

Seminar Series on Graph Neural Networks 05

# On the problem of oversmoothing and oversquashing

Yong-Min Shin

School of Mathematics and Computing (Computational Science and Engineering)

Yonsei University

2025.05.12



수학계산학부(계산과학공학)

School of Mathematics and Computing  
(Computational Science and Engineering)



광주과학기술원

Gwangju Institute of Science and Technology

## **Towards application of graph neural networks**

Towards efficient graph learning

Explainable graph neural networks

## **Fundamental topics on graph neural networks**

On the representational power of graph neural networks

A graph signal processing viewpoint of graph neural networks

From label propagation to graph neural networks

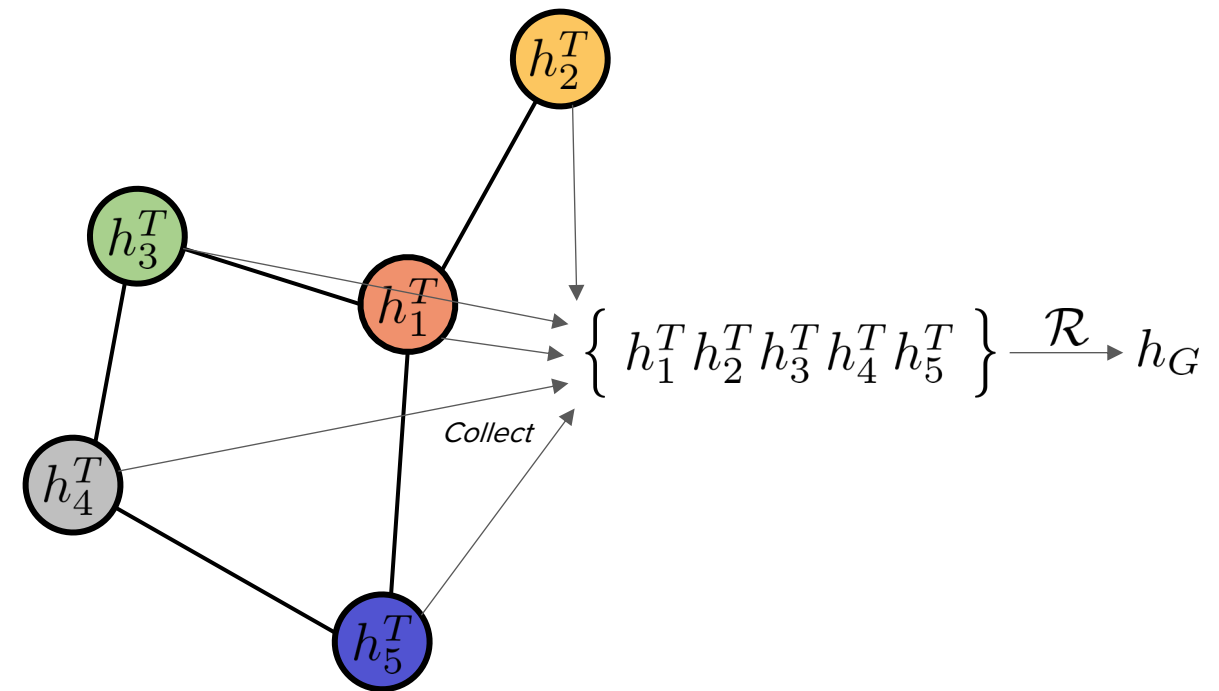
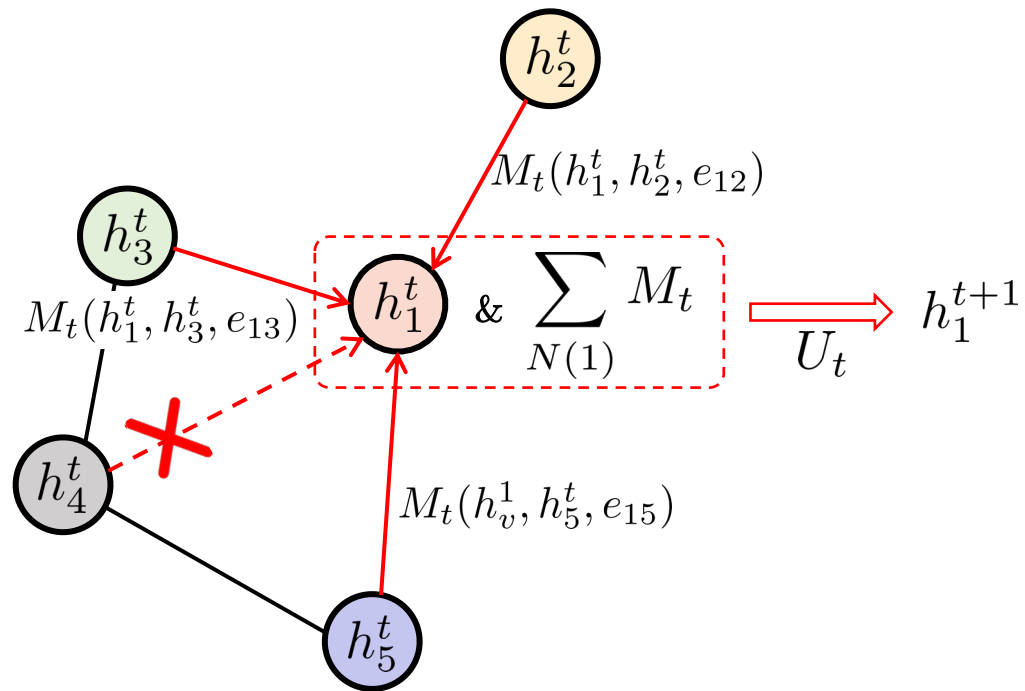
On the problem of oversmoothing and oversquashing

Introduction to graph mining and graph neural networks  
(Basic overview to kick things off)



1. Understanding **oversmoothing**
2. Understanding **oversquashing**
3. Understanding of several potential solutions: Deeper GNNs & Rewiring

# Recap: Message-passing in graph neural networks



## Definition of message passing (Gilmer)

### 1. Message passing phase (Aggregation)

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

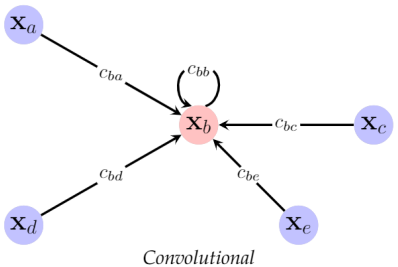
### 2. Update phase (Transformation)

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

### 3. Readout phase

$$h_G = \mathcal{R}(h_1^T, \dots, h_V^T)$$

# Recap: Message-passing in graph neural networks



Convolutional

$$\mathbf{h}_u = \phi \left( \mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} c_{uv} \psi(\mathbf{x}_v) \right)$$

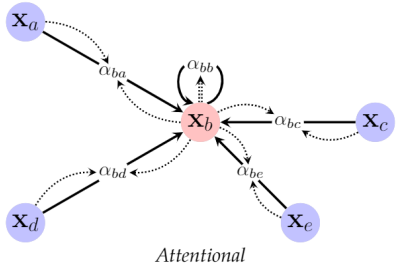
(Kipf & Welling, 2016, Defferrard et al., 2016, Wu et al., 2019)

