Advanced Deep Graph Learning: Deeper, Faster, Robuster, and Unsupervised

Yu Rong, Tingyang Xu, Yatao Bian, Junzhou Huang, Tencent AI Lab
Wenbing Huang, Fuchun Sun, Tsinghua University
Hong Cheng, The Chinese University of Hong Kong

Tutorial website: https://ai.tencent.com/ailab/ml/WWW-Deep-Graph-Learning.html

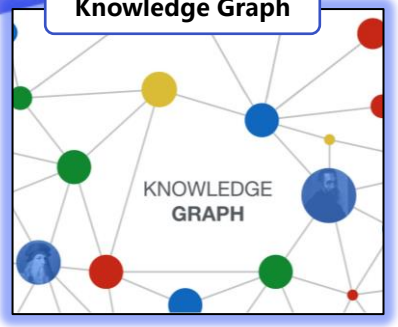
Tencent
AI Lab

清華大學
Tsinghua University

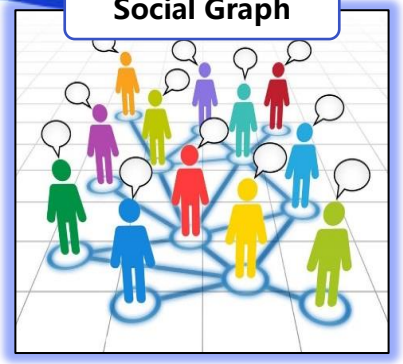
香港中文大學
The Chinese University of Hong Kong

Tencent AI Lab

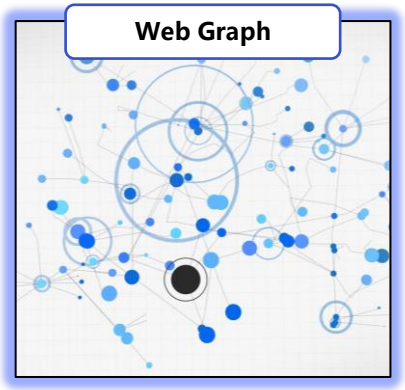
Knowledge Graph



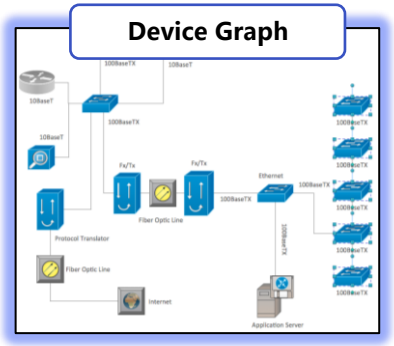
Social Graph



Web Graph

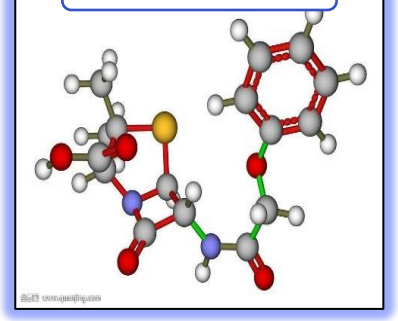


Device Graph

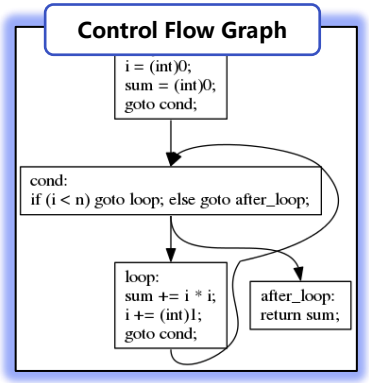


Graph is Everywhere

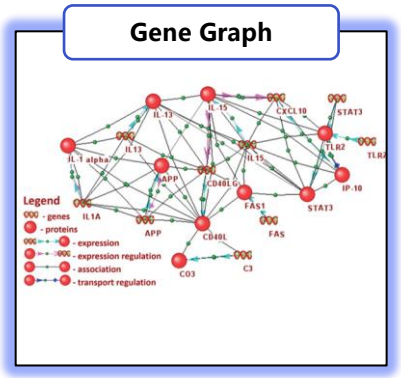
Molecular Graph



Control Flow Graph



Gene Graph

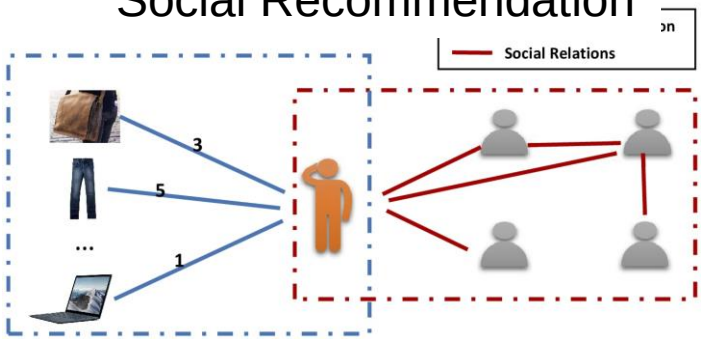


Brain Graph

Graph is Important

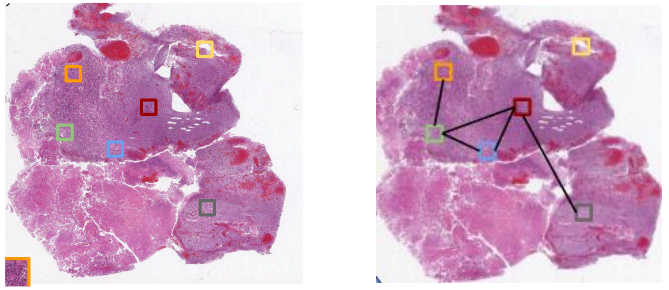
- Numerous real-world problems can be summarized as a set of tasks on graphs.

Social Recommendation



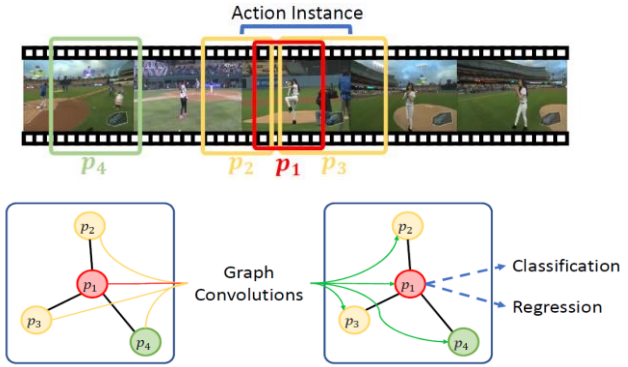
Fan W, Ma Y, Li Q, et al. 2019

Survival Analysis



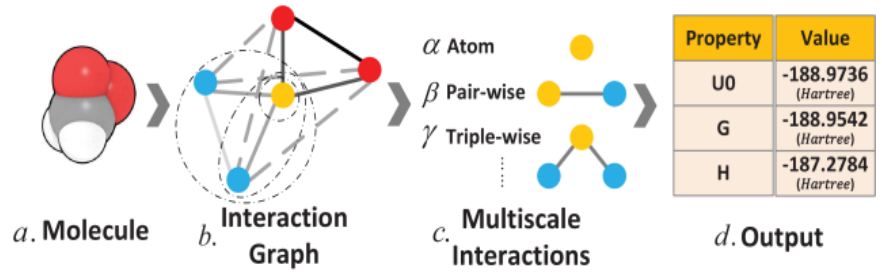
Li R, Yao J, Zhu X, et al. 2018

Temporal Action Localization



Zeng et al. 2019

Molecular Property Prediction

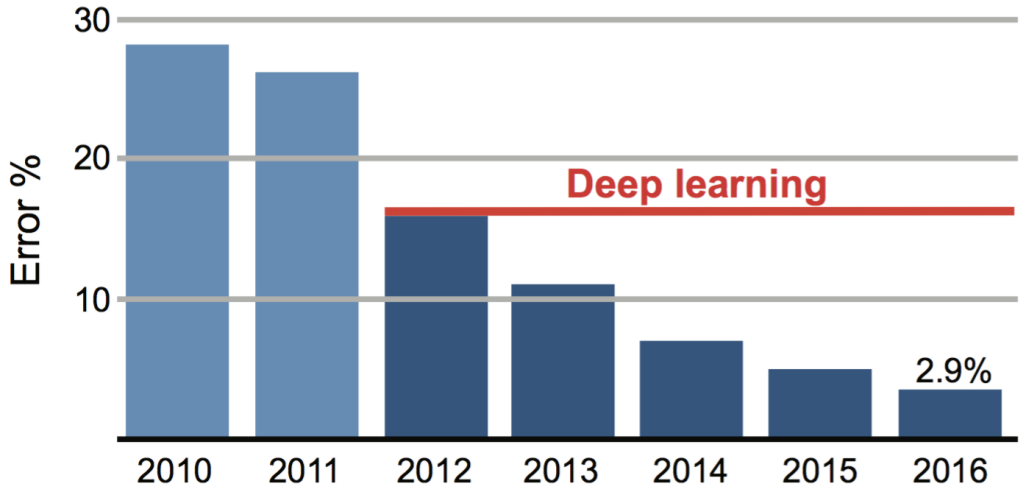


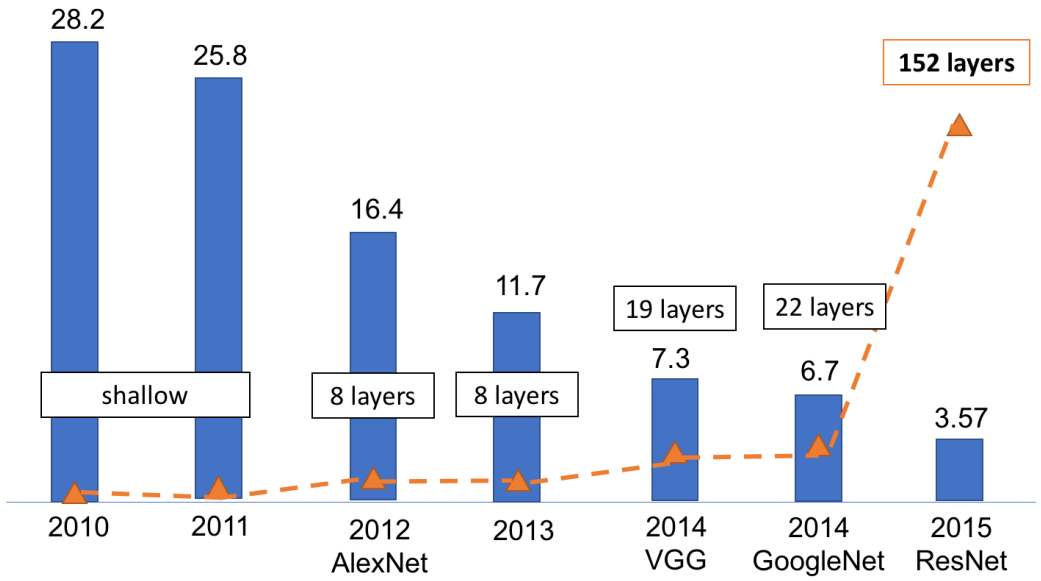
Property	Value
U0	-188.9736 (Hartree)
G	-188.9542 (Hartree)
H	-187.2784 (Hartree)

Lu C, Liu Q, Wang C, et al.

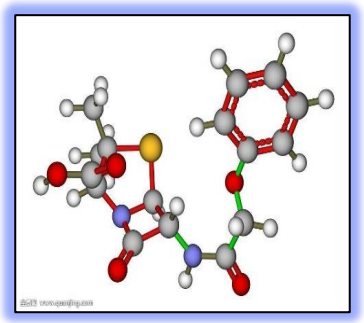
The Power of Deep Learning

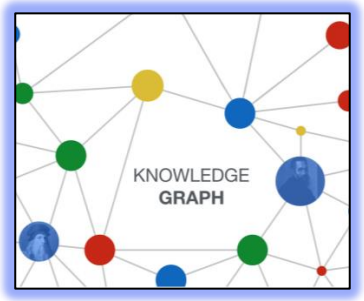
IMAGENET



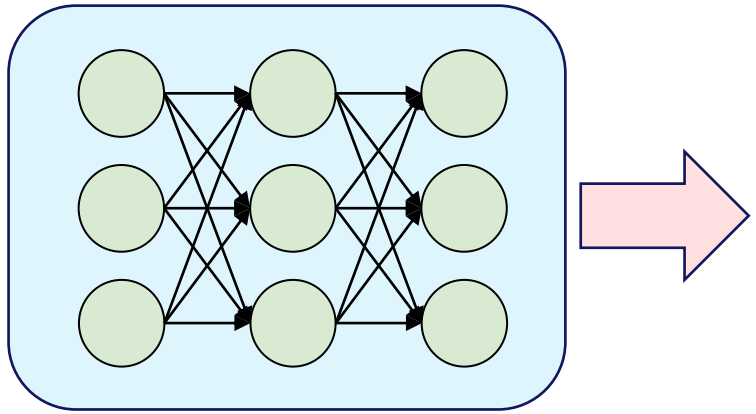


Deep Learning + Graphs = ?



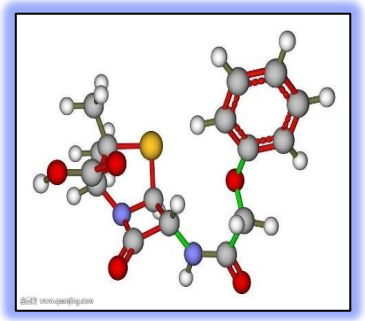


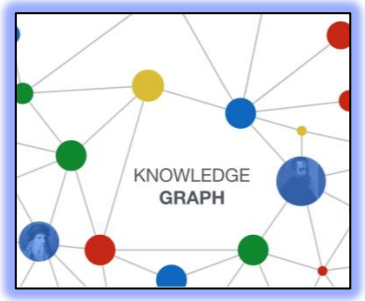
Deep Learning Model



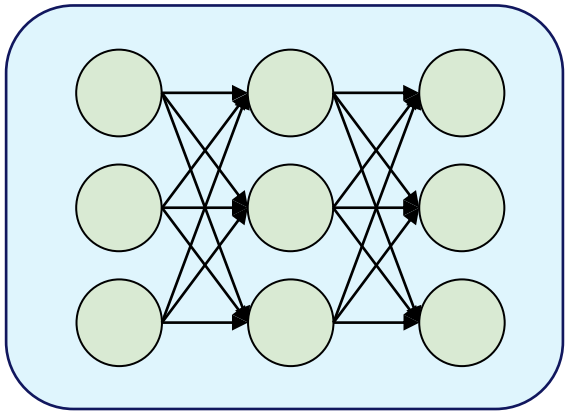
?

Deep Graph Learning

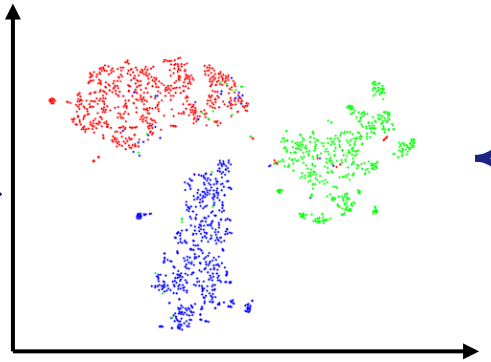




Neural Network



Graph/Node Representation

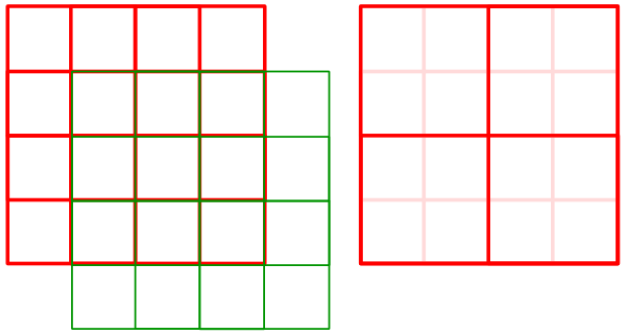


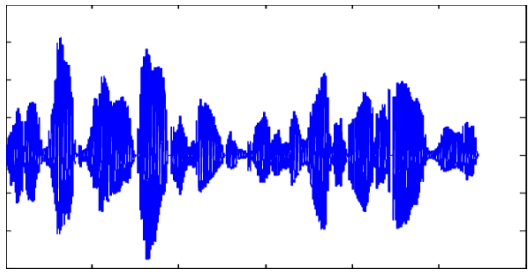
Applications

- Node Classification
- Link Prediction
- Community Detection
- Graph Generation
- ……

Make neural network model that can deal with graph data.

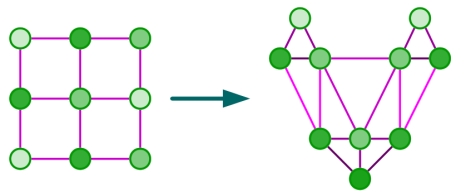
The Big Challenges



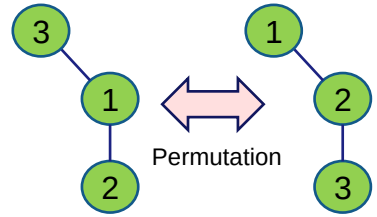


- Grid-like and sequence-like structure.
- Spital/sequential relations between pixels / units.

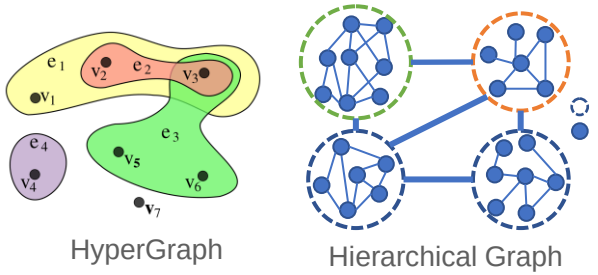
VS



Irregular



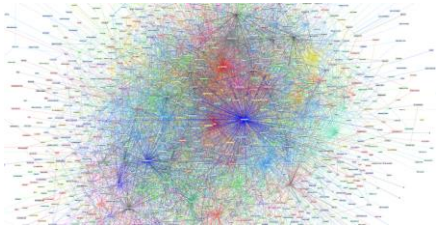
Permutation Invariant



HyperGraph

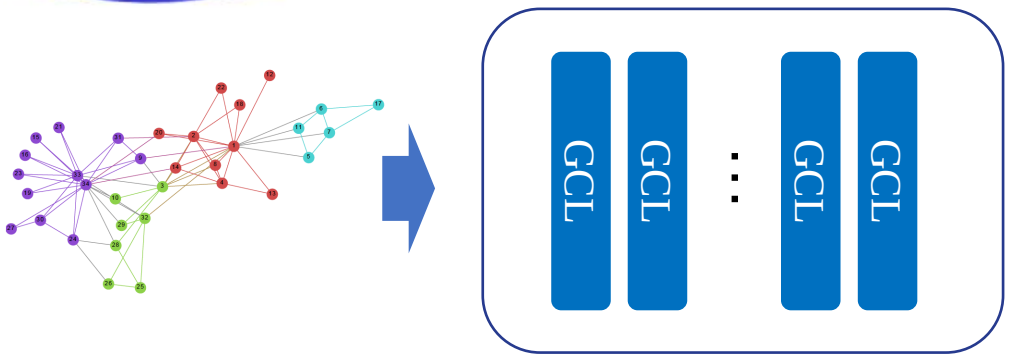
Hierarchical Graph

Complex variants

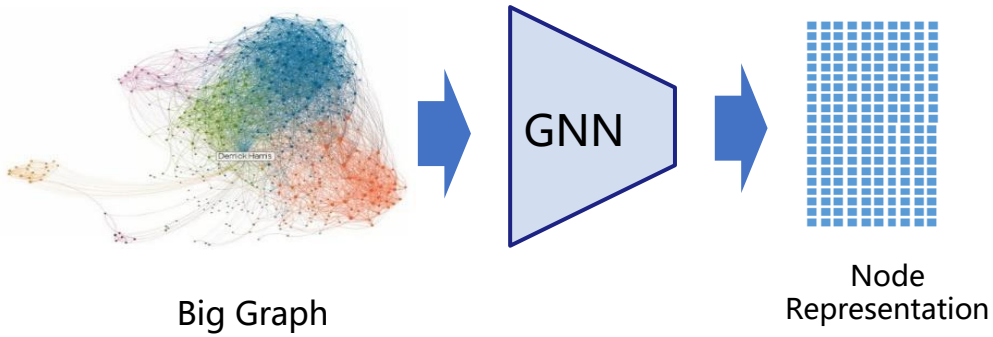


Large-scale instance

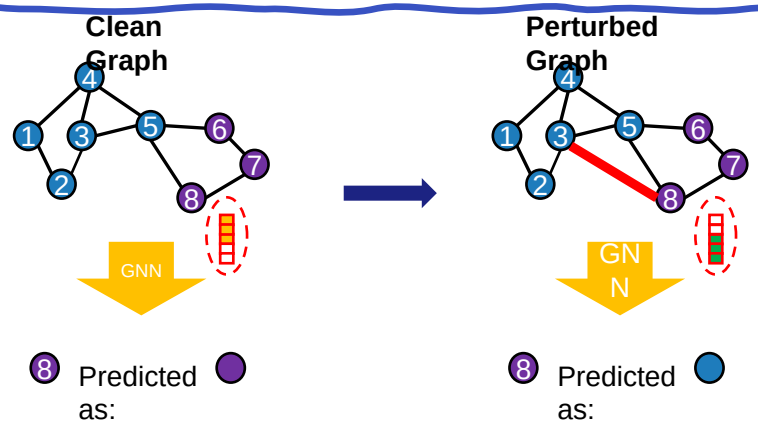
The Research Questions



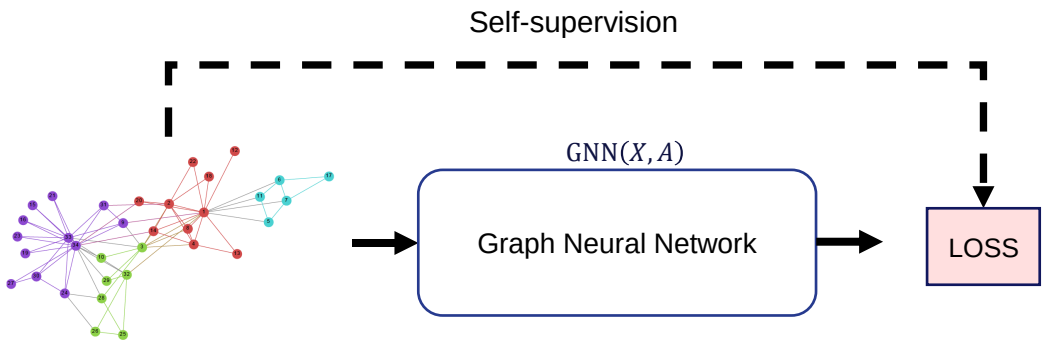
How to train the deeper graph neural networks?



How to extend the graph neural networks to process a large-scale graph?



How to enhance the robustness of graph neural network ?



How to conduce the self/un-supervised learning on graphs?

Tutorial Overview

Foundations

- Preliminary
- The Brief History of Graph Neural Networks

Advances

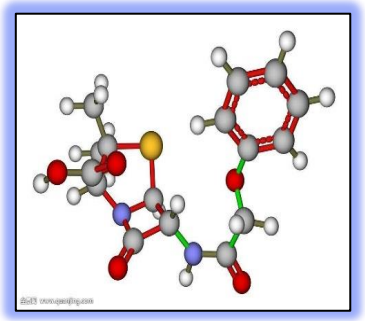
- Training Deep GNNs
- Scalability of GNNs
- Robustness of GNNs
- Self/Un-Supervised Learning of GNNs
- Other Advanced Topics

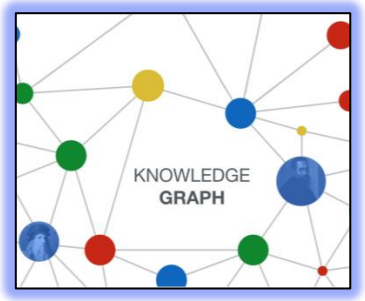
Applications

- Social Networks
- Medical Imaging

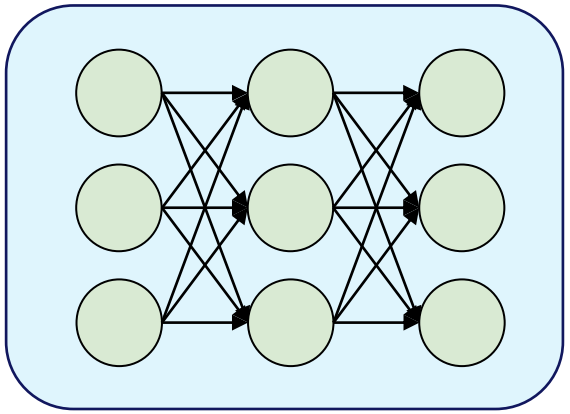
Preliminaries and Brief History of Graph Neural Networks

What is the Graph Neural Network?

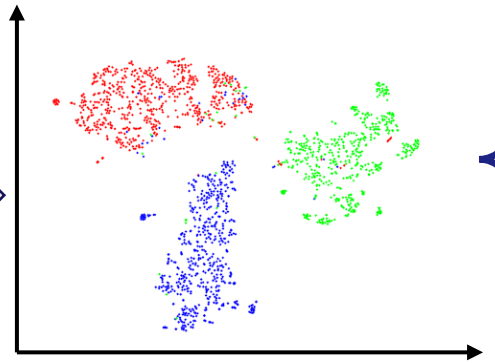




Graph Neural Network



Graph/Node Representation



Applications

- Node Classification
- Link Prediction
- Community Detection
- Graph Generation
- ……

Neural network model that can deal with graph data.

Graph Neural Network is not a New Thing

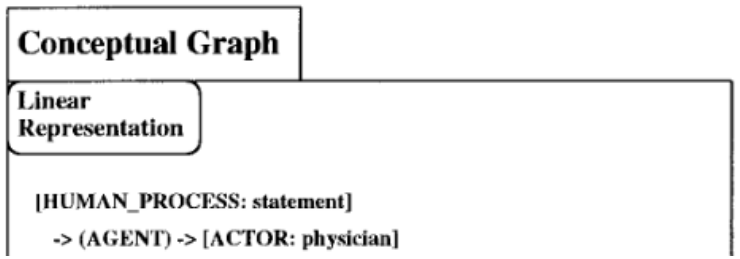
Sperduti, Alessandro and Starita, Antonina. 1997

714 IEEE TRANSACTIONS ON NEURAL NETWORKS, VOL. 8, NO. 3, MAY 1997

Supervised Neural Networks for the Classification of Structures

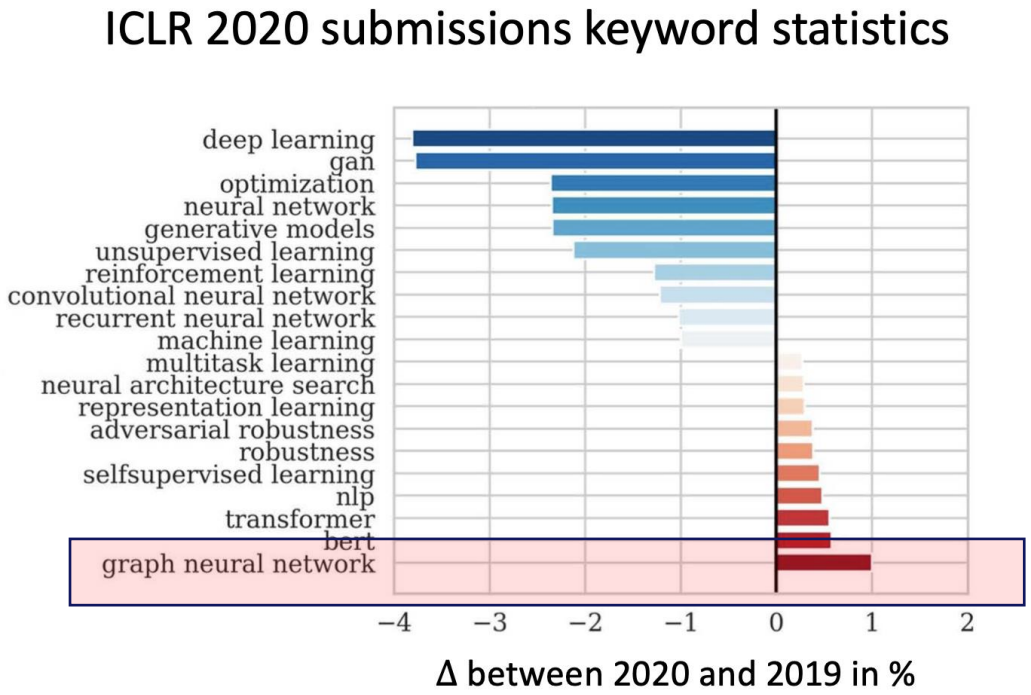
Alessandro Sperduti and Antonina Starita, *Member, IEEE*

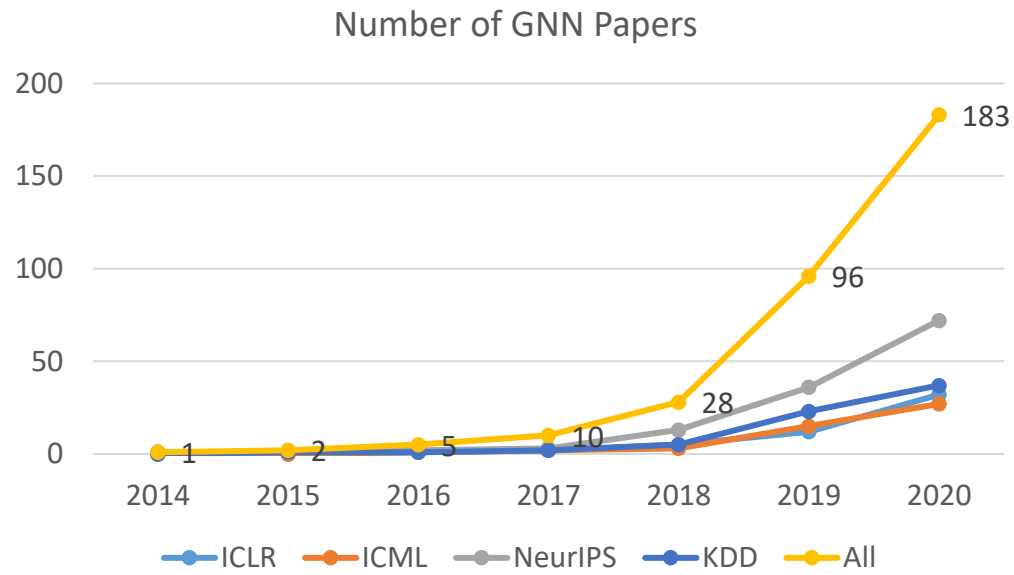
***Abstract*—Until now neural networks have been used for classifying unstructured patterns and sequences. However, standard neural networks and statistical methods are usually believed to be inadequate when dealing with complex structures because of their feature-based approach. In fact, feature-based approaches usually fail to give satisfactory solutions because of the sensitivity of the approach to the *a priori* selection of the features, and the incapacity to represent any specific information on the relationships among the components of the structures.**



Sperduti, Alessandro, and Antonina Starita. "Supervised neural networks for the classification of structures."

A Rapidly Growing Area

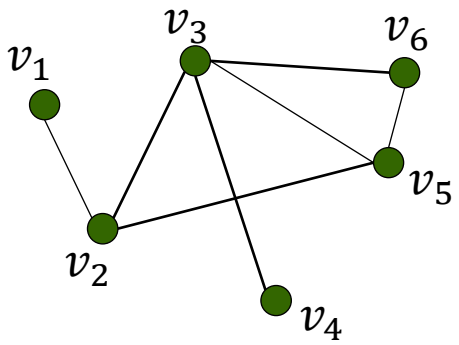




https://github.com/shaohua0116/ICLR2020-OpenReviewData

Preliminaries of Graph Learning

A topological graph



$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

which has n nodes and m edges

$$\mathcal{V} = \{v_1, \dots, v_n\}$$
$$\mathcal{E} = \{e_1, \dots, e_m\}$$

Node features:
$\boldsymbol{X} \in R^{n \times d}$

Adjacent matrix $\boldsymbol{A}$: $A_{ij} = 1$, existing edge between i and j
$A_{ij} = 0$, not exist edge between i and j

Degree matrix $\boldsymbol{D} = \mathrm{diag}(\mathrm{degree}(v_1), \dots, \mathrm{degree}(v_n))$

Laplacian matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$

Degree matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{pmatrix}$$

$-$

Adjacency matrix

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

$=$

Laplacian matrix

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & -1 & -1 & -1 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & -1 & -1 & 0 & 3 & -1 \\ 0 & 0 & -1 & 0 & -1 & 2 \end{pmatrix}$$

The Model of Graph Neural Networks

GNN 1.0

GNN 2.0

GNN 3.0

The Model of Graph Neural Networks

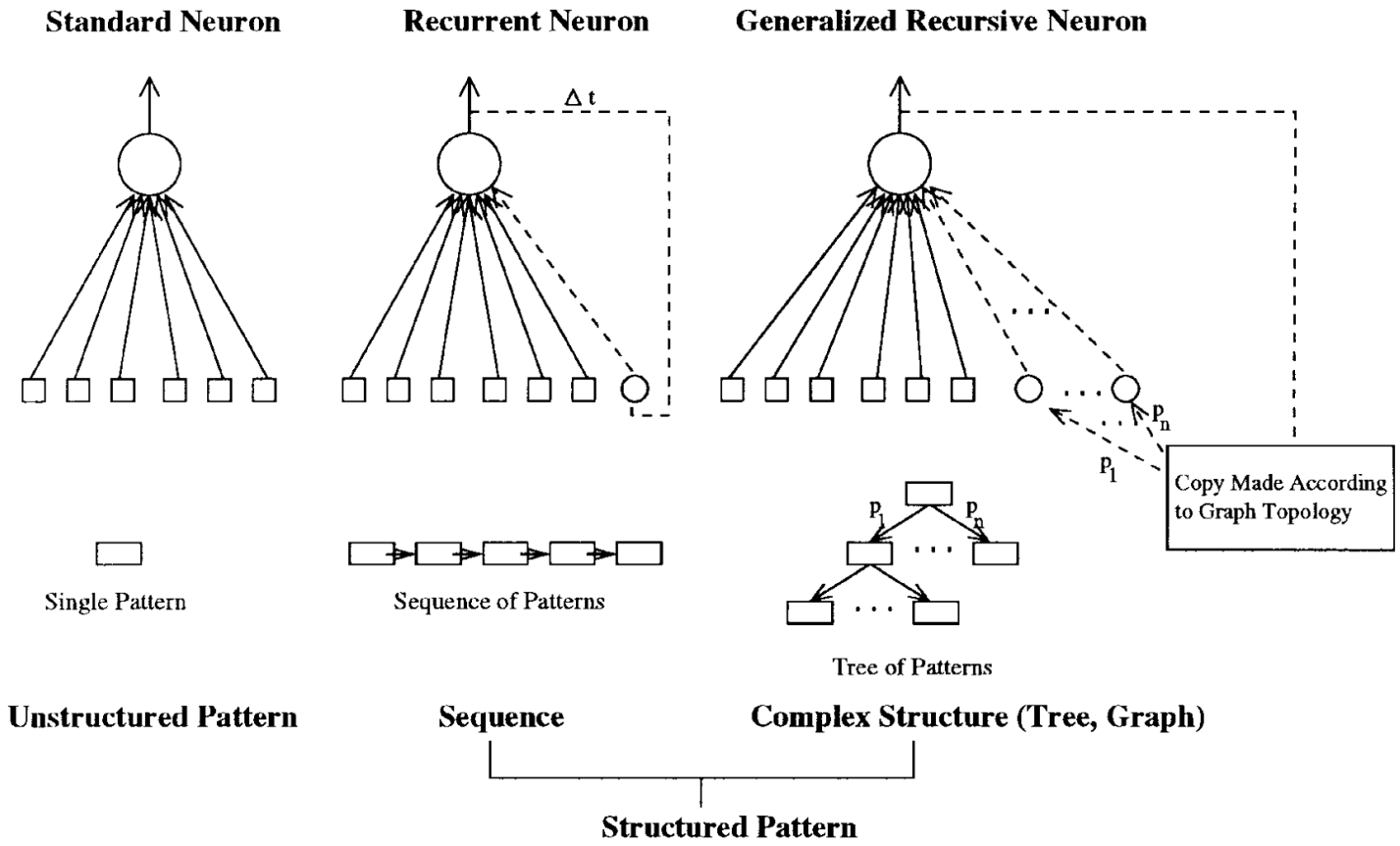
GNN 1.0

- Understanding GNN as RNN

GNN 2.0

GNN 3.0

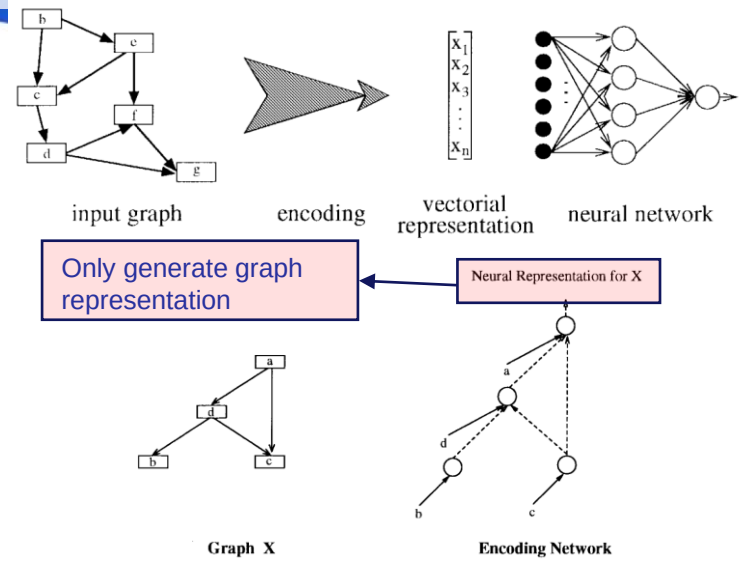
GNN 1.0: Understanding GNN as RNN



- The RNN on sequences can be generalized to trees and DAGs.

Sperduti, Alessandro, and Antonina Starita. 1997

GNN 1.0: Understanding GNN as RNN



From 2000 to 2010

Gori et.al (IJCNN 05) and Scarselli et.al (TNN 08) add **the output gate** for each node to generate the node representation in graphs. This model is called GraphRNN.

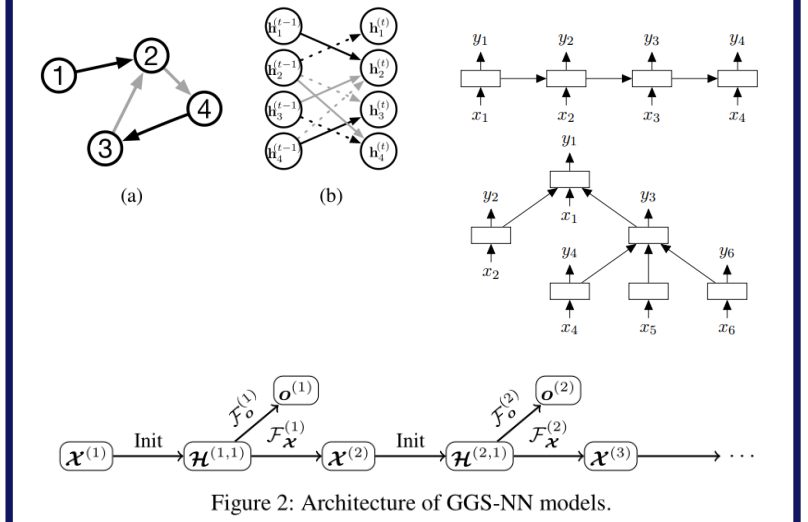
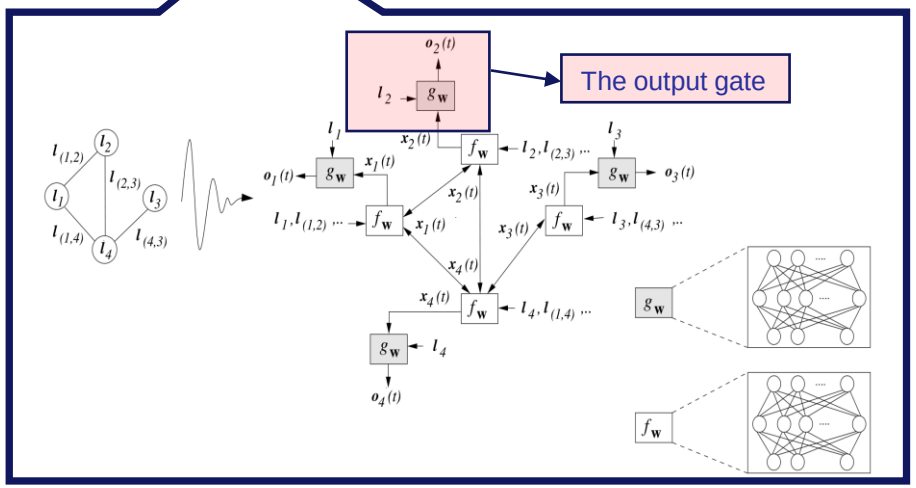


Figure 2: Architecture of GGS-NN models.

Before 2000

Sperduti, Alessandro, and Antonina Starita. (TNN 97) propose the **generalized recursive neuron** for the graph classification problem on Trees/DAGs.

This generalized recursive neuron **can only generate the graph representations**.



After 2010

Li, Yujia, et al. (ICLR 16) add gated recurrent units and modern optimization techniques to improve the performance of Scarselli et.al (TNN 09).
Tai, Kai Sheng et.al. (ACL 2015) extend LSTM to a tree-structured network topologies.

Sperduti, Alessandro, and Antonina Starita. 1997
Gori, Marco, Gabriele Monfardini, and Franco Scarselli. 2005
Scarselli, Franco, et al. 2008
Li, Yujia, et.al. 2015， Tai, Kai Sheng et.al, ,2015

Tencent AI Lab

The Brief History of Graph Neural Networks

GNN 1.0

- Understanding GNN as RNN

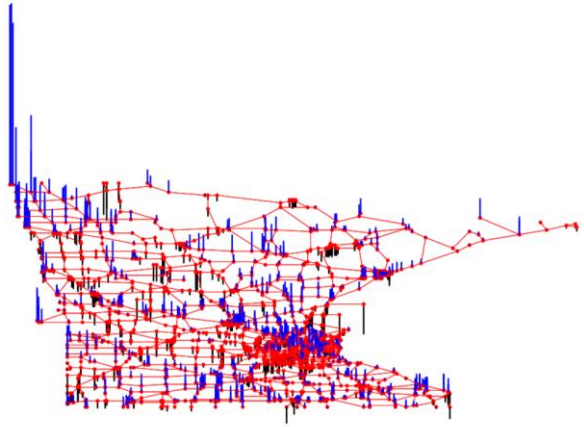
GNN 2.0

- Understanding GNN as Convolution

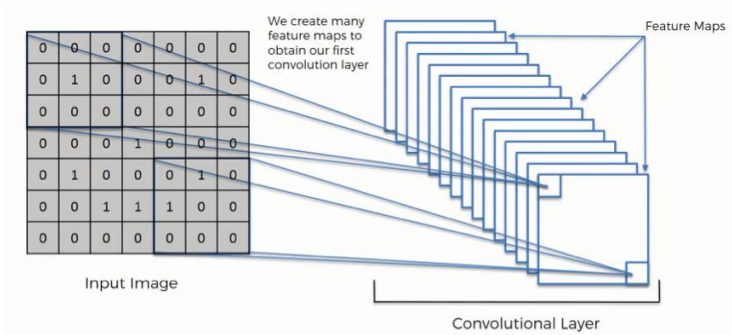
GNN 3.0

GNN 2.0: Understanding GNN as Convolution

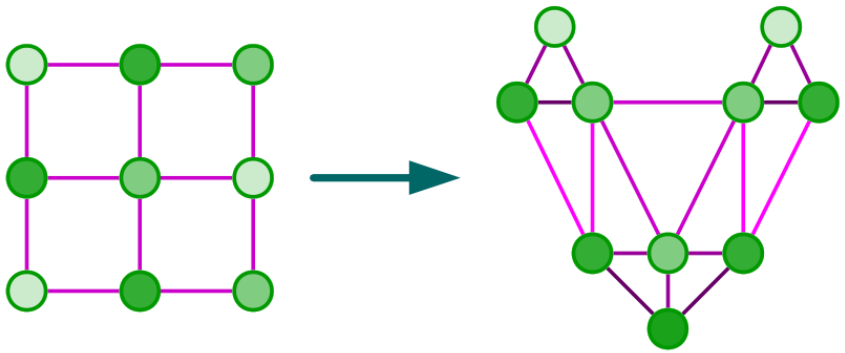
Graph Signal Processing



Convolutional Neural Networks

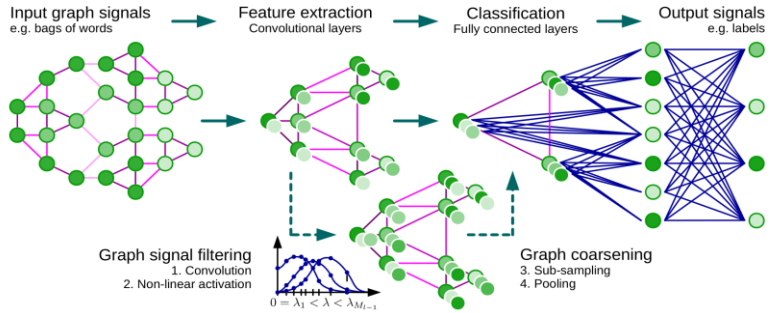


- How to perform the convolutions on graphs?
  - Irregular structures.
  - Weighted edges.
  - No orientation or ordering (in general).



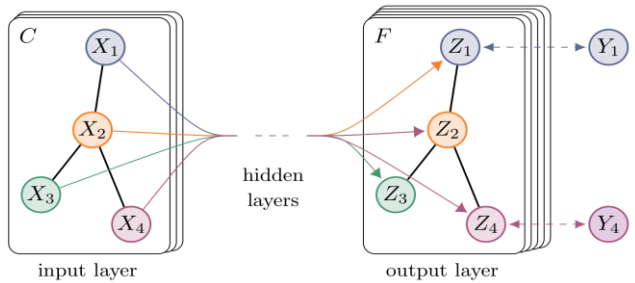
GNN 2.0: Understanding GNN as Convolution

ChebNet (NeurIPS 2016) [2]



- Build the connection between graph signal processing and graph convolution.
- Use Chebyshev polynomial to fast approximate the graph filtering in the spectral domain.

Graph Convolutional Network (ICLR 2017)

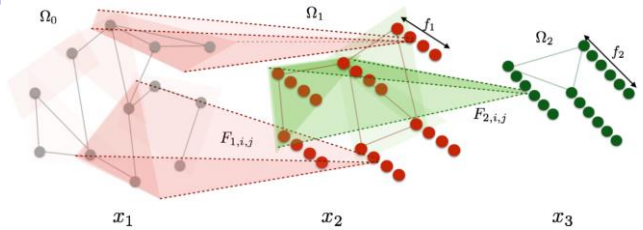


$$\boldsymbol{H}^{(l+1)} = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)})$$

- Approximate 1-order Chebyshev polynomial the in spatial domain.
- Layer-wise convolution to extend receptive field.
- **The practical convolutional model for graphs.**

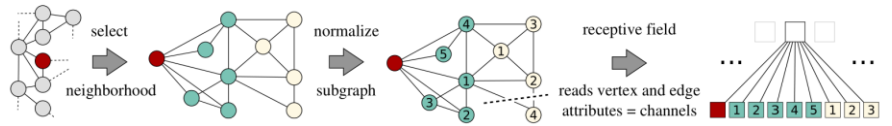
[1] Bruna, Joan, et al. 2014
[2] Defferrard, Michaël, et.al. 2016
[3] Niepert, Mathias, et.al. 2016
[4] Kipf, Thomas N., and Max Welling. 2017

Deep Locally Connected Networks(ICLR 2014) [1]



- Discuss two constructions on both spatial and spectral domain.
- Analog the convolution operation based on the Laplacian spectrum.
- **Additional eigen decomposition is needed.**

PATCHY-SAN (ICML 2016)

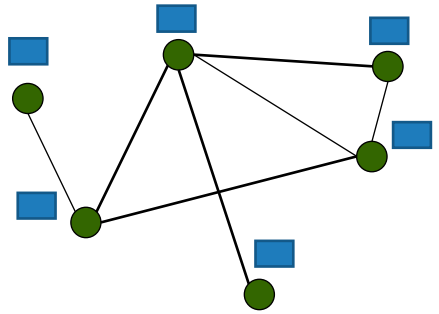


- Neighborhood sampling to construct receptive field.

Graph Signal Processing

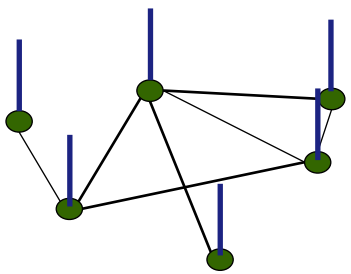
Graph signal:

h: $\mathcal{V} \rightarrow R^n$

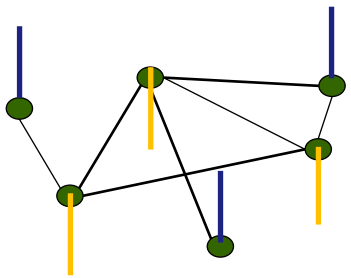


"**Frequency**" or "**Smoothness**" of the signal h

$$h^T \boldsymbol{L} h = \sum_{i<j} A_{ij}(h_i - h_j)^2$$



Low frequency graph signal



High frequency graph signal

Eigen decomposition of graph Laplacian

$$\boldsymbol{L} = \boldsymbol{U}\,\boldsymbol{\Lambda}\,\boldsymbol{U}^T$$

$$\mathbf{L} = \begin{bmatrix} | & & | \\ \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \\ | & & | \end{bmatrix} \begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix} \begin{bmatrix} — & \mathbf{u}_0 & — \\ & \vdots & \\ — & \mathbf{u}_{N-1} & — \end{bmatrix}$$

eigenvalues sorted non-decreasingly:

$$0 = \lambda_0 \leq \lambda_1 \leq \ldots \leq \lambda_{n-1}$$

The frequency of an eigenvector of L is its corresponding eigenvalue:

$$u_i^T \boldsymbol{L} u_i = u_i^T \lambda_i u_i = \lambda_i$$

Graph Convolution: Spectral domain → Spatial domain

Graph Convolution: input signal x, filter $\hat{g}$, graph Laplacian $\boldsymbol{L}$

$$y = x *_{\mathcal{G}} \hat{g} = \boldsymbol{U} \begin{pmatrix} \hat{g}(\lambda_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \hat{g}(\lambda_n) \end{pmatrix} \boldsymbol{U}^T x = \boldsymbol{U}\, \hat{g}(\boldsymbol{\Lambda}) \boldsymbol{U}^T x =: \hat{g}(\boldsymbol{L})\, x$$

Spectral

Parameterization: replace $\hat{g}(\boldsymbol{\Lambda})$ with $\hat{g}_\theta = diag(\theta)$

ChebNet (NeurIPS 2016): parameterize with Chebshev polynomials:

$$y = \hat{g}_\theta(\boldsymbol{L})x = \sum_{k=0}^{K-1} \theta_k\, T_k(\tilde{\boldsymbol{L}})x \qquad \tilde{\boldsymbol{L}} = \frac{1}{\lambda_{max}} \boldsymbol{L} - \boldsymbol{I}$$

GCN (ICLR 2017): simplified ChebNet $K = 1$, suppose $\lambda_{max} = 2$, $\theta := \theta_0 = -\theta_1$

$$\hat{g}_\theta(\boldsymbol{L})x = \left(\boldsymbol{I} + \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\right) x\theta$$

Spatial

Apply a renormalization trick: $\hat{g}_\theta(\boldsymbol{L})x = \tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}} x\theta$ $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ (add self-loop)

The Brief History of Graph Neural Networks

GNN 1.0

- Understanding GNN as RNN

GNN 2.0

- Understanding GNN as Convolution

GNN 3.0

- Variants of Convolutions
- GNN with Attention
- GNN with Graph Pooling

GNN 3.0: Variants of Convolutions

$$g_\theta * x = U g_\theta U^{\mathrm{T}} x \quad \Rightarrow \quad \boldsymbol{H}^{(l+1)} = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)})$$

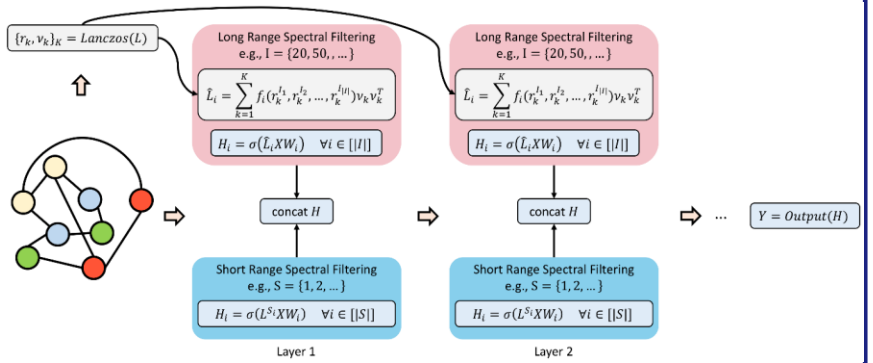
Lanczos Network [3]	Graph Wavelet Neural Network [1]	Hyperbolic GCN [2]

[1] Xu, Bingbing, et al. 2018 [2] Chami, Ines, et al. 2019 [3] Liao, Renjie, et al. 2019

GNN 3.0: Variants of Convolutions

$$g_\theta * x = U g_\theta U^{\mathrm{T}} x \quad \Rightarrow \quad \boldsymbol{H}^{(l+1)} = \sigma(\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)})$$

Lanczos Network [3]



- Employ Lanczos algorithm to obtain the low-rank approximation of the graph Laplacian $\mathrm{I} - \widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}$.
- Easy to construct multi-scale Graph Convolution.

Graph Wavelet Neural Network [1]

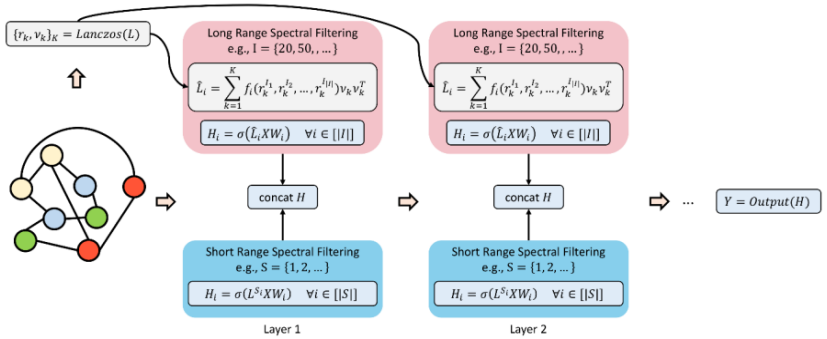
Hyperbolic GCN [2]

[1] Xu, Bingbing, et al. 2018 [2] Chami, Ines, et al. 2019 [3] Liao, Renjie, et al. 2019

GNN 3.0: Variants of Convolutions

$$g_\theta * x = U g_\theta U^{\mathrm{T}} x \quad \Rightarrow \quad \boldsymbol{H}^{(l+1)} = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)})$$

Lanczos Network [3]



- Employ Lanczos algorithm to obtain the low-rank approximation of the graph Laplacian $\mathrm{I} - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$.
- Easy to construct multi-scale Graph Convolution.

Graph Wavelet Neural Network [1]

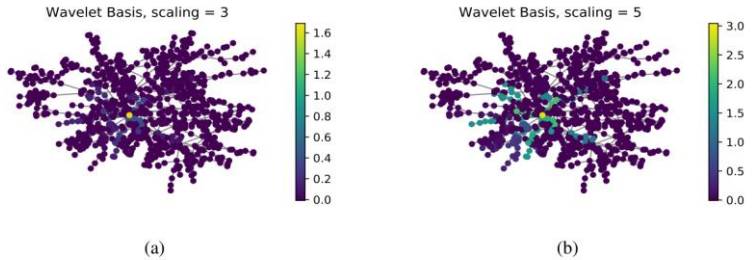


Figure 1: Wavelets on an example graph at (a) small scale and (b) large scale.

$$\boldsymbol{H}_{[:,j]}^{(l+1)} = \sigma\left(\psi_s \sum_{i=1}^{p} \boldsymbol{F}_{i.j}^{(l)} \psi_s^{-1} H_{[:,i]}^{(l)}\right), \quad j = 1, \ldots, q$$

- Use wavelet transform to replace Fourier transform in the original GCN.
- More localized convolution and flexible neighborhood.

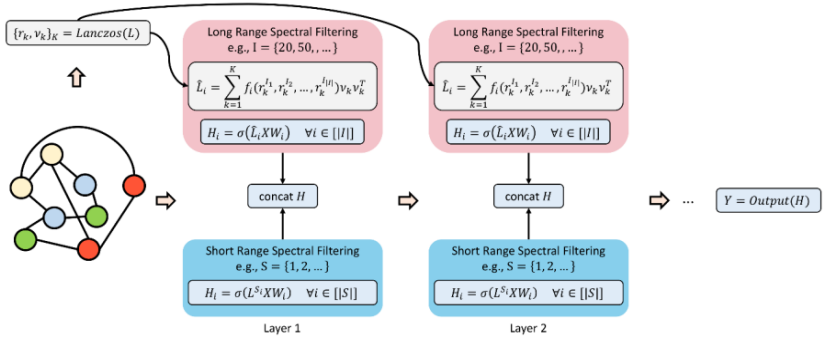
Hyperbolic GCN [2]

[1] Xu, Bingbing, et al. 2018 [2] Chami, Ines, et al. 2019 [3] Liao, Renjie, et al. 2019

GNN 3.0: Variants of Convolutions

$$g_\theta * x = U g_\theta U^{\mathrm{T}} x \quad \Rightarrow \quad \boldsymbol{H}^{(l+1)} = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{H}^{(l)}\boldsymbol{W}^{(l)})$$

Lanczos Network [3]



- Employ Lanczos algorithm to obtain the low-rank approximation of the graph Laplacian $\mathrm{I} - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$.
- Easy to construct multi-scale Graph Convolution.

Graph Wavelet Neural Network [1]

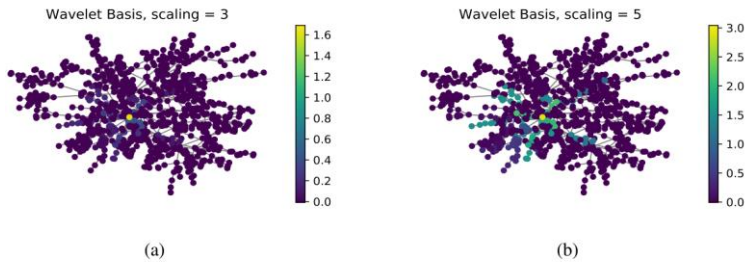
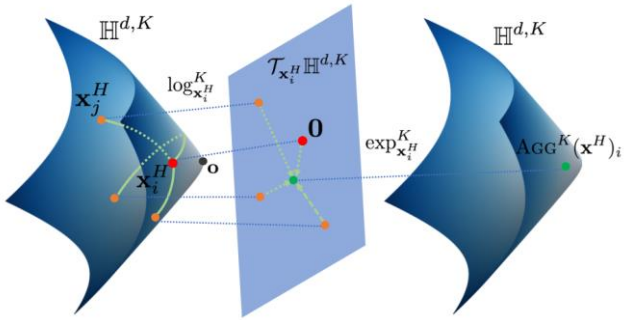


Figure 1: Wavelets on an example graph at (a) small scale and (b) large scale.

$$\boldsymbol{H}_{[:,j]}^{(l+1)} = \sigma\left(\psi_s \sum_{i=1}^{p} \boldsymbol{F}_{i.j}^{(l)} \psi_s^{-1} H_{[:,i]}^{(l)}\right), \quad j = 1, \ldots, q$$

- Use wavelet transform to replace Fourier transform in the original GCN.
- More localized convolution and flexible neighborhood.

