KDD2020

Deep Graph Learning: Foundations, Advances and Applications

Theme II: Advance Topics and Applications

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

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

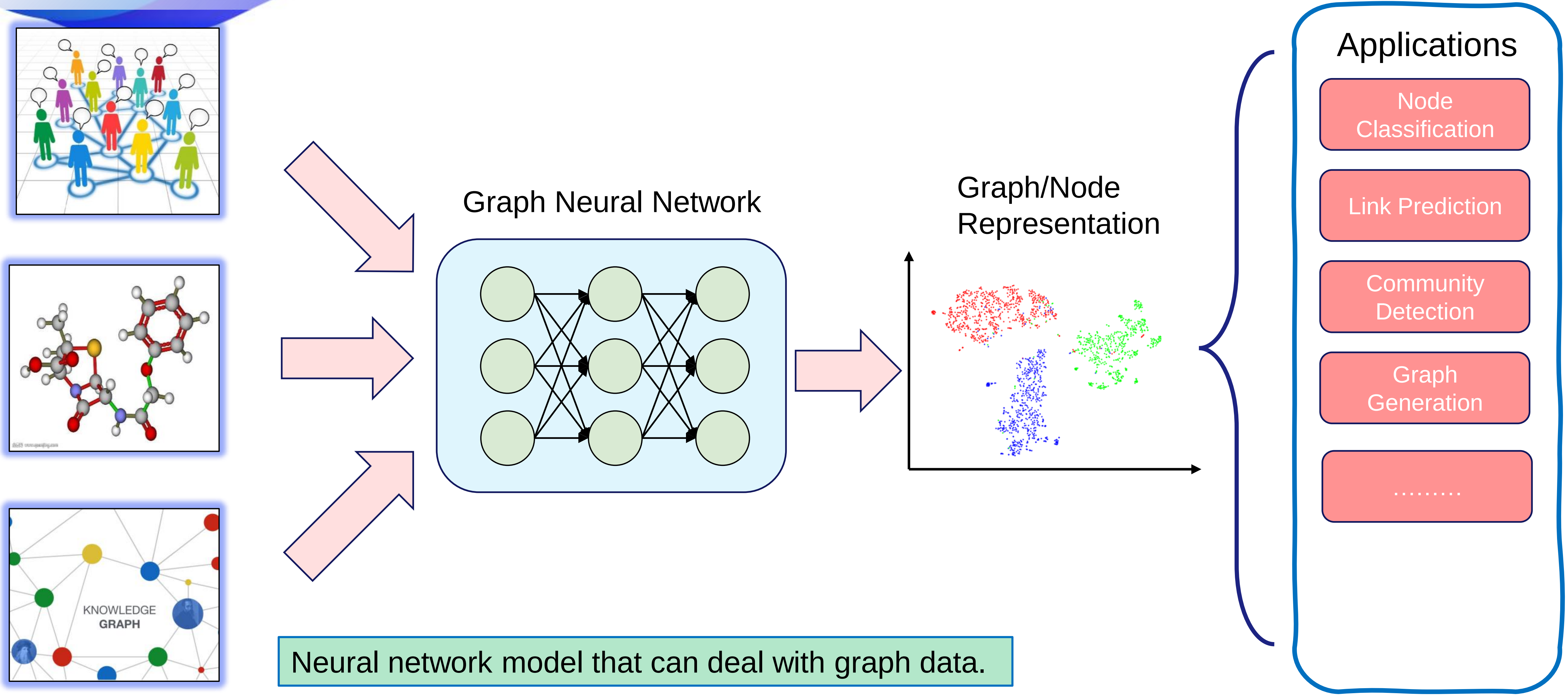
Tencent AI Lab

Agenda

- Brief History of Graph Neural Networks
- Advanced Topics in GNN:
  - Expressivity of GNNs
  - Training Deep GNNs
  - Scalability of GNNs
  - Self/Un-Supervised Learning of GNNs
- Applications:
  - GNN in Social Networks
  - GNN in Medical Imaging
- Future Directions

The Brief History of Graph Neural Networks

What is the Graph Neural Network?



Graph Neural Network

Graph/Node Representation

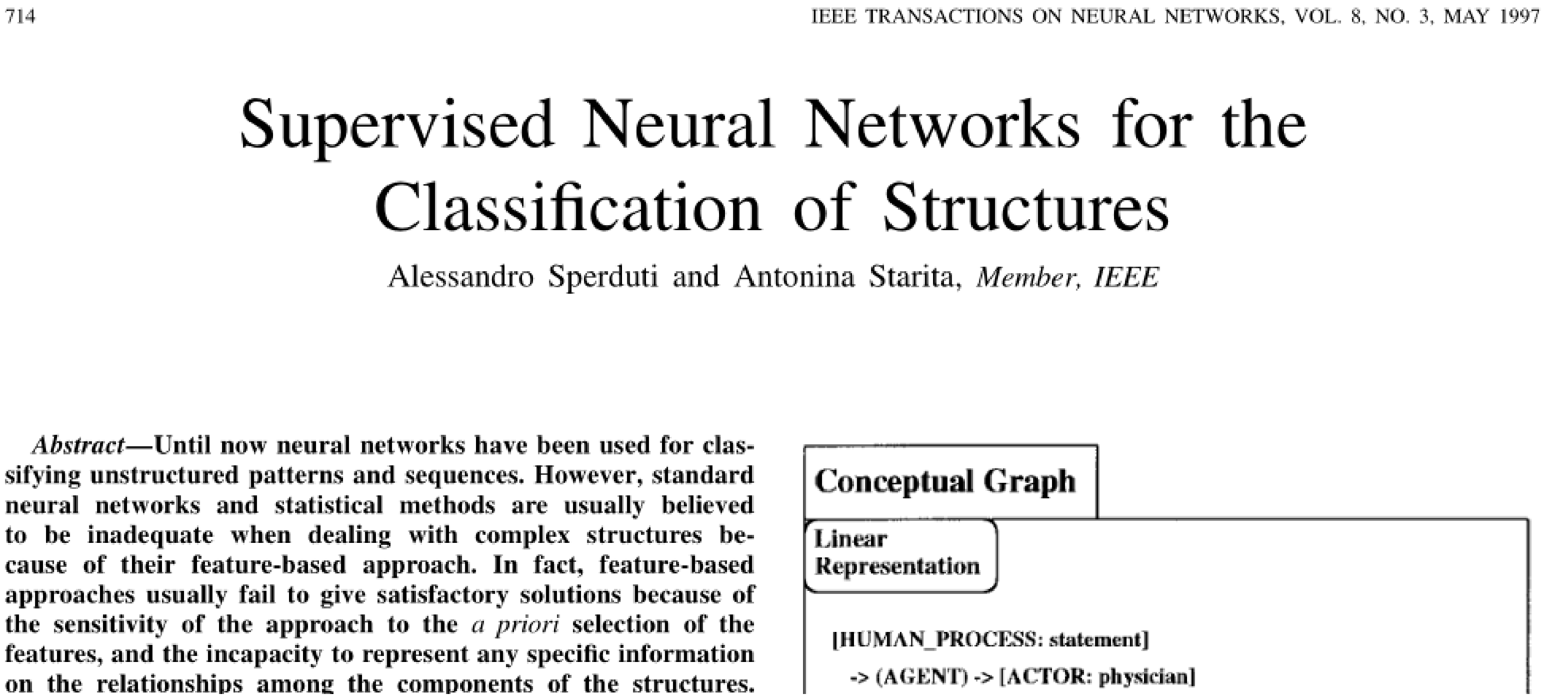
Applications

- 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

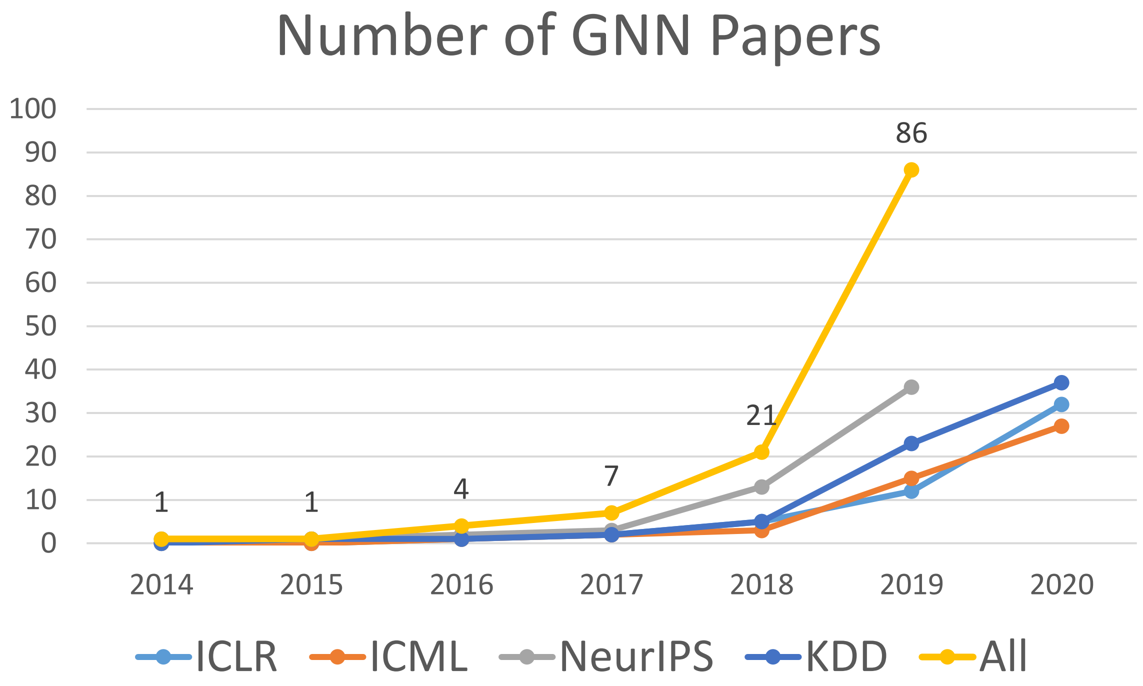
Supervised Neural Networks for the Classification of Structures

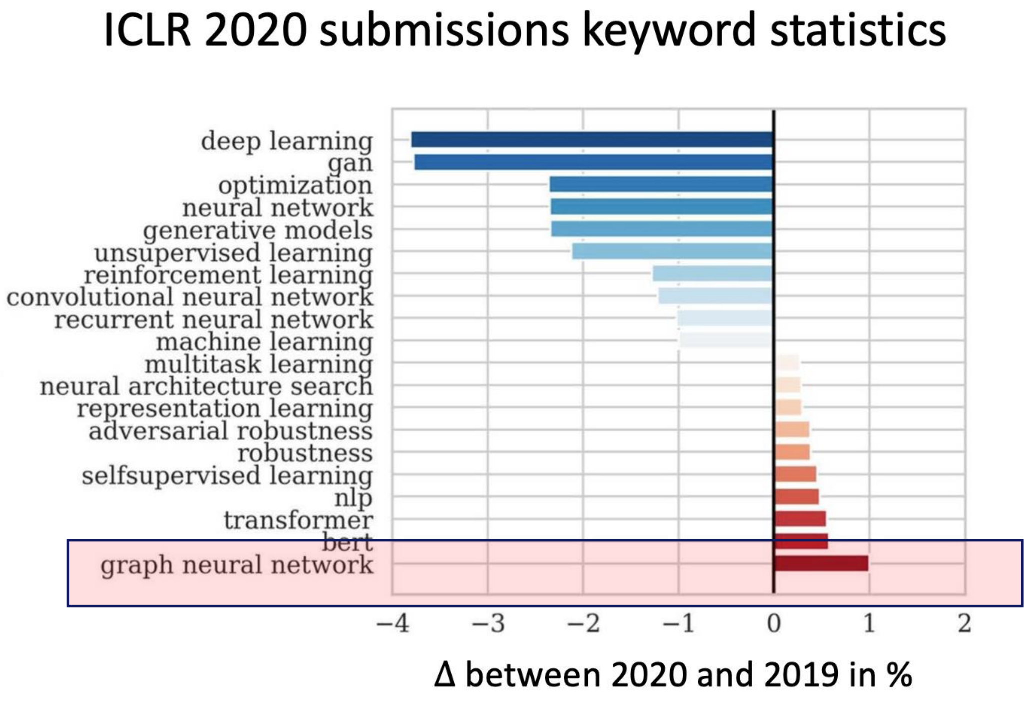
Alessandro Sperduti and Antonina Starita, *Member, IEEE*

