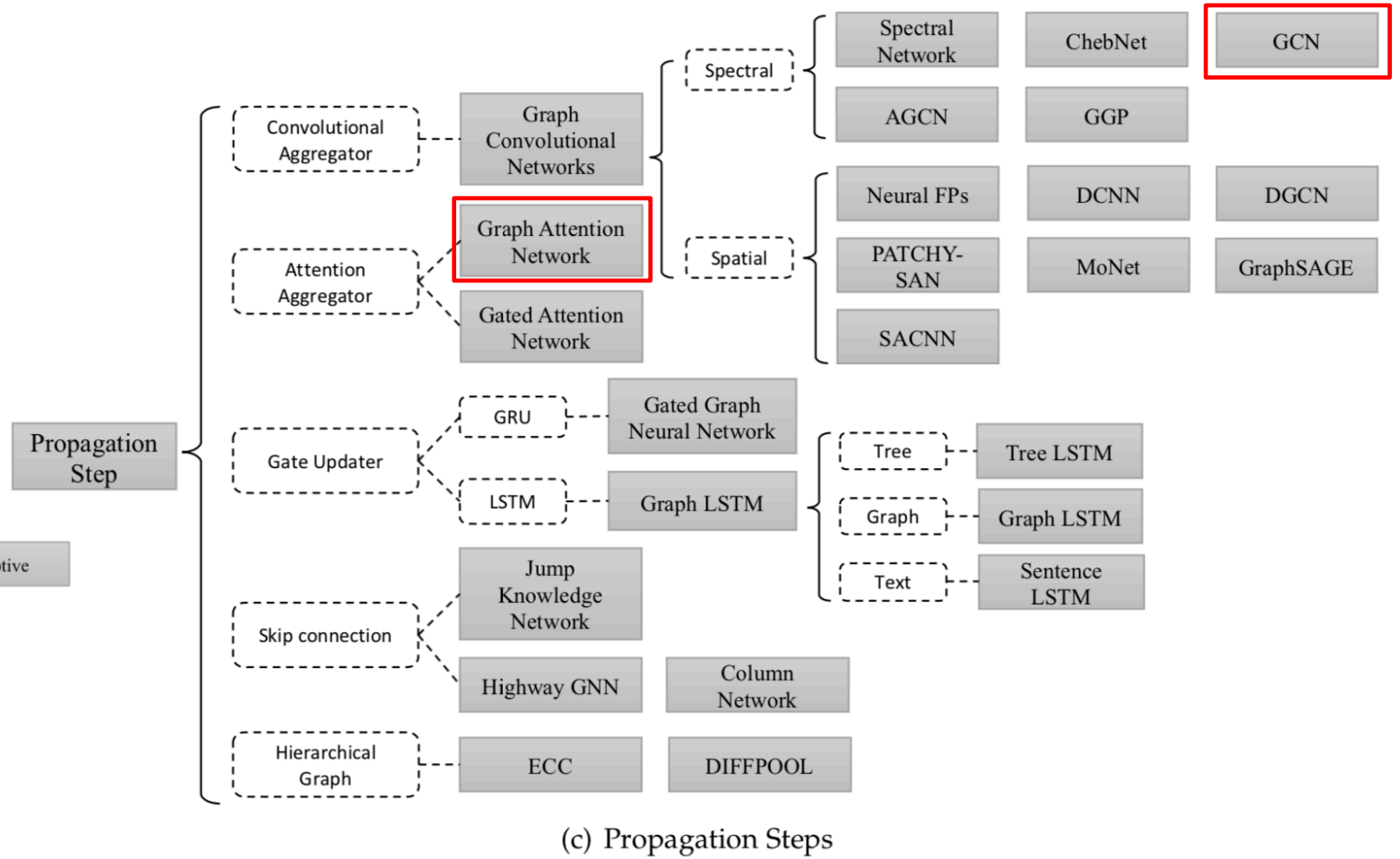
Thomas N. Kipf, Max Welling

presented by Sukwon Yun

Contents

1. Background





Zhou, Jie, et al. "Graph neural networks: A review of methods and applications." *arXiv preprint arXiv:1812.08434* (2018).

