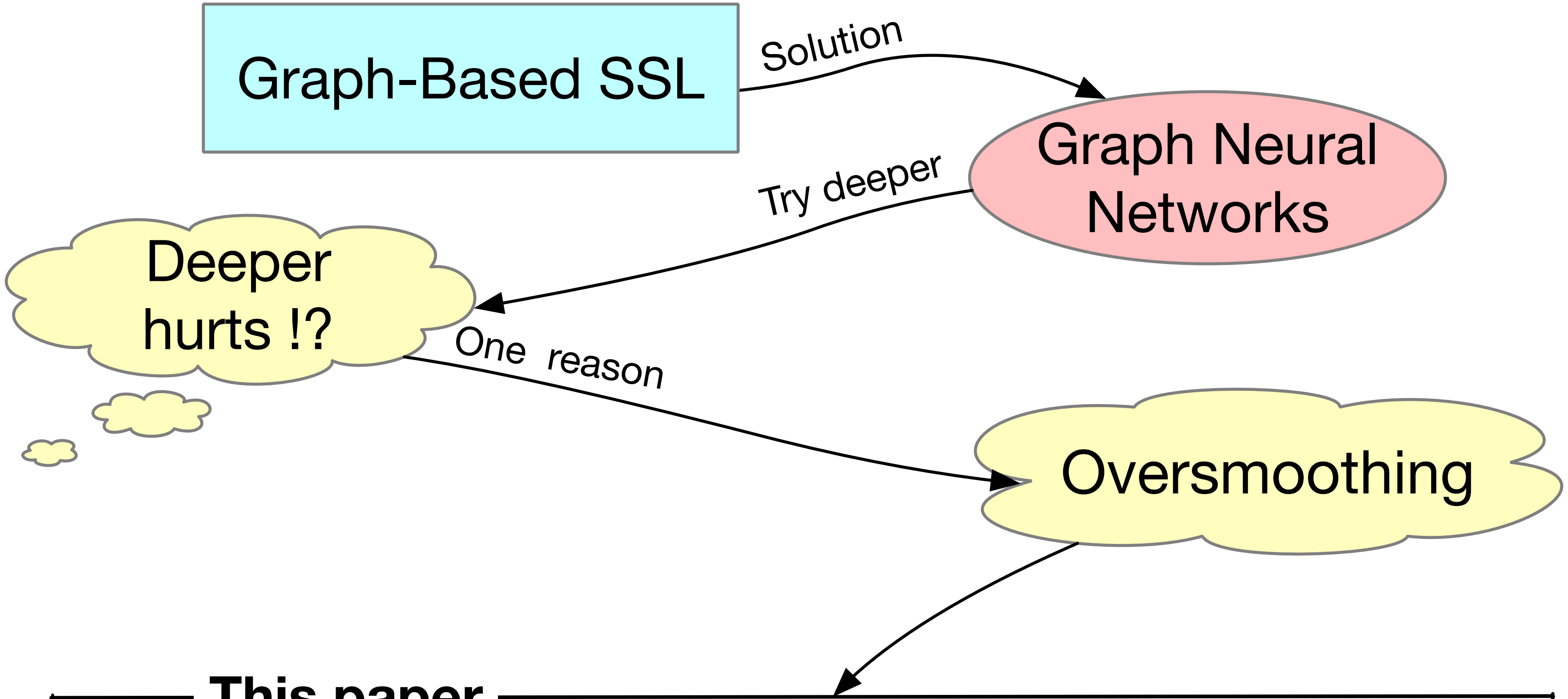
PairNorm: Tackling Oversmoothing in GNNS

Appearing in ICLR 2020:
International Conference on Learning Representations

Lingxiao Zhao, Leman Akoglu

Carnegie Mellon University

The Big Picture



This paper

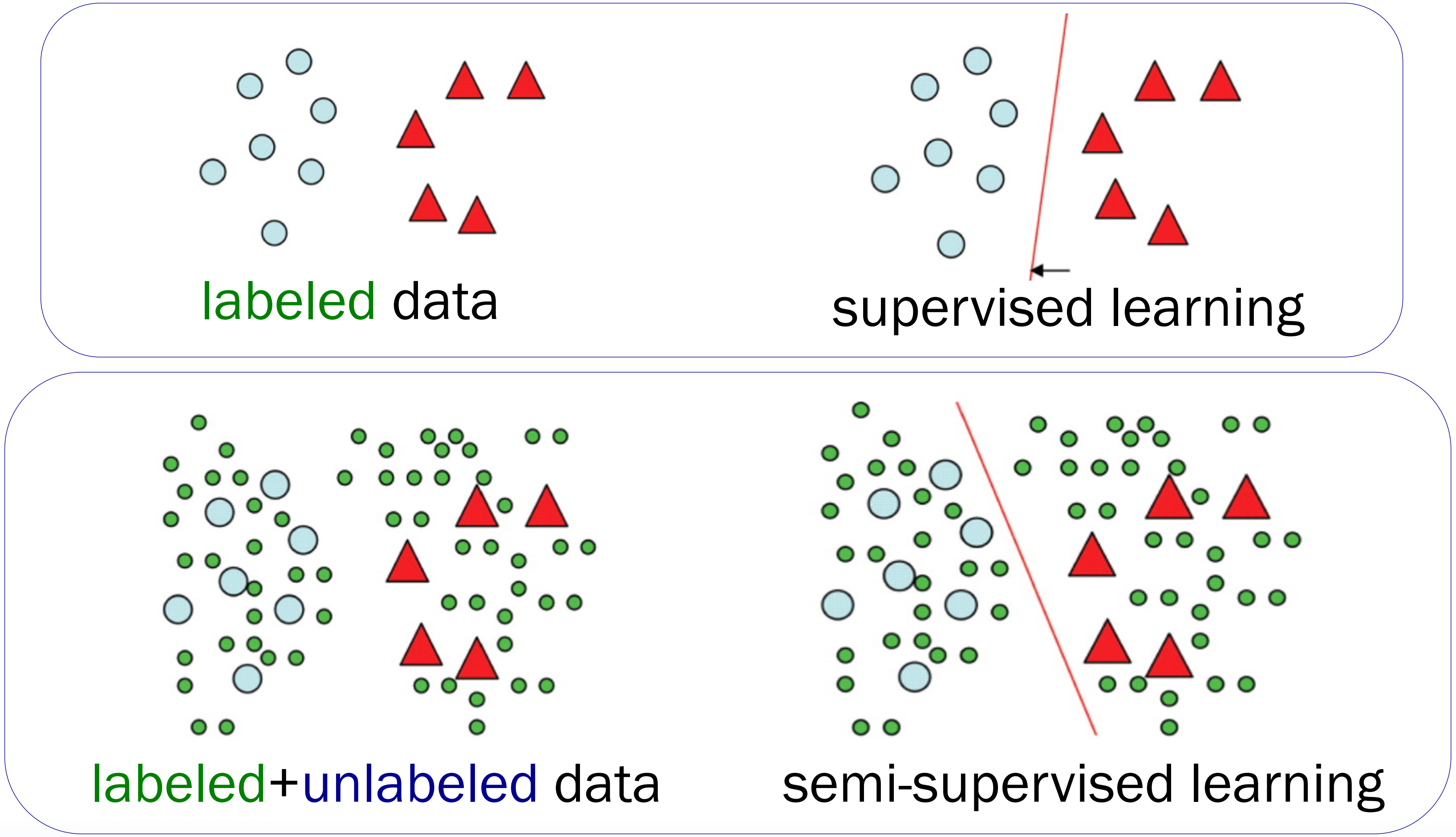
What is over smoothing? Any measures?
How to solve or relieve it for any GNNs?

