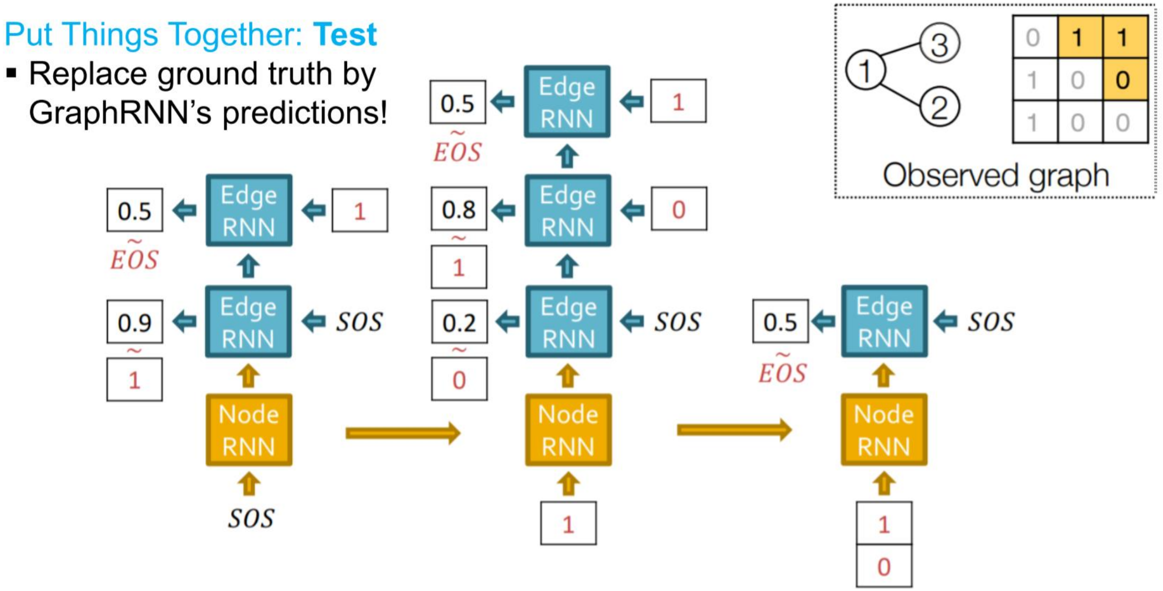
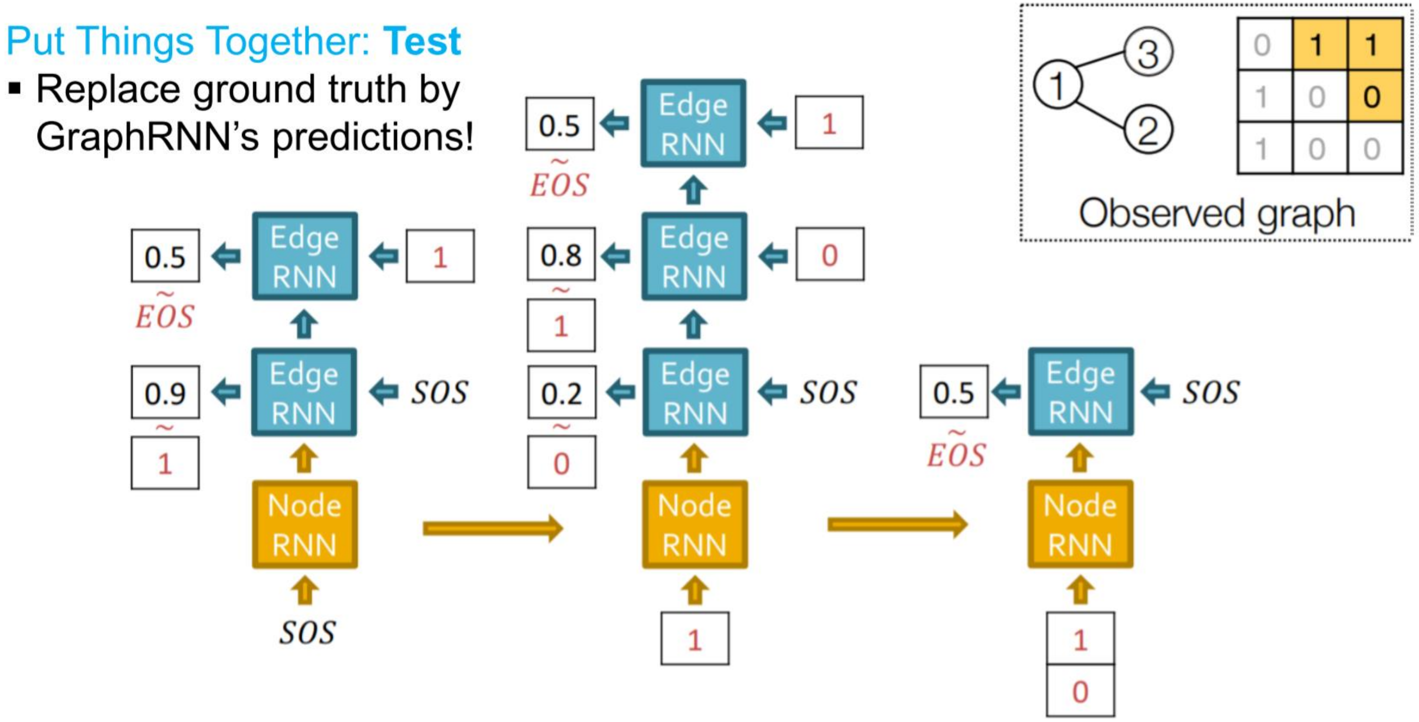
Summary Questions of the lecture

- What is the meaning of the output of each RNN cell in **GraphRNN**?
- The node-RNN at time-step t outputs the initial state of the edge-RNN. The edge-RNN sequentially outputs the probability that the current node t is connected to each existing node. The edge connectivity (1 for connected, 0 for not connected) is determined by sampling from the probability outputted by the edge-RNN. The edge-RNN stops when it outputs the end token EOS.



Summary Questions of the lecture

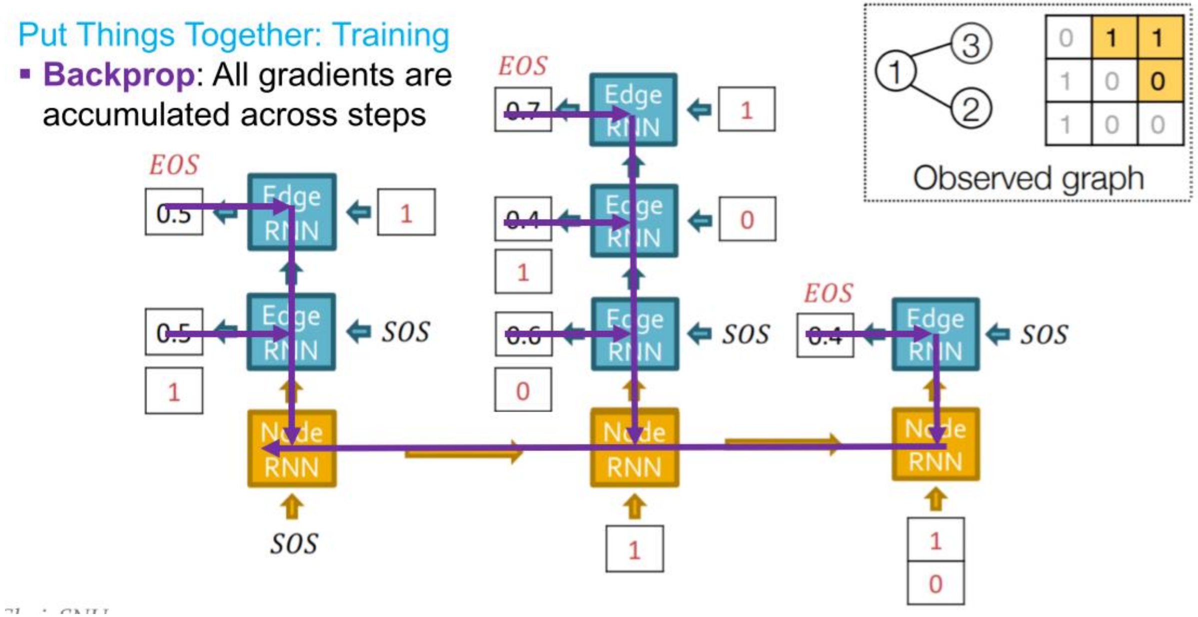
- How can we obtain the input of each RNN cell in **GraphRNN**?
- The first input of any RNN is the start token SOS. After that, the edge-RNN receives the binary output of the previous cell, and the node-RNN receives all of the outputs of the previous edge-RNN sequence as a vector.