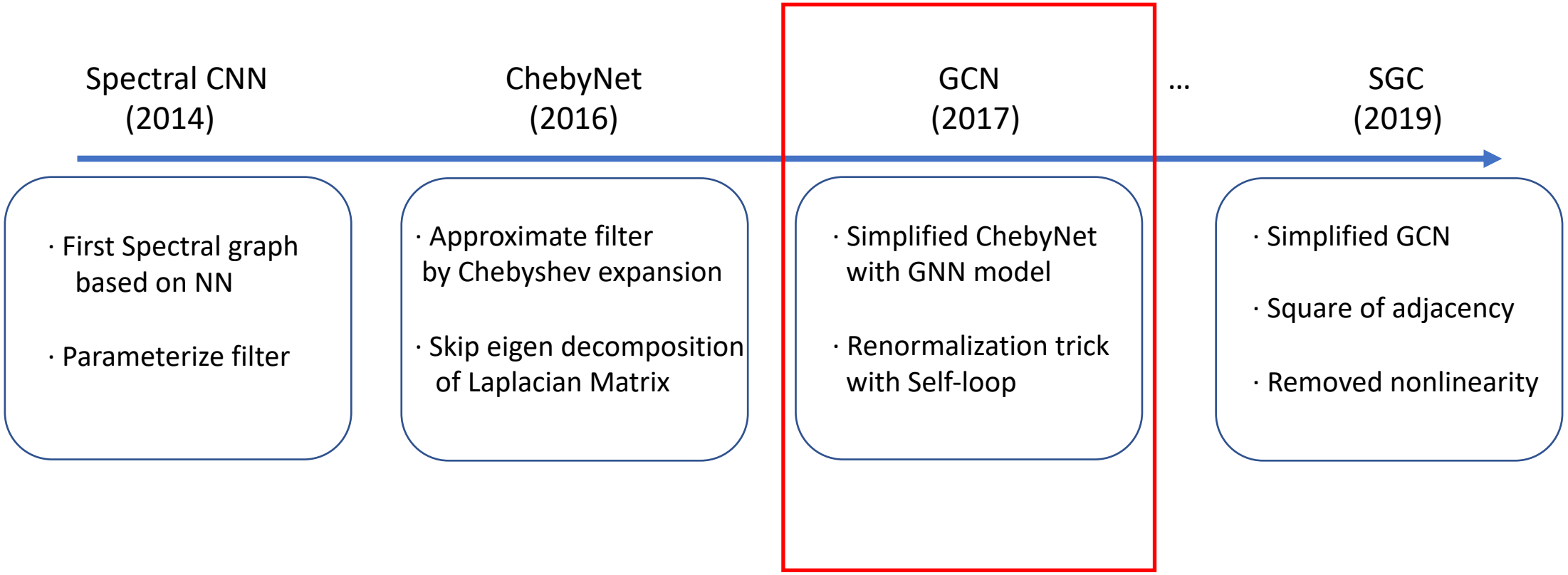
1. Background

Area	Application	Algorithm	Deep Learning Model	References
Text	Text classification	GCN	Graph Convolutional Network	[1], [23], [48] [2], [22], [46]
		GAT	Graph Attention Network	[68]
		DGCNN	Graph Convolutional Network	[106]
		Text GCN	Graph Convolutional Network	[107]
		Sentence LSTM	Graph LSTM	[62]
	Sequence Labeling (POS, NER)	Sentence LSTM	Graph LSTM	[62]
	Sentiment classification	Tree LSTM	Graph LSTM	[60]
	Semantic role labeling	Syntactic GCN	Graph Convolutional Network	[108]
	Neural machine translation	Syntactic GCN	Graph Convolutional Network	[109], [110]
		GGNN	Gated Graph Neural Network	[38]
	Relation extraction	Tree LSTM	Graph LSTM	[111]
		Graph LSTM	Graph LSTM	[44], [112]
		GCN	Graph Convolutional Network	[113]
	Event extraction	Syntactic GCN	Graph Convolutional Network	[114], [115]
	AMR to text generation	Sentence LSTM	Graph LSTM	[116]
		GGNN	Gated Graph Neural Network	[38]
	Multi-hop reading comprehension	Sentence LSTM	Graph LSTM	[117]
	Relational reasoning	RN	MLP	[96]
		Recurrent RN	Recurrent Neural Network	[118]
		IN	Graph Neural Network	[4]
Image	Social Relationship Understanding	GRM	Gated Graph Neural Network	[119]
	Image classification	GCN	Graph Convolutional Network	[120], [121]
		GGNN	Gated Graph Neural Network	[122]
		DGP	Graph Convolutional Network	[35]
		GSNN	Gated Graph Neural Network	[123]
	Visual Question Answering	GGNN	Gated Graph Neural Network	[119], [124], [125]
	Object Detection	RN	Graph Attention Network	[126], [127]
	Interaction Detection	GPNN	Graph Neural Network	[128]
		Structural-RNN	Graph Neural Network	[42]
	Region Classification	GCNN	Graph CNN	[129]
	Semantic Segmentation	Graph LSTM	Graph LSTM	[63], [130]
		GGNN	Gated Graph Neural Network	[131]
		DGCNN	Graph CNN	[132]
		3DGNN	Graph Neural Network	[133]
Science	Physics Systems	IN	Graph Neural Network	[4]
		VIN	Graph Neural Network	[91]
		GN	Graph Networks	[3]
	Molecular Fingerprints	NGF	Graph Convolutional Network	[51]
		GCN	Graph Convolutional Network	[99]
	Protein Interface Prediction	GCN	Graph Convolutional Network	[5]
	Side Effects Prediction	Decagon	Graph Convolutional Network	[134]
	Disease Classification	PPIN	Graph Convolutional Network	[135]
Knowledge Graph	KB Completion	GNN	Graph Neural Network	[6]
	KG Alignment	GCN	Graph Convolutional Network	[136]
Combinatorial Optimization		structure2vec	Graph Convolutional Network	[7]
		GNN	Graph Neural Network	[137]
		GCN	Graph Convolutional Network	[138]
		AM	Graph Attention Network	[139]
Graph Generation		NetGAN	Long short-term memory	[140]
		GraphRNN	Rucurrent Neural Network	[137]
		Regularizing VAE	Variational Autoencoder	[141]
		GCPN	Graph Convolutional Network	[142]
		MolGAN	Relational-GCN	[143]

Zhou, Jie, et al. "Graph neural networks: A review of methods and applications." *arXiv preprint arXiv:1812.08434* (2018).