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

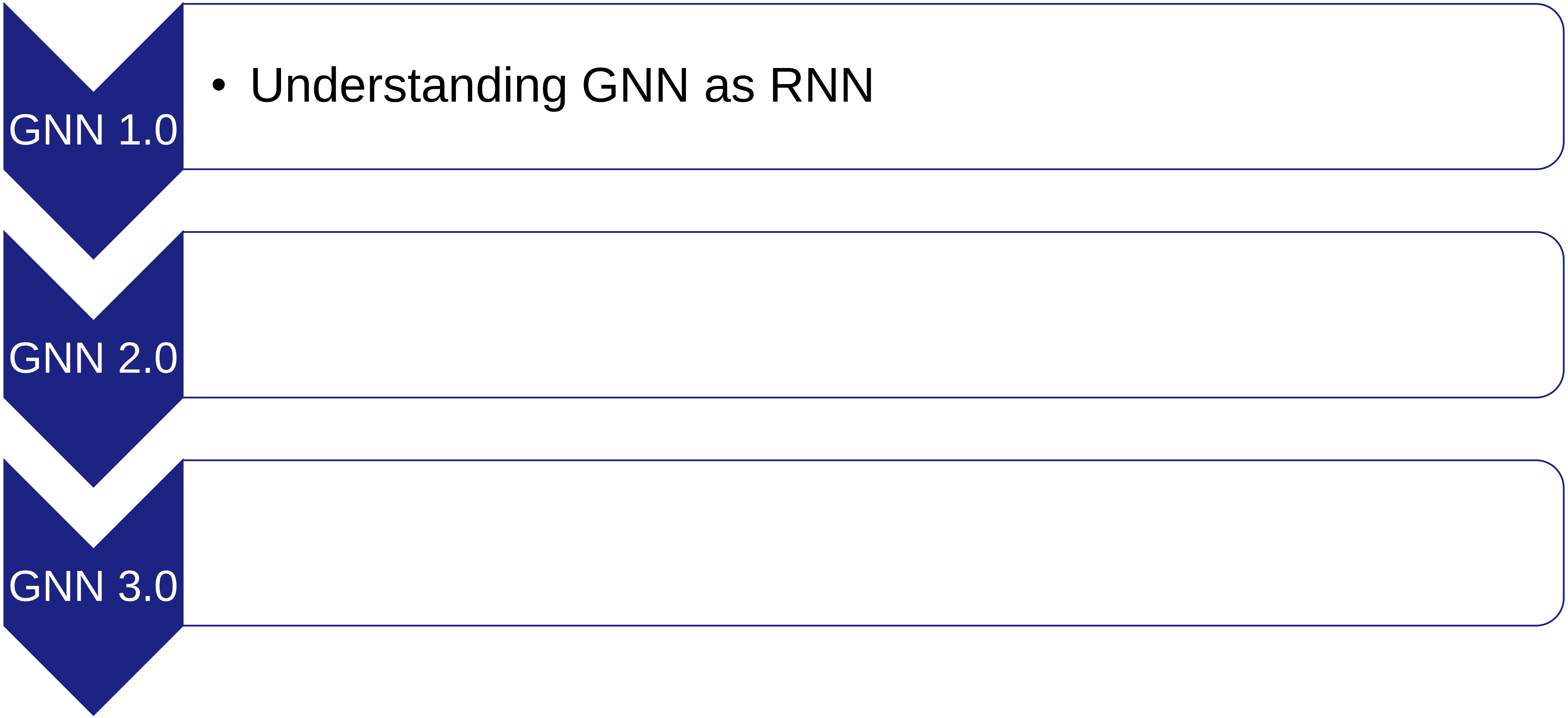
The Model of Graph Neural Networks

GNN 1.0

GNN 2.0

GNN 3.0

The Model of Graph Neural Networks



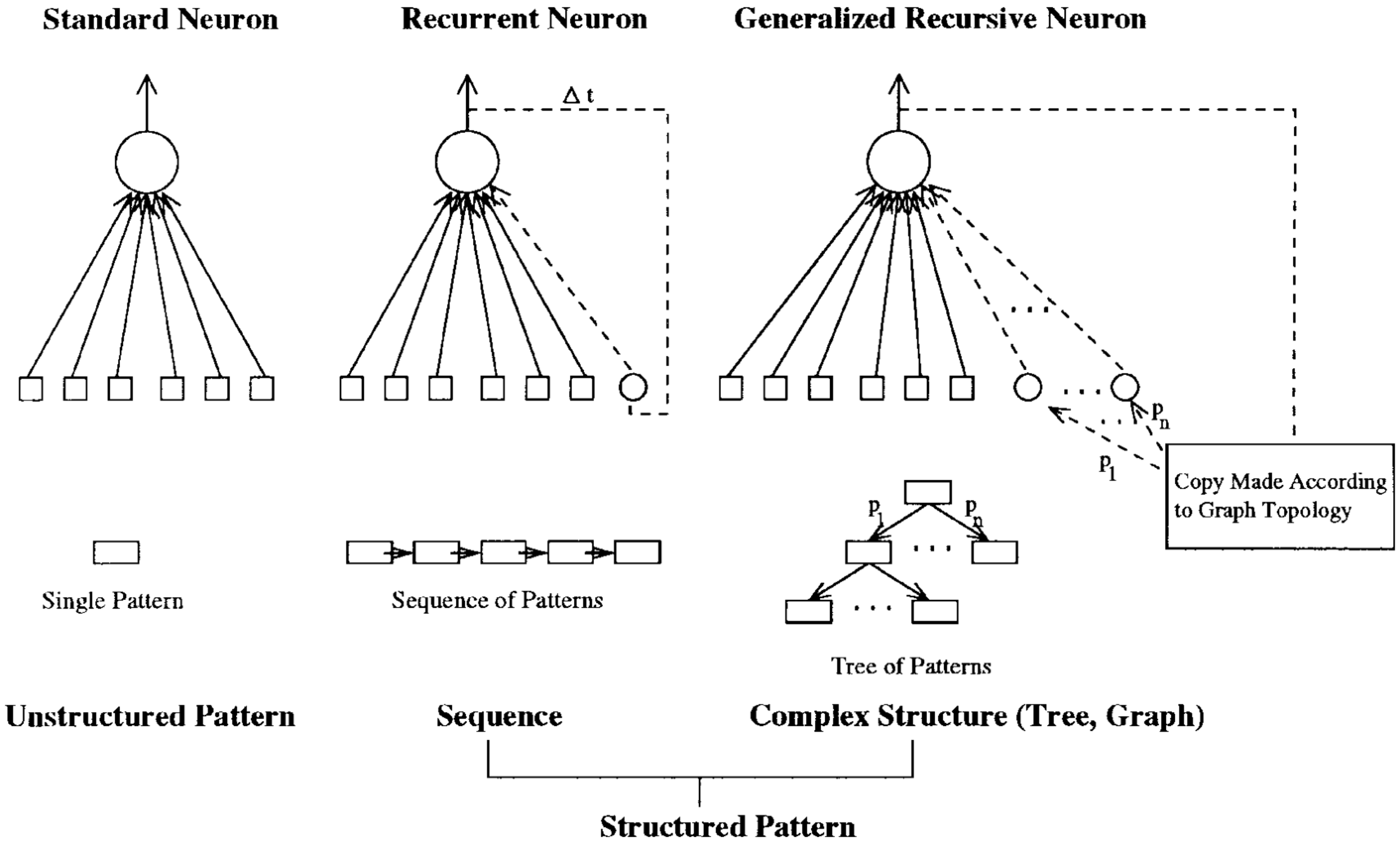
GNN 1.0

- Understanding GNN as RNN

GNN 2.0

GNN 3.0

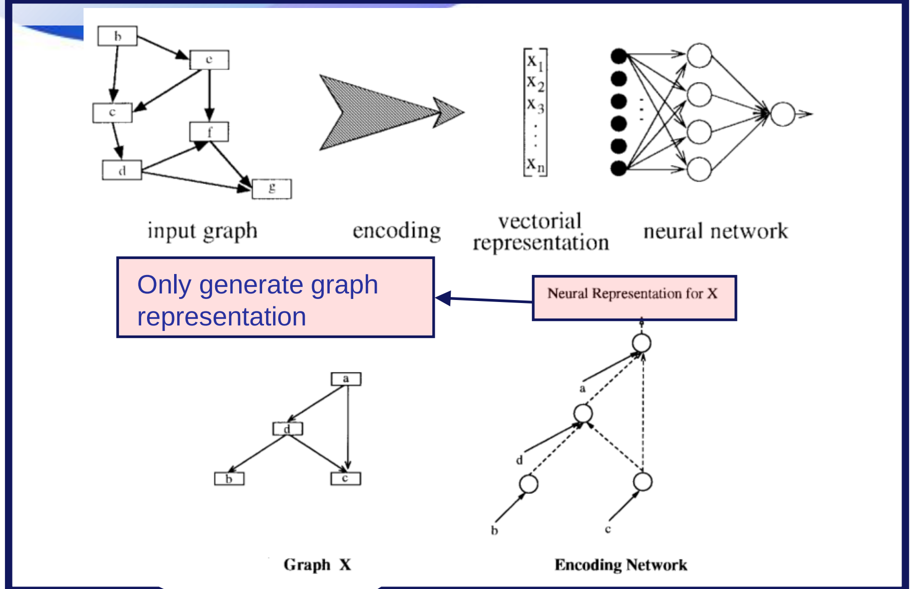
GNN 1.0: Understanding GNN as RNN



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

Sperduti, Alessandro, and Antonina Starita. 1997

GNN 1.0: Understanding GNN as RNN



Only generate graph representation

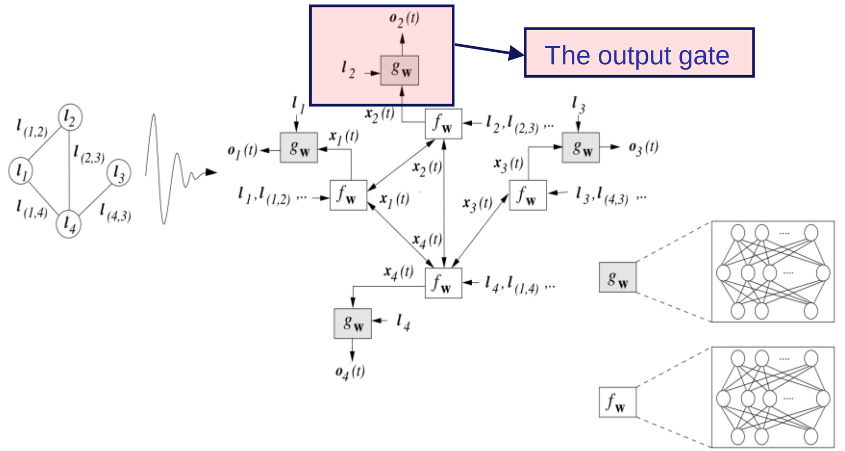
Before 2000

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**.

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.



The output gate

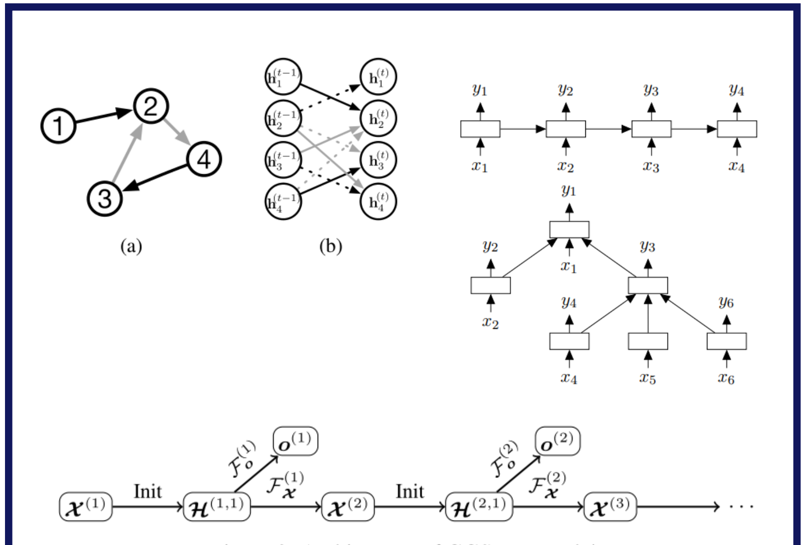


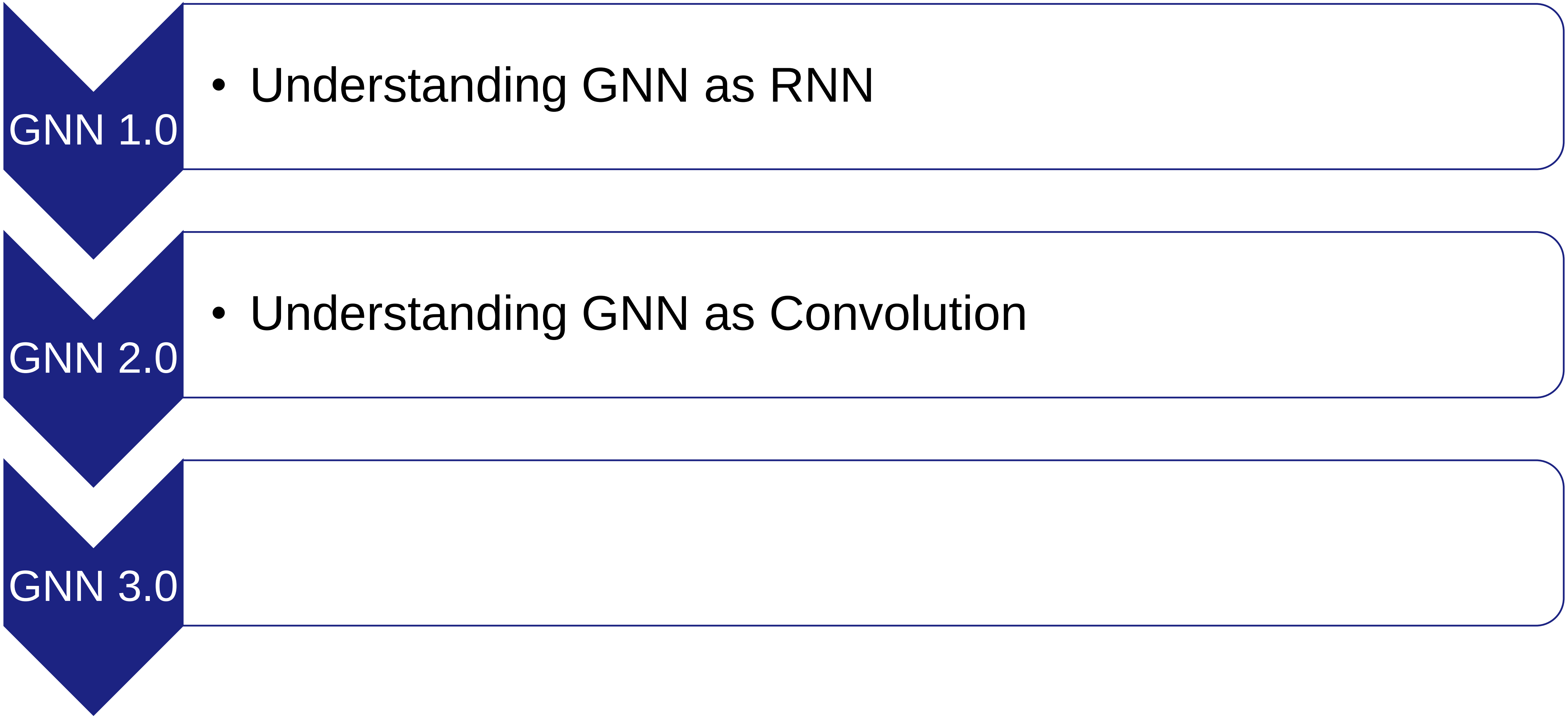
Figure 2: Architecture of GGS-NN models.

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

The Brief History of Graph Neural Networks



GNN 1.0

- Understanding GNN as RNN

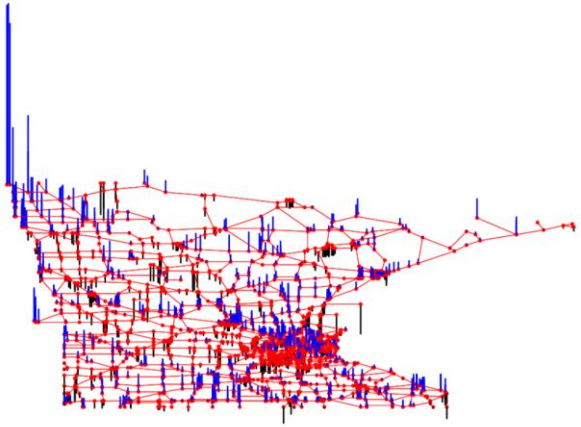
GNN 2.0

- Understanding GNN as Convolution

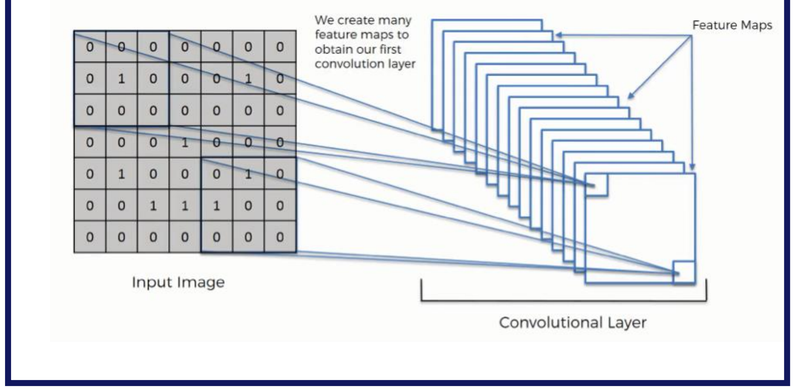
GNN 3.0

GNN 2.0: Understanding GNN as Convolution

Graph Signal Processing



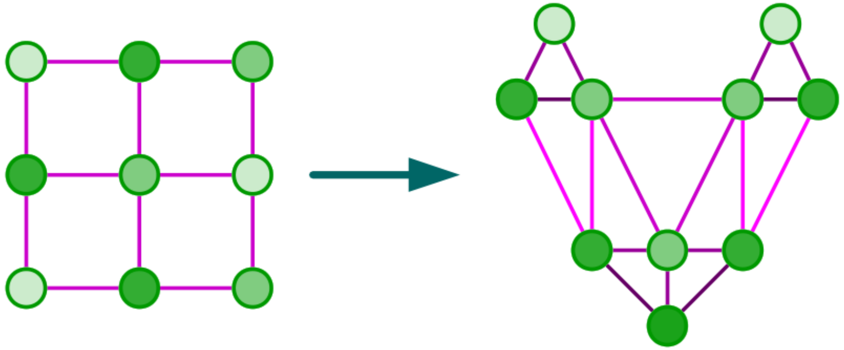
Convolutional Neural Networks



0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

We create many feature maps to obtain our first convolution layer
Feature Maps
Input Image
Convolutional Layer

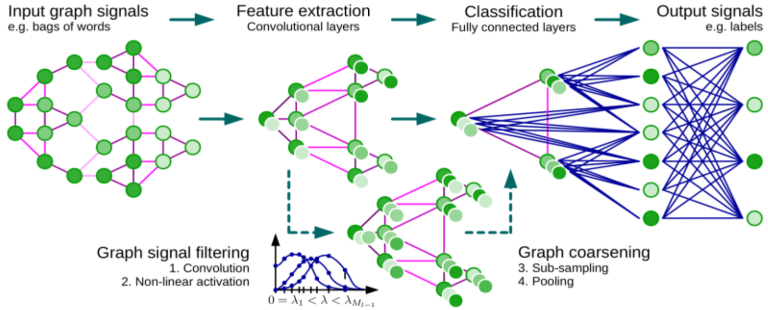
- How to perform the convolution on graphs?
  - Irregular structures.
  - Weighted edges.
  - No orientation or ordering (in general).



GNN 2.0: Understanding GNN as Convolution

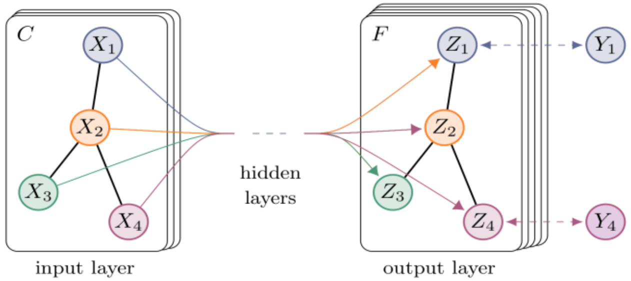
[1] Bruna, Joan, et al. 2014
[2] Defferrard, Michaël, et.al. 2016
[3] Niepert, Mathias, et.al. 2016
[4] Kipf, Thomas N., and Max Welling. 2017

ChebyNet (NIPS 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.

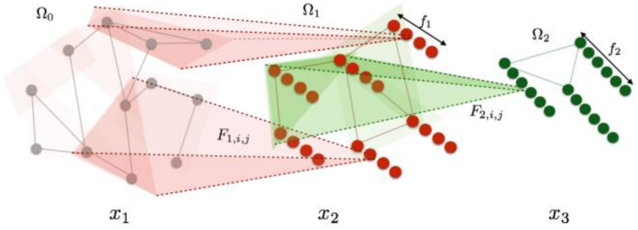
Graph Convolutional Network (ICLR 2017)



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

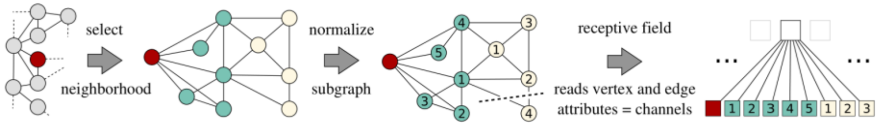
- Approximate 1-order Chebyshev polynomial the in spatial domain.
- Layer-wise convolution to extend receptive field.
- **The practical convolutional model for graphs.**

Deep Locally Connected Networks(ICLR 2014) [1]



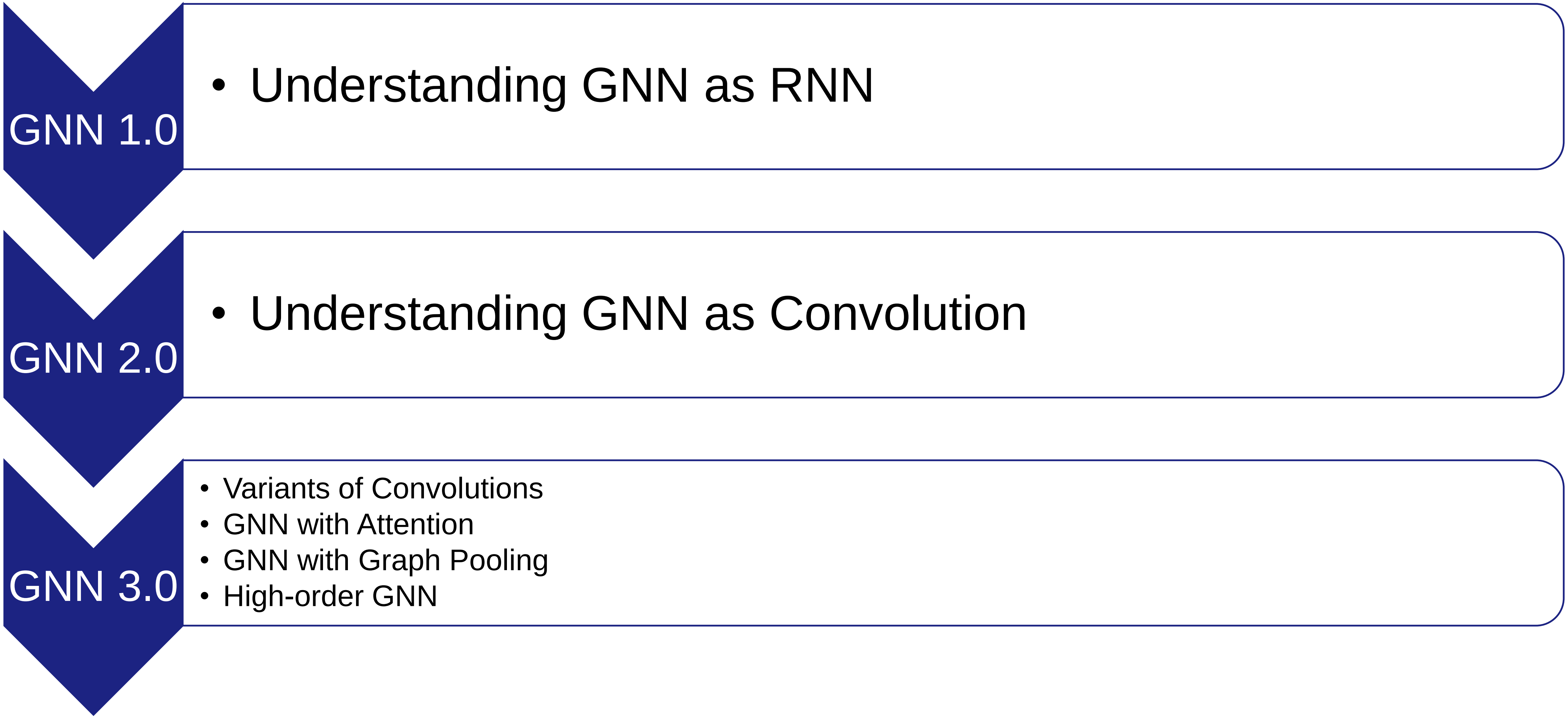
- Discuss two constructions on both spatial and spectral domain.
- Analog the convolution operation based on the Laplacian spectrum.
- **Additional eigen decomposition is needed.**

PATCHY-SAN (ICML 2016)



- Neighborhood sampling to construct receptive field.