Summary Questions of the lecture

- Explain the training method of **GraphRNN** in the view point of loss and training path.
- During training, both RNNs receive the ground truth as input, regardless of the output of the previous cell. Output probabilities of edge-RNNs are learned in the supervised manner with the binary cross-entropy loss. Since RNNs weights are shared, the gradients w.r.t. the weights are accumulated across time-steps.

$$L = -\sum_i [y_i^* \log y_i + (1 - y_i^*) \log(1 - y_i)]$$



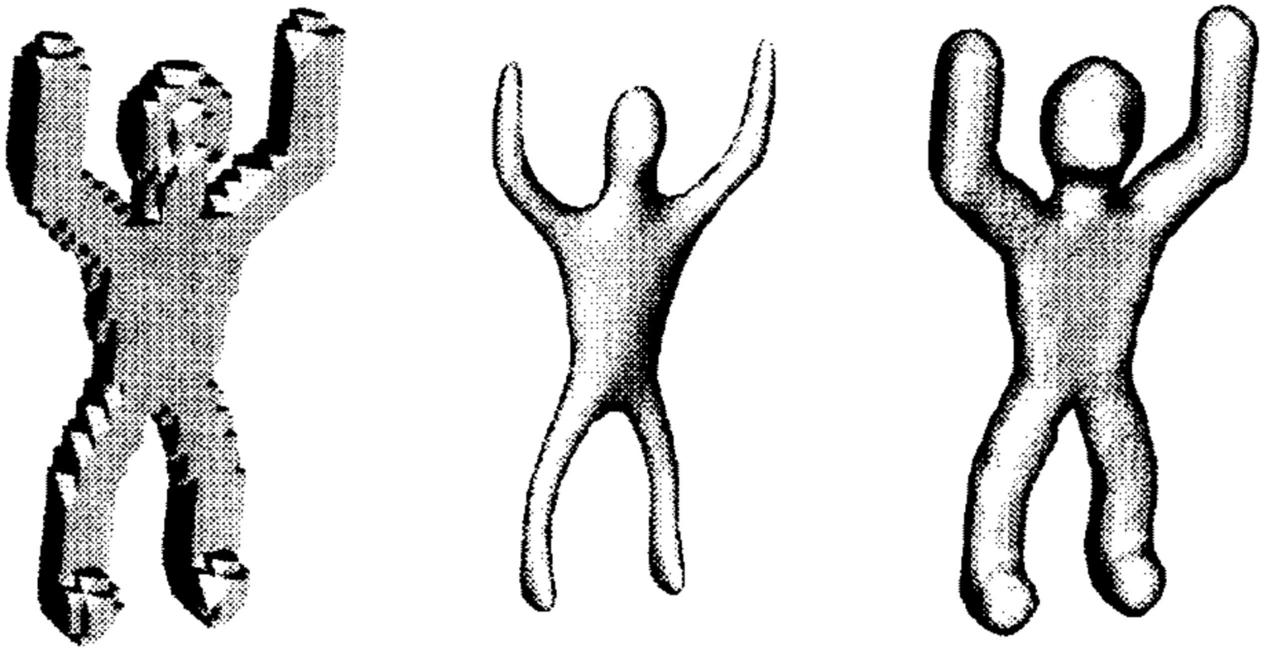
Outline of Lecture (5)

- Random Walks and Diffusion
- Diffusion in GCN
  - Propagation using graph diffusion
    - APPNP: Predict Then Propagate [ICLR'19]
    - Graph Diffusion-Embedding Networks [CVPR'19]
  - Making a new graph
    - Diffusion Improves Graph Learning [NIPS'19]
    - SSL with Graph Learning-Convolutional Networks [CVPR'19]

- Deep Generative Models For Graph
  - Problem of Graph Generation
  - ML Basics for Graph Generation
  - GraphRNN : Generating Realistic Graphs
  - Applications and Open Questions
- Tacking Oversmoothing
  - Oversmoothing in GCN
  - Taubin smoothing
  - Jumping knowledge networks
  - ResNet, DenseNet, and Dilated convolution
  - PairNorm; normalization layer for GNNs

Tackling Over-smoothing

Over-smoothing washes away graph signal on each node ….



Oversmoothing

Kipf et al. (ICLR 2017) SSL with GCN

Li et al., (AAAI 2018), Deeper Insights into Graph Convolutional Networks …, 최홍석 발표

Taubin (ICCV 1995) Taubin smoothing,

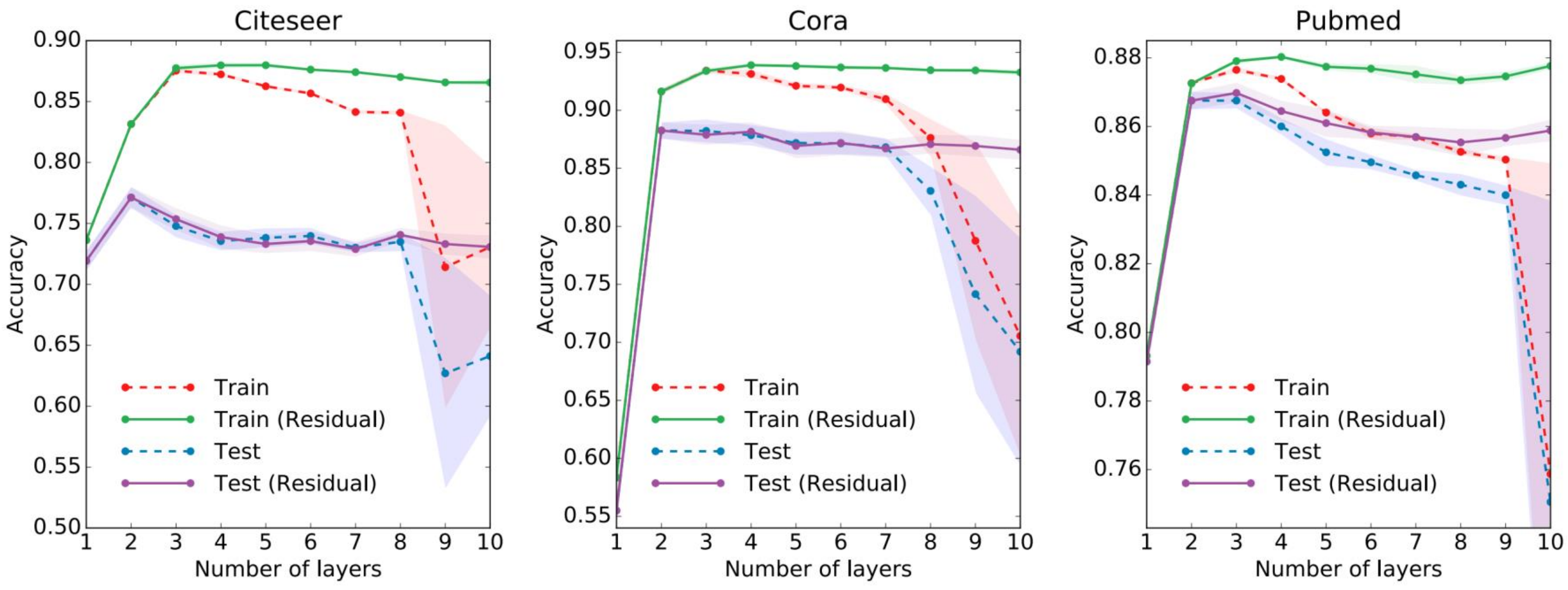
Xu et al. (ICML 2018) uses jumping knowledge networks

Li et al. (ICCV 2019) borrows the concept of ResNet, DenseNet, and Dilated convolution.

Zhao et al. (ICLR 2020) proposes PairNorm, the first normalization layer for GNNs.

Oversmoothing

- When the layers are deeper, the performances are degenerate harshly.
- The main cause of this phenomenon is over smoothing effects on GNNs.
- What is and how to circumvent over smoothing effects?