Agenda

- Background
  - Graph-based semi-supervised learning
  - Modern approach: graph neural networks
  - Oversmoothing problem of GNN
- Understanding oversmoothing
- PairNorm: tackling oversmoothing
- Semi-supervised node classification with missing feature

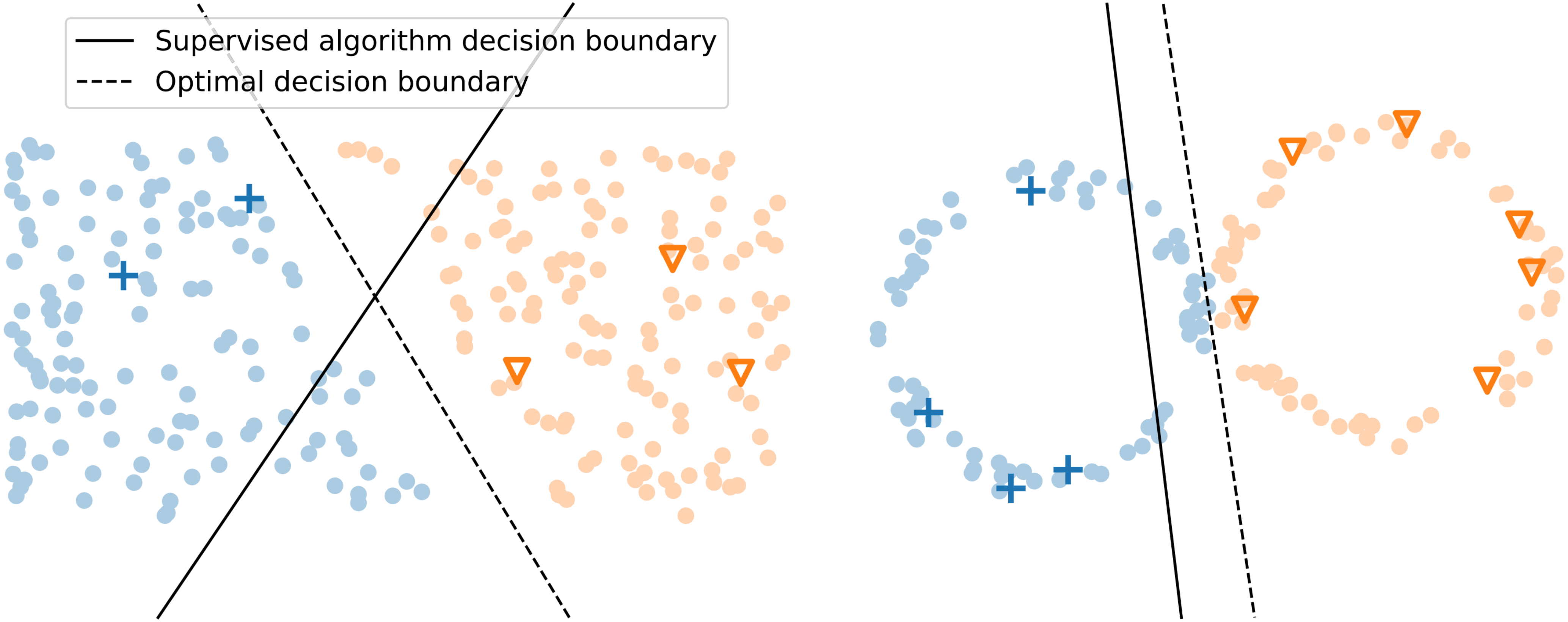
Background: SSL

- Semi Supervised Learning (SSL)



Background: SSL

- **Why SSL works:** assumptions link P(X) and P(Y|X)

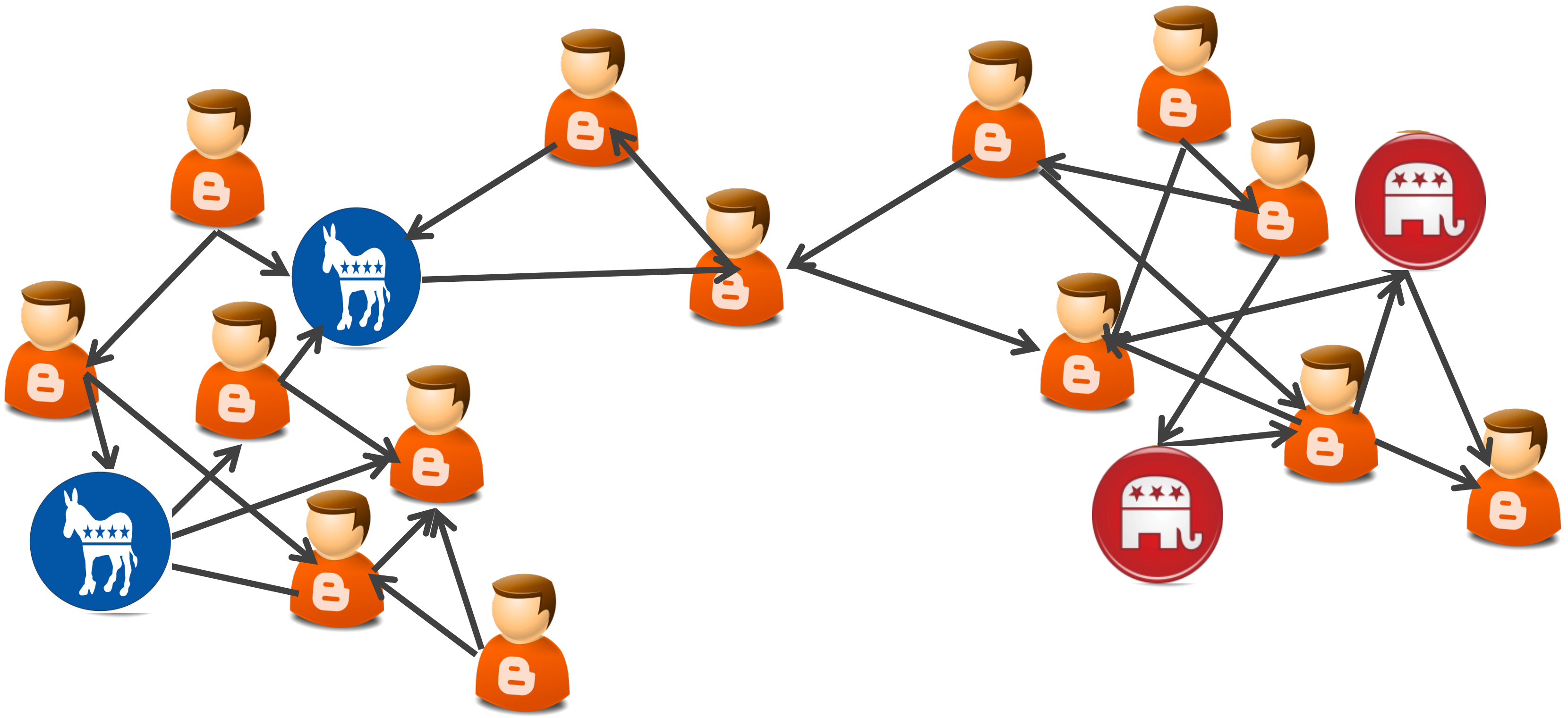


(a) Smoothness and low-density assumptions. **(b)** Manifold assumption.