GCN

ChebConv

SGC

**Convolutional**



Attentional

$$\mathbf{h}_u = \phi \left( \mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} a(\mathbf{x}_u, \mathbf{x}_v) \psi(\mathbf{x}_v) \right)$$

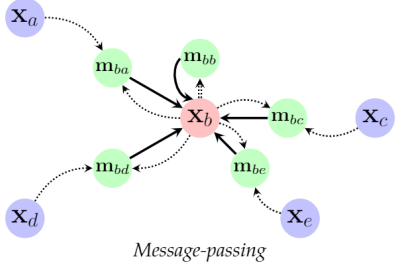
(Veličković et al., 2018, Monti et al., 2017, Zhang et al., 2018)

GAT

MoNet

GaAN

**Attentional**



Message-passing

$$\mathbf{h}_u = \phi \left( \mathbf{x}_u, \bigoplus_{v \in \mathcal{N}_u} \psi(\mathbf{x}_u, \mathbf{x}_v) \right)$$

(Gilmer et al., 2017, Battaglia et al., 2018)

MPNN

**Message passing**



*convolutional*  $\subseteq$  *attentional*  $\subseteq$  *message passing*

# GNN models are typically shallow

Looking at the following table from [1]...

**Table 8: Dataset-specific hyperparameter settings of GCN\* .**

Dataset	ResNet	Normalization	Dropout rate	GNNs layer $L$	Hidden dim	LR	epoch
Cora	False	False	0.7	3	512	0.001	500
Citeseer	False	False	0.5	2	512	0.001	500
Pubmed	False	False	0.7	2	256	0.005	500
Computer	False	LN	0.5	3	512	0.001	1000
Photo	True	LN	0.5	6	256	0.001	1000
CS	True	LN	0.3	2	512	0.001	1500
Physics	True	LN	0.3	2	64	0.001	1500
WikiCS	False	LN	0.5	3	256	0.001	1000

Widely used (\*homophilic) benchmark datasets in graph learning

In most cases, the **best performers stack either 2 or 3 GCN layers**

But it is typical to stack a lot of layers that results in deep models for better performance...

**What happens in GNNs when we start to stack multiple layers?**

[1] Luo et al., Classic GNNs are strong baselines: Reassessing GNNs for node classification, NeurIPS 2024 (Appendix A, Table 8)

\*Homophilic: Nodes connected by edges are likely to be similar/have save class labels (refer to previous session)

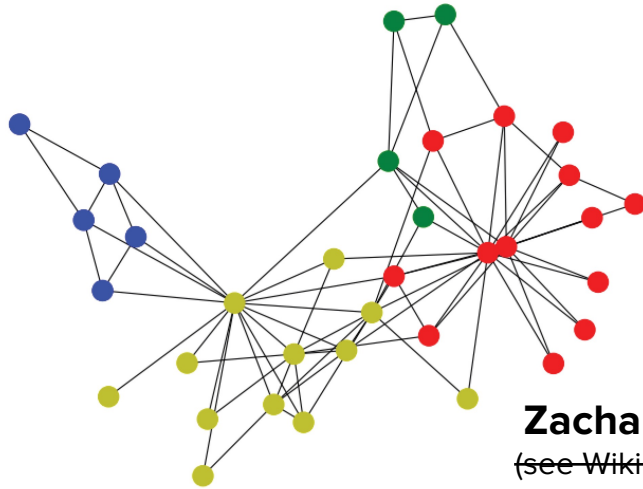
## **Understanding oversmoothing**

Li et al., Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning, AAAI 2018

\*Oono & Suzuki, Graph neural networks exponentially lose expressive power for node classification, ICLR 2020  
is also often cited on the matter of oversmoothing

# What is oversmoothing?

A phenomenon which the all node representations becomes indistinguishable as GNNs get deeper

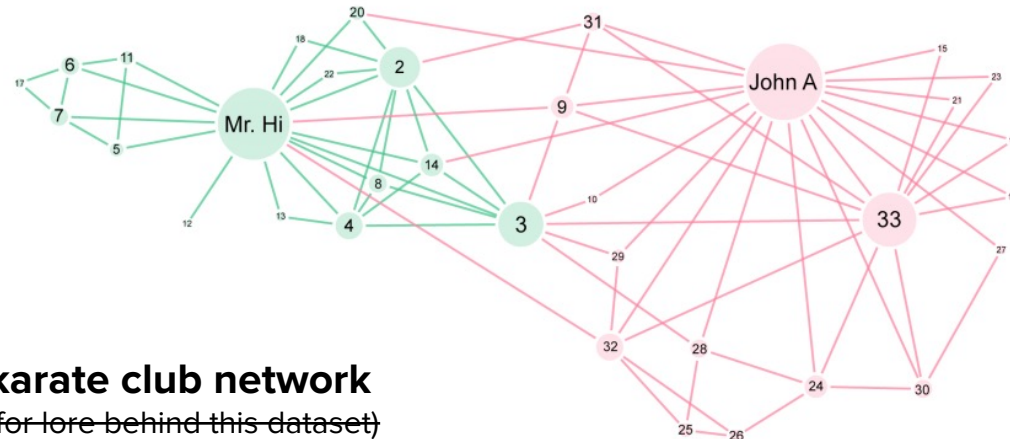


**Zachary's karate club network**

(see Wikipedia for lore behind this dataset)

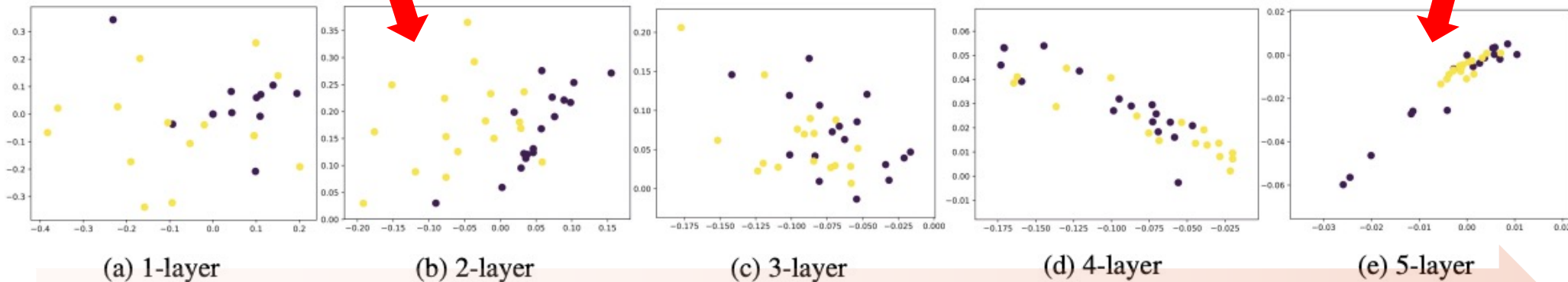
(Left visualization: My own)

(Right visualization: <https://studentwork.prattsi.org/infovis/labs/zacharys-karate-club/>)



Good separation in the early stages show that graph structure plays a critical role as long as information is mixed within the same cluster / class

Node embedding vectors starts to converge as the GCN model deepens, eventually reducing performance (this is true even with model training)