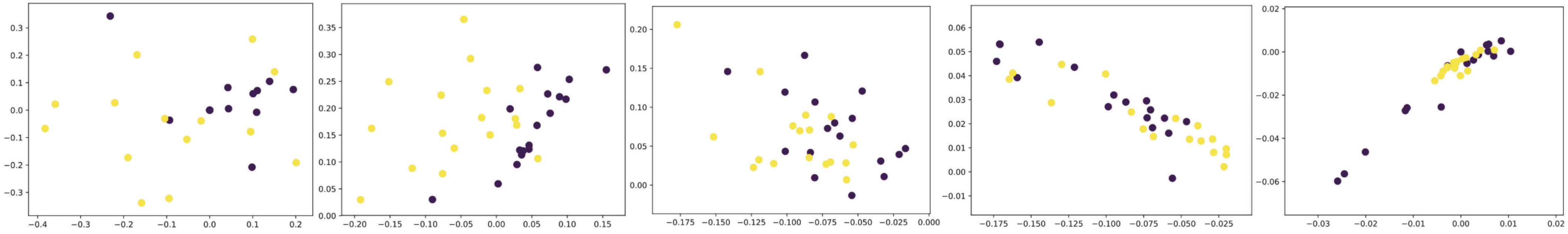


Overfitting?

Kipf et al. (ICLR 2017) Simplified ShevNet (GCN)

Oversmoothing

- When GCN goes deep, the performance can suffer from over smoothing where node representations from different clusters become mixed up.



(a) 1-layer (b) 2-layer (c) 3-layer (d) 4-layer (e) 5-layer

Li et al. (AAAI 2018)

Oversmoothing

- Li et al. (AAAI 2018) shows that GCN is a special form of Laplacian smoothing.
- They proved that oversmoothing washes away graph signal on each node.
- This means oversmoothing makes the node indistinguishable.

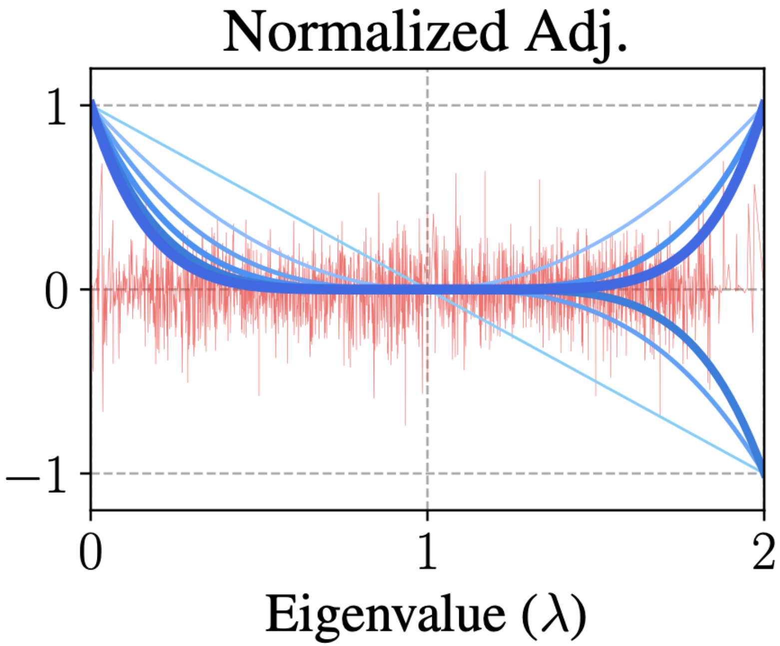
Theorem 1. *If a graph has no bipartite components, then for any* $\mathbf{w} \in \mathbb{R}^n$, *and* $\alpha \in (0,1]$,

$$\lim_{m \to +\infty} (I - \alpha L_{rw})^m \mathbf{w} = [\mathbf{1}^{(1)}, \mathbf{1}^{(2)}, \ldots, \mathbf{1}^{(k)}]\theta_1,$$

$$\lim_{m \to +\infty} (I - \alpha L_{sym})^m \mathbf{w} = D^{-\frac{1}{2}}[\mathbf{1}^{(1)}, \mathbf{1}^{(2)}, \ldots, \mathbf{1}^{(k)}]\theta_2,$$

where $\theta_1 \in \mathbb{R}^k, \theta_2 \in \mathbb{R}^k$, *i.e., they converge to a linear combination of* $\{\mathbf{1}^{(i)}\}_{i=1}^k$ *and* $\{D^{-\frac{1}{2}}\mathbf{1}^{(i)}\}_{i=1}^k$ *respectively.*

Laplacian smoothing



$\gamma = 1$

$$L = I - D^{-1/2} A D^{-1/2} = U \Lambda U^T$$

$$U = [u_1, u_2, \ldots, u_N]$$

$$\Lambda = diag[\lambda_1, \lambda_2, \ldots, \lambda_N]$$

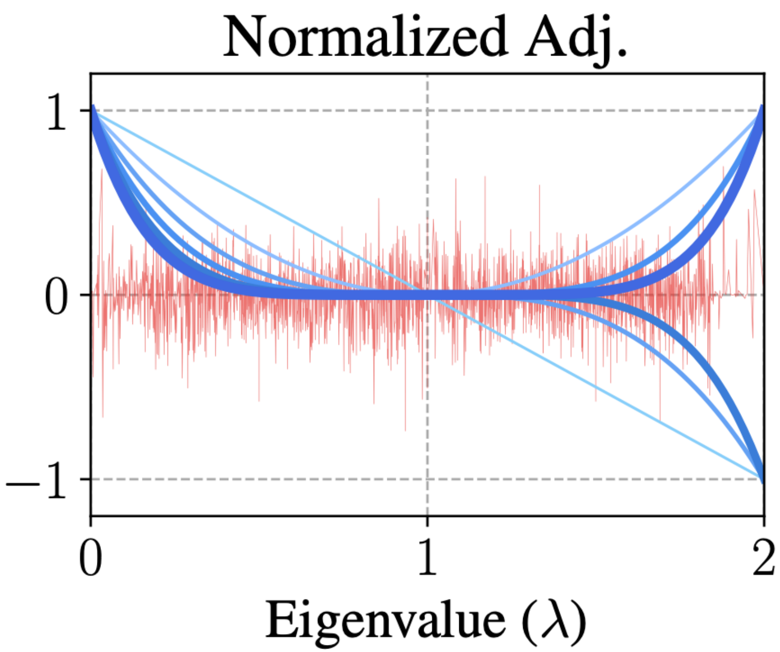
$$(0 = \lambda_1 \le \lambda_2 \le \ldots \le \lambda_N \le 2)$$

$$x_i' = (I - \gamma L) x_i = f(L) x_i$$

$$h_i = (I - \gamma L)^k x_i = f(L)^k x_i$$

$0 < \gamma < 1$ is scaling factor which controls the speed of the diffusion process.

Laplacian smoothing



$\gamma = 1$

$L = I - D^{-1/2} A D^{-1/2} = U \Lambda U^T$

$U = [u_1, u_2, \ldots, u_N]$

$\Lambda = diag[\lambda_1, \lambda_2, \ldots, \lambda_N]$

$(0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_N \leq 2)$

$h_i = (I - \gamma L)^k x_i = f(L)^k x_i$

$f(\lambda_i)^k = (1 - \gamma \lambda_i)^k$

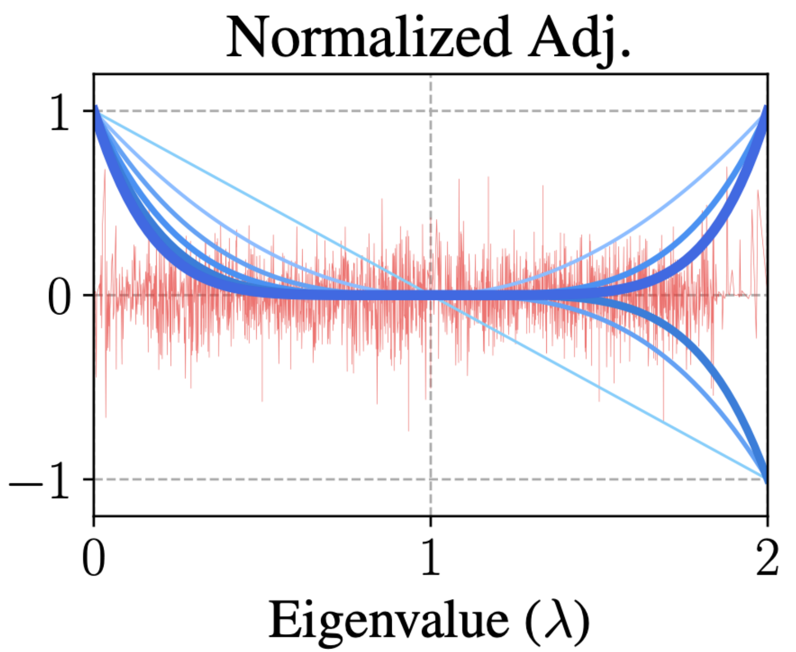
For every $\lambda \in (0,2]$,

since $|1 - \gamma \lambda_i| < 1$ for $0 < \gamma < 1$

we have $(1 - \gamma \lambda_i)^k \to 0$ when $k \to \infty$

except $f(0) = 1$

Laplacian smoothing



$\gamma = 1$

we have $(1-\gamma\lambda_i)^k \to 0$ when $k \to \infty$

except $f(0) = 1$

- This means that all the frequency components, other than the zero frequency component, are attenuated for large k.
- The eigenvector of zero frequency component is one vector, $(1,1,\ldots,1)^T$
- After lots of iteration the zero frequency component is preserved and the value of this is independent of the feature values!