Borrowed from https://link.springer.com/content/pdf/10.1007/s10994-019-05855-6.pdf

Background: Graph-Based SSL

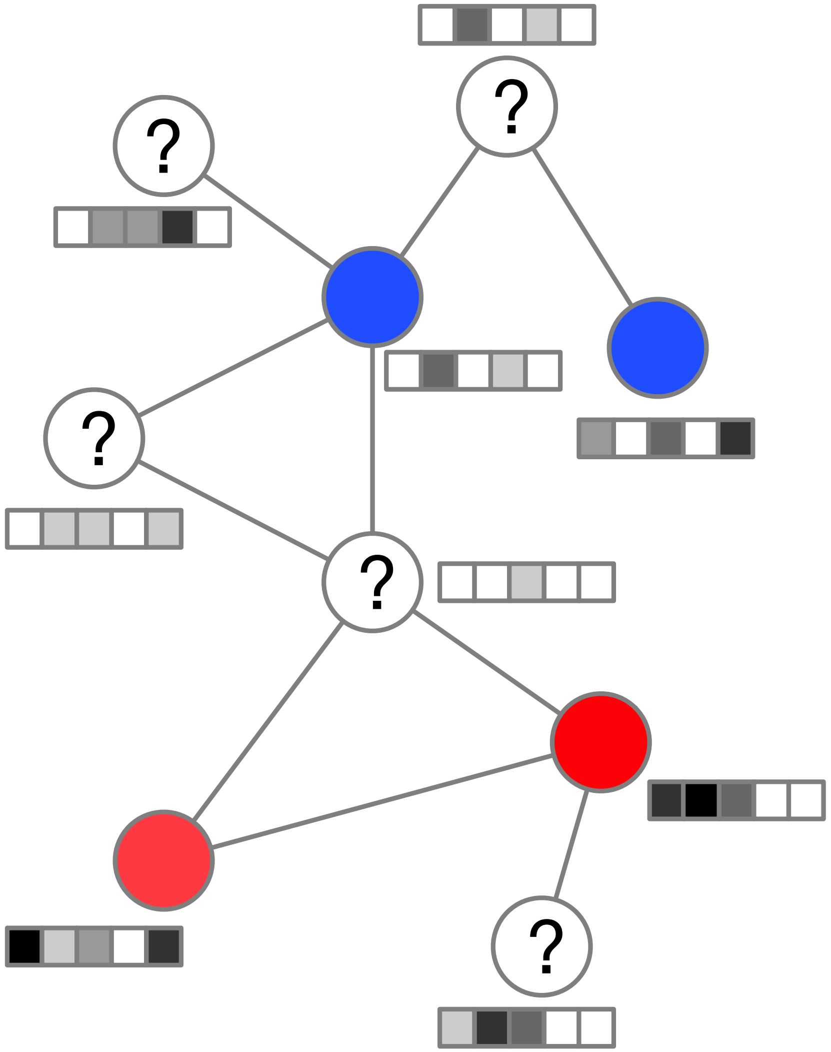
- Additional information: Graph



- Example: Political blog citations

Background: Graph-Based SSL

- Given
  - Set L of labeled nodes with attributes
  - Set U of unlabeled nodes with attributes
  - A graph A of all nodes
- Goal
  - Assign label Y to all unlabeled nodes in U
- Old approaches
  - Label Propagation [Zhu 03]
  - Deep Walk [Perozzi 14]
  - Iterative Classification [Lu 03]
  - ....



Modern Solution: Graph Convolutional Network

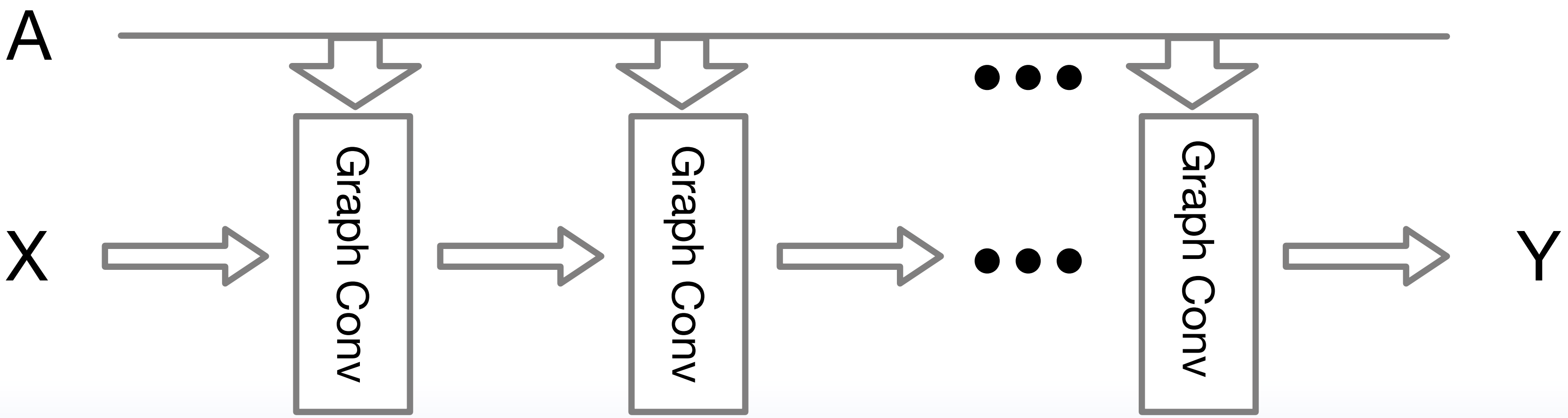
- Neural Network
  - Stacking of linear layer followed by nonlinear activation

$$H^{(t+1)} = \sigma(H^{(t)}W^{(t)} + b^{(t)})$$

[W, b are Params]

- Graph Convolutional Neural Network (GCN) [Kipf 17]
  - Stacking of GraphConv layer followed by nonlinear activation

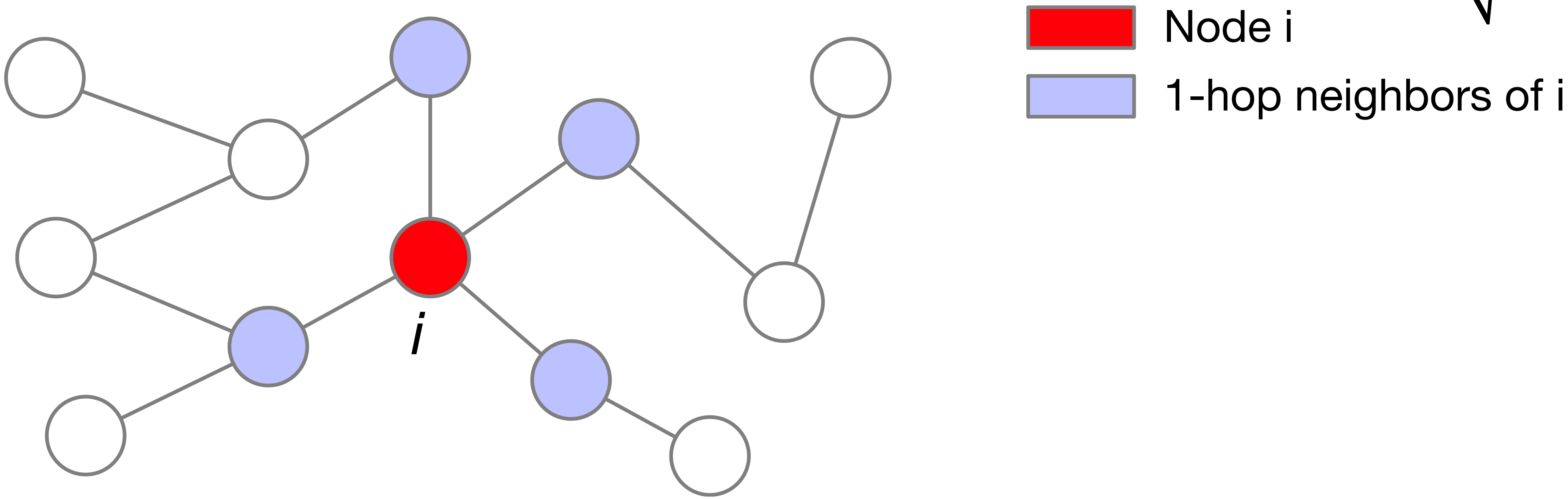
$$H^{(t+1)} = \sigma(\tilde{A}H^{(t)}W^{(t)} + b^{(t)})$$



Modern Solution: Graph Convolutional Network

- GraphConv: Micro Perspective

$$H^{(t+1)} = \sigma(\tilde{A}H^{(t)}W^{(t)} + b^{(t)}) \Rightarrow h_i^{(t+1)} = \sigma(W^{(t)} \sum_{j \in N_i \cup \{i\}} \frac{h_j^{(t+1)}}{\sqrt{|N_i||N_j|}})$$



- Stacking k layers expand the receptive field to k-hop neighbors!