Visualization of node embedding outputs, using an untrained GCN



# Where does the term over “smoothing” come from?

**Taubin smoothing:** Taubin, A signal processing approach to fair surface design, SIGGRAPH 1995

## ABSTRACT

In this paper we describe a new tool for interactive free-form fair surface design. By generalizing classical discrete Fourier analysis to two-dimensional *discrete surface signals* – functions defined on polyhedral surfaces of arbitrary topology –, we reduce the problem of surface smoothing, or fairing, to low-pass filtering. We describe a very simple surface signal low-pass filter algorithm that applies to surfaces of arbitrary topology. As opposed to other existing optimization-based fairing methods, which are computationally more expensive, this is a linear time and space complexity algorithm. With this algorithm, fairing very large surfaces, such as those obtained from volumetric medical data, becomes affordable. By combining this algorithm with surface subdivision methods we obtain a very effective fair surface design technique. We then extend the analysis, and modify the algorithm accordingly, to accommodate different types of constraints. Some constraints can be imposed without any modification of the algorithm, while others require the solution of a small associated linear system of equations. In particular, vertex location constraints, vertex normal constraints, and surface normal discontinuities across curves embedded in the surface, can be imposed with this technique.

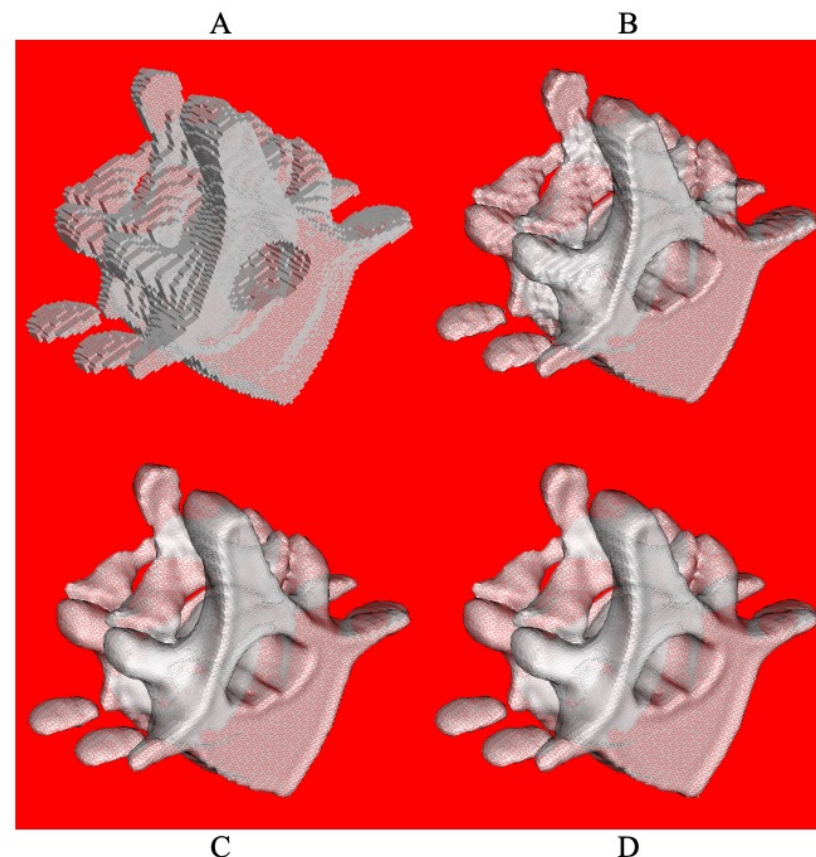
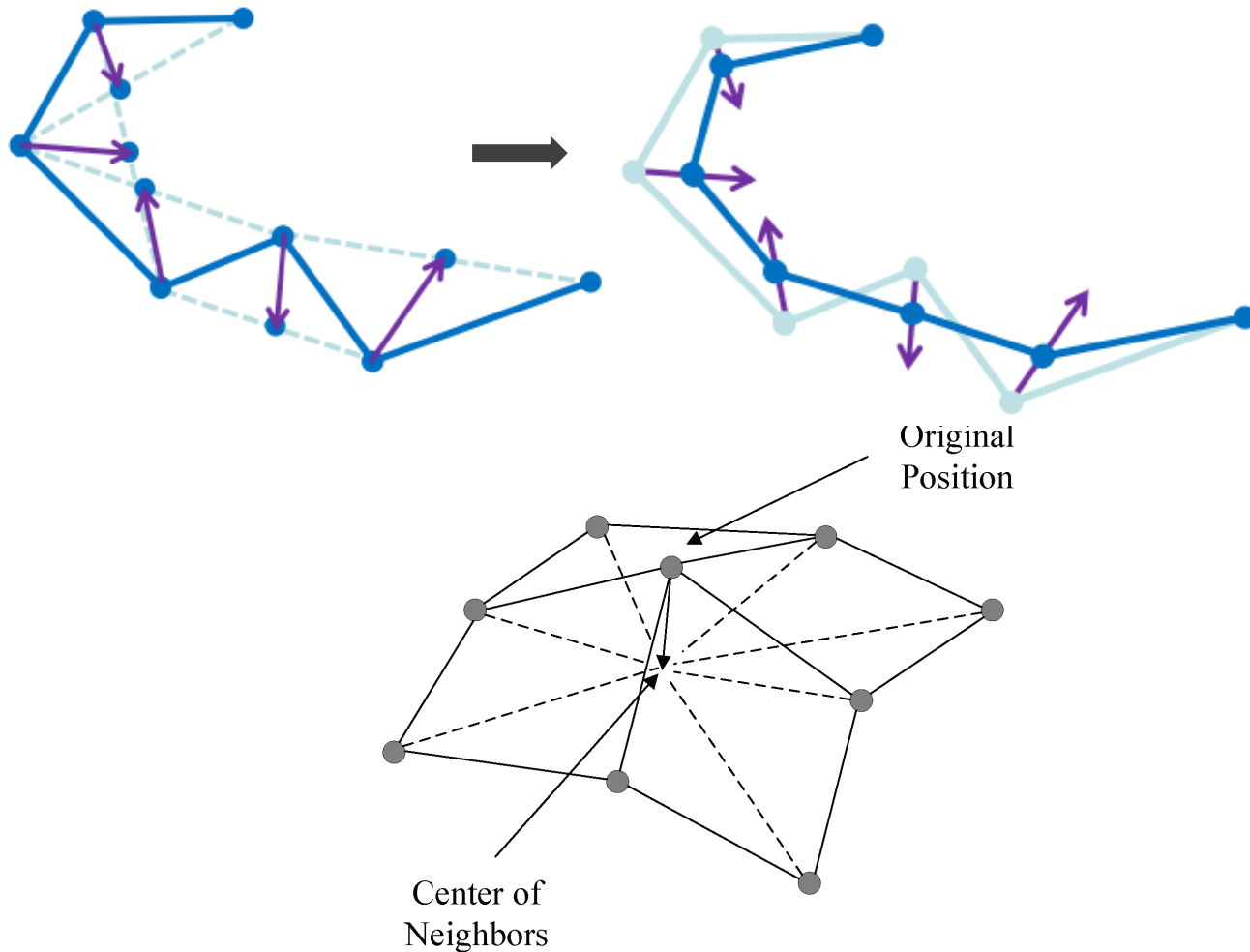


Figure 4: (A) Boundary surface of voxels from a CT scan. (B) Surface (A) after 10 non-shrinking smoothing steps. (C) Surface (A) after 50 non-shrinking smoothing steps. (D) Surface (A) after 100 non-shrinking smoothing steps.  $k_{PB} = 0.1$  and  $\lambda = 0.6307$  in (B), (C), and (D). Surfaces are flat-shaded to enhance the faceting effect.