Taubin smoothing (ICCV 1995)

- Taubin proposed second degree transfer function to solve the problem of shrinkage.

$$f(\lambda_i) = (1 - \gamma\lambda_i)(1 - \mu\lambda_i)$$

- Taubin smoothing can be interpreted as two consecutive steps of Laplacian smoothing with different scaling factors.

1. $\gamma > 0$: Laplacian smoothing step with positive scale factor (shrinking step)
2. $\mu < -\gamma < 0$: Laplacian smoothing step with negative scale factor (unshrinking step, Laplacian sharpening, high frequency amplification)

Taubin smoothing (ICCV 1995)

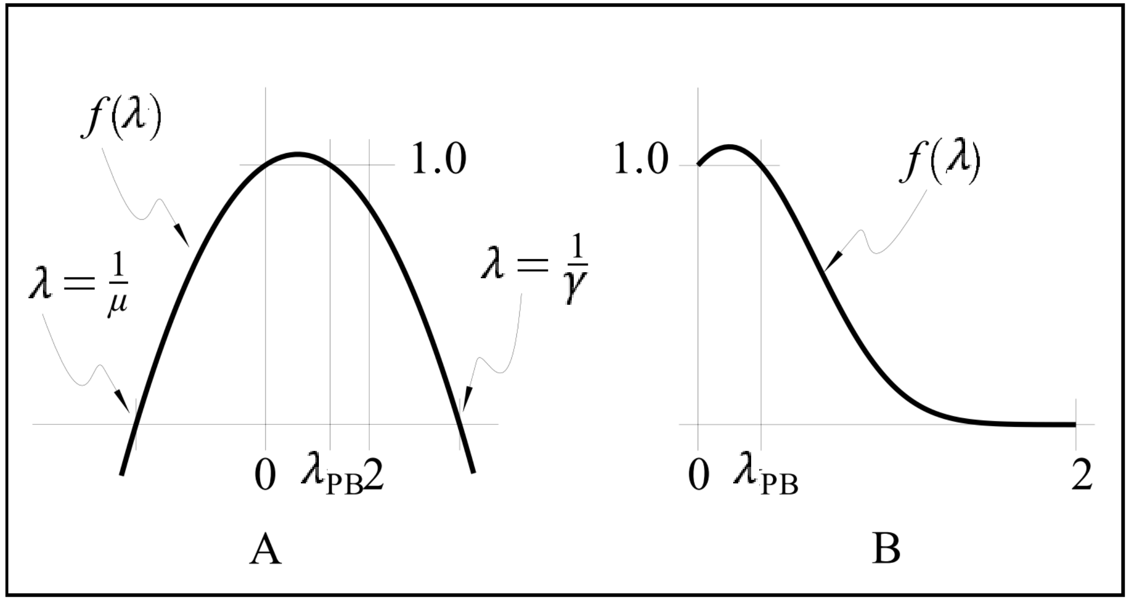


Figure 4: *Graph of transfer functions for the $\gamma|\mu$ algorithm. (A) $f(\lambda) = (1-\mu\lambda)(1-\gamma\lambda)$. (B) $f(\lambda) = ((1-\mu\lambda)(1-\gamma\lambda))^k$ with $k > 1$.*

Taubin smoothing

$$f(\lambda_i) = (1 - \gamma\lambda_i)(1 - \mu\lambda_i)$$

$$\mu < -\gamma < 0$$

Since $f(0) = 1$ and $\mu + \gamma < 0$, there is pass-band frequency λ_{PB} , such that $f(\lambda_{PB}) = 1$.

The value of λ_{PB} is $\lambda_{PB} = \frac{1}{\gamma} + \frac{1}{\mu}$.

Taubin smoothing (ICCV 1995)

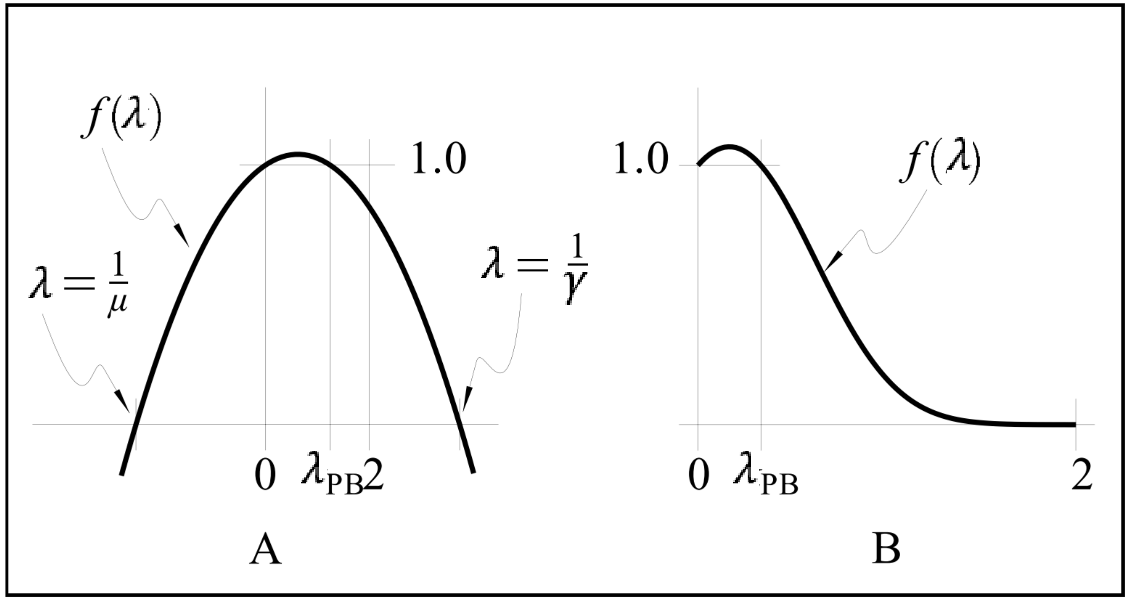


Figure 4: *Graph of transfer functions for the $\gamma|\mu$ algorithm. (A) $f(\lambda) = (1-\mu\lambda)(1-\gamma\lambda)$. (B) $f(\lambda) = ((1-\mu\lambda)(1-\gamma\lambda))^k$ with $k > 1$.*

$$f(\lambda_i) = (1-\gamma\lambda_i)(1-\mu\lambda_i)$$

$$\mu < -\gamma < 0, \qquad \lambda_{PB} = \frac{1}{\gamma} + \frac{1}{\mu}$$

region of interest $\lambda \in [0,2]$

pass-band: $\lambda = 0$ to $\lambda = \lambda_{PB}$.

As λ increases from $\lambda = \lambda_{PB}$ to $\lambda = 2$, $f(\lambda)$ decreases to zero.

The rate of decrease is controlled by the number of iterations k .