Hyperbolic GCN [2]



Construct the GCN in hyperbolic space.

- Smaller distortion.
- Suitable for scale-free and hierarchical structure.

- Hyperbolic feature transform.
  $\boldsymbol{h}_i^{(l+1),H} = (\boldsymbol{W}^{(l+1)} \otimes^{K_l} \boldsymbol{h}_i^{(l),H}) \oplus^{K_l} \boldsymbol{b}^{(l+1)}$
- Attention-based hyperbolic aggregation.
  $\boldsymbol{y}_i^{(l+1),H} = \mathrm{AGG}^{K_l}(\boldsymbol{h}^{(l),H})_i$

[1] Xu, Bingbing, et al. 2018 [2] Chami, Ines, et al. 2019 [3] Liao, Renjie, et al. 2019

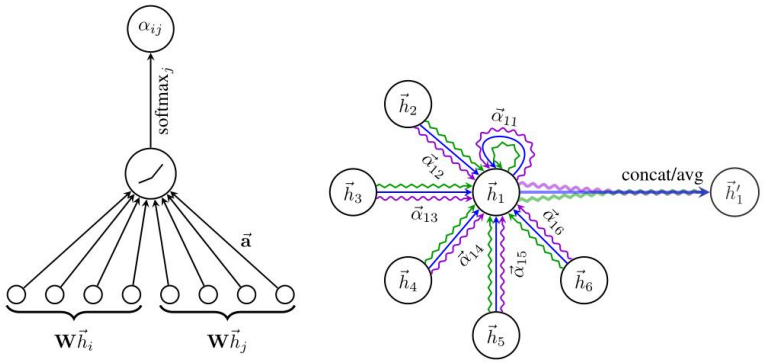
GNN 3.0: GNN with Attention

$$\boldsymbol{h}_i^{(l+1)} = \sigma\left(\sum_{j\in N(v_i)} S_{i,j}\, \boldsymbol{W}^{(l)} \boldsymbol{h}_j^{(l)}\right)$$

$$\boldsymbol{S} = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$$

Fixed during training

Graph Attention Network [1]



Replace the fixed aggregation weight a_{ij} to the learnable self-attention.

$$\boldsymbol{h}_i^{(l+1)} = \sigma\Big(\sum_{j\in N(v_i)} a_{ij}\, W^{(l)} \boldsymbol{h}_j^{(l)}\Big)$$

$$a_{ij} = \exp\left(\frac{\sigma\left(\boldsymbol{\alpha}^{\mathrm{T}}[\boldsymbol{W}\boldsymbol{h}_i || \boldsymbol{W}\boldsymbol{h}_j]\right)}{\sum_{k\in N(v_i)} \boldsymbol{\alpha}^{\mathrm{T}}[\boldsymbol{W}\boldsymbol{h}_i || \boldsymbol{W}\boldsymbol{h}_k]}\right)$$

Gated Attention Networks [2]

Spectral Graph Attention Network [3]

[1] Veličković, Petar, et al. 2018 [2] Zhang, Jiani, et al. 2018 [3] Chang, Heng, et al. 2020

GNN 3.0: GNN with Attention

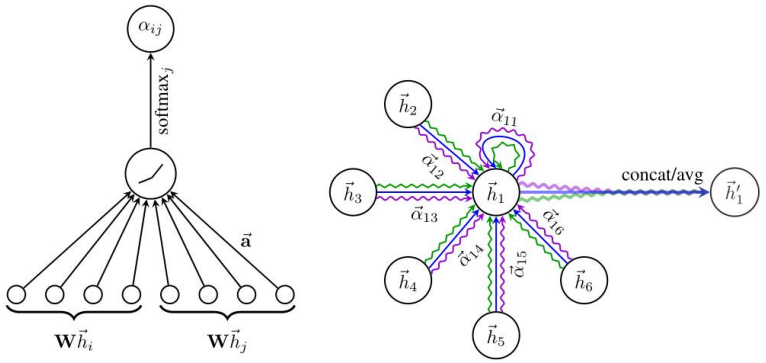
The original form:

$$\boldsymbol{h}_i^{(l+1)} = \sigma\left(\sum_{j\in N(v_i)} S_{i,j}\, \boldsymbol{W}^{(l)} \boldsymbol{h}_j^{(l)}\right)$$

$$\boldsymbol{S} = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}} \widetilde{\boldsymbol{A}} \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$$

Fixed during training

Graph Attention Network [1]

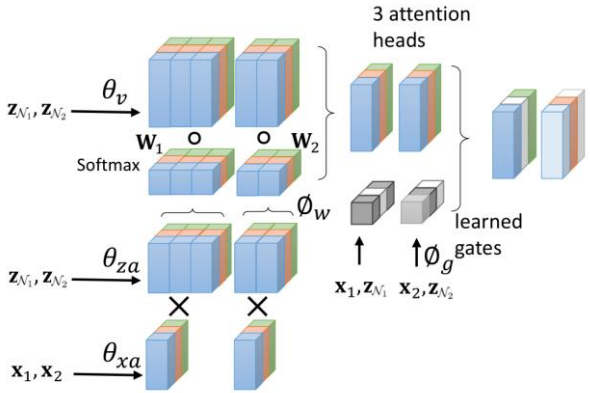


Replace the fixed aggregation weight a_{ij} to the learnable self-attention.

$$\boldsymbol{h}_i^{(l+1)} = \sigma\Big(\sum_{j\in N(v_i)} a_{ij}\, W^{(l)} \boldsymbol{h}_j^{(l)}\Big)$$

$$a_{ij} = \exp\Big(\frac{\sigma\left(\boldsymbol{\alpha}^{\mathrm{T}}[\boldsymbol{W}\boldsymbol{h}_i || \boldsymbol{W}\boldsymbol{h}_j]\right)}{\sum_{k\in N(v_i)} \boldsymbol{\alpha}^{\mathrm{T}}[\boldsymbol{W}\boldsymbol{h}_i || \boldsymbol{W}\boldsymbol{h}_k]}\Big)$$

Gated Attention Networks [2]



Add a learnable gate g_i^k to model the importance for each head.

$$\boldsymbol{h}_i^{(l+1)} = \sigma\Big(\sum_{k=1}^{K} g_i^k \sum_{j\in N(v_i)} a_{ij}\, W^{(l)} \boldsymbol{h}_j^{(l)}\Big)$$

K is the number of heads.

Spectral Graph Attention Network [3]

[1] Veličković, Petar, et al. 2018 [2] Zhang, Jiani, et al. 2018 [3] Chang, Heng, et al. 2020

GNN 3.0: GNN with Attention

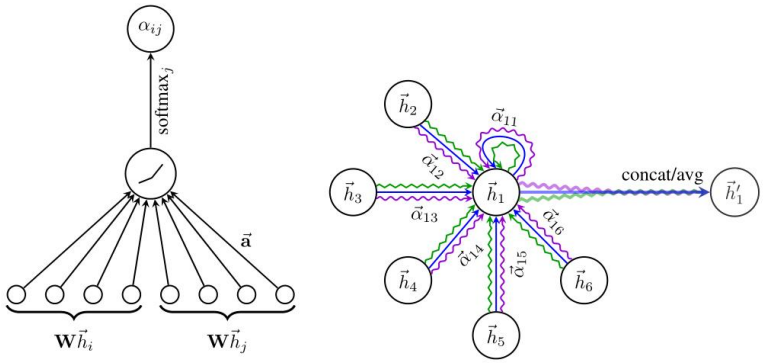
The original form:

$$\boldsymbol{h}_i^{(l+1)} = \sigma\left(\sum_{j\in N(v_i)} S_{i,j}\, \boldsymbol{W}^{(l)} \boldsymbol{h}_j^{(l)}\right)$$

$$\boldsymbol{S} = \widetilde{\boldsymbol{D}}^{-\frac{1}{2}}\widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{D}}^{-\frac{1}{2}}$$

Fixed during training

Graph Attention Network [1]

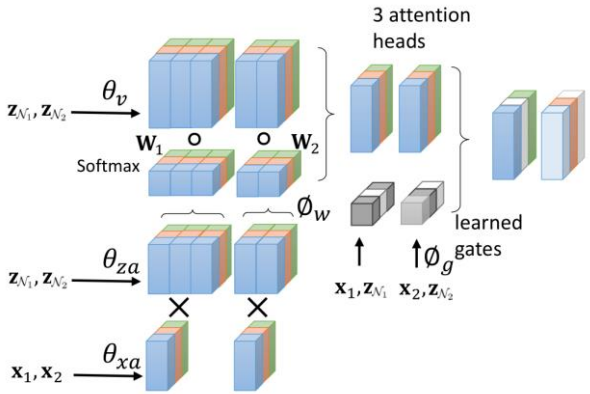


Replace the fixed aggregation weight a_{ij} to the learnable self-attention.

$$\boldsymbol{h}_i^{(l+1)} = \sigma\Big(\sum_{j\in N(v_i)} a_{ij}\, W^{(l)} \boldsymbol{h}_j^{(l)}\Big)$$

$$a_{ij} = \exp\Big(\frac{\sigma\left(\boldsymbol{\alpha}^{\mathrm{T}}[\boldsymbol{W}\boldsymbol{h}_i \| \boldsymbol{W}\boldsymbol{h}_j]\right)}{\sum_{k\in N(v_i)} \boldsymbol{\alpha}^{\mathrm{T}}[\boldsymbol{W}\boldsymbol{h}_i \| \boldsymbol{W}\boldsymbol{h}_k]}\Big)$$

Gated Attention Networks [2]

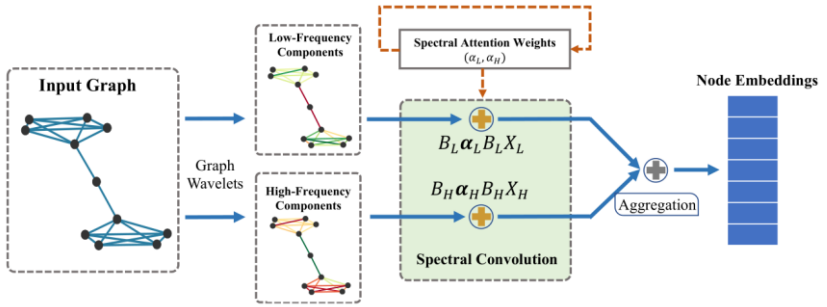


Add a learnable gate g_i^k to model the importance for each head.

$$\boldsymbol{h}_i^{(l+1)} = \sigma\Big(\sum_{k=1}^{K} g_i^k \sum_{j\in N(v_i)} a_{ij}\, W^{(l)} \boldsymbol{h}_j^{(l)}\Big)$$

K is the number of heads.

Spectral Graph Attention Network [3]



Apply the attention on the high / low-frequency components in spectral domain.

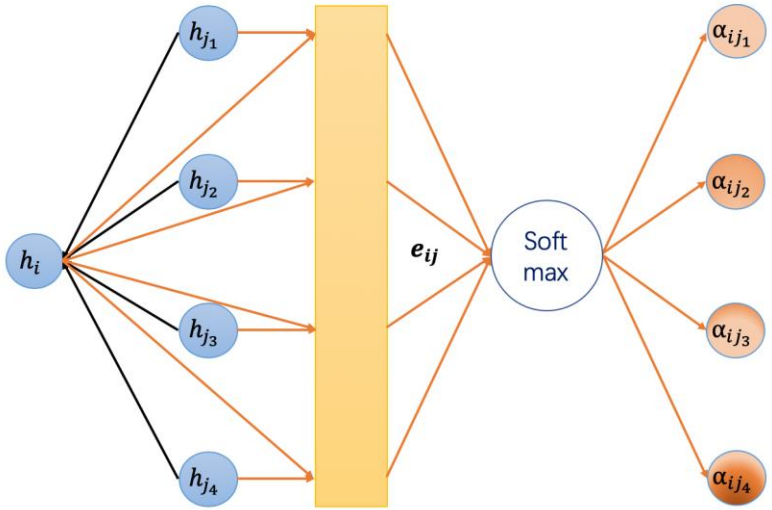
$$\boldsymbol{H}^{(l+1)} = \sigma\big(\mathrm{AGG}\big(\boldsymbol{B_L} a_L \boldsymbol{B_L} \boldsymbol{H}^{(l)}, \boldsymbol{B_H} a_H \boldsymbol{B_H} \boldsymbol{H}^{(l)}\big)\boldsymbol{W}^{(l)}\big)$$

$\boldsymbol{B} = [\boldsymbol{B_L}, \boldsymbol{B_H}]$ is the spectral graph wavelet bases.

[1] Veličković, Petar, et al. 2018 [2] Zhang, Jiani, et al. 2018 [3] Chang, Heng, et al. 2020

Graph Attention Network in Detail

Single head attention



$$\boldsymbol{h}_i^{(l+1)} = \sigma(\sum_{j \in N(v_i)} \alpha_{ij} \boldsymbol{W} \quad \boldsymbol{h}_j^{(l)})$$

$$\alpha_{ij} = \mathrm{Softmax}(e_{ij})$$

$$e_{ij} = \mathrm{LeakyReLU}(\boldsymbol{a}^T \, (\boldsymbol{W}\boldsymbol{h}_i \, || \boldsymbol{W}\boldsymbol{h}_j)$$

Multi-head attention

Enrich the model capacity and stabilize the learning process

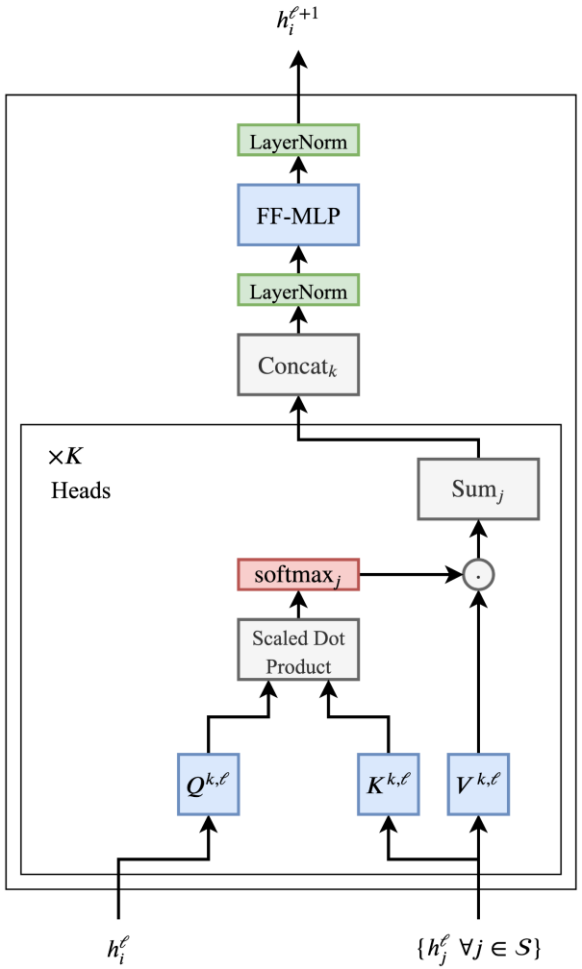
Each head has its own parameters
and their outputs can be merged in two ways:

- ❑ Concatenation
- ❑ Average

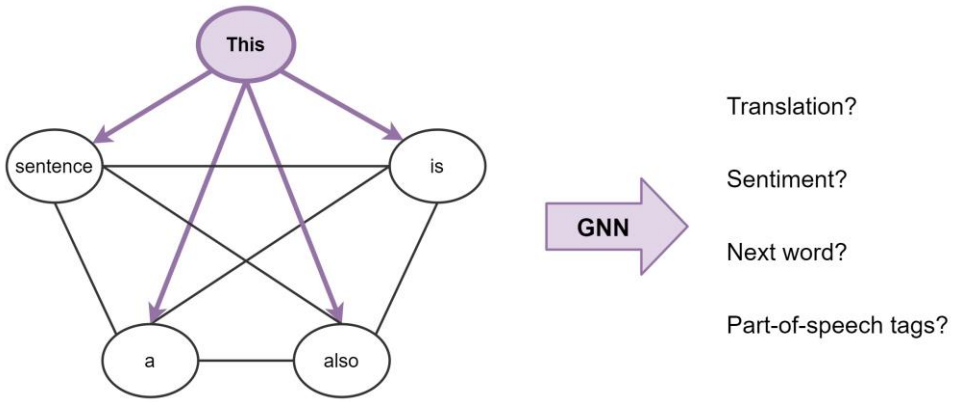
Attention weights learnt for the Cora dataset

Transformers as GNNs with Multi-head Attention

One layer of the multi-head QKV attention



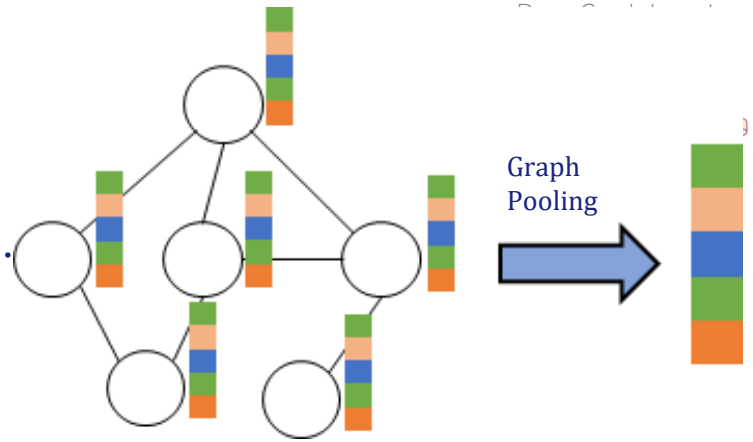
Transformer takes input sequence as a complete graph.



- Transformers can be viewed as GNNs with multi-head attention as the neighborhood aggregation function
- Transformers for NLP tasks treat the entire sequence as the neighborhood

Chaitanya Joshi. Transformers are graph neural networks, 2020. https:// graphdeeplearning.github.io/post/transformers-are-gnns/

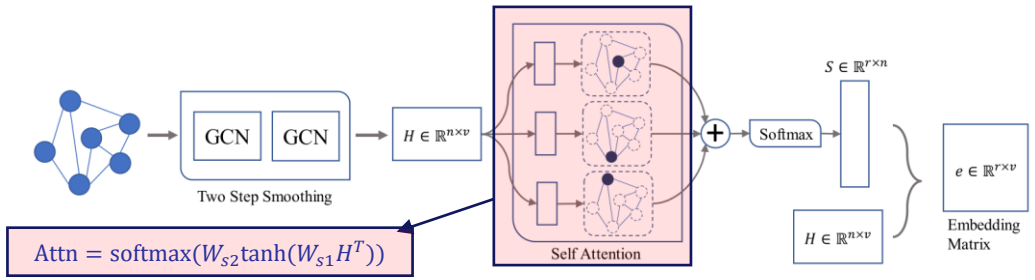
GNN 3.0: GNN with Graph Pooling



Graph Pooling/Coarsening: Convert the node representation to graph representation.

- The most straightforward way: Max/Mean Pooling
- SAGE: Attentive Pooling

Self-Attentive Graph Embedding (SAGE)

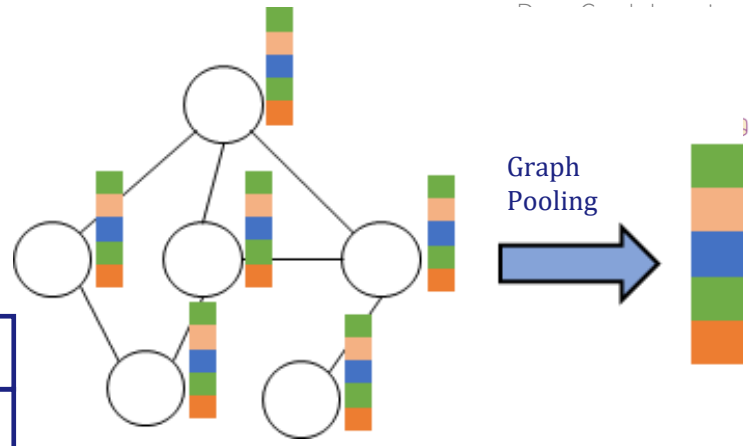


Introduce the self-attention mechanism to model the node importance during the pooling.

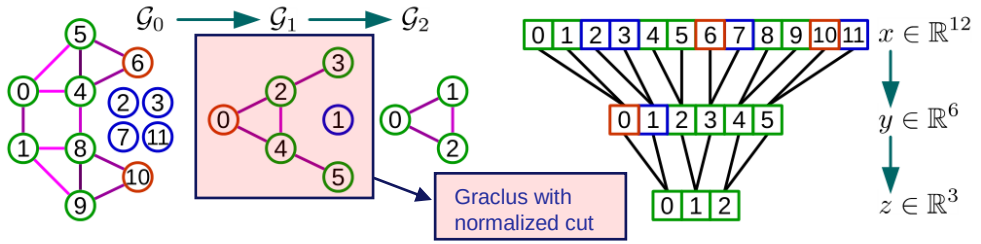
Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective."

GNN 3.0: GNN with Graph Pooling

Hierarchical Pooling

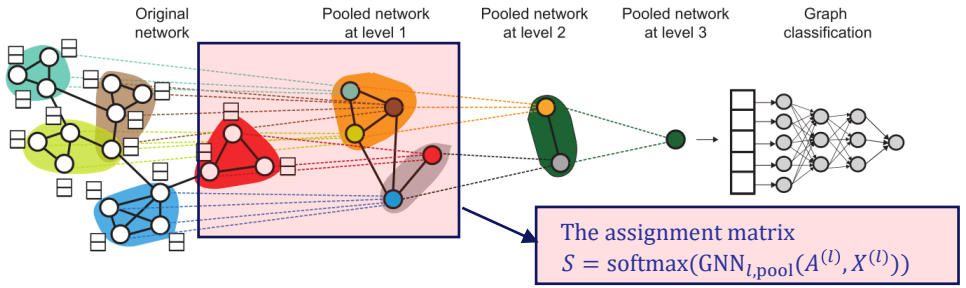


Graph Coarsening by Graph Cut [1]



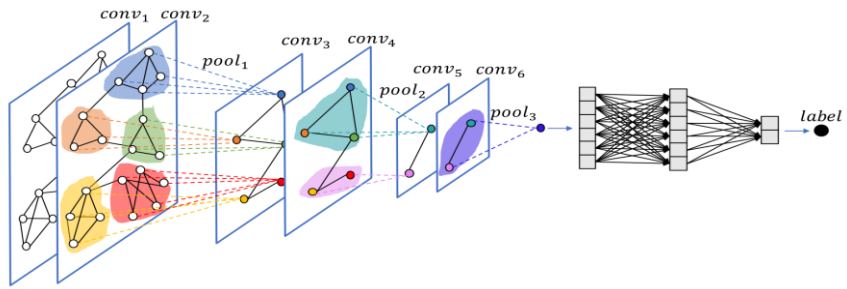
Graph Pooling with pre-defined subgraph by graph cut algorithm.

Differentiable Graph Pooling (DIFFPOOL)[2]



Learn the cluster assignment matrix to aggregate the node representations in a hierarchical way.

EigenPooling [3]



Incorporate the node features and local structures to obtain a better assignment matrix.

[1] Defferrard, Michaël, et.al. 2016 [2] Ying, Zhitao, et al. 2018 [3] Ma, Yao, et al. 2019

GNN Implementation: Message Passing Framework

- Message Passing Framework:
  - **Step 1:** Gather and transform the messages from neighbors:

$$\boldsymbol{m}_i^{(l+1)} = \mathrm{AGG}\left(\left\{M^{(l+1)}\left(\boldsymbol{h}_i^{(l)}, \boldsymbol{h}_j^{(l)}, \boldsymbol{e}_{i,j}\right) \mid j \in N(v_i)\right\}\right)$$

The message generation function.
Input: the state of current node, the state of the neighbor node and the edge features.

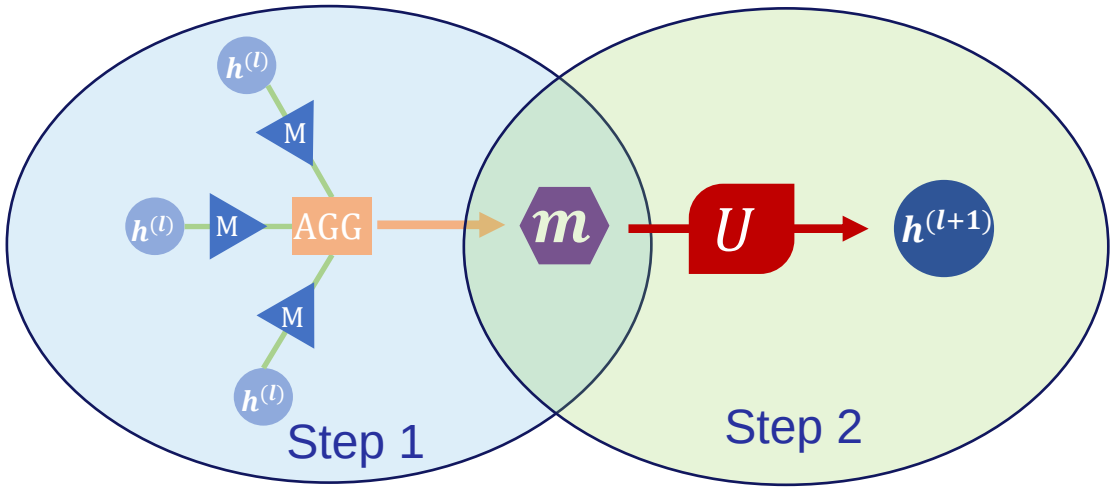
The neighborhood set of node.
E.g. 1-hop neighbors.

The aggregation function.
E.g. SUM/MEAN/LSTM

  - **Step 2:** Update the state of the target node.

$$\boldsymbol{h}_i^{(l+1)} = U^{(l+1)}\left(\boldsymbol{h}_i^{(l)}, \boldsymbol{m}_i^{(l+1)}\right)$$

The state update function.



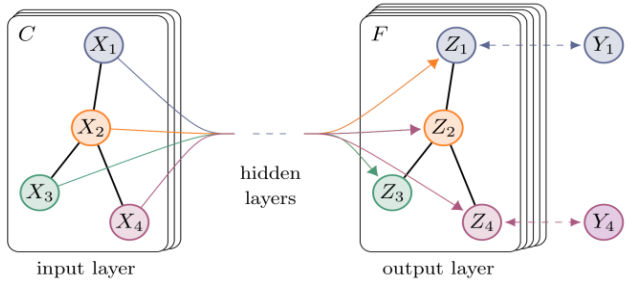
Gilmer, Justin, et al. "Neural Message Passing for Quantum Chemistry." ICML. 2017.

36

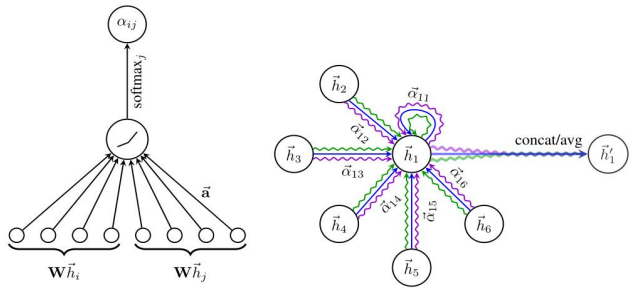
Realizations of Message Passing Neural Nets

Most of current GNNs can be formulated as a message passing process.

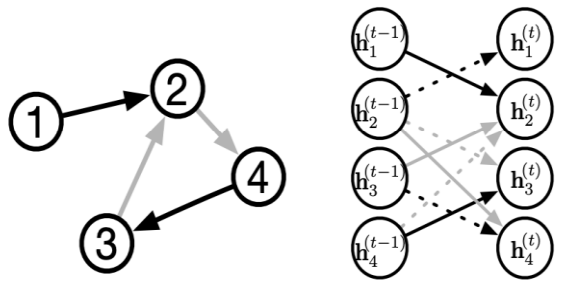
GCN (ICLR 2017)



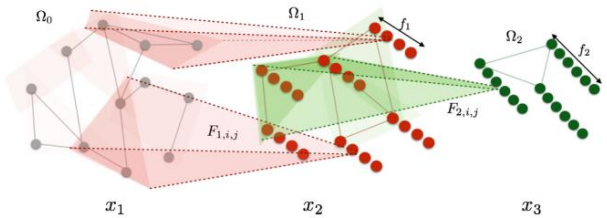
Graph Attention Network (ICLR 2018)



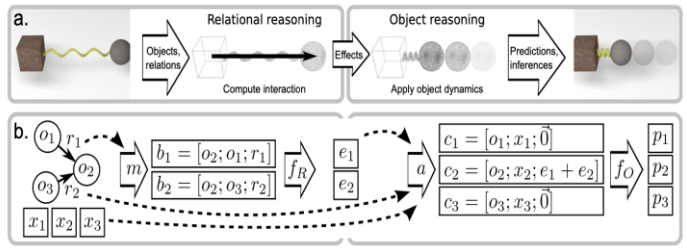
Gated GNNs (ICLR 2016)



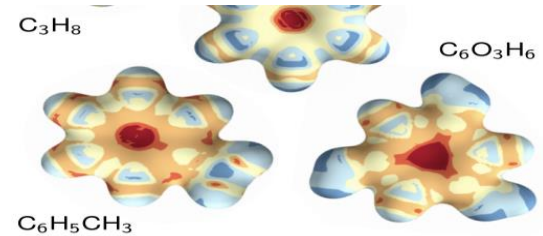
Deep Locally Connected Networks (ICLR 2014)



Interaction Networks (NeurIPS 2016)



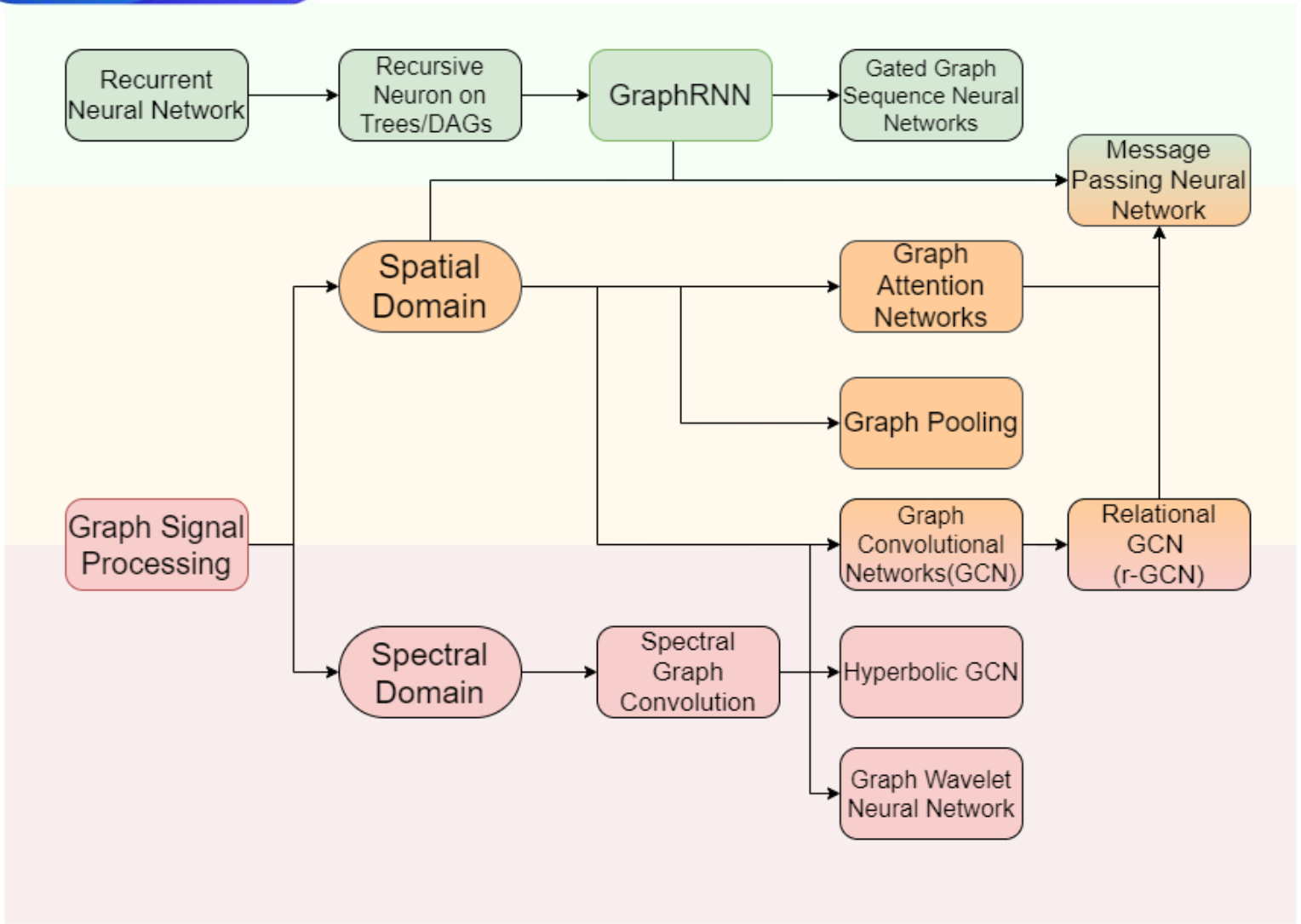
Deep Tensor Neural Networks (Nature Communications 2017)



Wang, Minjie, et al. "Deep graph library: A graph-centric, highly-performant package for graph neural networks." *arXiv preprint arXiv:1909.01315* (2019).

Fey, Matthias and Lenssen, Jan Eric Fast Graph Representation Learning with PyTorch Geometric. (2019). , cite arxiv:1903.02428.

Summary



Advanced topics

- Training Deep GNNs
- Scalability of GNNs
- Robustness of GNNs
- Self/Un-Supervised Learning of GNNs
- ….

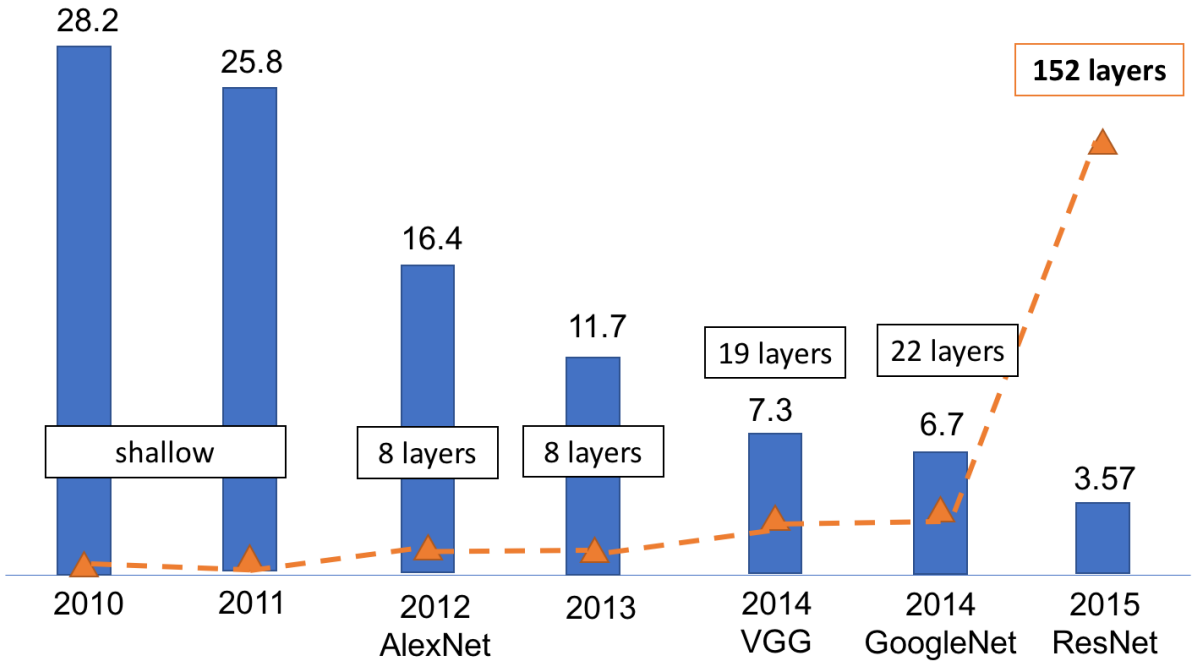
Training Deep GNNs

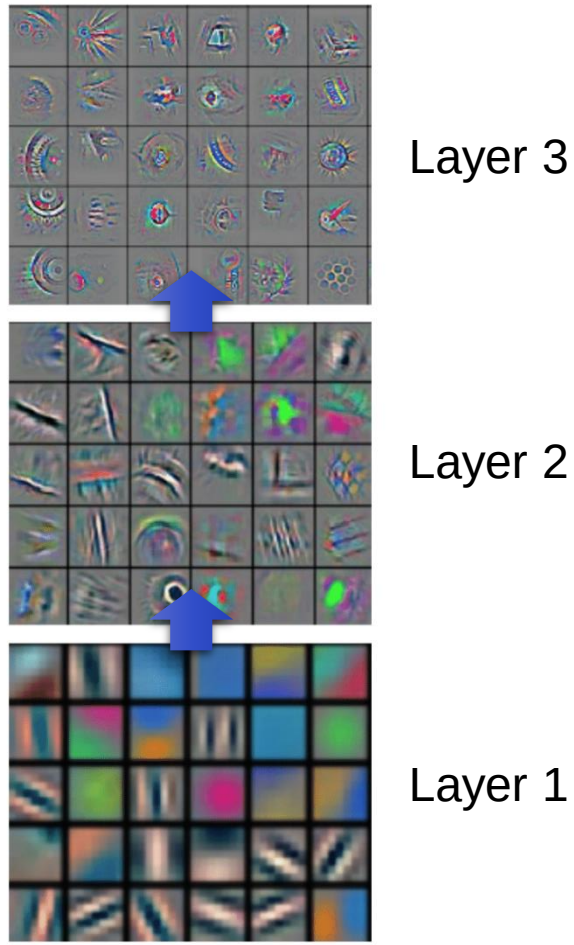
Training Deep GNNs

- Why do we need deep GNNs?
- Can GNNs simply go deep?
- What impedes GNNs to go deep?
- How to make GNNs deep?

The Power of Deep DNNs

- Unprecedented success of deep DNNs in computer vision
- Deep DNNs enable larger receptive fields
- Deep DNNs enable more expressivity

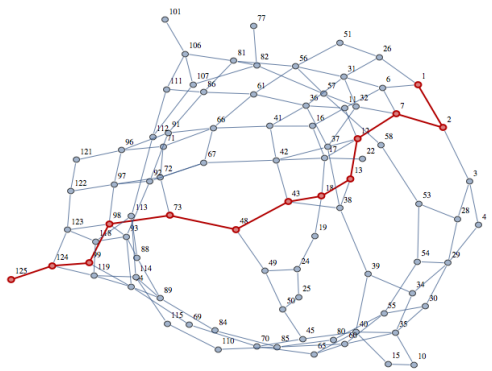




The Power of Deep GNNs

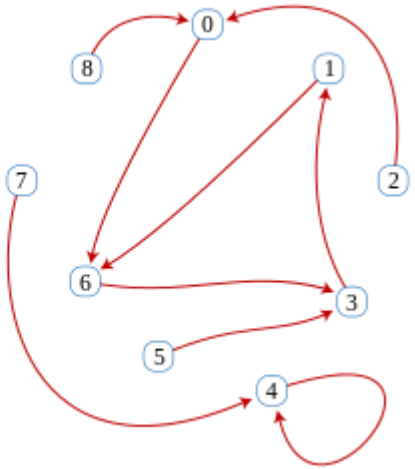
- Do GNNs need deep structures to enable larger receptive fields, too?
- What limits the expressive power of GNNs?
  - The depth d
  - The width w
- GNNs significantly lose their power when *capacity*, dw, is restricted

Shortest Path

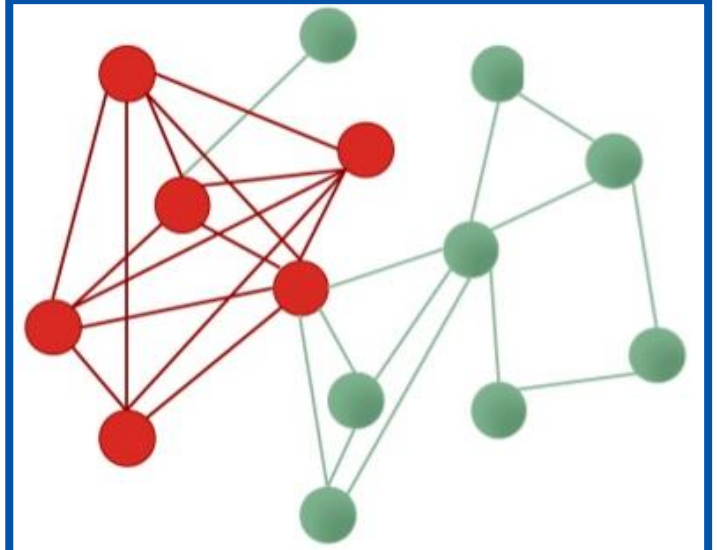


Cycle Detection

x	$f(x)$
0	6
1	6
2	0
3	1
4	4
5	3
6	3
7	4
8	0



Subgraph



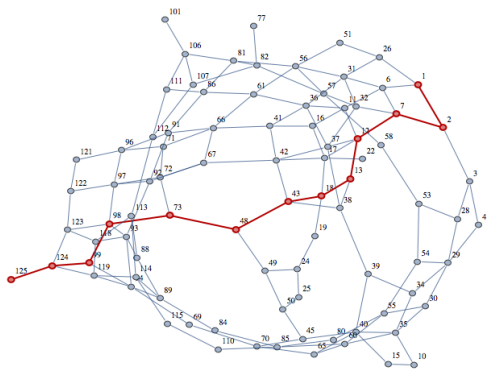
Loukas, Andreas. "What graph neural networks cannot learn: depth vs width." *International Conference on Learning Representations. 2020.*

The Power of Deep GNNs

- Do GNNs need deep structures to enable larger receptive fields, too?
- What limits the expressive power of GNNs?
  - The depth d
  - The width w
- GNNs significantly lose their power when *capacity*, dw, is restricted

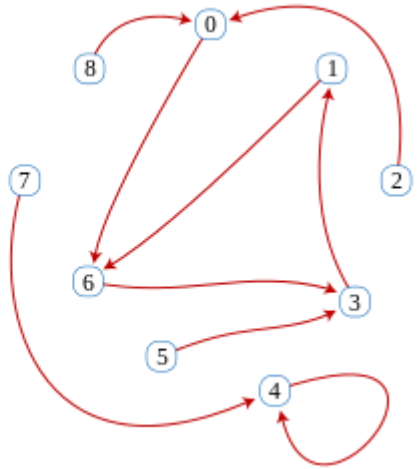
Yes

Shortest Path

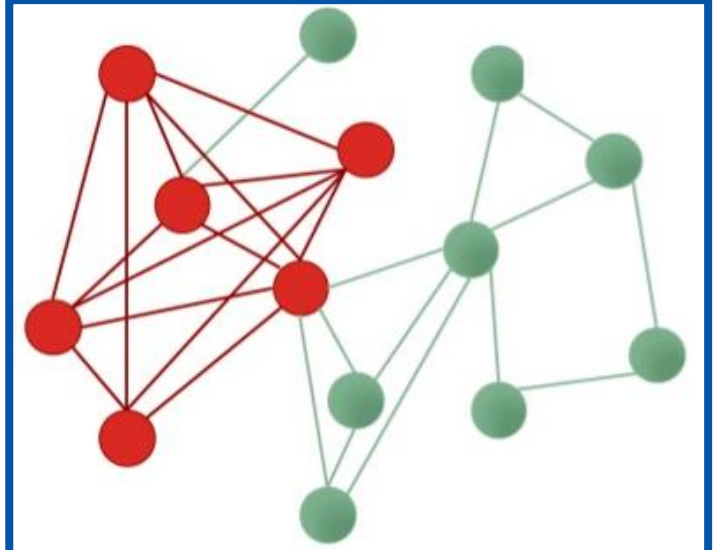


Cycle Detection

x	$f(x)$
0	6
1	6
2	0
3	1
4	4
5	3
6	3
7	4
8	0



Subgraph



Loukas, Andreas. "What graph neural networks cannot learn: depth vs width." *International Conference on Learning Representations. 2020.*

The Power of Deep GNNs

- The boundary of capacity for different problems

problem	*bound*	*problem*	*bound*
cycle detection (odd)	$dw = \Omega(n/\log n)$	shortest path	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$
cycle detection (even)	$dw = \Omega(\sqrt{n}/\log n)$	max. indep. set	$dw = \Omega(n^2/\log^2 n)$ for $w = O(1)$
subgraph verification*	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$	min. vertex cover	$dw = \Omega(n^2/\log^2 n)$ for $w = O(1)$
min. spanning tree	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$	perfect coloring	$dw = \Omega(n^2/\log^2 n)$ for $w = O(1)$
min. cut	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$	girth 2-approx.	$dw = \Omega(\sqrt{n}/\log n)$
diam. computation	$dw = \Omega(n/\log n)$	diam. $3/2$-approx.	$dw = \Omega(\sqrt{n}/\log n)$

(Loukas, ICLR'20)

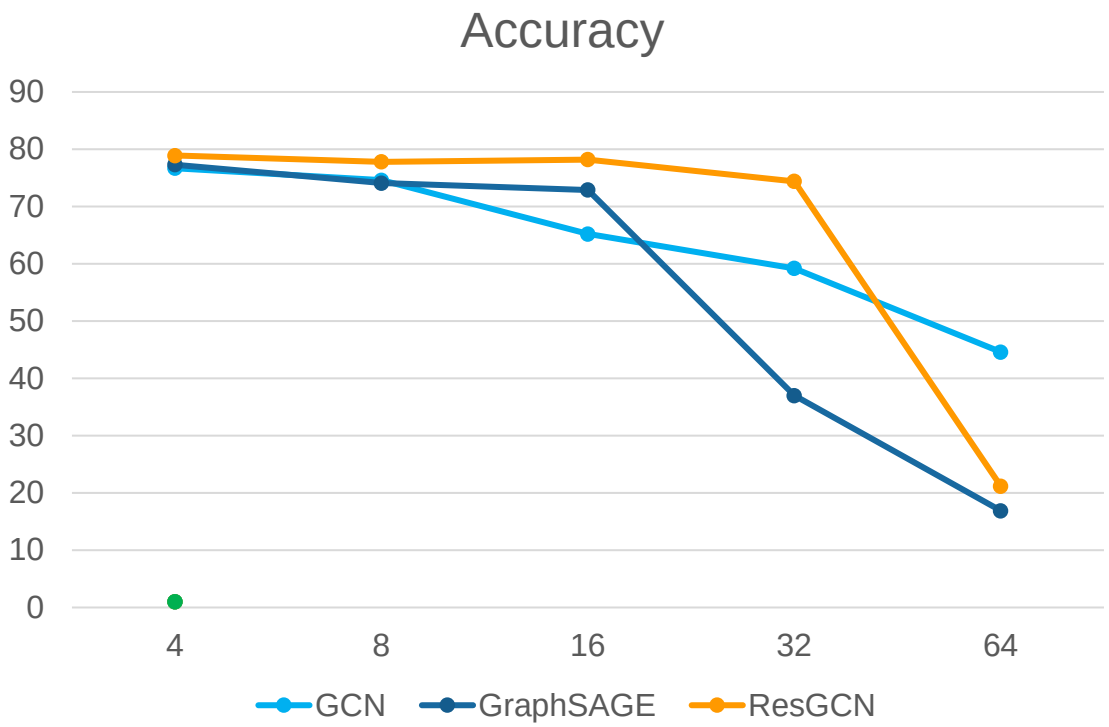
Loukas, Andreas. "What graph neural networks cannot learn: depth vs width." *International Conference on Learning Representations. 2020.*

Training Deep GNNs

- Why do we need deep GNNs?
- Can GNNs simply go deep?
  - **GCN**: Basic GCN
  - **GraphSAGE**: GCN with improved **aggregation**
  - **ResGCN**: leverage idea from **ResNet**
- What impedes GNNs to go deep?
- How to make GNNs deep?

GNNs are Shallow

- But can they really go deep? Not all



Citeseer	4 layers	16 layers	64 layers
GCN	76.7	65.2	44.6
GraphSAGE	77.3	72.9	16.9
ResGCN	78.9	78.2	21.2

(Rong et al, ICLR'20)

Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

Training deep GNNs

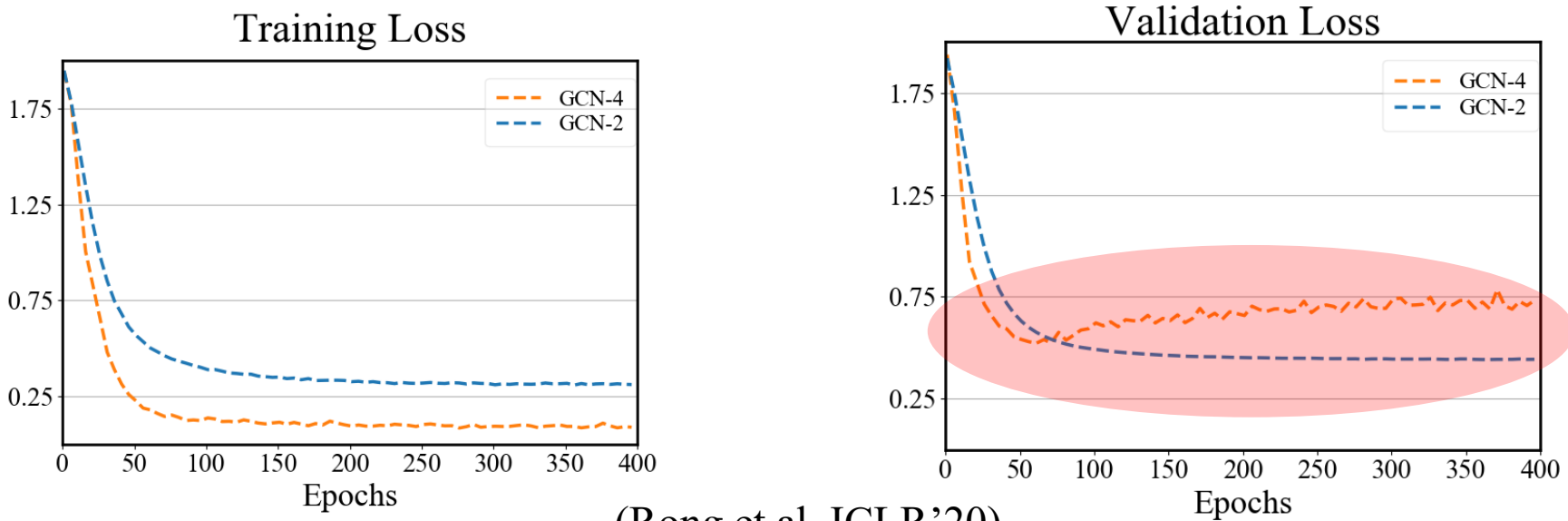
- Why do we need deep GNNs?
- Can GNNs simply go deep?
- What impedes GNNs to go deep?
  - Overfitting (Common)
  - Training dynamics (Common)
  - Over-smoothing (Graph Specific)
- How to make GNNs deep?

Training deep GNNs

- Why do we need deep GNNs?
- Can GNNs simply go deep?
- What impedes GNNs to go deep?
  - **Overfitting (Common)**
  - Training dynamics (Common)
  - Over-smoothing (Graph Specific)
- How to make GNNs deep?

Overfitting

- GNNs suffer from Overfitting



(Rong et al, ICLR'20)

- Too many parameters are established but only few of data points are provided

$$O(dh^2)$$

Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

Training deep GNNs

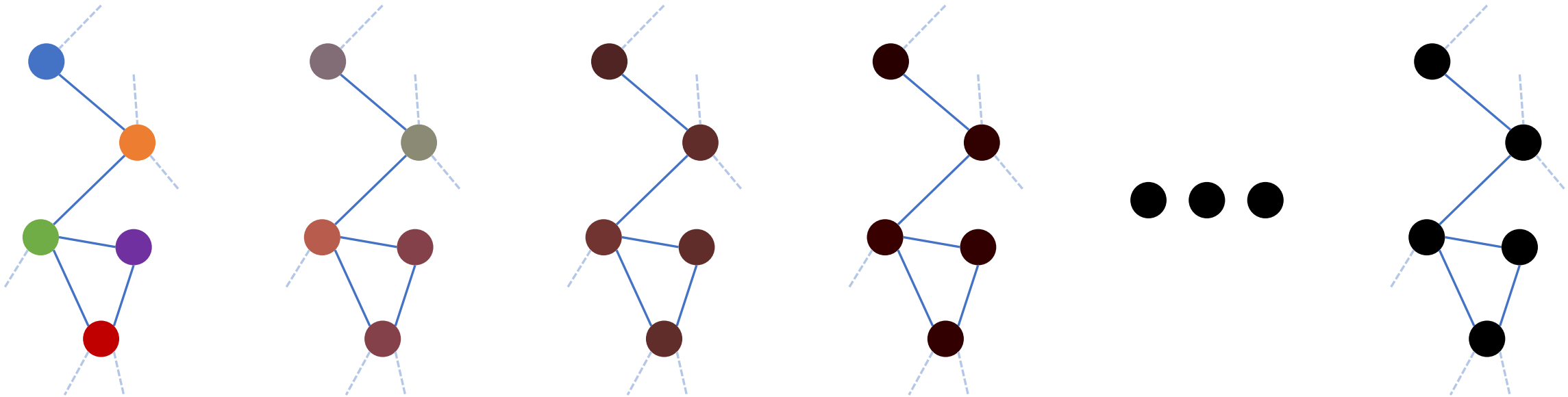
- Why do we need deep GNNs?
- Can GNNs simply go deep?
- What impedes GNNs to go deep?
  - Overfitting (Common)
  - **Training dynamics (Common)**
  - Over-smoothing (Graph Specific)
- How to make GNNs deep?

Training dynamics

l-layers gradient

$$\frac{dH_{l+1}}{dH_l} \cdot \frac{dH_l}{dH_{l-1}} \cdot \cdots \cdot \frac{dH_0}{dW_0} \leq (s_l \lambda_{m+1}) \cdot (s_{l-1} \lambda_{m+1}) \cdot \cdots \cdot \frac{dH_0}{dW_0}$$

The gradients vanish as the model go deep because $s_{1 \ldots l} \lambda_{m+1} < 1$



RGB as Features | Layer 1 | Layer 100 | Layer 200 | Layer 500
RGB=[0,0,0]

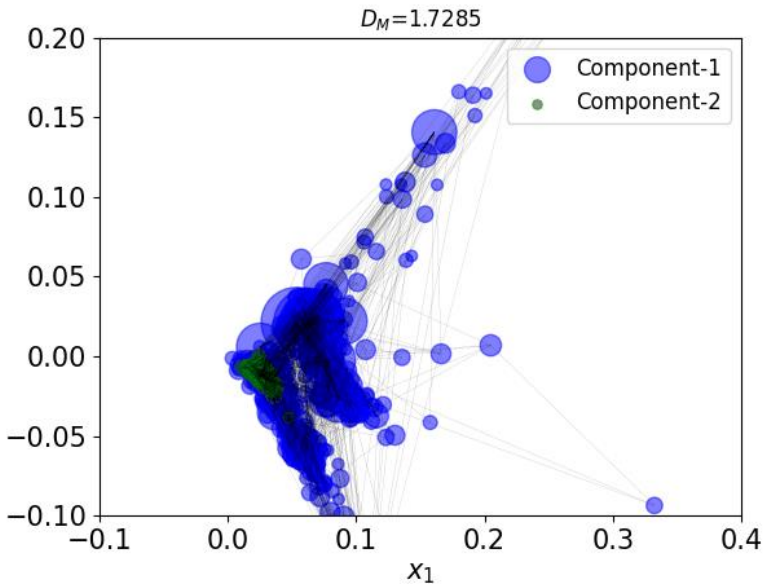
Training deep GNNs

- Why do we need deep GNNs?
- Can GNNs simply go deep?
- What impedes GNNs to go deep?
  - Overfitting (Common)
  - Training dynamics (Common)
  - **Over-smoothing (Graph Specific)**
- How to make GNNs deep?

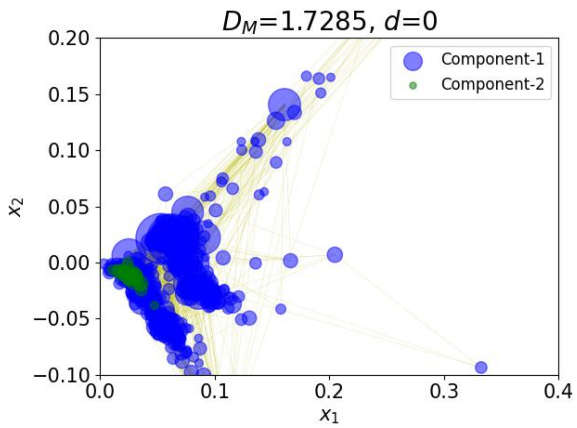
Over-Smoothing

- GNNs suffers from over-smoothing
  - Over-smoothing: node representations become **less distinguishable** with each other when the depth increases

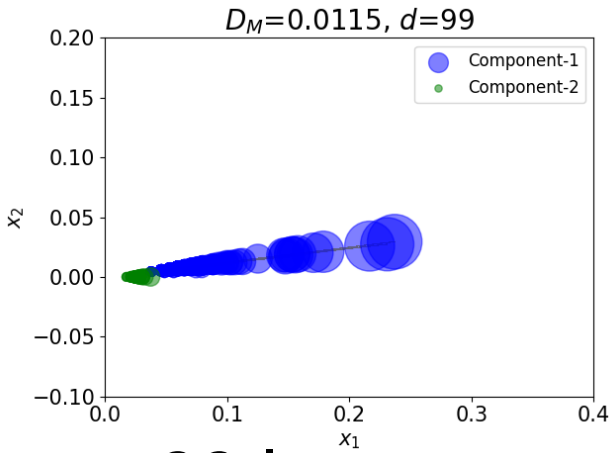
(Li et al. AAAI'18; Chen et al. AAAI'20; Oono et al. ICLR'20; Rong et al. ICLR'20; Huang et al. arXiv'20)



(Huang et al. arXiv'20)



Initial

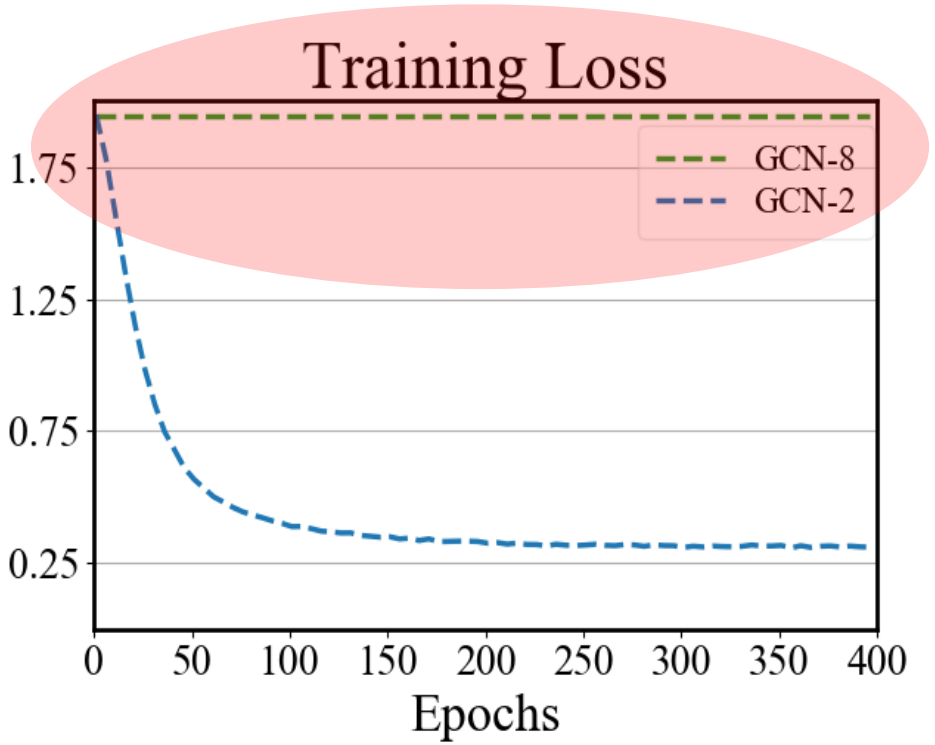


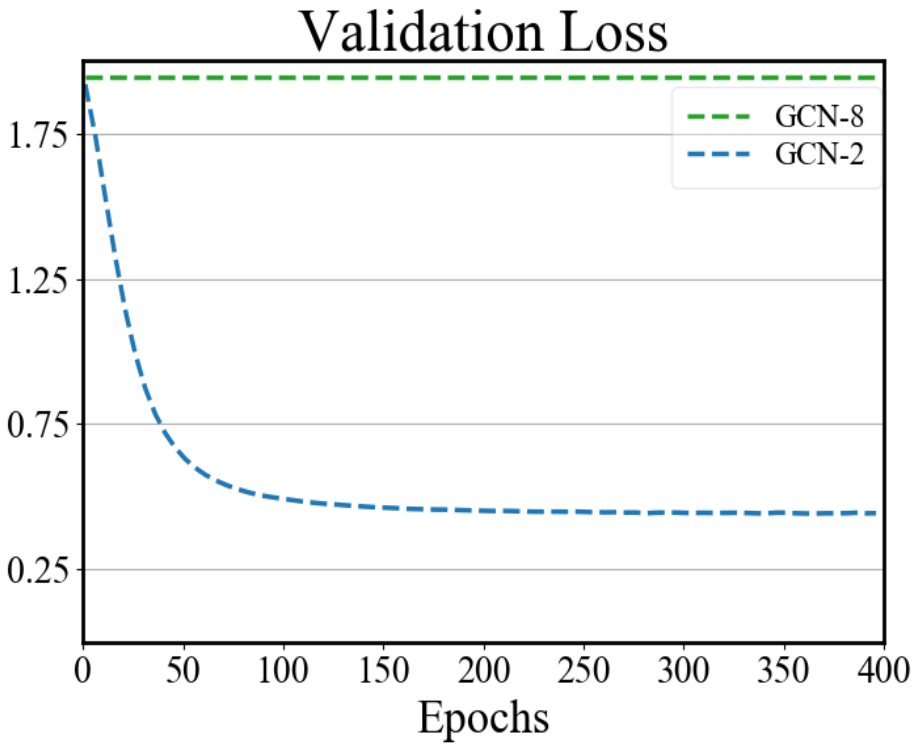
99 layers

Huang, Wenbing, et al. "Tackling Over-Smoothing for General Graph Convolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Over-Smoothing

- Over-smoothing also impedes training.