# Where does the term over “smoothing” come from?

## \*Laplacian smoothing



$$\hat{\mathbf{x}}_i = \underbrace{(1 - \gamma)}_{\text{Mixing coefficient}} \underbrace{\mathbf{x}_i}_{\text{Self value}} + \gamma \sum_{j \in \mathcal{N}(i)} \underbrace{\frac{a_{ij}}{|\mathcal{N}(i)|}}_{\text{Average of neighbor values}} \mathbf{x}_j$$

- \*Laplacian smoothing is used to *smooth* a polygonal mesh
- Iterative process and scalable to large meshes
- The value of each point is determined by average of its neighbors
- Taubin smoothing: Laplacian smoothing + prevent shrinking

Figure (top example): [https://graphics.Stanford.edu/courses/cs468-12-spring/LectureSlides/06\\_smoothing.pdf](https://graphics.Stanford.edu/courses/cs468-12-spring/LectureSlides/06_smoothing.pdf)

Figure (bottom example): Xiao et al., Efficient parallel algorithms for 3D Laplacian smoothing on the GPU, Applied Sciences 9(24) (2019)

\*(Li et al., AAAI 2018) mentions Laplacian smoothing and cites (Taubin, SIGGRAPH 1995), but Laplacian smoothing and Taubin smoothing are different methods (similar though)

# Why does oversmoothing happen in GNNs?

1. Graph convolutions are a special form of Laplacian smoothing.

$$\hat{\mathbf{x}}_i = (1 - \gamma)\mathbf{x}_i + \gamma \sum_{j \in \mathcal{N}(i)} \frac{a_{ij}}{|\mathcal{N}(i)|} \mathbf{x}_j \quad (\text{Laplacian smoothing})$$



$$\hat{X} = X - \gamma D^{-1} L X = (I - \gamma D^{-1} L) X \quad (\text{Laplacian smoothing, matrix form})$$



let  $\gamma = 1$

$$\hat{X} = D^{-1} A X \quad (\text{Row-normalized adjacency matrix})$$



$$\hat{X} = D^{-1/2} A D^{-1/2} X \quad (\text{Symmetrically normalized adjacency matrix})$$



**Graph convolution is a special form of Laplacian smoothing**

2. Repeated Laplacian smoothing results in **values for each vertices to converge to the same values**.

Theoretical analysis: Linear GCN

**Theorem (Li et al., 2018) (simplified)** For a graph with a single component, for any  $\mathbf{w} \in \mathbb{R}^n$  and  $\alpha \in (0, 1]$ ,

$$\lim_{m \rightarrow +\infty} (I - \alpha L)^m \mathbf{w} = \theta$$

for a fixed vector  $\theta$ .



“...by repeatedly applying Laplacian smoothing many times, the features of vertices within each connected component of the graph will converge to the same values”



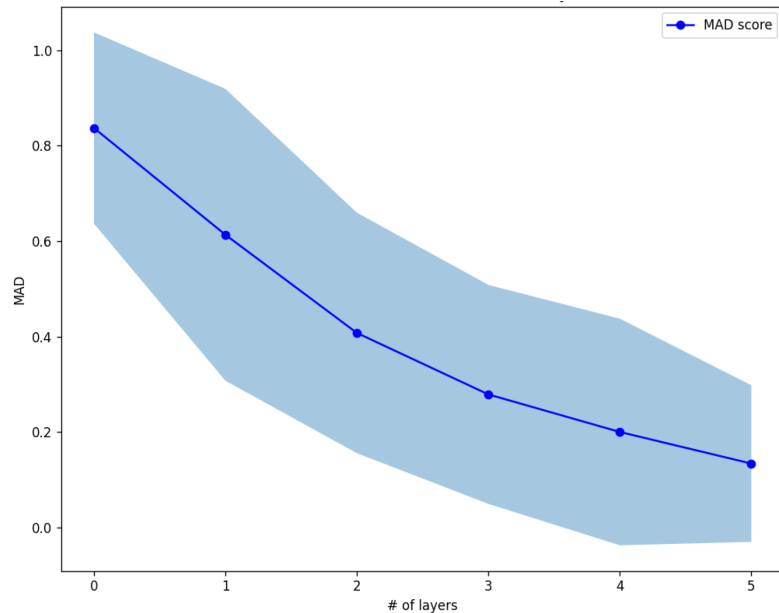
“Makes the features indistinguishable and hurt classification accuracy”

# How to measure oversmoothing?

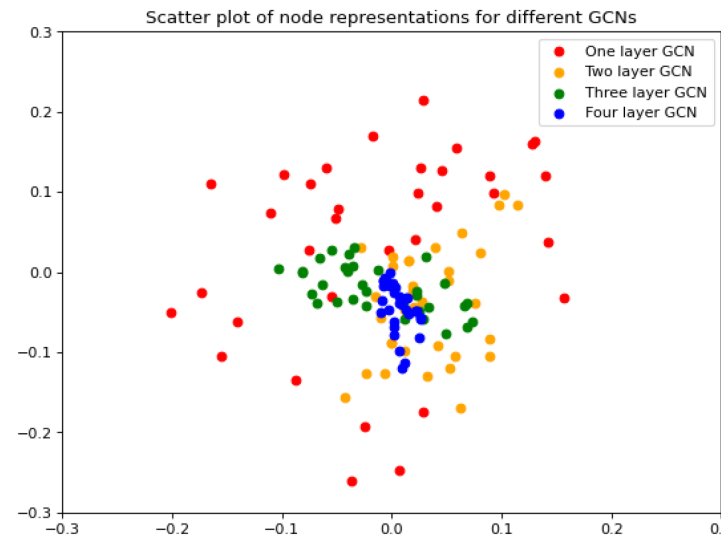
Chen et al., "Measuring and relieving the over-smoothing problem for graph neural networks from the topological view", AAAI 2020

$$\text{MAD}(H) = \frac{1}{|E|} \sum_{(i,j) \in E} D_{ij}$$

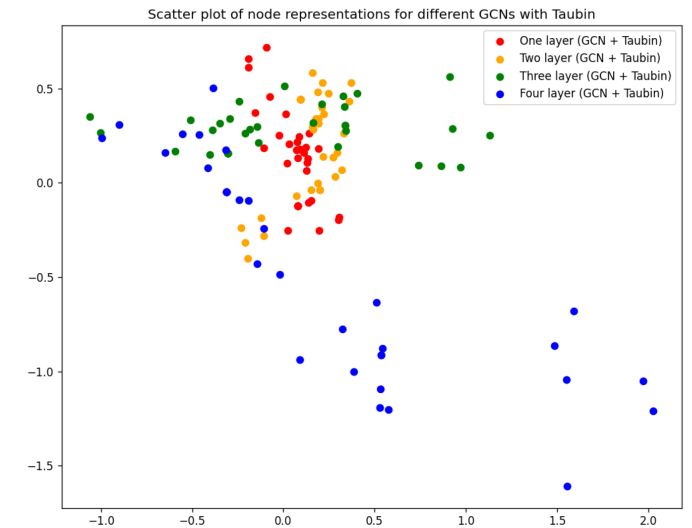
Mean Average Distance (MAD) score: Measures the average of the cosine similarity between all node representation pairs connected by an edge