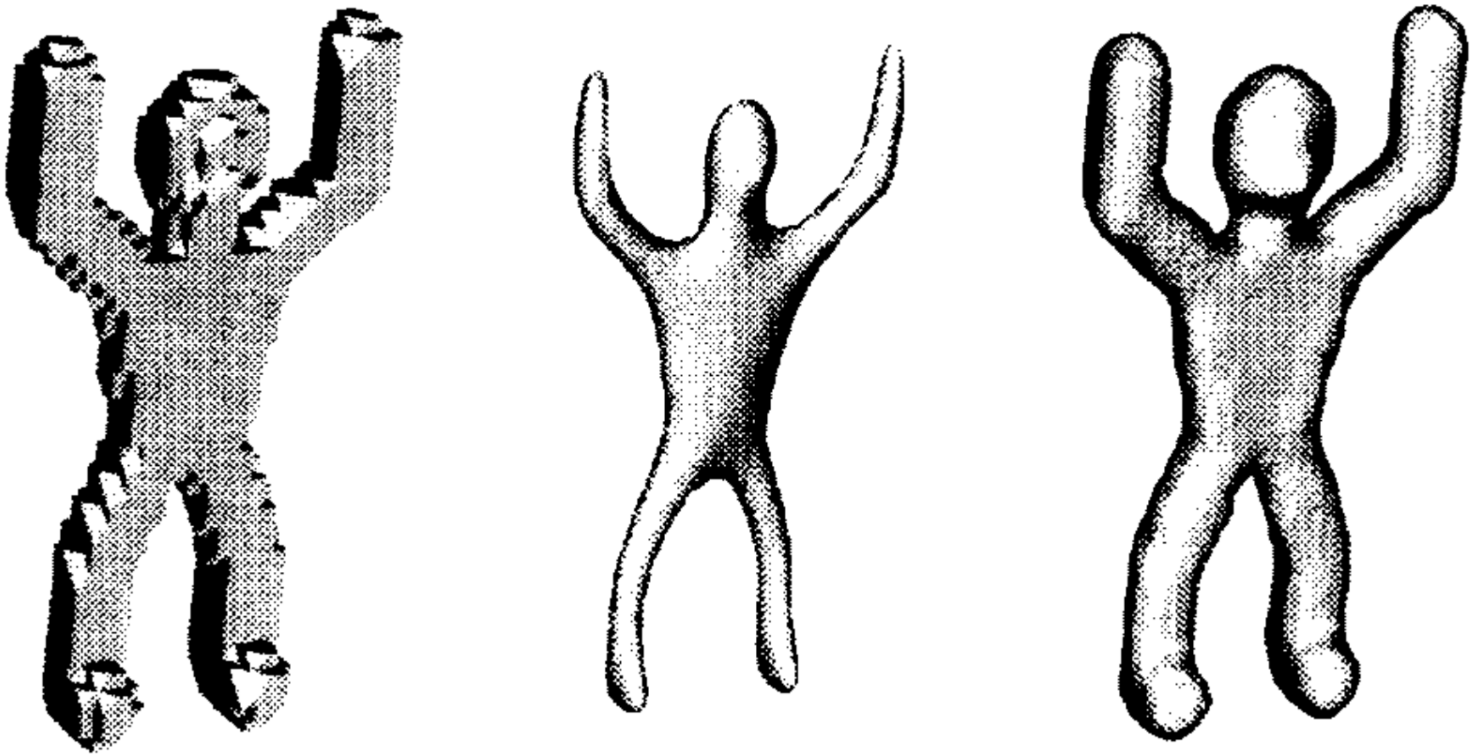
Taubin recommends $\lambda_{PB} = 0.1$

$1/\mu = -2.91, 1/\gamma = 3.01 \;\; 1/\mu = -1.91, 1/\gamma = 2.01$

$\rightarrow f(2) = (1-(1/3.01)2)(1+(1/2.91)\,2) =$0.33*1.69=0.56

Taubin smoothing (ICCV 1995)

Non-shrinking smoothing



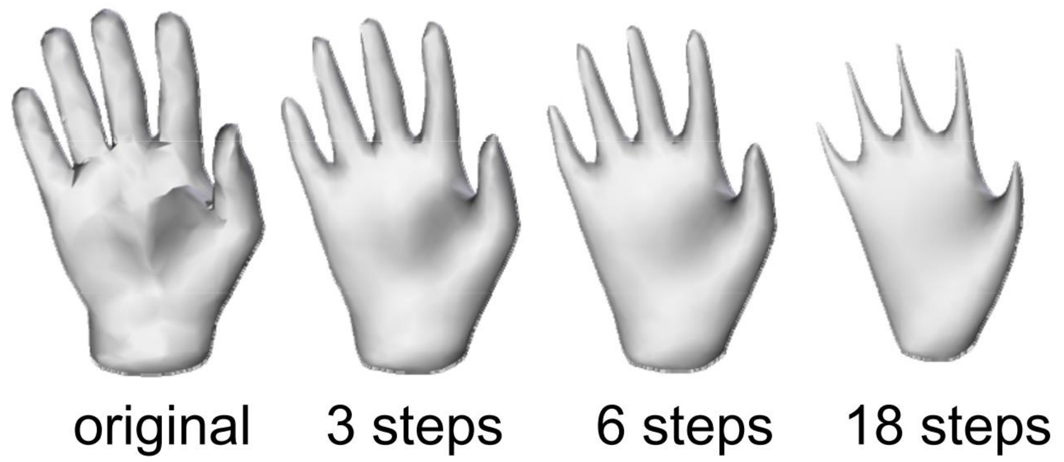
Laplacian smoothing

Taubin smoothing

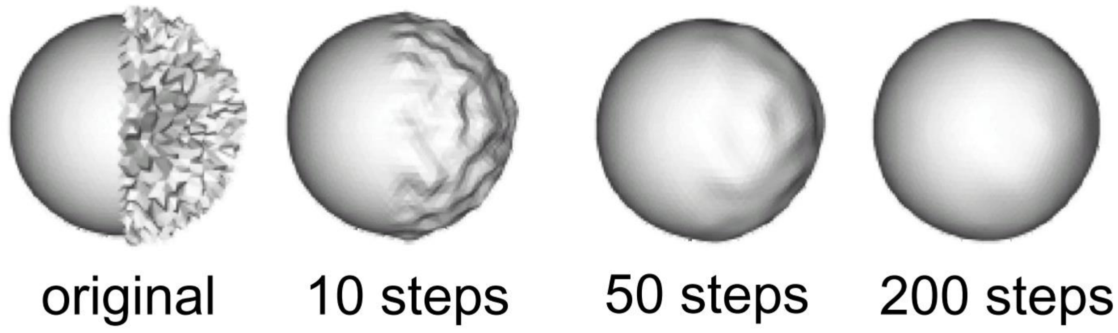
Taubin smoothing (ICCV 1995)