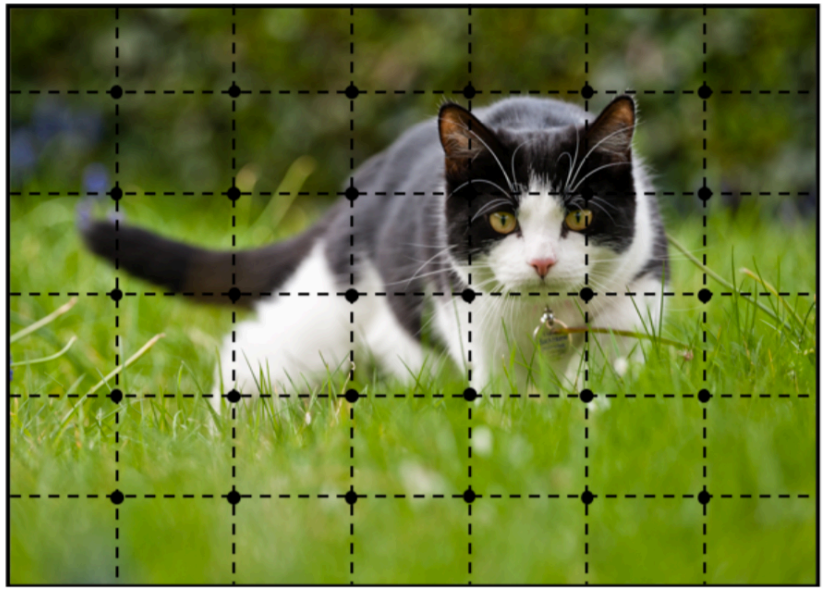
1. Background

- Flow of Spectral-based GNNs

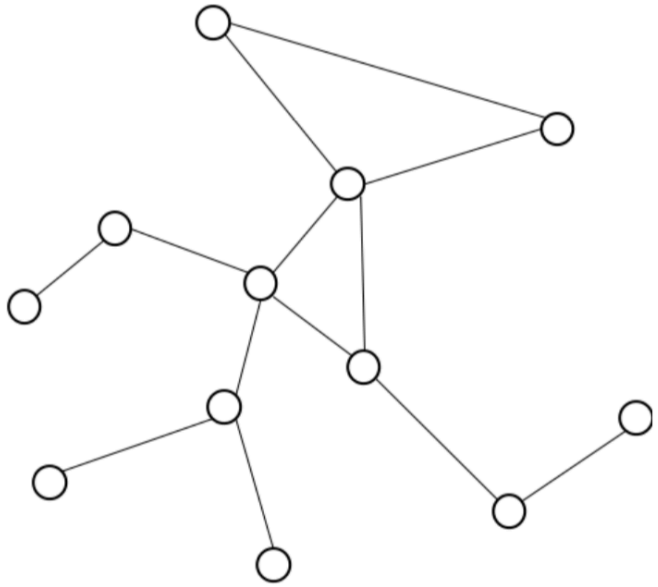


1. Background

- Euclidean vs Non-Euclidean



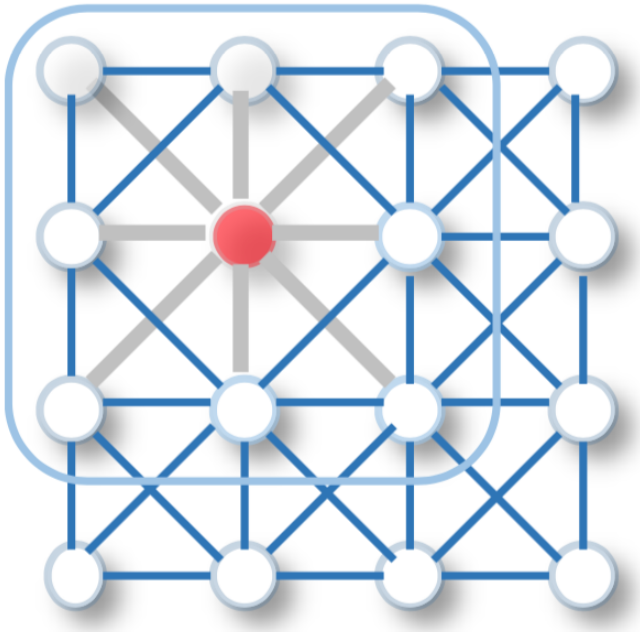
< Euclidean space >



< Non–Euclidean space >

1. Background

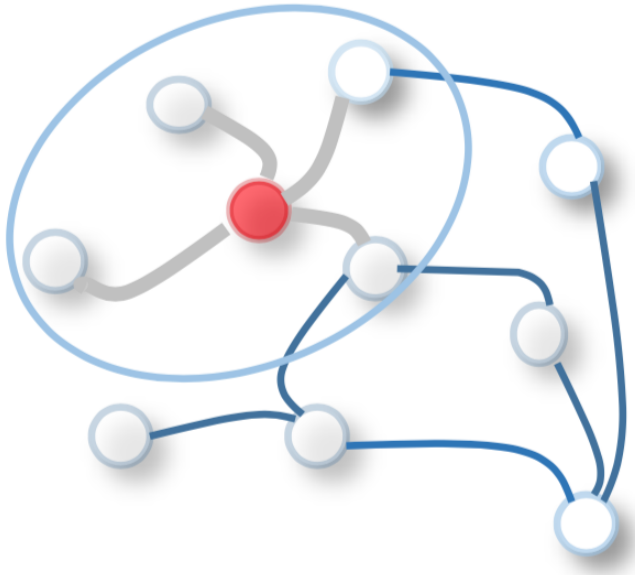
- CNN vs GCN



< 2D-Convolution >

- fixed filter (Weight sharing)
- Spatial
- Receptive field

≈

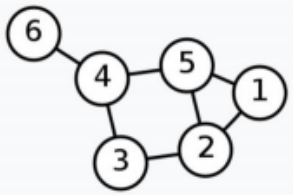


< Graph Convolution >

- Variable filter
- Spectral

1. Background

- Laplacian Matrix

	$\mathbf{D}$	$\mathbf{A}$
Labeled graph	**Degree matrix**	**Adjacency matrix**
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

Laplacian matrix

$$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$$

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \longrightarrow \text{Unnormalized Laplacian}$$

$$\begin{aligned} \mathbf{D}^{-\frac{1}{2}}\mathbf{L}\mathbf{D}^{-\frac{1}{2}} &= \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-\frac{1}{2}} \\ &= \mathbf{D}^{-\frac{1}{2}}(\mathbf{D}^{\frac{1}{2}} - \mathbf{A}\mathbf{D}^{-\frac{1}{2}}) \\ &= \mathbf{D}^{0} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \\ &= \boxed{\mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}} \longrightarrow \text{Normalized Laplacian} \end{aligned}$$

2. Introduction

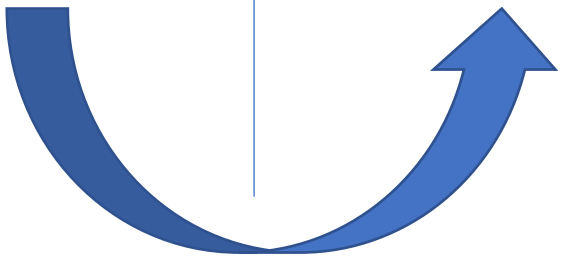
- Problem Definition

< Resource >

- Labeled / Unlabeled Node
- Feature / Edge of each Node

< Problem >

- To make inference on Unlabeled Data



Train by Supervised Loss

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_l} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

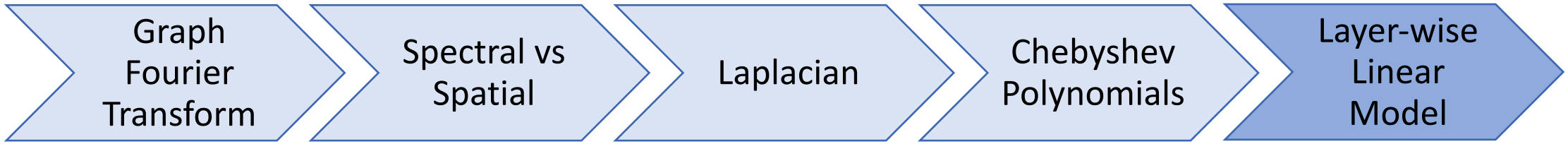
$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\ W^{(1)}\Big)$$