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
- High-order GNN

GNN 3.0: Variants of Convolutions

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

Lanczos Network [3]

Graph Wavelet Neural Network [1]

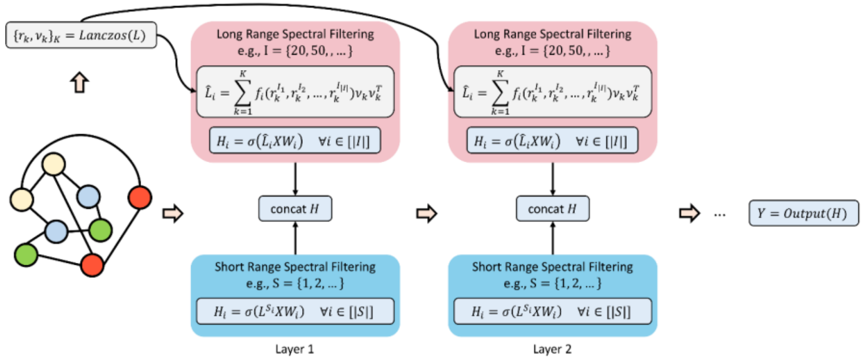
Hyperbolic GCN [2]

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

GNN 3.0: Variants of Convolutions

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

Lanczos Network [3]



$\{r_k, v_k\}_K = Lanczos(L)$
Long Range Spectral Filtering
e.g., I = {20, 50, , … }
$\hat{L}_i = \sum_{k=1}^{K} f_i(r_k^{I_1}, r_k^{I_2}, \dots, r_k^{I_{|I|}})v_k v_k^T$
$H_i = \sigma(\hat{L}_i X W_i) \quad \forall i \in [|I|]$
concat H
Short Range Spectral Filtering
e.g., S = {1, 2, … }
$H_i = \sigma(L^{S_i} X W_i) \quad \forall i \in [|S|]$
Layer 1
Long Range Spectral Filtering
e.g., I = {20, 50, , … }
$\hat{L}_i = \sum_{k=1}^{K} f_i(r_k^{I_1}, r_k^{I_2}, \dots, r_k^{I_{|I|}})v_k v_k^T$
$H_i = \sigma(\hat{L}_i X W_i) \quad \forall i \in [|I|]$
concat H
Short Range Spectral Filtering
e.g., S = {1, 2, … }
$H_i = \sigma(L^{S_i} X W_i) \quad \forall i \in [|S|]$
Layer 2
…
$Y = Output(H)$

- 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]

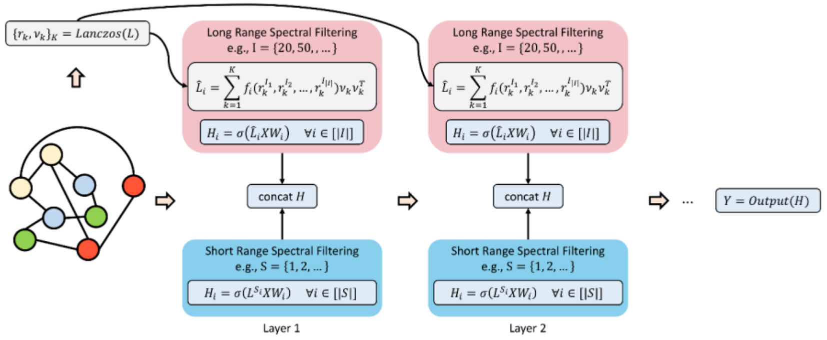
Hyperbolic GCN [2]

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

GNN 3.0: Variants of Convolutions

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

Lanczos Network [3]



$\{r_k, v_k\}_K = Lanczos(L)$
Long Range Spectral Filtering
e.g., I = {20, 50, , ... }
$\hat{L}_i = \sum_{k=1}^{K} f_i(r_k^{I_1}, r_k^{I_2}, \ldots, r_k^{I_{|I|}}) v_k v_k^T$
$H_i = \sigma(\hat{L}_i X W_i) \quad \forall i \in [|I|]$
concat H
Short Range Spectral Filtering
e.g., $S = \{1, 2, \ldots\}$
$H_i = \sigma(L^{S_i} X W_i) \quad \forall i \in [|S|]$
Layer 1
Long Range Spectral Filtering
e.g., I = {20, 50, , ... }
$\hat{L}_i = \sum_{k=1}^{K} f_i(r_k^{I_1}, r_k^{I_2}, \ldots, r_k^{I_{|I|}}) v_k v_k^T$
$H_i = \sigma(\hat{L}_i X W_i) \quad \forall i \in [|I|]$
concat H
Short Range Spectral Filtering
e.g., $S = \{1, 2, \ldots\}$
$H_i = \sigma(L^{S_i} X W_i) \quad \forall i \in [|S|]$
Layer 2
...
$Y = Output(H)$

- 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]

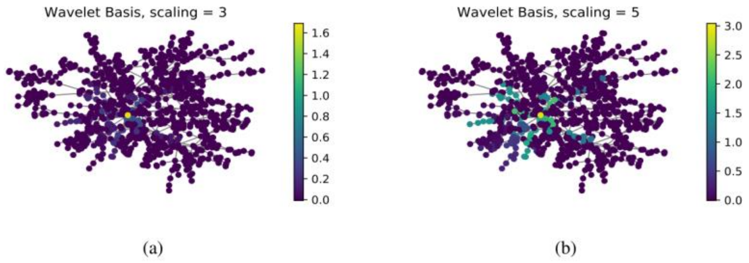


Figure 1: Wavelets on an example graph at (a) small scale and (b) large scale.

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

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

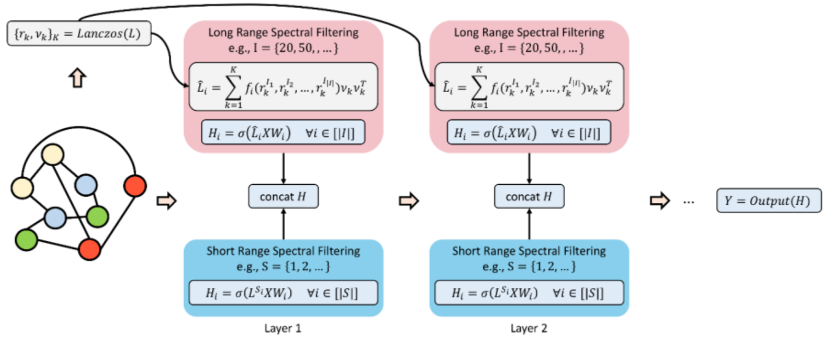
Hyperbolic GCN [2]

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

GNN 3.0: Variants of Convolutions

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

Lanczos Network [3]



$\{r_k, v_k\}_K = Lanczos(L)$
Long Range Spectral Filtering
e.g., I = {20, 50, , ... }
$\hat{L}_i = \sum_{k=1}^{K} f_i(r_k^{I_1}, r_k^{I_2}, \ldots, r_k^{I_{|I|}}) v_k v_k^T$
$H_i = \sigma(\hat{L}_i X W_i) \quad \forall i \in [|I|]$
concat H
Short Range Spectral Filtering
e.g., S = {1, 2, ... }
$H_i = \sigma(L^{S_i} X W_i) \quad \forall i \in [|S|]$
Layer 1
Long Range Spectral Filtering
e.g., I = {20, 50, , ... }
$\hat{L}_i = \sum_{k=1}^{K} f_i(r_k^{I_1}, r_k^{I_2}, \ldots, r_k^{I_{|I|}}) v_k v_k^T$
$H_i = \sigma(\hat{L}_i X W_i) \quad \forall i \in [|I|]$
concat H
Short Range Spectral Filtering
e.g., S = {1, 2, ... }
$H_i = \sigma(L^{S_i} X W_i) \quad \forall i \in [|S|]$
Layer 2
...
$Y = Output(H)$

- 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]

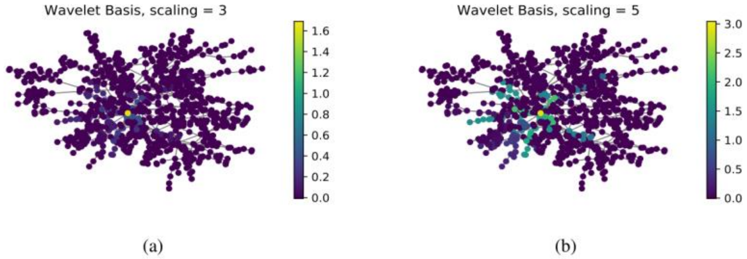
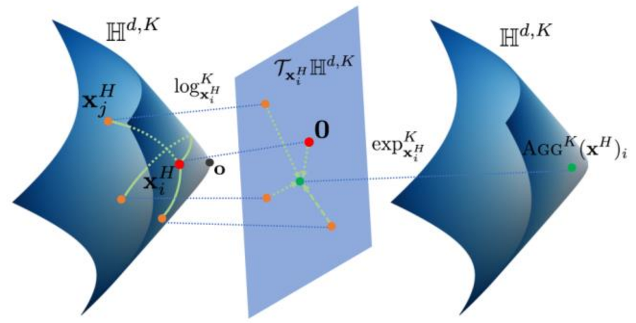


Figure 1: Wavelets on an example graph at (a) small scale and (b) large scale.

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

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

Hyperbolic GCN [2]



$\mathbb{H}^{d,K}$
$\mathbf{x}_j^H$
$\mathbf{x}_i^H$
$\log_{\mathbf{x}_i^H}^K$
$\mathcal{T}_{\mathbf{x}_i^H} \mathbb{H}^{d,K}$
$\mathbf{0}$
$\mathbf{o}$
$\exp_{\mathbf{x}_i^H}^K$
$\mathbb{H}^{d,K}$
$\mathrm{AGG}^K(\mathbf{x}^H)_i$

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.

$$\boldsymbol{y}_i^{(l+1),H} = \mathrm{AGG}^{K_l}\left(\boldsymbol{h}^{(l),H}\right)_i$$

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

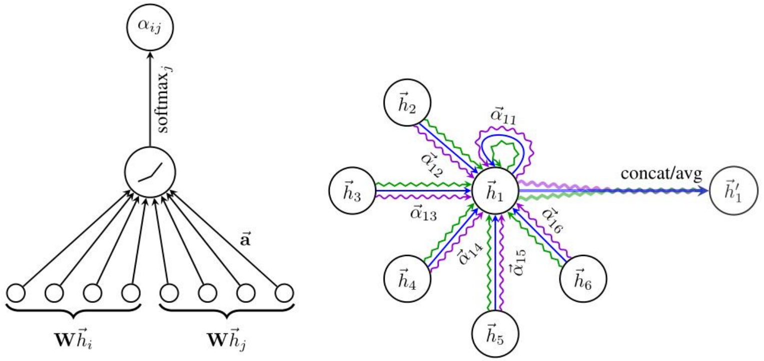
GNN 3.0: GNN with Attention

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

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

Fixed during training

Graph Attention Network [1]



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

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

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

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

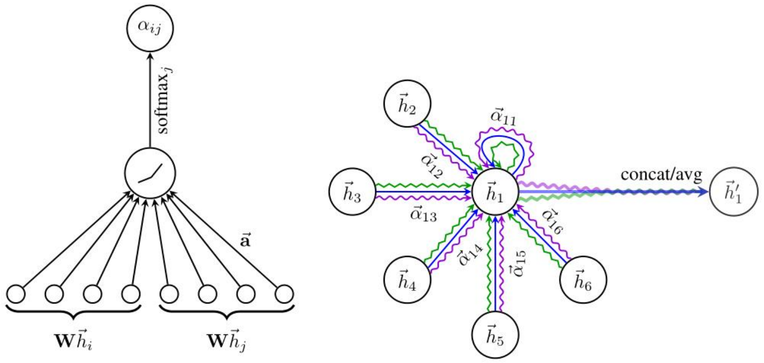
The original form:

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

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

Fixed during training

Graph Attention Network [1]

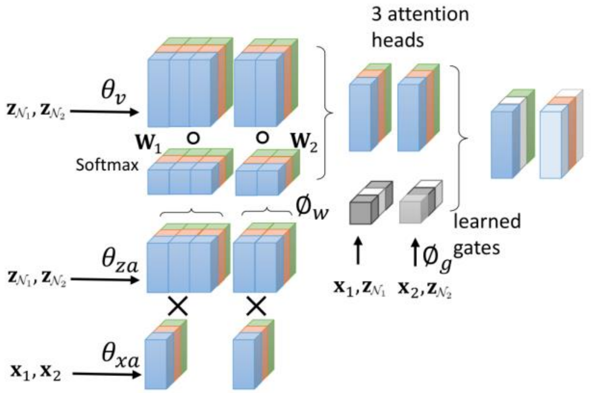


Replace the fixed aggregation weight a_{ij} to the learnable self-attention.

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

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

Gated Attention Networks [2]



3 attention heads
θ_v
$\mathbf{z}_{\mathcal{N}_1}, \mathbf{z}_{\mathcal{N}_2}$
$\mathbf{W}_1$
$\mathbf{W}_2$
Softmax
$\emptyset_w$
learned gates
$\emptyset_g$
$\mathbf{x}_1, \mathbf{z}_{\mathcal{N}_1}$
$\mathbf{x}_2, \mathbf{z}_{\mathcal{N}_2}$
θ_{za}
$\mathbf{z}_{\mathcal{N}_1}, \mathbf{z}_{\mathcal{N}_2}$
θ_{xa}
$\mathbf{x}_1, \mathbf{x}_2$

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

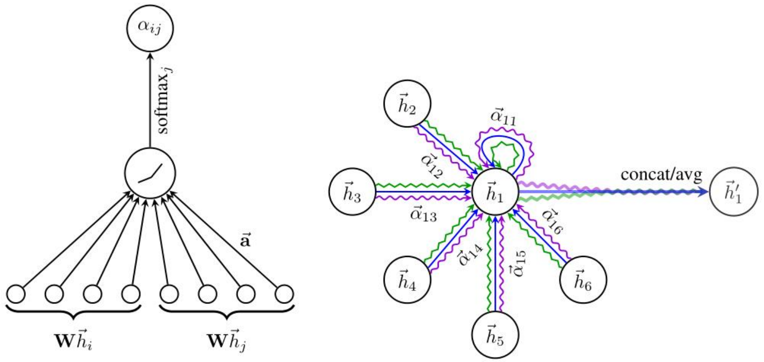
The original form:

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

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

Fixed during training

Graph Attention Network [1]

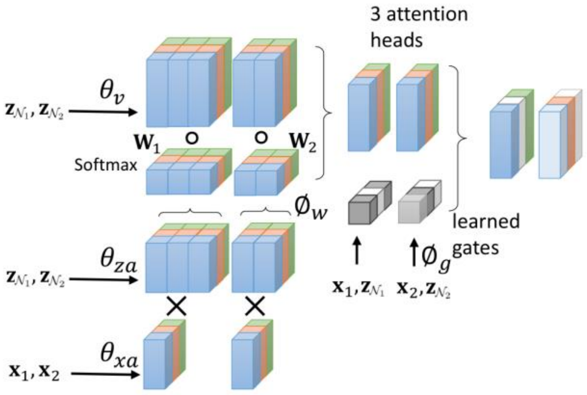


Replace the fixed aggregation weight a_{ij} to the learnable self-attention.

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

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

Gated Attention Networks [2]



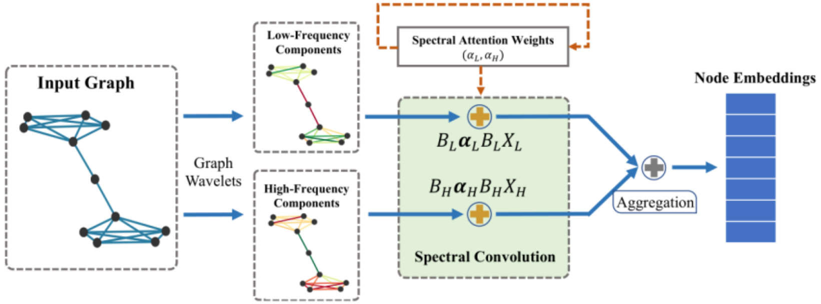
3 attention heads
θ_v
$\mathbf{z}_{\mathcal{N}_1}, \mathbf{z}_{\mathcal{N}_2}$
$\mathbf{W}_1$
$\mathbf{W}_2$
Softmax
ϕ_w
θ_{za}
$\mathbf{z}_{\mathcal{N}_1}, \mathbf{z}_{\mathcal{N}_2}$
learned gates
ϕ_g
$\mathbf{x}_1, \mathbf{z}_{\mathcal{N}_1}$
$\mathbf{x}_2, \mathbf{z}_{\mathcal{N}_2}$
θ_{xa}
$\mathbf{x}_1, \mathbf{x}_2$

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

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

K is the number of heads.

Spectral Graph Attention Network [3]



Input Graph
Graph Wavelets
Low-Frequency Components
High-Frequency Components
Spectral Attention Weights
(α_L, α_H)
$B_L \boldsymbol{\alpha}_L B_L X_L$
$B_H \boldsymbol{\alpha}_H B_H X_H$
Spectral Convolution
Aggregation
Node Embeddings

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

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

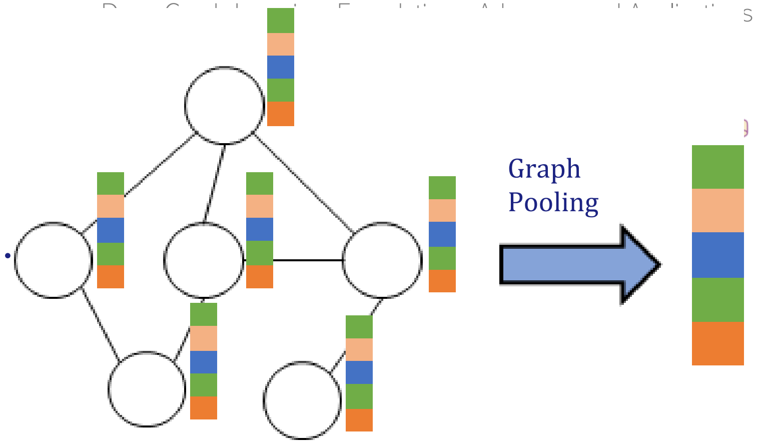
$\boldsymbol{B} = [\boldsymbol{B}_L, \boldsymbol{B}_H]$ is the spectral graph wavelet bases.