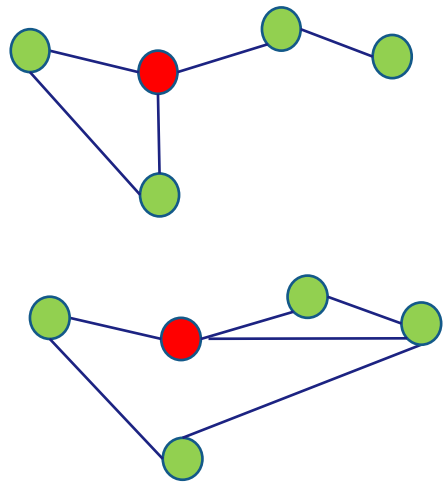




Huang, Wenbing, et al. "Tackling Over-Smoothing for General Graph Convolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

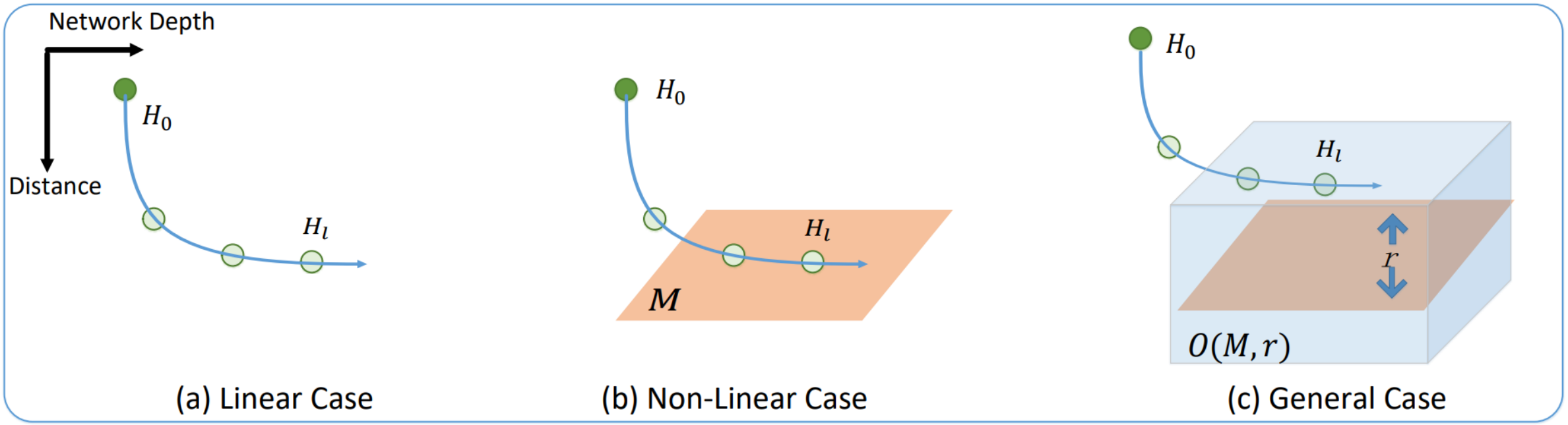
Over-Smoothing

- Why GNN works?
  - By message passing, GNN is able to capture the local structure;
  - Several works (Xu et al., 2019, Murphy et al., 2019) show that GNN is equivalent to the Weisfeiler-Lehman (WL) test under a careful design



Over-Smoothing

- When GCNs fail?
  - With linear activation
  - With ReLU activation
  - With ReLU and bias



Huang, Wenbing, et al. "Tackling Over-Smoothing for General Graph Convolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

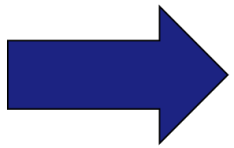
Over-Smoothing of Linear GCN

- When GCNs fail?
  - With linear activation

l-step Random Walk

Probability of walking

$$y = \hat{A}^l p_0 \text{ where } p_0(i) = 1/d(i)$$



Random Walks on Graph

- $V_{26} - V_{25} - V_{32} - V_3 - V_{10} \cdots$
- $V_5 - V_7 - V_{17} - V_6 - V_{11} \cdots$
- $V_{31} - V_{33} - V_{21} - V_{33} - V_{15}$

Tang, Jian, et al. "Line: Large-scale information network embedding." In WWW, 2015.

Li, Qimai, Zhichao Han, and Xiao-Ming Wu. "deep insights into graph convolutional networks for semi-supervised learning." *AAAI* 2018.

Over-Smoothing of Linear GCN

- When GCNs fail?
  - With linear activation

l-step Random Walk

$$y = \hat{A}^l p_0 \text{ where } p_0(i) = 1/d(i)$$

l-layer GCNs

$$Y = \hat{A}^l X W$$

Learnable Probability

Li, Qimai, Zhichao Han, and Xiao-Ming Wu. "deep insights into graph convolutional networks for semi-supervised learning." *AAAI* 2018.

Over-Smoothing of Linear GCN

- When GCNs fail?
  - With linear activation

l-step Random Walk

$$y = \hat{A}^l p_0 \text{ where } p_0(i) = 1/d(i)$$

l-layer GCNs

$$Y = \hat{A}^l XW$$

Eigen decomposition

$$Y = \sum_{i=1}^{n} \left(\lambda_i u_i u_i^\top\right)^l XW$$

Li, Qimai, Zhichao Han, and Xiao-Ming Wu. "deep insights into graph convolutional networks for semi-supervised learning." *AAAI* 2018.

Over-Smoothing of Linear GCN

- Rewrite eigen decomposition

Eigen decomposition

$$(\lambda_1 u_1 u_1^\top)^l XW + \cdots (\lambda_m u_m u_m^\top)^l XW + (\lambda_{m+1} u_{m+1} u_{m+1}^\top)^l XW + \cdots (\lambda_n u_n u_n^\top)^l XW$$

Li, Qimai, Zhichao Han, and Xiao-Ming Wu. "deep insights into graph convolutional networks for semi-supervised learning." *AAAI* 2018.

Over-Smoothing of Linear GCN

- Rewrite eigen decomposition

Eigen decomposition

$$(\lambda_1 u_1 u_1^\top)^l XW + \cdots (\lambda_m u_m u_m^\top)^l XW + (\lambda_{m+1} u_{m+1} u_{m+1}^\top)^l XW + \cdots (\lambda_n u_n u_n^\top)^l XW$$

- Suppose graph g has m connected components. It indicates

Eigenvalues

$$1 = \lambda_1 = \cdots = \lambda_m > \lambda_{m+1} > \cdots > \lambda_n > -1$$

Li, Qimai, Zhichao Han, and Xiao-Ming Wu. "deep insights into graph convolutional networks for semi-supervised learning." *AAAI* 2018.

Over-Smoothing of Linear GCN

- Rewrite eigen decomposition

Eigen decomposition

$$(\lambda_1 u_1 u_1^\top)^l XW + \cdots (\lambda_m u_m u_m^\top)^l XW + (\lambda_{m+1} u_{m+1} u_{m+1}^\top)^l XW + \cdots (\lambda_n u_n u_n^\top)^l XW$$

- Suppose graph g has m connected components. It indicates

Eigenvalues

$$1 = \lambda_1 = \cdots = \lambda_m > \lambda_{m+1} > \cdots > \lambda_n > -1$$

- When $l \to +\infty$, $\lambda_{m+1}, \ldots, \lambda_n \to 0$

Convergence

$$Y = \lim_{l \to \infty} \lambda_1^l u_1 u_1^\top XW + \cdots \lambda_m^l u_m u_m^\top XW + \lambda_{m+1}^l u_{m+1} u_{m+1}^\top XW + \cdots \lambda_n^l u_n u_n^\top XW$$

$\lambda_1^l \to 1$, $\lambda_m^l \to 1$, $\lambda_{m+1}^l \to 0$, $\lambda_n^l \to 0$

Li, Qimai, Zhichao Han, and Xiao-Ming Wu. "deep insights into graph convolutional networks for semi-supervised learning." *AAAI* 2018.

Over-Smoothing of Linear GCN

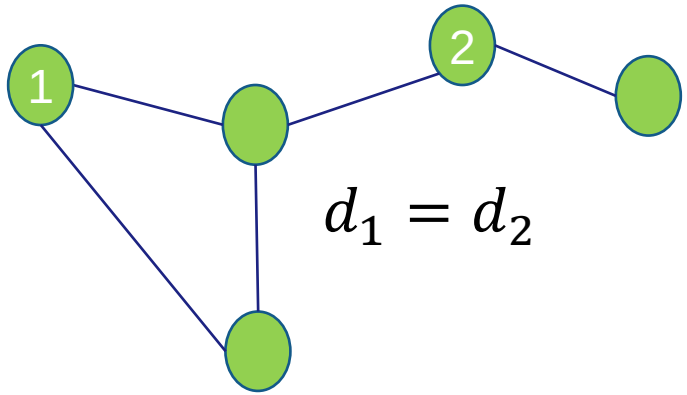
- When $l \to +\infty$, $\lambda_{m+1}, \ldots, \lambda_n \to 0$

Convergence

$$Y = u_1(u_1^\top XW) + \cdots u_m(u_m^\top XW), \text{where } u_i(j) = \tilde{d}_j^{\frac{1}{2}} \cdot \delta(\text{node } j \text{ in component } i)$$

degree

Node 1 is **indistinguishable** with node 2



Y

Node 1	$\tilde{d}_1^{\frac{1}{2}} z_1$	$\tilde{d}_1^{\frac{1}{2}} z_2$	$\tilde{d}_1^{\frac{1}{2}} z_3$	…
Node 2	$\tilde{d}_2^{\frac{1}{2}} z_1$	$\tilde{d}_2^{\frac{1}{2}} z_2$	$\tilde{d}_2^{\frac{1}{2}} z_3$	

The nodes within the same connected component are distinguishable only by their degrees

Li, Qimai, Zhichao Han, and Xiao-Ming Wu. "deep insights into graph convolutional networks for semi-supervised learning." *AAAI* 2018.

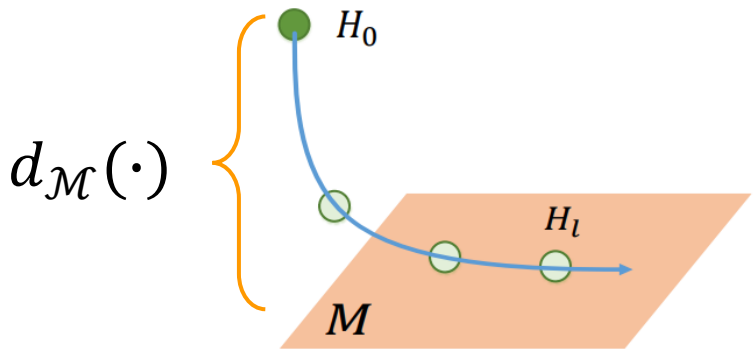
Over-Smoothing of Non-Linear GCN

- With ReLU activation
  - Similar to the linear case, but hard to derive the exact convergence point. Require the notion of subspace (Oono et al., ICLR'20):

$\mathcal{M}$ subspcae

Definition 1 (subspace). *Let* $\mathcal{M} := \{\boldsymbol{EC} | \boldsymbol{C} \in \mathbb{R}^{M \times C}\}$ *be an* M*-dimensional subspace in* $\mathbb{R}^{N \times C}$*, where* $\boldsymbol{E} \in \mathbb{R}^{N \times M}$ *is orthogonal,* i.e. $\boldsymbol{E}^{\mathrm{T}}\boldsymbol{E} = \boldsymbol{I}_M$*, and* $M \leq N$.

It is proved that (Oono et al., ICLR'20):
an infinite-layer GCN will converge to a certain point within a subspace $\mathcal{M}$



(b) Non-Linear Case

Oono, Kenta, and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification." *ICLR 2020.*

Over-Smoothing of Non-Linear GCN

Convergence of $\tilde{A}$

$$d_{\mathcal{M}}(\hat{A}X) \leq \lambda_{m+1} d_{\mathcal{M}}(X), \lambda_{m+1} < 1$$

Convergence

$$d_{\mathcal{M}}(H_{l+1}) = d_{\mathcal{M}}(\sigma(\hat{A}H_l W)) \leq \lambda_{m+1} s d_{\mathcal{M}}(H_l)$$

Oono, Kenta, and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification." *ICLR 2020.*

Over-Smoothing of Non-Linear GCN

Convergence of $\tilde{A}$

$$d_{\mathcal{M}}(\hat{A}X) \leq \lambda_{m+1} d_{\mathcal{M}}(X), \lambda_{m+1} < 1$$

Convergence of W

$$d_{\mathcal{M}}(XW_l) \leq s d_{\mathcal{M}}(X), s \leq 1$$

Convergence

$$d_{\mathcal{M}}(H_{l+1}) = d_{\mathcal{M}}\left(\sigma(\hat{A}H_l W)\right) \leq \lambda_{m+1} s d_{\mathcal{M}}(H_l)$$

Oono, Kenta, and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification." *ICLR 2020.*

Over-Smoothing of Non-Linear GCN

Convergence of $\tilde{A}$

$$d_{\mathcal{M}}(\hat{A}X) \leq \lambda_{m+1} d_{\mathcal{M}}(X), \lambda_{m+1} < 1$$

Convergence of W

$$d_{\mathcal{M}}(XW) \leq s d_{\mathcal{M}}(X), \qquad s_l \leq 1$$

Convergence of ReLU

$$d_{\mathcal{M}}(\sigma(X)) \leq d_{\mathcal{M}}(X)$$

Convergence

$$d_{\mathcal{M}}(H_{l+1}) = d_{\mathcal{M}}\left(\sigma(\hat{A}H_l W)\right) \leq \lambda_{m+1} s d_{\mathcal{M}}(H_l)$$

Oono, Kenta, and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification." *ICLR 2020.*

Over-Smoothing of GCNs with bias

- With ReLU and bias

GCNs with bias

$$H_{l+1} = \sigma\left(\hat{A}H_l W_l + b_l\right)$$

Huang, Wenbing, et al. "Tackling Over-Smoothing for General Graph Convolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Over-Smoothing of GCNs with bias

- H_L converges to a certain sub-cube $O(\mathcal{M}, r)$ with ReLU and bias

GCNs with bias

$$H_{l+1} = \sigma\left(\hat{A}H_l W + b\right)$$

Convergence of bias

$$d_{\mathcal{M}}(H_{l+1}) \leq \lambda_{m+1} s d_{\mathcal{M}}(H_l) + d_{\mathcal{M}}(b)$$

Huang, Wenbing, et al. "Tackling Over-Smoothing for General Graph Convolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Over-Smoothing of GCNs with bias

- With ReLU and bias

GCNs with bias

$$H_{l+1} = \sigma\left(\hat{A}H_l W + b\right)$$

Convergence of bias

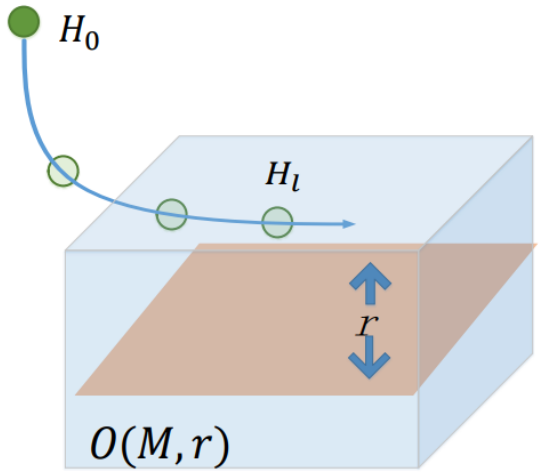
$$d_{\mathcal{M}}(H_{l+1}) \leq \lambda_{m+1} s d_{\mathcal{M}}(H_l) + d_{\mathcal{M}}(b)$$

GCN with bias

H_l converges to a certain sub-cube:

$$O(\mathcal{M}, r) = \{H_l | d_{\mathcal{M}}(H_l) < \frac{d_{\mathcal{M}}(b)}{1-\lambda_{m+1} s}\}$$

(Huang et al. arXiv'20)



(c) General Case

Huang, Wenbing, et al. "Tackling Over-Smoothing for General Graph Convolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Summary

Linear GCN

H_l converges to a certain point that can be exactly derived

Non-linear GCN

H_l converges to a certain point within a certain subspace

GCN with bias

H_l converges to a certain point within a certain sub-cube

Huang, Wenbing, et al. "Tackling Over-Smoothing for General Graph Convolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Other methods to measure over-smoothing

- One can explicitly measure over-smoothing using distance between node pairs (Chen et al., 2020)

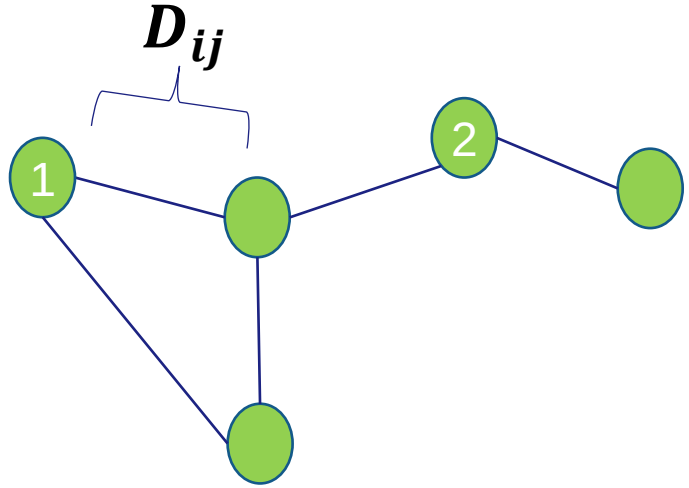
Pair Distance

$$D_{ij} = 1 - \frac{H_i \cdot H_j}{|H_i||H_j|}$$

The i-th row of the output

Mean Average Distance

$$\mathrm{MAD}_i = \frac{\sum_{i=0}^{n} \bar{D}_i^{tgt}}{\sum_{i=0}^{n} \mathbb{I}(\bar{D}_i^{tgt})}$$



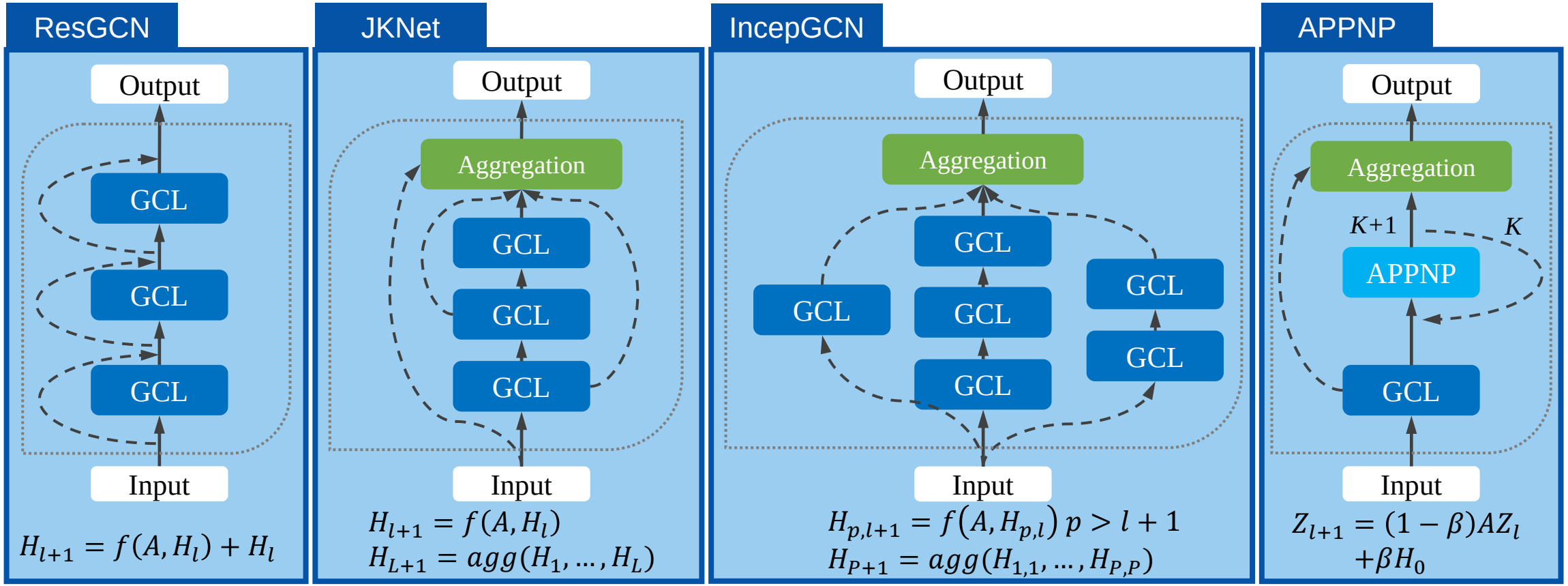
- Other metrics see PairNorm (Zhao et al., 2020); GroupNorm (Zhou et al., 2020)

Chen, et al. “Measuring and Relieving the Over-smoothing Problem for Graph Neural Networks from the Topological View.” AAAI 2020
Zhao, Lingxiao, and Leman Akoglu. "PairNorm: Tackling Oversmoothing in GNNs." *ICLR. 2020.*
Zhou et al. "Towards Deeper Graph Neural Networks with Differentiable Group Normalization Kaixiong Zhou Texas A&M University zkxiong@tamu.edu Xiao Huang The Hong." *NIPS 2020.*

Training deep GNNs

- Why do we need deep GNNs?
- Can GNNs simply go deep?
- What impedes GNNs to go deep?
  - Overfitting (Common)
  - Training dynamics (Common)
  - Over-smoothing (Graph Specific)
- How to make GNNs deep?
  - Architecture refinement
  - Manipulating input (DropEdge)
  - Layer normalizations

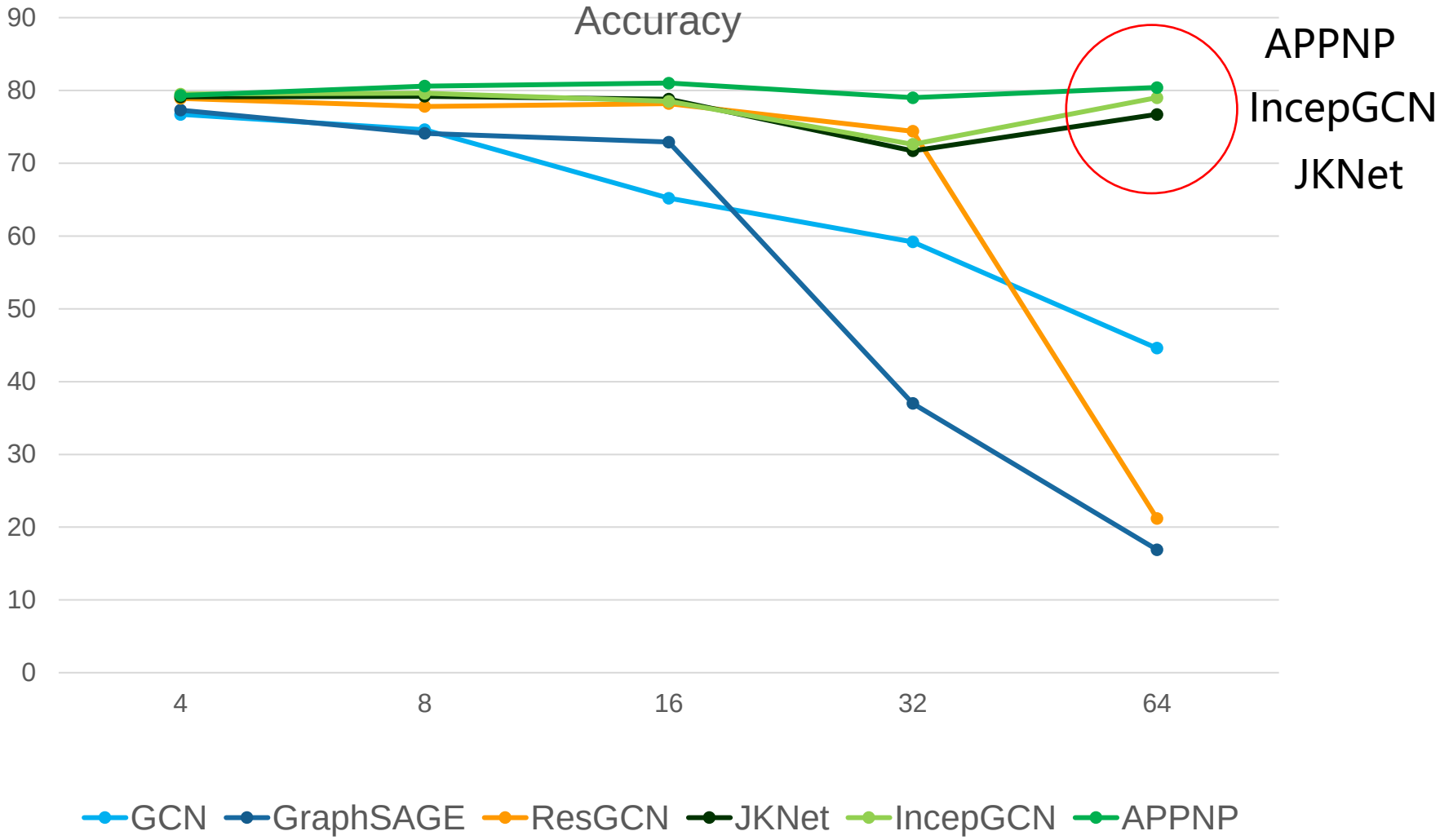
Shortcuts in Structures



- Other architectures including SGC (Wu et al., 2019), GCNII (Chen et al., 2020), etc.

Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

Shortcuts in Structures



Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

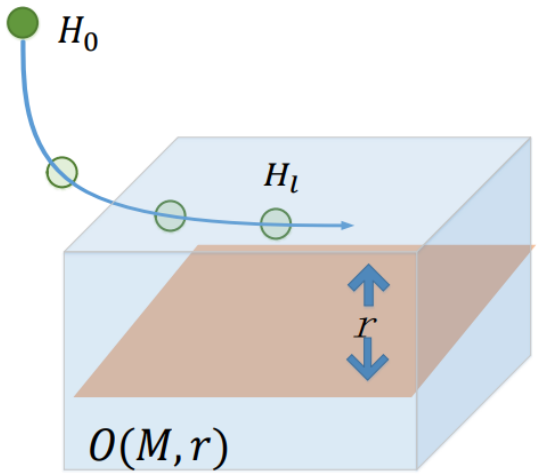
Shortcuts in Structures

- Residual connections are helpful (akin to CNNs)
- Do residual connections alleviate over-smoothing?

GCN with bias

$$d_{\mathcal{M}}(H_{l+1}) - r \leq v(d_{\mathcal{M}}(H_l) - r)$$

Indeed, general GCNs converge to a certain sub-cube $O(\mathcal{M}, r)$ with speed v^{-1} and radius r.



(c) General Case

Huang, Wenbing, et al. "Tackling Over-Smoothing for General Graph Convolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Shortcuts in Structures

General Case

Converging to sub−cube with speed v^{-1} and radius r

Basic GCN

Generic GCN

$$v = s\lambda_{m+1}$$
$$r = 0$$

GCN with bias

$$v = s\lambda_{m+1}$$
$$r = \frac{d_{\mathcal{M}}(b)}{1-v}$$

Different Structures

ResGCN

$$v = s\lambda_{m+1} + \alpha$$
$$r = 0$$

APPNP

$$v = (1-\beta)\ \lambda_{m+1}$$
$$r = \frac{\beta d_{\mathcal{M}}(H_0)}{1-v}$$

Huang, Wenbing, et al. "Tackling Over-Smoothing for General GraphConvolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Shortcuts in Structures

General Case

General models converge to $O(\mathcal{M}, r)$ with speed v^{-1}, when $l \to \infty$

Basic GCN

Generic GCN

$$v = s\lambda_{m+1}$$
$$r = 0$$

GCN with bias

$$v = s\lambda_{m+1}$$ r is nonzero

$$\boldsymbol{r = \frac{d_{\mathcal{M}}(b)}{1 - v}}$$

Different Structures

ResGCN

$$v = s\lambda_{m+1} + \alpha$$
$$r = 0$$

APPNP

$$v = (1 - \beta)\ \lambda_{m+1}$$
$$r = \frac{\beta d_{\mathcal{M}}(H_0)}{1 - v}$$

Huang, Wenbing, et al. "Tackling Over-Smoothing for General GraphConvolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Shortcuts in Structures

General Case

General models converge to $O(\mathcal{M}, r)$ with speed v^{-1}, when $l \to \infty$

Basic GCN

Generic GCN

$$v = s\lambda_{m+1}$$
$$r = 0$$

GCN with bias

$$v = s\lambda_{m+1}$$
$$r = \frac{d_{\mathcal{M}}(b)}{1 - v}$$

Different Structures

ResGCN

v is increased

$$\boldsymbol{v = s\lambda_{m+1} + \alpha}$$
$$r = 0$$

APPNP

$$v = (1 - \beta)\ \lambda_{m+1}$$
$$r = \frac{\beta d_{\mathcal{M}}(H_0)}{1 - v}$$

Huang, Wenbing, et al. "Tackling Over-Smoothing for General GraphConvolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Shortcuts in Structures

General Case

General models converge to $O(\mathcal{M}, r)$ with speed v^{-1}, when $l \to \infty$

Basic GCN

Generic GCN

$$v = s\lambda_{m+1}$$
$$r = 0$$

GCN with bias

$$v = s\lambda_{m+1}$$
$$r = \frac{d_{\mathcal{M}}(b)}{1 - v}$$

Different Structures

ResGCN

$$v = s\lambda_{m+1} + \alpha$$
$$r = 0$$

APPNP

$$\boldsymbol{v = (1-\beta)\ \lambda_{m+1}}$$
$$\boldsymbol{r = \frac{\beta d_{\mathcal{M}}(H_0)}{1 - v}}$$

v is decreased

r is nonzero

Huang, Wenbing, et al. "Tackling Over-Smoothing for General GraphConvolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Over-smoothing Layer

- For all cases, the over-smoothing speed v^{-1} is controlled by $\boldsymbol{\lambda_{m+1}}$ (the second-biggest eigenvalue of normalized adjacency matrix)

- So how to increase λ_{m+1}?

Alleviate Over-Smoothing by DropEdge

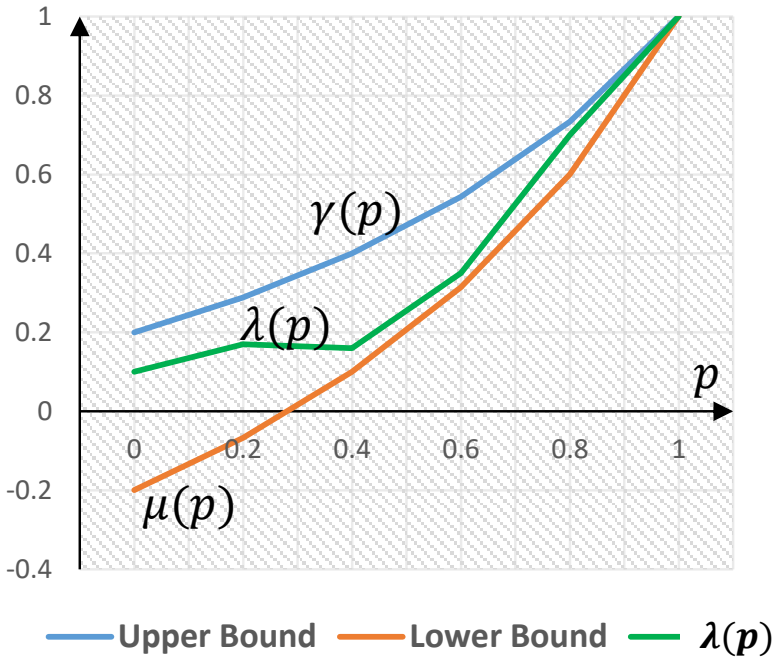
- So how to increase λ_{m+1}? Drop Edges!
- In expectation:

Upper bound

$$\mu(p) \leq \lambda_{m+1}(p) \leq \gamma(p)$$

Lower bound

Dropping rate



- Both μ and γ monotonically increase w.r.t. p;
- The gap $\gamma - \mu$ monotonically decreases w.r.t. p;

Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

Huang, Wenbing, et al. "Tackling Over-Smoothing for General GraphConvolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Alleviate Over-Smoothing by DropEdge

- Huang et al., 2020 has considered the re-normalization trick in our analyses, in contrast to Rong et al., 2020

Theorem 1. *We denote the original graph as $\mathcal{G}$ and the one after dropping certain edges out as $\mathcal{G}'$. Given a small value of ϵ, we assume $\mathcal{G}$ and $\mathcal{G}'$ will encounter the ϵ-smoothing issue with regard to subspaces $\mathcal{M}$ and $\mathcal{M}'$, respectively. Then, either of the following inequalities holds after sufficient edges removed.*

- *The relaxed smoothing layer only increases:* $\hat{l}(\mathcal{M}, \epsilon) \leq \hat{l}(\mathcal{M}', \epsilon)$;
- *The information loss is decreased:* $N - dim(\mathcal{M}) > N - dim(\mathcal{M}')$.

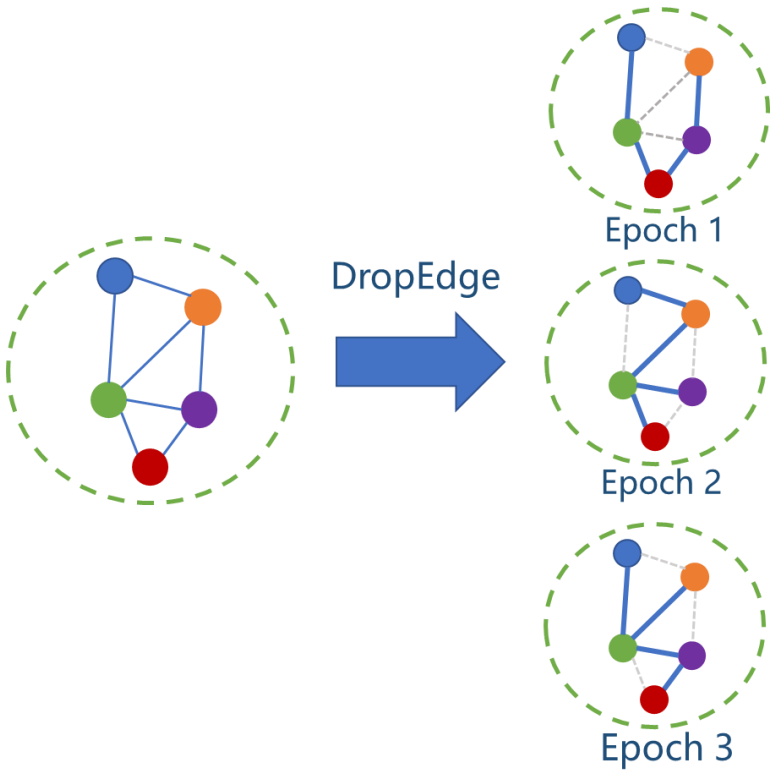
(Rong et al., 2020)

Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

Huang, Wenbing, et al. "Tackling Over-Smoothing for General GraphConvolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Alleviate Over-Smoothing by DropEdge

- Besides, DropEdge can prevent over-fitting as well!



Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

Huang, Wenbing, et al. "Tackling Over-Smoothing for General GraphConvolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Alleviate Over-Smoothing by Adjacency Matrix

- DropEdge results

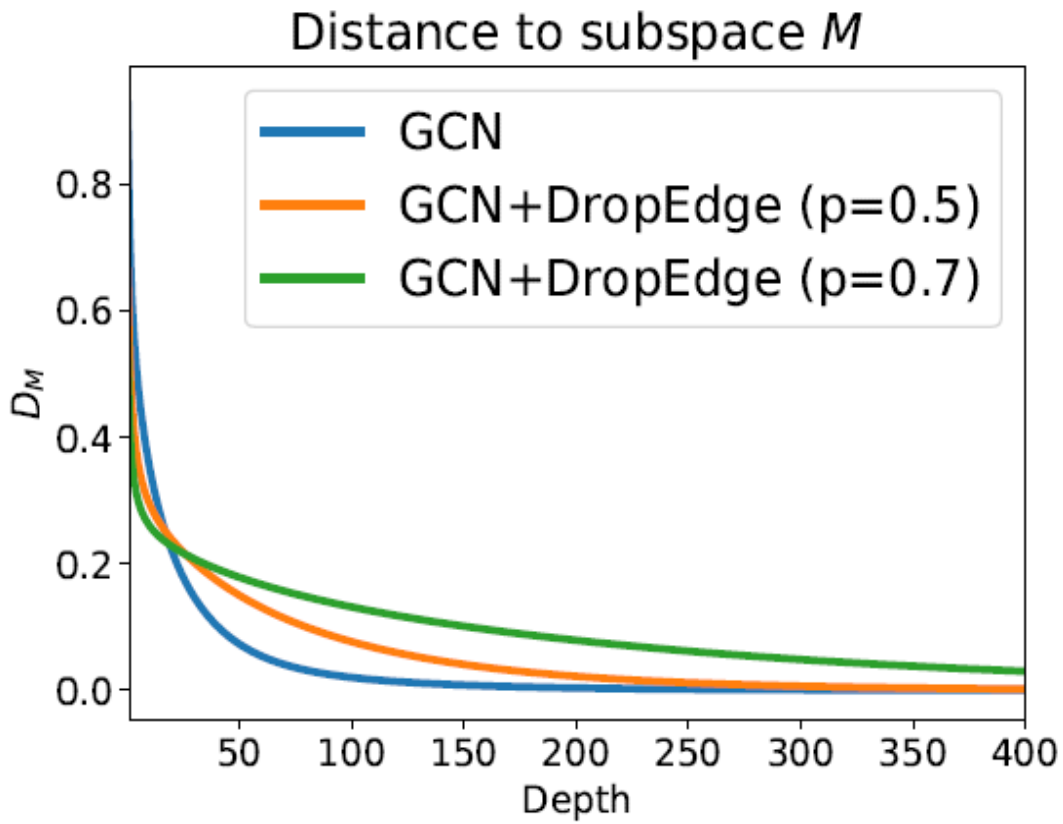
Citeseer	4 layers	DropEdges	16 layers	DropEdges	64 layers	DropEdges
GCN	76.7	79.2(+2.5)	65.2	76.8(+11.6)	44.6	45.6(+1.0)
ResGCN	78.9	78.8(-0.1)	78.2	79.4(+1.2)	21.2	75.3(+54.1)
JKNet	79.1	80.2(+1.1)	78.8	80.1(+1.3)	76.7	80.0(+3.3)
IncepGCN	79.5	79.9(+0.4)	78.5	80.2(+1.7)	79.0	79.9(+0.9)
GraphSAGE	77.3	79.2(+1.9)	72.9	74.5(+1.6)	16.9	25.1(+8.2)
APPNP	80.3	80.8(+0.5)	80.2	81.1(+0.9)	80.4	81.3(+0.9)

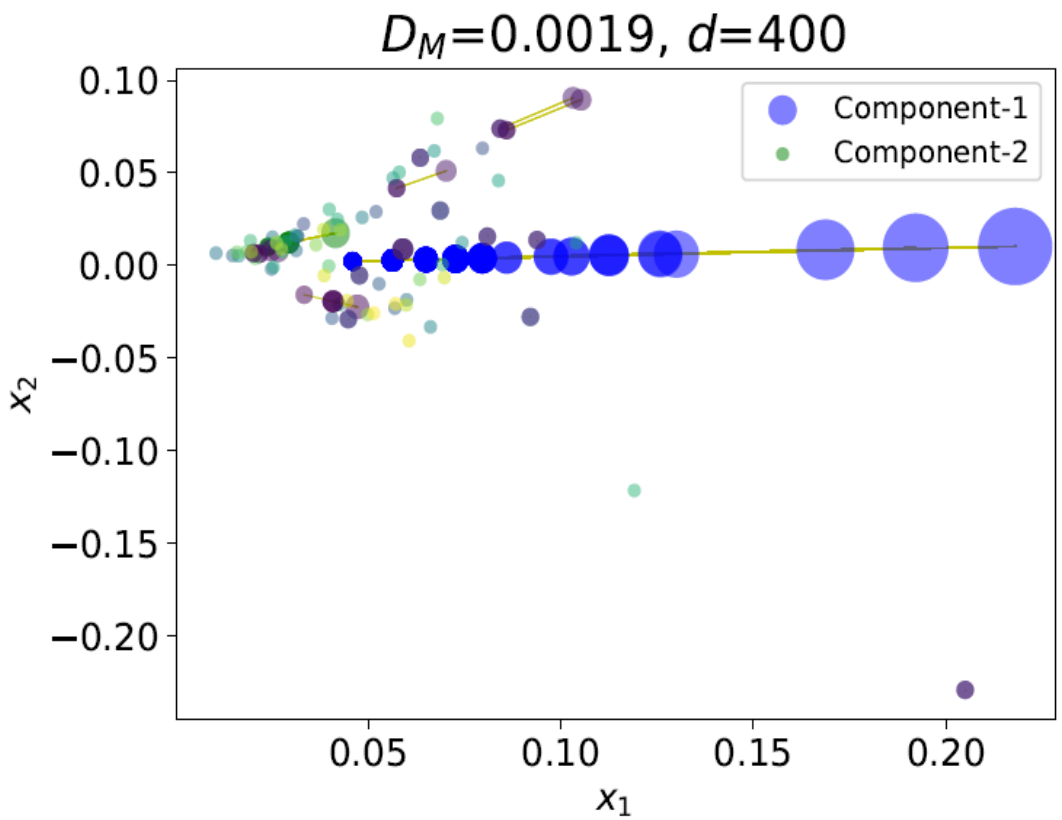
Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

Huang, Wenbing, et al. "Tackling Over-Smoothing for General GraphConvolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Alleviate Over-Smoothing by Adjacency Matrix

- DropEdge results





Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

Huang, Wenbing, et al. "Tackling Over-Smoothing for General GraphConvolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Alleviate Over-Smoothing by Weights

Convergence speed

$$v = \lambda_{m+1} s$$

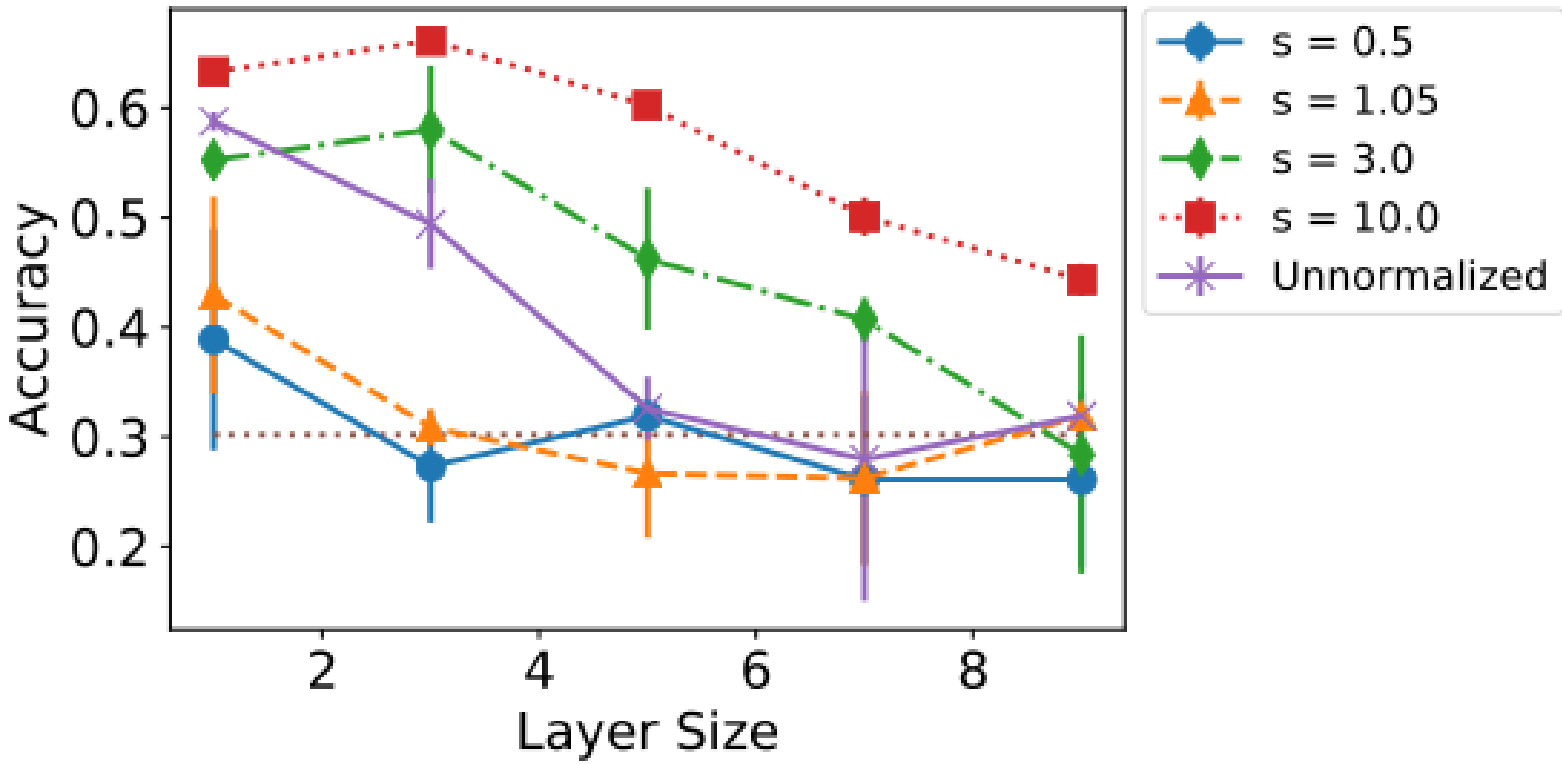
Weights

- Similarly, increasing s will also increase v. So how to increase s? Increase the initial W_ls.

Oono, Kenta, and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification." *ICLR 2020.*

Alleviate Over-Smoothing by Weights

- Try different s as initial



Oono, Kenta, and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification." *ICLR 2020.*

Training deep GNNs

- Why do we need deep GNNs?
- Can GNNs simply go deep?
- What impedes GNNs to go deep?
  - Overfitting (Common)
  - Training dynamics (Common)
  - Over-smoothing (Graph Specific)
- How to make GNNs deep?
  - Architecture refinement
  - Manipulating input (DropEdge)
  - Layer normalizations

Pair Norm: Center and Rescale

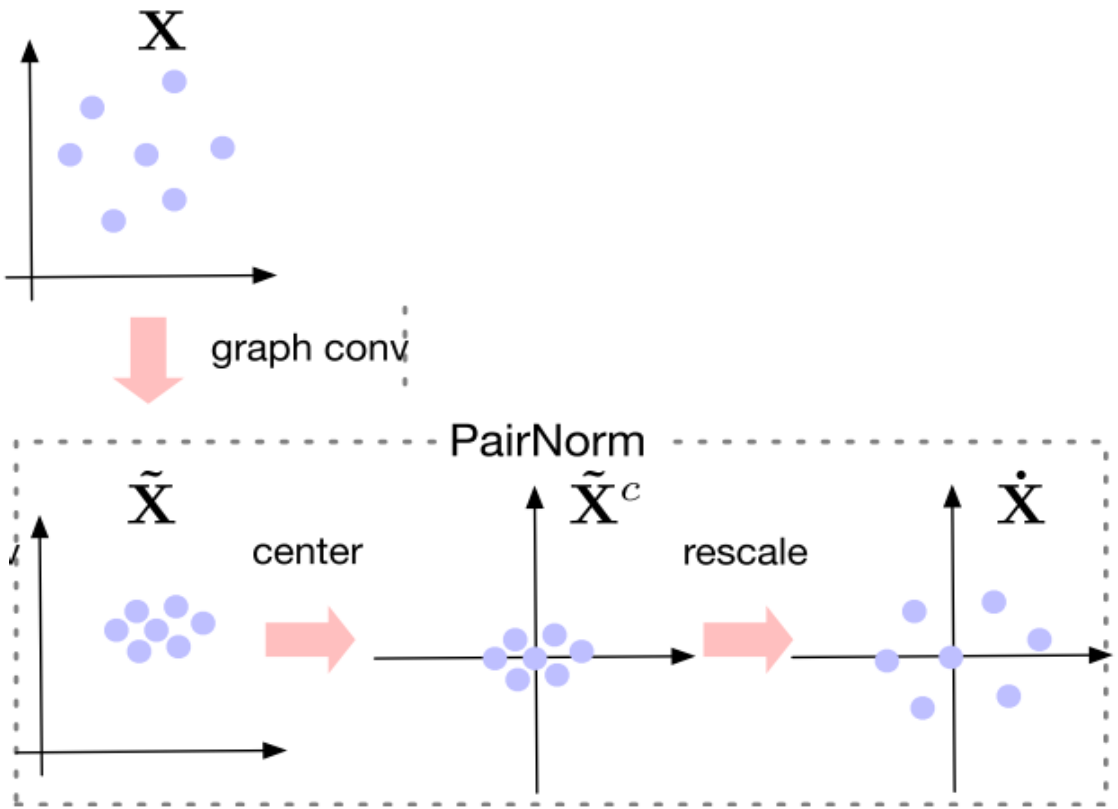
- PairNorm: Center and rescale (normalize) GCN outputs $\tilde{X} := \mathrm{GCN}(A, X)$ to keep the **total pairwise squared distance** *unchanged*

Center

$$\tilde{x}_i^c = \tilde{x}_i - \frac{1}{n}\sum_{i=1}^{n} \tilde{x}_i$$

Rescale

$$\dot{x}_i = s\sqrt{n}\frac{\tilde{x}_i^c}{\sqrt{\|\tilde{X}^c\|_F^2}}$$



- See also GroupNorm (Zhou et al., 2020)

Zhao, Lingxiao, and Leman Akoglu. "PairNorm: Tackling Oversmoothing in GNNs." *ICLR. 2020.*

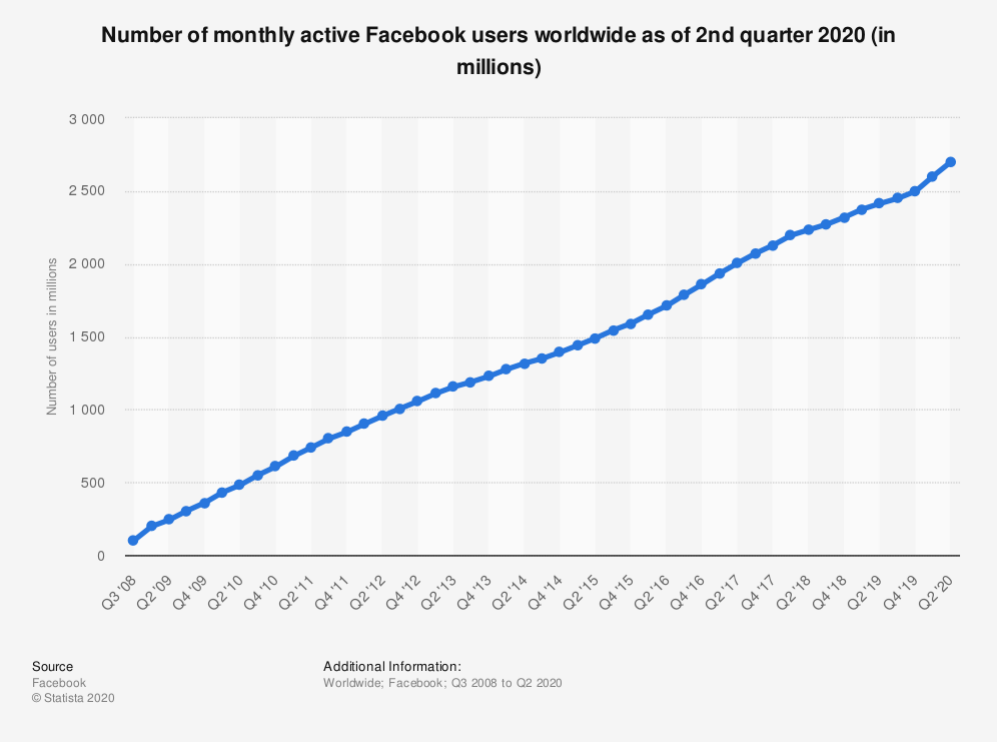
Break

5 min

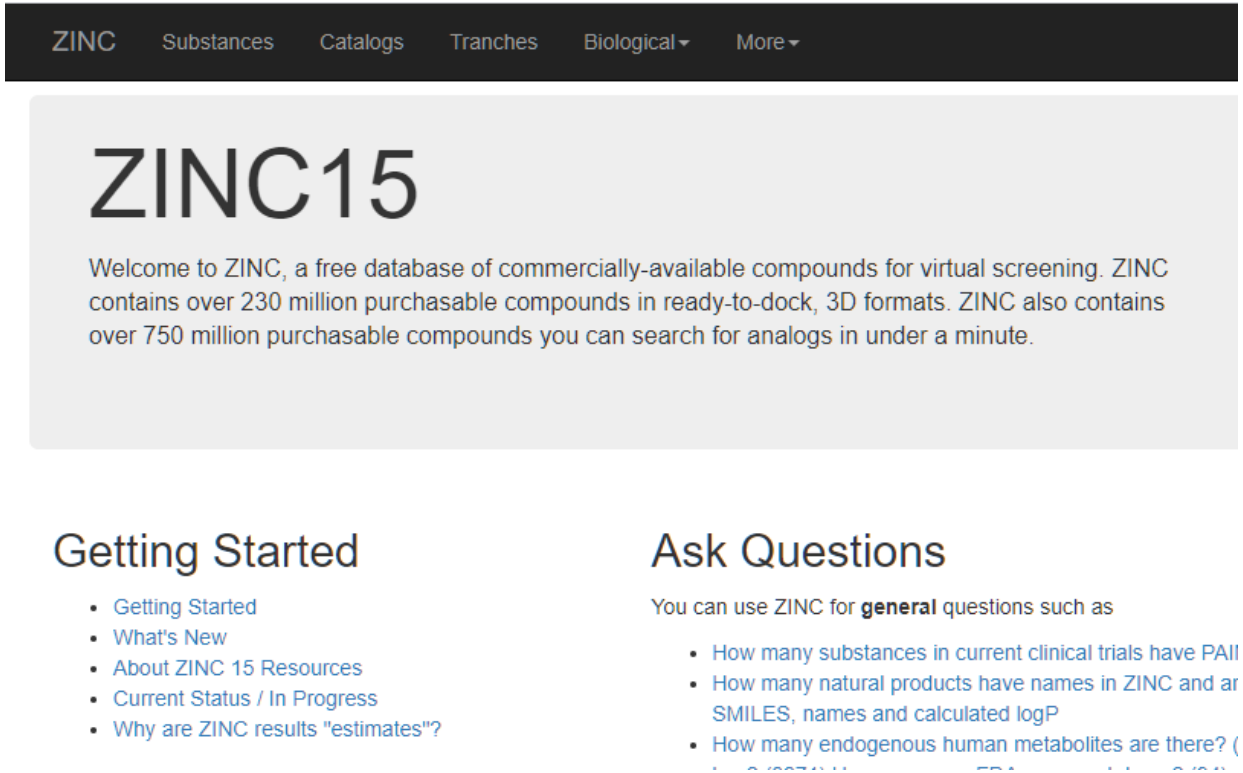
Scalability of GNNs

Graph in the Real-world Can be Very Large

- Large scale:



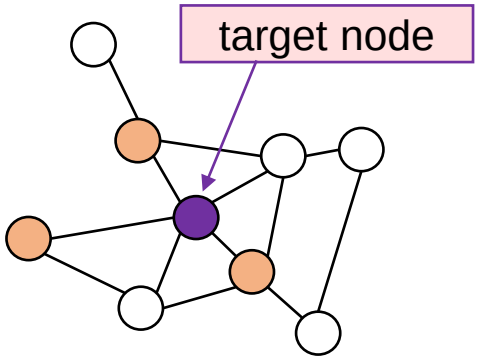
Large number:



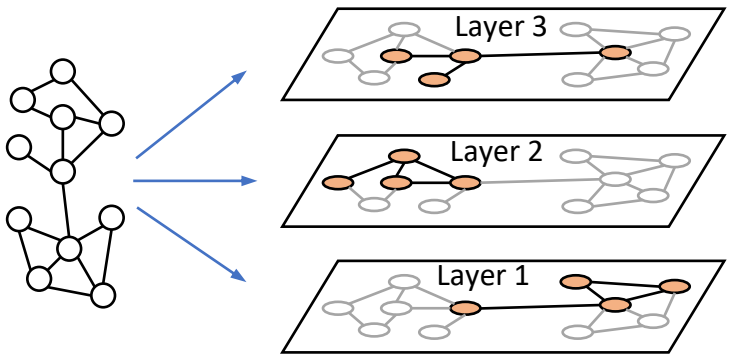
https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/
https://zinc15.docking.org/

Three Paradigms

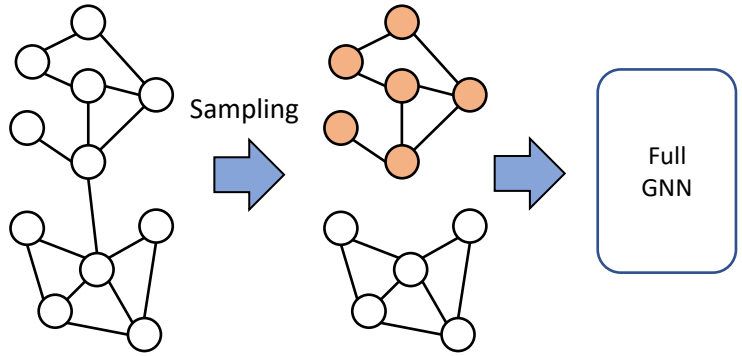
- Why the original GNN fails on large graph?
  - Large memory requirement.
  - Inefficient gradient update.
- Three paradigms toward large-scale GNN:



Node-wise Sampling



Layer-wise Sampling

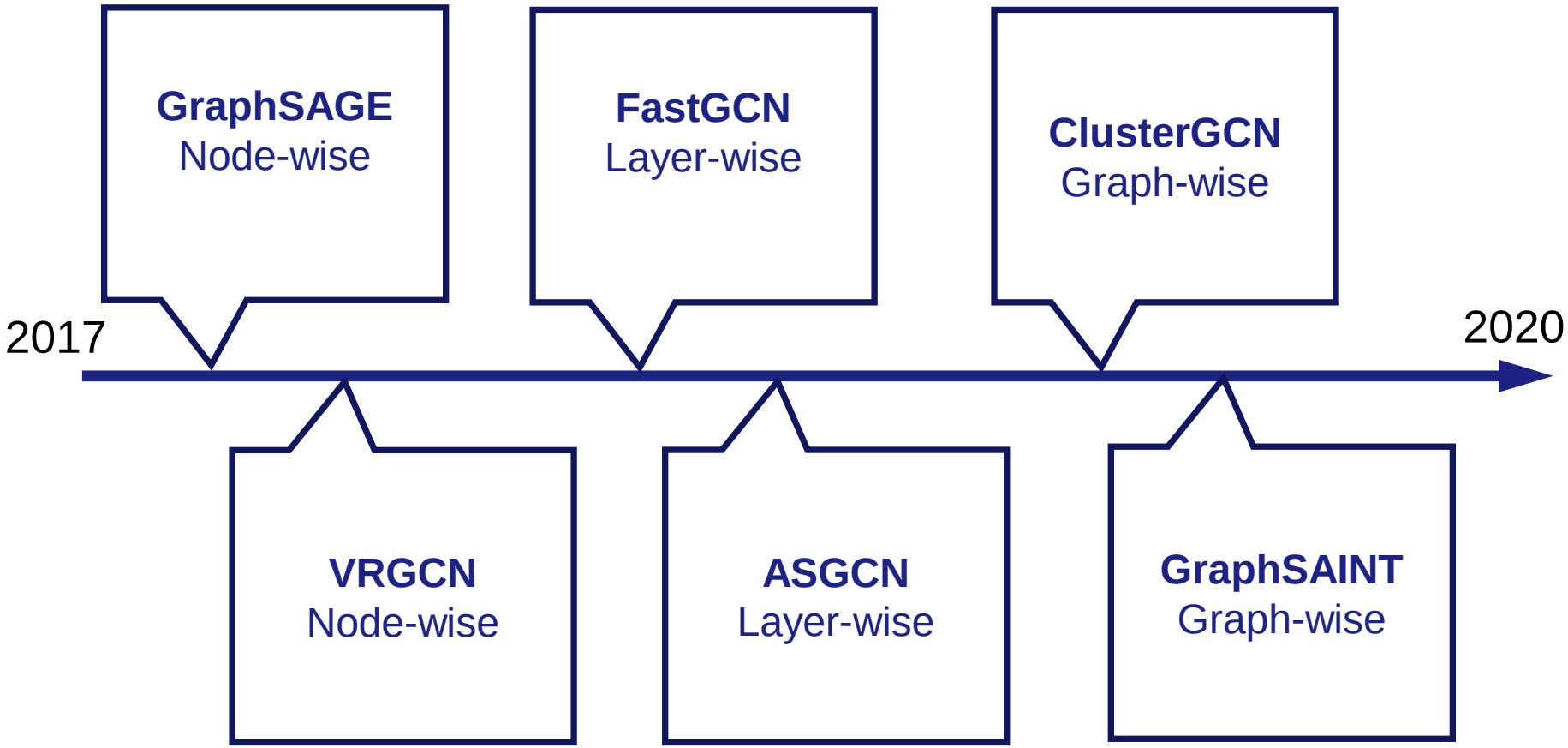


Graph-wise Sampling

Two issues towards the large-scale GNNs

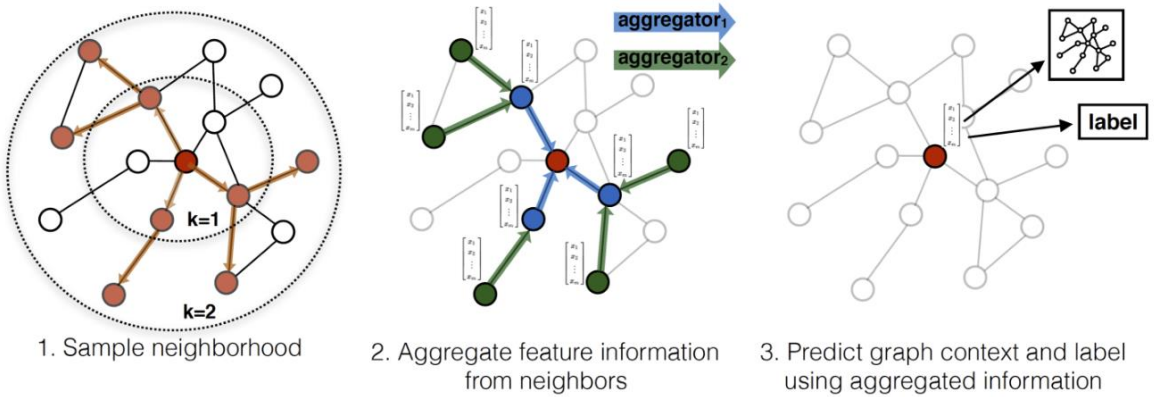
- How to design efficient sampling algorithm?
- How to guarantee the sampling quality?