2. Introduction

$$Z = f(X, A) = softmax\left(\hat{A}\ ReLu\left(AXW^{(0)}\right)w^{(1)}\right)$$

2.1 Spectral Graph Convolutions

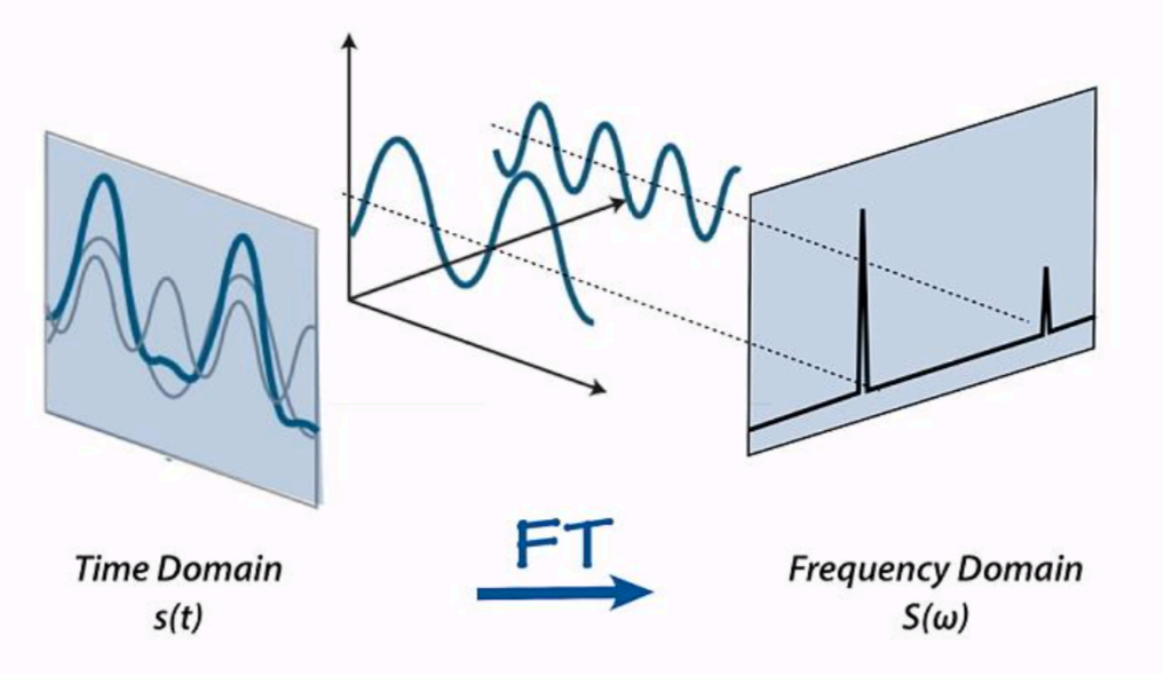
2.2 Layer-wise Linear Model



2. Introduction

- Graph Fourier Transform – Fourier Transform

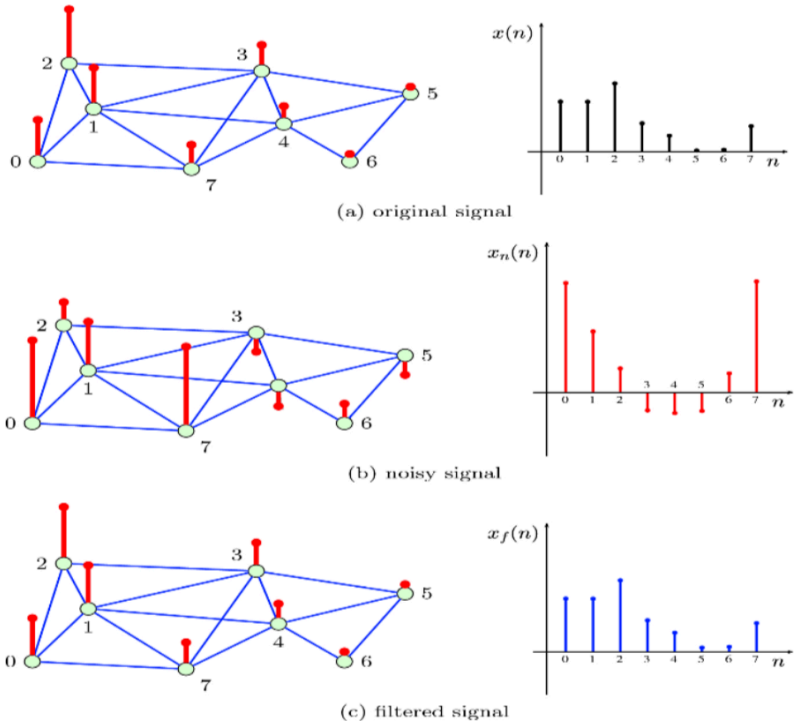
<General>



Signal → Frequency

Project on Fourier Basis

<Graph>



Graph Signal → Frequency

Node label

Difference on
Central, Neighbor Node

2. Introduction

- Graph Fourier Transform – Convolution Theorem

" *Convolution in one domain (Graph domain) equals point-wise multiplication in the other domain (Fourier domain)* "

$$\mathcal{F}(f \star g) = \mathcal{F}\{f\} \odot \mathcal{F}\{g\}$$

$$x *_G g = \mathcal{F}^{-1}(\mathcal{F}(x) \odot \mathcal{F}(g))$$

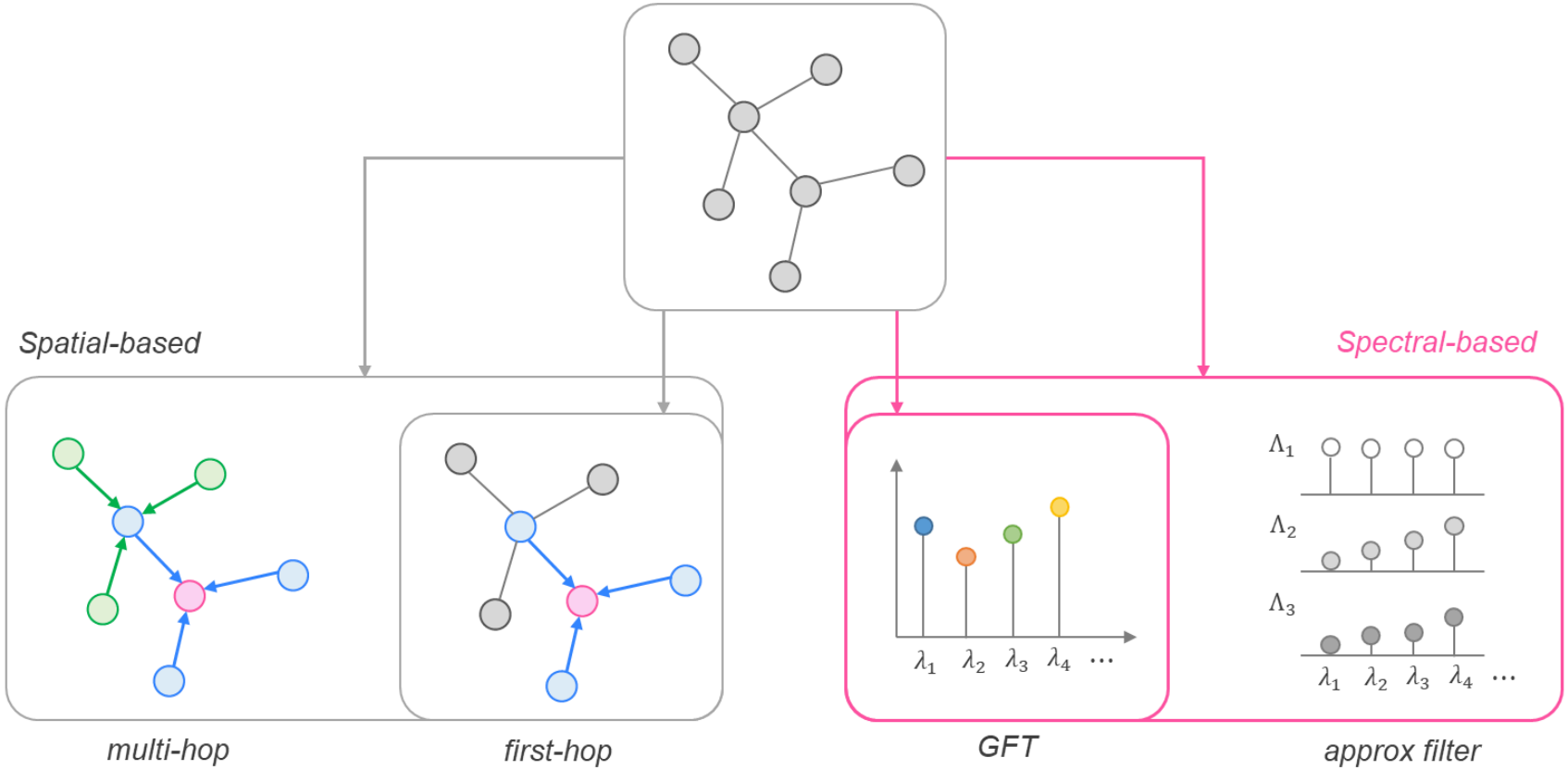
< 2D-Convolution >

< Graph Convolution >

" Fourier Transform enables Convolution "