**Observation 1:** MAD scores decrease as the number of layers increase for an untrained GCN on the Karate club dataset (averaged over 10 times)

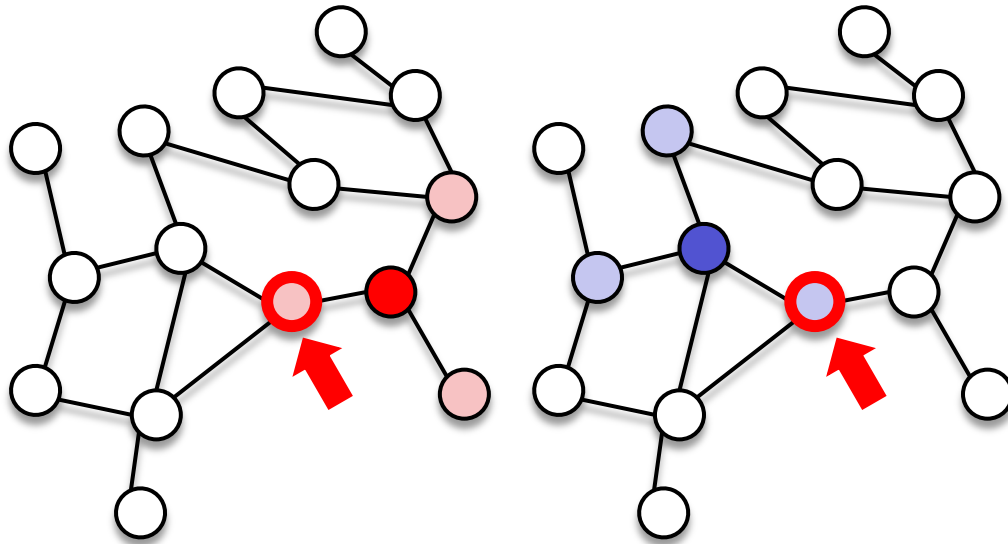


**Observation 2:** Oversmoothing happens naturally as we stack GCN layers (left, untrained, Karate club dataset). Conceptually, Taubin smoothing can prevent such collapse, which we can observe (right, same experiment setting).

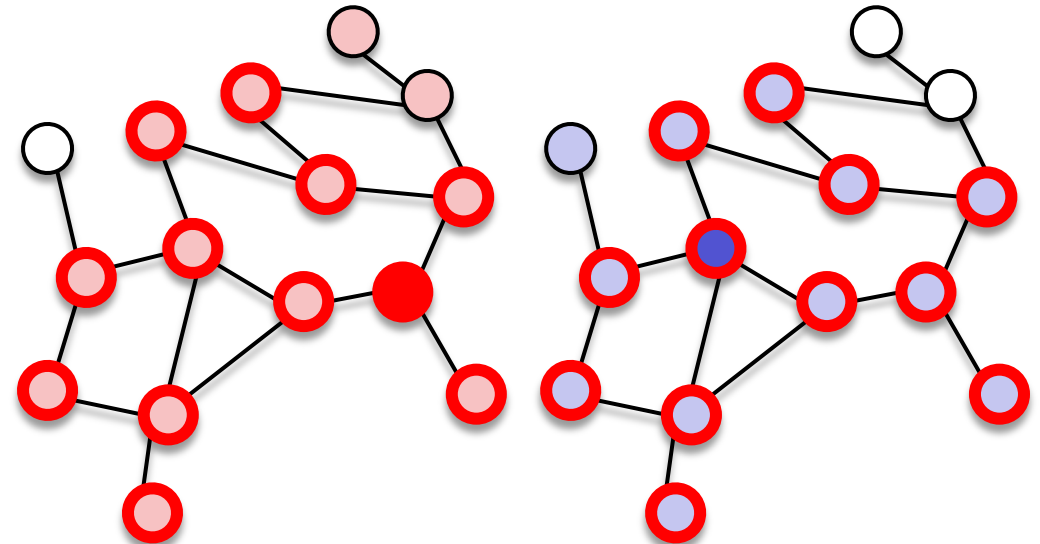


# Intuitively understanding oversmoothing

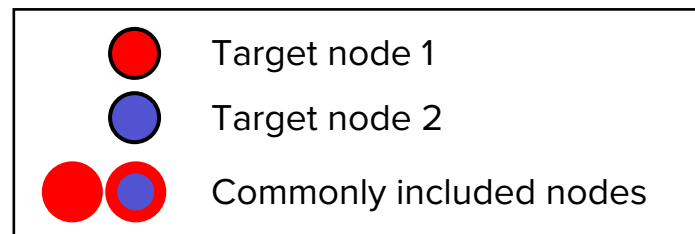
Q. What nodes are included in the computation of each target node?



1 layer GNN



3 layer GNN



Intuitively, we can understand the problem of oversmoothing as follows:

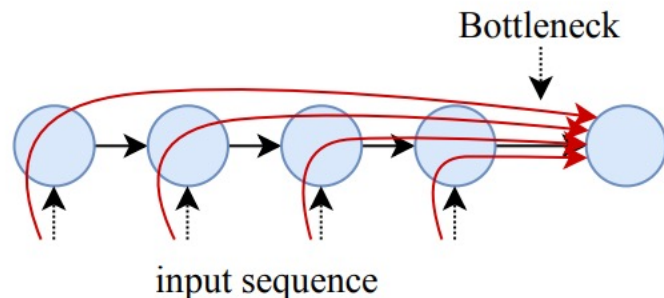
- (1) As the number of layers increase, it is more likely that the portion of commonly included nodes to also increase
- (2) ...which makes the GNN model more difficult to distinguish since the input information becomes more similar

# **Understanding oversquashing**

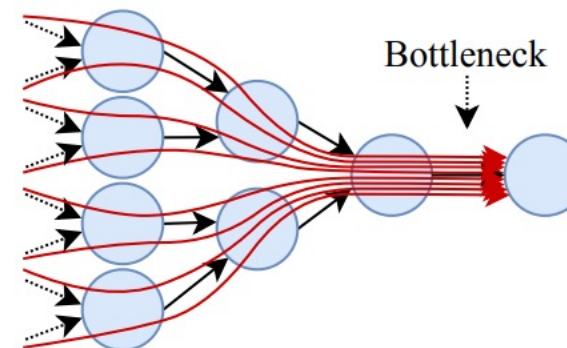
Alon & Yahav, On the bottleneck of graph neural networks and its practical implications, ICLR 2021