Overview



Acknowledgement: The idea of this slide is taken from: https://drive.google.com/file/d/1Va7WaXOcNS37z779WLbZ-H54mXEJy8cB/view?usp=sharing

Node-wise Sampling: GraphSAGE



- The architecture:

$$\mathbf{h}_{\mathcal{N}(v)}^{k} \leftarrow \mathrm{AGGREGATE}_{\mathrm{k}}(\{\mathbf{h}_{u}^{k-1}, \forall \mathrm{u} \in \mathcal{N}(v)\})$$

$$\boldsymbol{h}_{v}^{k} \leftarrow \sigma(\mathbf{W}^{\mathrm{k}} \cdot \mathrm{CONCAT}(\mathbf{h}_{v}^{k-1}, \mathbf{h}_{\mathcal{N}(v)}))$$

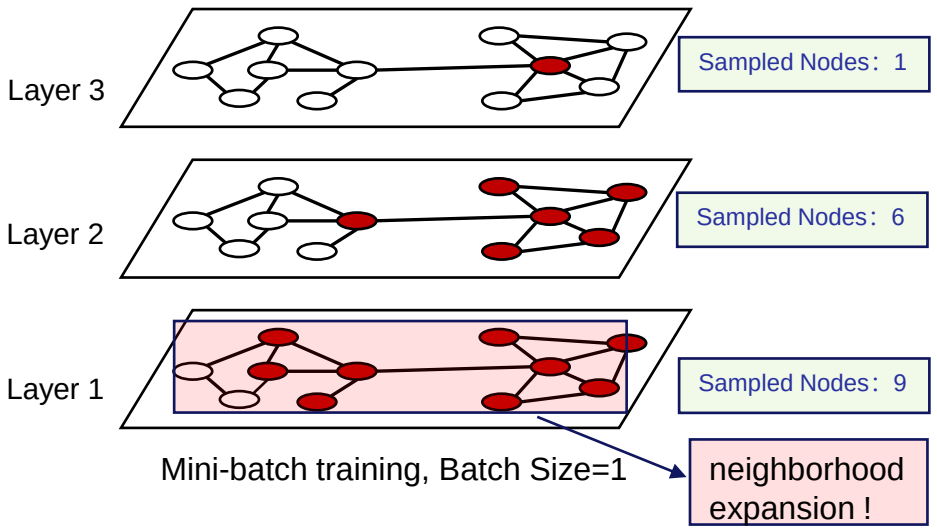
Generalized Aggregator
- Mean aggregator (GCN)
- Pooling aggregator
- LSTM aggregator
- .....

Use Concertation instead of SUM

Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs."

Towards large-scale GraphSAGE

Mini-batch training

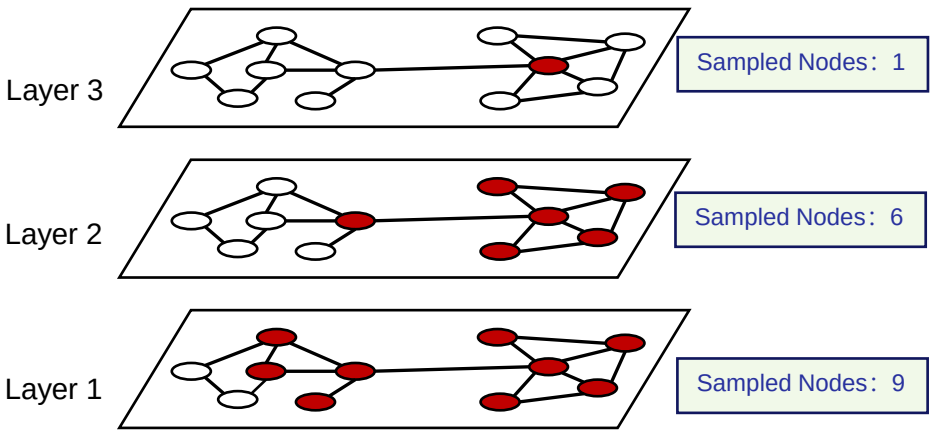


- Sample target nodes as a mini-batch;
- Only consider the nodes that used to compute the representation in the batch.

Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs."

Towards large-scale GraphSAGE

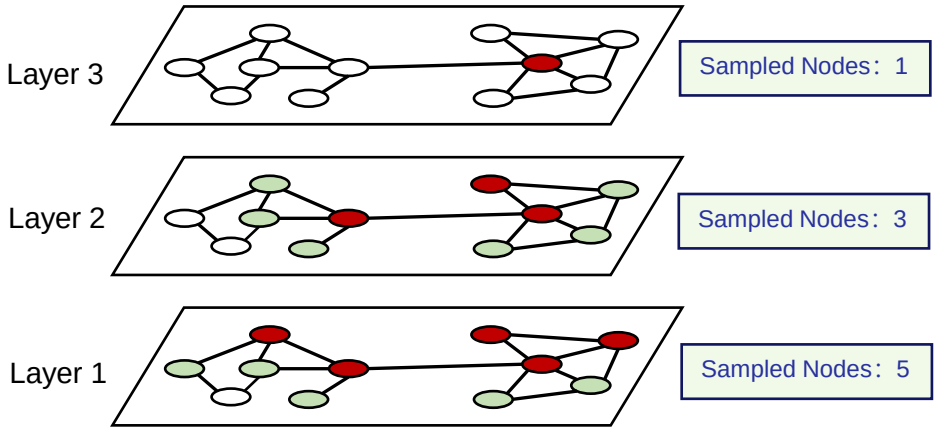
Mini-batch training



Mini-batch training, Batch Size=1

- Sample target nodes as a mini-batch;
- Only consider the nodes that used by computing the representation in the batch.

Fixed-size neighbor sampling



Fix-size neighbor sampling S=2

- Sample fixed-size (S_i for layer i) set of neighbors for computing.
- Number of nodes at input layer: $O(|V|^K) \rightarrow O(\prod_{i=1}^{K} S_i)$

Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs."

Node-wise Sampling: GraphSAGE

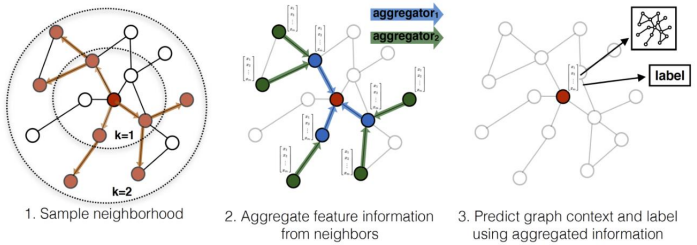
- Pros:
  - Generalized aggregator.
  - Mini-batch training and fixed-size neighbor sampling.
- Cons:
  - Neighborhood expansion on deeper GNNs.
  - No guarantees for the sampling quality.

Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs."

GraphSAGE Extension: PinSAGE

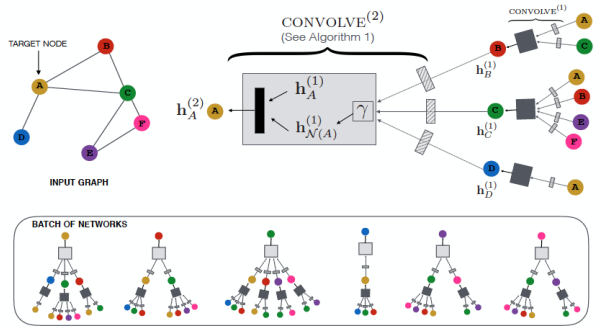
- An early-stage industrial GNN-based recommendation system.

GraphSAGE



- K-hop Neighbor
- Random Sampling
- MEAN/MAX Pooling

PinSAGE



- Random Walk
- Importance Sampling
- Importance Pooling
- Curriculum Training
- MapReduce Inference

Ying, Rex, et al. "Graph convolutional neural networks for web-scale recommender systems."

Node-wise Sampling: VR-GCN

$\mathcal{N}(u)$: the neighbor set of node u.
$\widehat{\mathcal{N}}^{(l)}(u) \subset \mathcal{N}(u)$: the sampled neighbor set of node u at layer l.
$\boldsymbol{P}$: the normalized adjacency matrix.
$\widehat{\boldsymbol{P}}^{(l)}$: the sampled normalized adjacency matrix at layer l.
$D^{(l)}$:the sampled size for each node at layer l/

- GraphSAGE: Neighborhood Sampling

$$\mathrm{NS}_u^{(l)} \coloneqq R \sum_{v \in \widehat{\mathcal{N}}^{(l)}(u)} P_{uv} \boldsymbol{h}_v^{(l)}, \; R = \mathcal{N}(u)/D^{(l)}$$

  - biased sampling / larger variance → larger sample size $\widehat{\mathcal{N}}^{(l)}(u)$.

- Control Variate Based Estimator (**CV Sampler**):
  - Maintain the historical hidden embedding $\bar{\boldsymbol{h}}_v^{(l)}$for a better estimation.
  - **Variance reduction → Variance elimination → Smaller sample size $\widehat{\mathcal{N}}^{(l)}(u)$.**
- VR-GCN:

$$\boldsymbol{H}^{(l+1)} = \sigma\left(\widehat{\boldsymbol{P}}^{(l)}\left(\boldsymbol{H}^{(l+1)} - \bar{\boldsymbol{H}}^{(l)}\right) + \boldsymbol{P}\bar{\boldsymbol{H}}^{(l)}\right)\boldsymbol{W}^{(l)}$$

The sampled normalized adjacency matrix at layer l.

Historical hidden embedding

- **One more thing:** CVD Sampler -- Control Variate for Dropout.

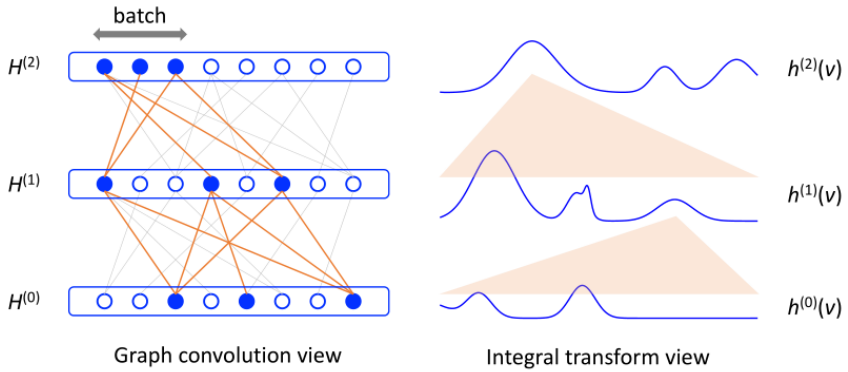
Chen, Jianfei, Jun Zhu, and Le Song. "Stochastic Training of Graph Convolutional Networks with Variance Reduction."

Node-wise Sampling: VR-GCN

- Pros:
  - Analyze the variance reduction on node-wise sampling.
  - Successfully reducing the size of samples.
- Cons:
  - Additional memory consuming for storing the historical hidden embeddings.

Chen, Jianfei, Jun Zhu, and Le Song. "Stochastic Training of Graph Convolutional Networks with Variance Reduction."

Layer-wise Sampling: FastGCN



The functional generalization of GCN

$$L = E_{v\sim P}\left[g\left(h^{(M)}(v)\right)\right] = \int g\left(h^{(M)}(v)\right) dP(v)$$

Integral Transform

The estimation

$$L_{t_0,t_1,\ldots,t_M} := \frac{1}{t_M}\sum_{i=1}^{t_M} g(h_{t_M}^{(M)}(u_i^{(M)}))$$

The i.i.d. node sample at each layer → bootstrapping

FastGCN: sampling fixed number of nodes at each layer.

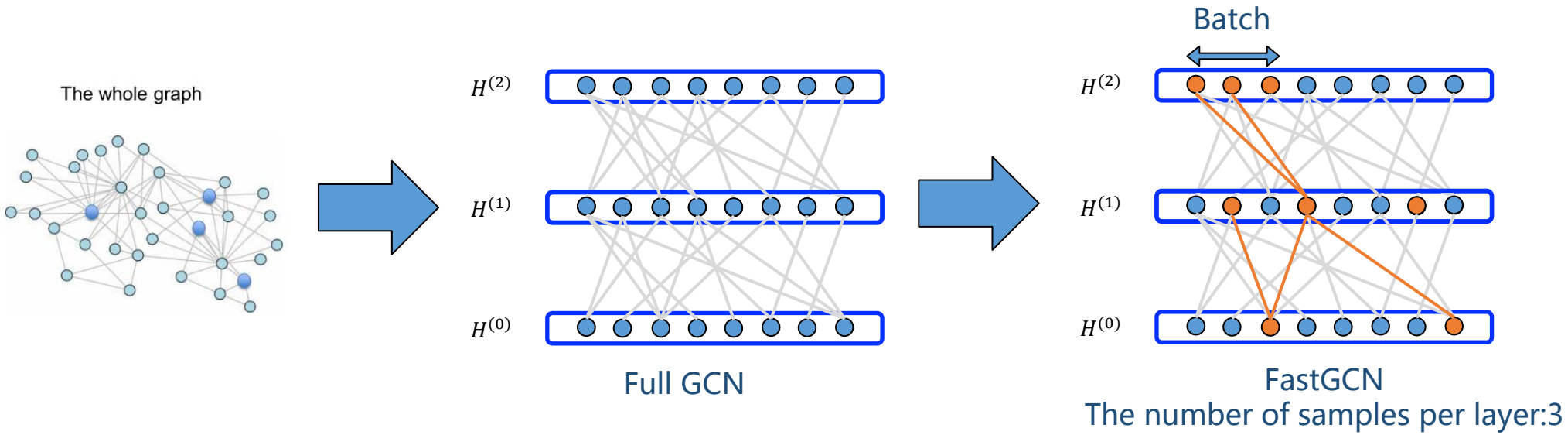
Chen, Jie, Tengfei Ma, and Cao Xiao. "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling."

Layer-wise Sampling: FastGCN

- Towards the variance reduction: importance sampling.

$$q(u) = \|\hat{A}(:,u)\|^2 / \sum_{u' \in V} \|\hat{A}(:,u')\|^2, \quad u \in V$$

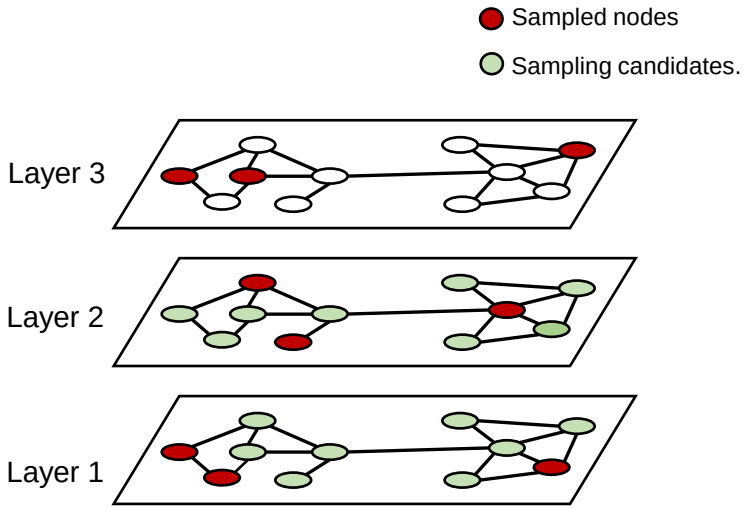
This sampling probability keeps the same for each layer.



Chen, Jie, Tengfei Ma, and Cao Xiao. "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling."

Layer-wise Sampling: FastGCN

- Pros:
  - Avoid neighborhood expansion problem.
  - Sample method with quality guarantee.
- Cons:
  - Failed to capture the between-layer correlations.
  - Performance compromise.



FastGCN
The number of samples per layer:3

Chen, Jie, Tengfei Ma, and Cao Xiao. "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling."

Layer-wise Sampling: ASGCN

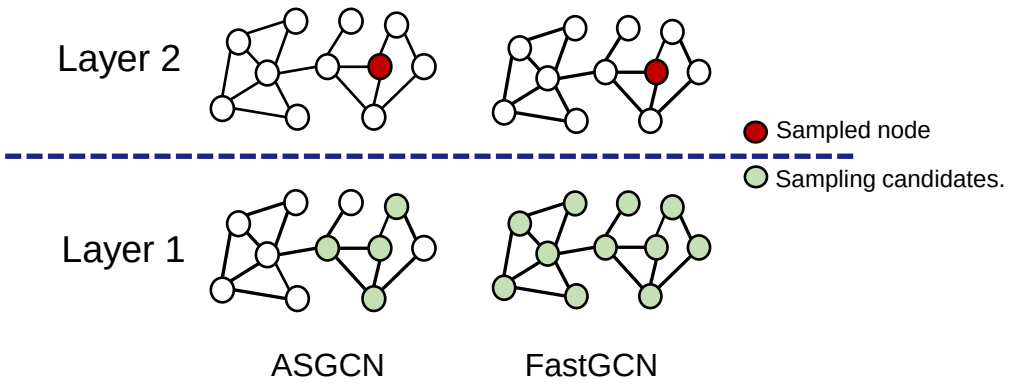
$x(u_j)$: the node feature of node u_j.
$p(u_j|v_i)$: the probability of sampling node u_j given node v_i.
$\hat{a}(v_i, u_j)$: The entry of node v_i and u_j in re-normalization of the adjacency matrix $\hat{A}$.
$\hat{\mu}_q(v_i)$: the output hidden embeddings of node v_i.

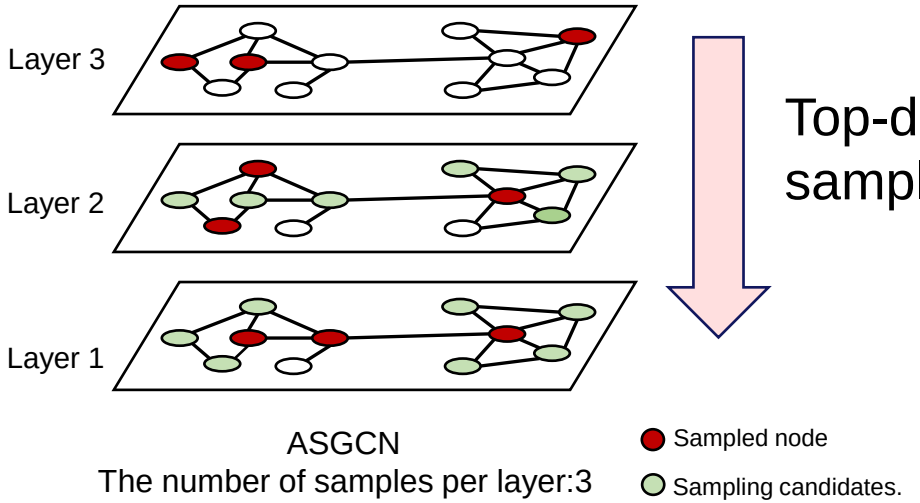
- Adaptive layer-wise sampling:
  - The sampling probability of lower layers depends on the upper ones.

$$q(u_j) = \frac{p(u_j|v_i)}{p(u_j|v_1 \dots v_n)} \quad s.t. \boxed{p(u_j|v_i)} = \frac{\hat{a}(v_i,u_j)}{N(v_i)}, N(v_i) = \sum_{j=1}^{n} \boxed{\hat{a}(v_i, u_j)}$$

the probability of sampling node u_j given node v_i.

The entry of node v_i and u_j in re-normalization of the adjacency matrix $\hat{A}$.





Huang, Wenbing, et al. "Adaptive sampling towards fast graph representation learning."

Layer-wise Sampling: ASGCN

- Parameterized for **explicit variance reduction**.

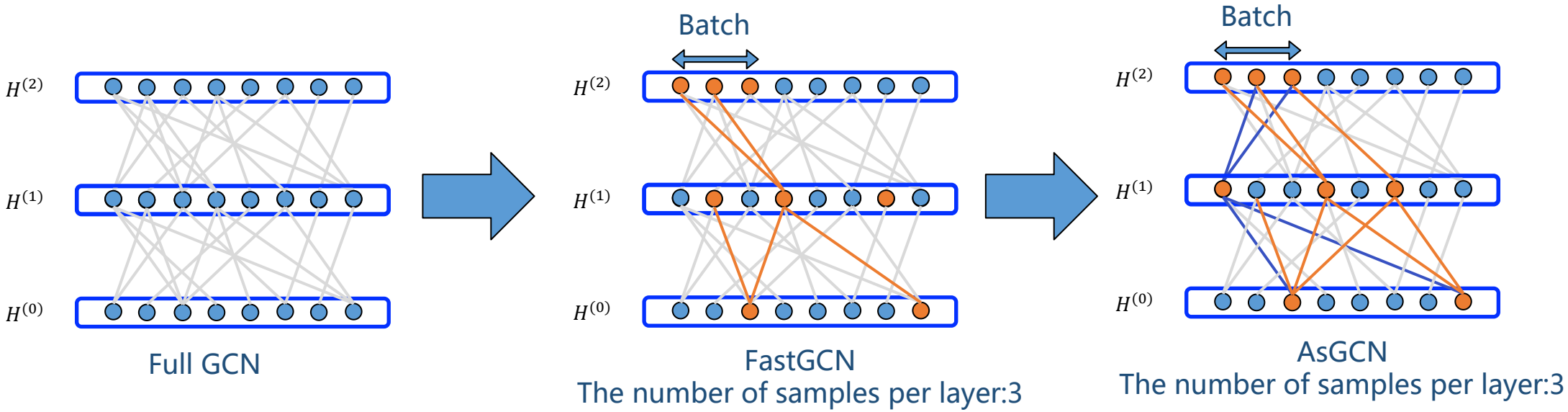
$$q^*(u_j) = \frac{\sum_{i=1}^{n} p(u_j|v_i)|g(x(u_j))|}{\sum_{j=1}^{N}\sum_{i=1}^{n} p(u_j|v_i)|g(x(v_j))|}, \quad g(x(u_j)) = W_g x(u_j)$$

$x(u_j)$: the node feature of node u_j.
$p(u_j|v_i)$: the probability of sampling node u_j given node v_i.
$\hat{a}(v_i, u_j)$: The entry of node v_i and u_j in re-normalization of the adjacency matrix $\hat{A}$.
$\hat{\mu}_q(v_i)$: the output hidden embeddings of node v_i.

- Optimize the sampler $q^*(u_j)$ to minimize the variance:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}_c(y_i, \bar{y}(\hat{\mu}_q(v_i))) + \lambda \mathrm{Var}_q(\hat{\mu}_q(v_i)),$$

Can be estimated by the sampled instances.



Huang, Wenbing, et al. "Adaptive sampling towards fast graph representation learning."

Layer-wise Sampling: ASGCN

- Self-dependent shared attention:

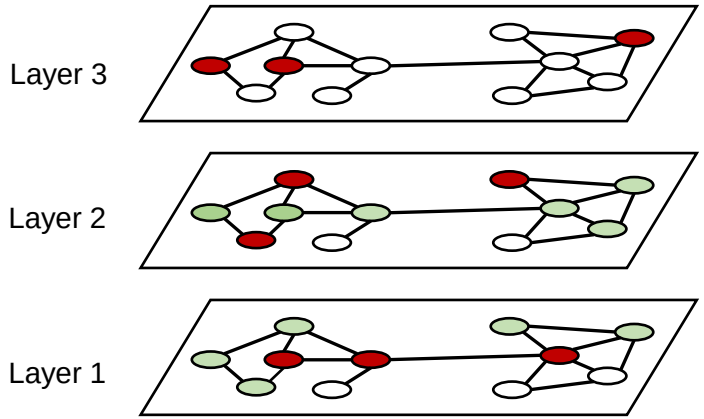
$$a(x(v_i), x(u_j)) = \frac{1}{n}\mathrm{ReLu}(W_1 g(x(v_i)) + W_2 g(x(u_j)))$$

- Preserving second-order proximities by skip connections:

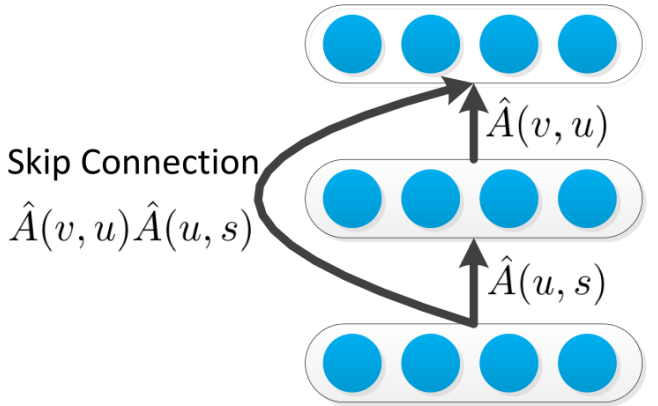
$$h_{skip}^{(l+1)}(v_i) = \sum_{j=1}^{n} \hat{a}_{skip}(v_i, s_j) h^{(l-1)}(s_j) W_{skip}^{(l-1)}, \quad i = 1, \cdots, n,$$

  - where, $\hat{a}_{skip}$ is estimated by:

$$\hat{a}_{skip}(v_i, s_j) \approx \sum_{k=1}^{n} \hat{a}(v_i, u_k)\hat{a}(u_k, s_j).$$



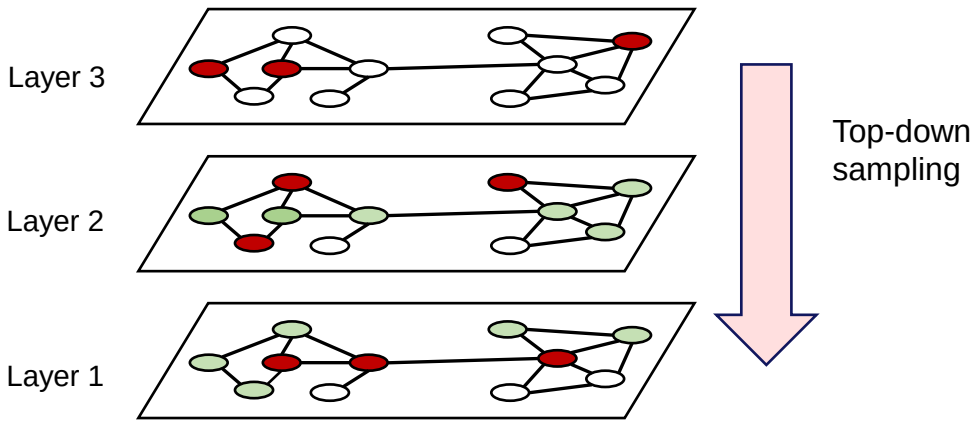
ASGCN



Huang, Wenbing, et al. "Adaptive sampling towards fast graph representation learning."

Layer-wise Sampling: ASGCN

- Pros:
  - Good performance.
  - Better variance control.
- Cons:
  - Additional dependence during sampling.



ASGCN

Github: https://github.com/huangwb/AS-GCN

Huang, Wenbing, et al. "Adaptive sampling towards fast graph representation learning."

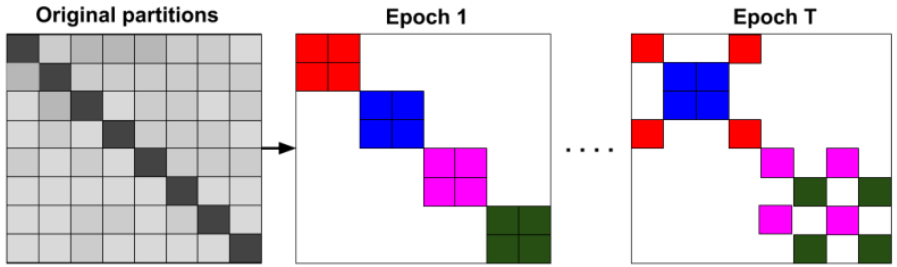
Graph-wise Sampling: ClusterGCN

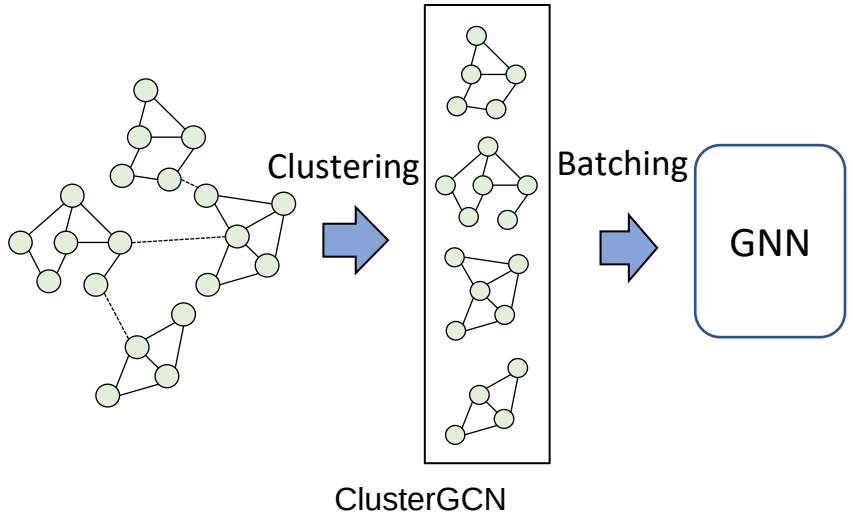
- Extract small clusters based efficient clustering algorithms.

$$\bar{G} = [G_1, \cdots, G_c] = [\{\mathcal{V}_1, \mathcal{E}_1\}, \cdots, \{\mathcal{V}_c, \mathcal{E}_c\}],$$

$$\bar{A} = \begin{bmatrix} A_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_{cc} \end{bmatrix}, \Delta = \begin{bmatrix} 0 & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & 0 \end{bmatrix},$$

- Random batching at the subgraph level.

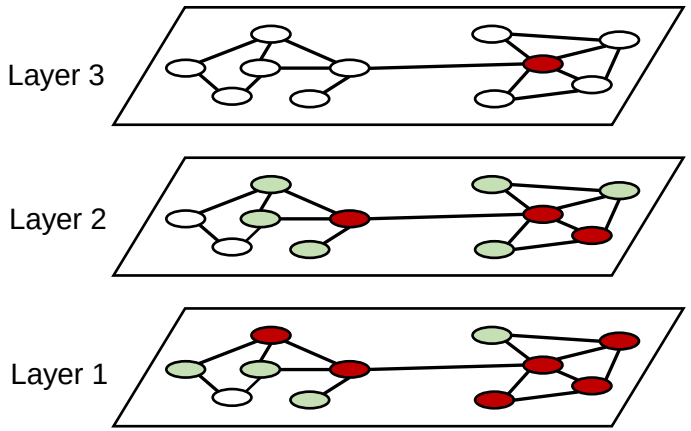




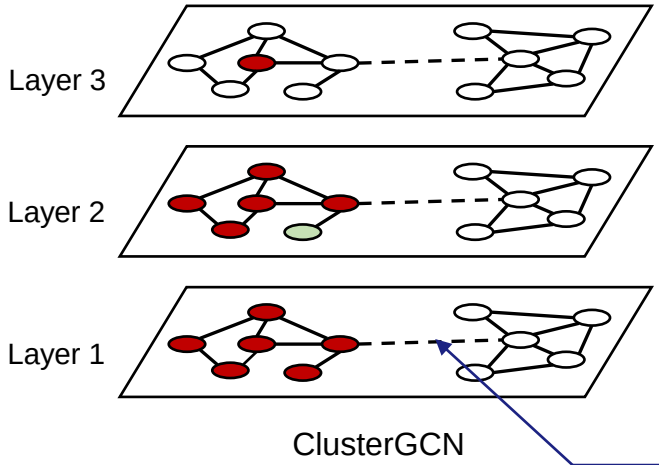
Chiang, Wei-Lin, et al. "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks."

Graph-wise Sampling: ClusterGCN

- Neighbor expansion control.



Fix-size neighbor sampling S=2



ClusterGCN

Only sample the nodes in the clusters

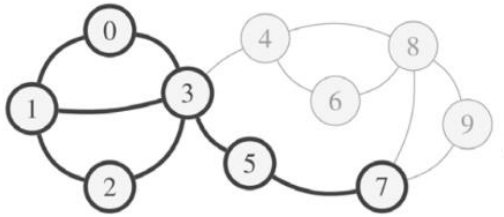
Chiang, Wei-Lin, et al. "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks."

Graph-wise Sampling: ClusterGCN

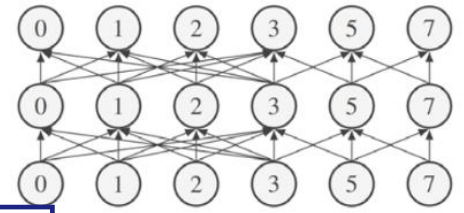
- Pros:
  - Good performance / Good memory usage.
  - Alleviate the neighborhood expansion problem in traditional mini-batch training.
- Cons:
  - Empirical results without analyzing the sampling quality.

Chiang, Wei-Lin, et al. "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks."

Graph-wise Sampling: GraphSAINT



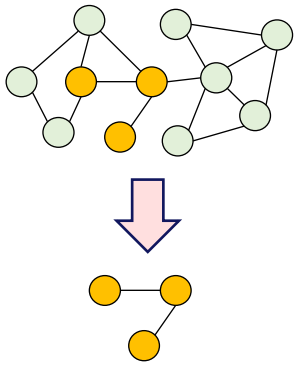
$\mathcal{G}_s = \text{SAMPLE}(\mathcal{G})$



Full GCN on $\mathcal{G}_s$

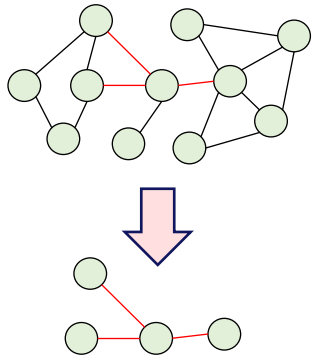
- Directly sample a subgraph for mini-batch training according to subgraph sampler.
- Sampler construction

Node sampler



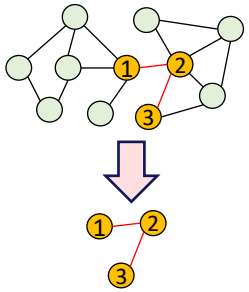
Uniformly sample nodes.

Edge sampler



Sample edge with probably $p_{u,v} \propto 1/d_u + 1/d_v$

Random walk sampler



Sample edge with probably $p_{u,v} \propto \boldsymbol{B}_{u,v} + \boldsymbol{B}_{v,u}$

- $B_{u,v}$: the probability of a random walk to start at u and end at v in L hops.

Zeng, Hanqing, et al. "GraphSAINT: Graph Sampling Based Inductive Learning Method."

Graph-wise Sampling: GraphSAINT

- How to eliminate the bias introduce by the sampler?
  - Loss normalization:

$$\mathcal{L}_{\text{batch}} = \sum_{v \in G_s} L_v / \lambda_v \, , \lambda_v = |V| p_v .$$

  - Aggregation normalization:

$$a(u, v) = p_{u,v} / p_v$$

p_v: the probability of a node $v \in V$ being sampled.
$p_{u,v}$: the probability of an edge $(u, v) \in E$ being sampled.

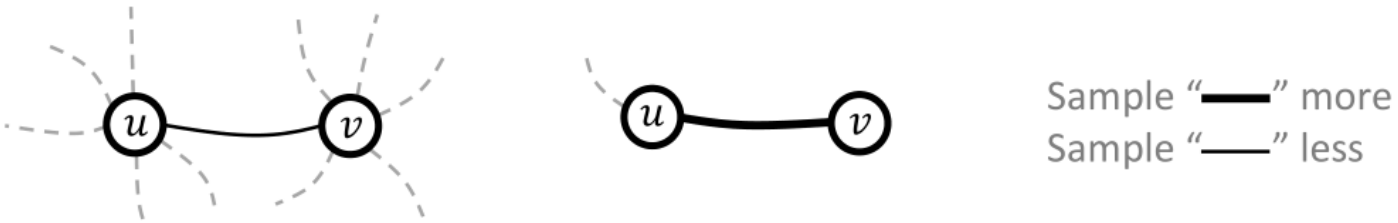
Zeng, Hanqing, et al. "GraphSAINT: Graph Sampling Based Inductive Learning Method."

Graph-wise Sampling: GraphSAINT

p_v: the probability of a node $v \in V$ being sampled.
$p_{u,v}$: the probability of a edge $(u, v) \in E$ being sampled.

- How to reduce the sampling variance?
  - The optimal edge probability for variance minimization:

$$p_{u,v} \propto 1/d_u + 1/d_v$$



Zeng, Hanqing, et al. "GraphSAINT: Graph Sampling Based Inductive Learning Method."

Graph-wise Sampling: GraphSAINT

Dataset	Nodes	Edges	Degree	Feature	Classes	Train / Val / Test
PPI	14,755	225,270	15	50	121 (m)	0.66 / 0.12 / 0.22
Flickr	89,250	899,756	10	500	7 (s)	0.50 / 0.25 / 0.25
Reddit	232,965	11,606,919	50	602	41 (s)	0.66 / 0.10 / 0.24
Yelp	716,847	6,977,410	10	300	100 (m)	0.75 / 0.10 / 0.15
Amazon	1,598,960	132,169,734	83	200	107 (m)	0.85 / 0.05 / 0.10

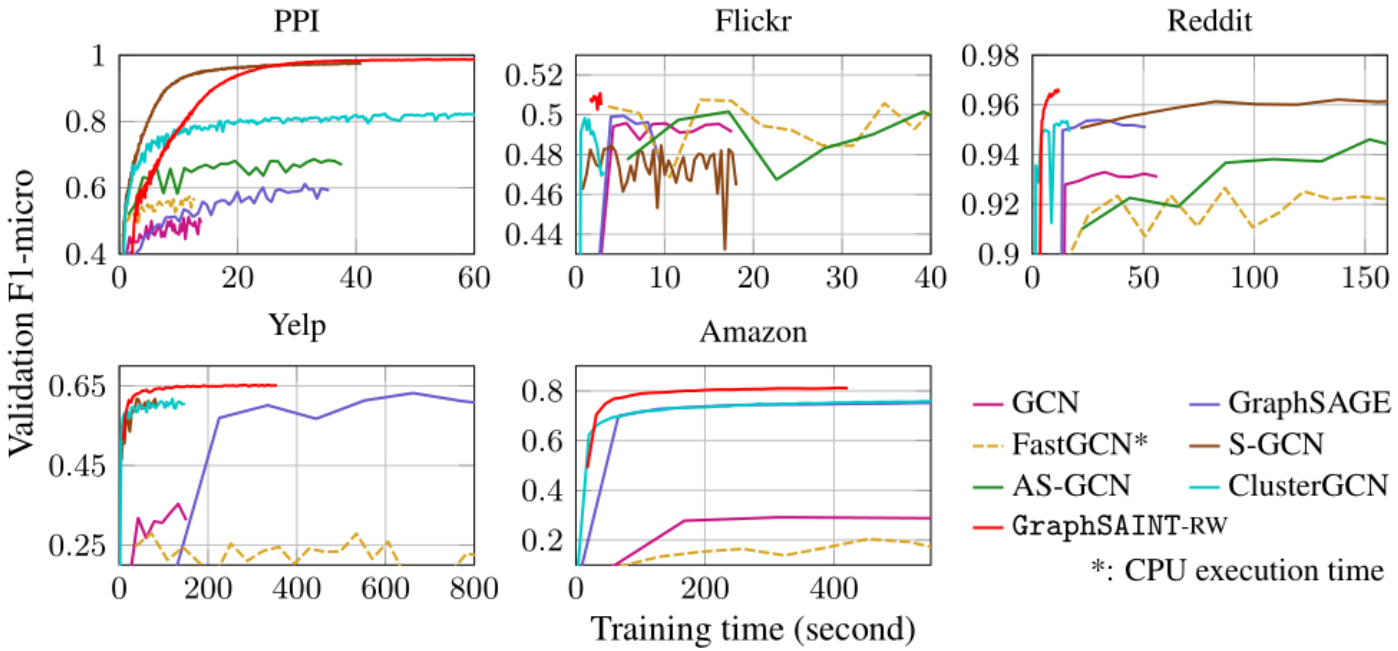


Figure 2: Convergence curves of 2-layer models on GraphSAINT and baselines

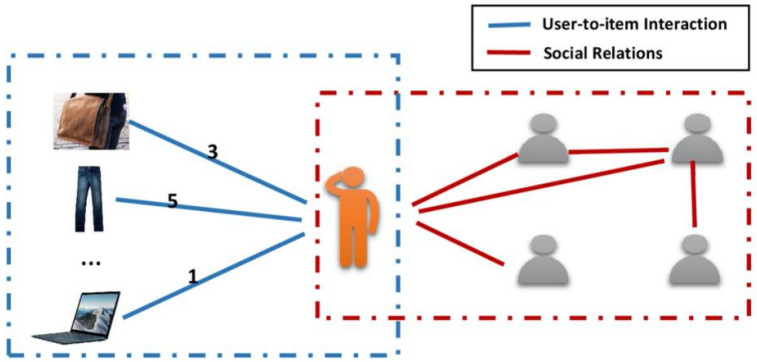
- Highly flexible and extensible (graph samplers, GNN architectures, etc.)
- Good performance (accuracy, speed)

Zeng, Hanqing, et al. "GraphSAINT: Graph Sampling Based Inductive Learning Method."

Large-Scale GNN in Real-World Applications

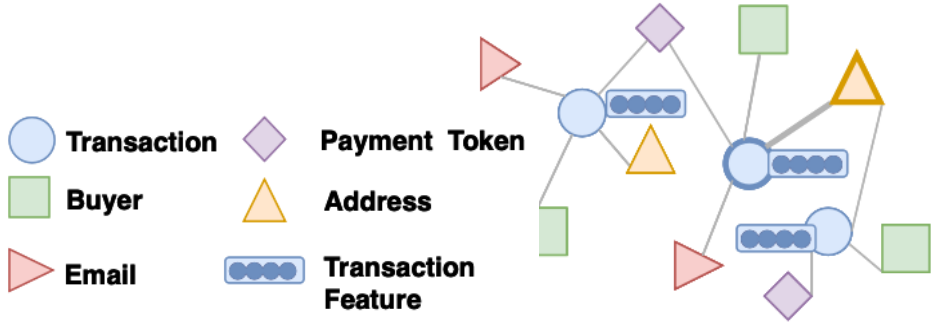
- Large-scale GNN has been successfully applied to real-world applications.

Social Recommendation



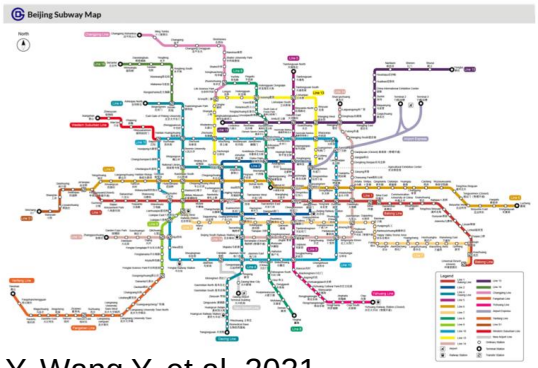
Fan W, Ma Y, Li Q, et al.

Fraud Transaction Detection



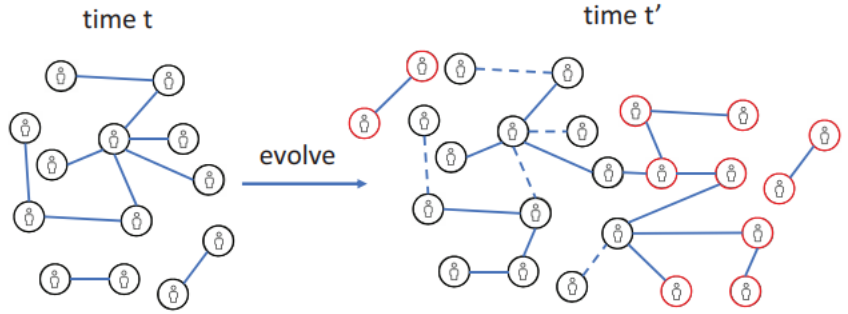
Rao S X, Zhang S, Han Z, et al. . 2020

Traffic Forecasting



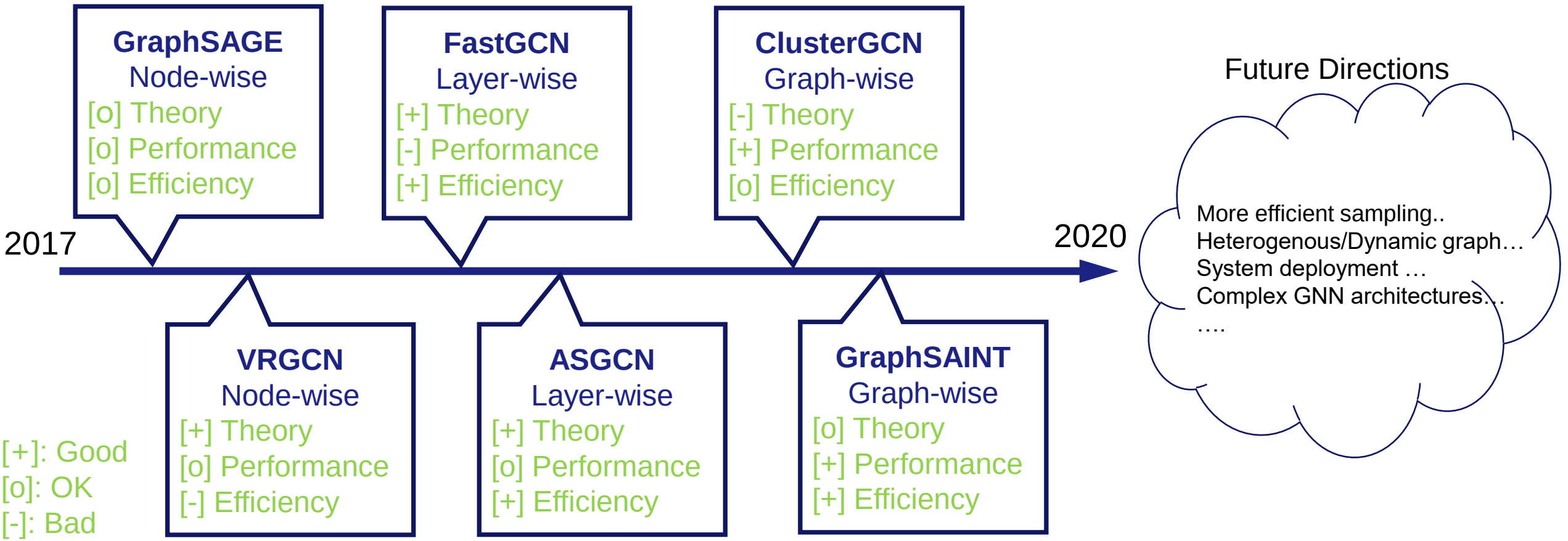
Wang X, Ma Y, Wang Y, et al. 2021

Bitcoin Transaction Forecasting



Wei W, Zhang Q, Liu L. 2020

Summary



Acknowledgement: The idea of this slide is taken from: https://drive.google.com/file/d/1Va7WaXOcNS37z779WLbZ-H54mXEJy8cB/view?usp=sharing

Robustness of GNNs

Adversarial Attacks on Deep Learning

Classified as panda

x

Small adversarial noise

ϵ

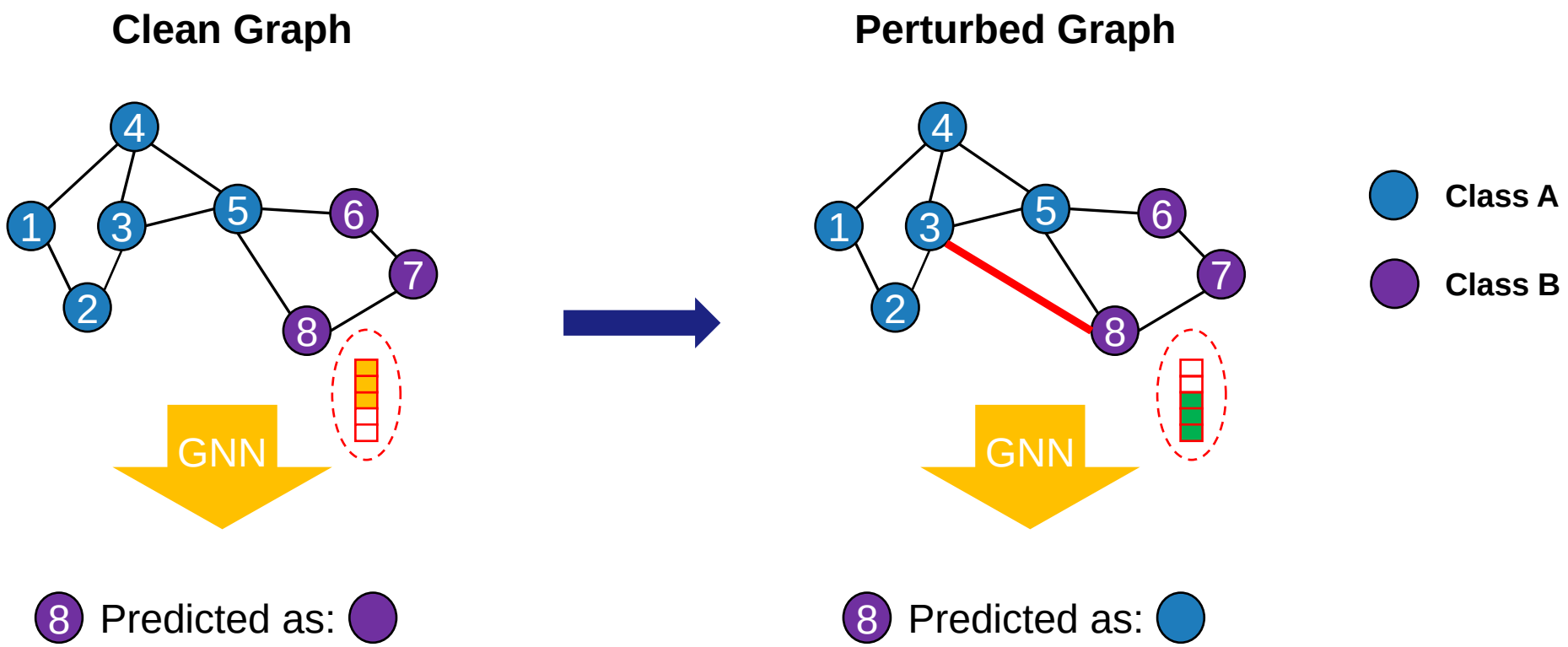
Classified as gibbon

x'

Find x' satisfying $\left||x' - x|\right| \leq \Delta$ s.t. $C(x') \neq y$

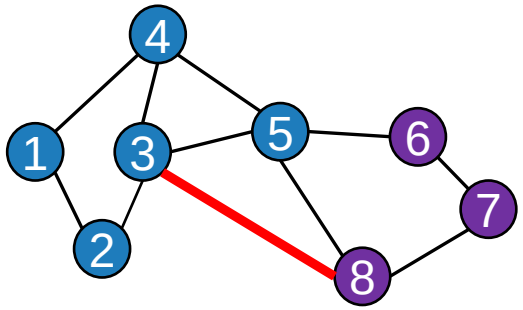
Adversarial Attacks on GNNs

- Adversarial example on GNNs aims to change the prediction of GNNs by modifying the edges and features.