2. Introduction

- Spatial vs Spectral



reflect neighbor nodes

GFT based, eigen-decomposition

2. Introduction

- Laplacian

The classical Fourier transform

$$\hat{f}(\xi) := \langle f, e^{2\pi i \xi t} \rangle = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

is the expansion of a function f in terms of the complex exponentials, which are the eigenfunctions of the one-dimensional Laplace operator [1]:

$$-\Delta(e^{2\pi i \xi t}) = -\frac{\partial^2}{\partial t^2} e^{2\pi i \xi t} = (2\pi\xi)^2 e^{2\pi i \xi t}.$$

$$g_\theta \star x = Lx = U g_\theta^* U^T x$$

Laplacian Matrix – Eigen Decomposition

U → *Fourier Basis*

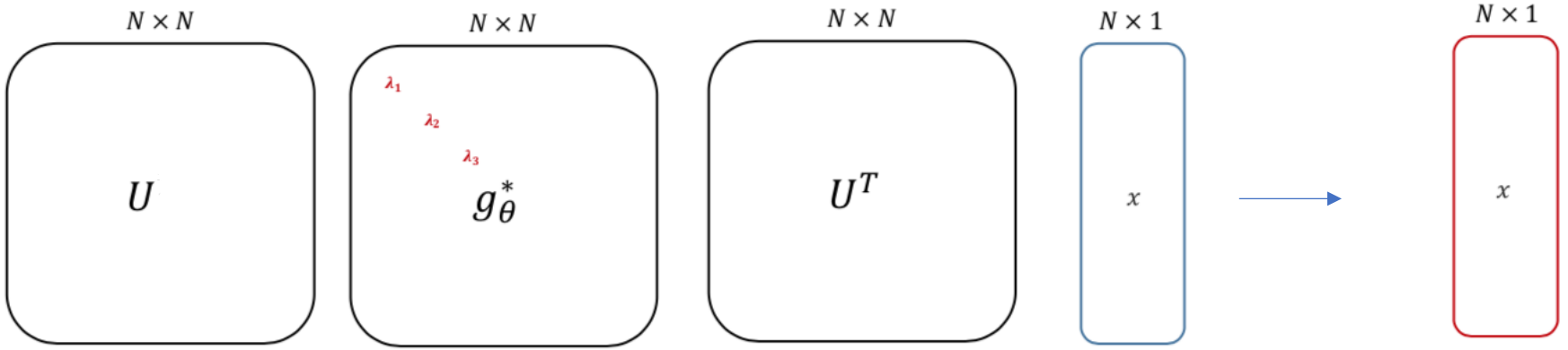
2. Introduction

Multiplying Laplacian in feature vector = Fourier & Inverse Fourier Transform

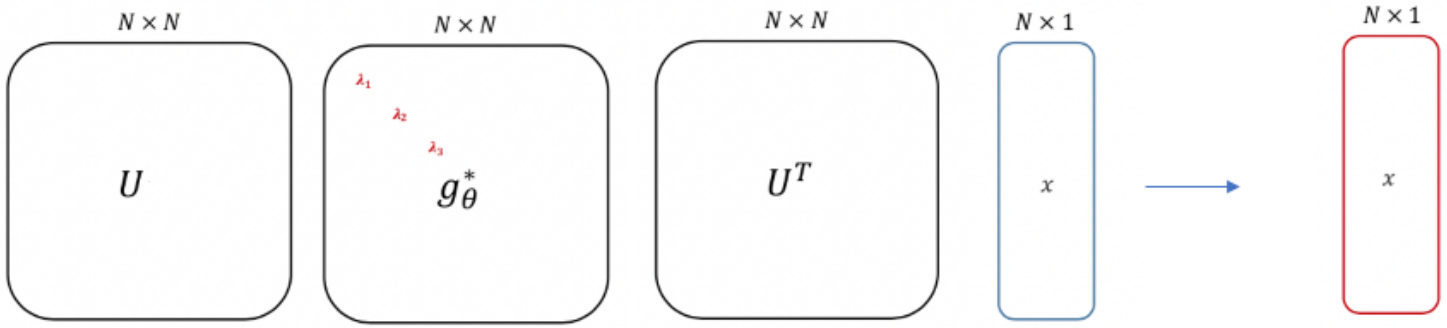
x → $U^T x$ → $(g_\theta^*) U^T x$ → $(U) g_\theta^* U^T x$

Laplacian

Signal — Graph Fourier Transform — Filtering $g_\theta^* = g_\theta^*(\Lambda)$ — Inverse Graph Fourier Transform



2. Introduction

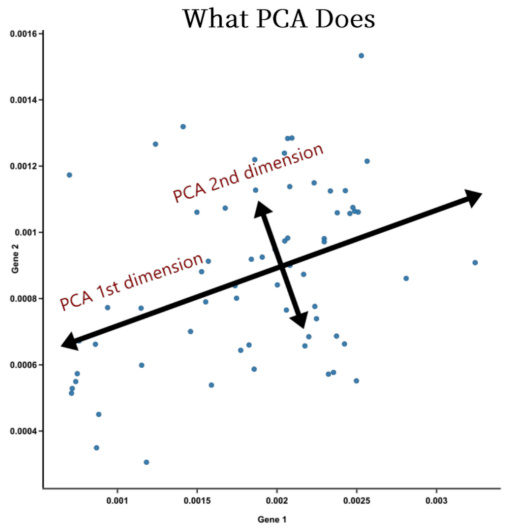


" Convolution in one domain (Graph domain) equals point-wise multiplication in the other domain (Fourier domain) "

Applying Laplacian Matrix means Convolution

Principal Component Analysis (PCA) : Maximize variance

Graph Fourier Transform (GFT) : Minimize Frequency



2. Introduction

- Chebyshev Polynomials

$$g'_\theta(\Lambda) = \sum_{k=0}^{K} \theta'_k \Lambda^k = \theta'_0\Lambda^0 + \theta'_1\Lambda^1 + \theta'_2\Lambda^2 + \cdots + \theta'_K\Lambda^K \sim \text{Polynomial}$$

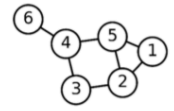
$$g_\theta \star x = U g_\theta U^\top x$$