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

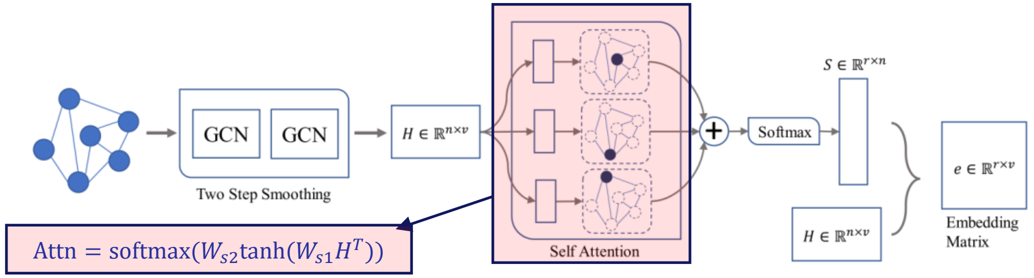
GNN 3.0: GNN with Graph Pooling

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)

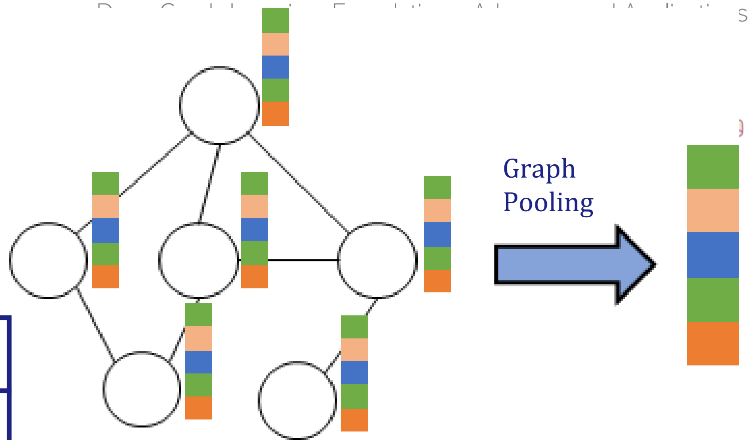

GCN
GCN
Two Step Smoothing
$H \in \mathbb{R}^{n\times v}$
Self Attention
Softmax
$S \in \mathbb{R}^{r\times n}$
$H \in \mathbb{R}^{n\times v}$
$e \in \mathbb{R}^{r\times v}$
Embedding
Matrix
$\mathrm{Attn} = \mathrm{softmax}(W_{s2}\tanh(W_{s1}H^T))$

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

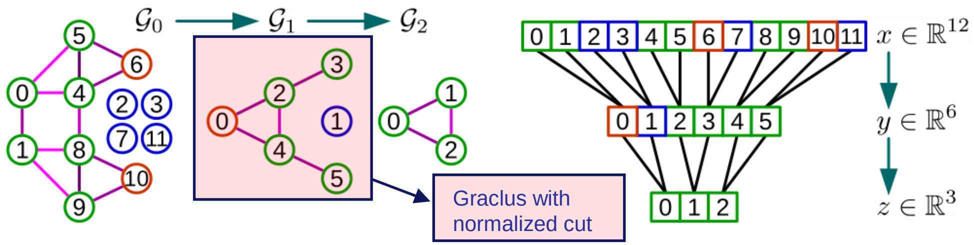
Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective."

GNN 3.0: GNN with Graph Pooling

Hierarchical Pooing



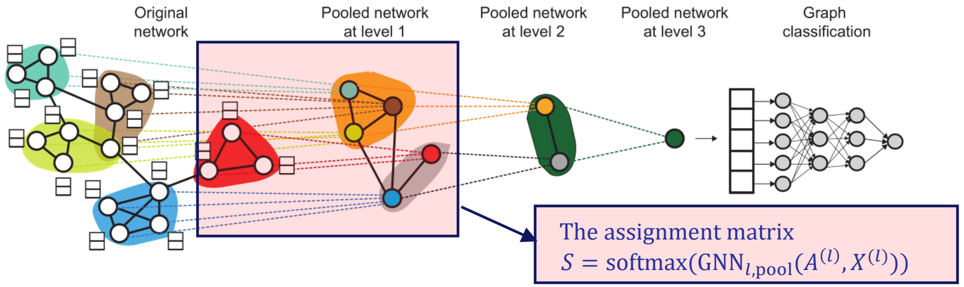
Graph Coarsening by Graph Cut [1]



Graclus with normalized cut

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

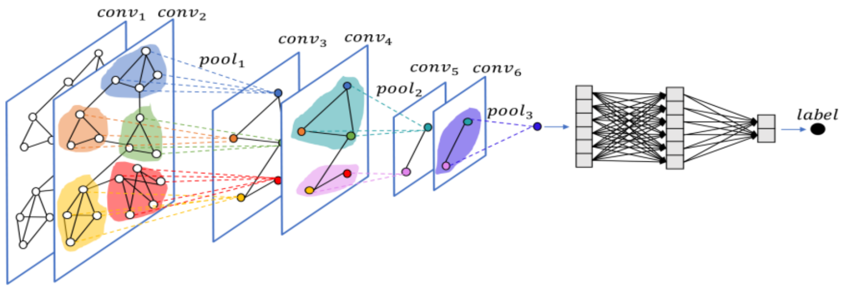
Differentiable Graph Pooling (DIFFPOOL)[2]


Original network
Pooled network at level 1
Pooled network at level 2
Pooled network at level 3
Graph classification

The assignment matrix
$S = \mathrm{softmax}(\mathrm{GNN}_{l,\mathrm{pool}}(A^{(l)}, X^{(l)}))$

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

EigenPooling [3]



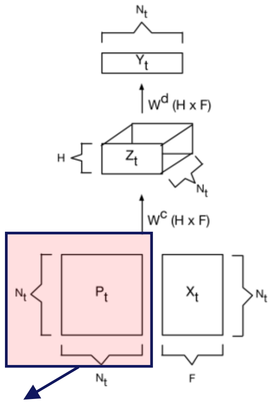
Incorporate the node features and local structures to obtain a better assignment matrix.

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

GNN 3.0: High-order GNN

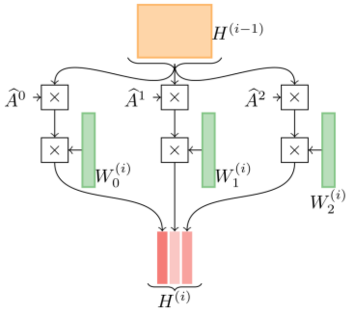
High-order GNN: extending the receptive field to encode high-order proximities in graphs.

DCNN



A $n \times H \times n$ tensor stacking the power series of the transition matrix P.

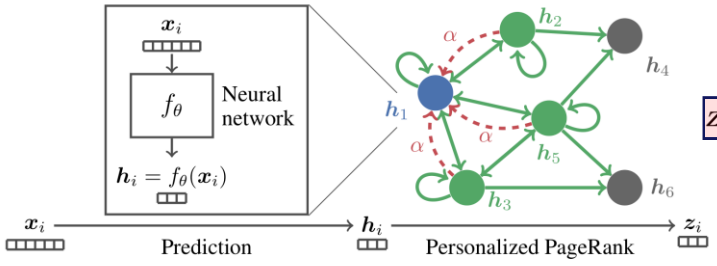
MixHop



$$H^{(l+1)} = ||_{j \in P} \sigma(\boxed{\widehat{A}^j} H^{(l)} W_j^{(l)})$$

The normalized j-order adjacency matrix

APPNP



$$\begin{aligned} \boldsymbol{Z}^{(0)} &= \boldsymbol{H} = f_\theta(\boldsymbol{X}), \\ \boxed{\boldsymbol{Z}^{(k+1)} = (1-\alpha)\hat{\hat{\boldsymbol{A}}}\boldsymbol{Z}^{(k)} + \alpha \boldsymbol{H},} \\ \boldsymbol{Z}^{(K)} &= \text{softmax}\left((1-\alpha)\hat{\hat{\boldsymbol{A}}}\boldsymbol{Z}^{(K-1)} + \alpha \boldsymbol{H}\right), \end{aligned}$$

Incorporate the personalize page rank to capture the better locality of the target node.

[1] Atwood, James, and Don Towsley. 2016 [2] Abu-El-Haija, Sami, et al. 2019 [3] Klicpera, Johannes, et.al. 2018

GNN Implementation: Message Passing Framework

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

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

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

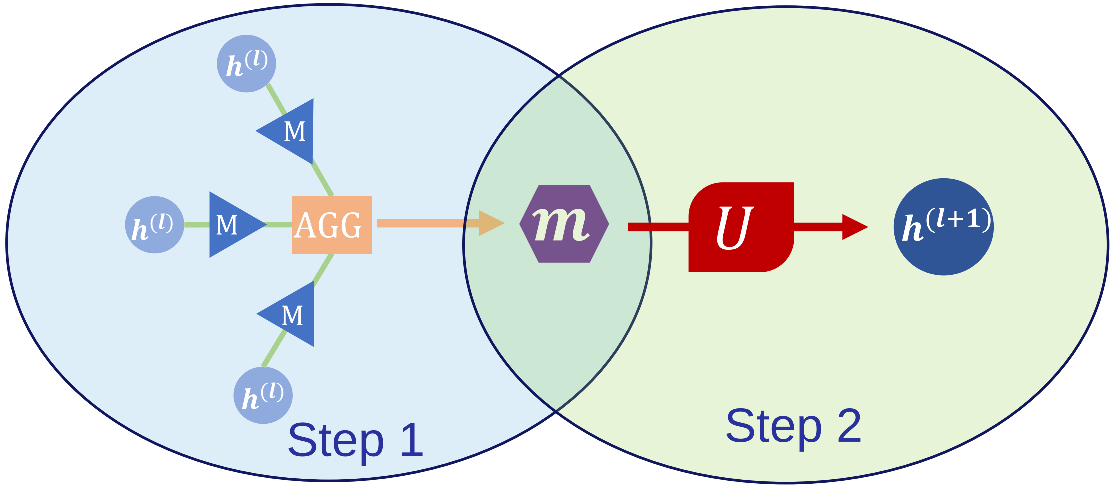
The neighborhood set of node.
E.g. 1-hop neighbors.

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

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

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

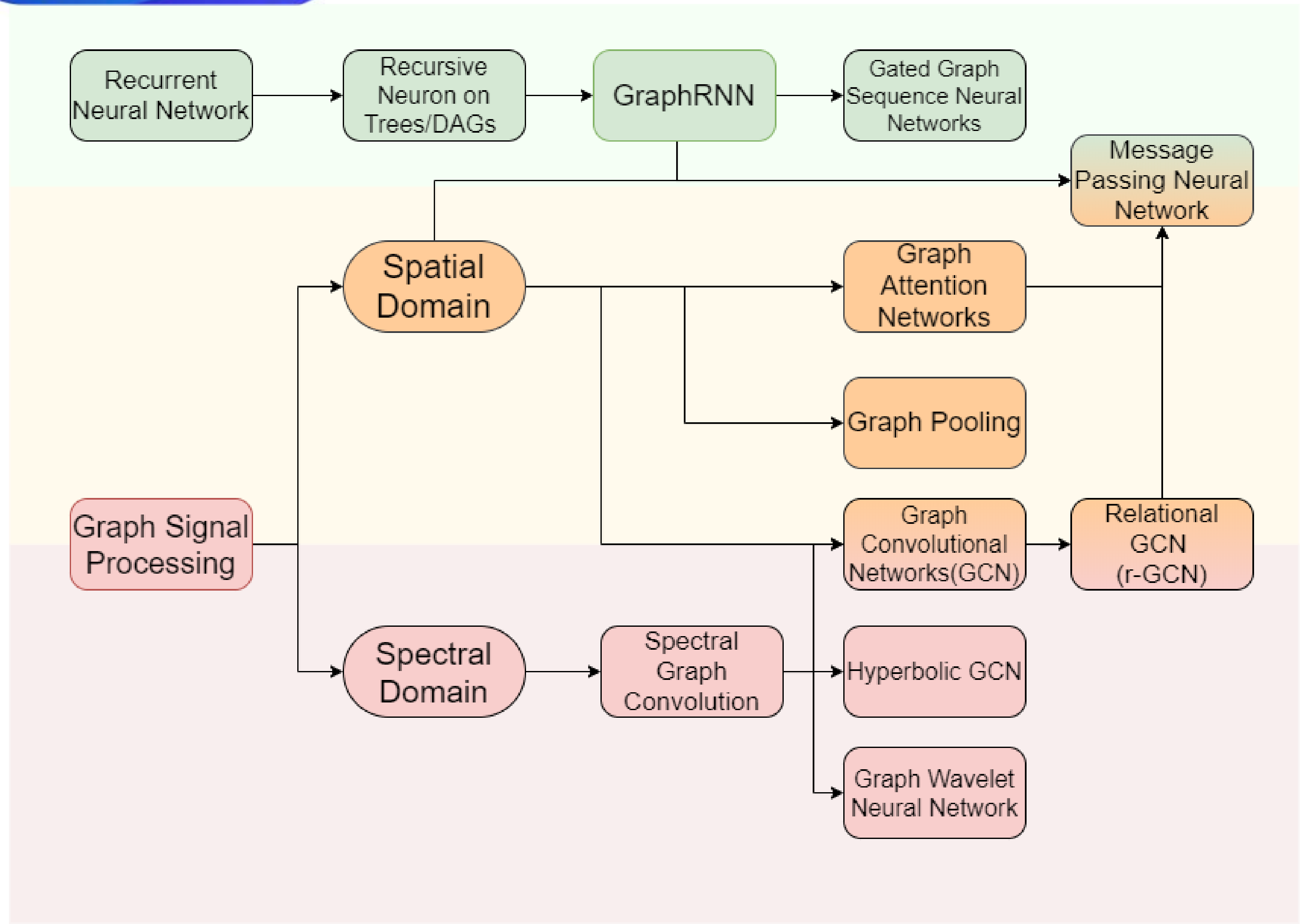
The state update function.



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

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

Summary


Recurrent Neural Network
Recursive Neuron on Trees/DAGs
GraphRNN
Gated Graph Sequence Neural Networks
Message Passing Neural Network
Spatial Domain
Graph Attention Networks
Graph Pooling
Graph Signal Processing
Graph Convolutional Networks(GCN)
Relational GCN (r-GCN)
Spectral Domain
Spectral Graph Convolution
Hyperbolic GCN
Graph Wavelet Neural Network

Advanced topics

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

Expressivity of GNNs

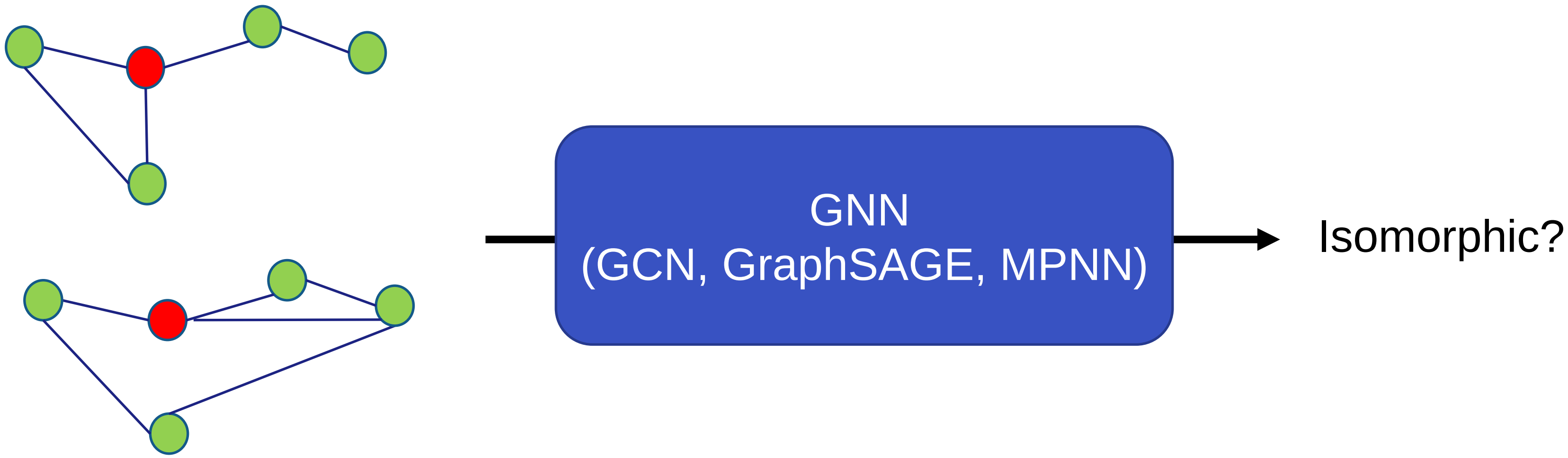
What can GNNs compute?