Non-shrinking smoothing

Laplacian smoothing



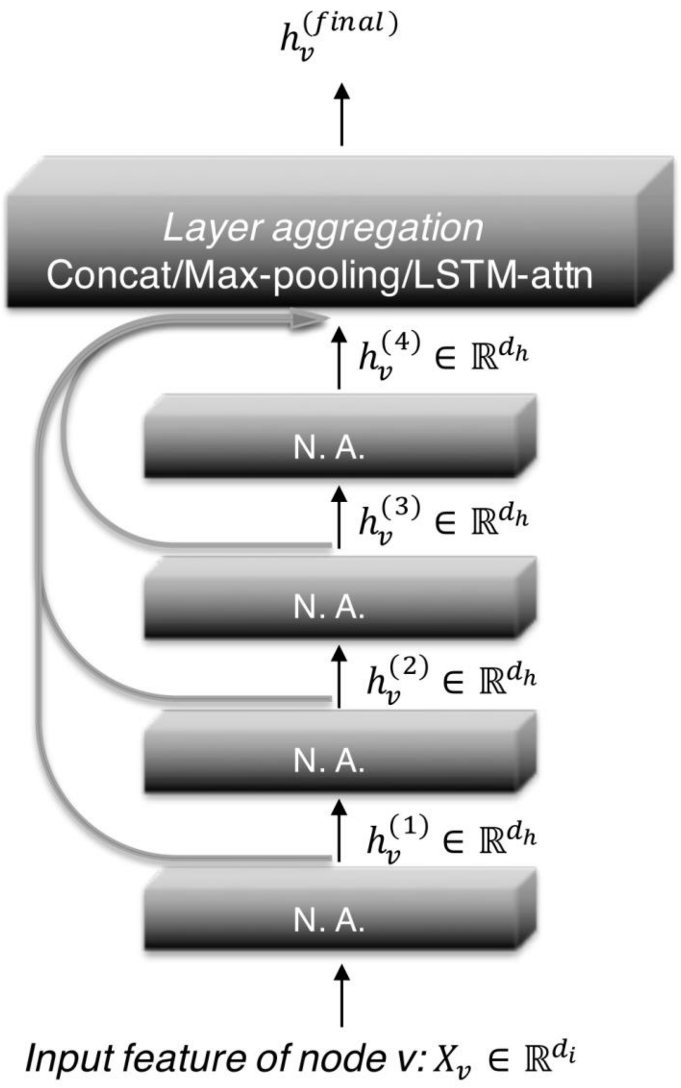
Taubin smoothing



Tackling Oversmoothing

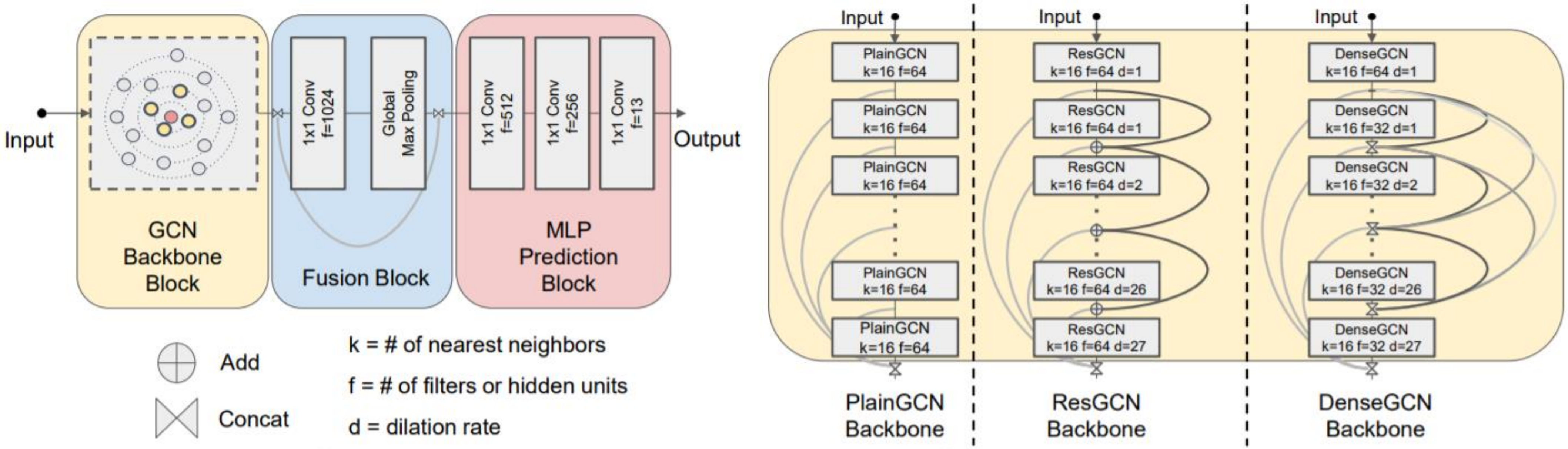
- Xu et al. (ICML 2018) uses jumping knowledge networks (kinds of skip connection) to alleviate the over smoothing issue.

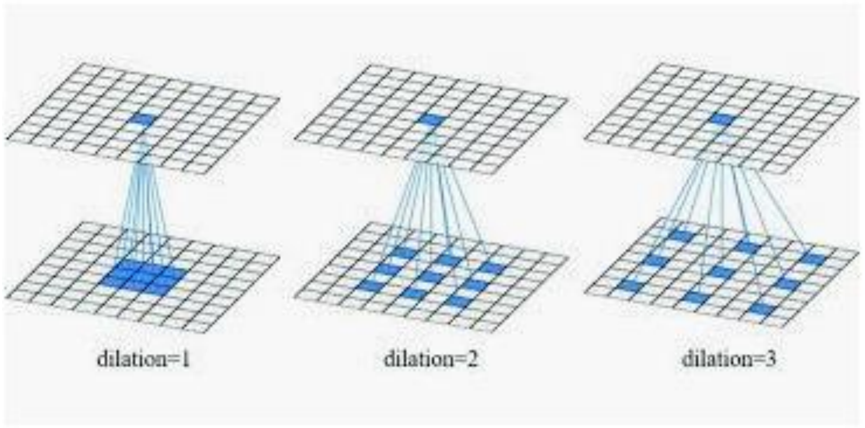
Model	Citeseer	Model	Cora
GCN (2)	77.3 (1.3)	GCN (2)	88.2 (0.7)
GAT (2)	76.2 (0.8)	GAT (3)	87.7 (0.3)
JK-MaxPool (1)	77.7 (0.5)	JK-Maxpool (6)	**89.6** (0.5)
JK-Concat (1)	**78.3** (0.8)	JK-Concat (6)	89.1 (1.1)
JK-LSTM (2)	74.7 (0.9)	JK-LSTM (1)	85.8 (1.0)



Tackling Oversmoothing

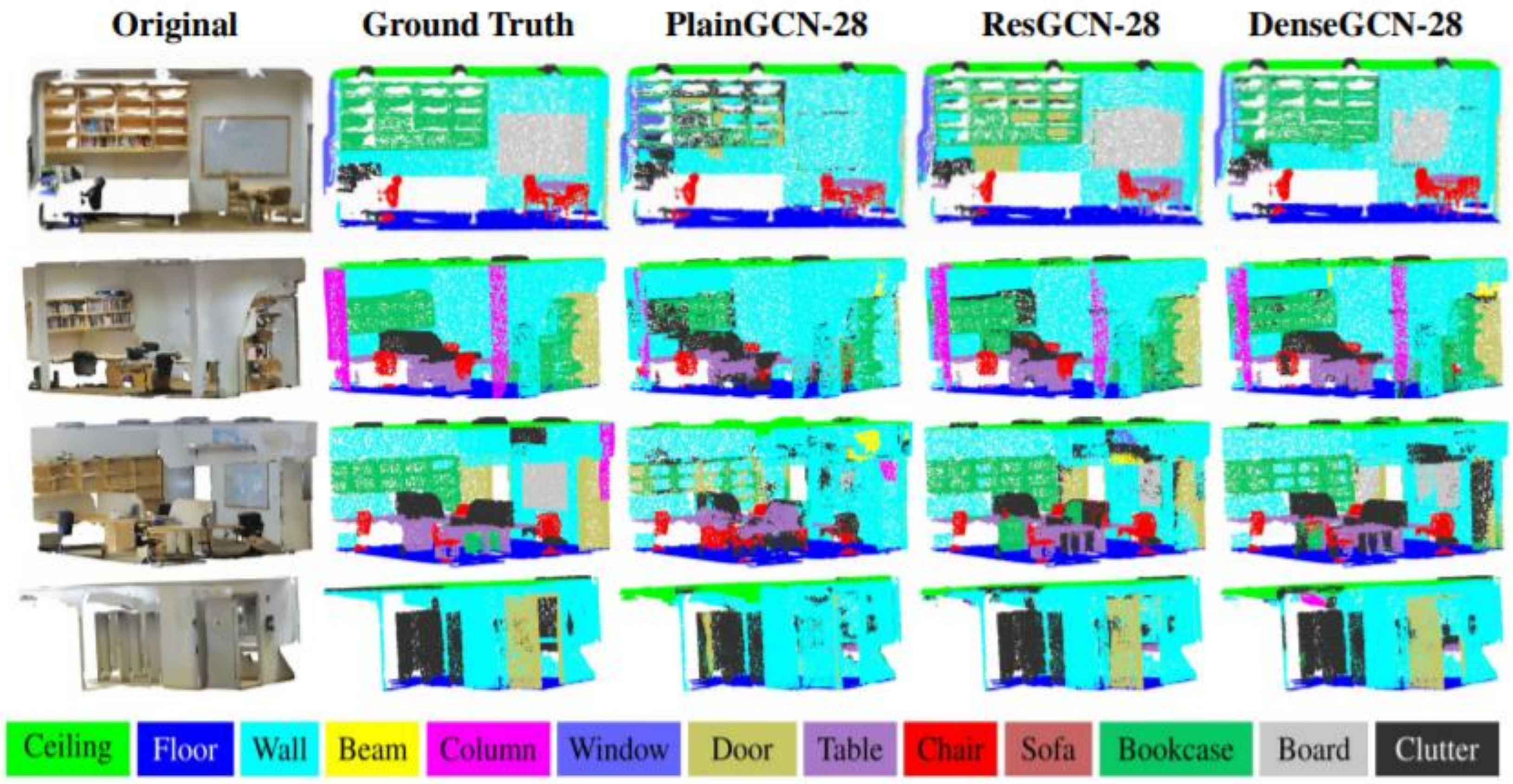
- Li et al. (ICCV 2019) borrows the concept of computer vision; ResNet, DenseNet, and Dilated convolution.