Other variants of GNN

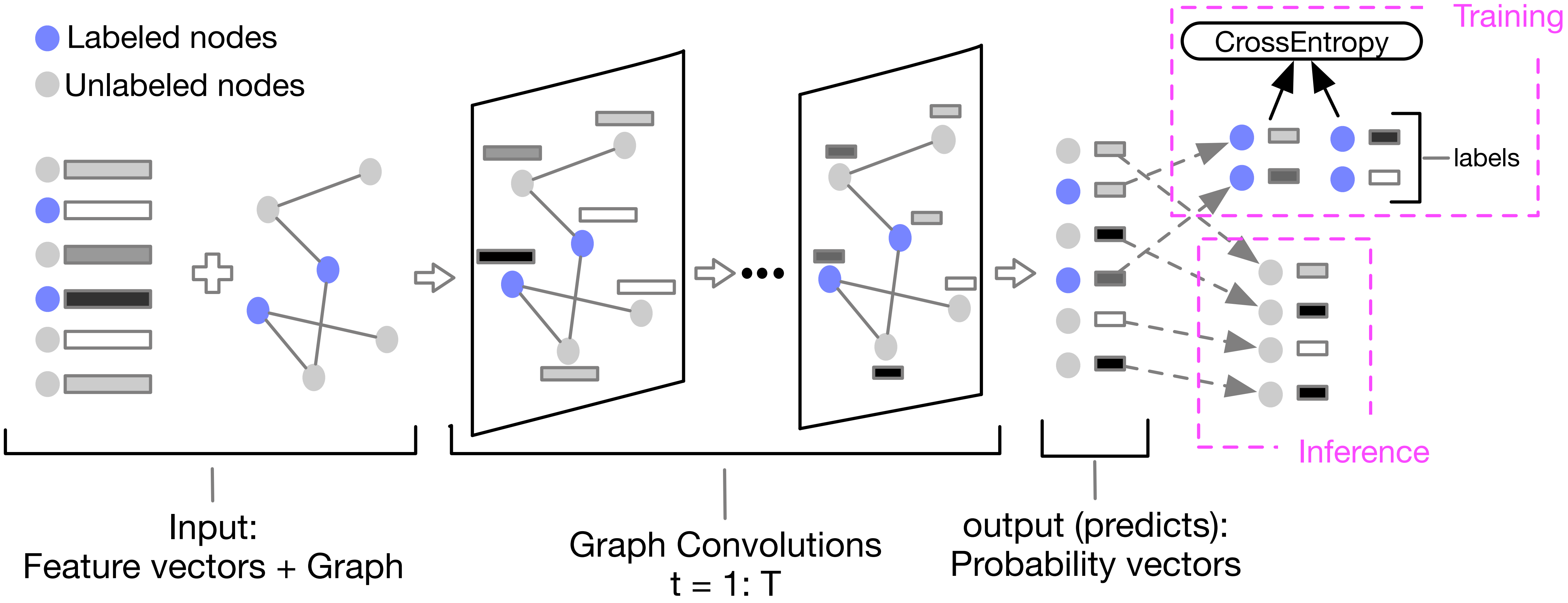
- Graph Attention Network [Velickovic et al., 2018]
- GraphSAGE [Hamilton et al., 2017]
- Deep graph infomax [Velickovic et al., 2019]
- Graph Markov Neural Networks [Qu et al., 2019]
- …

Refer to a good survey of GNN:

Wu, Zonghan et al., *arXiv preprint arXiv:1901.00596* (2019).

A comprehensive survey on graph neural networks.

Apply GCN to SSL



Oversmoothing problem: view 1

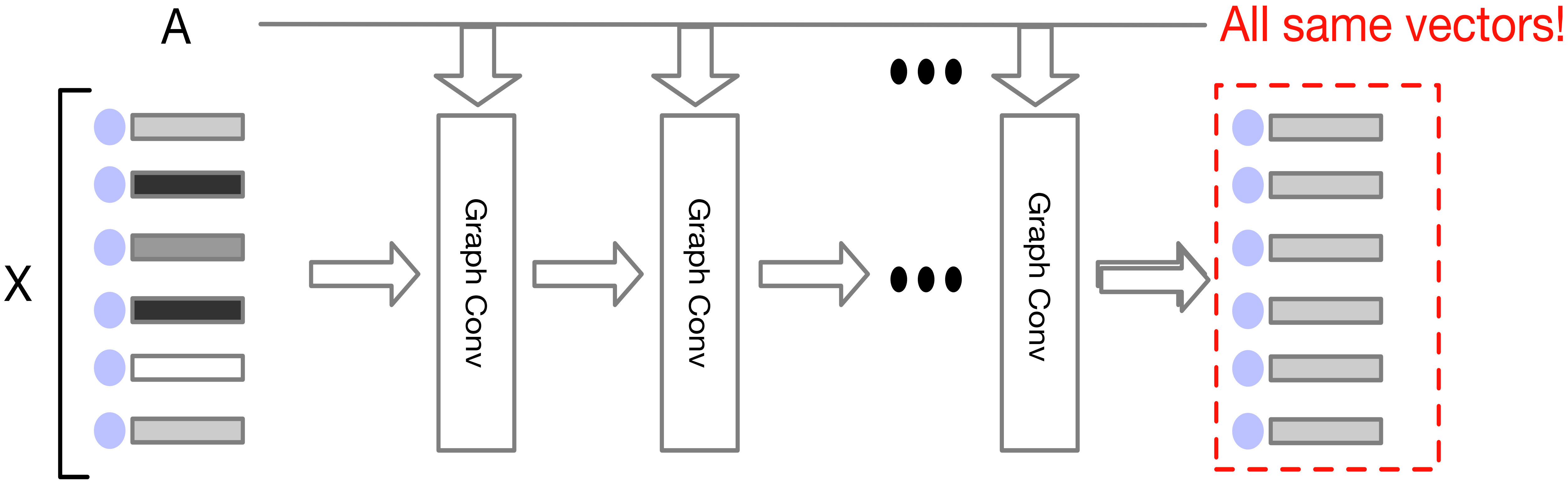
- **Intuition:** More layers => larger receptive fields => more information
- **Observation:** hurt the performance (after more than 2~3 layers)?
- **Why?**
    1. More parameters => overfitting
    2. More layers => vanishing gradient => hard to train
    3. Oversmoothing of graph convolution operation [our focus]
- Recall that in GCN:

$$\boldsymbol{h}_i^{l+1} = \boldsymbol{\sigma}\left(\boldsymbol{W}^l \sum_{j \in N_i \cup \{i\}} \frac{\boldsymbol{h}_j^l}{\sqrt{|N_i||N_j|}}\right)$$

Intuitively, keeping aggregating information from neighbors can make all nodes **indistinguishable**!

Node-wise oversmoothing

Node-wise oversmoothing



Oversmoothing problem: view 2

- Different view from matrix form (ignoring parameters)

$$H^{l+1} = \widetilde{A}_{sym} H^{l}$$

$$\Longrightarrow \quad H^{N} = \widetilde{A}_{sym}^{N} H^{0} = \widetilde{A}_{sym}^{N} X$$

- There is a stationary distribution for $\widetilde{A}_{sym}$, if viewing it as transition matrix of random walk.