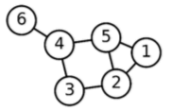
$$U\Lambda U^T U\Lambda U^T = U\Lambda^2 U^T$$

$$= U\left(\sum_{k=0}^{K-1} \theta_k \Lambda^k\right) U^T x = \sum_{k=0}^{K-1} \theta_k U\Lambda^k U^T x = \sum_{k=0}^{K-1} \theta_k L^k x$$

$\theta_0 L^0 x$ $\theta_1 L^1 x$ $\theta_2 L^2 x$

$= \Theta_0 L^0$ + $\Theta_1 *$ L^1 (1-hop neighborhood) + $\Theta_2 *$ L^2 (2-hop neighborhood)





2. Introduction

$$g_\theta \star x = U g_\theta U^\top x$$

$$g_\theta \star x = \sum_{k=0}^{K-1} \theta_k L^k x = \theta_0 L^0 x + \theta_1 L^1 x + \cdots \theta_{K-1} L^{K-1} x$$

But, Computationally Expensive

<Chebyshev Polynomial>

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$$

$$T_0(x) = 1 \qquad T_1(x) = x$$

$$g'_\theta \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L}) x = \theta'_0 T_0(\tilde{\mathrm{L}}) x + \theta'_1 T_1(\tilde{\mathrm{L}}) x + \theta'_2 T_2(\tilde{\mathrm{L}}) x + \cdots + \theta'_K T_K(\tilde{\mathrm{L}}) x$$

2. Introduction

$$g'_\theta \star x \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})x = \theta'_0 T_0(\tilde{\mathrm{L}})x + \theta'_1 T_1(\tilde{\mathrm{L}})x + \theta'_2 T_2(\tilde{\mathrm{L}})x + \cdots + \theta'_K T_K(\tilde{\mathrm{L}})x$$

$$g'_\theta \star x \approx \theta'_0 T_0(\tilde{\mathrm{L}})x + \theta'_1 T_1(\tilde{\mathrm{L}})x \quad (\mathrm{K} = 1)$$

$$\approx \theta'_0 \cdot 1 \cdot x + \theta'_1 (L - I_N)x \quad \leftarrow \quad \tilde{\mathrm{L}} = \frac{2}{\lambda_{max}}\mathrm{L} - I_N, where\ \lambda_{max} = 2$$

$$\approx \theta'_0 \cdot 1 \cdot x - \theta'_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x \quad \leftarrow \quad L = I_N - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

$$\approx \theta\left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right)x \quad \leftarrow \quad \theta = \theta'_0 = -\theta'_1$$

$$Z = \tilde{D}^{-\frac{1}{2}} \tilde{A}\, \tilde{D}^{-\frac{1}{2}} X\, \Theta$$

Renormalization trick $\leftarrow$

$$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

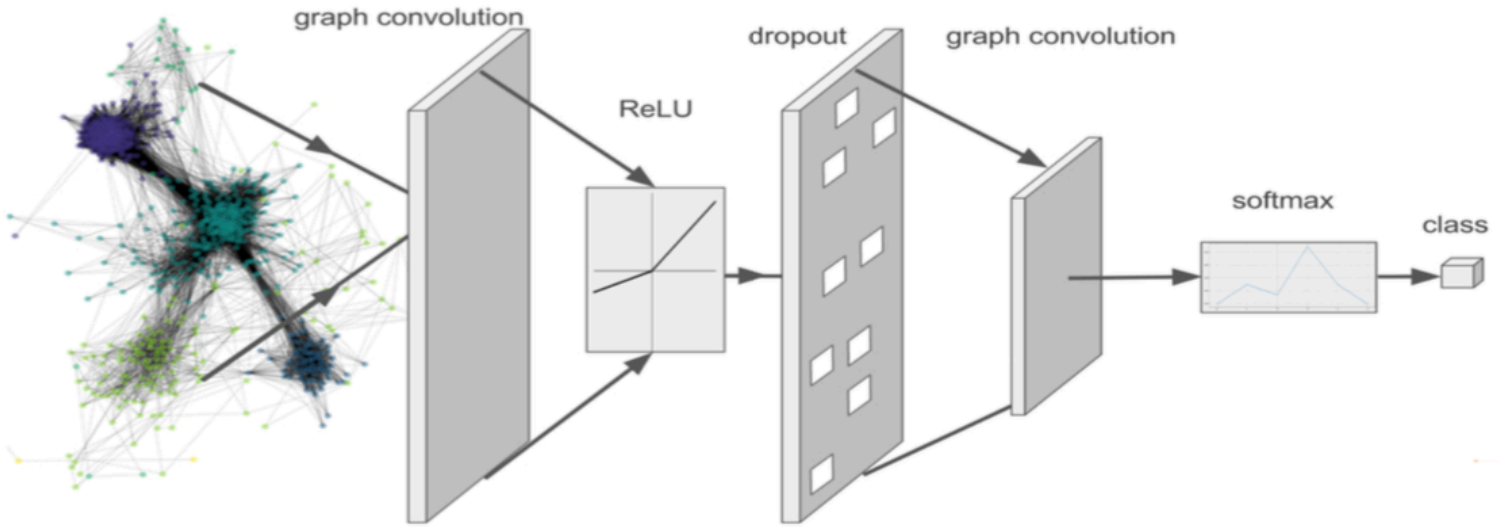
$$\tilde{A} = A + I_N \qquad \tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$$

2. Introduction

- Layer-wise Linear Model

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \leftarrow Z = \tilde{D}^{-\frac{1}{2}}\tilde{A}\,\tilde{D}^{-\frac{1}{2}}X\,\Theta$$

$$Z = f(X, A) = \mathrm{softmax}\left(\hat{A}\ \mathrm{ReLU}\left(\hat{A}XW^{(0)}\right)W^{(1)}\right) \leftarrow \hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$$



2. Introduction

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\ W^{(1)}\Big)$$

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{reg} \longrightarrow \mathcal{L} = -\sum_{l \in \mathcal{Y}_l} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

< Resource >

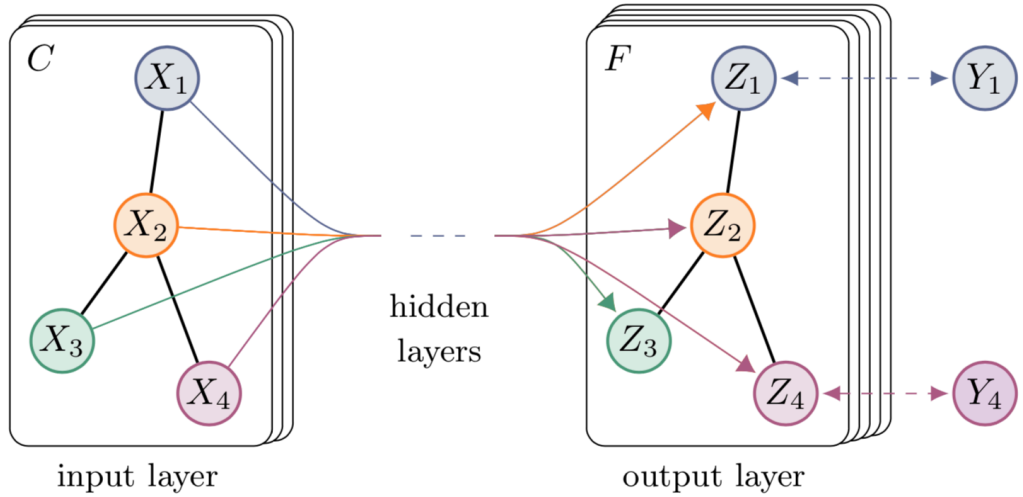
- Labeled / Unlabeled Node
- Feature / Edge of each Node