I. Graph Isomorphism	2. Function Approximation	3. Graph Property Detection/Optimization/Estimation
Graph classification	Predicting the chemical property of molecule	Finding the shortest path between two given nodes
HO-GNN [1]; GIN [2]	IGNs [3,4,5]	GraphMoments [6]; CPNGNN [7]; [8,9]

[1] Morris et al. 2019; [2] Xu et al. 2019; [3] Maron et al. 2019a; [4] Maron et al. 2019b; [5] Maron et al. 2019c;
[6] Dehmamy et al. 2019; [7] Sato et al. 2019; [8] Loukas 2020; [9] Garg et al. 2020;

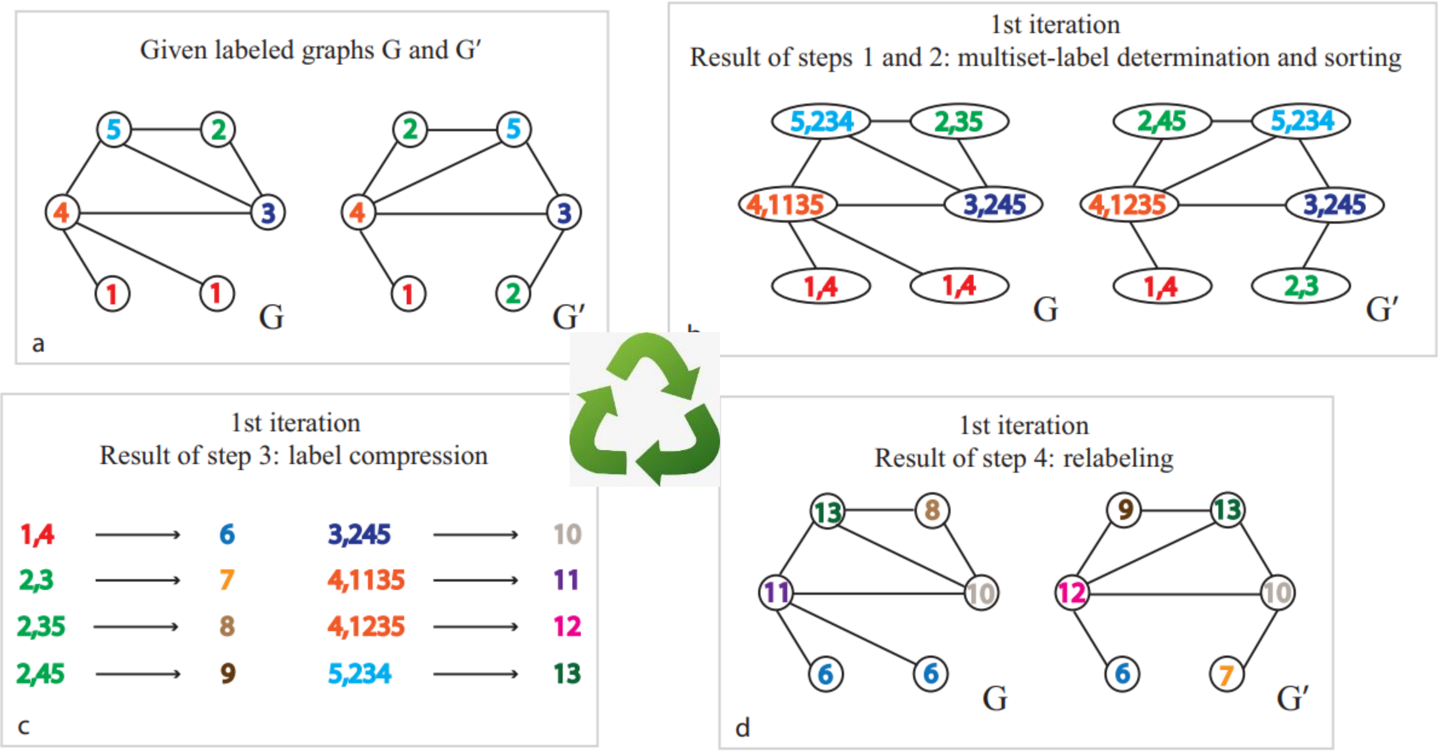
1. The Graph Isomorphism (GI) view

Given any two graphs, can GNN determine if they are isomorphic or not?



1. The Graph Isomorphism (GI) view

GI is NP problem, mostly solved by Weisfeiler-Lehman (WL) test (1968)



Figures from Shervashidze et al. 2011

For each iteration:

- Step-1: neighborhood aggregation
- Step-2: label compression by hashing
- Step-3: relabeling

1. The Graph Isomorphism (GI) view

Xu et al. (2019) and Morris et al. (2019) proved that,

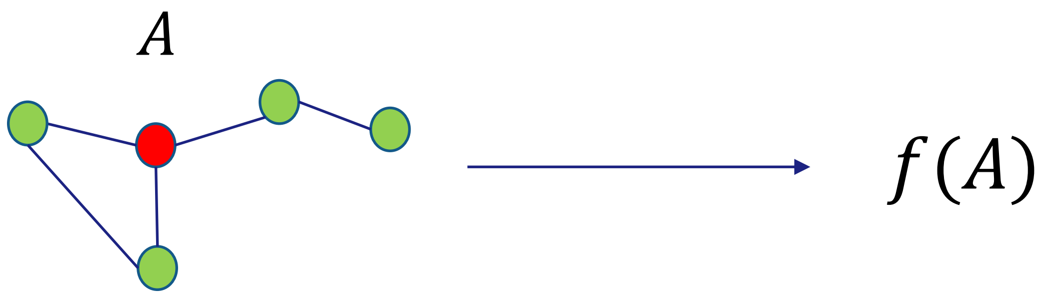
$$\mathbf{GNN} \leq \mathbf{WL}$$

Xu et al. (2019) further proved, if the aggregation\readout functions are injective,

$$\mathbf{GNN} = \mathbf{WL}$$

2. The Function Approximation (FA) view

For any function on graphs, if there is a GNN approximating it up to an arbitrary accuracy?

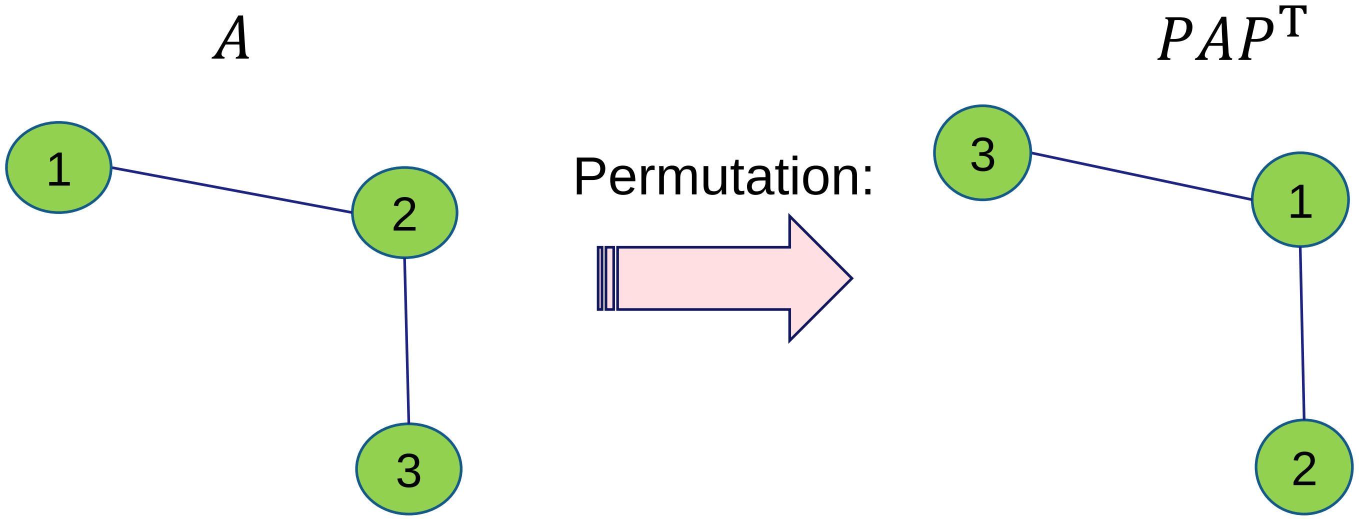


$$\exists \mathrm{GNN}, s.t. \left|\left|\mathrm{GNN} - f\right|\right| < \epsilon?$$

This kind of universality theorem has been proved for typical DNNs (Cybenko, 1989; Hornik, 1991)

2. The Function Approximation (FA) view

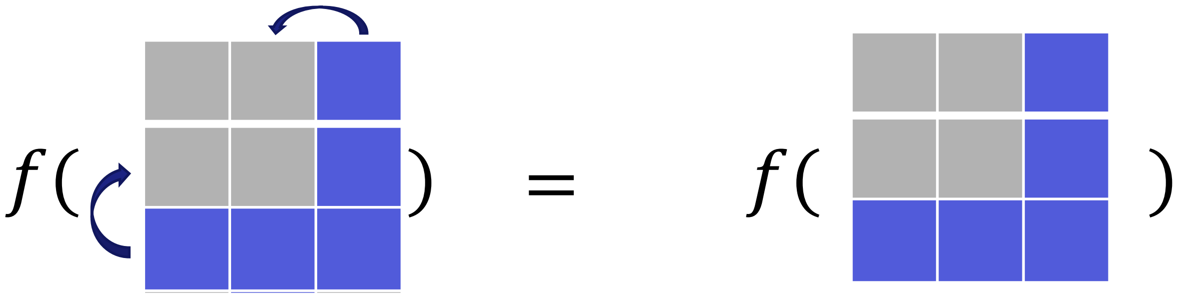
Function on graphs is symmetric w.r.t. node permutation



2. The Function Approximation (FA) view

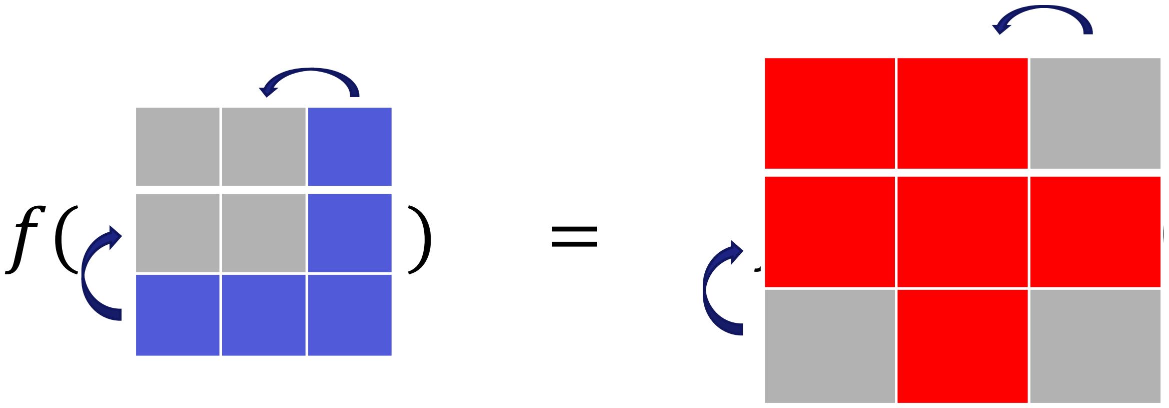
G-invariant function: Permutation does not change the output, e.g. graph classification

$$f\left(PAP^{\mathrm{T}}\right) = f(A)$$



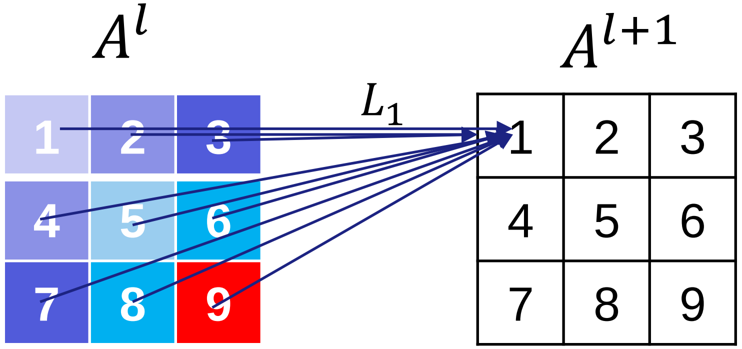
G-equivariant function: Permutation is preserved in the output, e.g. node classification

$$f\left(PAP^{\mathrm{T}}\right) = Pf(A)P^{\mathrm{T}}$$



2. The Function Approximation (FA) view

What does G-invariant/equivariant function look like?



Linear function on graphs is defined as:

$$A^{l+1} = f\left(A^l\right) = L * A^l$$

Enforcing the invariance and equivariance, we have (Maron et al. 2019a):

G-Invariant Layer:

$$P^{\otimes k}\mathrm{vec}(L) = \mathrm{vec}(L)$$

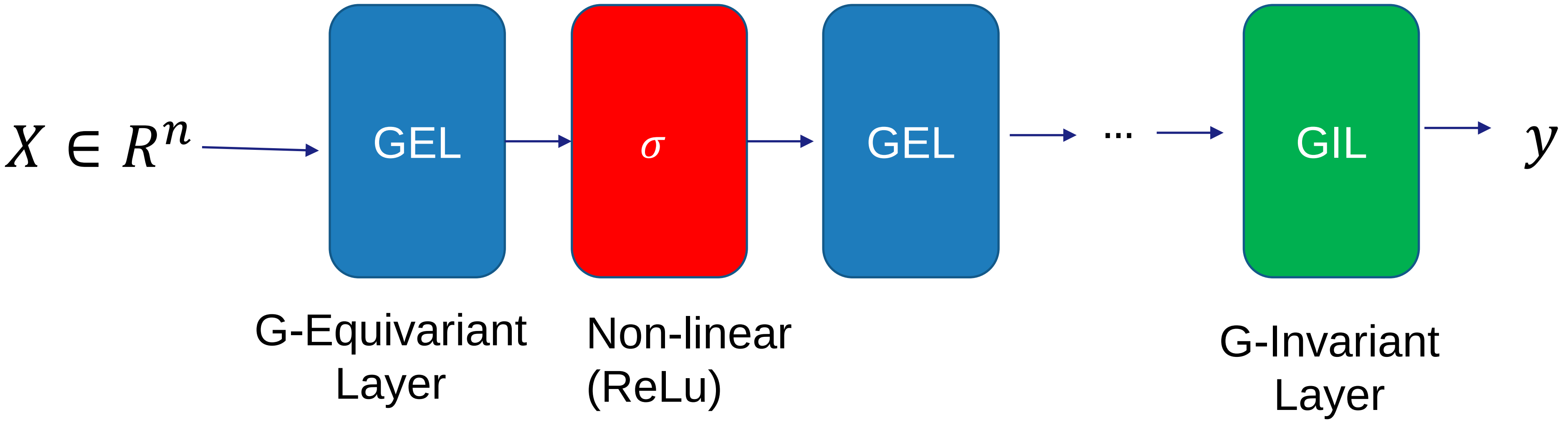
G-Equivariant Layer:

$$P^{\otimes 2k}\mathrm{vec}(L) = \mathrm{vec}(L)$$

2. The Function Approximation (FA) view

G-Invariant Network (INN)

(Maron et al. 2019b)

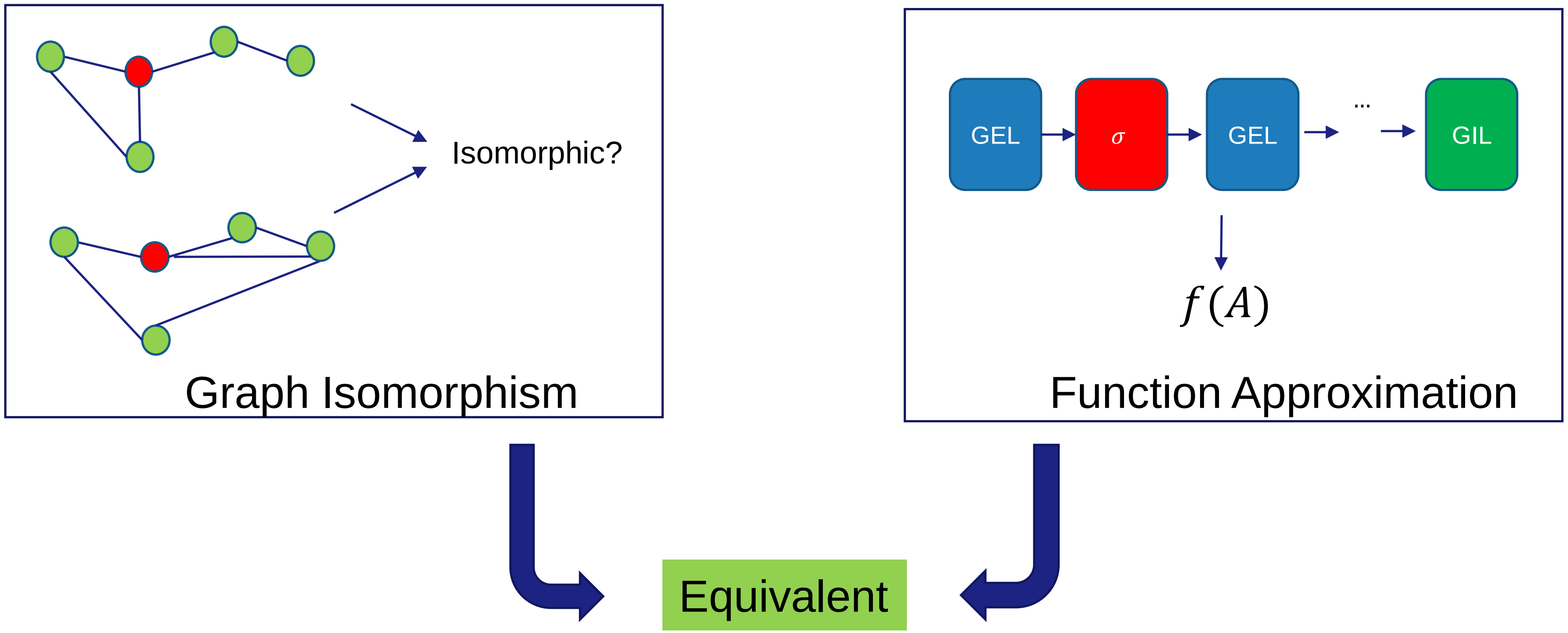


2. The Function Approximation (FA) view

[**Universality Theorem**, Maron et al. 2019b] There exists a G-Invariant network (if high-order hidden tensors are allowed) that approximates any G-invariant function to an arbitrary precision.

$$\forall \boldsymbol{f}, \forall \boldsymbol{\varepsilon} > \mathbf{0}, \exists \boldsymbol{W}, \boldsymbol{s}.\boldsymbol{t}. \left|\mathbf{INN}_{\boldsymbol{w}} - \boldsymbol{f}\right| < \boldsymbol{\varepsilon}$$

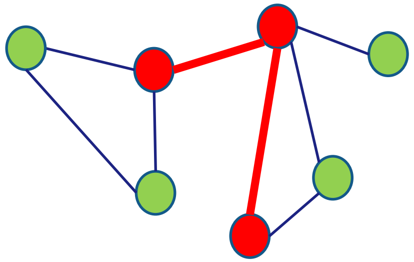
Graph Isomorphism or Function Approximation?



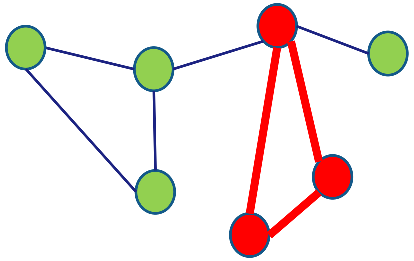
(Chen et al. 2019)

3. Not just graph identification

Are GNNs expressive enough to solve the following problems?



Finding the shortest path?



If a graph contains a circle?