# What is oversquashing?

## Too much information to process for deeper GNNs



(a) The bottleneck of RNN seq2seq models



(b) The bottleneck of graph neural networks

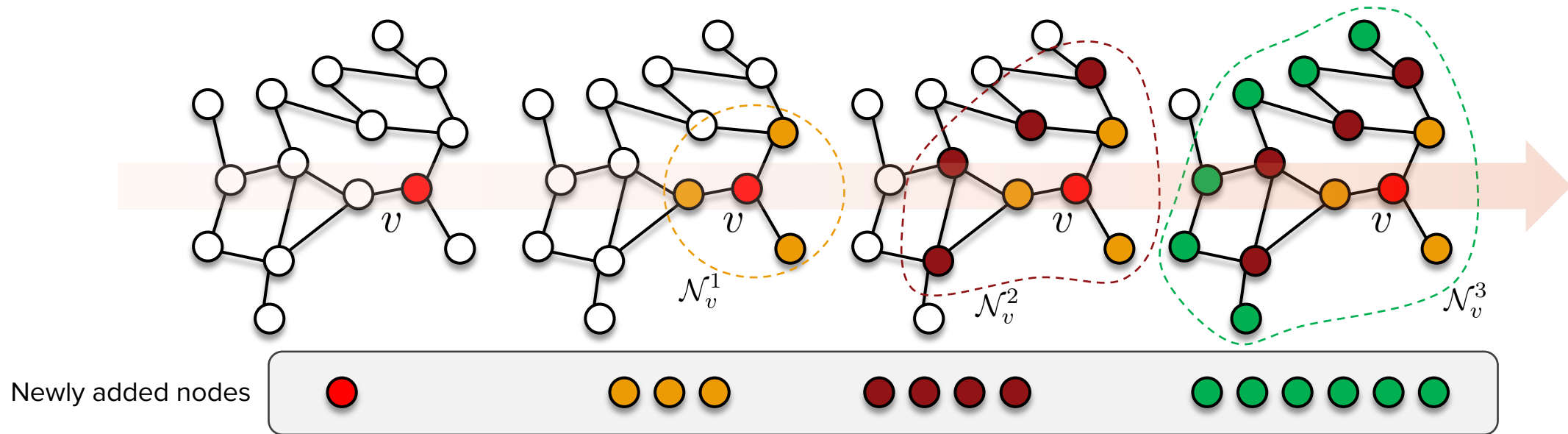
Figure 1: The bottleneck that existed in RNN seq2seq models (before attention) is strictly more harmful in GNNs: information from a node's exponentially-growing receptive field is compressed into a fixed-size vector. Black arrows are graph edges; red curved arrows illustrate information flow.

Although similar to oversmoothing, it focuses on different aspects of deep GNNs

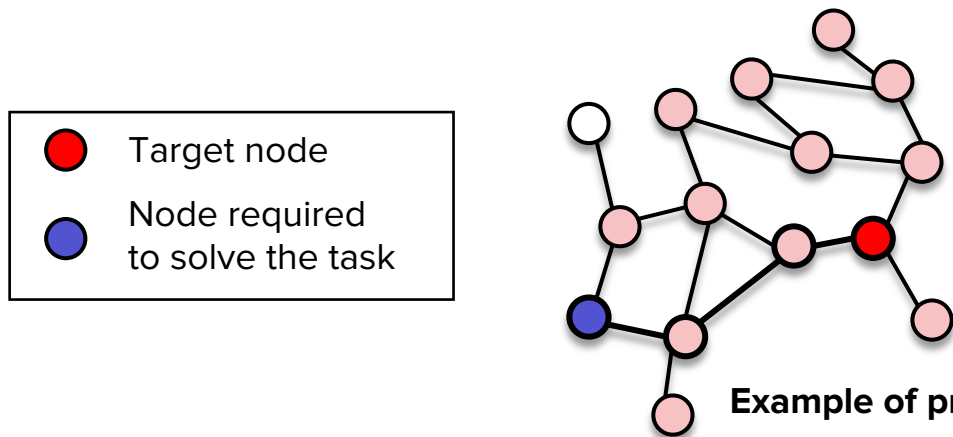
- **Oversmoothing:** Focusing on node representation collapse, observe a group of vectors.  
'Cannot differentiate between two nodes that should be different from each other'
- **Oversquashing:** Focusing on expression of information to a limited vector, observe for a single vector.  
'Too much information is squashed into a fixed-length vector, similar to RNN-type models'



# Why/When does oversquashing happen?



The number of nodes included in the computation (**receptive field**) tends to increase exponentially w.r.t. number of GNN layers. Intuitive understanding: Assume each node has an average of  $d$  neighboring nodes, then the total number of nodes will be  $d^L$  for  $L$  GNN layers.



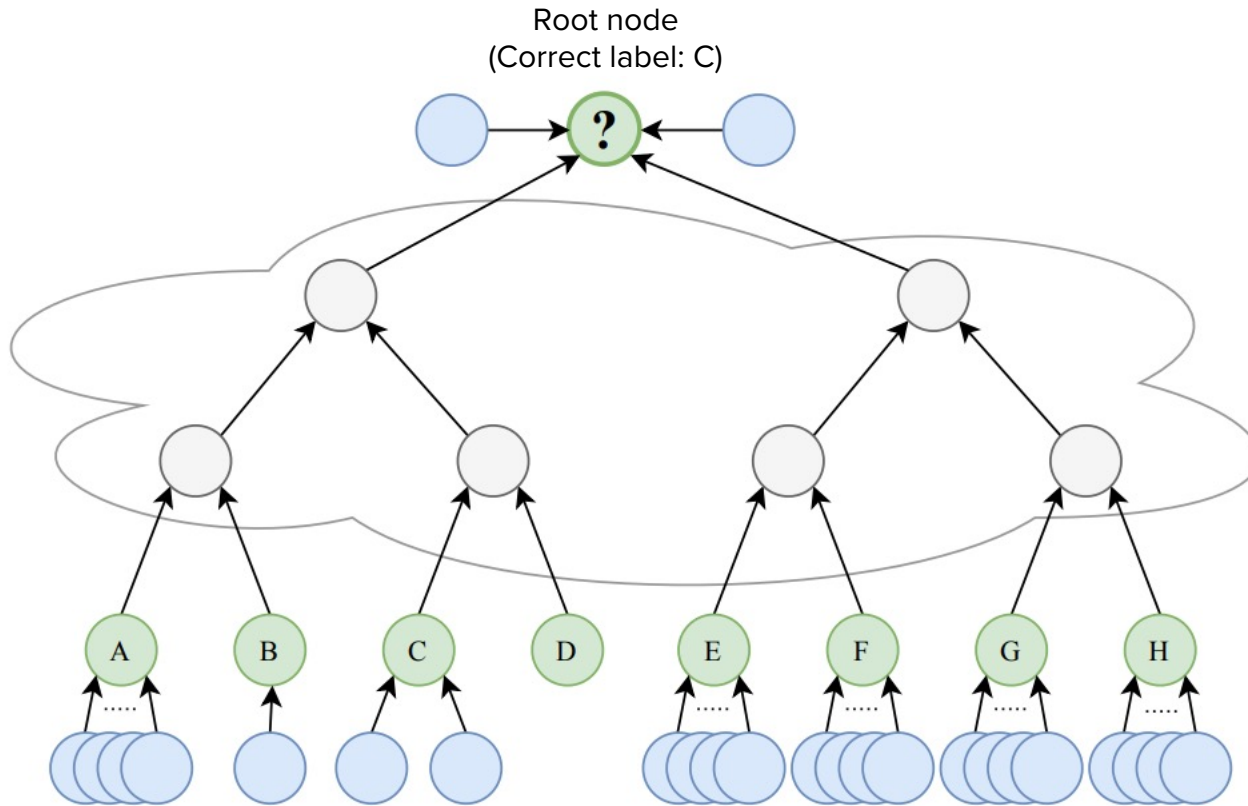
Assume a task that requires at least 3 GNN layers to solve, which pressures the GNN model to encode information of all 14 nodes to a single representation vector.