< Problem >

- To make inference on Unlabeled Data

Train by Supervised Loss

< Problem Definition >



3. Results

- Dataset

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

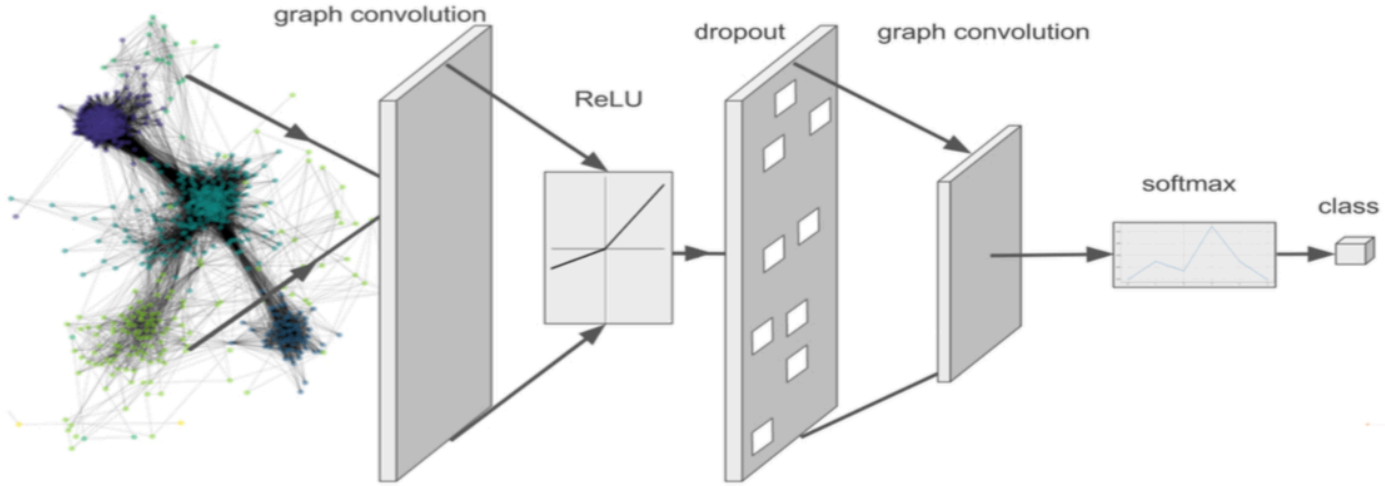
3. Results

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

Description		Propagation model	Citeseer	Cora	Pubmed
Chebyshev filter (Eq. 5)	$K = 3$	$\sum_{k=0}^{K} T_k(\tilde{L})X\Theta_k$	69.8	79.5	74.4
	$K = 2$		69.6	81.2	73.8
1st-order model (Eq. 6)		$X\Theta_0 + D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta_1$	68.3	80.0	77.5
Single parameter (Eq. 7)		$(I_N + D^{-\frac{1}{2}}AD^{-\frac{1}{2}})X\Theta$	69.3	79.2	77.4
Renormalization trick (Eq. 8)		$\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\Theta$	**70.3**	**81.5**	**79.0**
1st-order term only		$D^{-\frac{1}{2}}AD^{-\frac{1}{2}}X\Theta$	68.7	80.5	77.8
Multi-layer perceptron		$X\Theta$	46.5	55.1	71.4

4. Implementation

$$Z = f(X, A) = softmax\left(\hat{A}\, ReLu\left(AXW^{(0)}\right)w^{(1)}\right)$$