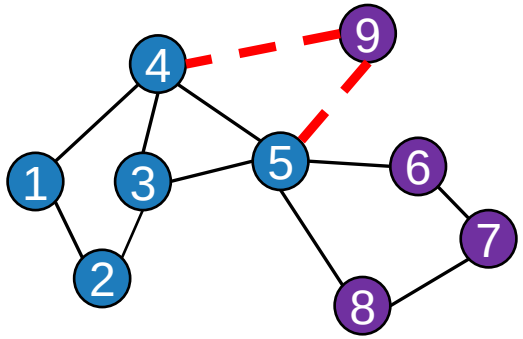


Types of Perturbations

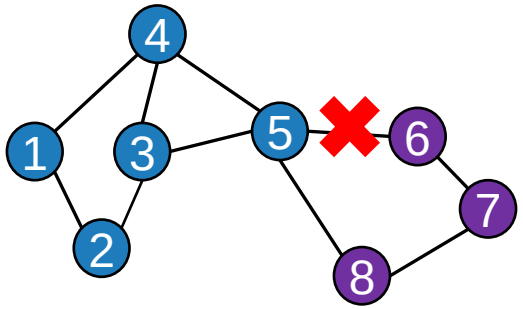
Edge Insertion



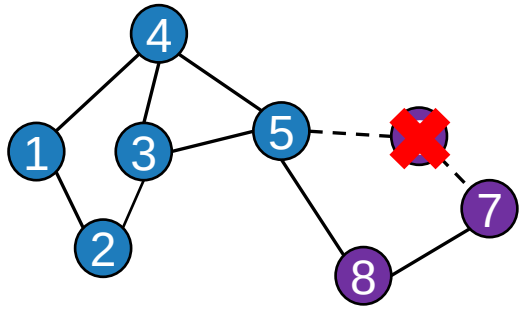
Node Injection



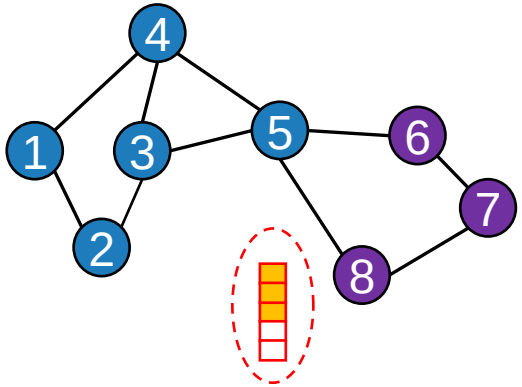
Edge Deletion



Node Deletion

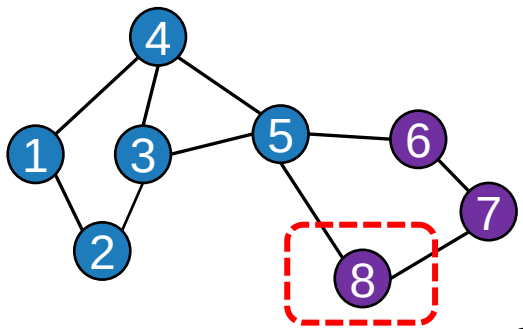


Feature Modification



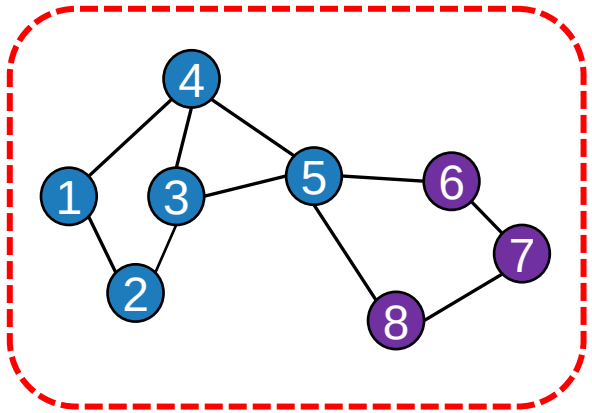
Types of Attacking Type

Targeted Attack



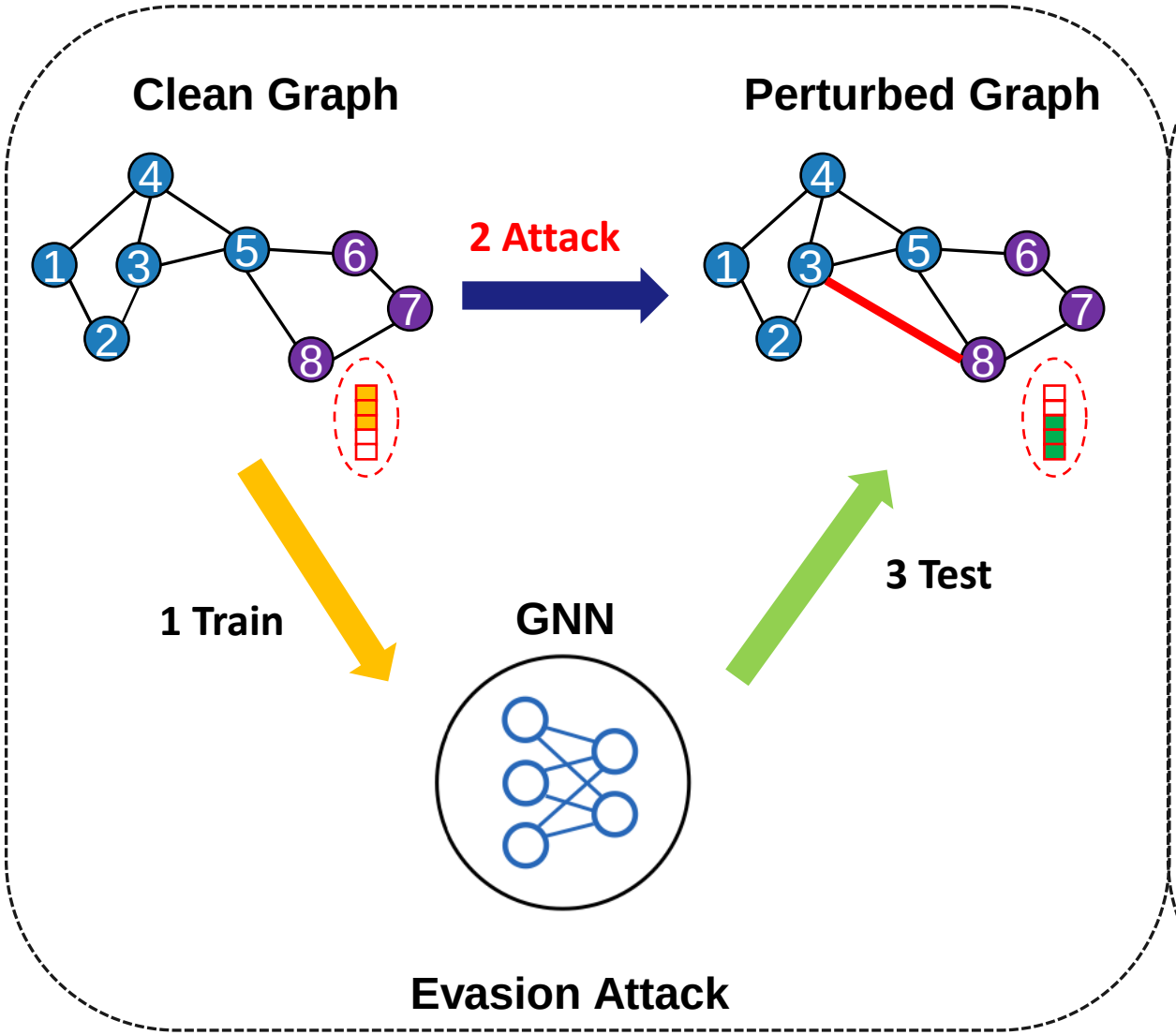
8 **Target Node**

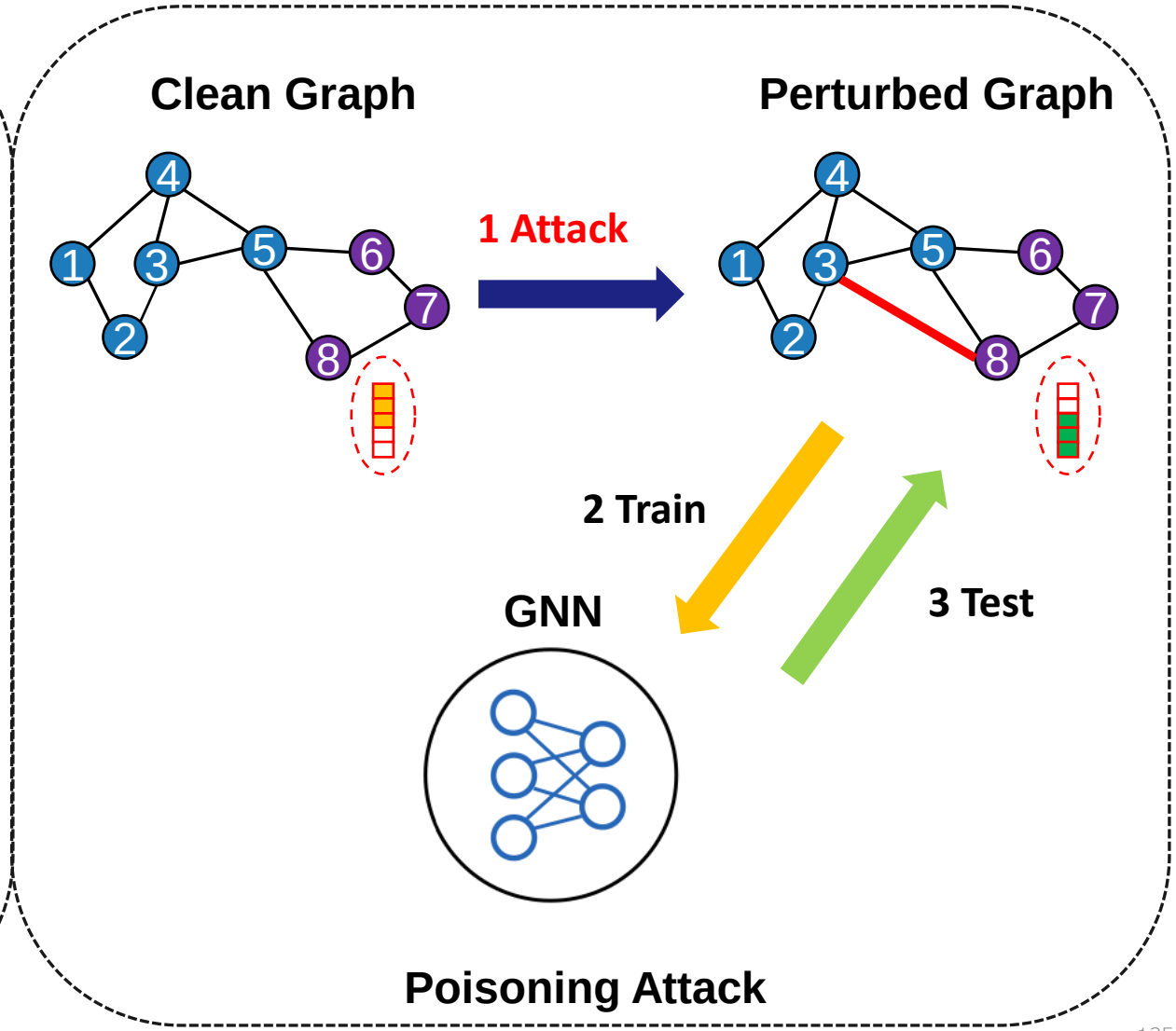
Non-Targeted Attack



All nodes as candidates

Evasion & Poisoning Attack





Attacker's Knowledge

- Attacker's knowledge means how much information an attacker knows about the model that he aims to attack.

Settings	Model Parameters	Model Predictions	Labels	Training Input
White-**B**ox **A**ttack (**WBA**)	✓	✓	✓	✓
Gray-**B**ox **A**ttack (**GBA**)		✓	✓	✓
Restrict **B**lack-box **A**ttack (**RBA**)				✓

Attack Methods

Attack Methods	Node Injection	Edge Insertion/ Deletion	Feature Modification	Targeted	Non-Targeted	Evasion	Poisoning	Black-box	RL based
Nettack (KDD 18)		✓	✓	✓		✓	✓		
RL-S2V (ICML 18)		✓		✓		✓		✓	✓
$\mathcal{A}_{class}$ (ICML 19)		✓		✓	✓		✓	✓	
Metattack (ICLR 19)		✓			✓		✓		
GF-Attack (AAAI 20)		✓		✓		✓		✓	
CD-Attack (WWW 20)		✓		✓			✓	✓	
NIPA (WWW 20)	✓				✓		✓		✓

Attack Methods

Attack Methods	Node Injection	Edge Insertion/ Deletion	Feature Modification	Targeted	Non-Targeted	Evasion	Poisoning	Black-box	RL based
Nettack (KDD 18)		✓	✓	✓		✓	✓		
RL-S2V (ICML 18)		✓		✓		✓		✓	✓
$\mathcal{A}_{class}$ (ICML 19)		✓		✓	✓		✓	✓	
Metattack (ICLR 19)		✓			✓		✓		
GF-Attack (AAAI 20)		✓		✓		✓		✓	
CD-Attack (WWW 20)		✓		✓			✓	✓	
NIPA (WWW 20)	✓				✓		✓		✓

Nettack (KDD 18)

- General form of graph adversarial attack as a bi-level optimization problem (poisoning setting):

$$\arg\max_{\hat{A},\hat{X}} \mathcal{L}_{atk}\left(f_{\theta^*}(A', X')\right) = \sum_{u \in V_t} \ell\left(f_{\theta^*}(A', X')_u, c_{old,u}\right)$$

Prediction of attacked model

Prediction of clean model

$$s.t. \quad \theta^* = arg\min_{\theta} \mathcal{L}_{train}(f_\theta(A', X')) \ , |A' - A| + |X' - X| \leq \Delta$$

- **Types of attack**:
  - Targeted attack, Poison/Evasion Attack, White-Box Attack
  - Structure and feature modifications

A: adjacent matrix
X: node features matrix
A': modified structure
X': modified feature
V_t: set of target nodes
$c_{old,u}$: the predicted class label of the clean model.
Δ: perturbation budget

- **Core idea**: Establishing a linear surrogate model:

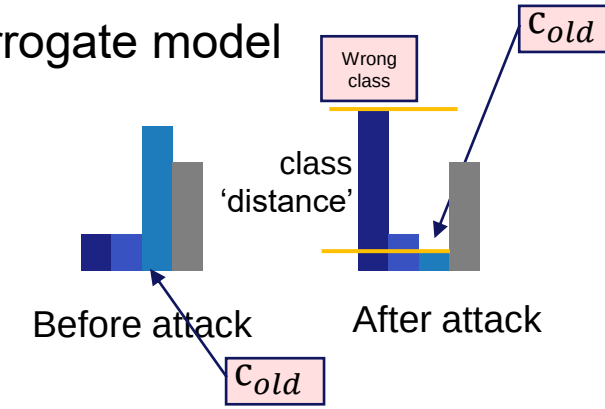
$$f_{S,\theta}(A, X) = \text{softmax}\left(\hat{A}\, ReLU\left(\hat{A}X\Theta^{(1)}\right)\Theta^{(2)}\right) = \text{softmax}(\hat{A}^2 X\Theta)$$

$\Theta^{(1)}\Theta^{(2)}$, trained on the clean data.

Nettack (KDD 18)

Optimization approach: one perturbation that maximizes the class 'distance' of the surrogate model $f_{S,\theta}(A, X)$ before and after attack.

$$\ell\left(f_{S,\theta^*}(A,X)\,, c_{old,u}\right) = \max_{c \neq c_{old}} Z(u)_c - Z(u)_{c_{old}} \quad Z = f_{S,\theta^*}(A,X)$$

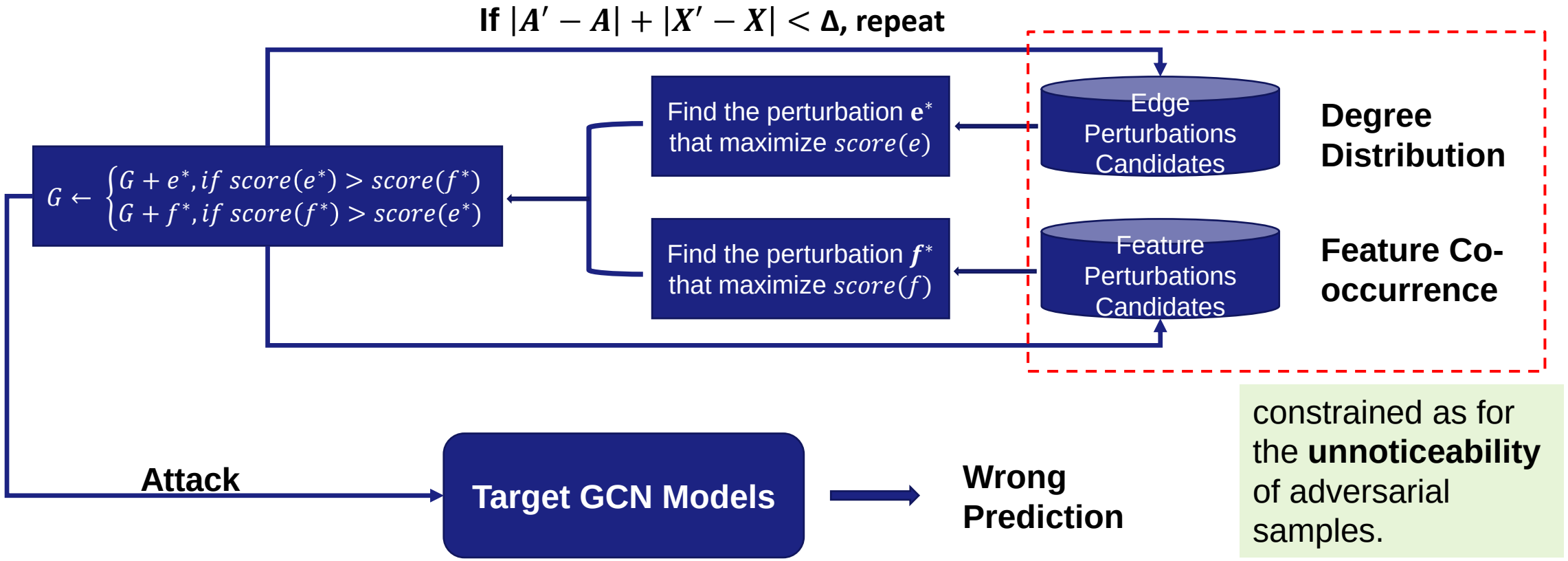


Attack on edge: $score(e) = \ell\left(f_{S,\theta^*}(A', X)\,, c_{old,u}\right)$, $A' := A \pm e$

Attack on feature: $score(f) = \ell\left(f_{S,\theta^*}(A, X')\,, c_{old,u}\right)$, $X' := X \pm f$

Nettack (KDD 18)

Optimization approach: one perturbation that maximizes the class 'distance' of the surrogate model $f_{S,\theta}(A, X)$ before and after attack.



Overview

[KDD18] Adversarial Attacks on Neural Networks for Graph Data

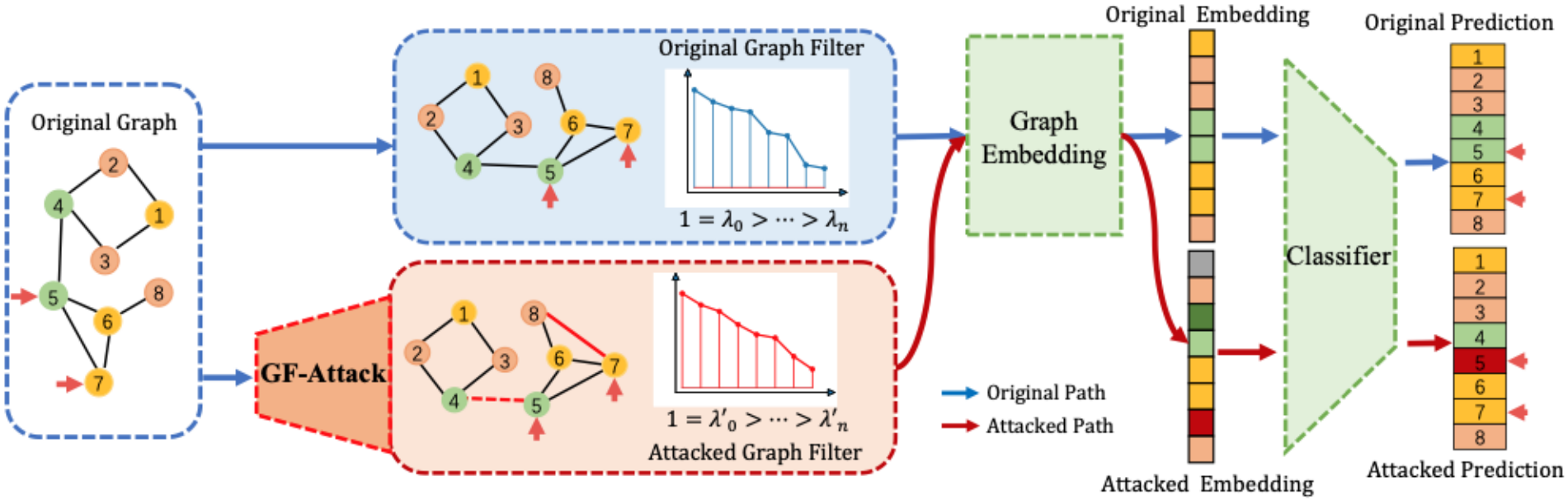
GF-Attack (AAAI 20)

- How to perform the attack under RBA setting?
- View of graph signal processing → attacking on the graph filter(***GF-Attack***).

Graph filtering: $\tilde{X} = \mathscr{H}(X)$

Feature transformation: $X' = \sigma(\tilde{X}\Theta)$

- **Types of attack**:
  - Target attack, Restricted Black-Box Attack
  - Structure modifications



$\mathcal{H}$: graph filter
S: graph-shift filter
X: node features matrix
θ: weights
$\tilde{X}$: transformed features matrix

GF-Attack (AAAI 20)

- **Core idea**: attacking the graph filter $\mathcal{H}_{\text{model}} = h(S)$ for different graph embedding models.
- Measuring the quality of output embedding Z as the T-rank approximation problem:

$$\mathscr{L}(A', Z) = \|h(S')X - h(S')_T X\|_F^2,$$

$\mathcal{H}$: graph filter
S: graph-shift filter
$h(.)$: a polynomial function.
X: node features matrix
θ: weights
$\tilde{X}$: transformed features matrix
λ_i and $\boldsymbol{u}_i$ are an eigen-pair of graph filter $\mathcal{H}$.

- By solving this problem, this is equivalent to optimize:

$$\arg\max_{A'} \sum_{i=T+1}^{n} \lambda_i'^2 \cdot \sum_{i=T+1}^{n} \|\mathbf{u}_i^T X\|_2^2,$$
$$\text{s.t. } \|A' - A\| = 2\beta.$$

Egin-pair of graph filter h(S)

If we know the graph filter of target model, we can perform the adversarial attack without any additional information(model parameters, predictions, labels)!

GF-Attack (AAAI 20)

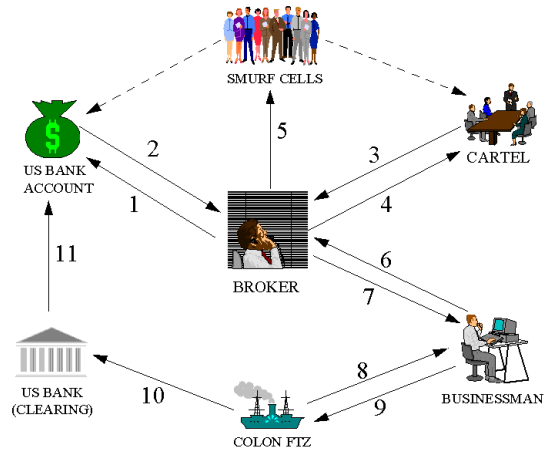
Graph Embedding Models	Graph-shift filter S	Polynomial Function $h(x)$
GCN	$L^{sym} - I_n$	$h(x) = x$
SGC	$L^{sym} - I_n$	$h(x) = x$
ChebyNet	$L^{sym} - I_n$	$h(x) = \sum_{k=0}^{K} T_k(x)$
LINE	$I_n - L^{rw}$	$h(x) = x$
DeepWalk	$I_n - L^{rw}$	$h(x) = \sum_{k=0}^{K} x^k$

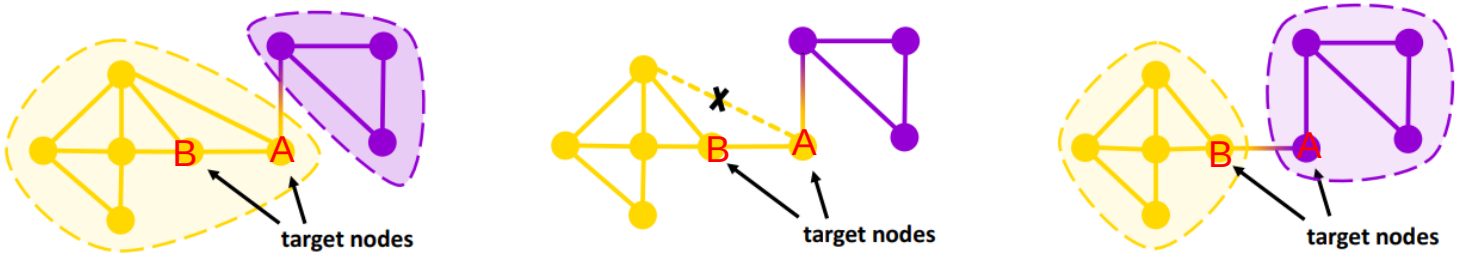
S: graph-shift filter;
$\mathrm{h}(\mathrm{x})$: polynomial function used for constructing graph filter $\mathcal{H}$.

[AAAI20] A Restricted Black-box Adversarial Framework Towards Attacking Graph Embedding Models

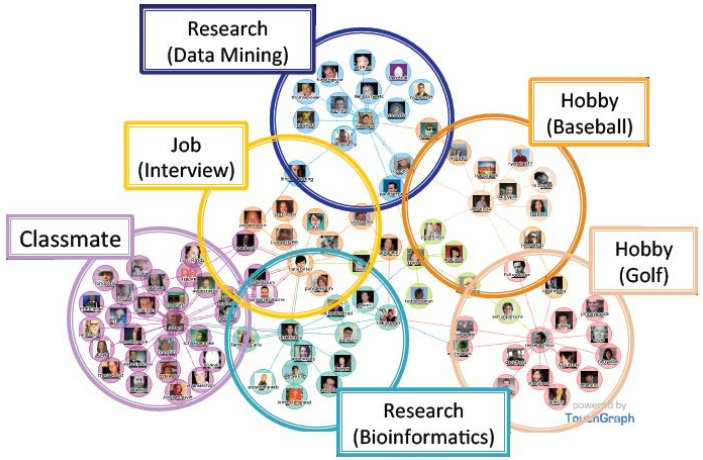
CD-Attack (WWW 20)

- Go beyond node classification → hiding individuals from **community detection**.
- **Broad applications**:
  - Money laundering in transaction networks.
  - Personal privacy protection in social networks.
- **Types of attack**:
  - Target attack, Restricted Black-box Attack
  - Structure modifications





(a) clean graph (b) small perturbation (c) attacked graph

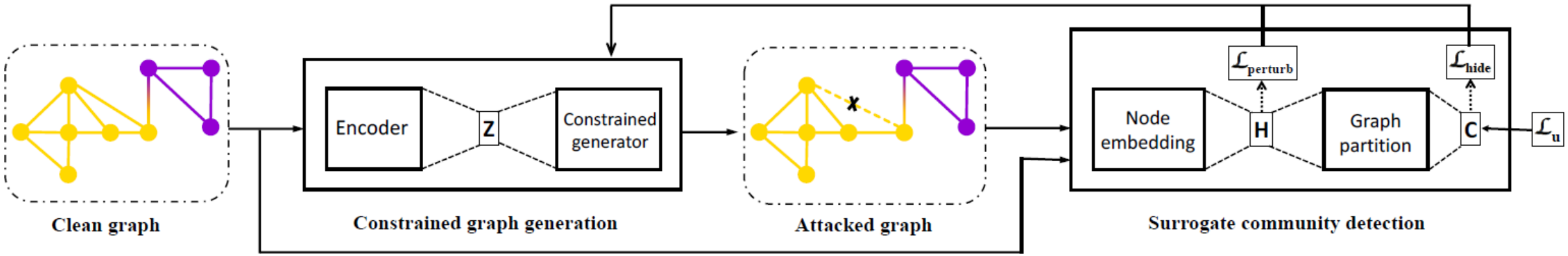


Target: hiding individuals from community detection by small modification on the graph structure.

[WWW20] Adversarial Attack on Community Detection by Hiding Individuals

CD-Attack (WWW 20)

- **Challenges**:
  - Huge space to locate the perturbation positions
  - No knowledge about the targeted community detection methods (RBA)
- **Core idea**: Modeling the interaction between community detection and perturbed graph:
  - **Constrained graph generation**: VGAE with efficient sampling.
  - **Surrogate community detection**: GNN (efficiency) and normalized cut (generalization)
  - **Interaction between two modules:** Actor-Critic

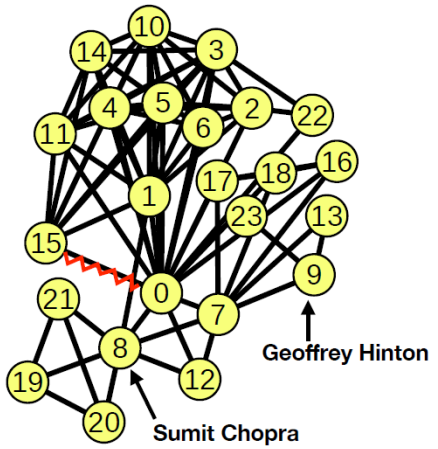


Experimental Results.

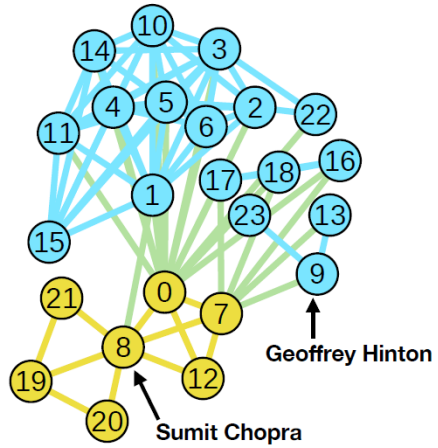
Data sets	DBLP-medium		Finance-medium		DBLP-large		Finance-large	
-	M1	M2	M1	M2	M1	M2	M1	M2
DICE	6.35%	41.44%	3.25%	53.55%	6.91%	50.77%	3.66%	72.11%
MBA	7.33%	44.88%	2.80.%	52.00%	7.04%	53.67%	3.08%	72.11%
RTA	6.55%	42.22%	2.53%	42.66%	6.25%	44.66%	1.74%	55.11%
CD-ATTACK	**13.72%**	**52.00%**	**3.32%**	**63.00%**	**8.11%**	**57.62%**	**4.14%**	**87.33%**

The hiding quality (Higher the better)

Case Study: Attack node2vec (Grover and Leskovec, 2016) + K-means on DBLP.



(a) clean graph



(b) modified graph

0: Yann LeCun
1: Raia T. Hadsell
2: Koray Kavukcuoglu
3: Pierre Sermanet
4: Ayse Erkan
5: Jan Ben
6: Urs Muller
7: M. Ranzato
8: Sumit Chopra
9: Geoffrey Hinton
10: Marco Scoffier
11: Matt Grimes
12: Fu Jie Huang
13: Volodymyr Mnih
14: Urs Miller
15: Chris Crudele
16: J. M. House
17: R. C. Flagan
18: P. E. Taylor
19: Andrew Caplin
20: John Leahy
21: Trivikraman Thampy
22: Soumith Chintala
23: Andrew Denis Brown

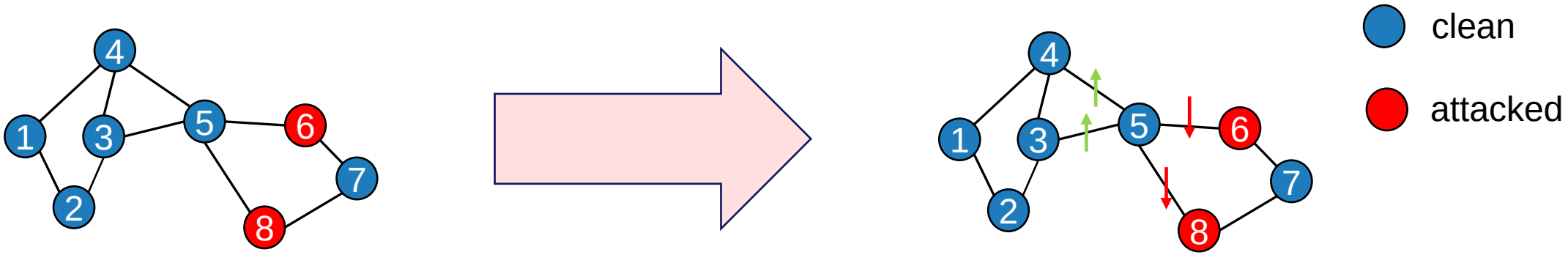
Defense against Adversarial Attacks

Defense strategies on graph data can be categorized as:

- **Attention mechanism**
- **Graph purification (preprocessing)**

Attention mechanism (RobustGCN)

- Using the attention mechanism to reduce the influence of adversarial attack.



- **RobustGCN** assumes that the adversarial sample nodes may have **high prediction uncertainty**.
- **Core idea:** Giving lower attention scores to adversarial edges to reduce their impact.

[KDD19] Robust Graph Convolutional Networks Against Adversarial Attacks

Attention mechanism (RobustGCN)

- Prediction uncertainty is captured by **Gaussian-based Graph Convolution Layer** (GGCL).

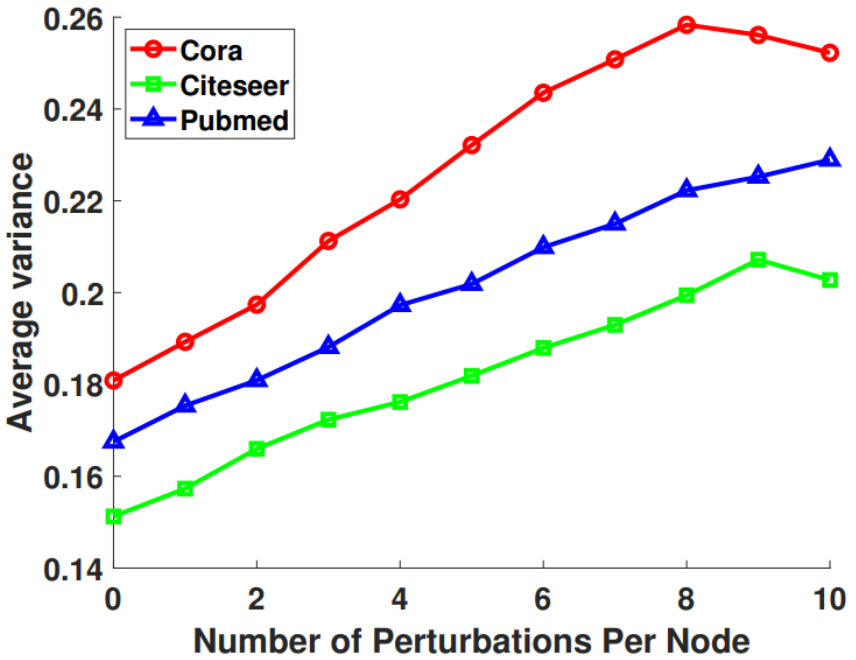
- **Approach**:
  - Using Gaussian distribution to capture the uncertainty in GGCL:

$$\boldsymbol{h}_{N(i)}^{(l)} = \sum_{j \in N(i)} \frac{1}{\sqrt{\widetilde{\boldsymbol{D}}_{ii}\widetilde{\boldsymbol{D}}_{jj}}} \boldsymbol{h}_{\mathrm{j}}^{(l)} \sim N\left(\sum_{j \in N(i)} \frac{1}{\sqrt{\widetilde{\boldsymbol{D}}_{ii}\widetilde{\boldsymbol{D}}_{jj}}} \boldsymbol{\mu}_{\mathrm{j}}^{(l)}, diag\left(\sum_{j \in N(i)} \frac{1}{\widetilde{\boldsymbol{D}}_{ii}\widetilde{\boldsymbol{D}}_{jj}} \boldsymbol{\sigma}_{\mathrm{j}}^{(l)} \right) \right)$$

  - Applying the attention mechanism to assign the nodes with **high variance** with lower weights when aggregation:

$$\boldsymbol{h}_{N(i)}^{(l)} = \sum_{j \in N(i)} \frac{1}{\sqrt{\widetilde{\boldsymbol{D}}_{ii}\widetilde{\boldsymbol{D}}_{jj}}} (\boldsymbol{h}_{\mathrm{j}}^{(l)} \odot \boldsymbol{\alpha}_j^l)$$

$$\boldsymbol{\alpha}_j^{(l)} = \exp(-\gamma \boldsymbol{\sigma}_j^{(l)})$$

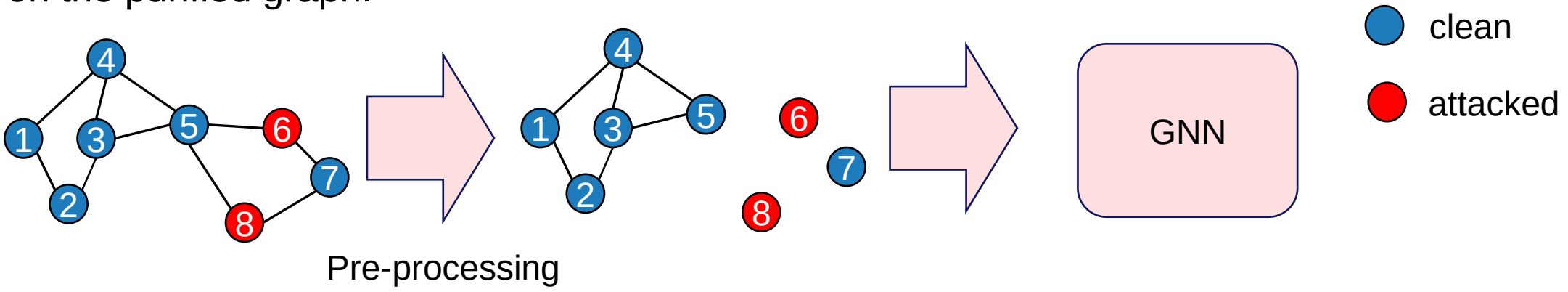


high variance → more perturbation

[KDD19] Robust Graph Convolutional Networks Against Adversarial Attacks

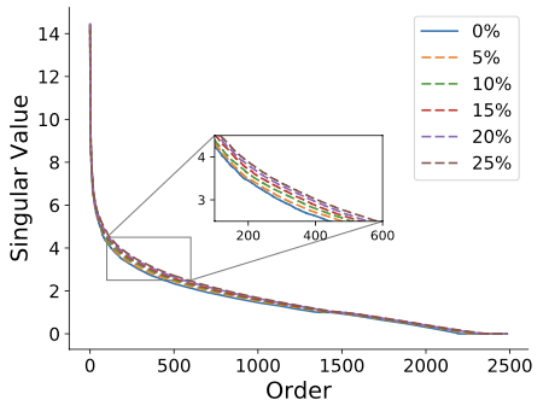
Graph purification (Pro-GNN)

- The purification method firstly purifies the disturbed graph structure data, and then trains the GNN model on the purified graph.

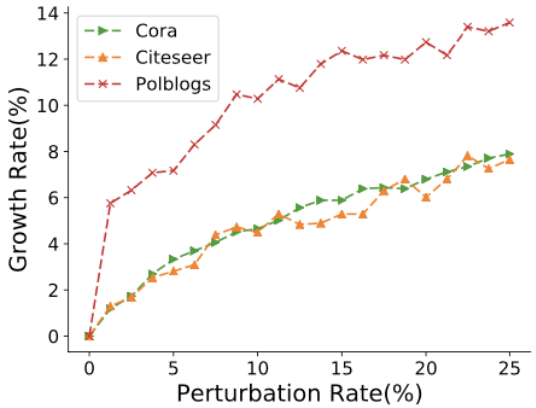


Graph purification (Pro-GNN)

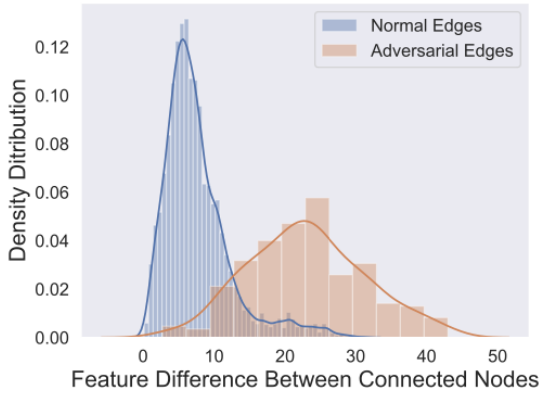
- **Pro-GNN:** explores the intrinsic properties that shared by real-world graph and uses these properties to purify poisoned graphs as well as learn a robust GNN.



Larger singular values



Larger rank



Non-smooth feature

[KDD 20] Graph Structure Learning for Robust Graph Neural Network

Graph purification (Pro-GNN)

- **Approach**:
  - Add loss to ensure the graph is **low-rank** and **sparse**:

$$\arg\min_{\mathbf{S}\in\mathcal{S}} \mathcal{L}_0 = \|\mathbf{A}-\mathbf{S}\|_F^2 + \alpha\|\mathbf{S}\|_1 + \beta\|\mathbf{S}\|_*,\ s.t., \mathbf{S} = \mathbf{S}^\top,$$

  - Add loss to penalize the **rapid changes in features** between adjacent nodes:

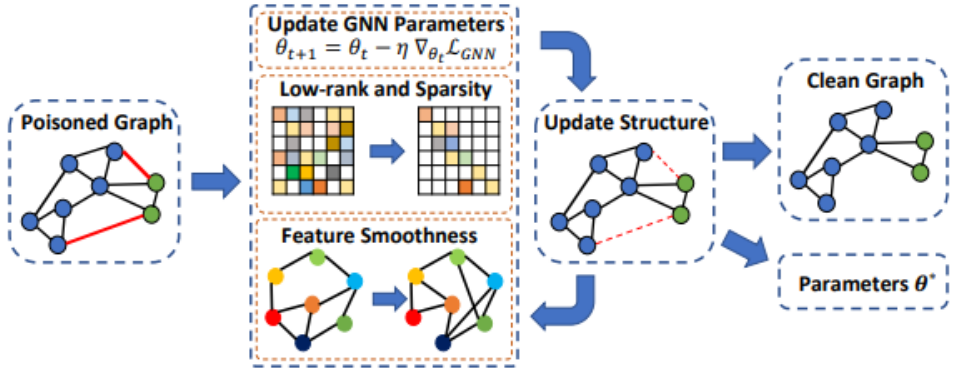
$$\arg\min_{\mathbf{S}\in\mathcal{S}} \mathcal{L} = \mathcal{L}_0 + \lambda\cdot\mathcal{L}_s = \mathcal{L}_0 + \lambda\, tr(\mathbf{X}^T\hat{\mathbf{L}}\mathbf{X}),\ s.t.,\ \mathbf{S} = \mathbf{S}^\top,$$

  - Jointly learning the desired properties of graphs and the GNN model:

$$\begin{aligned}&\arg\min_{\mathbf{S}\in\mathcal{S},\theta} \mathcal{L} = \mathcal{L}_0 + \lambda\mathcal{L}_s + \gamma\mathcal{L}_{GNN}\\ &= \|\mathbf{A}-\mathbf{S}\|_F^2 + \alpha\|\mathbf{S}\|_1 + \beta\|\mathbf{S}\|_* + \gamma\mathcal{L}_{GNN}(\theta,\mathbf{S},\mathbf{X},\mathcal{Y}_L) + \lambda tr(\mathbf{X}^T\hat{\mathbf{L}}\mathbf{X})\\ &\ s.t. \qquad \mathbf{S} = \mathbf{S}^\top,\end{aligned}$$

Effective but time-consuming.

$\boldsymbol{S}$: the clean adjacent matrix we would like to learn



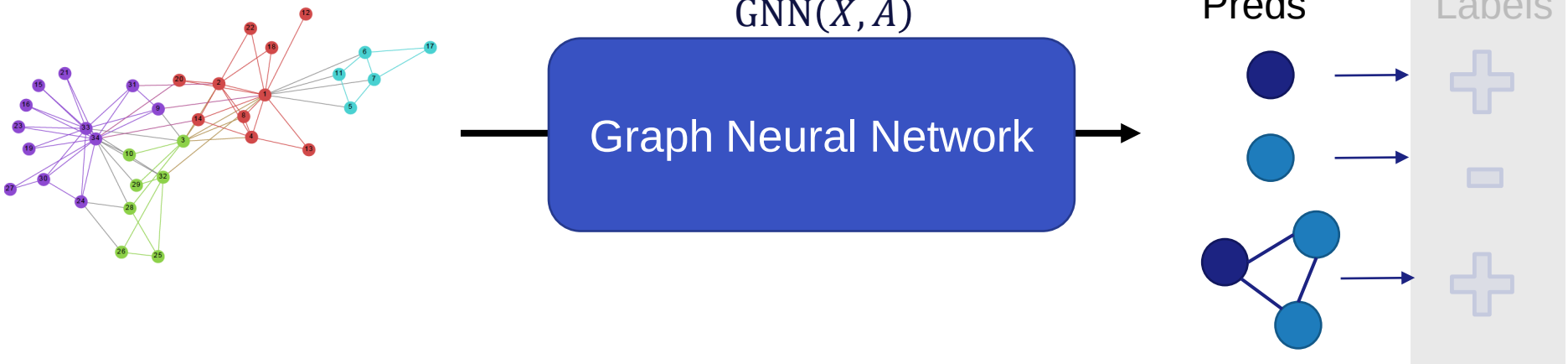
[KDD 20] Graph Structure Learning for Robust Graph Neural Network

Defense Methods

Defense Methods	Attention Mechanism	Preprocessing
GNN-Jaccard (IJCAI 19)		✓
RGCN (KDD 19)	✓	
Pro-GNN (KDD 20)		✓
GNN-SVD (WSDM 20)		✓
GNNGUARD (NeurIPS 20)	✓	
VPN (AAAI 21)	✓	

Self/Un-Supervised Learning of GNNs

What we discussed before are supervised



- **Labels are scarce**, e.g. molecular property
- **Training/Testing tasks are Non I.I.D.**

Existing Self-Supervised GNNs

	Node-Classification	Link/Metapath Prediction	Graph-Classification	Graph Reconstruction
Predictive Methods	EP-B [2], GraphSAGE [3], GROVER [7]	S^2GRL[11] SELAR[12]	N-gram Graph [4], PreGNN [5] , GROVER [7] GCC [6]	
Information-based Methods	DGI [8], GMI [9]		InfoGraph [10]	VGAE [1] VRVGA[13] SIG-VAE[14]

[1] Kipf & Welling 2016; [2] Durán & Niepert 2017; [3] Hamilton et al. 2017;
[4] Liu et al. 2019; [5] Hu et al. 2020; [6] Qiu et al. 2020; [7] Rong et al. 2020;
[8] Veli˘ckovi´c et al. 2019; [9] Peng et al. 2020; [10] Sun et al. 2020
[11] Peng, Zhen, et al. 2020 [12] Hwang, Dasol, et al. 2020
[13] Pan, Shirui, et al. 2018 [14] Hasanzadeh, Arman, et al. 2019

Tencent AI Lab

“In self-supervised learning, the system learns to predict part of its input from other parts of its input.” ---- by Yann Lecun

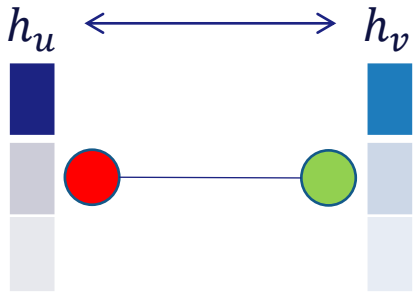
Graphs are highly structured!

Node Classification

- Two typical ways to formulate training loss

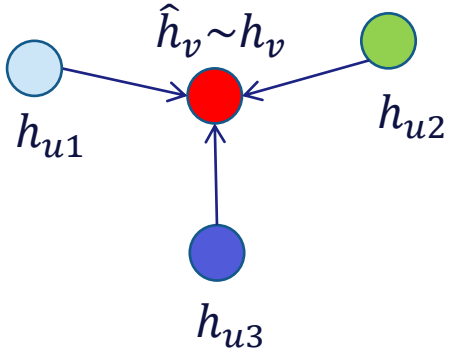
I. Enforcing Adjacent Similarity

GraphSAGE



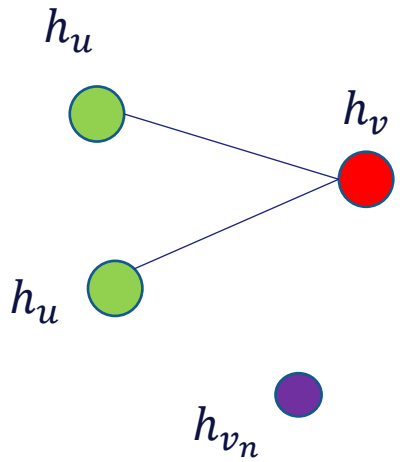
II. Reconstruction from Neighbors

EP-B



I Enforcing Adjacent Similarity

- GraphSAGE (Hamilton et al. 2017)



Enforcing nearby nodes to have similar representations, while enforcing disparate nodes to be distinct:

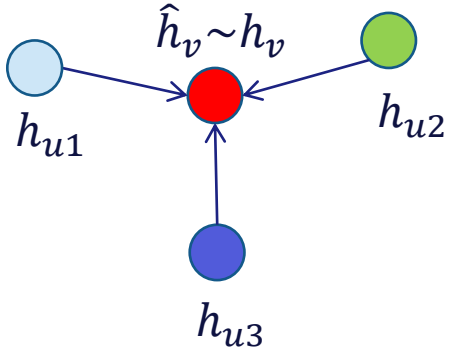
$$\min \; -E_{u\sim N(v)} \log\left(\sigma(h^T{}_u h_v)\right) - \lambda E_{v_n\sim P_n(v)} [\log(\sigma(-h^T{}_{v_n} h_v))]$$

Positive Samples Negative Samples

h_v: representation of target node;
h_u: representation of neighbor/positive node;
h_{v_n}: representation of disparate/negative node;
$P_n(v)$: negative sampling.

II Reconstruction from neighbors

- EP-B (Durán & Niepert, 2017)



The objective is to minimize the reconstruction error (regulated by the error to other nodes):

$$\min \sum_{u \in V \setminus \{v\}} [\gamma + d(\tilde{\boldsymbol{h}}_v, h_v) - d(\tilde{\boldsymbol{h}}_v, h_u)]_+$$

Hinge loss

Positive Samples

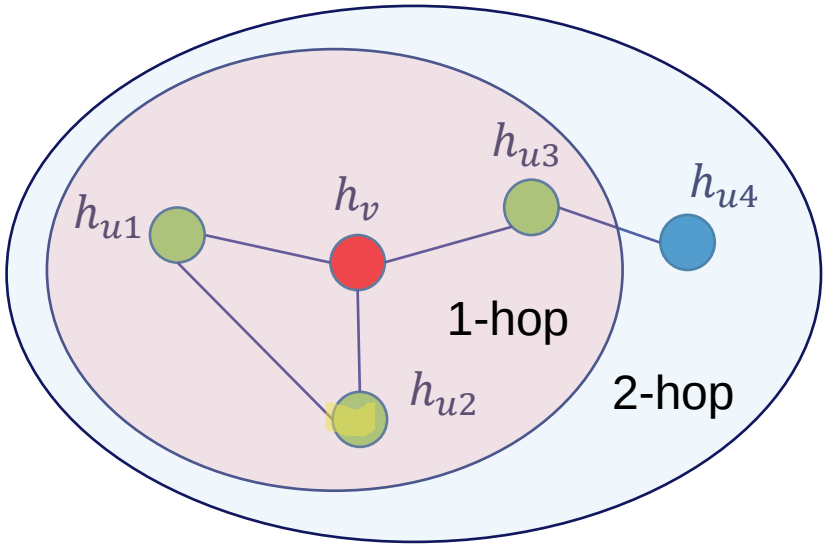
Negative Samples

h_v: representation of target node;
h_u: representation of nodes except v;
$\tilde{\boldsymbol{h}}_v$: $\mathrm{AGG}(h_l | l \in N(v))$ is the reconstruction from neighbors;
γ: the bias

Durán & Niepert. Learning Graph Representations with Embedding Propagation.

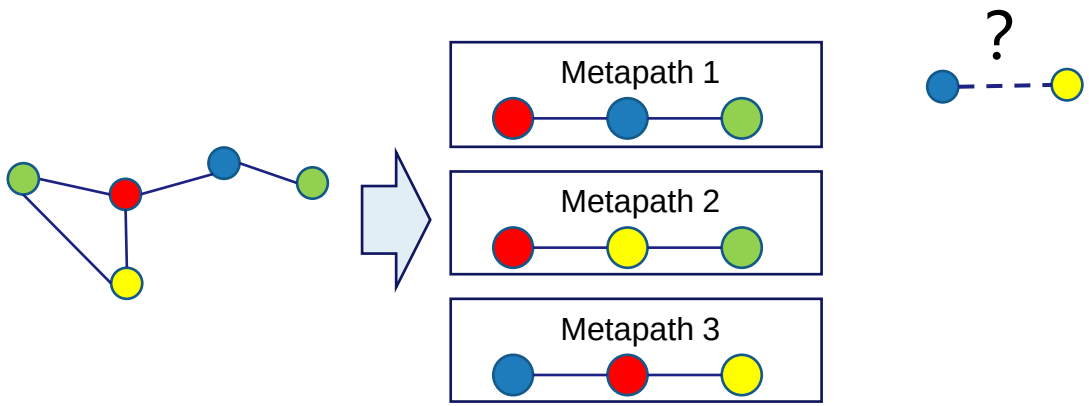
Link/Metapath Prediction

- S^2GRL(Peng, Zhen, et al. 2020)



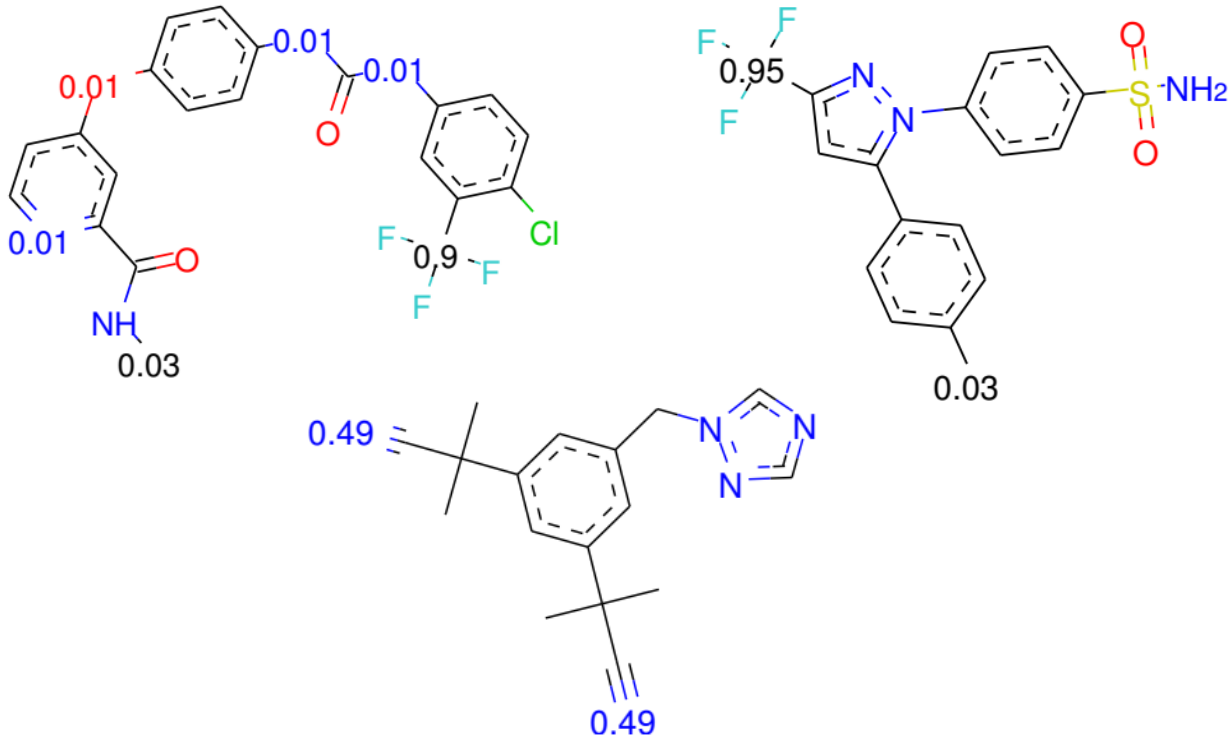
Predict hop counts (K-hop connectivity)

- SELAR(Hwang, Dasol, et al. 2020)



Predict the type of meta path between two nodes

How about graph classification/regression?



Toxicity?

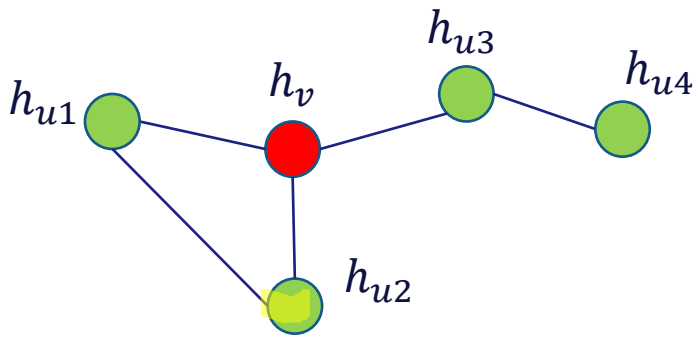
Solubility?

…

N-Gram Graph

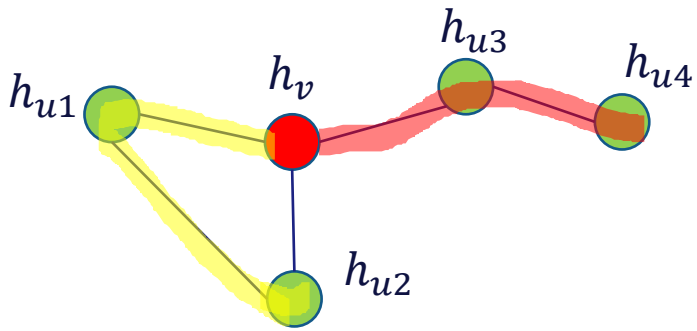
- (Liu et al. 2019)

Stage I: Node Representation



First learn node representations by CBoW-like pipeline

Stage II: Graph Representation



For all n-gram paths: $f_p = \prod_{i \in p} h_i$;

$$f_{(n)} = \sum_{p \in \text{n-gram}} f_p$$

Graph Representation: $F = [f_{(1)}, \ldots, f_{(T)}]$

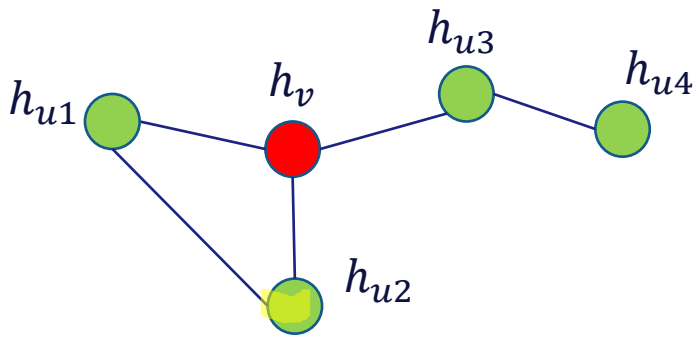
Equivalent to a GNN that needs no training

Liu et al. N-Gram Graph: Simple Unsupervised Representation for Graphs, with Applications to Molecules.

PreGNN: Node- and Graph-level Pretraining

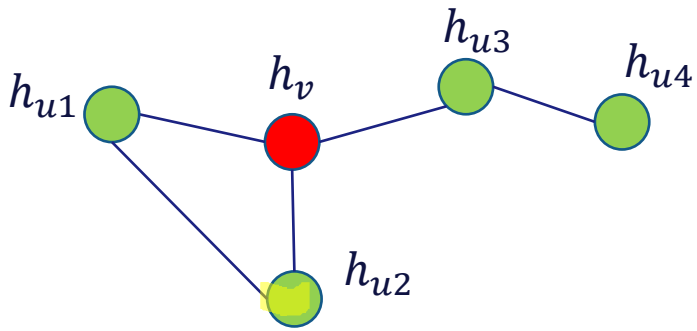
- (Hu et al. 2020)

Stage I: Node Representation



First learn node representations by ***Context Prediction or Attribute Masking***

Stage II: Graph Representation



Then perform graph-level multi-task ***Supervised Training***

$$h_G = \text{Readout}(h_v | v \in G)$$

$$min \quad \text{CrossEntropy}(h_G, y_G)$$

Both node- and graph- level training are crucial!

Hu et al. Strategies for Pre-training Graph Neural Networks.

PreGNN

- (Hu et al. 2020)

Stage I: Node Representation

Enforcing node representation to be similar to its contextual structures:

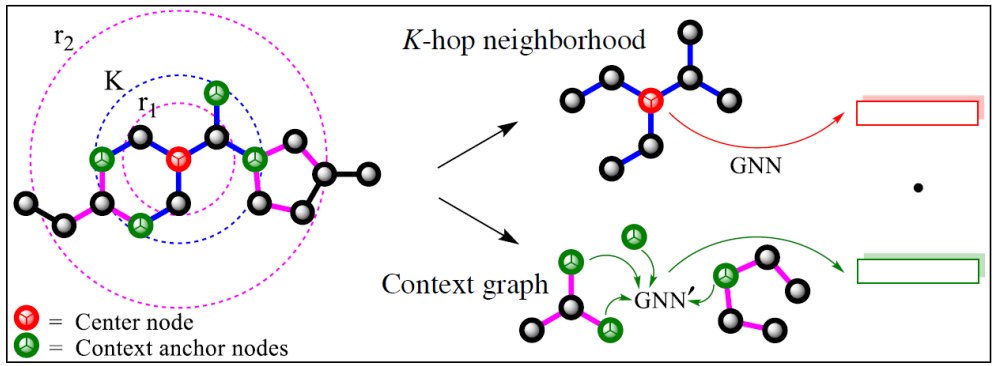
$$\min -\log\left(\sigma\left((h_v{}^K)^T C_v{}^G\right)\right) - I(v \neq v')log(1-\sigma((h_v{}^K)^T C_{v'}{}^{G'}))$$

Positive Samples

Negative Samples

Degenerates to EP-B, if r1=0, r2=K=1

Context Prediction Attribute Masking



$h_v{}^K$: K-hop information

$C_v{}^G$: Structures between r1 and r2 -hop

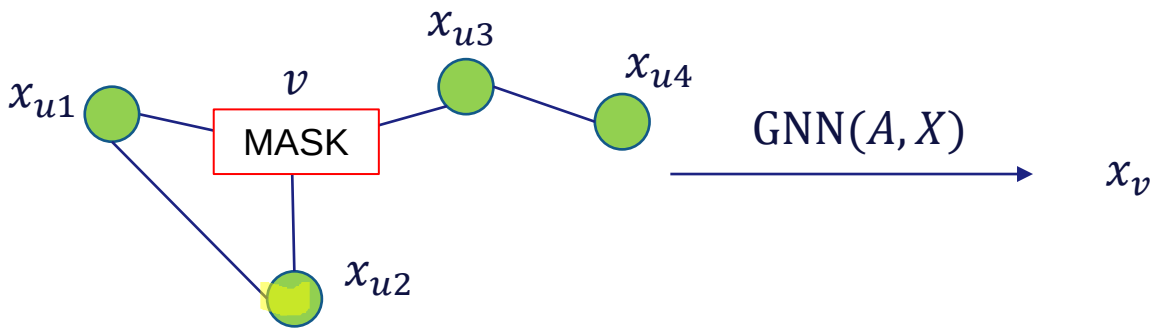
Hu et al. Strategies for Pre-training Graph Neural Networks.