Feature-wise oversmoothing

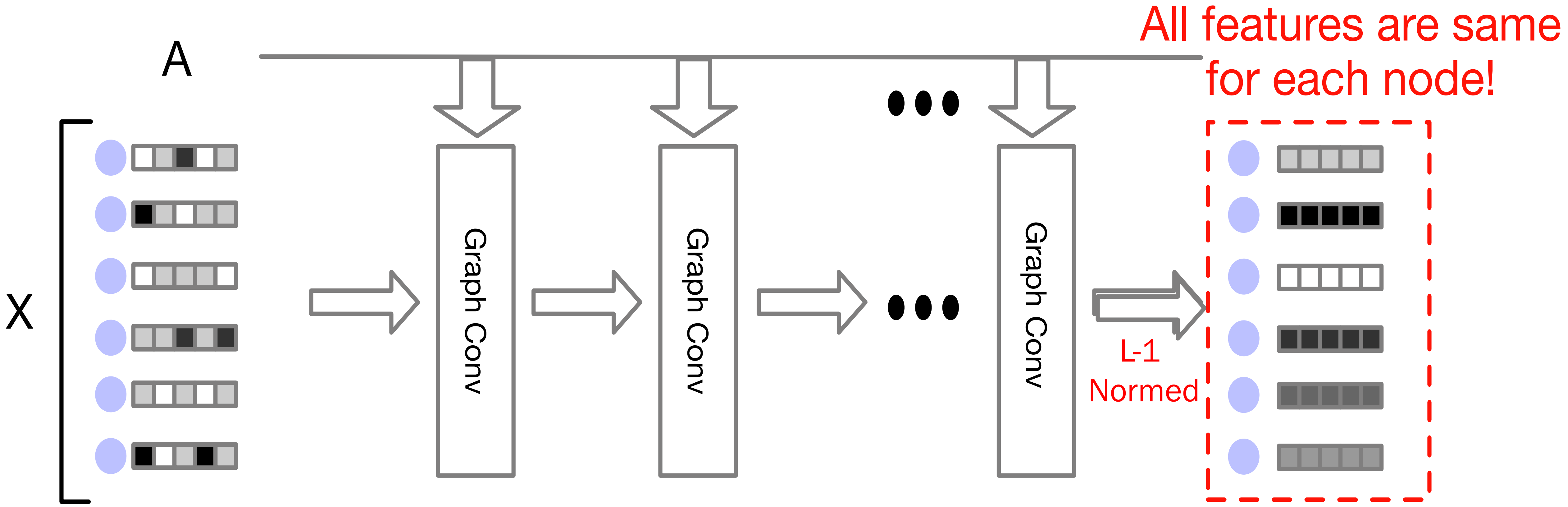
Let $\mathbf{x}_{\cdot j} \in \mathbb{R}^n$ denote the j-th column of $\mathbf{X}$. Then, for *any* $\mathbf{x}_{\cdot j} \in \mathbb{R}^n$:

$$\lim_{k \to \infty} \tilde{\mathbf{A}}_{\text{sym}}^{k} \mathbf{x}_{\cdot j} = \boldsymbol{\pi}_j \quad \text{and} \quad \frac{\boldsymbol{\pi}_j}{\|\boldsymbol{\pi}_j\|_1} = \boldsymbol{\pi} \,,$$

$\boldsymbol{\pi} \in \mathbb{R}^n$ satisfies $\boldsymbol{\pi}_i = \frac{\sqrt{deg_i}}{\sum_i \sqrt{deg_i}}$ for all $i \in [n]$.

Typo: $x_{\cdot j} \in \mathbb{R}^{+n}$

Feature-wise oversmoothing



Overswmoothing washes away the original features!

Agenda

- Background
- Understanding oversmoothing
  - Focusing on oversmoothing: via SGC
  - Measures: row-diff and col-diff
  - Results for semi-supervised node classification(**SSNC**) problem
- PairNorm: tackling oversmoothing
- Semi-supervised node classification with missing feature

Simplified graph convolutional network (SGC)

- SGC [Wu et al., 2019]

Graph convolution: $\boldsymbol{H}^{l+1} = \widetilde{\boldsymbol{A}}_{\boldsymbol{sym}}\boldsymbol{H}^{l}$

Readout: $\boldsymbol{O} = Softmax(\boldsymbol{H}^{N}\boldsymbol{W}^{N})$

- Removing parameters and activation functions of GCN
  Add a linear transformation in readout

- Achieve similar result as GCN for SSL

Focusing on oversmoothing: via SGC

- Recall reasons:
  1. More parameters => overfitting
  2. More layers => vanishing gradient => hard to train
  3. Oversmoothing of graph convolution operation

- To decouple oversmoothing from other factors, we study the oversmoothing problem using SGC model. [Wu et al., 2019]

$$\widehat{\boldsymbol{Y}} = \mathrm{softmax}(\tilde{\mathbf{A}}_{\mathrm{sym}}^{K}\ \mathbf{X}\ \mathbf{W})$$

  i. **No effect of overfiiting:** fixed number of parameters
  ii. **No effect of vanishing gradient:** not a ”deep” model

Measures: row-diff and col-diff

- **Row-diff**: measuring node-wise oversmoothing

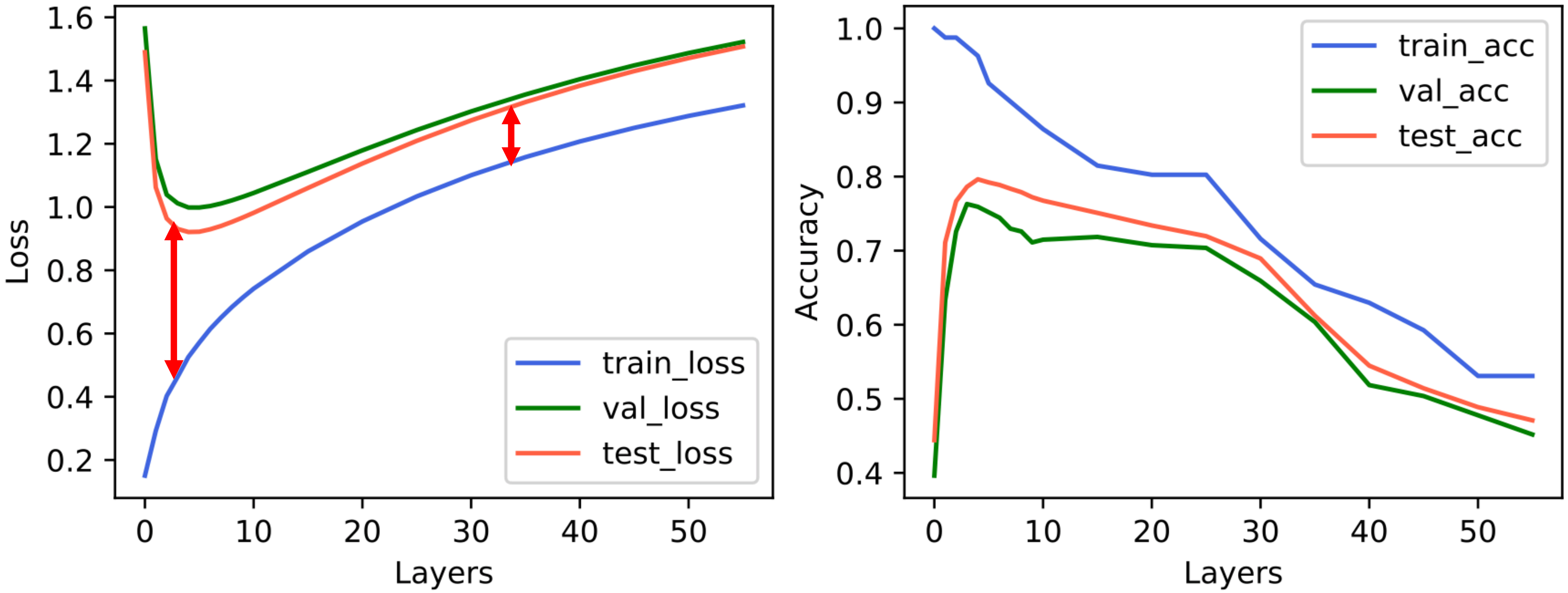
$$\text{row-diff}(\mathbf{H}^{(k)}) = \frac{1}{n^2} \sum_{i,j \in [n]} \left\| \mathbf{h}_i^{(k)} - \mathbf{h}_j^{(k)} \right\|_2$$