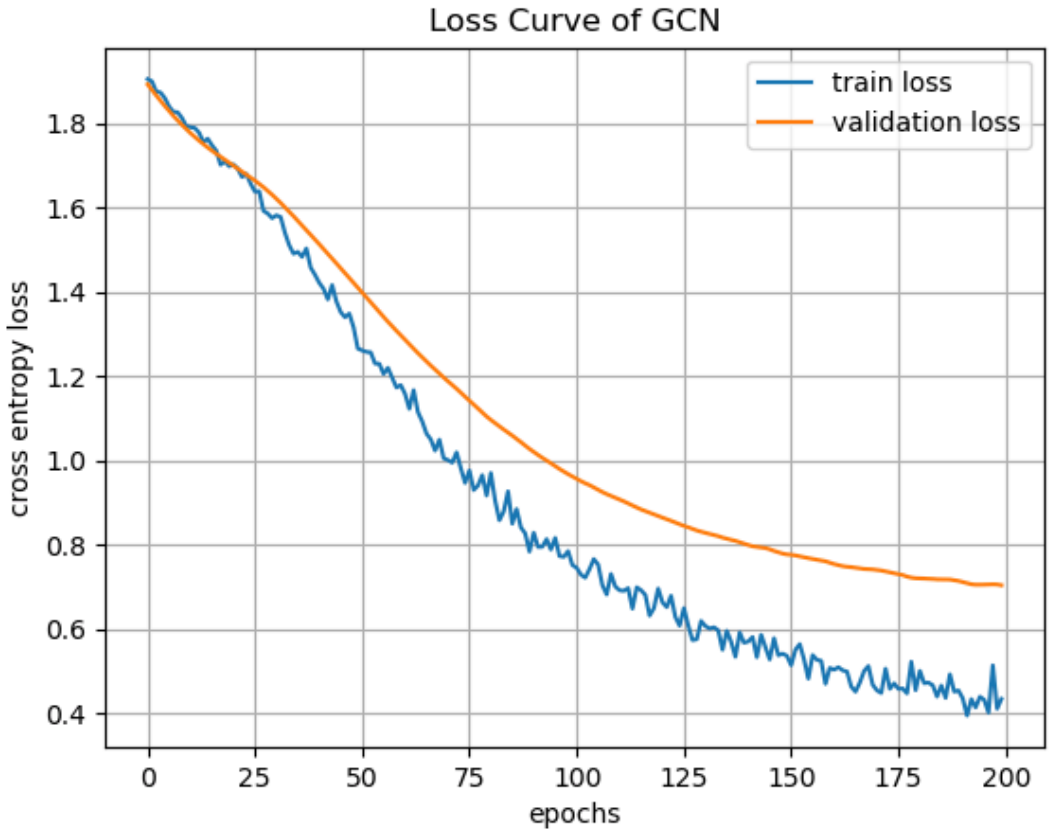


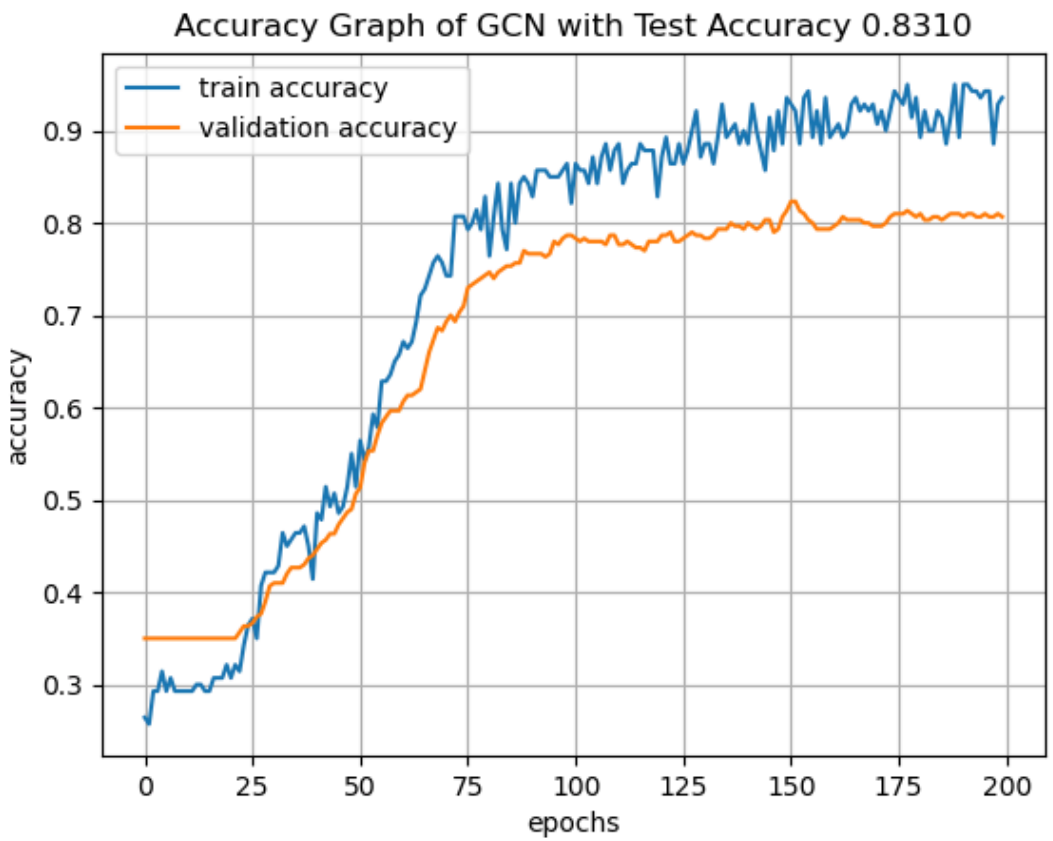

```python
class GCN(nn.Module):
    def __init__(self, n_feat, n_hidden, n_class, dropout):
        super(GCN, self).__init__()

        self.gc1 = GraphConvolution(n_feat, n_hidden)
        self.gc2 = GraphConvolution(n_hidden, n_class)
        self.dropout = dropout

    def forward(self, x, adj):
        x = F.relu(self.gc1(x, adj))
        x = F.dropout(x, self.dropout, training = self.training)
        x = self.gc2(x,adj)
        return F.log_softmax(x, dim=1)
```

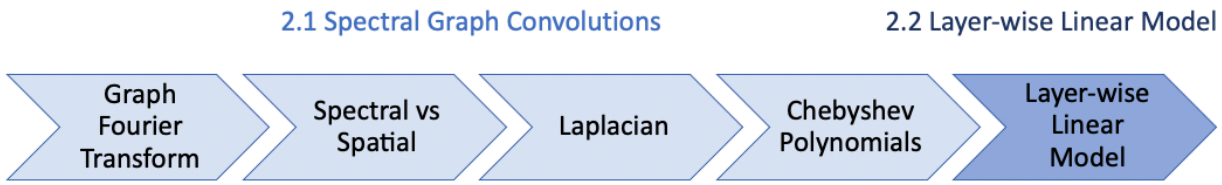
4. Implementation

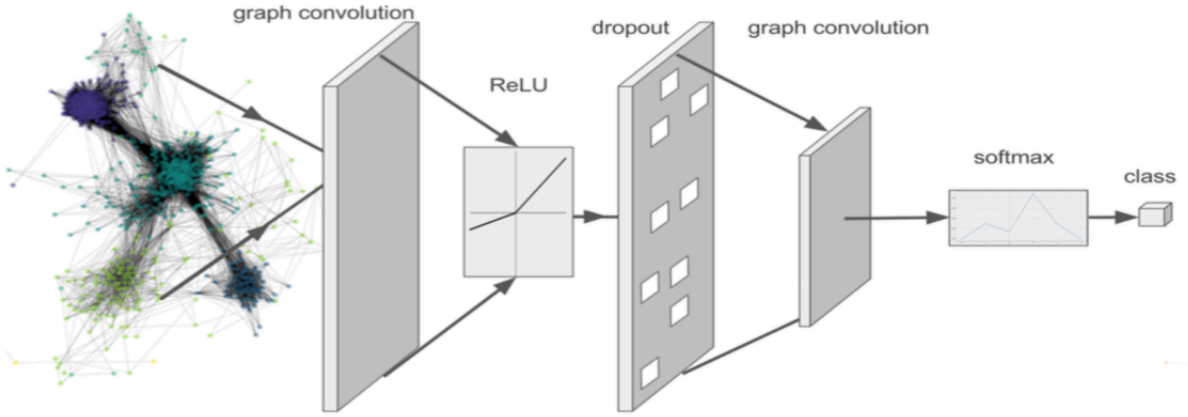




Time per epoch: 0.011s

5. Key takeaways





- Fourier Transform enabled Convolution
- Layer wise propagation based on first-order approximation
- Semi-supervised Loss term without regularization term

5. Discussion

- (-) Full Batch Gradient Descent -> Mini-batch Gradient Descent
- (-) Undirected -> Directed Edges and Edges Features
- (-) Same weight on Self Connection & Neighborhood Edge
- (-) More layers of convolutions
- (-) Convolution without Fourier Transform
- (-) Generalization of new graph

References

(1) Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016)

(2) Zhou, Jie, et al. "Graph neural networks: A review of methods and applications." arXiv preprint arXiv:1812.08434 (2018).

(3) Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." IEEE transactions on neural networks and learning systems (2020).

(4) Hoon Sang Yoon, 20210122_Graph_Overview, DSBA, School of Industrial Management Engineering, Korea University (2021)

(5) [Paper Review] Semi-supervised Classification with Graph Convolutional Networks, https://www.youtube.com/watch?v=F-JPKccMP7k&t=1710s

(6) [#30.Lec] Basic of Graph Convolution Network - 딥러닝 홀로서기, https://www.youtube.com/watch?v=YL1jGgcY78U&t=2251s