**Oversquashing** becomes a serious problem when the **problem radius** (i.e., minimal required interaction length to solve a problem) is **high** (i.e., requiring the model to incorporate long-range dependencies)



# Experiment: NEIGHBORMATCH problem

## Synthetic dataset that simulates long-range dependency tasks



### <Dataset design>

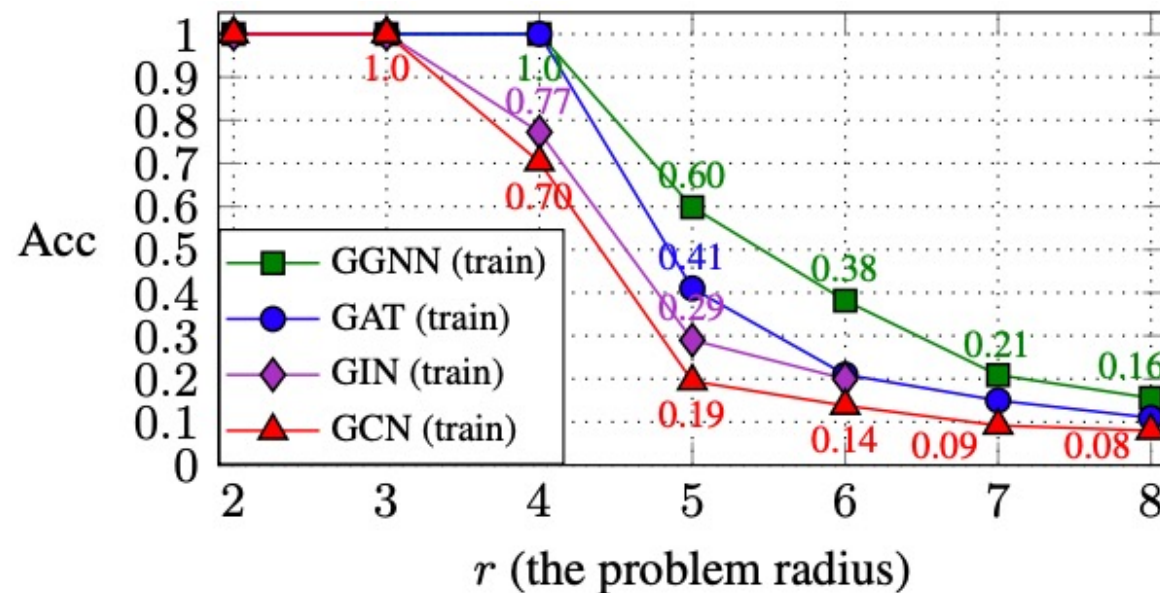
- We generate a tree-like graph where the task is to classify the correct label for the root node (see node with ?).
- Each leaf nodes are connected with a certain number of blue nodes.
- In the dataset, the **correct label to the root node** is defined as the label of the leaf node with the same blue nodes as the root node.
- In the example on the left, the correct label for the root node is C, since the **root node has 2 blue nodes**, and the C leaf node **also has 2 blue nodes**.

### <Implications to the GNN model>

- The GNN model generates an embedding vector for the root node.
- During this procedure, the GNN model **must aggregate information of all nodes into a single vector** (i.e., embedding vector for the root node), causing an **oversquashing** problem.
- Also, since the depth of the tree explicitly determines the problem radius, the number of layers of the GNN model must match this number.

# Experiment: NEIGHBORMATCH problem

Experiment directly shows the effect of oversquashing



- The number of GNN layers is always set to the minimal number of layers required to solve the problem.
- As the problem radius of NEIGHBORMATCH increases, **oversquashing starts to kick in** and therefore the performance drops significantly even though the minimal required number of GNN layers are met.

# **Understanding of several potential solutions: Deeper GNNs & Rewiring**

**(and slightly more)**

# Potential solutions: Deeper GNN models

**GCNII** (Chen et al., Simple and Deep Graph Convolutional Networks, ICML 2020)

Original GCN model:  $\mathbf{H}^{(\ell+1)} = \sigma \left( \tilde{\mathbf{P}}\mathbf{H}^{(\ell)}\mathbf{W}^{(\ell)} \right)$



$$\mathbf{H}^{(l+1)} = \sigma \left( \left( (1 - \alpha_l)\tilde{\mathbf{P}}\mathbf{H}^{(l)} + \alpha_l\mathbf{H}^{(0)} \right) \left( (1 - \beta_l)\mathbf{I} + \beta_l\mathbf{W}^{(l)} \right) \right)$$

Constant skip connections  
from the initial layer

Repeated residual  
connections for each layer

# Potential solutions: Deeper GNN models

**GCNII** (Chen et al., Simple and Deep Graph Convolutional Networks, ICML 2020)

*Table 2.* Summary of classification accuracy (%) results on Cora, Citeseer, and Pubmed. The number in parentheses corresponds to the number of layers of the model.

Method	Cora	Citeseer	Pubmed
GCN	81.5	71.1	79.0
GAT	83.1	70.8	78.5
APNP	83.3	71.8	80.1
JKNet	81.1 (4)	69.8 (16)	78.1 (32)
JKNet(Drop)	83.3 (4)	72.6 (16)	79.2 (32)
Incep(Drop)	83.5 (64)	72.7 (4)	79.5 (4)
GCNII	<b>85.5 ± 0.5 (64)</b>	<b>73.4 ± 0.6 (32)</b>	<b>80.2 ± 0.4 (16)</b>
GCNII*	<b>85.3 ± 0.2 (64)</b>	<b>73.2 ± 0.8 (32)</b>	<b>80.3 ± 0.4 (16)</b>