Yes, if the **depth** and **width** are beyond certain bounds, with sufficiently discriminative node attributes (Loukas 2020)

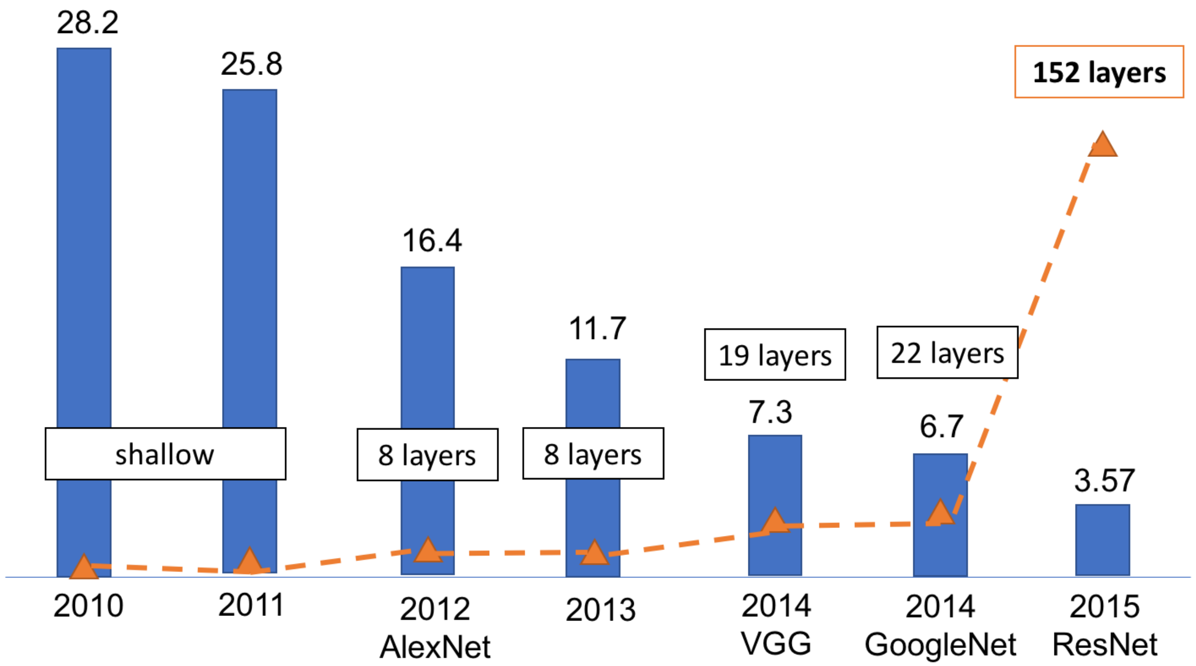
Training Deep GNNs

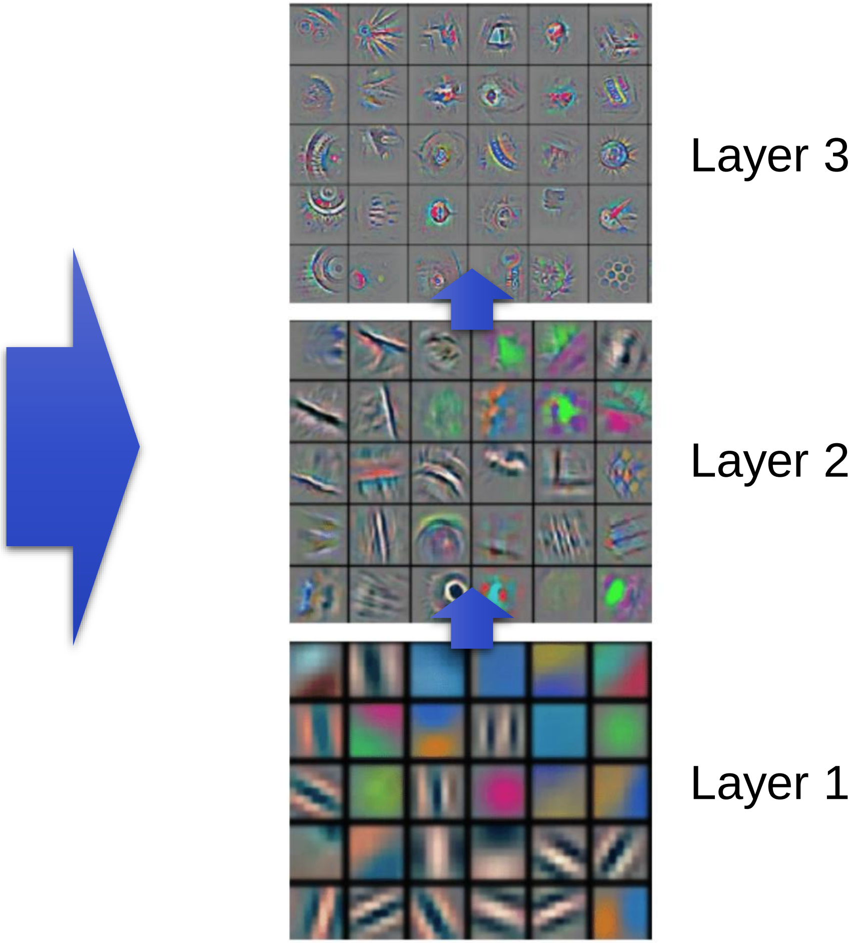
Training Deeper GNNs

- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
- How to alleviate over-smoothing?
- How to overcome training dynamics?

The Power of Deeper DNNs

- Unprecedented success of deep DNNs in computer vision
- Deeper DNNs enable larger receptive fields

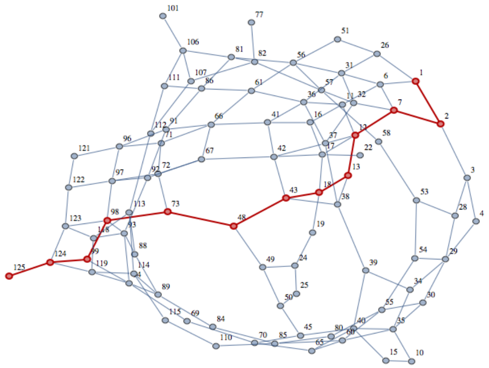




The Power of Deeper GNNs

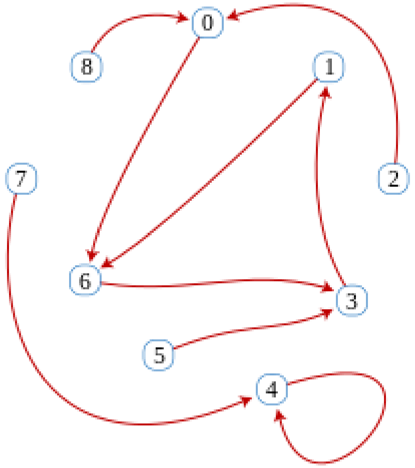
- Do GNNs need deeper structures to enable larger receptive fields, too?
- What limits the expressive power of GNNs?
  - The depth d
  - The width w
- GNNs significantly lose their power when *capacity*, dw, is restricted

Shortest Path

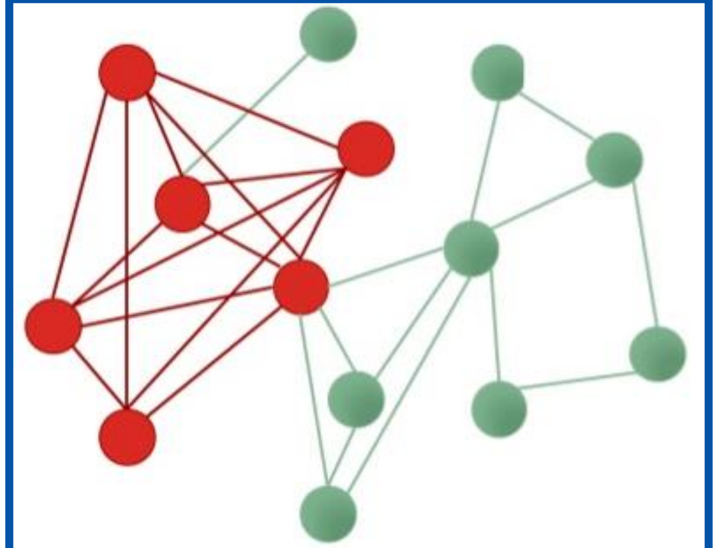


Cycle Detection

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.*

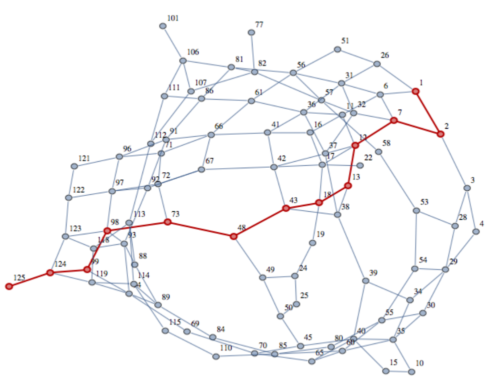
The Power of Deeper GNNs

- Do GNNs need deeper structures to enable larger receptive fields, too?
- What limits the expressive power of GNNs?
  - The depth d
  - The width w

Yes

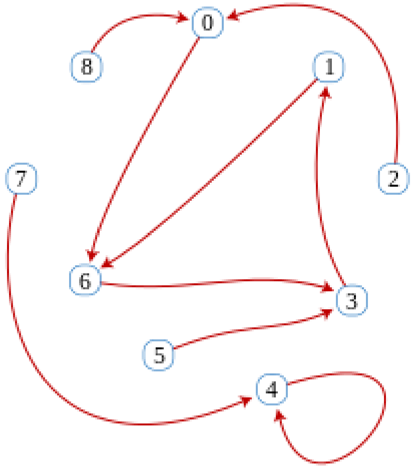
- GNNs significantly lose their power when *capacity*, dw, is restricted

Shortest Path

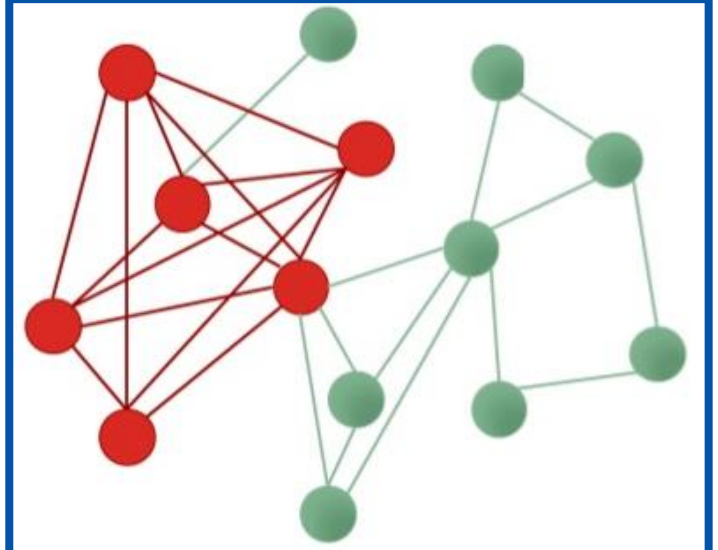


Cycle Detection

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



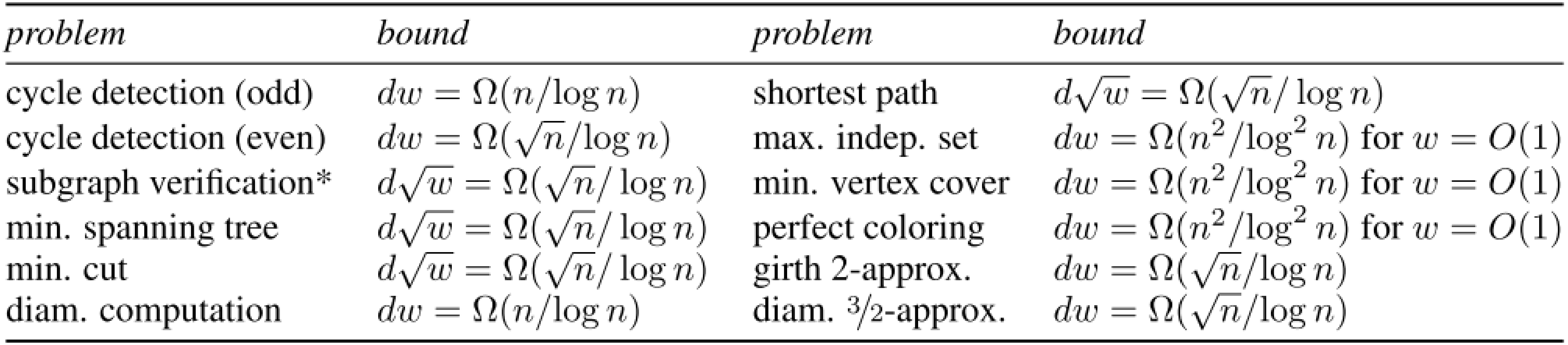
Subgraph



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

The Power of Deeper 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.	$dw = \Omega(\sqrt{n}/\log n)$

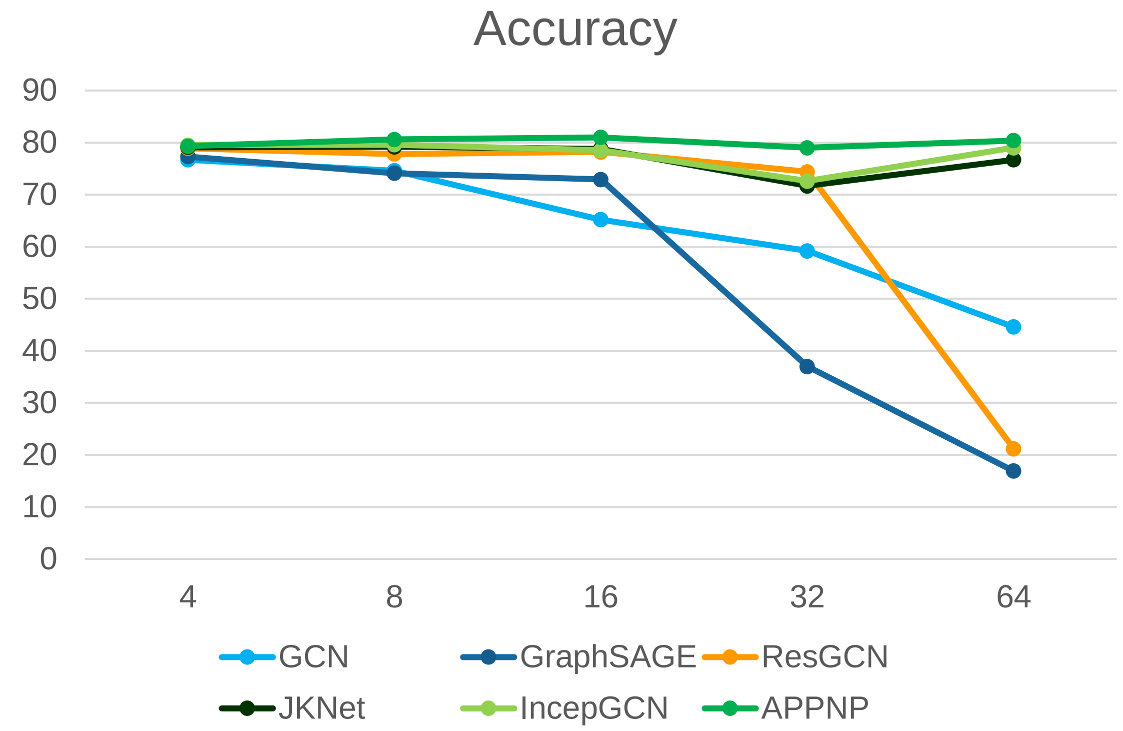
Loukas, Andreas. "What graph neural networks cannot learn: depth vs width." *International Conference on Learning Representations. 2020.*

Training Deeper GNNs

- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
  - **GCN**: Basic GCN
  - **GraphSAGE**: GCN with improved **aggregation**
  - **JKNet**: leverage idea from **DenseNet**
  - **ResGCN**: leverage idea from **ResNet**
  - **IncepGCN**: leverage idea from **Inception-v3**
  - **APPNP**: leverage idea from **PageRank**
- What impedes GNNs to go deeper?
- How to alleviate over-smoothing?
- How to overcome training dynamics?

GNNs are Shallow

- But can they really go deeper? 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**
JKNet	79.1	78.8	76.7
IncepGCN	79.5	78.5	79
APPNP	79.3	81.0	80.4

- What is the underlying reason of going deeper?

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

Training Deeper GNNs

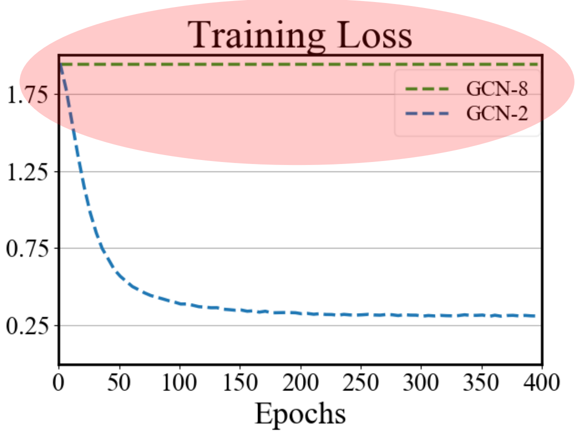
- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
  - **Over-smoothing (Graph Specific)**
  - Overfitting (Common)
  - Training dynamics (Common)
- How to alleviate over-smoothing?
- How to overcome training dynamics?

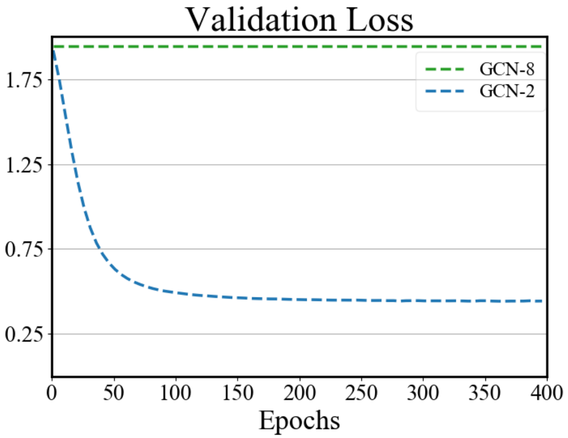
Training Deeper GNNs

- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
  - **Over-smoothing (Graph Specific)**
  - Overfitting (Common)
  - Training dynamics (Common)
- How to alleviate over-smoothing?
- How to overcome training dynamics?