PreGNN

- (Hu et al. 2020)

  Stage I: Node Representation

Mask random node/edge attribute and predict it, just like Bert:



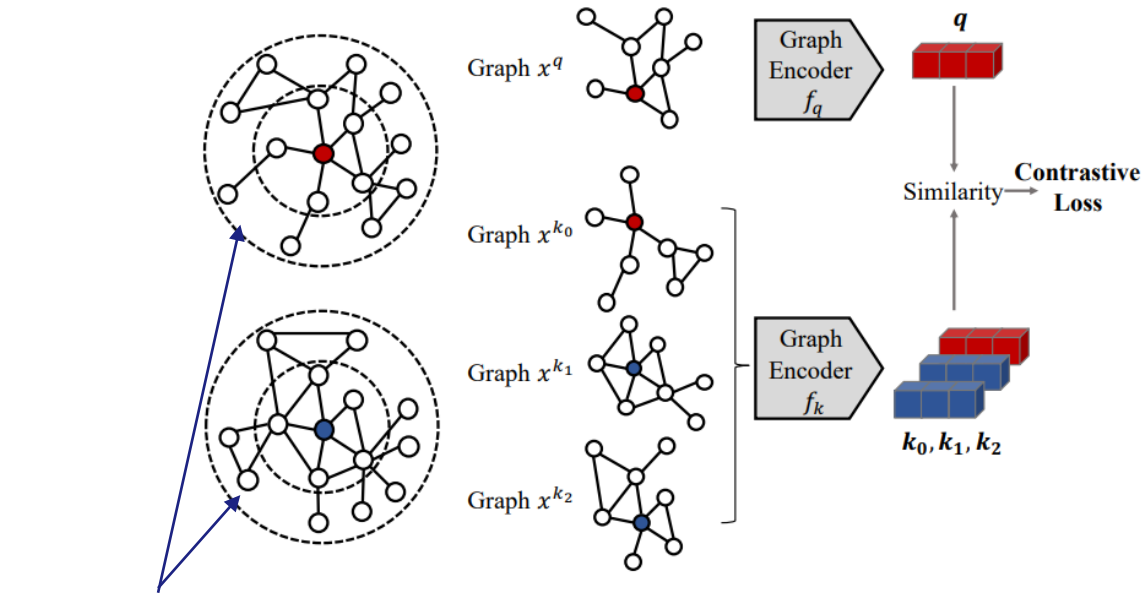
Context Prediction
Attribute Masking

Hu et al. Strategies for Pre-training Graph Neural Networks.

GCC: Contrastive learning

- (Qiu et al. 2020)

Both N-Gram Graph and PreGNN do **not** perform **graph-level unsupervised** training:



$$\min -\log \frac{\exp(q^T k_+/\tau)}{\sum_{i=0}^{K} \exp(q^T k_i/\tau)}$$

InfoNCE

q: representation of different graphs;
k: key of different graphs;
k_+: positive key generated by random graph perturbation
τ: temperature

r-ego network

But, GCC only conducts graph-level pre-training, without node-level distinguishment

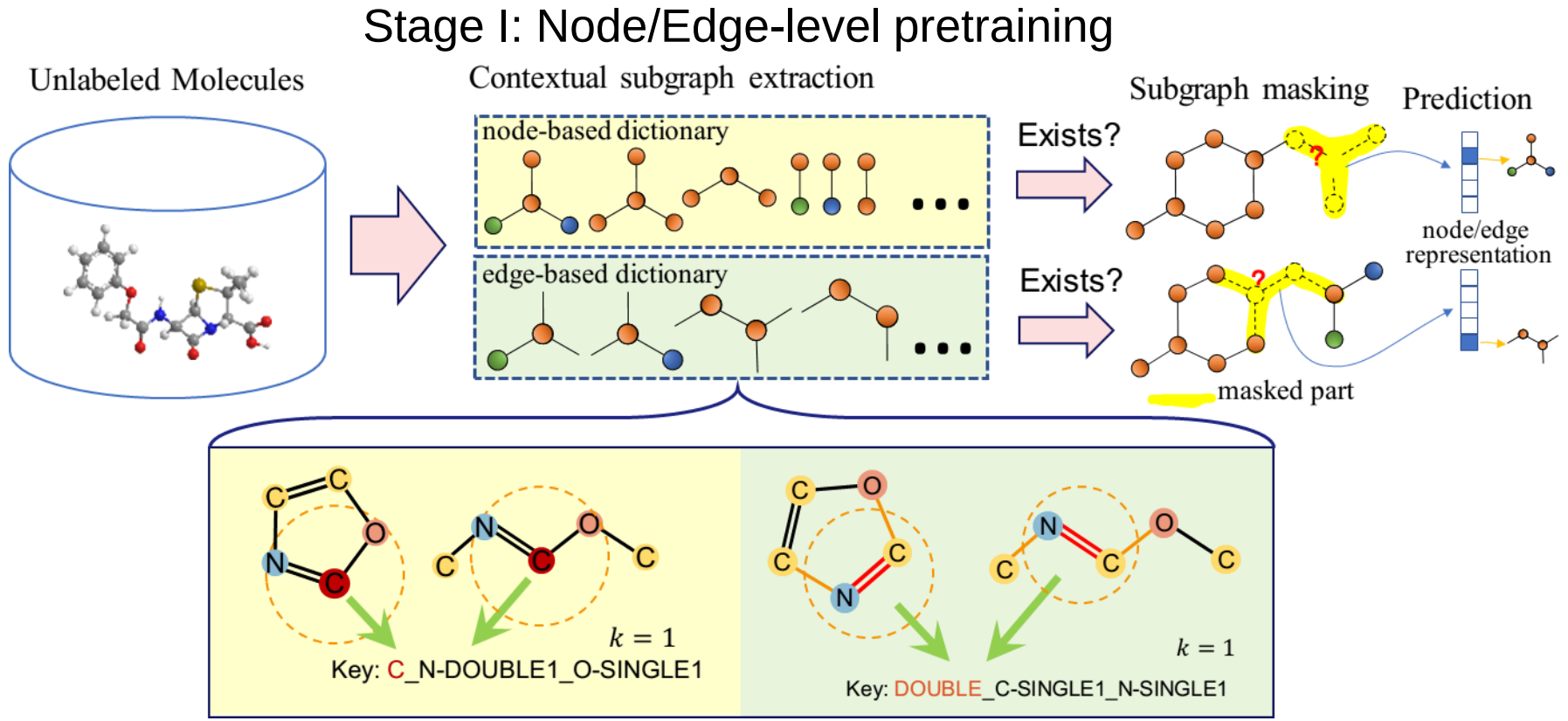
Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training.

GROVER (Rong et al. 2020)

Methods	Node-Level Self-Supervised	Graph-Level Self-Supervised
N-Gram Graph	✓	✗
PreGNN	✓	✗
GCC	✗	✓
GROVER	✓	✓

GROVER

- (Rong et al. 2020)



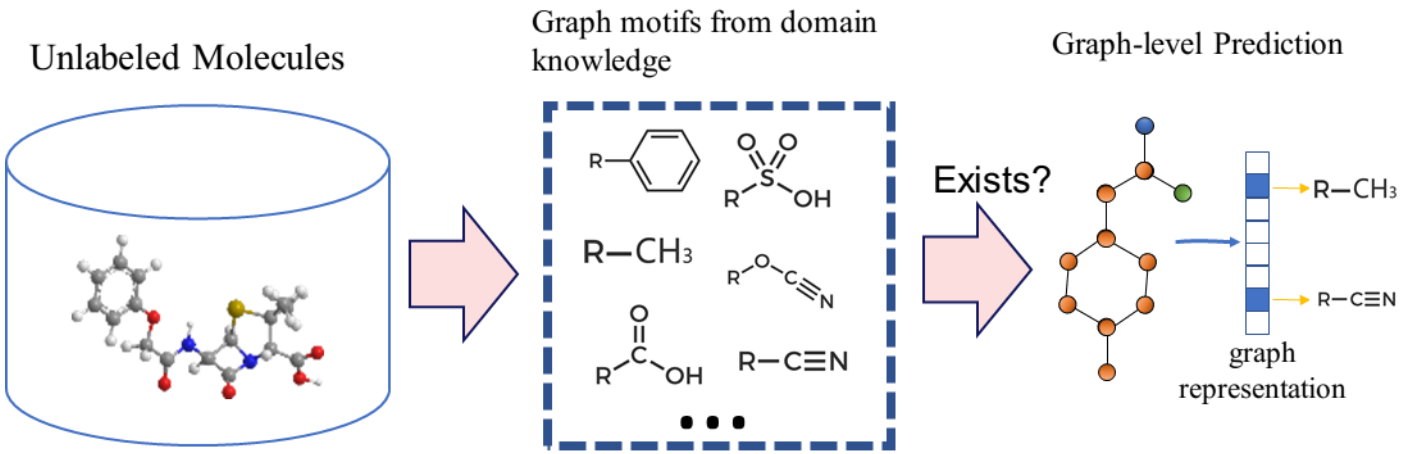
Predicting node/edge contexts instead of node labels can better capture local structures (multi-label)

Rong et al. GROVER: Self-supervised Message Passing Transformer on Large-scale Molecular Data.

GROVER

- (Rong et al. 2020)

Stage II: Graph-level pretraining



Predicting a graph if contains pre-defined graph motifs.

Rong et al. GROVER: Self-supervised Message Passing Transformer on Large-scale Molecular Data.

GROVER

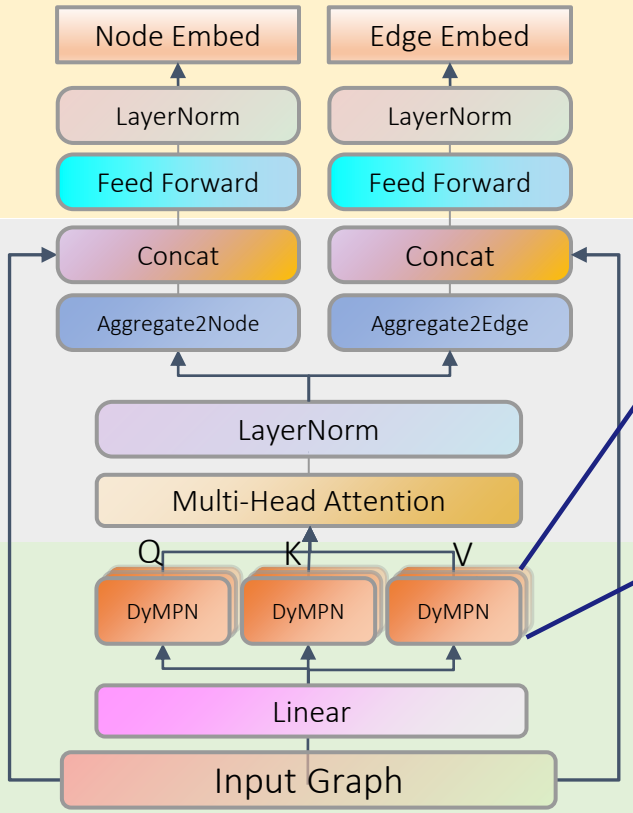
- One more thing: GTransformer

We build a more expressive and transformer-alike model: GTransformer

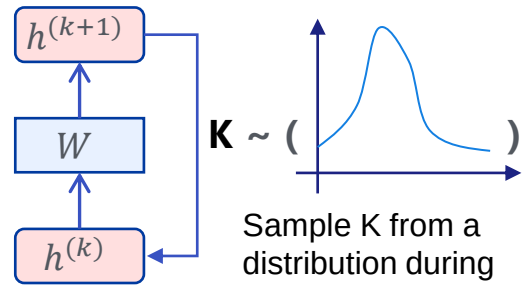
- Output for both node embedding and edge embedding.

- Multi-Head Attention: model **global interaction** between nodes/edges.
- Long-range Residual Connection: alleviating the vanishing gradient and over-smoothing.

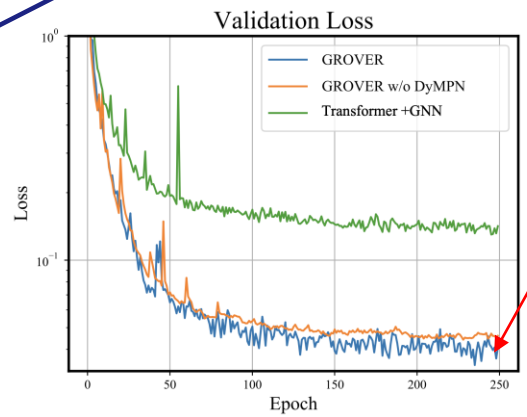
- MPNN: Extract **local structural information** of graphs.
- dyMPN: Randomize the message passing hops for the dynamic receptive field modeling.



Sample a random-hop MPNN at each iteration



Sample K from a distribution during training.



Better generalization ability

Rong et al. GROVER: Self-supervised Message Passing Transformer on Large-scale Molecular Data.

We pre-train GROVER with **100 million** parameters on **10 million** unlabeled molecules collected from ZINC15 and Chembl

Rong et al. GROVER: Self-supervised Message Passing Transformer on Large-scale Molecular Data.

GROVER

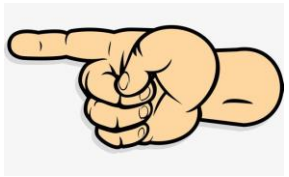
Molecular classification

Classification (Higher is better)						
Dataset # Molecules	BBBP 2039	SIDER 1427	ClinTox 1478	BACE 1513	Tox21 7831	ToxCast 8575
TF_Robust [39]	$0.860_{(0.087)}$	$0.607_{(0.033)}$	$0.765_{(0.085)}$	$0.824_{(0.022)}$	$0.698_{(0.012)}$	$0.585_{(0.031)}$
GraphConv [23]	$0.877_{(0.036)}$	$0.593_{(0.035)}$	$0.845_{(0.051)}$	$0.854_{(0.011)}$	$0.772_{(0.041)}$	$0.650_{(0.025)}$
Weave [22]	$0.837_{(0.065)}$	$0.543_{(0.034)}$	$0.823_{(0.023)}$	$0.791_{(0.008)}$	$0.741_{(0.044)}$	$0.678_{(0.024)}$
SchNet [44]	$0.847_{(0.024)}$	$0.545_{(0.038)}$	$0.717_{(0.042)}$	$0.750_{(0.033)}$	$0.767_{(0.025)}$	$0.679_{(0.021)}$
MPNN [13]	$0.913_{(0.041)}$	$0.595_{(0.030)}$	$0.879_{(0.054)}$	$0.815_{(0.044)}$	$0.808_{(0.024)}$	$0.691_{(0.013)}$
DMPNN [61]	$0.919_{(0.030)}$	$0.632_{(0.023)}$	$0.897_{(0.040)}$	$0.852_{(0.053)}$	$0.826_{(0.023)}$	$0.718_{(0.011)}$
MGCN [29]	$0.850_{(0.064)}$	$0.552_{(0.018)}$	$0.634_{(0.042)}$	$0.734_{(0.030)}$	$0.707_{(0.016)}$	$0.663_{(0.009)}$
AttentiveFP [59]	$0.908_{(0.050)}$	$0.605_{(0.060)}$	$0.933_{(0.020)}$	$0.863_{(0.015)}$	$0.807_{(0.020)}$	$0.579_{(0.001)}$
N-GRAM [28]	$0.912_{(0.013)}$	$0.632_{(0.005)}$	$0.855_{(0.037)}$	$0.876_{(0.035)}$	$0.769_{(0.027)}$	-[2]
HU. et.al[18]	$0.915_{(0.040)}$	$0.614_{(0.006)}$	$0.762_{(0.058)}$	$0.851_{(0.027)}$	$0.811_{(0.015)}$	$0.714_{(0.019)}$
$\text{GROVER}_{\text{base}}$	$0.936_{(0.008)}$	$0.656_{(0.06)}$	$0.925_{(0.013)}$	$0.878_{(0.016)}$	$0.819_{(0.020)}$	$0.723_{(0.010)}$
$\text{GROVER}_{\text{large}}$	$\mathbf{0.940}_{(0.019)}$	$\mathbf{0.658}_{(0.023)}$	$\mathbf{0.944}_{(0.021)}$	$\mathbf{0.894}_{(0.028)}$	$\mathbf{0.831}_{(0.025)}$	$\mathbf{0.737}_{(0.010)}$

Rong et al. GROVER: Self-supervised Message Passing Transformer on Large-scale Molecular Data.

Existing Self-Supervised GNNs

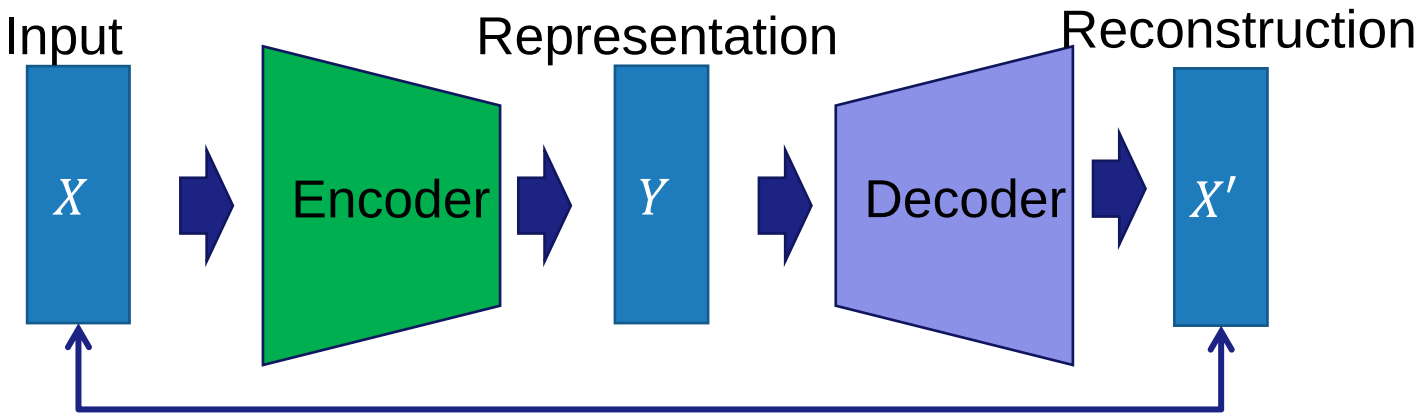
	Node-Classification	Link/Metapath Prediction	Graph-Classification	Graph Reconstruction
Predictive Methods	EP-B [2], GraphSAGE [3], GROVER [7]	S^2GRL[11] SELAR[12]	N-gram Graph [4], PreGNN [5] , GROVER [7] GCC [6]	
Information-based Methods	DGI [8], GMI [9]		InfoGraph [10]	VGAE [1] VRVGA[13] SIG-VAE[14]



[1] Kipf & Welling 2016; [2] Durán & Niepert 2017; [3] Hamilton et al. 2017;
[4] Liu et al. 2019; [5] Hu et al. 2020; [6] Qiu et al. 2020; [7] Rong et al. 2020;
[8] Veli˘ckovi´c et al. 2019; [9] Peng et al. 2020; [10] Sun et al. 2020
[11] Hwang, Dasol, et al. 2020 [12] Peng, Zhen, et al.
[13] Pan, Shirui, et al. 2018 [14] Hasanzadeh, Arman, et al. 2019

What makes a good representation?

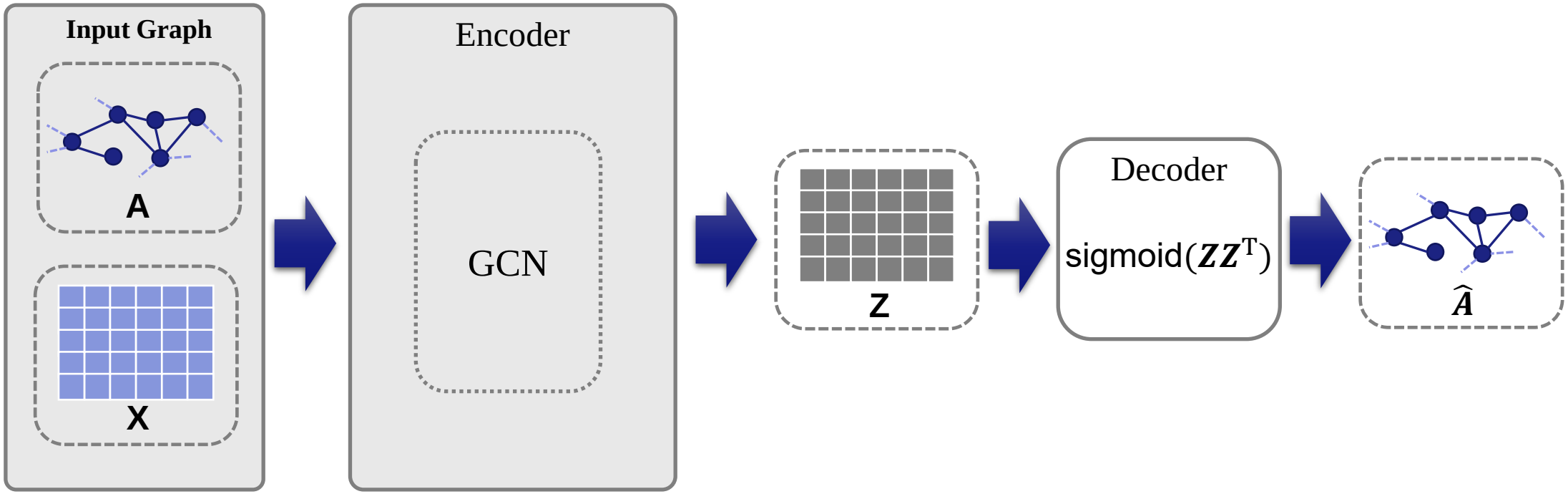
Auto-Encoder (AE)



"One natural criterion that we may expect any good representation to meet, at least to some degree, is to **retain a significant amount of information about the input**." by Vincent et al. 2010

Hinton & Salakhutdinov 2006; Vincent et al. 2010

Graph Auto-Encoders (VGAE)



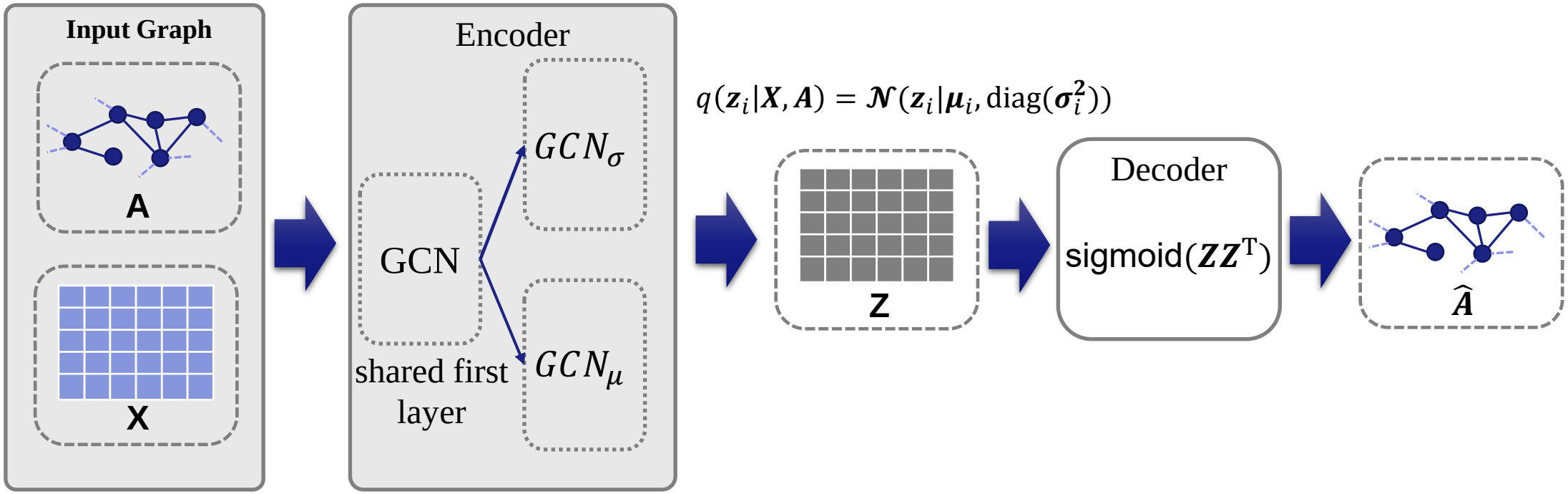
The overall loss:

$$L = \mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A})}[\log p(\boldsymbol{A}|\boldsymbol{Z})]$$

The reconstruction loss.

Kipf, T. N., & Welling, M. (2016).

Variational Graph Auto-Encoders (VGAE)



The overall loss:

$$L = \mathbb{E}_{q(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A})}[\log p(\boldsymbol{A}|\boldsymbol{Z})] - \mathrm{KL}[q(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{A})||p(\boldsymbol{Z})]$$

The reconstruction loss.

The KL divergence between q(.) and p(.)

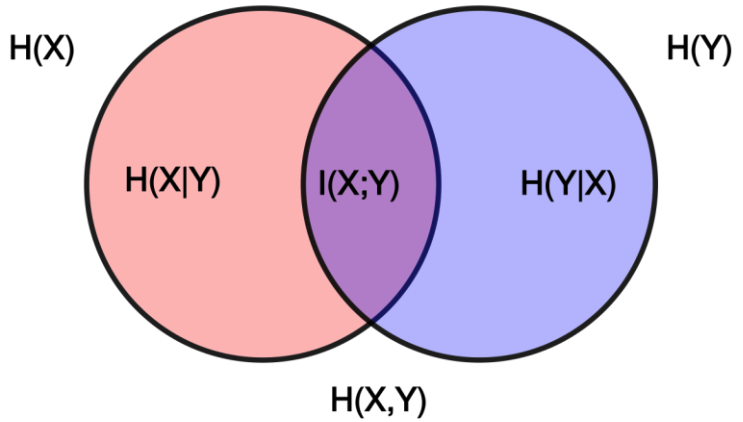
Kipf, T. N., & Welling, M. (2016).

What makes a good representation?

- A more direct way, other than AE?
  - Yes, **Mutual Information (MI)**.

$$I(X;Y) = D_{KL}(p(X)p(Y)||p(X,Y))$$
$$= H(X) - H(X|Y)$$

Entropy Conditional Entropy



- $0 \leq I(X;Y) \leq H(X)$ or H(Y);
- $I(X;Y) = 0$ iff X and Y are independent random variables;
- $I(X;Y) = H(X) = H(Y)$, if X and Y are determinately related, i.e. $H(X|Y) = 0$

AE is a lower bound of MI

(Hjelm et al. 2019)

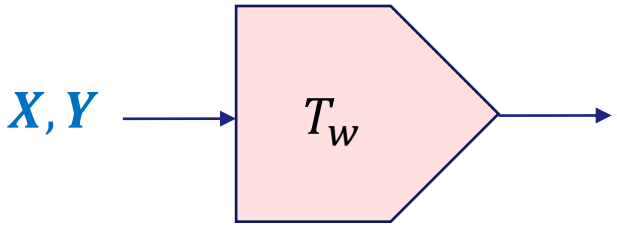
$$I(X;Y) = H(X) - H(X|Y) \geq H(X) - R(X|Y)$$

Mutual Information

Reconstruction error

Computing MI is hard and not end-to-end, until recently (CPC, Oord et al., 2018; MINE, Belghazi et al., 2018; Nowozin et al., 2016; Hjelm et al. 2019)

Estimating/Maximizing MI (Hjelm et al. 2019)



$$\max_{T_w} \hat{I}_{T_w}(X;Y) \to \max I(X;Y)$$

Maximize Lower bound of MI

MINE (Belghazi et al., 2018):

$$I^{\text{MINE}}(X;Y) \triangleq E_{p(X,Y)}[T_w(x,y)] - \log E_{p(X)p(Y)}[\exp(T_w(x,y))]$$

JSD MI estimator (Nowozin et al., 2016):

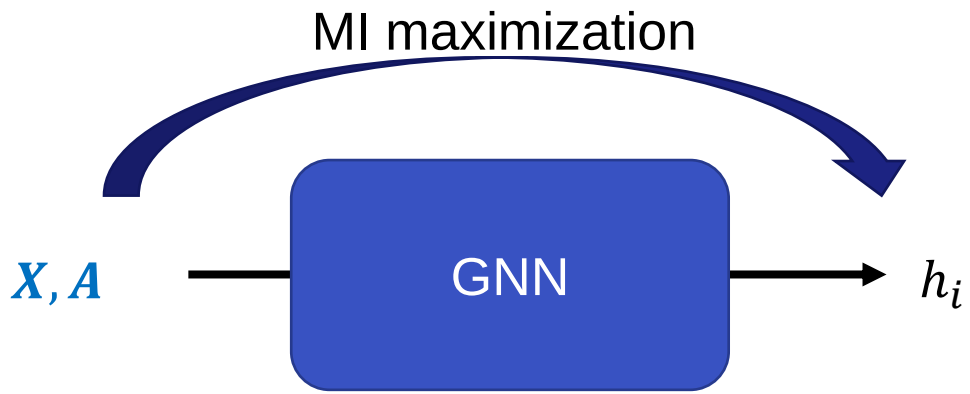
$$I^{\text{JSD}}(X;Y) \triangleq E_{p(X,Y)}\left[\log \sigma\left(T_w(x,y)\right)\right] + E_{p(X)p(Y)}[\log(1-\sigma\left(T_w(x,y)\right))]$$

InfoNCE MI estimator (Oord et al., 2018):

$$I^{\text{NCE}}(X;Y) \triangleq E_{p(X,Y)}[\log \frac{\exp T_w(x,y)}{\sum_{x' \sim p(X)} \exp T_w(x',y)}]$$

Deep Graph Infomax (DGI)

- (Velickovic et al. 2019)



The JSD MI estimator is applied:

$$\max_{\text{GNN}} I(X, A; h_i) \approx \max \log(D(h_i; \boldsymbol{X}, \boldsymbol{A})) + \log(1 - D(\tilde{h}_i; \boldsymbol{X}, \boldsymbol{A}))$$

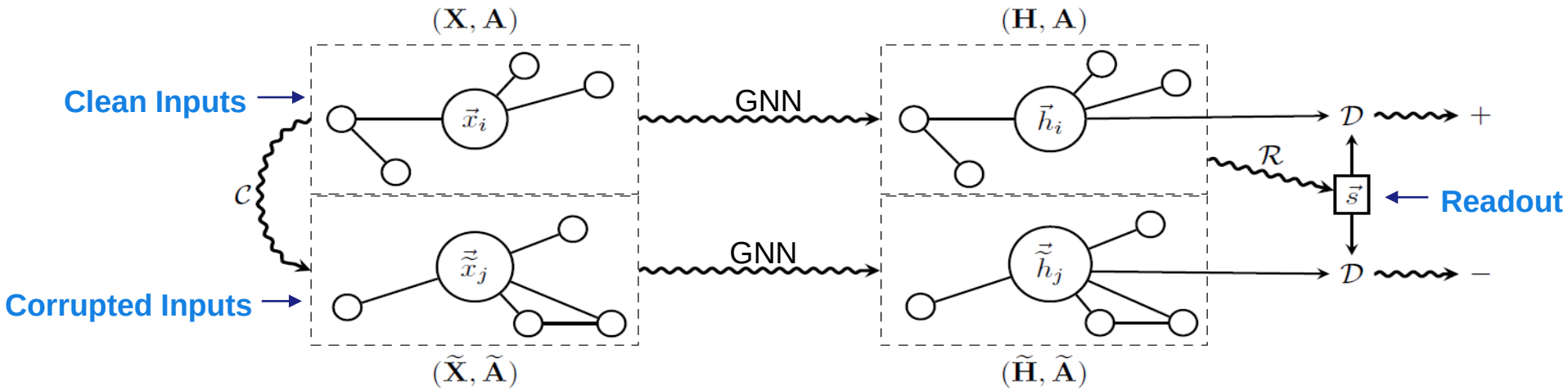
$h_i = \text{GNN}(X, A)$ $\tilde{h}_i$ negative sample

Deep Graph Infomax (DGI)

It is hard to directly compute $D(\tilde{h}_i; \boldsymbol{X}, \boldsymbol{A})$, thus DGI resorts to readout $\boldsymbol{s} = R(\boldsymbol{X}, \boldsymbol{A})$:

$$\max_{GNN} I(X, A; h_i) \approx \max \log(D(h_i; \boldsymbol{X}, \boldsymbol{A})) + \log(1 - D(\tilde{h}_i; \boldsymbol{X}, \boldsymbol{A}))$$

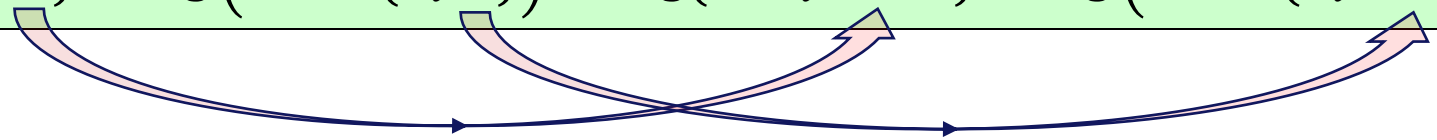
$$\max_{\text{GNN}} I(X, A; h_i) = \max_{\text{GNN}} \log(D(h_i; \boldsymbol{s})) + \log(1 - D(\tilde{h}_i; \boldsymbol{s}))$$



Deep Graph Infomax (DGI)

It can be proved that, if the readout $s = R(X, A)$ is injective,

$$\log\big(D(h_i; \boldsymbol{s})\big) + \log\Big(1 - D\big(\tilde{h}_i; \boldsymbol{s}\big)\Big) = \log\big(D(h_i; \boldsymbol{X}, \boldsymbol{A})\big) + \log\Big(1 - D\big(\tilde{h}_i; \boldsymbol{X}, \boldsymbol{A}\big)\Big)$$



It can be also proved that, if $|\boldsymbol{X}| = |\boldsymbol{s}|$ is finite,

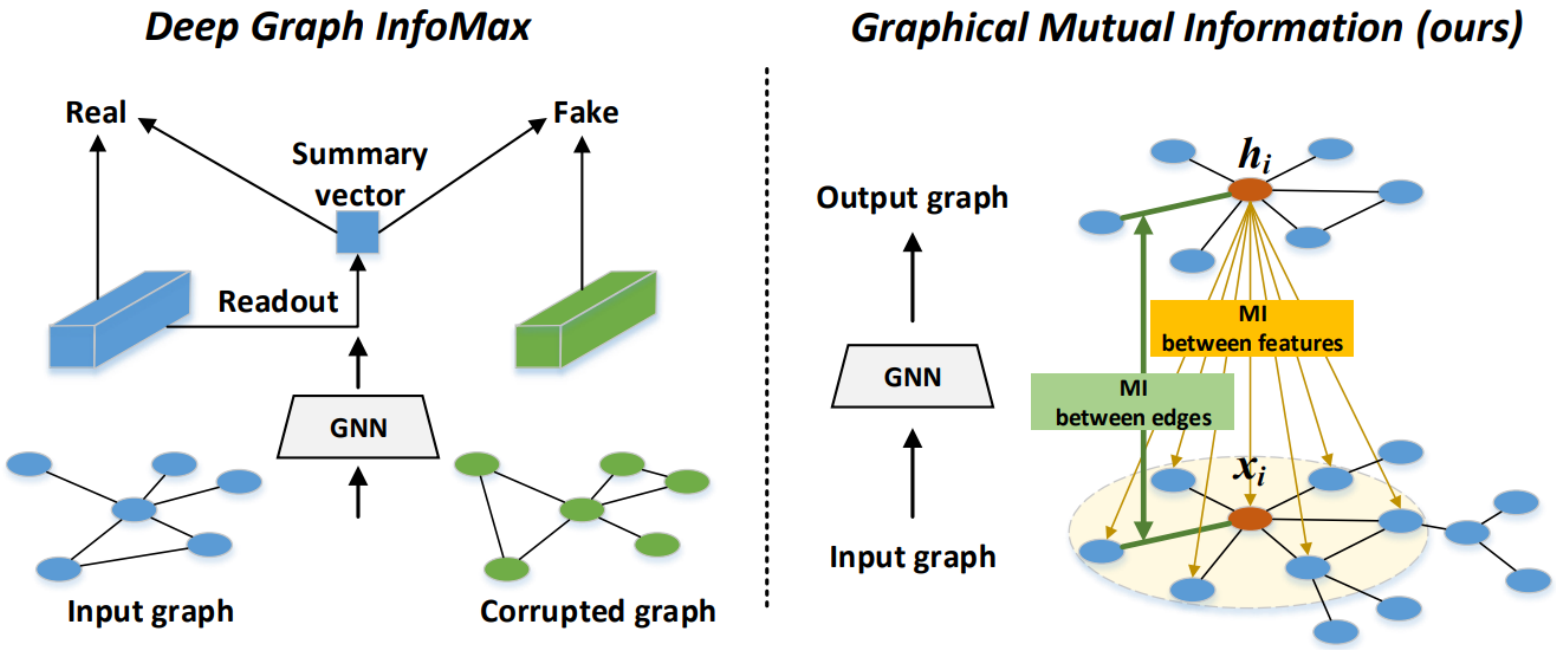
$$\max \log\big(D(h_i; \boldsymbol{s})\big) + \log\Big(1 - D\big(\tilde{h}_i; \boldsymbol{s}\big)\Big) = \max I(h_i; X, A)$$

- Some issues in DGI

  - Computing MI requires the injectivity of readout function — Indirect
  - It resorts to graph corruption to generate negative samples — Inefficient
  - Distinct encoders and corruption functions for different tasks

GMI: Graphical Mutual Information

- (Peng et al. 2020)



The basic idea of GMI is to compute the MI directly.

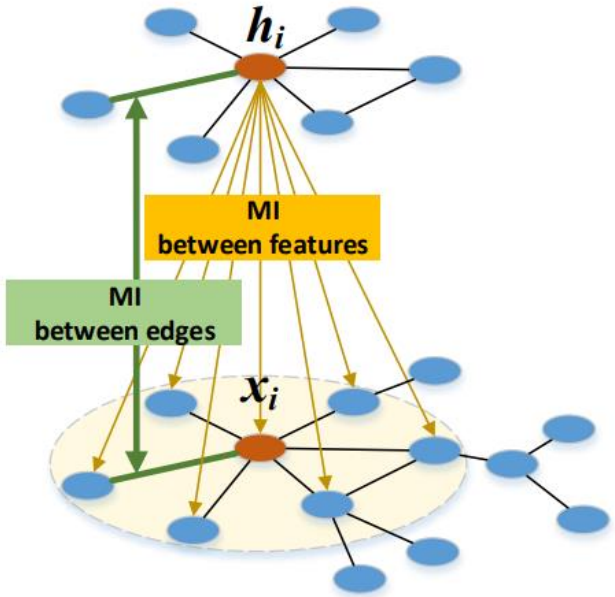
Peng et al. Graph Representation Learning via Graphical Mutual Information Maximization.

GMI: Graphical Mutual Information

We define that,

$$I(X, A; h_i) \approx I(X; h_i) + \sum_{j \in N(i)} I(\sigma(h_i{}^T h_j); A_{ij})$$

Feature MI　　Topology MI



- It is both feature- and edge- aware;
- No need to readout or corruption;
- Feature MI can be further decomposed;

The basic idea of GMI is to compute the MI directly.

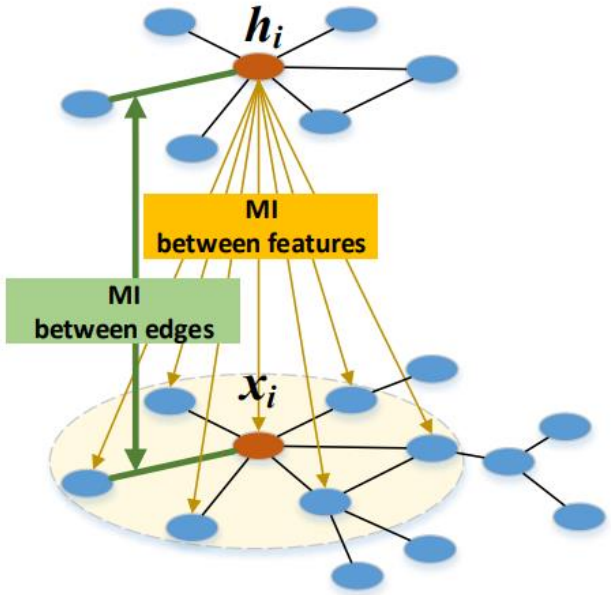
Peng et al. Graph Representation Learning via Graphical Mutual Information Maximization.

GMI: Graphical Mutual Information

- (Peng et al. 2020)

It can be proved that, if certain mild condition meets,

$$I(X; h_i) = \sum_{j \in N(i)} w_{ij} I(x_j;\ h_i),\ \text{for } 0 \le w_{ij} \le 1$$



The global MI is decomposed into a weighted sum of local MIs.
It is not a bad idea to let $w_{ij} = \sigma({h_i}^T h_j)$

We then apply the JSD MI estimator to compute $I(x_j;\ h_i)$ and $I(\sigma({h_i}^T h_j); A_{ij})$

Peng et al. Graph Representation Learning via Graphical Mutual Information Maximization.

GMI: Graphical Mutual Information

- Node Classification

We use a universal backbone (GCN) for all tasks, different from DGI

Algorithm	Transductive			Inductive	
	Cora	Citeseer	PubMed	Reddit	PPI
EP-B loss	79.4 ± 0.1	69.3 ± 0.2	78.6 ± 0.2	93.8 ± 0.03	61.8 ± 0.04
DGI loss	82.2 ± 0.2	72.2 ± 0.2	78.9 ± 0.3	94.3 ± 0.02	62.3 ± 0.02
FMI (ours)	78.3 ± 0.1	72.0 ± 0.2	**79.1 ± 0.3**	**94.7 ± 0.03**	**64.8 ± 0.03**
GMI-mean (ours)	**82.7 ± 0.1**	**73.0 ± 0.3**	**80.1 ± 0.2**	**95.0 ± 0.02**	**65.0 ± 0.02**
GMI-adaptive (ours)	**83.0 ± 0.3**	**72.4 ± 0.1**	**79.9 ± 0.2**	**94.9 ± 0.02**	**64.6 ± 0.03**

Codes: https://github.com/zpeng27/GMI

Peng et al. Graph Representation Learning via Graphical Mutual Information Maximization.

GMI: Graphical Mutual Information

- Link Prediction

We use an universal backbone (GCN) for all tasks

Algorithm	Cora			BlogCatalog			Flickr			PPI
	20.0%	50.0%	70.0%	20.0%	50.0%	70.0%	20.0%	50.0%	70.0%	22.7%
DGI	95.6±0.3	94.6±0.4	94.4±0.2	77.2±0.4	76.4±0.4	75.5±0.3	90.3±0.3	89.0±0.4	74.1±0.7	77.4±0.1
FMI (ours)	**97.2±0.2**	**95.2±0.1**	**95.0±0.1**	**81.2±0.2**	**79.5±0.4**	75.1±0.2	**92.7±0.3**	**92.2±0.3**	**90.6±0.4**	**79.8**±0.2
GMI (ours)	**97.9±0.3**	**96.4±0.2**	**96.3±0.1**	**84.1±0.3**	**83.6±0.2**	**82.5±0.1**	**92.0±0.2**	**90.1±0.3**	**88.5±0.2**	**80.0±0.2**

Codes: https://github.com/zpeng27/GMI

Peng et al. Graph Representation Learning via Graphical Mutual Information Maximization.

Summary

	Node-Classification	Link/Metapath Prediction	Graph-Classification	Graph Reconstruction
Predictive Methods	EP-B [2], GraphSAGE [3], GROVER [7]	S^2GRL[11] SELAR[12]	N-gram Graph [4], PreGNN [5] , GROVER [7] GCC [6]	
Information-based Methods	DGI [8], GMI [9]		InfoGraph [10]	VGAE [1] VRVGA[13] SIG-VAE[14]

Break

5 min

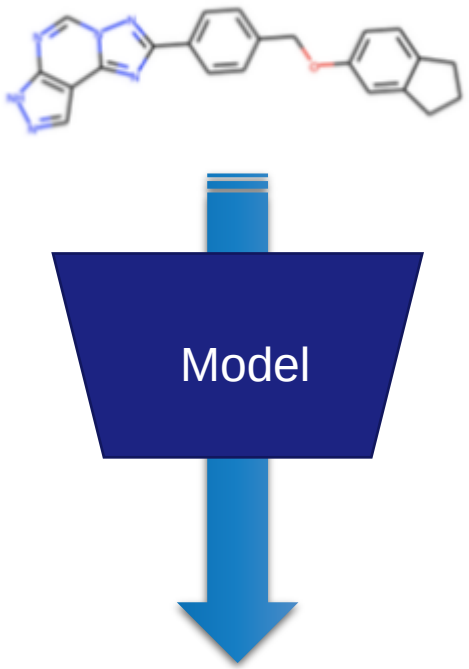
Other Advanced Topics of GNNs

Subgraph Recognition

Problem of Subgraph Recognition

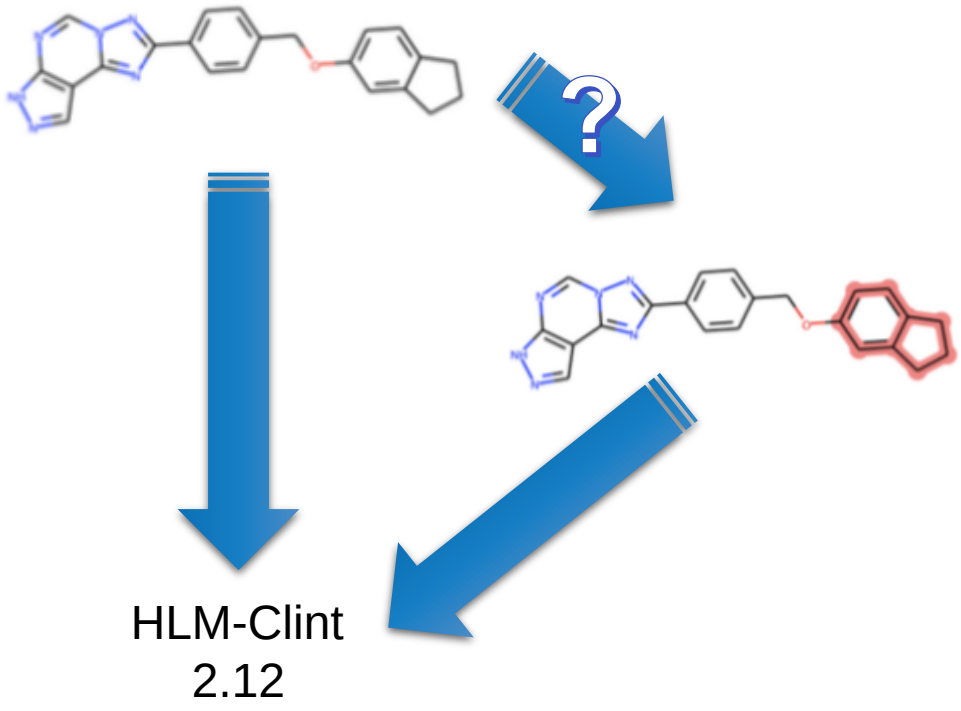
Graph Classification

- Predict the property of a graph



Subgraph Recognition

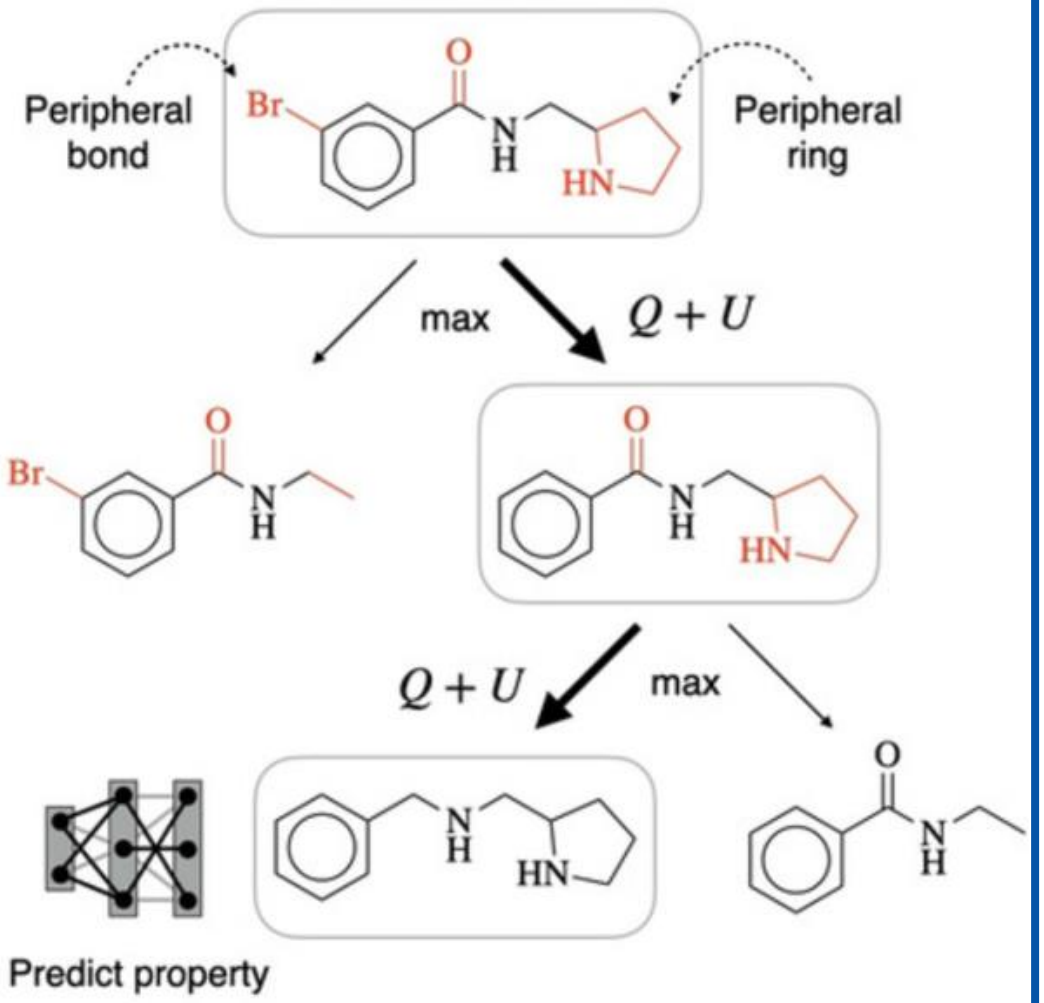
- Recognize the sub-graph that most predict the property



Yu et al. Graph Information Bottleneck for Subgraph Recognition. ICLR 2021

Monte Carlo Tree Search

MCTS (Jin et al. 2020)



Solving Steps

- Iteratively remove peripheral bonds and rings to find subgraphs
- Evaluate each subgraph using the fixed property predictor
- Q and U functions are MCTS parameters

Weakness

- Need **many iterations** to delete components **step by step**
- Require **two stages** with a **fixed** property predictor
- Seek in a **large** search space

Jin et al. Multi-objective molecule generation using interpretable substructures. ICML 2020

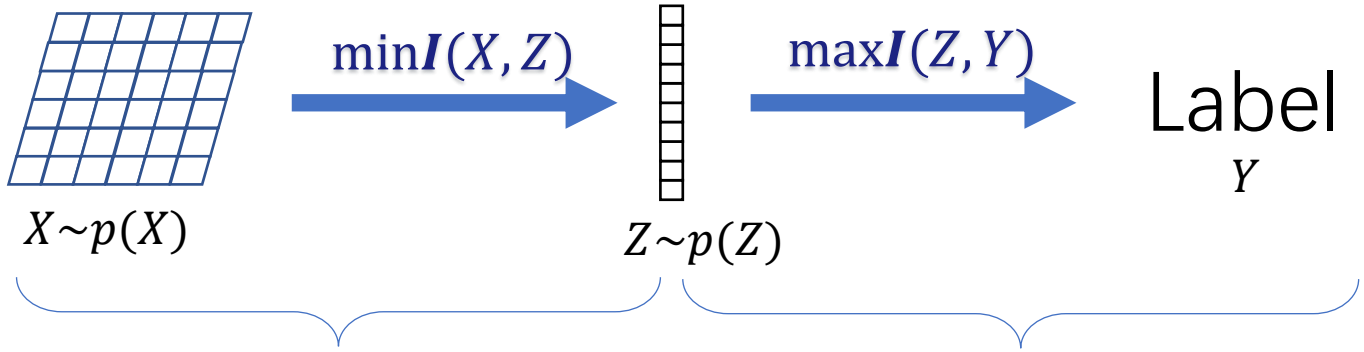
Information Bottleneck

Information Bottleneck

$$\max_Z I(Y,Z) - \beta I(X,Z)$$

- Where $I(\cdot,\cdot)$ indicates the Mutual Information of two variables
- Compressed information between X and Z
- Most informative Z to reflect the properties Y

Traditional IB objective: $\max_Z I(Y,Z) - \beta I(Z,X)$



Alemi et al. Deep Variational Information Bottleneck. ICLR 2017

Variational Information Bottleneck

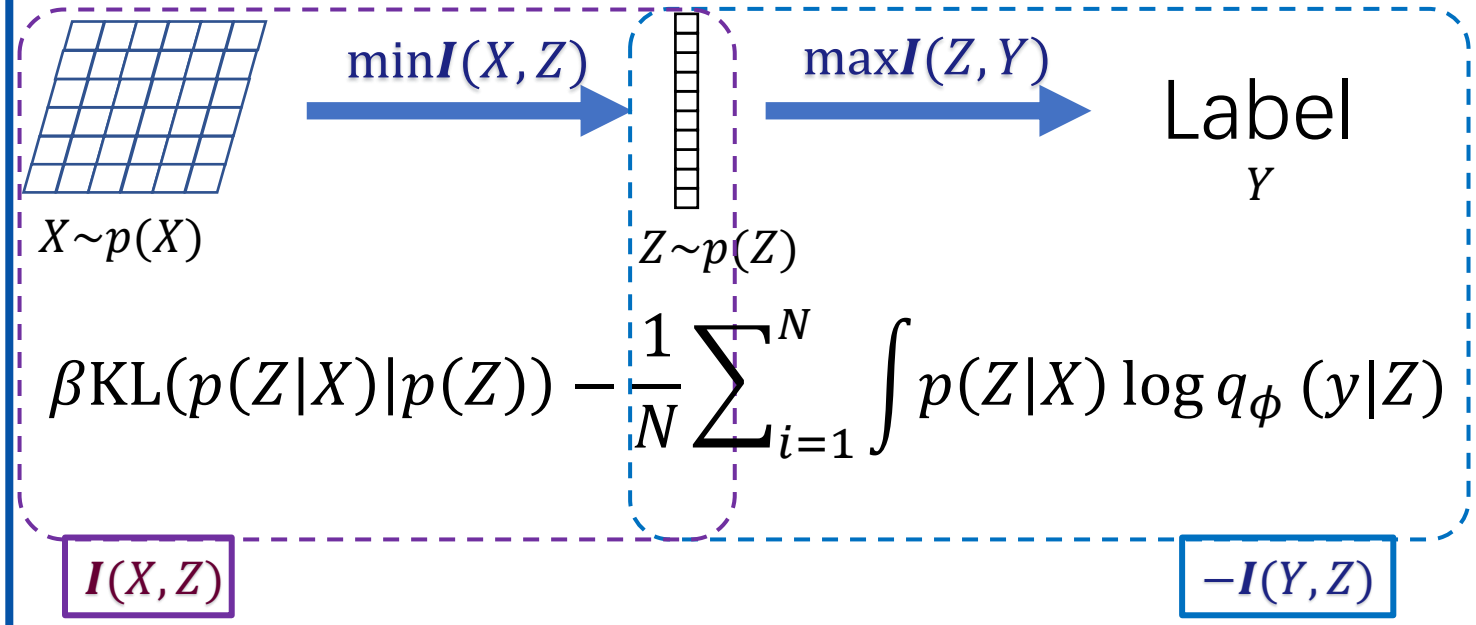
Information Bottleneck

$$\max_Z I(Y,Z) - \beta I(X,Z)$$

- Where $I(\cdot,\cdot)$ indicates the Mutual Information of two variables
- Solution: Variational information bottleneck

$$\mathcal{L}_{VIB} = \frac{1}{N}\sum_{i=1}^{N}\int p(Z|X)\log q_\phi(y|Z) - \beta \mathrm{KL}(p(Z|X)|p(Z))$$

Traditional IB objective: $\max_Z I(Y,Z) - \beta I(X,Z)$



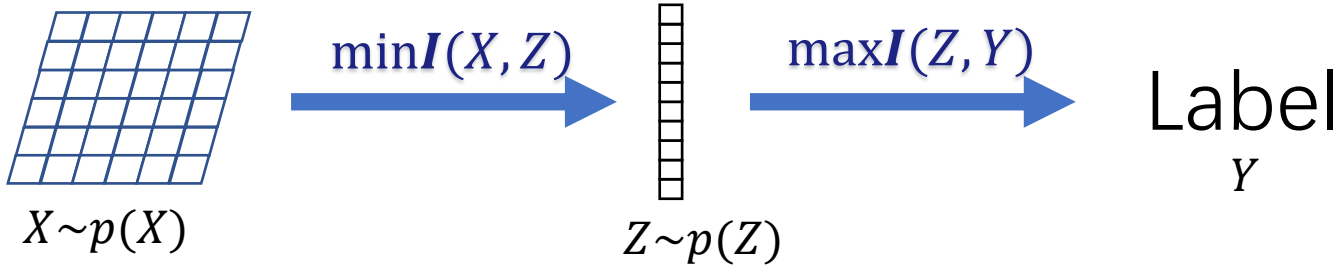
Alemi et al. Deep Variational Information Bottleneck. ICLR 2017

Graph Information Bottleneck

Problem

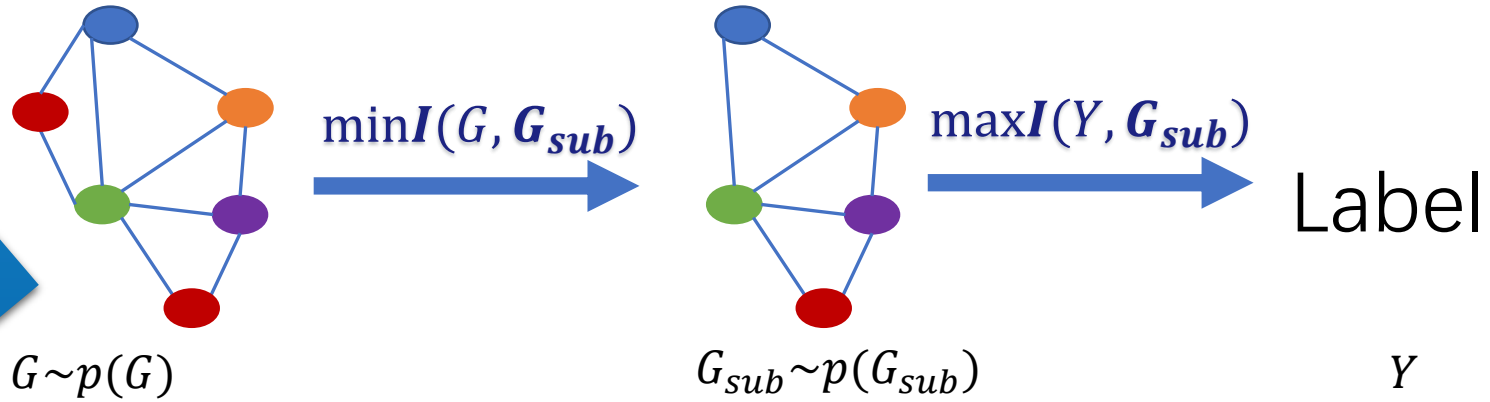
Can we find an informative yet compressed subgraph, G_{sub}, from the graph, G?