- **Col-diff**: measuring feature-wise oversmoothing

$$\text{col-diff}(\mathbf{H}^{(k)}) = \frac{1}{d^2} \sum_{i,j \in [d]} \left\| \mathbf{h}_{\cdot i}^{(k)} / \|\mathbf{h}_{\cdot i}^{(k)}\|_1 - \mathbf{h}_{\cdot j}^{(k)} / \|\mathbf{h}_{\cdot j}^{(k)}\|_1 \right\|_2$$

Measuring oversmoothing effect

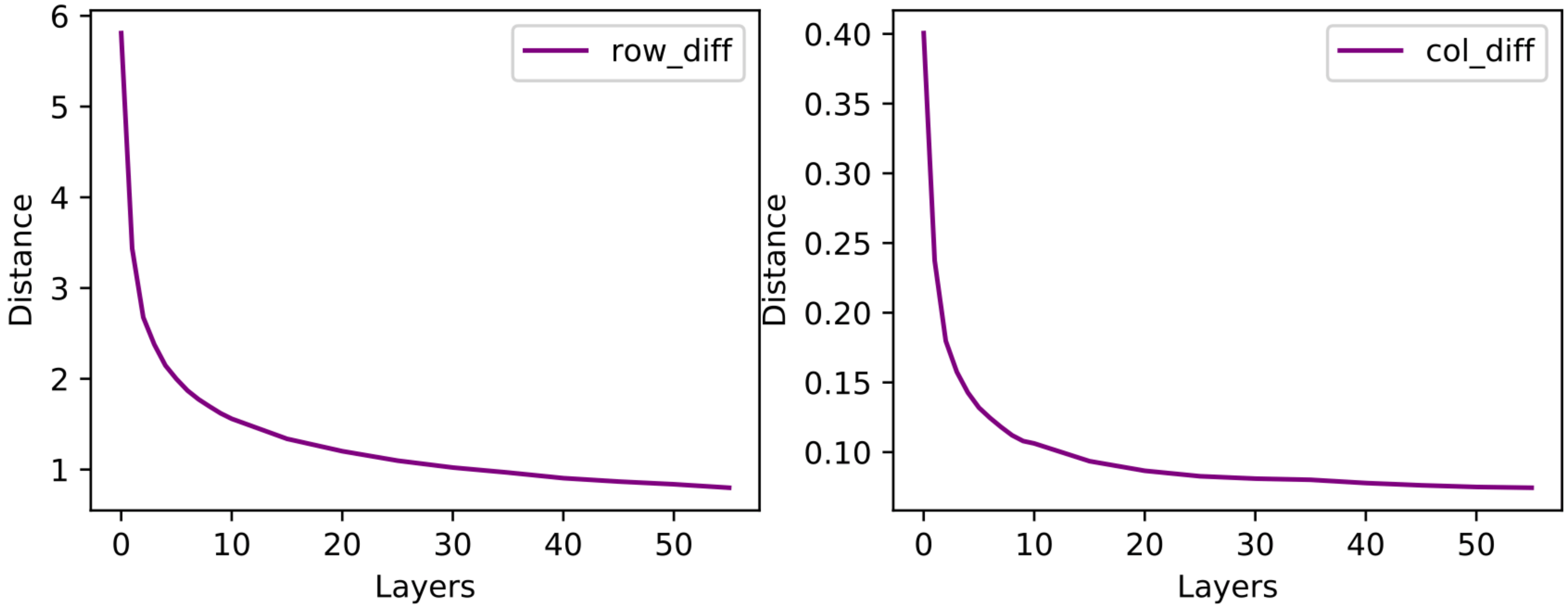
- Data: Cora, random split with 3%train, 10%valiation, 87% test
- Model: SGC



- Learning harder materials makes you perform better in exam! But can not be too hard!   [Human learning perspective]

Measuring oversmoothing level

- Cora dataset: random split with 3%train, 10%valiation, 87% test



- Both node-wise and feature-wise oversmoothing are happening

Agenda

- Background
- Understanding oversmoothing
- **PairNorm: tackling oversmoothing**
  - PairNorm: normalizing total pairwise distance
  - Effective of SGC+PairNorm over SSNC problem
- Semi-supervised node classification with missing feature

PairNorm: normalizing pairwise squared distance

- **Design:** Normalization layer after GraphConv
- **Intuition:** keep total pairwise squared distance (TPSD) constant across layers to prevent node-wise oversmoothing. Combining with GraphConv can push away pairs that are not connected.
- **Notation:**

  $\mathbf{X}$: input of GraphConv
  $\tilde{\mathbf{X}}$: input of PairNorm, also the output of GraphConv
  $\dot{\mathbf{X}}$: output of PairNorm

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

- **Goal:**

$$\text{TPSD}(\mathbf{X}) = \text{TPSD}(\dot{\mathbf{X}})$$

PairNorm: normalizing pairwise squared distance

- TPSD pairwise calculation: $\mathcal{O}(n^2 d)$ complexity.

- Notice that TPSD can be rewritten as:

$$\operatorname{TPSD}(\tilde{\mathbf{X}}) = \sum_{i,j \in [n]} \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2^2 = 2n^2 \left(\frac{1}{n} \sum_{i=1}^{n} \|\tilde{\mathbf{x}}_i\|_2^2 - \|\frac{1}{n} \sum_{i=1}^{n} \tilde{\mathbf{x}}_i\|_2^2 \right)$$

- First term: the mean squared length of node representations

- Second term: the squared length of node representations

PairNorm

- Operations of PairNorm: Center-and-Scale

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

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

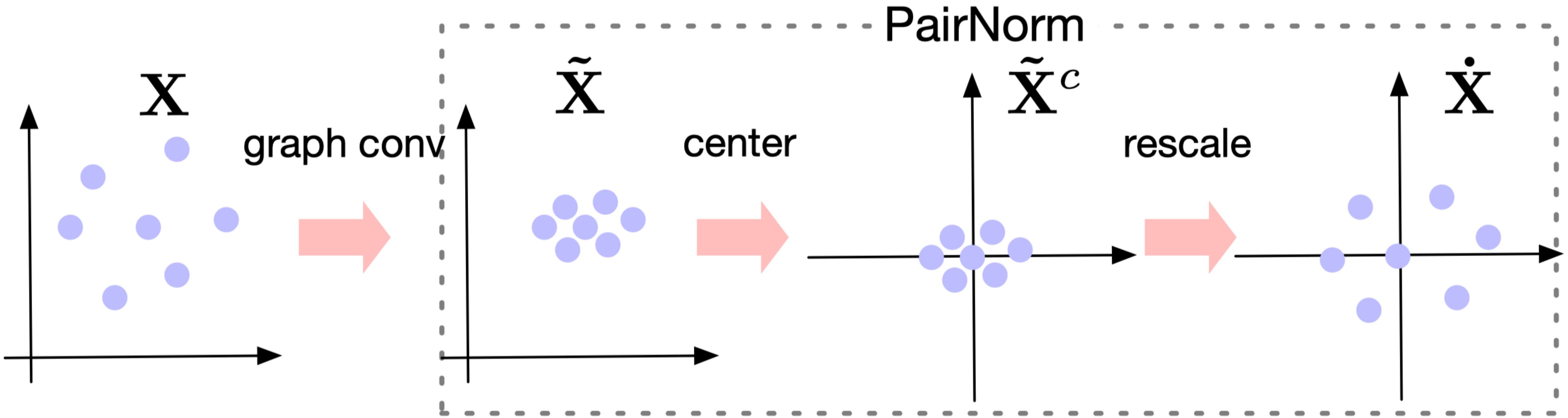
- After PairNorm:

$\dot{\mathbf{X}} := \textsc{PairNorm}(\tilde{\mathbf{X}})$ has row-wise mean $\mathbf{0}$ (i.e., is centered)

And $\text{TPSD}(\dot{\mathbf{X}}) = 2n\|\dot{\mathbf{X}}\|_F^2 = 2n^2s^2$, where **s** is a hyperparameter.