Over-Smoothing

- GNNs suffers from over-smoothing

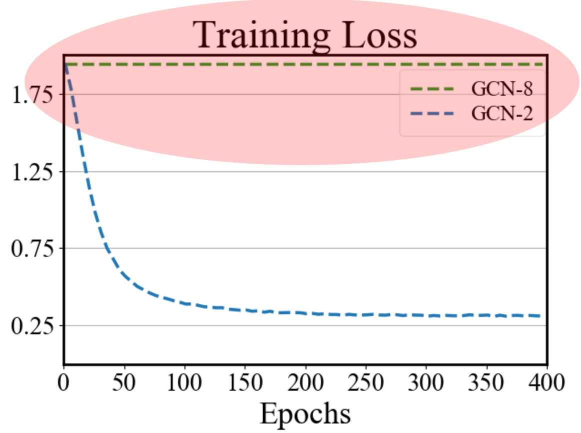


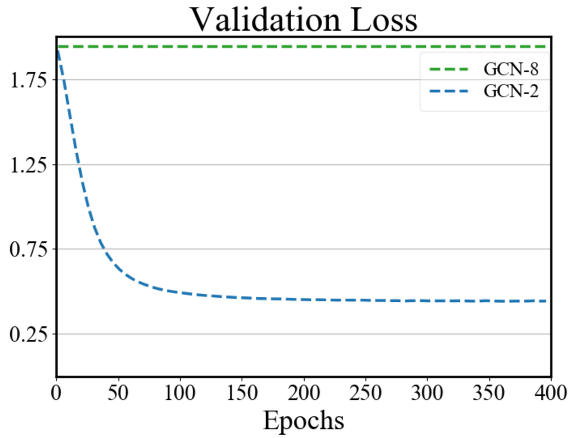


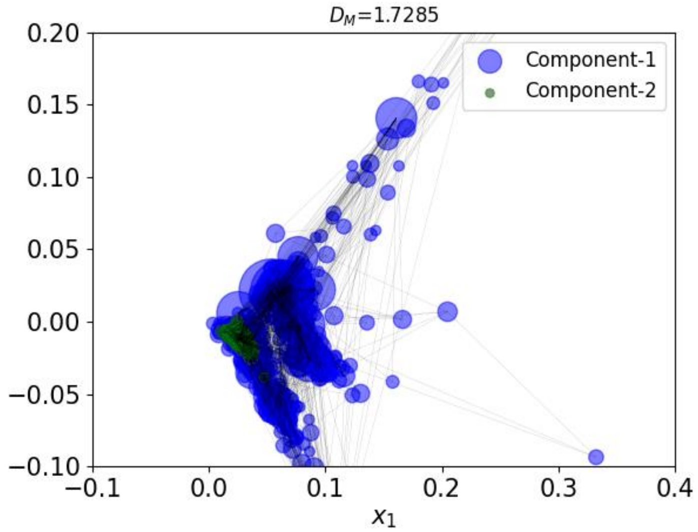
Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

Over-Smoothing

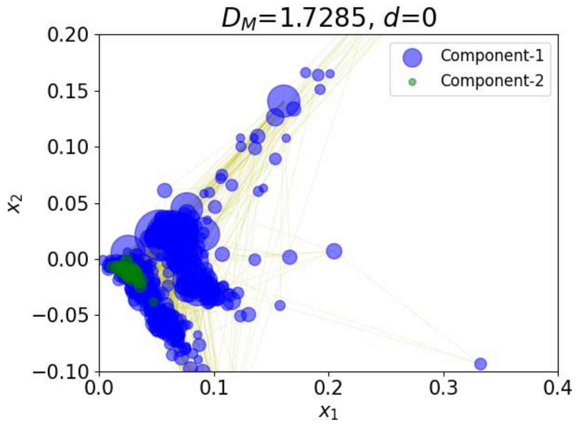
- GNNs suffers from over-smoothing



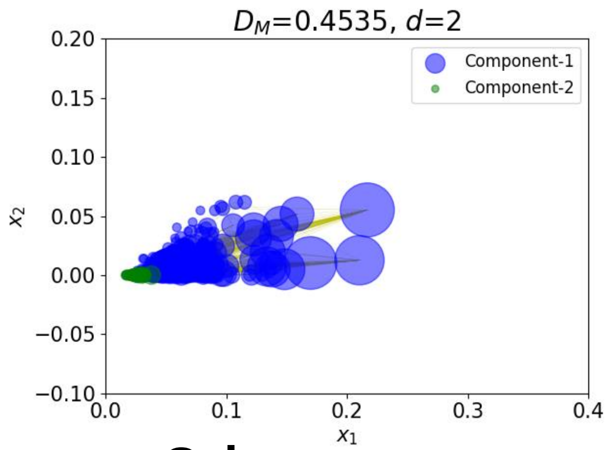




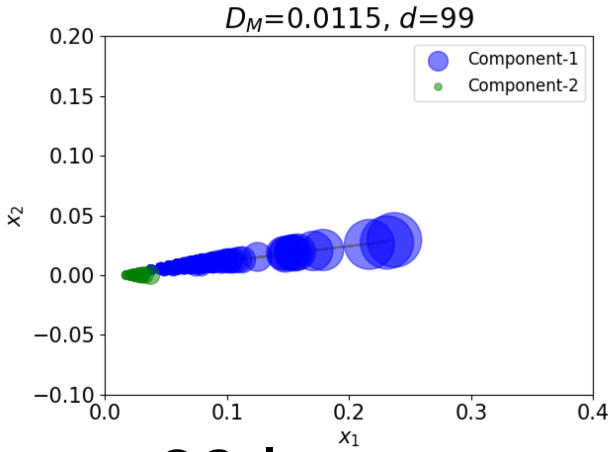
- As the layers go deeper, the hidden variables converge to a subspace



Initial



2 layers



99 layers

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

Over-Smoothing of Linear GCN

- Why GCN works?
  - Laplacian smoothing →symmetric Laplacian smoothing
  - The **weighted** average of itself and its **neighbors**→ the new feature of a vertex

$$\tilde{L} = \tilde{D} - \tilde{A}$$

$$\tilde{D}^{-1}\tilde{L} \to \tilde{D}^{-\frac{1}{2}}\tilde{L}\tilde{D}^{-\frac{1}{2}}$$

$$\gamma = 1$$

Laplacian Smoothing

$$Y = \left(I - \gamma\tilde{D}^{-1}\tilde{L}\right)X$$

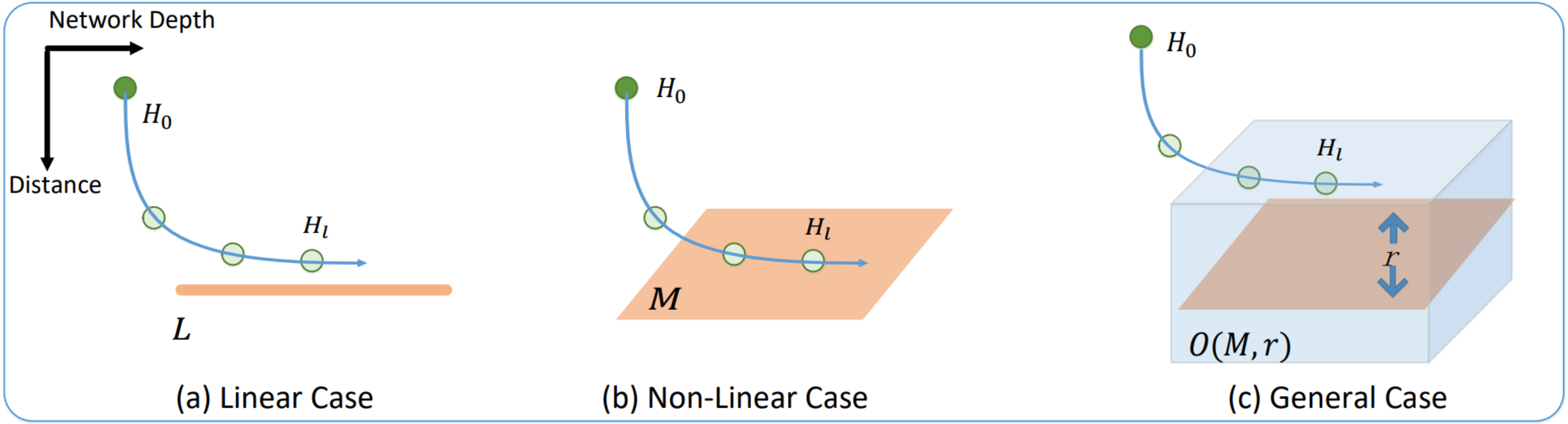
GCNs

$$Y = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}XW$$

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

Over-Smoothing

- When GCNs fail?
  - H_L converges to a certain point with linear activation
  - H_L converges to a certain subspace $\mathcal{M}$ with ReLU activation
  - H_L converges to a certain sub-cube $O(\mathcal{M}, r)$ with ReLU and bias



(a) Linear Case (b) Non-Linear Case (c) General Case

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

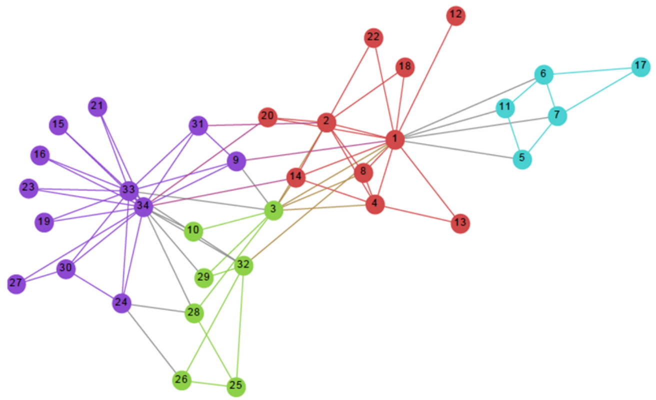
Over-Smoothing of Linear GCN

- When GCNs fail?
  - H_L converges to a certain point with linear activation

l-step Random Walk

Probability of walking

$$Y = \left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\right)^{l} P_0 \text{ where } p_{ij} = \boxed{1/d(i)} \text{ if } (i,j) \in \mathcal{E}$$



Random Walks on Graph

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

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

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

Over-Smoothing of Linear GCN

- When GCNs fail?
  - H_L converges to a certain point with linear activation

l-step Random Walk

$$Y = \left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}\right)^{l} P_0 \text{ where } p_{ij} = 1/d(i) \text{ if } (i,j) \in \mathcal{E}$$

l-layer GCNs

$$Y = \left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}\right)^{l} \boxed{XW}$$

Learnable Probability

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

Over-Smoothing of Linear GCN

- When GCNs fail?
  - H_L converges to a certain point with linear activation

l-step Random Walk

$$Y = \left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}\right)^{l} P_0 \text{ where } p_{ij} = 1/d(i) \text{ if } (i,j) \in \mathcal{E}$$

l-layer GCNs

$$Y = \left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}\right)^{l} XW$$

Eigen decomposition

$$Y = \sum_{i=1}^{n} \widetilde{D}^{-\frac{1}{2}}\left(\lambda_i u_i u_i^{\top}\right)^{l} XW$$

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

Over-Smoothing of Linear GCN

- Rewrite eigen decomposition

Eigen decomposition

$$\tilde{D}^{-\frac{1}{2}}(\lambda_1 u_1 u_1^\top)^l XW + \cdots \tilde{D}^{-\frac{1}{2}}(\lambda_m u_m u_m^\top)^l XW + \tilde{D}^{-\frac{1}{2}}(\lambda_{m+1} u_{m+1} u_{m+1}^\top)^l XW + \cdots \tilde{D}^{-\frac{1}{2}}(\lambda_n u_n u_n^\top)^l XW$$

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

Over-Smoothing of Linear GCN

- Rewrite eigen decomposition

Eigen decomposition

$$\tilde{D}^{-\frac{1}{2}}(\lambda_1 u_1 u_1^\top)^l XW + \cdots \tilde{D}^{-\frac{1}{2}}(\lambda_m u_m u_m^\top)^l XW + \tilde{D}^{-\frac{1}{2}}(\lambda_{m+1} u_{m+1} u_{m+1}^\top)^l XW + \cdots \tilde{D}^{-\frac{1}{2}}(\lambda_n u_n u_n^\top)^l XW$$

- Suppose graph $\mathcal{g}$ 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. "Deeper insights into graph convolutional networks for semi-supervised learning." *AAAI* 2018.

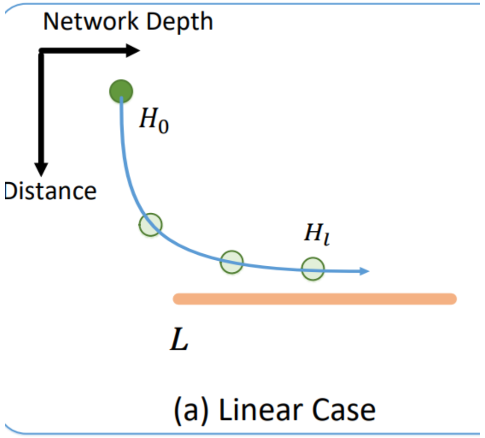
Over-Smoothing of Linear GCN

- Rewrite eigen decomposition

Eigen decomposition

$$\tilde{D}^{-\frac{1}{2}}(\lambda_1 u_1 u_1^\top)^l XW + \cdots \tilde{D}^{-\frac{1}{2}}(\lambda_m u_m u_m^\top)^l XW + \tilde{D}^{-\frac{1}{2}}(\lambda_{m+1} u_{m+1} u_{m+1}^\top)^l XW + \cdots \tilde{D}^{-\frac{1}{2}}(\lambda_n u_n u_n^\top)^l XW$$

- Suppose graph $\mathcal{g}$ has m connected components. It indicates



(a) Linear Case

Eigenvalues

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

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

Eigen decomposition

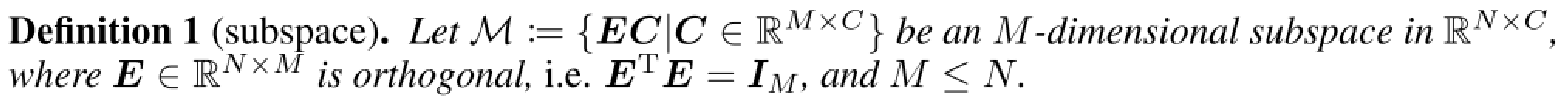
$$\lim_{l\to\infty} \tilde{D}^{-\frac{1}{2}}\underbrace{(\lambda_1 u_1 u_1^\top)^l}_{\boxed{1}} XW + \cdots \tilde{D}^{-\frac{1}{2}}\underbrace{(\lambda_m u_m u_m^\top)^l}_{\boxed{1}} XW + \tilde{D}^{-\frac{1}{2}}\underbrace{(\lambda_{m+1} u_{m+1} u_{m+1}^\top)^l}_{\boxed{0}} XW + \cdots \tilde{D}^{-\frac{1}{2}}\underbrace{(\lambda_n u_n u_n^\top)^l}_{\boxed{0}} XW$$

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

Over-Smoothing of Non-Linear GCN

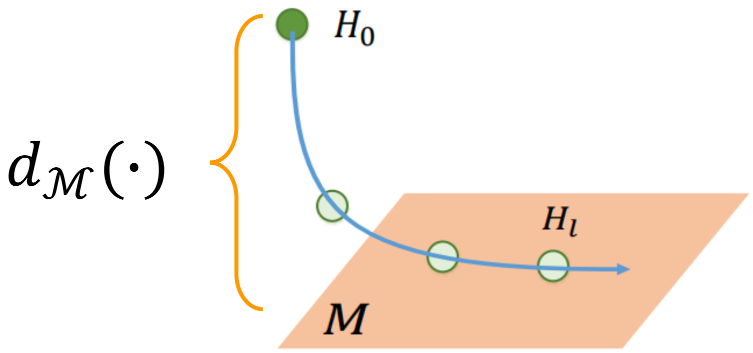
- H_L converges to a certain subspace $\mathcal{M}$ with ReLU activation
  - We define a subspace $\mathcal{M}$ first

$\mathcal{M}$ subspcae



Definition 1 (subspace). *Let* $\mathcal{M} := \{\boldsymbol{E}\boldsymbol{C} | \boldsymbol{C} \in \mathbb{R}^{M \times C}\}$ *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* $M \leq N$.

- $d_{\mathcal{M}}(\cdot)$ refers to the distance to the subspace $\mathcal{M}$



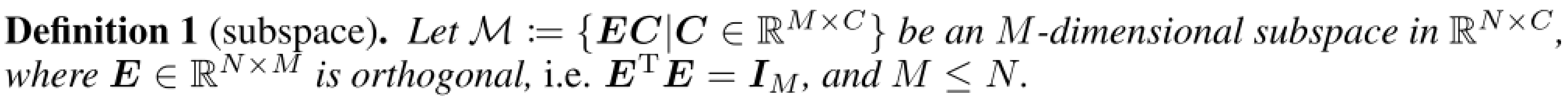
(b) Non-Linear Case

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

Over-Smoothing of Non-Linear GCN

- H_L converges to a certain subspace $\mathcal{M}$ with ReLU activation
  - We define a subspace $\mathcal{M}$ first

$\mathcal{M}$ subspcae



Definition 1 (subspace). *Let* $\mathcal{M} := \{\boldsymbol{E}\boldsymbol{C} | \boldsymbol{C} \in \mathbb{R}^{M\times C}\}$ *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* $M \leq N$.