Tackling Oversmoothing

- [Li et al. (ICCV 2019)](#) borrows the concept of computer vision; ResNet, DenseNet, and Dilated convolution.



Tackling Oversmoothing

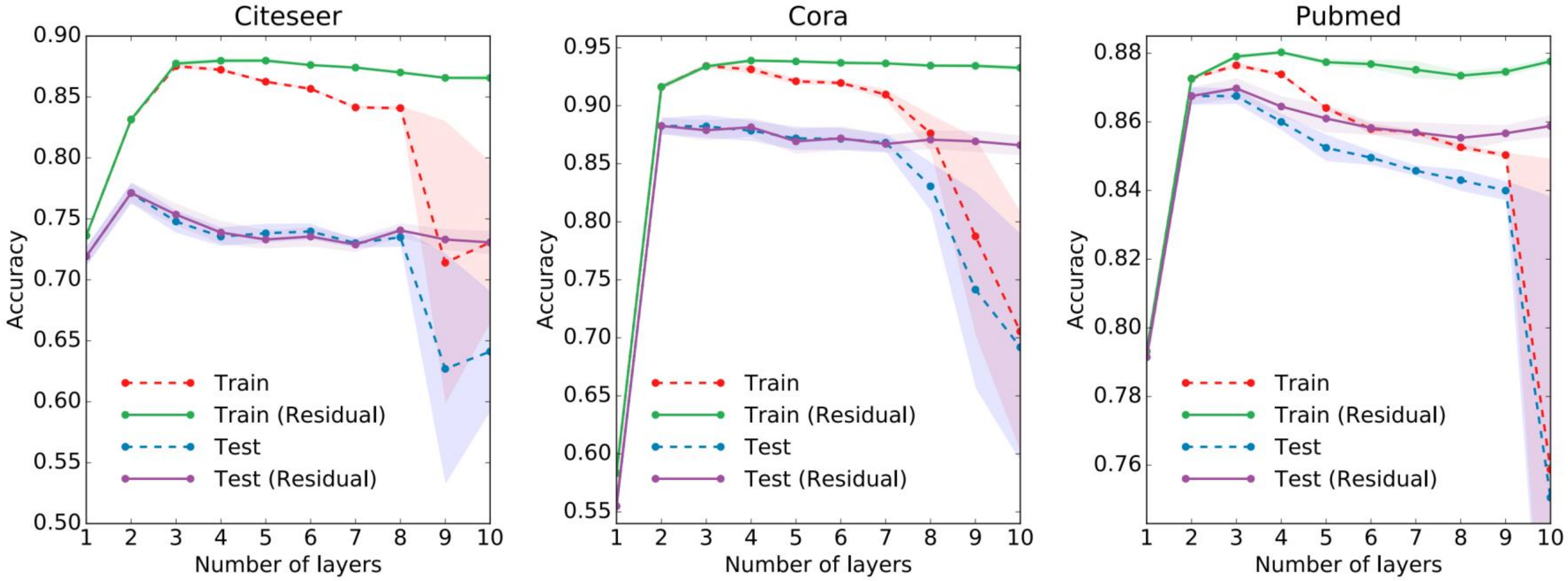
- Li et al. (ICCV 2019) borrows the concept of computer vision; ResNet, DenseNet, and Dilated convolution.

Method	OA	mIOU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [27]	78.5	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	54.1	42.0	9.6	38.2	29.4	35.2
MS+CU [8]	79.2	47.8	88.6	**95.8**	67.3	36.9	24.9	48.6	52.3	51.9	45.1	10.6	36.8	24.7	37.5
G+RCU [8]	81.1	49.7	90.3	92.1	67.9	**44.7**	24.2	52.3	51.2	58.1	47.4	6.9	39.0	30.0	41.9
PointNet++ [29]	-	53.2	90.2	91.7	73.1	42.7	21.2	49.7	42.3	62.7	59.0	19.6	45.8	48.2	45.6
3DRNN+CF [49]	**86.9**	56.3	92.9	93.8	73.1	42.5	25.9	47.6	59.2	60.4	**66.7**	24.8	**57.0**	36.7	51.6
DGCNN [42]	84.1	56.1	-	-	-	-	-	-	-	-	-	-	-	-	-
ResGCN-28 (*Ours*)	85.9	**60.0**	**93.1**	95.3	**78.2**	33.9	**37.4**	**56.1**	**68.2**	**64.9**	61.0	**34.6**	51.5	**51.1**	**54.4**

Table 2. **Comparison of *ResGCN-28* with state-of-the-art on S3DIS Semantic Segmentation**. We report average per-class results across all areas for our reference model *ResGCN-28*, which has 28 GCN layers, residual graph connections, and dilated graph convolutions, and state-of-the-art baselines. *ResGCN-28* outperforms state-of-the-art by almost 4%. It also outperforms all baselines in 9 out of 13 classes. The metrics shown are overall point accuracy (OA) and mean IoU (mIoU). '-' denotes not reported and **bold** denotes best performance.

Oversmoothing

- When the layers are deeper, the performances are degenerate harshly.
- The main cause of this phenomenon is over smoothing effects on GNNs.
- What is and how to circumvent over smoothing effects?



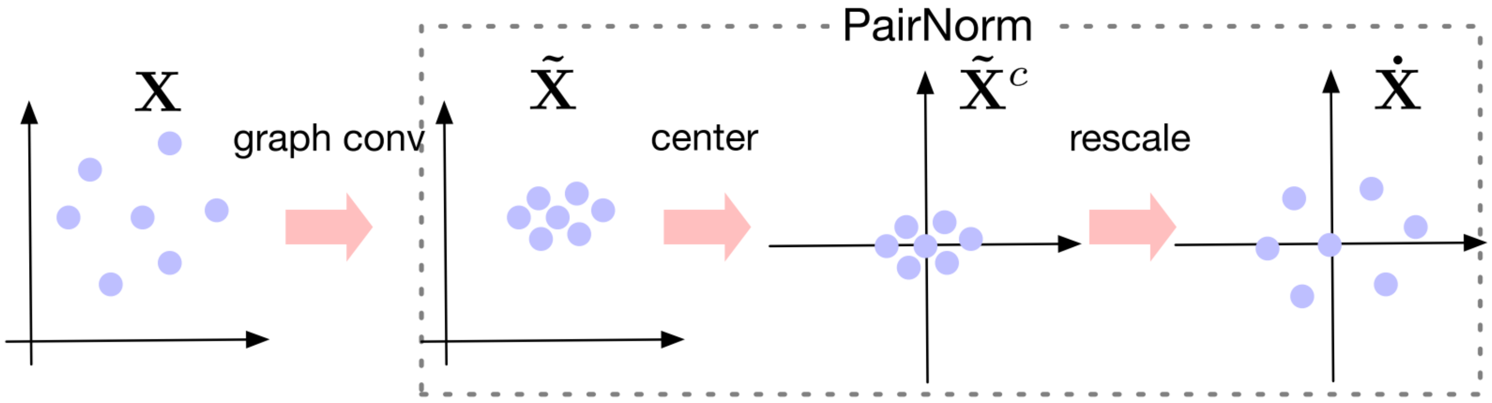
Overfitting?

[Kipf et al. (ICLR 2017)](#) Simplified ShevNet (GCN)

Tackling Oversmoothing

- Zhao et al. (ICLR 2020) proposes PairNorm, the first normalization layer for GNNs. They keep the total pairwise squared distance.

$$\min_{\bar{\mathbf{X}}} \sum_{i \in \mathcal{V}} \|\bar{\mathbf{x}}_i - \mathbf{x}_i\|_{\tilde{\mathbf{D}}}^2 + \sum_{(i,j) \in \mathcal{E}} \|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\|_2^2 - \lambda \sum_{(i,j) \notin \mathcal{E}} \|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\|_2^2$$



$$\sum_{(i,j) \in \mathcal{E}} \|\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_j\|_2^2 + \sum_{(i,j) \notin \mathcal{E}} \|\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_j\|_2^2 = \sum_{(i,j) \in \mathcal{E}} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 + \sum_{(i,j) \notin \mathcal{E}} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2$$

$$\mathrm{TPSD}(\dot{\mathbf{X}}) := \sum_{i,j \in [n]} \|\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_j\|_2^2 \quad = \quad \mathrm{TPSD}(\mathbf{X})$$

Tackling Oversmoothing

- Zhao et al. (ICLR 2020) proposes PairNorm, the first normalization layer for GNNs. They keep the total pairwise squared distance.