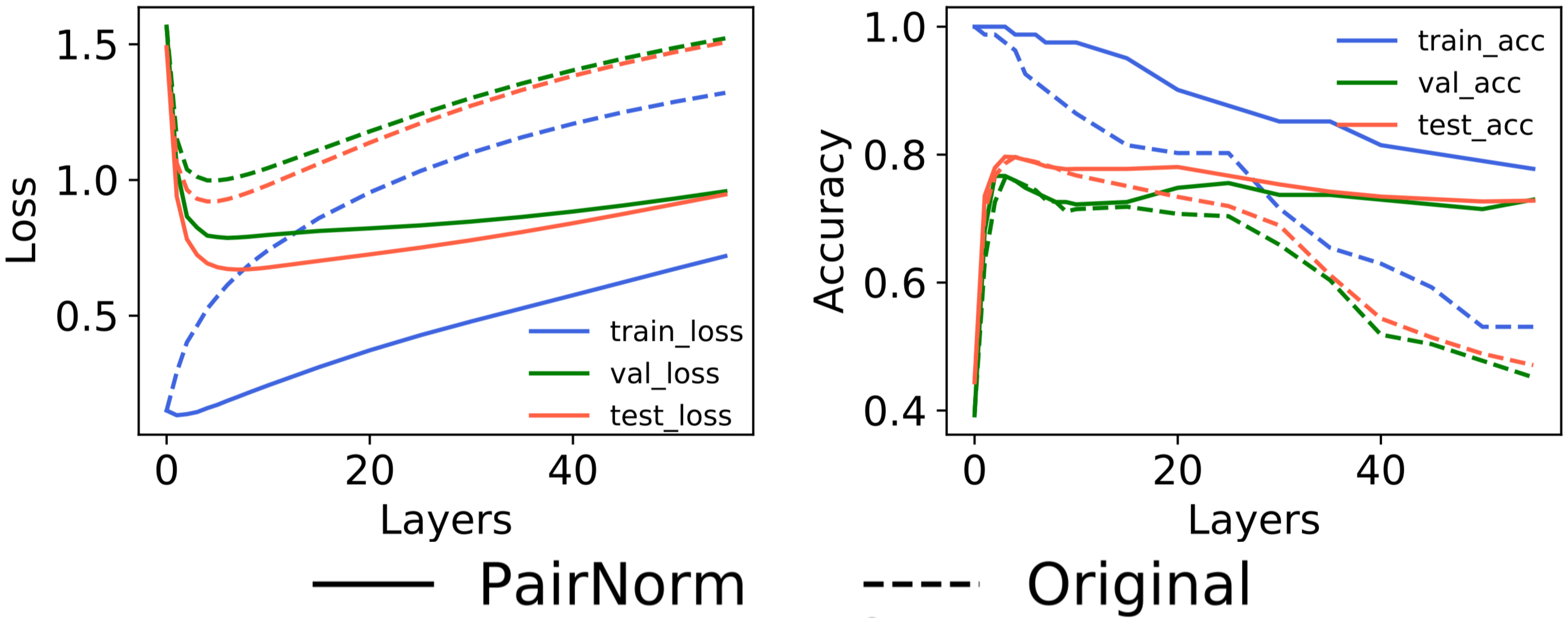
PairNorm

- Visualizing the procedure of PairNorm:



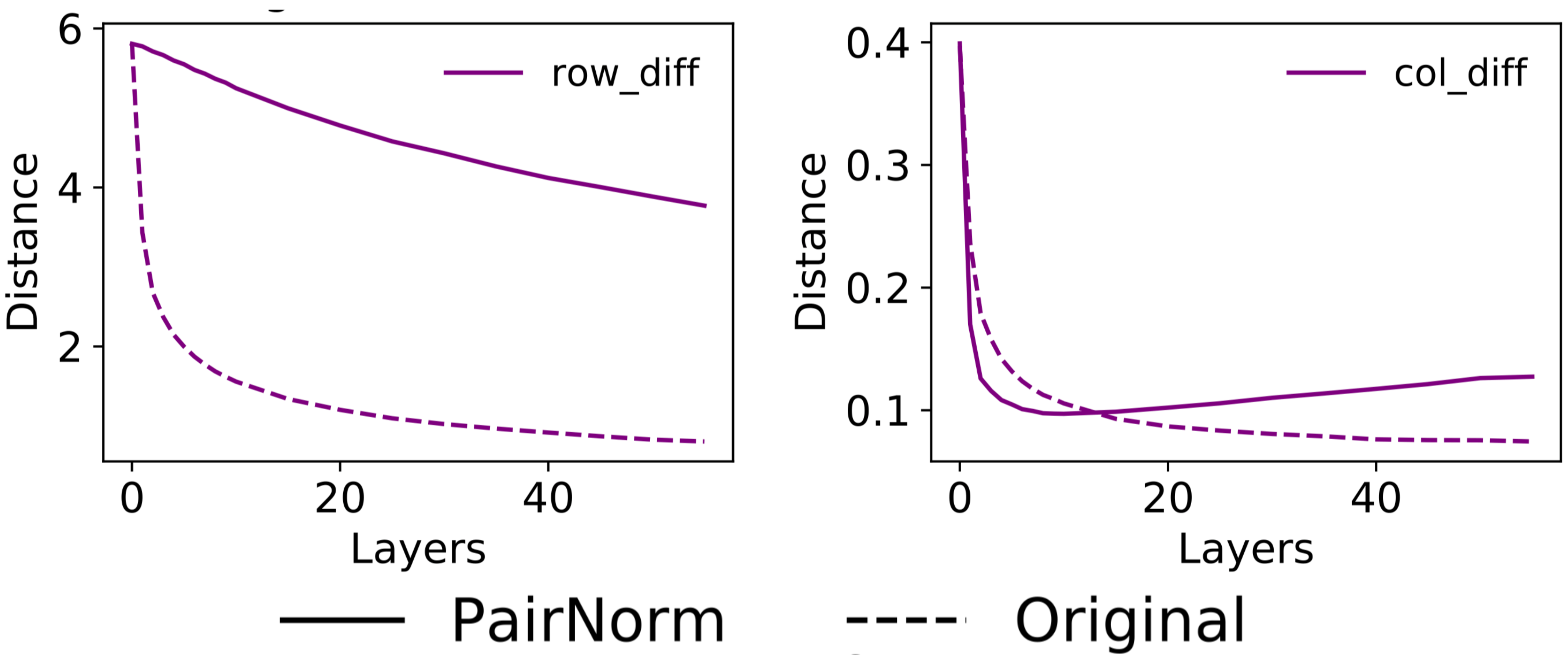
Effective of SGC+PairNorm over SSNC problem