Traditional IB objective: $\max_Z I(Y,Z) - \beta I(Z,X)$



We leverage the idea from Information Bottleneck

GIB objective: $\max_{G_{sub}} I(Y,G_{sub}) - \beta I(G,G_{sub})$



Yu et al. Graph Information Bottleneck for Subgraph Recognition. ICLR 2021

Graph Information Bottleneck

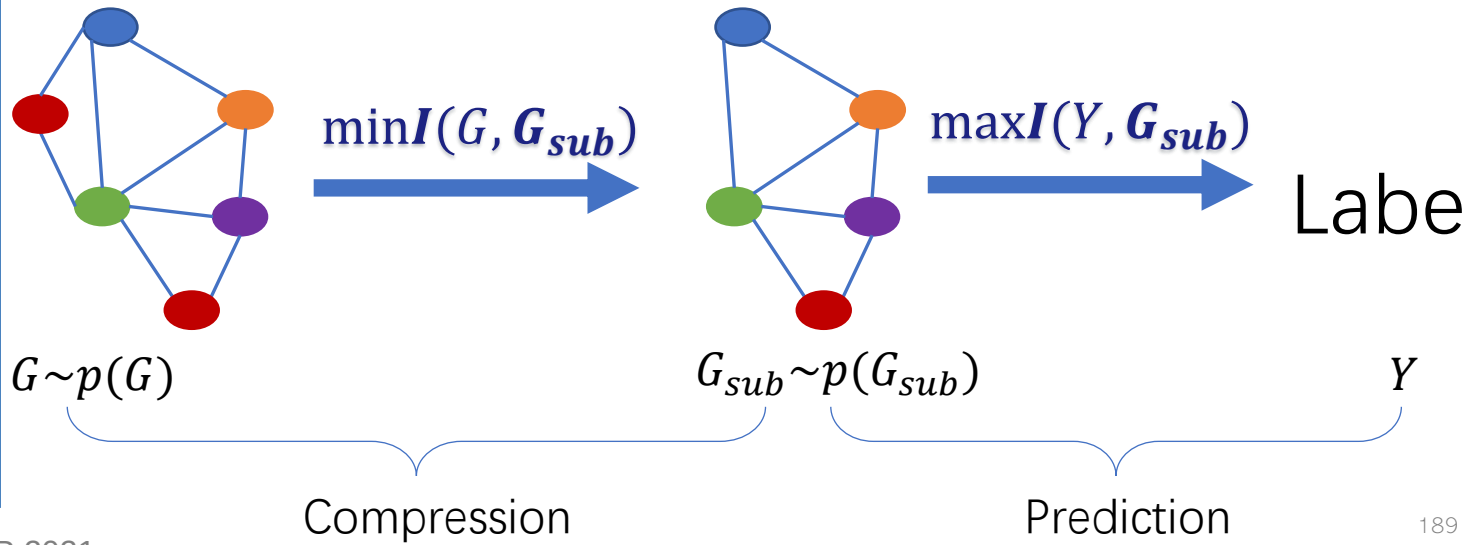
VIB to Graph?

- Use variational information bottleneck? **NO**

$$\frac{1}{N}\sum_{i=1}^{N}\int p(G_{sub}|G)\log q_\phi(y|G_{sub}) - \beta \mathrm{KL}(p(G_{sub}|G)|p(G_{sub}))$$

- The distribution of $p(G_{sub})$ is unknown

GIB objective: $\max_{G_{sub}} I(Y, G_{sub}) - \beta I(G, G_{sub})$



Yu et al. Graph Information Bottleneck for Subgraph Recognition. ICLR 2021

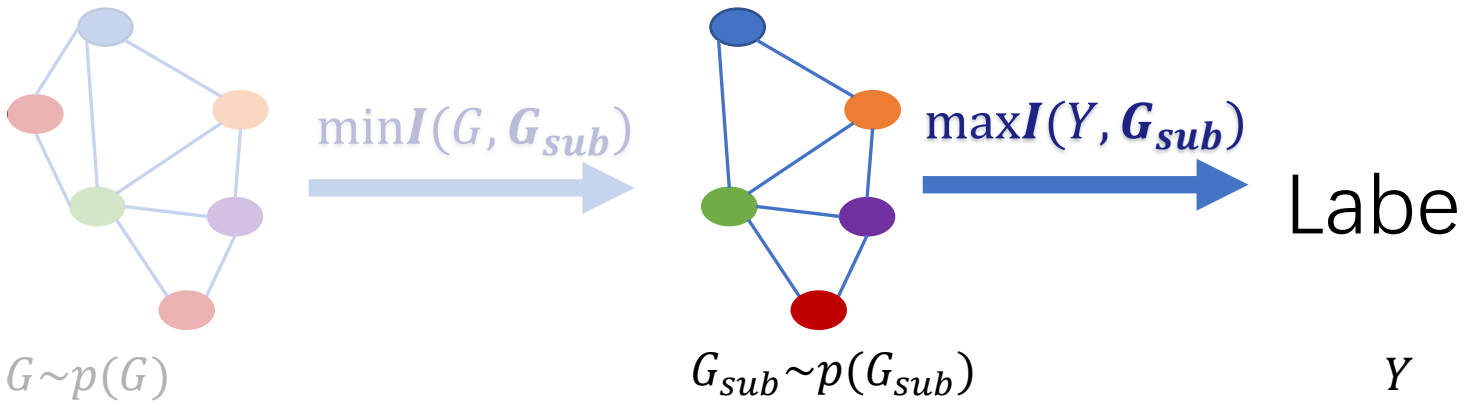
GIB: Mutual Information between Y and G_{sub}

$I(Y, G_{sub})$

- Maximize mutual information between the property and the sub-graph, $I(Y, G_{sub})$

$$I(Y, G_{sub}) \geq \frac{1}{N}\sum_{i=1}^{N} q_{\phi_1}(y|G_{sub})$$
$$=: -\mathcal{L}_{cls}(q_{\phi_1}(y|G_{sub}), y_{gt})$$

GIB objective: $\max_{G_{sub}} I(Y, G_{sub}) - \beta I(G, G_{sub})$



GIB: Mutual Information between G and G_{sub}

$I(G, G\_sub)$

- We fix G_sub to approximate $\mathcal{L}_{MI}$ by f_{ϕ_2}
- DONSKER-VARADHAN representation of the KL-divergence (Sampling)

$$I(G, G_{sub}) = \max_{\phi_2} \mathcal{L}_{MI}(\phi_2, G_{sub})$$

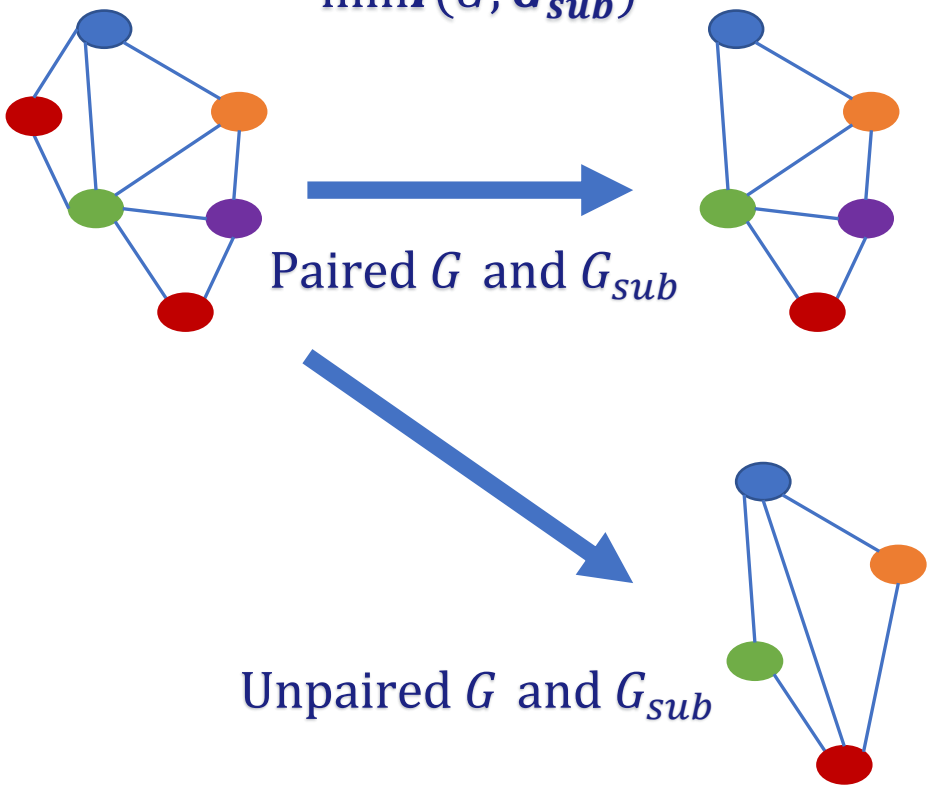
$$\approx \frac{1}{N}\sum_{i=1}^{N} f_{\phi_2}(G_i, G_{sub,i}) - \log \frac{1}{N}\sum_{i=1,j\neq i}^{N} e^{f_{\phi_2}(G_i, G_{sub,j})}$$

Paired G and G_{sub} in a batch

Unpaired G and G_{sub} in a batch

GIB objective: $\max_{G_{sub}} I(Y, G_{sub}) - \beta I(G, G_{sub})$

$\min \boldsymbol{I}(G, \boldsymbol{G_{sub}})$



Yu et al. Graph Information Bottleneck for Subgraph Recognition. ICLR 2021

GIB: Overall Objective Function

$$\min_{G_{sub}} -\boldsymbol{I}(Y, \boldsymbol{G_{sub}})$$

$$\min_{G_{sub},\phi_1} \mathcal{L}(G_{sub}, \phi_1, \phi_2^*) = \mathcal{L}_{cls}\left(q_{\phi_1}(y|G_{sub}), y_{gt}\right) + \beta \mathcal{L}_{MI}(\phi_2^*, G_{sub})$$

$$s.t. \quad \phi_2^* = \operatorname*{argmax}_{\phi_2} \mathcal{L}_{MI}(\phi_2, G_{sub}^*)$$

$$\min_{G_{sub}} \boldsymbol{I}(\boldsymbol{G}, \boldsymbol{G_{sub}})$$

$f_{\phi_2}(\cdot) \rightarrow I(\boldsymbol{G}, \boldsymbol{G_{sub}})$, inner loop

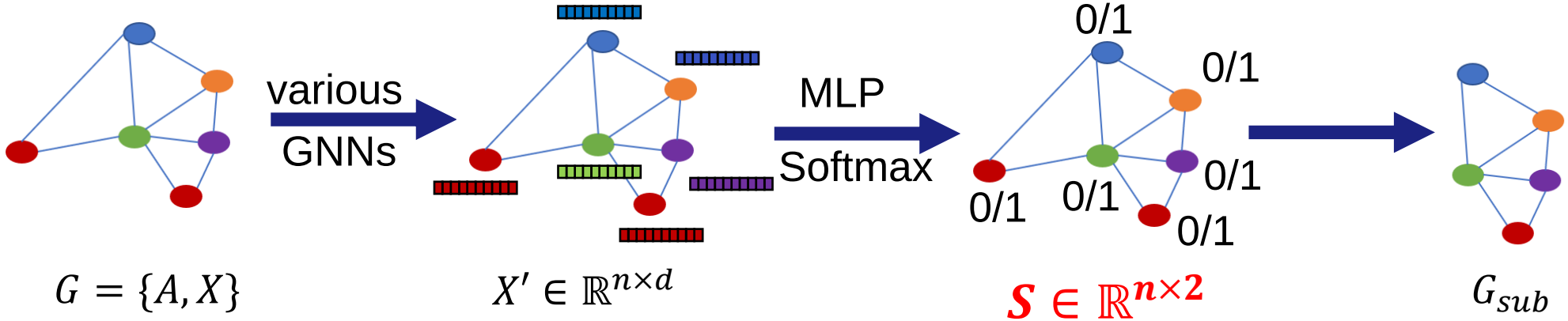
Problem

How to dynamically learn a $\boldsymbol{G_{sub}}$ for the graph, $\boldsymbol{G}$?

Yu et al. Graph Information Bottleneck for Subgraph Recognition. ICLR 2021

GIB: How to get G_{sub}

Subgraph Generator



Connectivity Loss

$$\mathcal{L}_{con} = ||Norm(\boldsymbol{S}^{\top}\boldsymbol{A}\boldsymbol{S}) - \boldsymbol{I}_2||_F$$

where $Norm$ denotes row-wise normalization. This loss regularize:

- Select **distinctive nodes**;
- Get **similar assignment** in neighborhoods;
- **Avoid trivial solutions** that all nodes are assigned to G_{sub} or $\overline{G_{sub}}$.

Yu et al. Graph Information Bottleneck for Subgraph Recognition. ICLR 2021

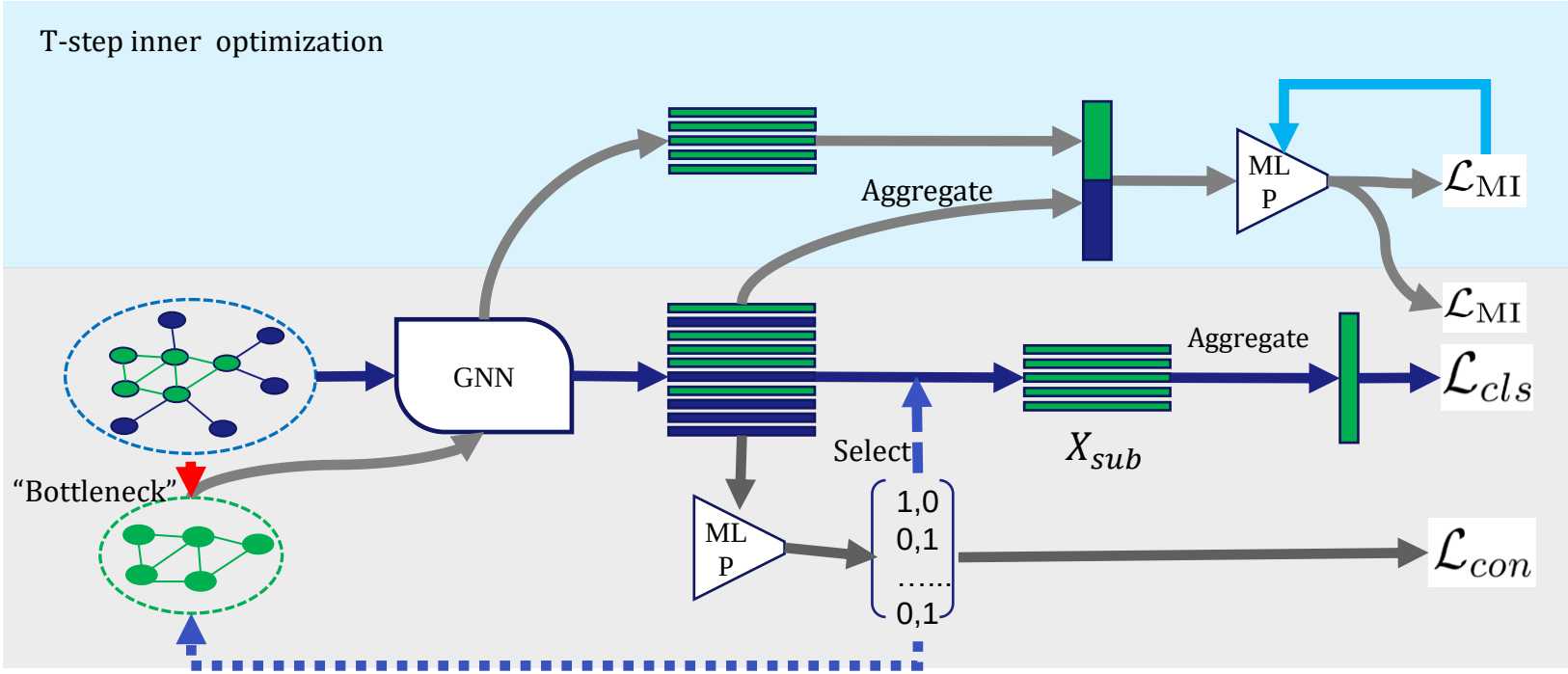
Graph Information Bottleneck

Objective Function

$$\min_{G_S,\phi_1} \mathcal{L}(G_S,\phi_1,\phi_2^*) = \mathcal{L}_{cls}\left(q_{\phi_1}(y|G_S), y_{gt}\right) + \beta\mathcal{L}_{MI}(\phi_2^*, G_S) + \mathcal{L}_{con}(S)$$

$$s.t. \quad \phi_2^* = \operatorname*{argmax}_{\phi_2} \mathcal{L}_{MI}(\phi_2, G_S^*)$$

Flowchart



Yu et al. Graph Information Bottleneck for Subgraph Recognition. ICLR 2021

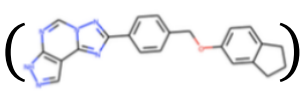
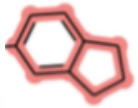
Improve Graph Classification

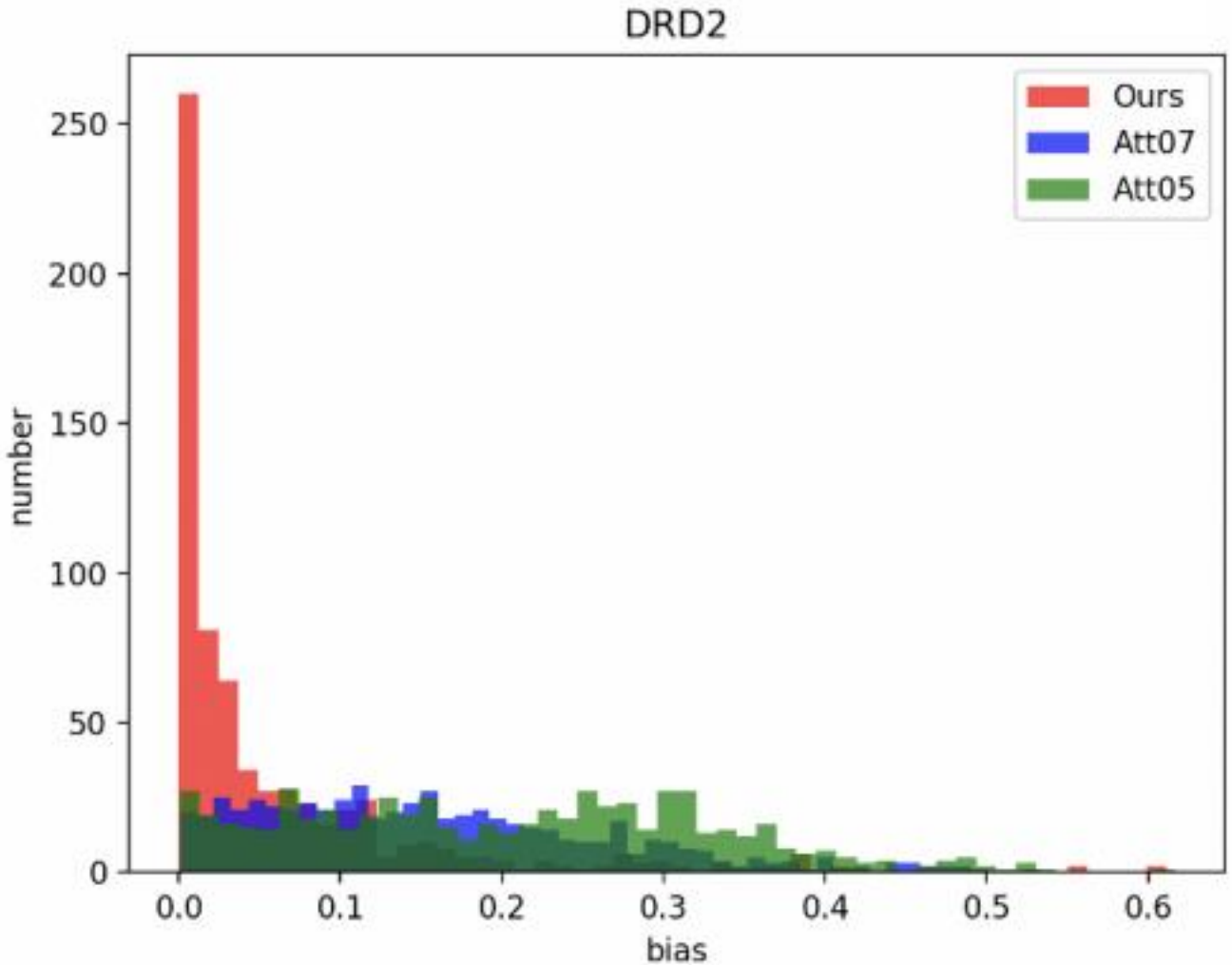
Method	MUTAG	PROTEINS	IMDB-BINARY	DD
SortPool	**0.844 ± 0.141**	0.747 ± 0.044	0.712 ± 0.047	0.732 ± 0.087
ASAPool	0.743 ± 0.077	0.721 ± 0.043	0.715 ± 0.044	0.717 ± 0.037
DiffPool	0.839 ± 0.097	0.727 ± 0.046	0.709 ± 0.053	0.778 ± 0.030
EdgePool	0.759 ± 0.077	0.723 ± 0.044	0.728 ± 0.044	0.736 ± 0.040
AttPool	0.721 ± 0.086	0.728 ± 0.041	0.722 ± 0.047	0.711 ± 0.055
GCN	0.743±0.110	0.719±0.041	0.707 ± 0.037	0.725 ± 0.046
GraphSAGE	0.743±0.077	0.721 ± 0.042	0.709 ± 0.041	0.729 ± 0.041
GIN	0.825±0.068	0.707 ± 0.056	0.732 ± 0.048	0.730 ± 0.033
GAT	0.738 ± 0.074	0.714 ± 0.040	0.713 ± 0.042	0.695 ± 0.045
GAT + DropEdge	0.743±0.081	0.711±0.043	0.710±0.041	0.717±0.035
GCN+GIB	0.776 ± 0.075	0.748 ± 0.046	0.722 ± 0.039	0.765 ± 0.050
GraphSAGE+GIB	0.760 ± 0.074	0.734 ± 0.043	0.719 ± 0.052	**0.781 ± 0.042**
GIN+GIB	0.839 ± 0.064	**0.749 ± 0.051**	**0.737 ± 0.070**	0.747 ± 0.039
GAT+GIB	0.749 ± 0.097	0.737 ± 0.044	0.729 ± 0.046	0.769 ± 0.040
GAT+GIB+DropEdge	0.754±0.085	0.737±0.037	0.731±0.003	0.776±0.034

Yu et al. Graph Information Bottleneck for Subgraph Recognition. ICLR 2021

195/33

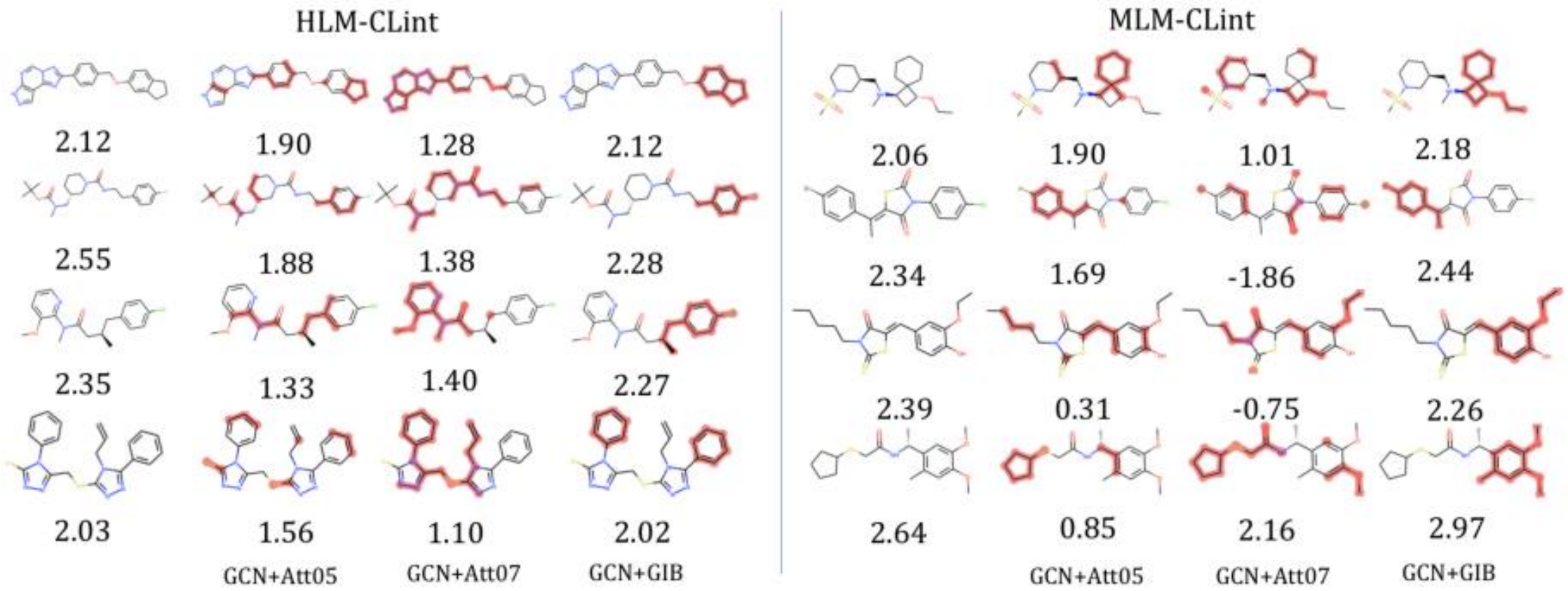
Consistent Prediction between Graph and Subgraph

$$\Delta_{pred} = |Pred(\text{  }) - Pred(\text{  })|$$



Yu et al. Graph Information Bottleneck for Subgraph Recognition. ICLR 2021

Interpretable Subgraphs

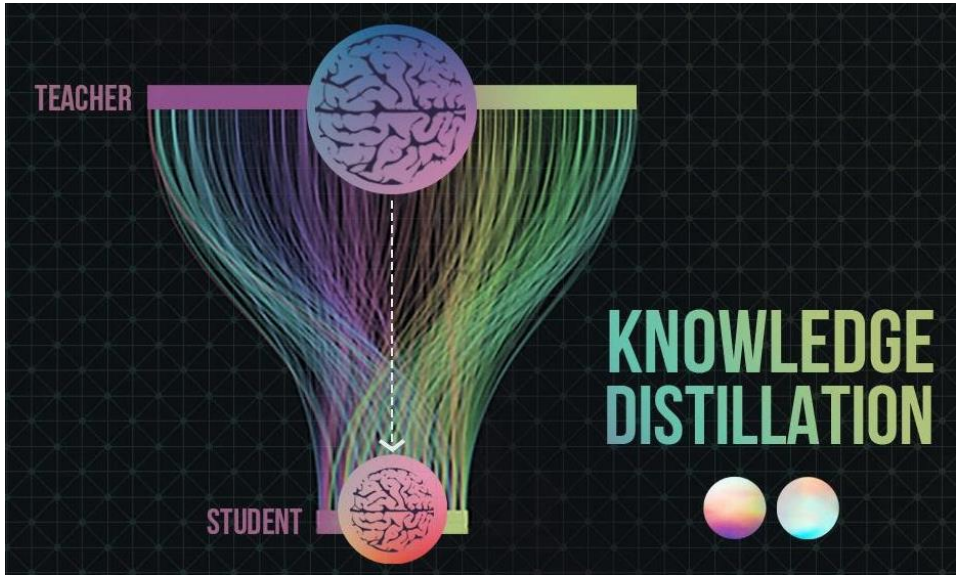


HLM-CLint & MLM-CLint: vitro human and mouse liver microsome metabolic stability (base 10 logrithm of mL/min/g)

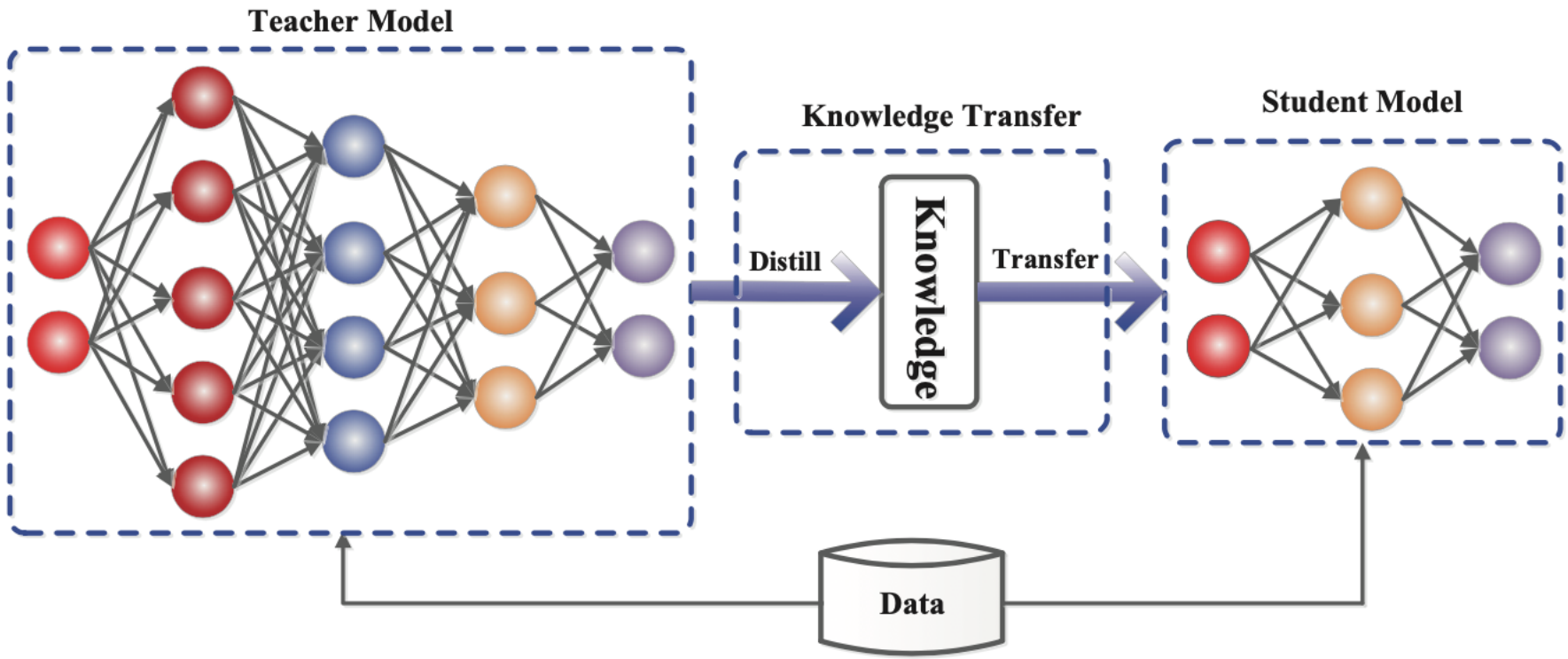
Yu et al. Graph Information Bottleneck for Subgraph Recognition. ICLR 2021

Other Advanced Topics of GNNs

Knowledge Distillation for GNNs



Knowledge Distillation on CNNs

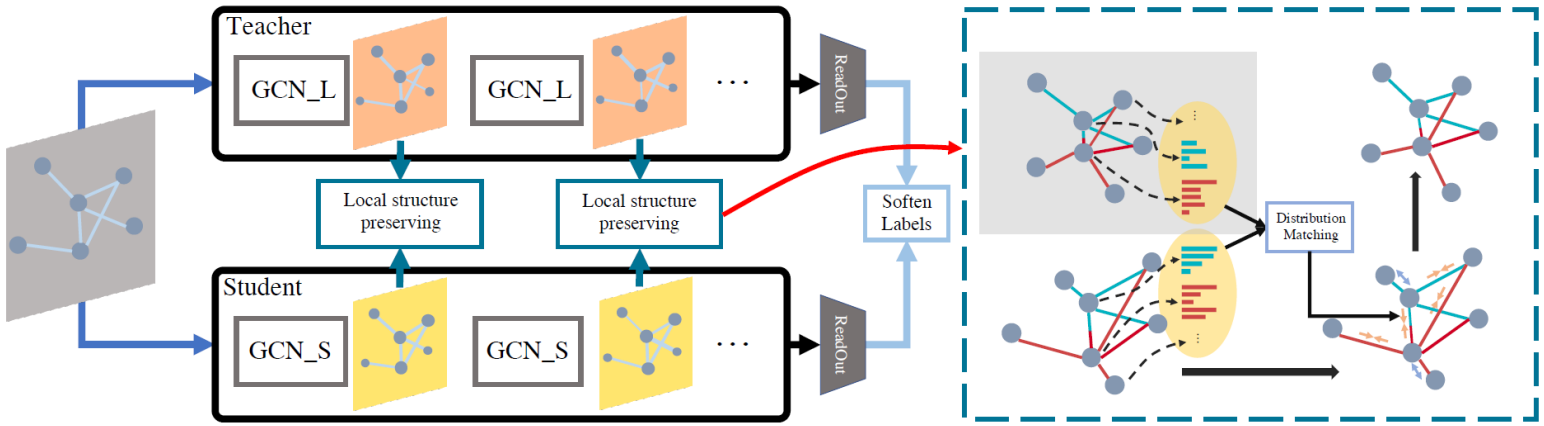


Knowledge Distillation (KD) has demonstrated its effectiveness on improving compact and lightweight Convolutional Neural Networks (CNNs).

Hinton, Geoffrey et al. "Distilling the knowledge in a neural network." arXiv preprint arXiv:1503.02531 (2015)
Gou, Jianping, et al. "Knowledge distillation: A survey." International Journal of Computer Vision (2021): 1-31.

Teacher-Student Knowledge Distillation on GNNs

Recently, Yang et al. proposed to combine KD with the training of Graph Neural Networks (GNNs).



However, the conventional teacher-student knowledge distillation framework leads to two major concerns:

1. Training inefficiency
2. Sophisticated design of a qualified teacher model

Yuan et al. Revisiting Knowledge Distillation via Label Smoothing Regularization CVPR 2020
Yang et al. Distilling Knowledge from Graph Convolutional Networks. CVPR 2020

Why Self-distilling Graph Neural Networks?

- Knowledge Distillation on GNNs → Self-Distillation on a single GNN

  (1) Training Inefficiency;

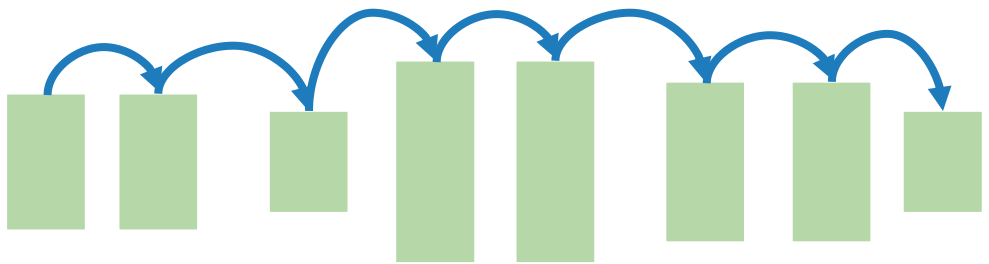
  (2) Difficulty on searching for a qualified teacher

  →

  (1) Marginal cost increase;

  (2) Perform knowledge extraction and transfer among layers

- How to conduct self KD in GNNs?
  - Nodes tend to lose information when GNNs go deeper
  - can distill some knowledge from **shallow** GNN layers!



Chen, Yuzhao et al. "On Self-distilling Graph Neural Networks." arXiv, 2020

Knowledge Definition: Neighborhood Discrepancy Rate (NDR)

Naïve solution: define edge-wise difference: $\left\|\mathbf{X}_{v\cdot}^{(l)} - \mathbf{X}_{c\cdot}^{(l)}\right\|_p$

Proposition

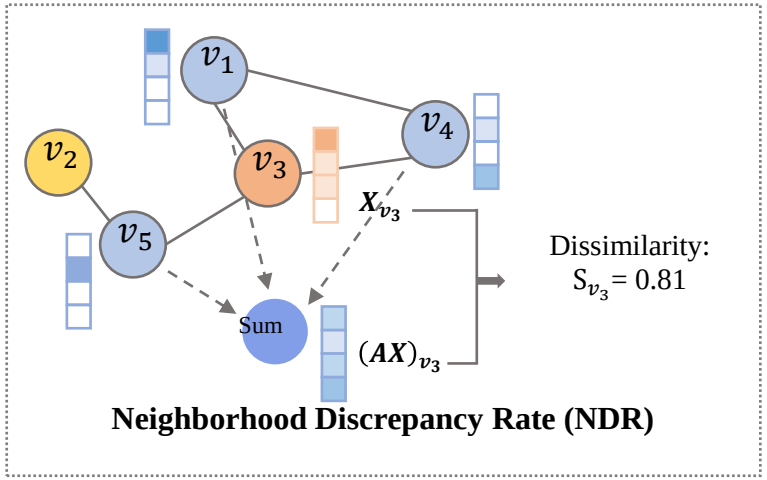
$$\forall \epsilon > 0,\ \left\|\mathbf{X}_{v\cdot}^{(l)} - (\mathbf{D}^{-1}\mathbf{A}\mathbf{X})_{v\cdot}^{(l)}\right\|_p < \sum_{c\in \mathrm{N}(v)} \left\|\mathbf{X}_{v\cdot}^{(l)} - \mathbf{X}_{c\cdot}^{(l)}\right\|_p < \epsilon, \text{ when } l \to \infty$$

- Knowledge Definition
  - Neighborhood Discrepancy Rate (NDR) of node v:

$$S_v^{(l)} = 1 - \frac{\mathbf{X}_{v\cdot}^{(l)}\left(\mathbf{A}\mathbf{X}^{(l)}\right)_{v\cdot}^{\mathrm{T}}}{\left\|\mathbf{X}_{v\cdot}^{(l)}\right\|_2 \left\|\left(\mathbf{A}\mathbf{X}^{(l)}\right)_{v\cdot}\right\|_2}$$

  - NDR vector of the graph:

$$\mathbf{s}^{(l)} = (S_1^{(l)}, \ldots, S_N^{(l)})$$

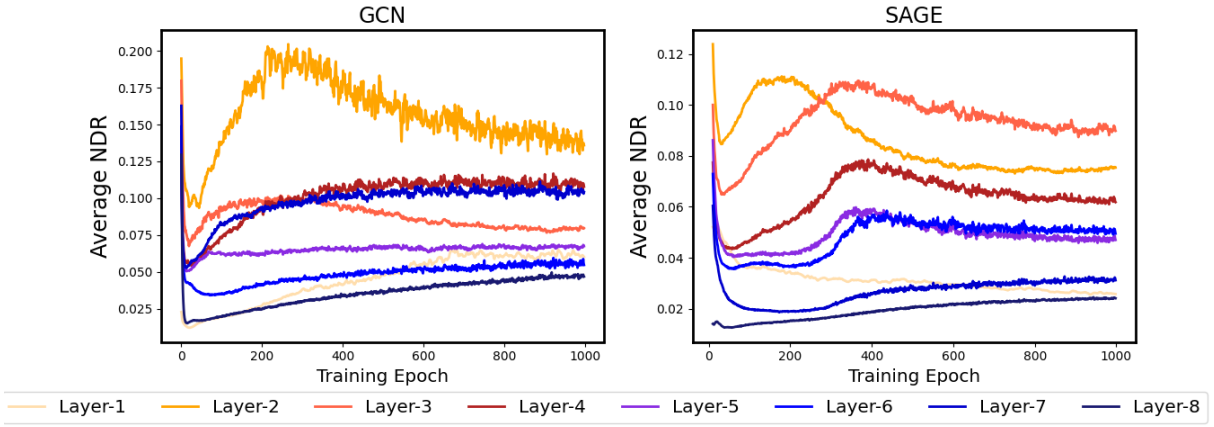


Chen, Yuzhao et al. "On Self-distilling Graph Neural Networks." arXiv, 2020

202

Adaptive Distilling Strategy: Adaptive Discrepancy Retaining (ADR)

- Adaptive Discrepancy Retaining (ADR)
  - Progressive (layer-by-layer) transfer.
  - Adaptively select the initial supervision target.
    - $l^* = \underset{k\in[L-1]}{\operatorname{argmax}} \|\mathbf{s}^{(k)}\|$
  - Explicit teacher selection.
  - Regularizer formulation:



$$\mathcal{L}_{\mathrm{N}} = \sum_{l=l^*,\ldots,L-1} 1(\|\mathbf{s}^{(l)}\| > \|\mathbf{s}^{(l+1)}\|) d^2(\mathbf{s}^{(l+1)}, \mathbf{s}^{(l)}), \qquad d^2(\mathbf{s}^{(l+1)}, \mathbf{s}^{(l)}) = \|\mathbf{s}^{(l+1)} - \mathrm{StopGradient}(\mathbf{s}^{(l)})\|_2^2$$

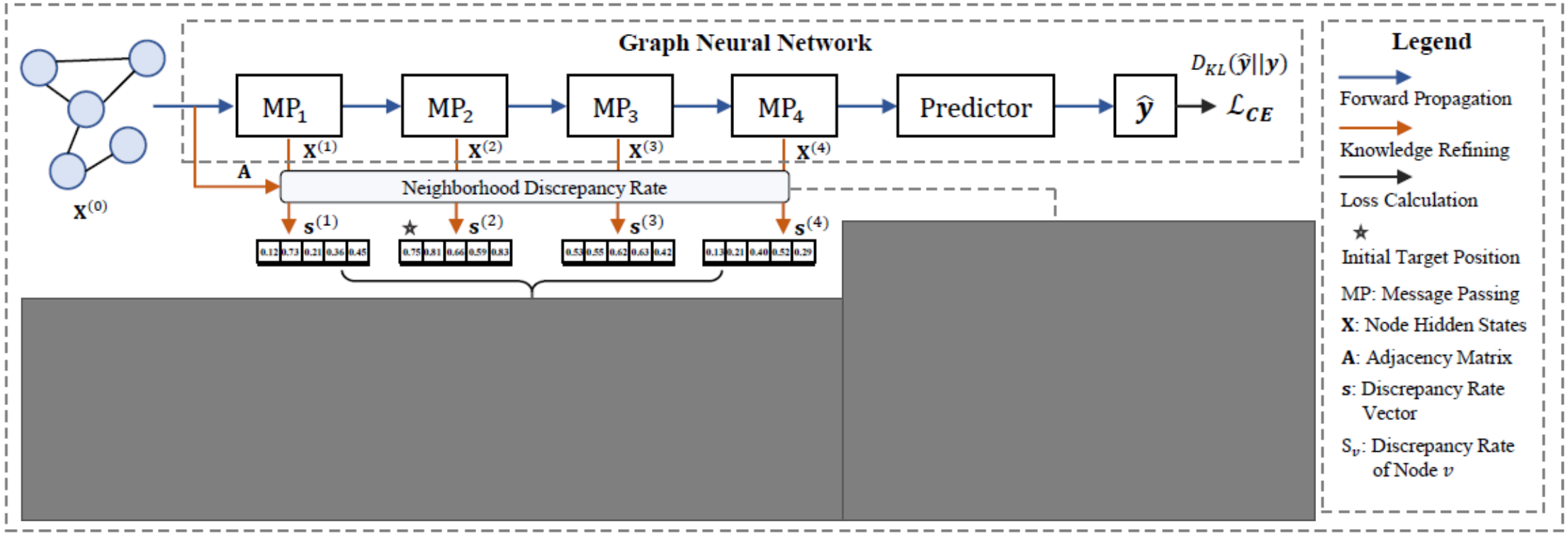
- Generic GNN Self-Distillation Framework by considering various granularities
  - knowledge over arbitrary subgraph
  - knowledge over the entire graph

Chen, Yuzhao et al. "On Self-distilling Graph Neural Networks." arXiv, 2020

203

Overall GNN Self-KD Framework

- (Teacher-free) Self-Distillation on GNNs
  - (1) Marginal cost increase; (2) Perform knowledge extraction and transfer among layers

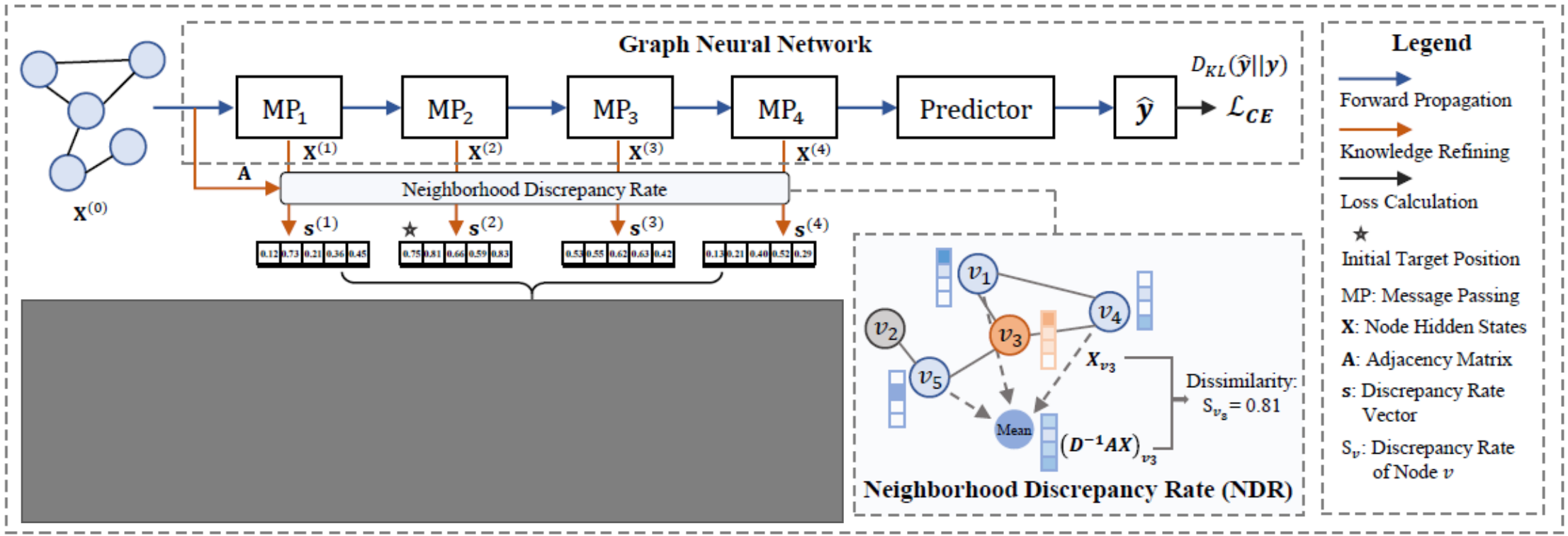


Chen, Yuzhao et al. “On Self-distilling Graph Neural Networks.” arXiv, 2020

204

Overall GNN Self-KD Framework

- (Teacher-free) Self-Distillation on GNNs
  - (1) Marginal cost increase; (2) Perform knowledge extraction and transfer among layers

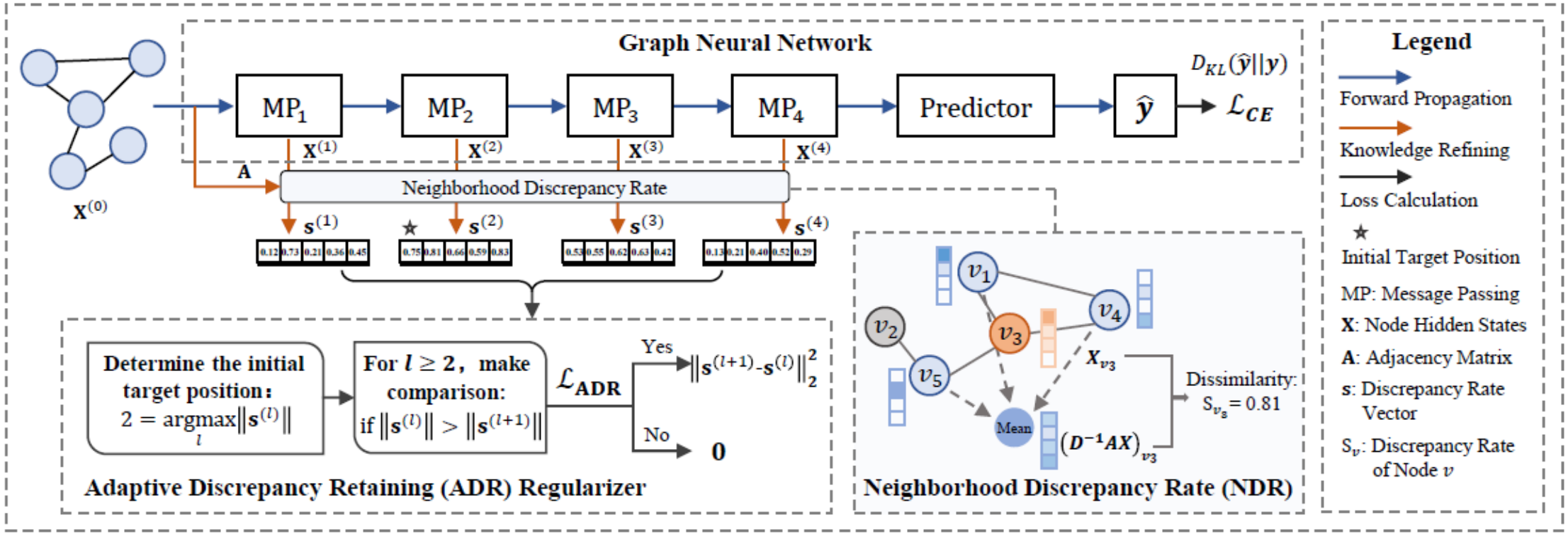


Chen, Yuzhao et al. “On Self-distilling Graph Neural Networks.” arXiv, 2020

205

Overall GNN Self-KD Framework

- (Teacher-free) Self-Distillation on GNNs
  - (1) Marginal cost increase; (2) Perform knowledge extraction and transfer among layers



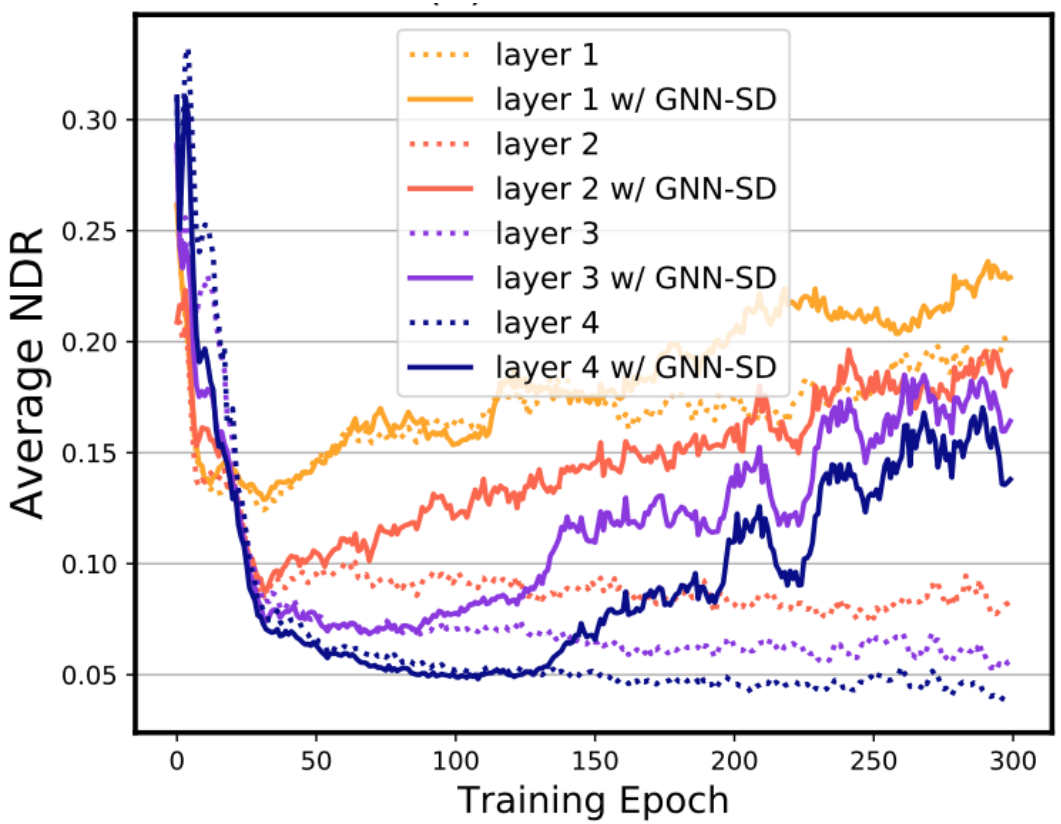
Chen, Yuzhao et al. “On Self-distilling Graph Neural Networks.” arXiv, 2020

206

The Effect of the NDR Regularizer

NDR between training w/ (solid line) and w/o (dotted) the GNN-SD strategy

higher better



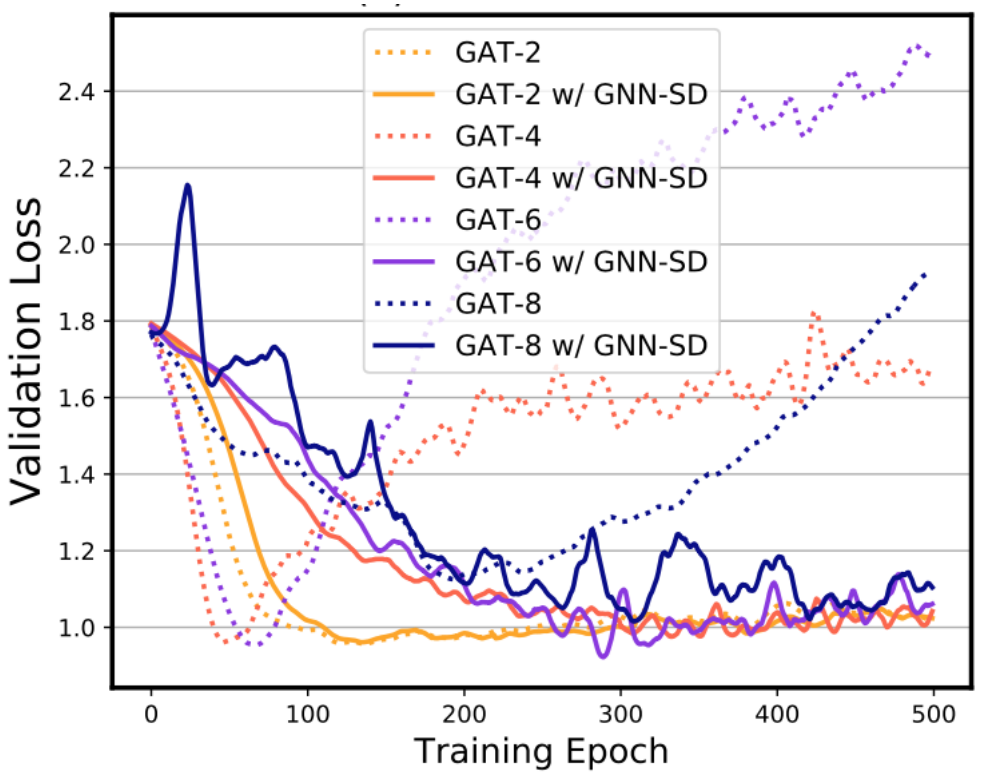
The self-distillation strategy could improve the NDR for different layers

Chen, Yuzhao et al. "On Self-distilling Graph Neural Networks." arXiv, 2020

GNN-SD Improves Generalization Performance

GAT-n: GAT with n layers
dotted lines: w/o GNN-SD
solid lines: w GNN-SD

lower better



The self-distillation strategy constantly improves the generalization performance by achieving lower validation losses

Chen, Yuzhao et al. "On Self-distilling Graph Neural Networks." arXiv, 2020

Applications

GNN in Social Networks

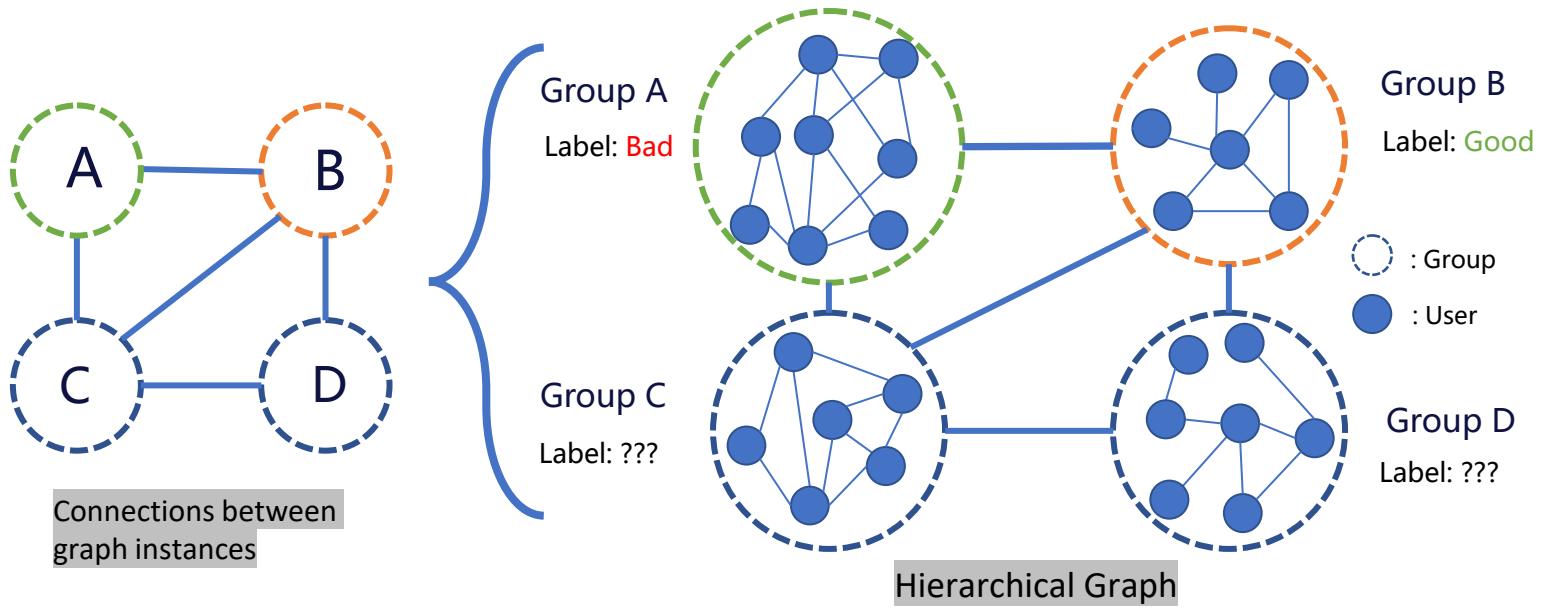
GNN in Social Networks

- "Semi-supervised graph classification: A hierarchical graph perspective." **WWW 2019**

- "Inverse Graph Identification: Can We Identify Node Labels Given Graph Labels?" **arXiv 2020**

Hierarchical Graph Classification

- **Hierarchical Graph:** A set of graph instances are interconnected via edges.
  - Social network with group structure.
  - Document (graph-of-words) collection with citation relation.