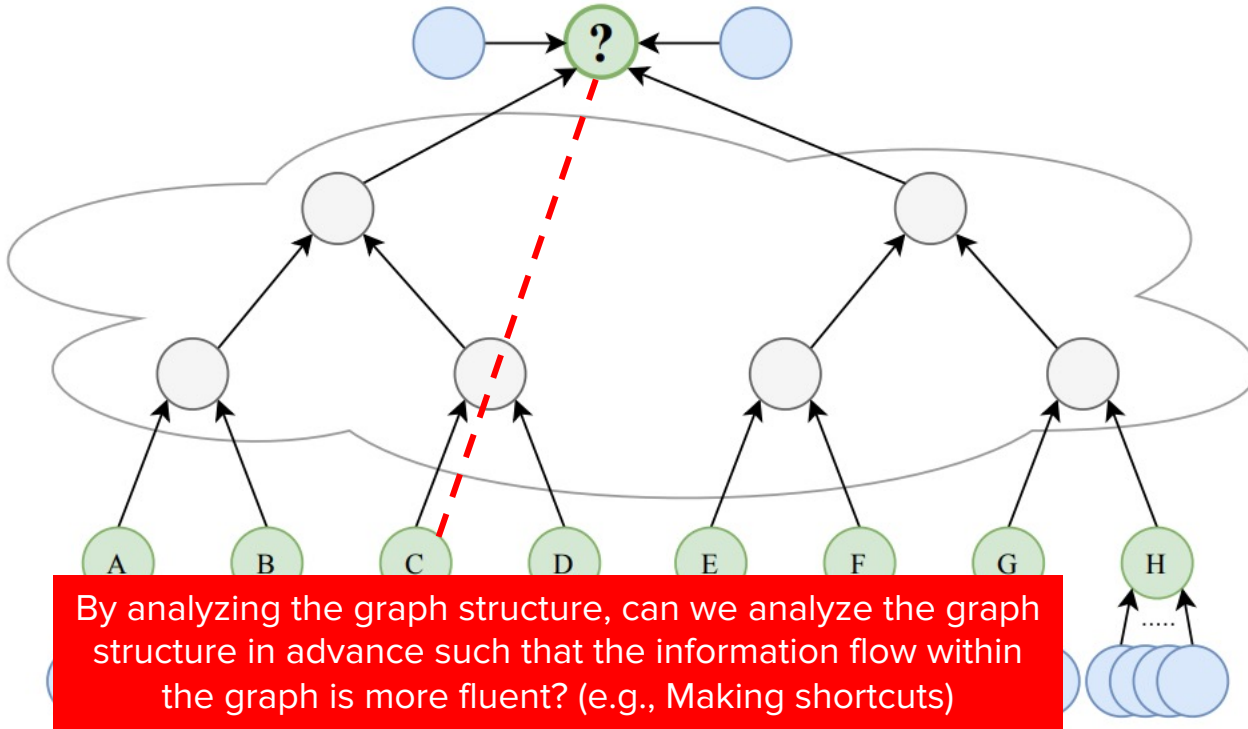
Empirical results show that the best performance can be achieved with GNN models with **much deeper configurations**.





	Squirrel	Chameleon	Amazon-Ratings	Roman-Empire	Minesweeper	Questions
Metric	Accuracy↑	Accuracy↑	Accuracy↑	Accuracy↑	ROC-AUC↑	ROC-AUC↑
<b>GCN*</b>	<b>45.01</b> ± 1.63	<b>46.29</b> ± 3.40	<b>53.80</b> ± 0.60	<b>91.27</b> ± 0.20	<b>97.86</b> ± 0.24	<b>79.02</b> ± 0.60
(-) Normalization	44.13 ± 2.03	-	53.68 ± 0.82	90.53 ± 0.33	96.94 ± 1.96	-
(-) Dropout	42.89 ± 1.28	45.28 ± 4.78	51.37 ± 0.34	85.10 ± 0.61	94.28 ± 2.29	76.58 ± 0.40
(-) Residual Connections	43.14 ± 1.82	-	51.14 ± 0.34	74.84 ± 0.62	86.45 ± 0.89	75.87 ± 4.47

The benefit of residual connections are also empirically demonstrated by (Luo et al., NeurIPS 2024)

# Potential solutions: Graph rewiring

**SDRF** (Chen et al., Understanding over-squashing and bottlenecks on graphs via curvature, ICLR 2022)

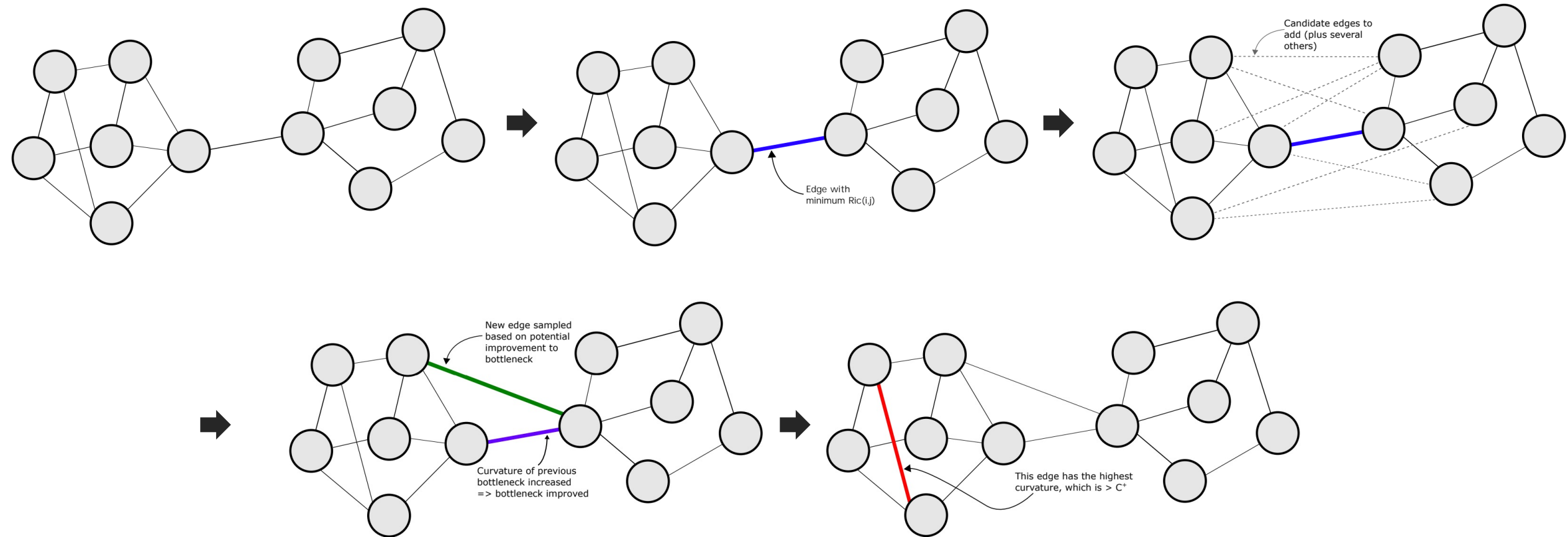


	Cycle $C_{n \geq 5}$	Grid $G_n$	Clique $K_n$	Tree $T_r$
Graph				
$\text{Ric}_G$	0	0	$\frac{n}{n-1}$	$\frac{4}{r+1} - 2$

**Theorem (informal):** Oversquashing happens along edges with low Ricci curvatures

# Potential solutions: Graph rewiring

**SDRF** (Chen et al., Understanding over-squashing and bottlenecks on graphs via curvature, ICLR 2022)



Materials heavily borrowed from the slide:

Giovanni, "On over-squashing in graph neural networks: from theoretical understanding to graph-rewiring solutions", Complex Network Analysis Discussion Group, 6 March 2023

[https://iit.demokritos.gr/wp-content/uploads/2023/04/slides\\_curvature\\_-20.pdf](https://iit.demokritos.gr/wp-content/uploads/2023/04/slides_curvature_-20.pdf)

1. Oversmoothing is a phenomenon where the node embedding vectors collapse as the GNNs go deeper
2. Oversquashing is a phenomenon where the receptive field is too big for the model to compress into a single vector
3. Two potential solutions: Deeper GNN architectures by skip/residual connections, graph rewiring as preprocessing



**Thank you!**

Please feel free to ask any questions :)

*[jordan7186.github.io](https://jordan7186.github.io)*