- Cora dataset: random split with 3%train, 10%valiation, 87% test



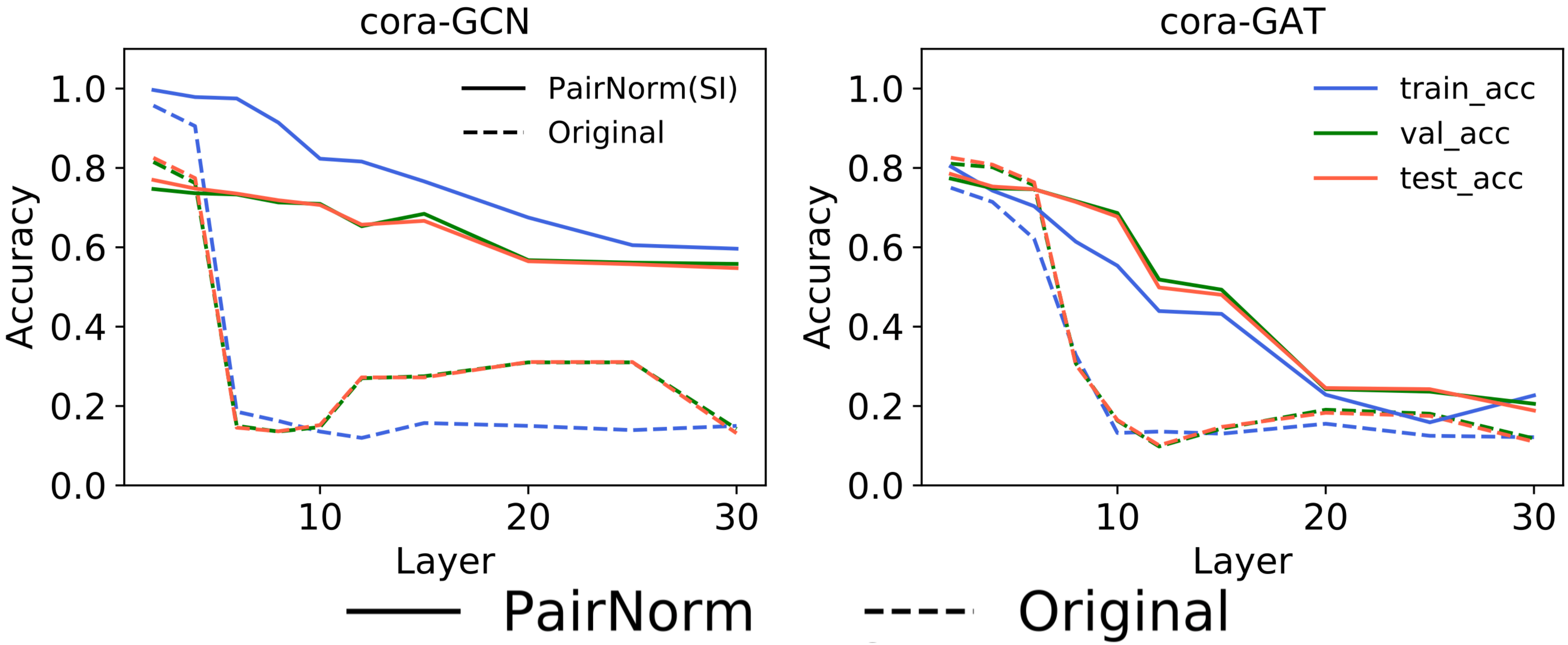
Effective of SGC+PairNorm over SSNC problem

- Cora dataset: random split with 3%train, 10%valiation, 87% test



Effective of GCN/GAT+PairNorm

- Cora dataset: **original** split with 5%train, 10%valiation, 85% test



Agenda

- Background
- Understanding oversmoothing
- PairNorm: tackling oversmoothing
- **Semi-supervised node classification with missing feature**
  - Definition and real-world applications
  - Effective of SGC+PairNorm over SSNC with missing feature

SSNC with missing feature

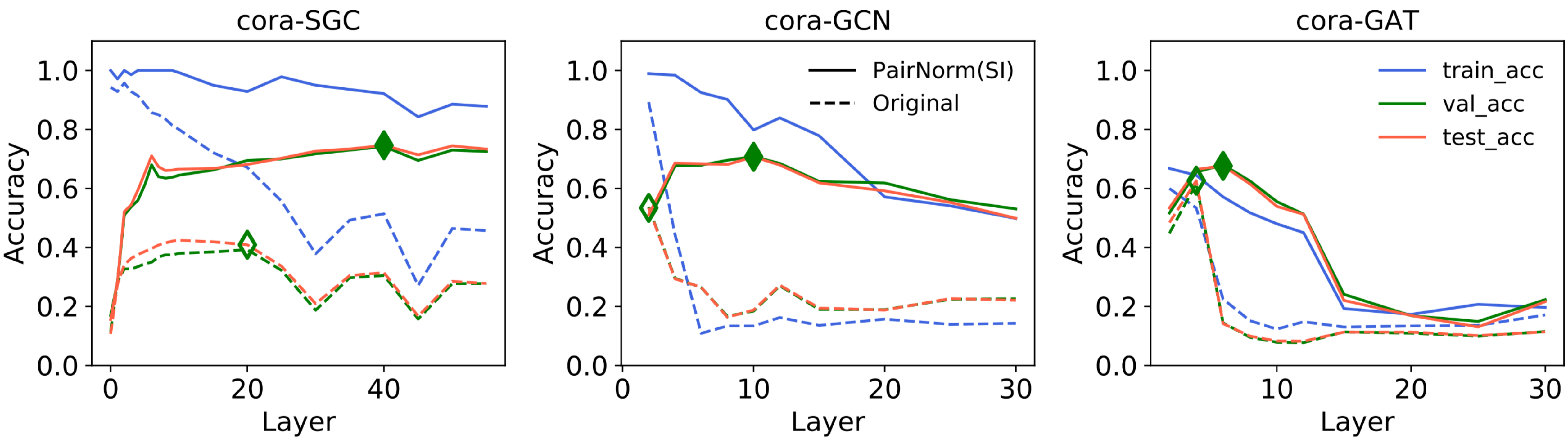
- GNN achieves best performance for SSNC problem with only 2~3 layers, where **overSmoothing** is not really happened.
- The ceiling performance of GNN over SSNC problem is not affected by oversmoothing problem.
- To demonstrate the ability of PairNorm, we investigate a new setting: SSNC with missing feature
- Assume that a certain percentage **(randomly)** of nodes don't have feature – their features are missing due to **privacy protection** or **limited records.**

SSNC with missing feature (SSNC-MF)

- Real-world applications
  1. Credit lending problem of identifying low- vs high-risk customers.
  2. Cold-start problem for graph structured data
  3. Privacy protection: company can only reveals a small fraction of data.
  4. ....
- To our knowledge, this is the first work to study SSNC with missing feature using GNN models.

Effective of SGC/GAT/GCN+PairNorm over SSNC-MF

- Cora dataset, original split. 100% missing of validation and test set.



The green diamond represent where we get highest validation accuracy

More results of SGC+PairNorm

Table 1: Comparison of 'vanilla' vs. PairNorm-enhanced SGC performance in `Cora`, `Citeseer`, `Pubmed`, and `CoauthorCS` for SSNC-MV problem, with missing rate ranging from 0% to 100%. Showing test accuracy at $\#L$ (K in Eq. 4) layers, at which model achieves best validation accuracy.

Missing Percentage		0%		20%		40%		60%		80%		100%	
Dataset	Method	Acc	#L	Acc	#L	Acc	#L	Acc	#L	Acc	#L	Acc	#L
`Cora`	SGC	**0.815**	4	**0.806**	5	0.786	3	0.742	4	0.733	3	0.423	15
	SGC-PN	0.811	7	0.799	7	**0.797**	7	**0.783**	20	**0.780**	25	**0.745**	40
`Citeseer`	SGC	0.689	10	0.684	6	**0.668**	8	**0.657**	9	0.565	8	0.290	2
	SGC-PN	**0.706**	3	**0.695**	3	0.653	4	0.641	5	**0.590**	50	**0.486**	50
`Pubmed`	SGC	0.754	1	0.748	1	0.723	4	0.746	2	0.659	3	0.399	35
	SGC-PN	**0.782**	9	**0.781**	7	**0.778**	60	**0.782**	7	**0.772**	60	**0.719**	40
`CoauthorCS`	SGC	0.914	1	0.898	2	0.877	2	0.824	2	0.751	4	0.318	2
	SGC-PN	**0.915**	2	**0.909**	2	**0.899**	3	**0.891**	4	**0.880**	8	**0.860**	20

Summary

Thank you!

- Defined and measured oversmoothing effect
- A better understanding of GNNs (still immature area)
- Proposed solution: **PairNorm**
  - Solid **theoretical analysis** over SGC
  - **General** ”patch” for any GNN layers
  - **First** normalization layer designed for GNN
- First one to investigate a new scenario, SSL with missing feature, in GNN area.