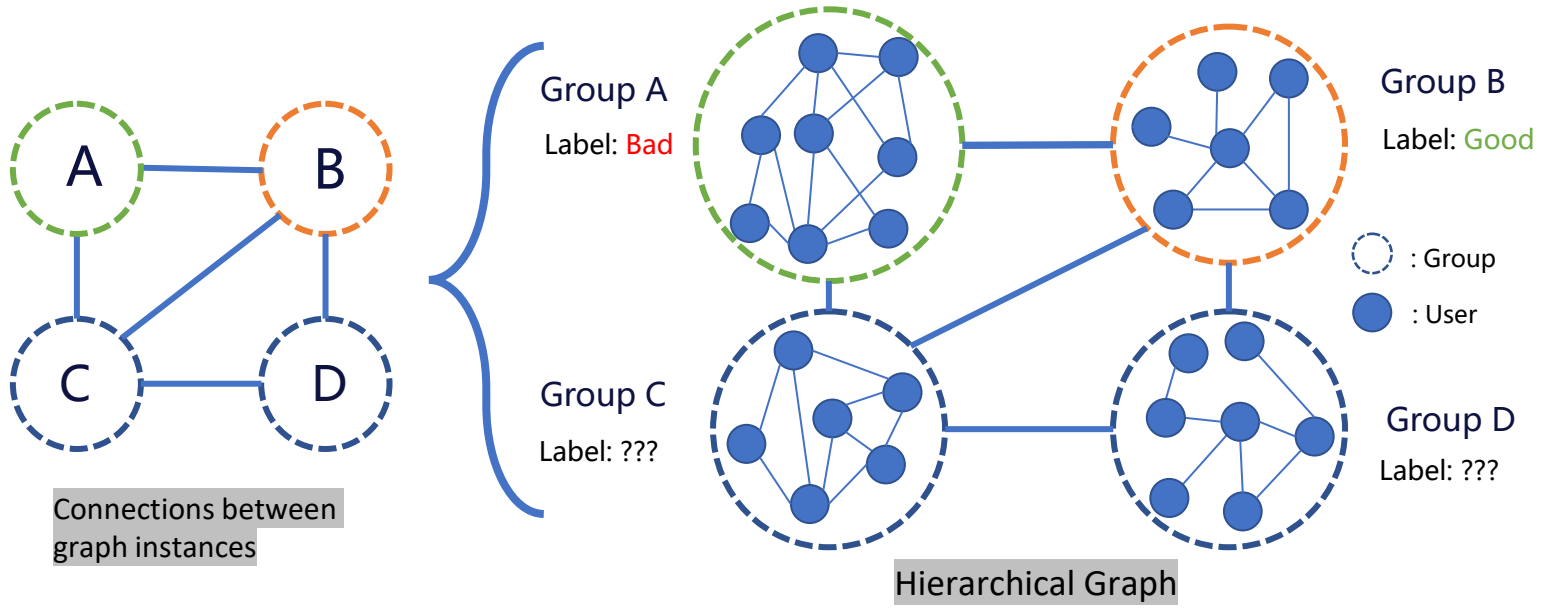


- **The Problem:** predicts the class label of graph instances in a hierarchical graph.

Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective."

Hierarchical Graph Classification

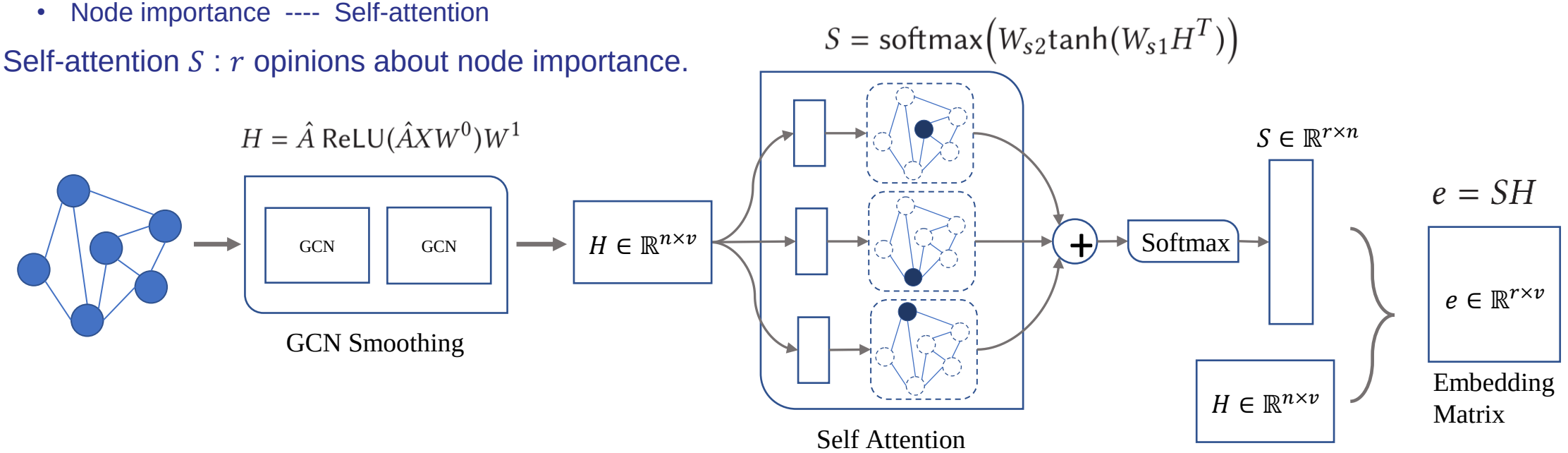
- **The Problem:** predicts the class label of graph instances in a hierarchical graph.
- **Challenges:**
  - How to represent the graphs with arbitrary size into a fixed-length vector?
  - How to incorporate the information of instance level and hierarchical level?



Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective."

Graph Instance Level: Self-Attentive Graph Embedding

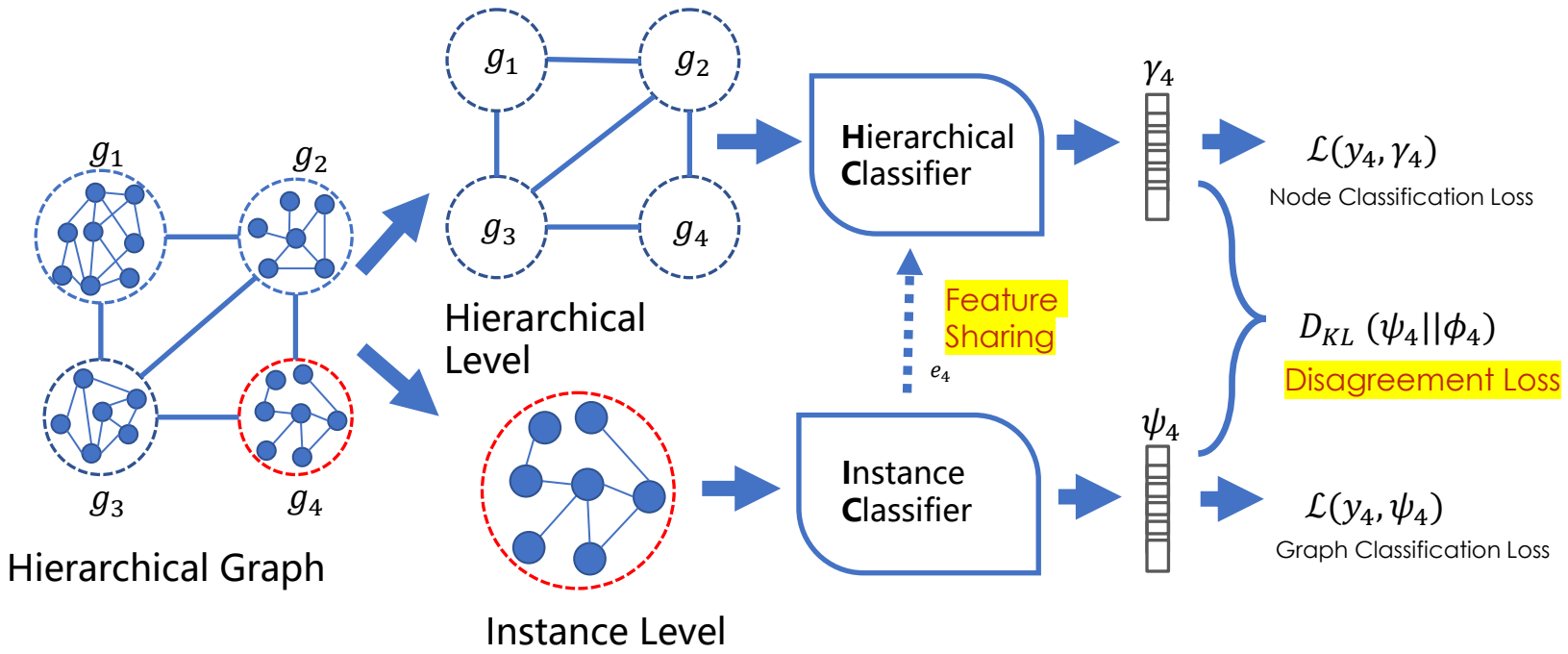
- How to represent the graphs with arbitrary size into a fixed-length vector?
- Graph representation learning at different level:
  - Node Level: $G(V,E) \rightarrow H^{n\times v}$
  - Graph Level: $G(V,E) \rightarrow e^{v}$
- **SAGE: Self-Attentive Graph Embedding**
  - Size invariance ---- Self-attention
  - Permutation invariance ---- GCN Smoothing
  - Node importance ---- Self-attention
- Self-attention S : r opinions about node importance.



Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective."

The Unified Model

- How to incorporate the information of instance level and hierarchical level?
  - **Instance Level Model :** Graph Level Learning (SEGA)
  - **Hierarchical Level Model:** Node Level Learning (GCN)
- **Feature Sharing:** Concatenate the output of SEGA to the input of GCN.
- **Disagreement Loss:** The disagreement between instance classifier and hierarchical classifier should be minimized.



The overall loss: $\min \zeta(G_l) + \xi(G_u)$,

The supervised loss:

$$\zeta(G_l) = \sum_{g_i \in G_l} (\mathcal{L}(y_i, \psi_i) + \mathcal{L}(y_i, \gamma_i)),$$

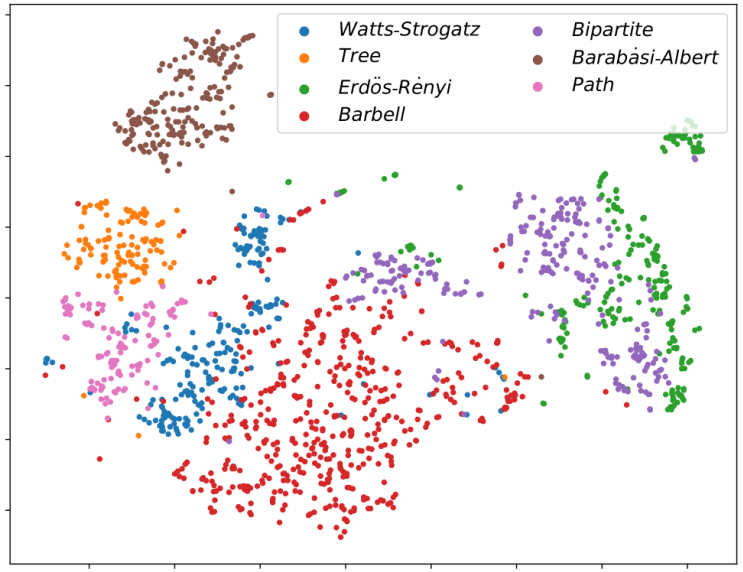
The disagreement loss:

$$\xi(G_u) = \sum_{g_i \in G_u} D_{KL}(\gamma_i || \psi_i),$$

Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective."

Performance of SAGE

- Performance of synthesized graphs



Two-dimensional visualization of graph embeddings generated from the synthesized graph instances using SAGE.

- Performance on the protein classification task:

Table 1: Statistics of PROTEINS and D&D

	PROTEINS	D&D
Max number of nodes	620	5748
Avg number of nodes	39.06	284.32
Number of graphs	1113	1178

Table 2: Accuracy of different classifiers

Approach	PROTEINS	D&D
SP	75.07±0.54%	-
RW	74.22±0.42%	-
GK	71.67±0.55%	78.45±0.26%
WL	72.92±0.56%	77.95±0.70%
PSCN	75.89±2.76%	77.12±2.41%
graph2vec	73.30±2.05%	-
SAGE	**77.26**±2.28%	**80.88**±2.33%

Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective."

GNN in Social Networks

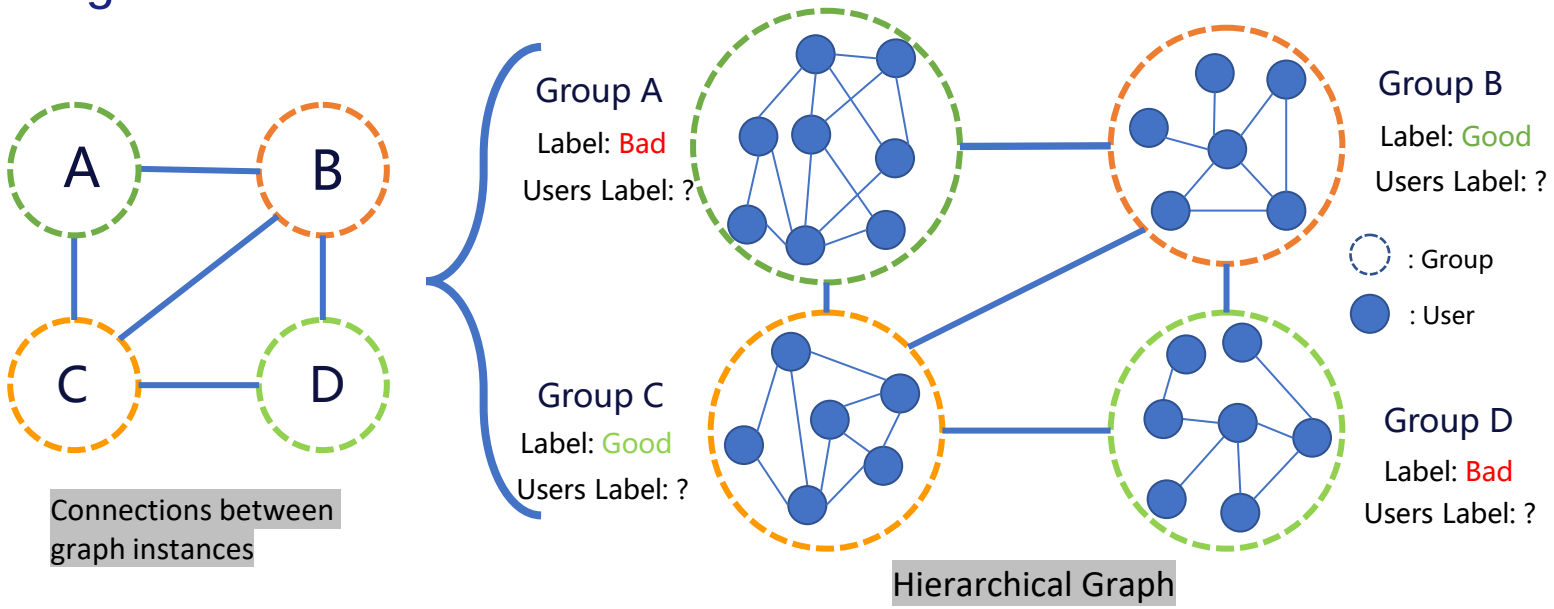
- "Semi-supervised graph classification: A hierarchical graph perspective." **WWW 2019**

- "Inverse Graph Identification: Can We Identify Node Labels Given Graph Labels?" **arXiv 2020**

Inverse Graph Identification (IGI)

Inverse Graph Identification (IGI):

- **The problem:** infers the information or properties of nodes or even edges and sub-graphs in each graph instance.
- A node clustering case of IGI:

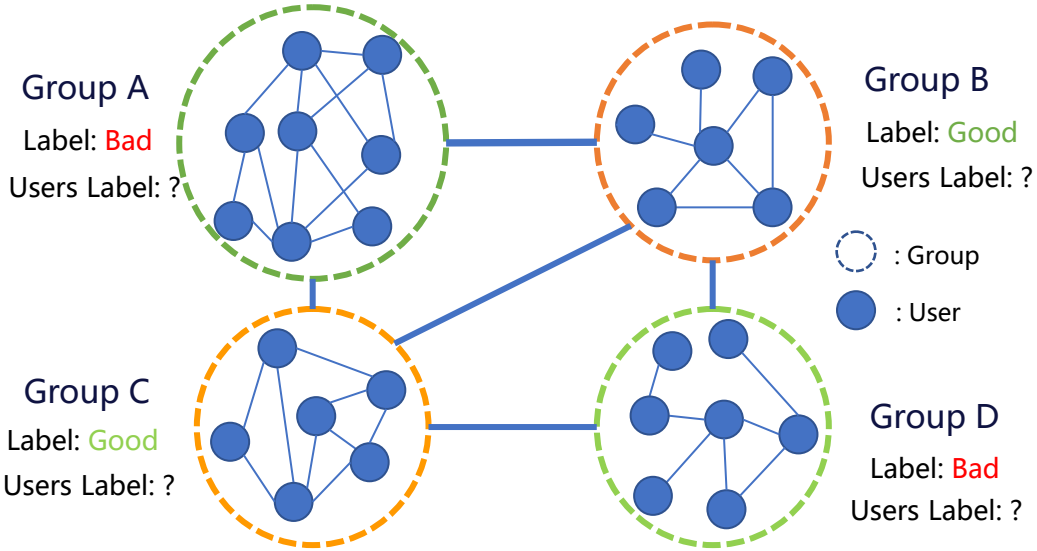


- E.g., social media, point cloud segmentation, etc.

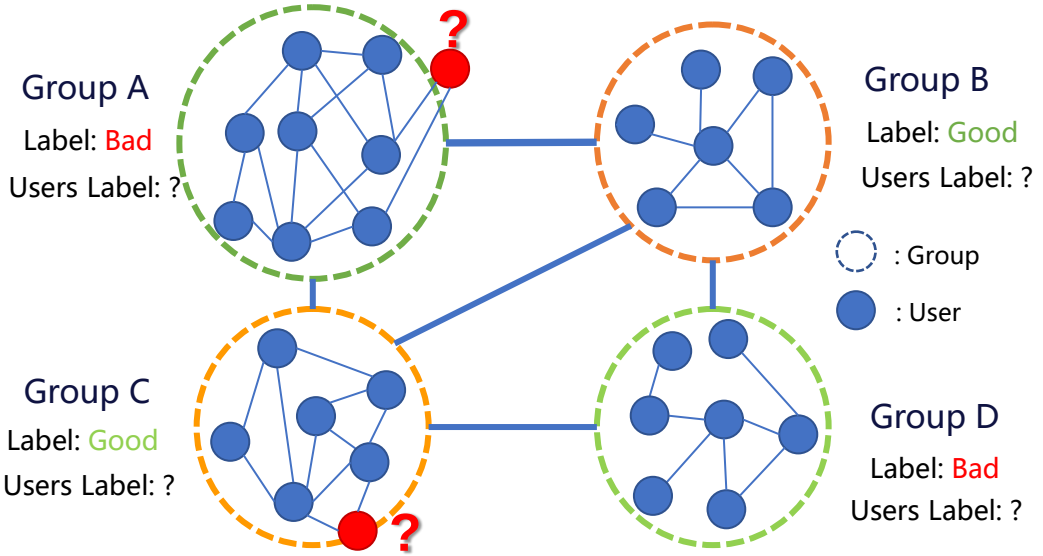
Bian, Tian, et al. "Inverse Graph Identification: Can We Identify Node Labels Given Graph Labels?"

IGI: Different Cases

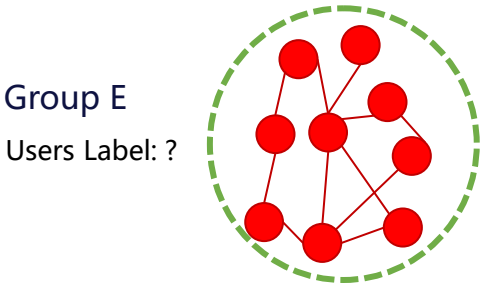
Case 1: Infer existing node labels



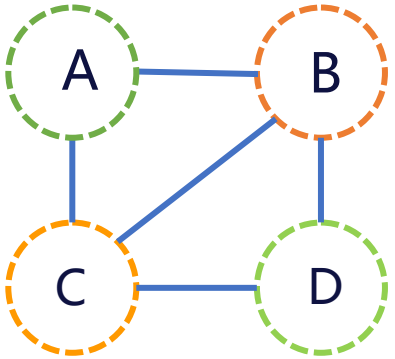
Case 2: Infer new node labels



Case 3: Infer node labels of new graphs



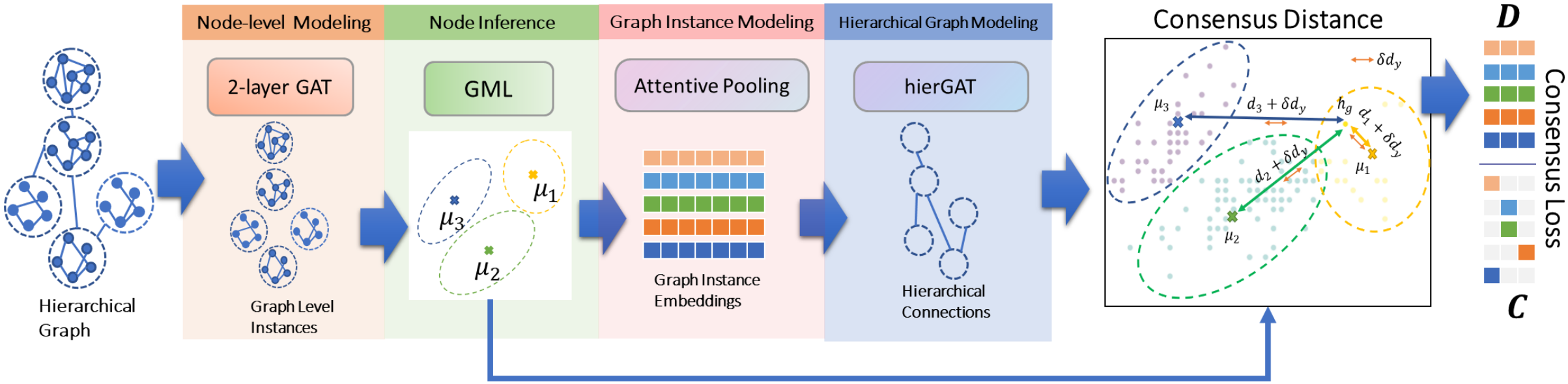
Case 4: With hierarchical graph



Bian, Tian, et al. "Inverse Graph Identification: Can We Identify Node Labels Given Graph Labels?"

IGI: Inverse Graph Identification on Hierarchical Graph

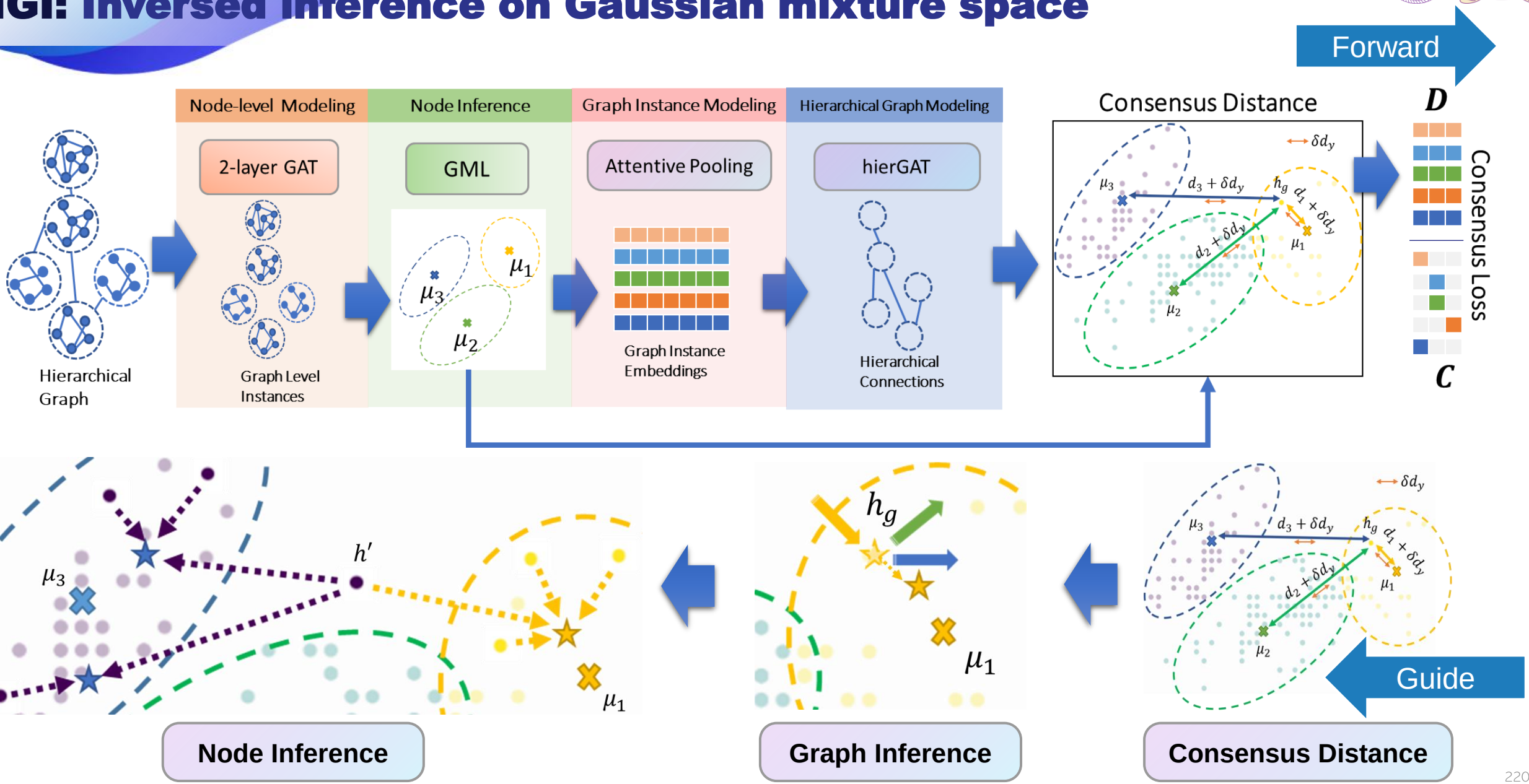
Gaussian Mixture Graph Convolutional Networks (GMGCN):



- **Gaussian Mixture Layer (GML):** $h' = \frac{1}{C}\sum_{c=1}^{C} \exp(-\frac{1}{2\sigma_c^2}(h-\mu_c)^2)$
- **Consensus Distance:** $\boldsymbol{d}_n = \left[\left\|h_g^n - \mu_1\right\|_2, \left\|h_g^n - \mu_2\right\|_2, \ldots, \left\|h_g^n - \mu_c\right\|_2\right] \in \mathbb{R}^c$
- **Consensus Loss:** $\mathcal{L}_{con} = \text{cross\_entropy}_n\left(\text{softmax}(-\boldsymbol{d}_n - \delta \times \text{onehot}(c_n) \odot \boldsymbol{d}_n), c_n\right)$
- **Node Inference:** $z_i = arg\max_c similarity(h'_i, \mu_c)$

Bian, Tian, et al. "Inverse Graph Identification: Can We Identify Node Labels Given Graph Labels?"

IGI: Inversed inference on Gaussian mixture space



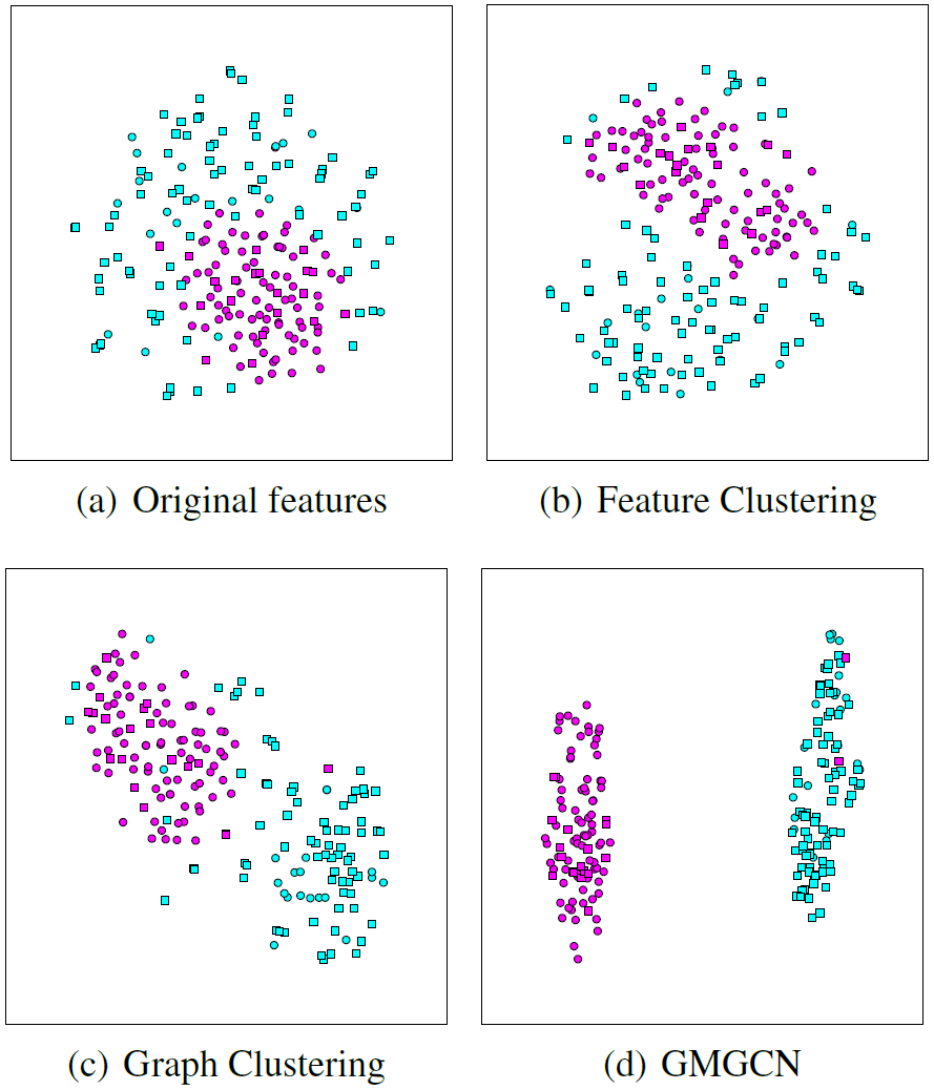
Bian, Tian, et al. "Inverse Graph Identification: Can We Identify Node Labels Given Graph Labels?"

IGI: Inverse Graph Identification on Hierarchical Graph

- Performance of synthesized graphs

methods	single graph					
	orignal nodes		new nodes		new graphs	
	NMI	ARI	NMI	ARI	NMI	ARI
Feature Clustering	49.46(1.89)	50.78(2.48)	50.11(2.07)	49.33(2.56)	47.48(2.48)	48.03(3.32)
Graph Clustering	80.65(8.18)	85.51(7.96)	85.85(8.47)	87.72(8.52)	78.72(10.47)	84.25(10.33)
MIL	76.47(0.00)	81.39(0.00)	75.42(0.00)	80.76(0.00)	75.77(0.00)	80.63(0.00)
ATTGCN	14.25(9.76)	12.45(11.64)	19.69(9.54)	13.44(11.38)	13.25(9.60)	12.22(11.10)
GMGCN	**92.26(0.22)**	**95.71(0.13)**	**94.70(0.33)**	**96.08(0.26)**	**92.98(0.24)**	**96.16(0.16)**

methods	hierarchical graph					
	orignal nodes		new nodes		new graphs	
	NMI	ARI	NMI	ARI	NMI	ARI
ATTGCN	19.09(12.34)	17.48(13.52)	24.64(12.34)	19.09(13.23)	20.54(11.92)	20.00(13.37)
GMGCN	**92.88(0.31)**	**96.05(0.17)**	**95.16(0.33)**	**96.42(0.25)**	**96.36(0.29)**	**96.39(0.16)**



(a) Original features (b) Feature Clustering

(c) Graph Clustering (d) GMGCN

Bian, Tian, et al. "Inverse Graph Identification: Can We Identify Node Labels Given Graph Labels?"

IGI: Inverse Graph Identification on Hierarchical Graph

- Performance of social media datasets (PHEME)

	Charlie Hebdo	Ferguson	Germanwings Crash	Ottawa Shooting	Sydney Siege
Method	NMI	NMI	NMI	NMI	NMI
Feature Clustering	25.46(0.00)	24.56(0.00)	48.84(0.00)	**34.56(0.00)**	19.45(0.00)
Graph Clustering	3.26(2.26)	1.08(0.71)	0.82(0.40)	4.49(3.07)	4.93(1.92)
MIL	5.69(0.00)	4.08(0.00)	0.61(0.00)	0.60(0.00)	19.68(0.00)
Feature Clustering+feature augmentation	25.42(0.00)	24.57(0.00)	48.84(0.00)	34.33(0.00)	19.21(0.00)
Graph Clustering+feature augmentation	6.08(3.06)	3.45(2.90)	39.14(11.78)	6.22(4.15)	6.83(4.28)
MIL+feature augmentation	22.32(0.00)	0.14(0.00)	0.28(0.00)	0.11(0.00)	19.98(0.00)
GMGCN	**47.51(3.27)**	**48.35(4.08)**	**48.85(2.14)**	32.58(3.63)	**41.00(3.93)**
Method	ARI	ARI	ARI	ARI	ARI
Feature Clustering	23.46(0.00)	23.23(0.00)	47.29(0.00)	32.08(0.00)	16.45(0.00)
Graph Clustering	6.56(6.30)	4.04(2.24)	3.71(1.93)	10.66(7.68)	11.78(5.57)
MIL	9.38(0.00)	13.93(0.00)	1.10(0.00)	2.09(0.00)	**39.51(0.00)**
Feature Clustering+feature augmentation	23.41(0.00)	23.23(0.00)	48.29(0.00)	31.85(0.00)	16.20(0.00)
Graph Clustering+feature augmentation	10.11(6.02)	9.15(6.51)	10.12(4.47)	8.91(7.99)	12.65(6.99)
MIL+feature augmentation	42.51(0.00)	1.08(0.00)	0.04(0.00)	0.04(0.00)	33.71(0.00)
GMGCN	**52.38(3.15)**	**55.26(3.21)**	**54.95(1.02)**	**37.51(2.23)**	37.79(2.18)

Bian, Tian, et al. "Inverse Graph Identification: Can We Identify Node Labels Given Graph Labels?"

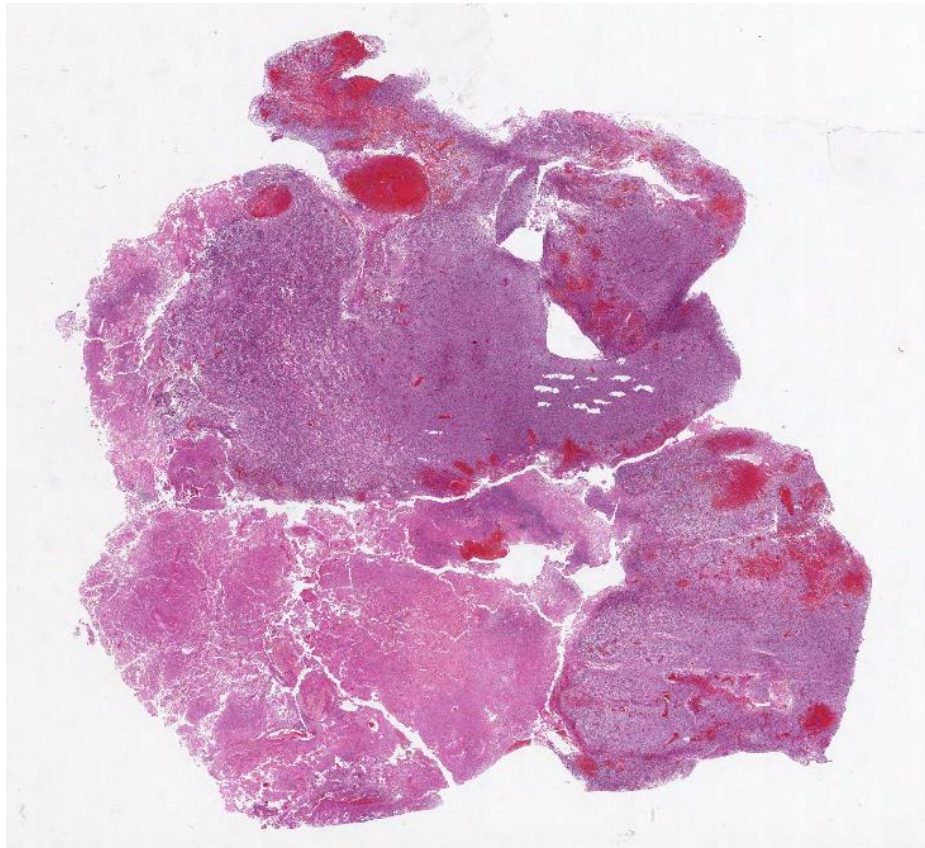
Applications

GNN in Medical Imaging

GNN in Medical Imaging

- "Graph CNN for Survival Analysis on Whole Slide Pathological Images", **MICCAI 2018**
- "Graph Convolutional Nets for Tool Presence Detection in Surgical Videos", **IPMI 2019**
- "Graph Attention Multi-instance Learning for Accurate Colorectal Cancer Staging", **MICCAI 2020**

- **Survival Prediction**
  - Predict the risk of a certain event occurs.
  - Event: part failure, drug adverse reaction or death.
  - Application: provides suggestion for clinical inte
- **Whole Slide Images**
  - Large: single WSI size >0.5 GB.
  - Complicated: millions of cells.
  - Combine local and global features.



- **Cox** $\lambda(t|X_i) = \lambda_0(t)\exp(\beta_1 X_{i1} + \cdots + \beta_p X_{ip}) = \lambda_0(t)\exp(X_i \cdot \beta)$

- **Partial likelihood for event happens on subject *i* :**

$$L_i(\beta) = \frac{\lambda(Y_i|X_i)}{\sum_{j:Y_j \geq Y_i} \lambda(Y_i|X_j)} = \frac{\cancel{\lambda_0(Y_i)}\theta_i}{\sum_{j:Y_j \geq Y_i} \cancel{\lambda_0(Y_i)}\theta_j} = \frac{\theta_i}{\sum_{j:Y_j \geq Y_i} \theta_j}$$

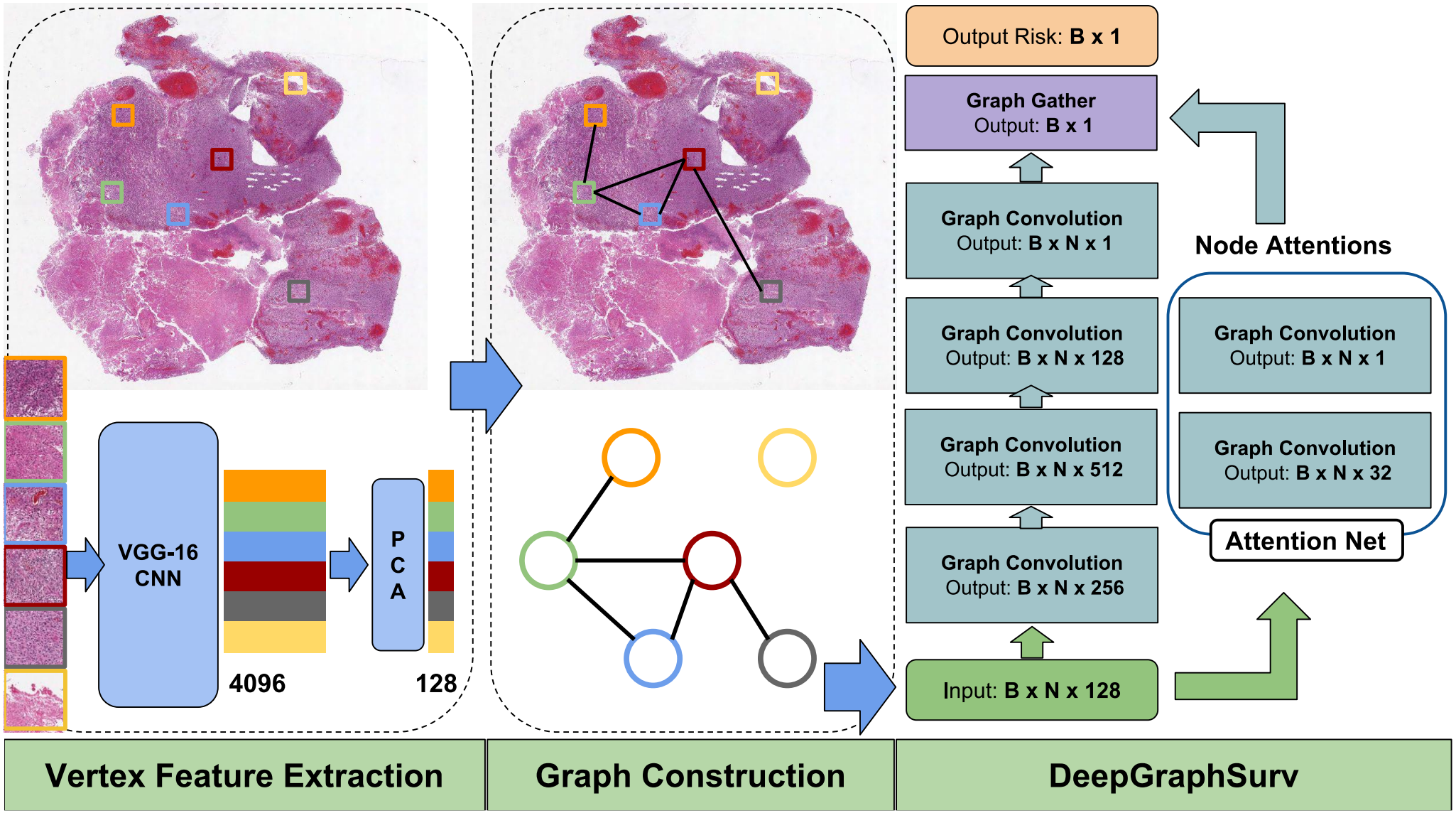
$$\theta_j = \exp(X_j \cdot \beta)$$

where, Y is the observation time**.**

- **Join likelihood of all subjects:** $L(\beta) = \prod_{i:C_i=1} L_i(\beta)$

- **Log likelihood as object function:**

$$\ell(\beta) = \sum_{i:C_i=1} \left(X_i \cdot \beta - \log \sum_{j:Y_j \geq Y_i} \theta_j \right)$$

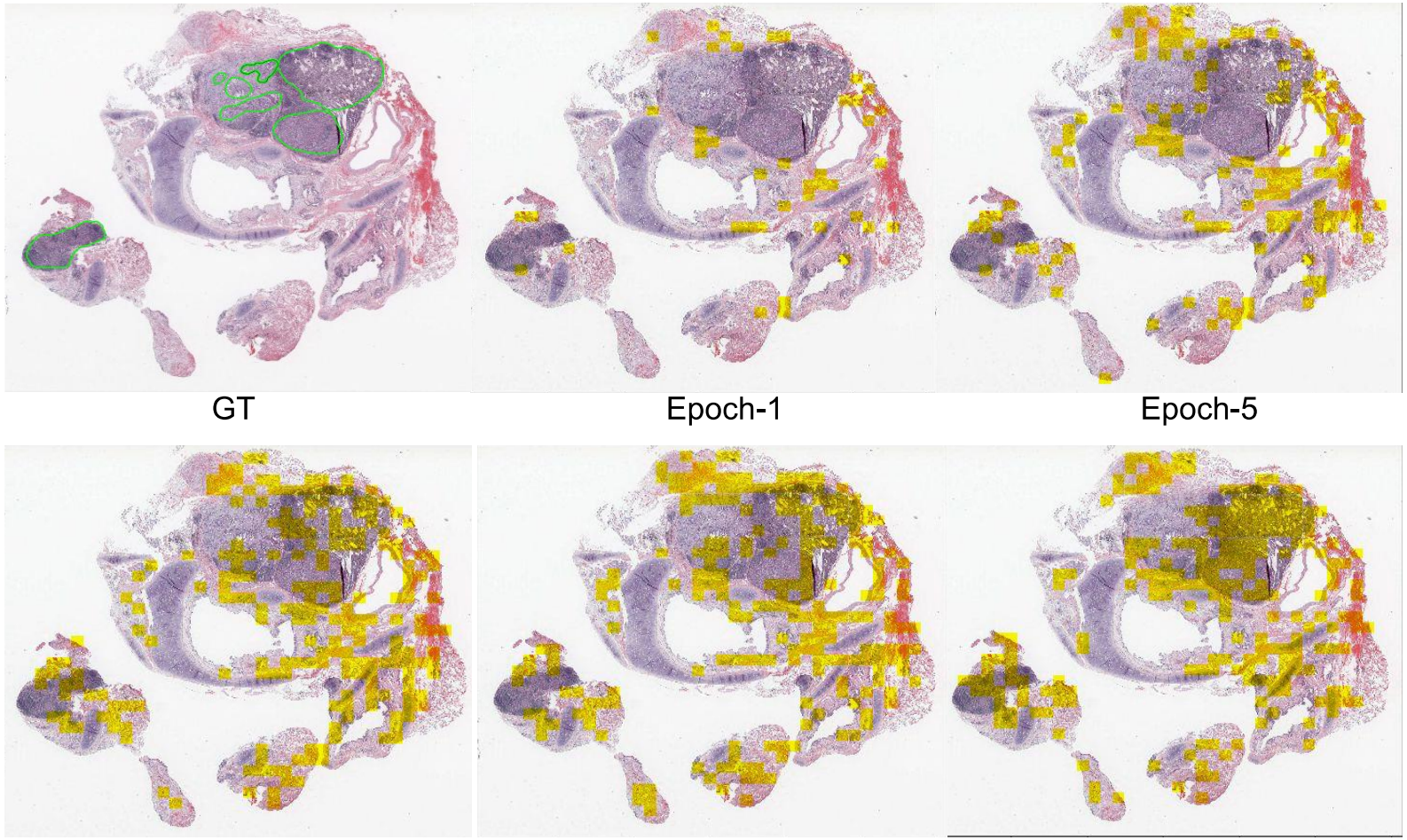

Output Risk: B x 1
Graph Gather
Output: B x 1
Graph Convolution
Output: B x N x 1
Graph Convolution
Output: B x N x 128
Graph Convolution
Output: B x N x 512
Graph Convolution
Output: B x N x 256
Input: B x N x 128
Node Attentions
Graph Convolution
Output: B x N x 1
Graph Convolution
Output: B x N x 32
Attention Net
VGG-16
CNN
4096
PCA
128
Vertex Feature Extraction
Graph Construction
DeepGraphSurv

- **Pathological Images and Patient Survival Time and Label**
  - TCGA, The Cancer Genome Atlas
  - NLST, National Lung Screening Trials

Database	Cancer Subtype	No. Patient	No. WSI	Quality	Avg. Size
TCGA	LUSC	463	535	medium	0.72 GB
TCGA	GBM	365	491	low	0.50 GB
NLST	ADC & SCC	263	425	high	0.74 GB

- **Evaluation Metrics**- C-index: the fraction of all pairs of patients whose predicted survival times are correctly ordered.

- **Yellow regions： high attention values**
  - High attention patches : values > 0.9 (attention values (0, 1))

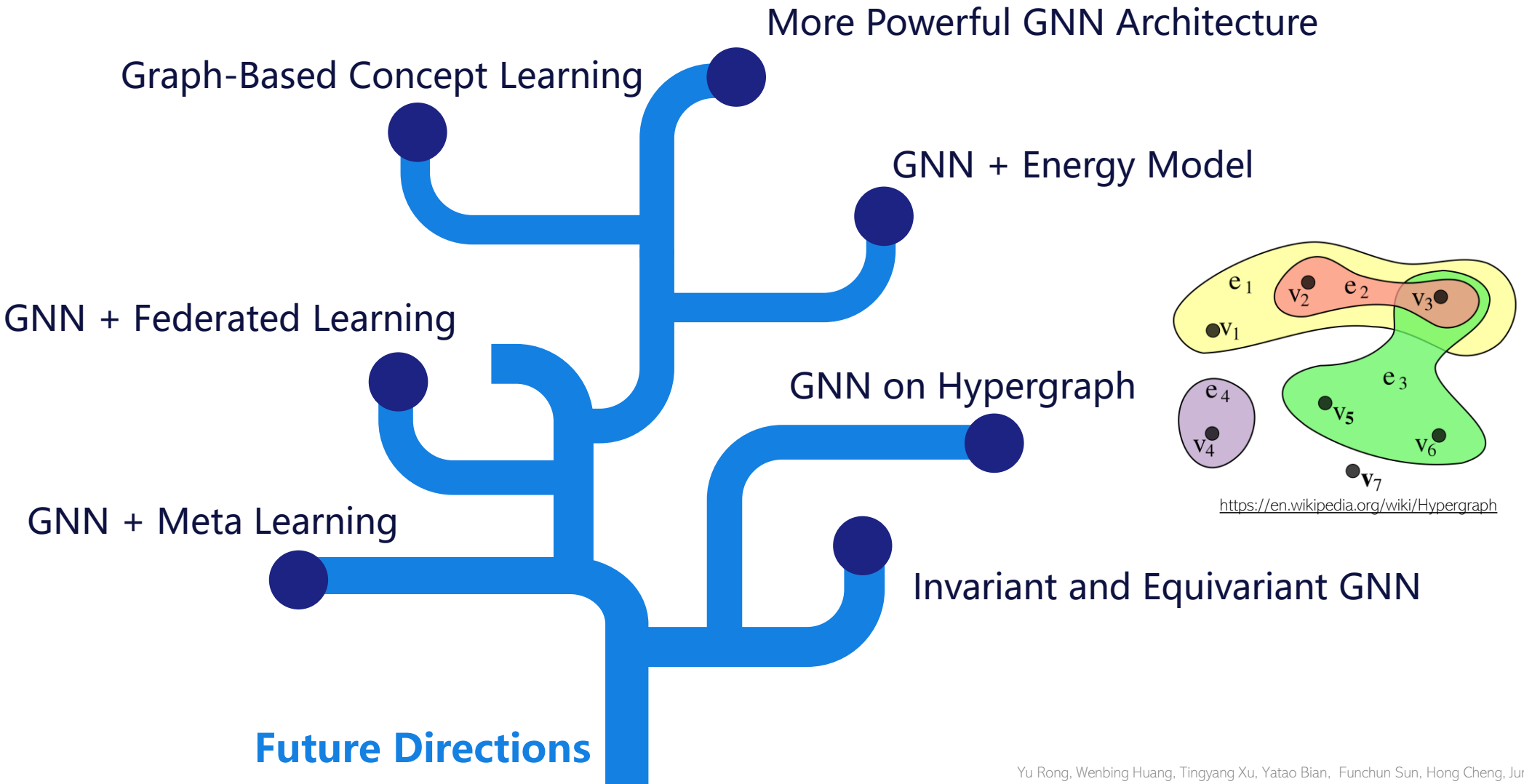


Tencent AI Lab

Model	LUSC	GBM	NLST
LASSO-Cox [19]	0.5280	0.5574	0.4738
LASSO-Cox⋆	**0.5663**	0.5165	**0.5663**
BoostCI [17]	0.5633	0.5543	0.5705
BoostCI⋆	**0.5800**	0.5130	**0.5716**
EnCox [20]	0.5216	0.5597	0.4883
EnCox⋆	**0.5740**	0.5231	**0.5742**
RSF [12]	0.5066	0.5570	0.5964
RSF⋆	**0.5492**	0.5193	0.5491
MTLSA [16]	0.5386	0.5787	0.6042
MTLSA⋆	0.5247	0.5630	0.5573
WSISA [21]	0.6380	0.5760	0.6539
GCN-Cox [8]	0.6280	0.5901	0.6845
DeepGraphSurv	**0.6606**	**0.6215**	**0.7066**

* Use our graph features for the survival model.

Future Directions



https://en.wikipedia.org/wiki/Hypergraph

Bibliography

Bibliography

- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. NeurIPS, 2018.
- Abu-El-Haija, Sami, et al. "MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing." International Conference on Machine Learning. 2019.
- Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. ICLR 2017.
- Atwood, James, and Don Towsley. "Diffusion-convolutional neural networks." Advances in neural information processing systems. 2016.
- Belghazi et al. Mine: mutual information neural estimation. ICML, 2018.
- Bian, Tian, et al. "Inverse Graph Identification: Can We Identify Node Labels Given Graph Labels?." arXiv preprint arXiv:2007.05970 (2020).
- Bian, Tian, et al. "Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks." AAAI 2020.
- Bojchevski, Aleksandar, and Stephan Günnemann. "Adversarial attacks on node embeddings via graph poisoning." International Conference on Machine Learning. PMLR, 2019.
- Bruna, Joan, et al. "Spectral networks and locally connected networks on graphs." arXiv preprint arXiv:1312.6203 (2013).
- Chami, Ines, et al. "Hyperbolic graph convolutional neural networks." Advances in neural information processing systems. 2019.
- Chang, Heng, et al. "A restricted black-box adversarial framework towards attacking graph embedding models." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 04. 2020.
- Chang, Heng, et al. "Spectral Graph Attention Network." arXiv preprint arXiv:2003.07450 (2020).
- Chen et al. On the equivalence between graph isomorphism testing and function approximation with GNNs. NeurIPS, 2019.
- Chen, Jianfei, Jun Zhu, and Le Song. "Stochastic Training of Graph Convolutional Networks with Variance Reduction." International Conference on Machine Learning. 2018.
- Chen, Jie, Tengfei Ma, and Cao Xiao. "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling." International Conference on Learning Representations. 2018.
- Chen, Yuzhao, et al. "On Self-Distilling Graph Neural Network." arXiv preprint arXiv:2011.02255 (2020).
- Chiang, Wei-Lin, et al. "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks." Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019.
- Cybenko, G. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4):303–314, 1989.
- Dai, Hanjun, et al. "Adversarial attack on graph structured data." International conference on machine learning. PMLR, 2018.
- Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." Advances in neural information processing systems. 2016.
- Dehmamy et al. Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology. NeurIPS, 2019.
- Durán & Niepert. Learning Graph Representations with Embedding Propagation. NeurIPS, 2017.
- Entezari, Negin, et al. "All you need is low (rank) defending against adversarial attacks on graphs." Proceedings of the 13th International Conference on Web Search and Data Mining. 2020.
- Fan, Wenqi, et al. "Graph neural networks for social recommendation." The World Wide Web Conference. 2019.
- Garg et al. Generalization and Representational Limits of Graph Neural Networks. ICML, 2020.

Bibliography

- Gilmer, Justin, et al. "Neural Message Passing for Quantum Chemistry." ICML. 2017.
- Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples." ICLR 2015.
- Gori, Marco, Gabriele Monfardini, and Franco Scarselli. "A new model for learning in graph domains." Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.. Vol. 2. IEEE, 2005.
- Gou, Jianping, et al. "Knowledge distillation: A survey." International Journal of Computer Vision (2021): 1-31.
- Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems. 2017.
- Hinton, G. E., & Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. Science, 2006.
- Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." arXiv preprint arXiv:1503.02531 (2015).
- Hjelm et al. Learning deep representations by mutual information estimation and maximization. ICLR, 2019.
- Hornik, K. Approximation capabilities of multilayer feedforward networks. Neural networks, 4(2):251–257, 1991.
- Hu et al. Strategies for Pre-training Graph Neural Networks. ICLR, 2020.
- Huang, Wenbing, et al. "Adaptive sampling towards fast graph representation learning." Advances in neural information processing systems. 2018.
- Jin, Ming, et al. "Power up! Robust Graph Convolutional Network via Graph Powering." AAAI 2021
- Jin, Wei, et al. "Graph structure learning for robust graph neural networks." Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2020.
- Jin, Wengong, Regina Barzilay, and Tommi Jaakkola. "Multi-objective molecule generation using interpretable substructures." International Conference on Machine Learning. PMLR, 2020.
- Junchi, Yu, et al. “Graph Information Bottleneck for Subgraph Recognition.”, ICLR 2021.
- Kipf & Welling. Variational Graph Auto-Encoders. arXiv, 2016.
- Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).
- Klicpera, Johannes, Aleksandar Bojchevski, and Stephan Günnemann. "Predict then Propagate: Graph Neural Networks meet Personalized PageRank." International Conference on Learning Representations. 2018.
- Li, Jia, et al. "Adversarial attack on community detection by hiding individuals." Proceedings of The Web Conference 2020. 2020.
- Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective." The World Wide Web Conference. 2019.
- Li, Yujia, et al. "Gated graph sequence neural networks." arXiv preprint arXiv:1511.05493 (2015).
- Liao, Renjie, et al. "LanczosNet: Multi-Scale Deep Graph Convolutional Networks." International Conference on Learning Representations. 2018.
- Liu et al. N-Gram Graph: Simple Unsupervised Representation for Graphs, with Applications to Molecules. NeurIPS, 2019.
- Loukas, What graph neural networks cannot learn: depth vs width. ICLR, 2020.
- Ma, Yao, et al. "Graph convolutional networks with eigenpooling." Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019.

Bibliography

- Morris et al. Weisfeiler and Leman Go Neural Higher-order Graph Neural Networks. AAAI, 2019.
- Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov. "Learning convolutional neural networks for graphs." International conference on machine learning. 2016.
- Nowozin et al. f-gan: Training generative neural samplers using variational divergence minimization. NeurIPS, 2016.
- Peng et al. Graph Representation Learning via Graphical Mutual Information Maximization. WWW, 2020.
- Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. KDD, 2020.
- Rong et al. GROVER: Self-supervised Message Passing Transformer on Large-scale Molecular Data. arXiv, 2020.
- Sato et al. Approximation Ratios of Graph Neural Networks for Combinatorial Problems. NeurIPS, 2019.
- Scarselli, Franco, et al. "The graph neural network model." IEEE Transactions on Neural Networks 20.1 (2008): 61-80.
- Shervashidze et al. Weisfeiler-Lehman Graph Kernels. JRML, 2011.
- Sperduti, Alessandro, and Antonina Starita. "Supervised neural networks for the classification of structures." IEEE Transactions on Neural Networks 8.3 (1997): 714-735.
- Sun et al. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. ICLR, 2020.
- Tai, Kai Sheng, Richard Socher, and Christopher D. Manning. "Improved semantic representations from tree-structured long short-term memory networks." arXiv preprint arXiv:1503.00075 (2015).
- Veličković et al. Deep Graph Infomax. ICLR, 2019.
- Veličković, Petar, et al. "Graph Attention Networks." International Conference on Learning Representations. 2018.
- Vincent et al. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. JMLR, 2010.
- Wang, X., & Gupta, A. Videos as Space-Time Region Graphs. ECCV, 2018.
- Xu et al. How Powerful Are Graph Neural Networks. ICLR, 2019.
- Xu, Bingbing, et al. "Graph Wavelet Neural Network." International Conference on Learning Representations. 2018.
- Xu, Kaidi, et al. "Topology attack and defense for graph neural networks: An optimization perspective." IJCAI 2019.
- Yan et al. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI, 2018.
- Yang, Yiding, et al. "Distilling knowledge from graph convolutional networks." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.
- Ying, Rex, et al. "Graph convolutional neural networks for web-scale recommender systems." Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018.
- Ying, Zhitao, et al. "Hierarchical graph representation learning with differentiable pooling." Advances in neural information processing systems. 2018.
- Yuan, Li, et al. "Revisiting knowledge distillation via label smoothing regularization." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

Bibliography

- Zeng, et al. Graph convolutional networks for temporal action localization. ICCV, 2019
- Zeng, Hanqing, et al. "GraphSAINT: Graph Sampling Based Inductive Learning Method." International Conference on Learning Representations. 2020.
- Zhang, Jiani, et al. "Gaan: Gated attention networks for learning on large and spatiotemporal graphs." arXiv preprint arXiv:1803.07294 (2018).
- Zhang, Xiang, and Marinka Zitnik. "Gnnguard: Defending graph neural networks against adversarial attacks." arXiv preprint arXiv:2006.08149 (2020).
- Zhu, Dingyuan, et al. "Robust graph convolutional networks against adversarial attacks." Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019.
- Zügner, Daniel, Amir Akbarnejad, and Stephan Günnemann. "Adversarial attacks on neural networks for graph data." Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2018.
- Zügner, Daniel, and Stephan Günnemann. "Adversarial Attacks on Graph Neural Networks via Meta Learning." International Conference on Learning Representations. 2018.