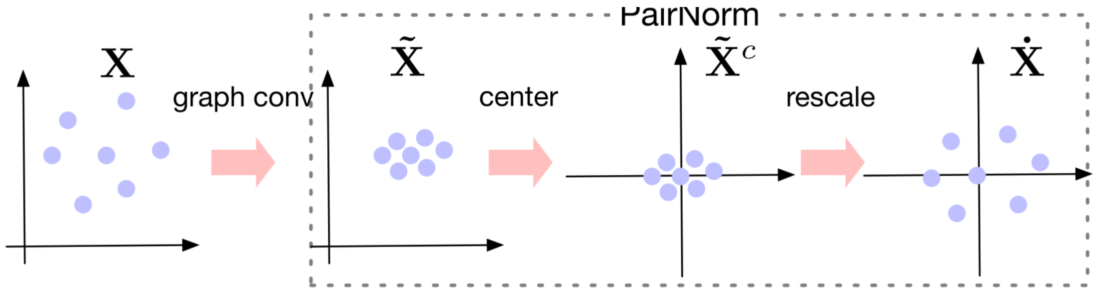
$$\mathrm{TPSD}(\dot{\mathbf{X}}) := \sum_{i,j\in[n]} \|\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_j\|_2^2 \quad = \quad \mathrm{TPSD}(\mathbf{X})$$

$$\tilde{\mathbf{x}}_i^c = \tilde{\mathbf{x}}_i - \frac{1}{n}\sum_{i=1}^{n} \tilde{\mathbf{x}}_i \qquad \text{(Center)}$$

$$\tilde{\mathbf{X}} = \tilde{\mathbf{A}}_{\mathrm{sym}}\mathbf{X}$$

$$\dot{\mathbf{x}}_i = s \cdot \frac{\tilde{\mathbf{x}}_i^c}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} \|\tilde{\mathbf{x}}_i^c\|_2^2}} = s\sqrt{n} \cdot \frac{\tilde{\mathbf{x}}_i^c}{\sqrt{\|\tilde{\mathbf{X}}^c\|_F^2}} \qquad \text{(Scale)}$$

$$\mathrm{TPSD}(\dot{\mathbf{X}}) = 2n\|\dot{\mathbf{X}}\|_F^2 = 2n\sum_i \|s \cdot \frac{\tilde{\mathbf{x}}_i^c}{\sqrt{\frac{1}{n}\sum_i \|\tilde{\mathbf{x}}_i^c\|_2^2}}\|_2^2 = 2n\frac{s^2}{\frac{1}{n}\sum_i \|\tilde{\mathbf{x}}_i^c\|_2^2}\sum_i \|\tilde{\mathbf{x}}_i^c\|_2^2 = 2n^2 s^2$$



Tackling Oversmoothing

- Zhao et al. (ICLR 2020) proposes PairNorm, the first normalization layer for GNNs. They keep the total pairwise squared distance.

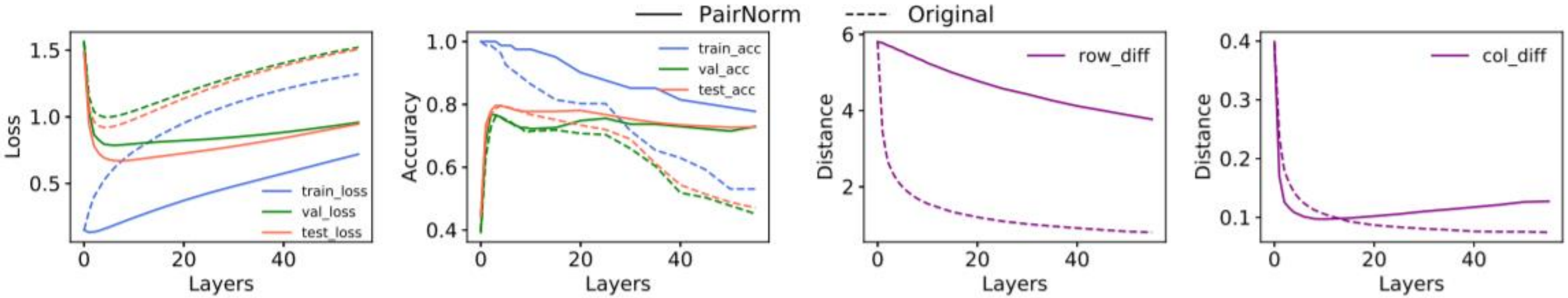


Figure 1: (best in color) SGC's performance (dashed lines) with increasing graph convolutions (K) on `Cora` dataset (train/val/test split is 3%/10%/87%). For each K, we train SGC in 500 epochs, save the model with the best validation accuracy, and report all measures based on the saved model. Measures row-diff and col-diff are computed based on the final layer representation of the saved model. (Solid lines depict after applying our method PAIRNORM, which we discuss in §3.2.)

Outline of Lecture

- Introduction
- Graph Spectral Theory
  - Definition of Graph
  - Graph Laplacian
  - Laplacian Smoothing
- Graph Node Clustering
  - Minimum Graph Cut
  - Ratio Graph Cut
  - Normalized Graph Cut
- Manifold Learning
  - Spectral Analysis in Riemannian Manifolds
  - Dimension Reduction, Node Embedding
- Semi-supervised Learning (SSL) : conti.
  - Self-Training Methods
  - SSL with SVM
  - SSL with Graph using MinCut
  - SSL with Graph using Harmonic Functions
  - SSL with Graph using Regularized Harmonic Functions
  - SSL with Graph using Soft Harmonic Functions
  - SSL with Graph using Manifold Regularization (out of sample extension)
  - SSL with Graph using Laplacian SVMs
  - SSL with Graph using Max-Margin Graph Cuts
  - Online SSL
  - SSL for large graph

Outline of Lecture

- Review: Convolution Neural Networks (CNN)
  - Feedforward Neural Networks
  - Convolution Integral (Temporal)
  - Convolution Sum (Temporal)
  - Circular Convolution Sum
  - Convolution Sum (Spatial)
  - Convolutional Neural Networks
- Graph Convolution Networks (GCN)
  - What are issues on GCN
  - Graph Filtering in GCN
  - Graph Pooling in GCN
- Original GNN (Scarselli et al. 2005)
- Spectral GCN
  - Spectral Filtering
  - Graph Spectral Filtering in GCN
  - Spectral Graph CNN (Bruna et al. ICLR 2014)
  - ChebNet (Defferard et al. NIPS 2016)
  - Simplified ChebNet (Kipf & Welling, ICLR 2017)
- Spatial GCN
  - Spatial View of Simplified ChebNet
  - GraphSage (Hamilton et al. NIPS 2017)
  - GAT : Graph Attention (Veličković et al. ICLR 2018)
  - MPNN: Message Passing (Glimer et al. ICML 2017)
  - **gPool:** Graph U-Nets (Gao et al. ICML 2019)
  - **DiffPool:** Differentiable Pooling (Ying et al. NeurIPS 2018)
  - **EigenPooling:** EigenPooling (Ma et al. KDD 2019)

Outline of Lecture

- Link Analysis
  - Directed Graph
    - Strongly Connected Graph
    - Directed Acyclic Graph
  - Link Analysis Algorithms
    - PageRank (Ranking of Nodes)
    - Random Teleports
    - Google Matrix
    - Sparse Matrix Formulation
    - Personalized PageRank
    - Random Walk with Restart
- Random Walks and Diffusion
- Propagation using graph diffusion
  - APPNP: Predict Then Propagate [ICLR'19]
  - Graph Diffusion-Embedding Networks [CVPR'19]
- Making a new graph
  - Diffusion Improves Graph Learning [NIPS'19]
  - SSL with Graph Learning-Convolutional Networks [CVPR'19]
- Deep Generative Models For Graph
  - Problem of Graph Generation
  - ML Basics for Graph Generation
  - GraphRNN : Generating Realistic Graphs
  - Applications and Open Questions
- Tacking Over-smoothing
  - Oversmoothing in GCN
  - Taubin smoothing
  - Jumping knowledge networks
  - ResNet, DenseNet, and Dilated convolution
  - PairNorm; normalization layer for GNNs

Summary Questions of the lecture

- Why does multiple Laplacian smoothing lead to over-smoothing?
- What is Taubin smoothing?
- Explain PairNorm for GCN.