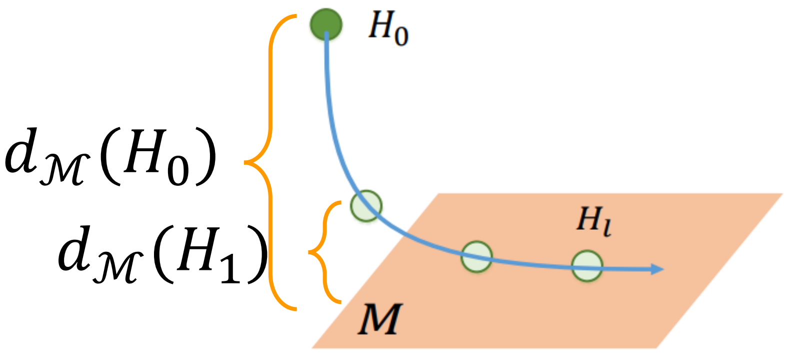
- $d_\mathcal{M}(\cdot)$ refers to the distance to the subspace $\mathcal{M}$
- $d_\mathcal{M}(\cdot)$ converges as the layers go deeper

Convergence

$$d_\mathcal{M}(H_{l+1}) = d_\mathcal{M}\left(\sigma(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H_lW_l)\right) \leq s_l\lambda_{m+1}d_\mathcal{M}(H_l)$$

$\lambda_{m+1} < 1$ is the largest non-one eigenvalue

$s_l \leq 1$ is the maximum singular value of W_l



(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 $\tilde{A}$

$$d_{\mathcal{M}}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\right) \leq \lambda_{m+1} d_{\mathcal{M}}(X), \qquad \lambda_{m+1} < 1$$

Convergence

$$d_{\mathcal{M}}(H_{l+1}) = d_{\mathcal{M}}\left(\sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H_l W_l)\right) \leq \lambda_{m+1} s_l d_{\mathcal{M}}(H_l)$$

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 $\tilde{A}$

$$d_{\mathcal{M}}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\right) \leq \lambda_{m+1} d_{\mathcal{M}}(X), \qquad \lambda_{m+1} < 1$$

Convergence of W

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

Convergence

$$d_{\mathcal{M}}(H_{l+1}) = d_{\mathcal{M}}\left(\sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H_l W_l)\right) \leq \lambda_{m+1} s_l d_{\mathcal{M}}(H_l)$$

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 $\tilde{A}$

$$d_{\mathcal{M}}\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X\right) \leq \lambda_{m+1} d_{\mathcal{M}}(X), \qquad \lambda_{m+1} < 1$$

Convergence of W

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

Convergence of ReLU

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

Convergence

$$d_{\mathcal{M}}(H_{l+1}) = d_{\mathcal{M}}\left(\sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H_l W_l\right)\right) \leq \lambda_{m+1} s_l d_{\mathcal{M}}(H_l)$$

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

Over-Smoothing of GCNs with bias

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

GCNs with bias

$$H_{l+1} = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H_l W_l + b_l\right)$$

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

$$H_{l+1} = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H_l W_l + b_l\right)$$

Convergence of bias

$$d_{\mathcal{M}}(H_{l+1}) \le \lambda_{m+1} s_l d_{\mathcal{M}}(H_l) + d_{\mathcal{M}}(b_l)$$

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

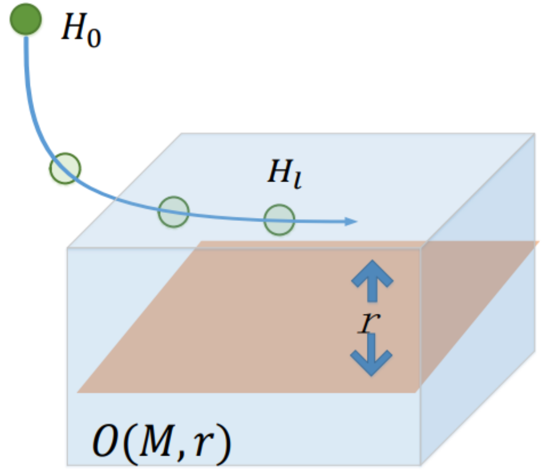
$$H_{l+1} = \sigma\left(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H_l W_l + b_l\right)$$

Convergence of bias

$$d_{\mathcal{M}}(H_{l+1}) \le \lambda_{m+1} s_l d_{\mathcal{M}}(H_l) + d_{\mathcal{M}}(b_l)$$

GCN with bias

$$\lim_{l\to+\infty} d_{\mathcal{M}}(H_l) \le \begin{cases} \dfrac{d_{\mathcal{M}}(b_{max})}{1-\lambda_{m+1}s_{max}}, \text{with } \lambda_{m+1}s_l < 1 \\ \infty, \text{with } \lambda_{m+1}s_l > 1 \end{cases}$$



(c) General Case

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

Overall Over-Smoothing?

Linear GCN

$$\lim_{l\to+\infty}\left(I-\gamma L_{sym}\right)^{l}XW=\widetilde{D}^{-\frac{1}{2}}\left[\mathbf{1}^{i}(XW)^{i}\right]_{i=1}^{m}$$

Non-linear GCN

$$\lim_{l\to+\infty}d_{\mathcal{M}}(H_l)\leq\left(\lambda_{m+1}s_{max}\right)^{l}d_{\mathcal{M}}(X)=\begin{cases}0,\text{with }\lambda_{m+1}s_l<1\\ \infty,\text{with }\lambda_{m+1}s_l>1\end{cases}$$

GCN with bias

$$\lim_{l\to+\infty}d_{\mathcal{M}}(H_l)\leq\begin{cases}\dfrac{d_{\mathcal{M}}(b_{max})}{1-\lambda_{m+1}s_{max}},\text{with }\lambda_{m+1}s_l<1\\ \infty,\text{with }\lambda_{m+1}s_l>1\end{cases}$$

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

Universal Over-Smoothing

General Case

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

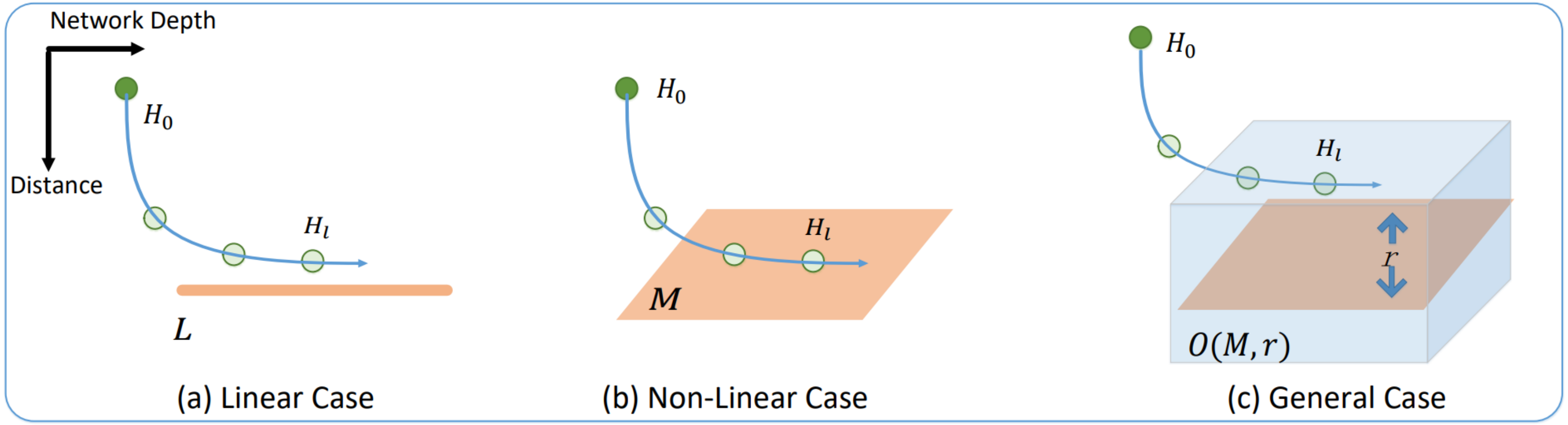
Basic GCN

Non-linear GCN

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

GCN with bias

$$v = s_{max}\lambda_{m+1}$$
$$r = \frac{d_{\mathcal{M}}(b_{max})}{1 - v}$$



(a) Linear Case (b) Non-Linear Case (c) General Case

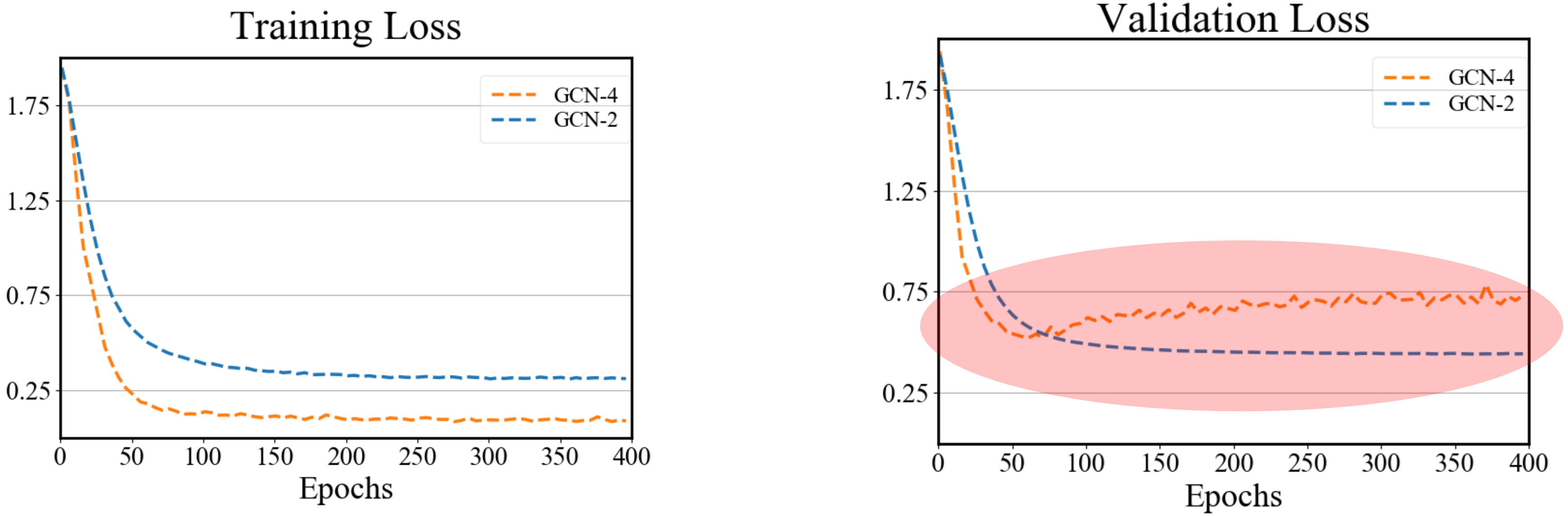
Huang, Wenbing, et al. "Tackling Over-Smoothing for General Graph Convolutional Networks." arXiv preprint arXiv: 2008.09864, 2020

Training Deeper GNNs

- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
  - **Over-smoothing (Graph Specific)**
  - Overfitting (Common)
  - Training dynamics (Common)
- How to alleviate over-smoothing?
- How to overcome training dynamics?

Overfitting

- GNNs suffer from Overfitting



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

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

Training Deeper GNNs

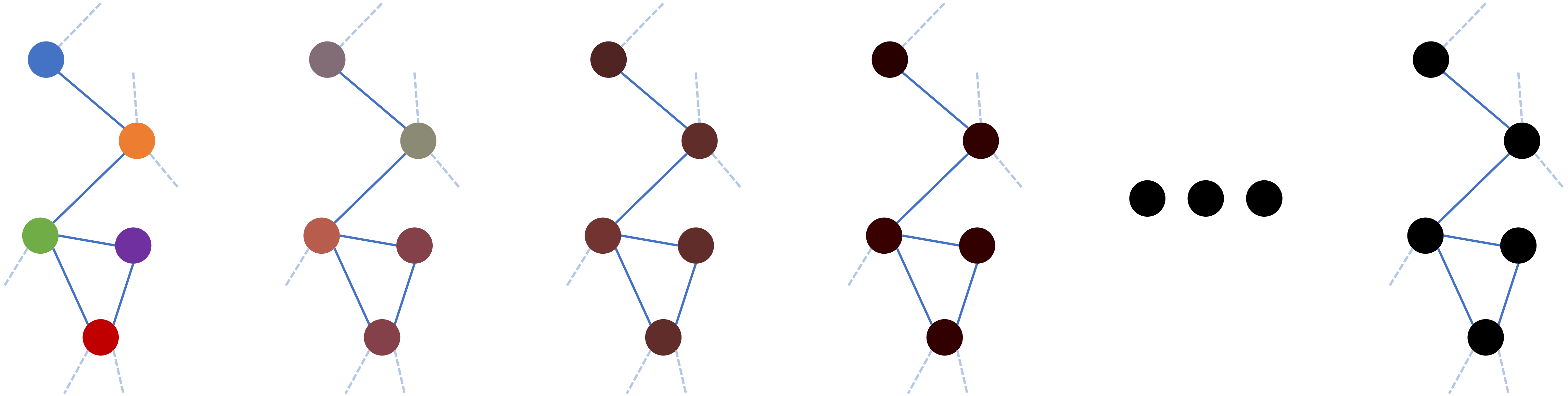
- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
  - **Over-smoothing (Graph Specific)**
  - Overfitting (Common)
  - Training dynamics (Common)
- How to alleviate over-smoothing?
- How to overcome training dynamics?

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 deeper because $s_{1\ldots l}\lambda_{m+1} < 1$



Training Deeper GNNs

- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
  - **Over-smoothing (Graph Specific)**
  - Overfitting (Common)
  - Training dynamics (Common)
- How to alleviate over-smoothing?
  - Deal with adjacency matrix
  - Deal with weights
- How to overcome training dynamics?

Over-smoothing Layer

- When does over-smoothing happen?
  - Let's take basic GCN as example: $v = \lambda_{m+1} s_{max}, r = 0$

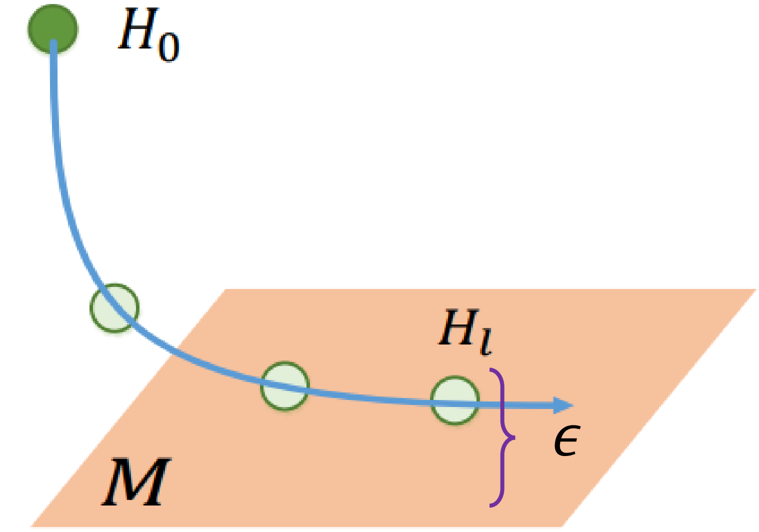
Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

Over-smoothing Layer

- When does over-smoothing happen?
  - Let's take basic GCN as example: $v = \lambda_{m+1} s_{max}, r = 0$

ϵ-smoothing

$$d_{\mathcal{M}}(H_l) \leq (\lambda_{m+1} s_{max})^l d_{\mathcal{M}}(X) < \epsilon, \forall l \geq L$$



(b) Non-Linear Case

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

Over-smoothing Layer

- When does over-smoothing happen?
  - Let’s take basic GCN as example: $v = \lambda_{m+1} s_{max}, r = 0$

ϵ-smoothing

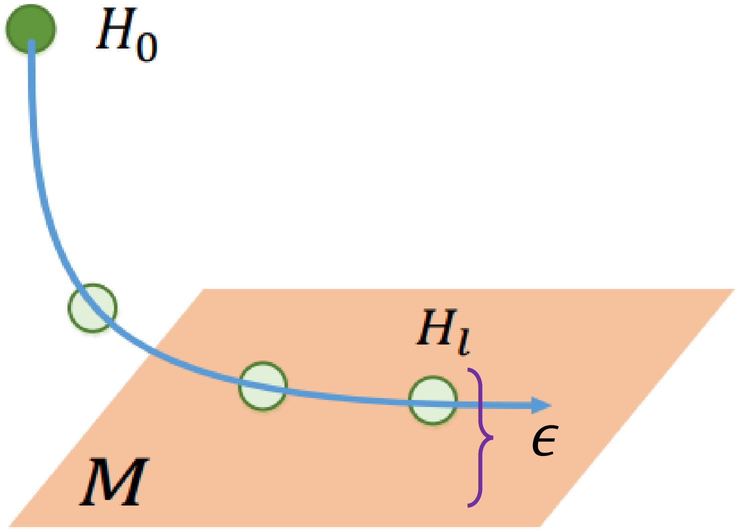
$$d_{\mathcal{M}}(H_l) \le (\lambda_{m+1} s_{max})^l d_{\mathcal{M}}(X) < \epsilon, \forall l \ge L$$

ϵ-smoothing layer

$$l^*(\mathcal{M}, \epsilon) := \left\{ \min_l d_{\mathcal{M}}(H_l) < \epsilon \right\}$$

Relaxed ϵ-smoothing layer

$$\hat{l}(\mathcal{M}, \epsilon) = \left\lceil \frac{\log\left(\frac{\epsilon}{d_{\mathcal{M}}(X)}\right)}{\log(\lambda_{m+1} s_{max})} \right\rceil$$



(b) Non-Linear Case

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

Over-smoothing Layer

- How to alleviate over-smoothing?

Relaxed ϵ-smoothing layer

$$\hat{l}(\mathcal{M},\epsilon) = \left\lceil \frac{\log\left(\frac{\epsilon}{d_{\mathcal{M}}(X)}\right)}{\log(\underbrace{\boxed{\lambda_{m+1}}}_{\text{Adjacency Matrix}}\underbrace{\boxed{s_{max}}}_{\text{Weights}})} \right\rceil$$

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

Training Deeper GNNs

- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
- How to alleviate over-smoothing?
  - Deal with adjacency matrix
  - Deal with weights
- How to overcome training dynamics?

Alleviate Over-Smoothing by Adjacency Matrix

Relaxed ϵ-smoothing layer

$$\hat{l}(\mathcal{M},\epsilon) = \left\lceil \frac{\log\left(\frac{\epsilon}{d_{\mathcal{M}}(X)}\right)}{\log(\boxed{\lambda_{m+1}}\, s_{max})} \right\rceil$$

Adjacency Matrix

- How adjacency matrix affects on over-smoothing?

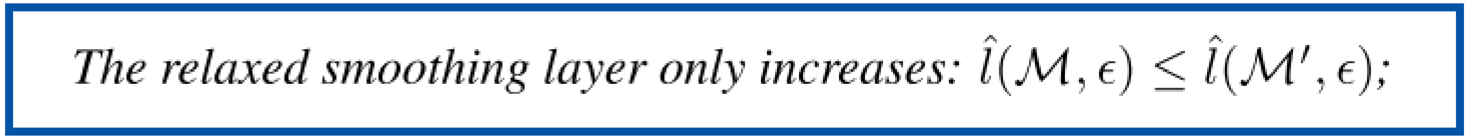
$$\left.\begin{array}{r} \lambda_{m+1}\uparrow \Rightarrow \log(\lambda_{m+1}s_{max})\uparrow \\ \lambda_{m+1}s_{max} < 1 \end{array}\right\} \Rightarrow \hat{l}(\mathcal{M},\epsilon)\uparrow$$

- So how to increase λ_{m+1}?

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

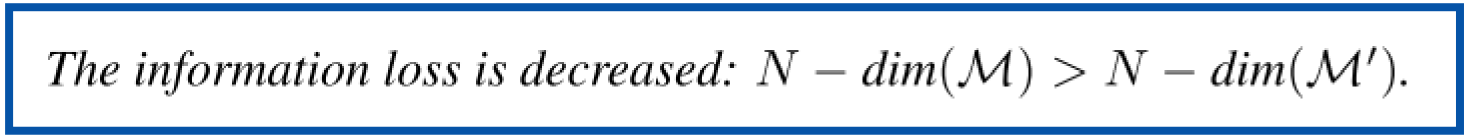
Alleviate Over-Smoothing by Adjacency Matrix

- So how to increase λ_{m+1}? Drop Edges!
- When drop edges:
  - The spread speed of the information is decreased

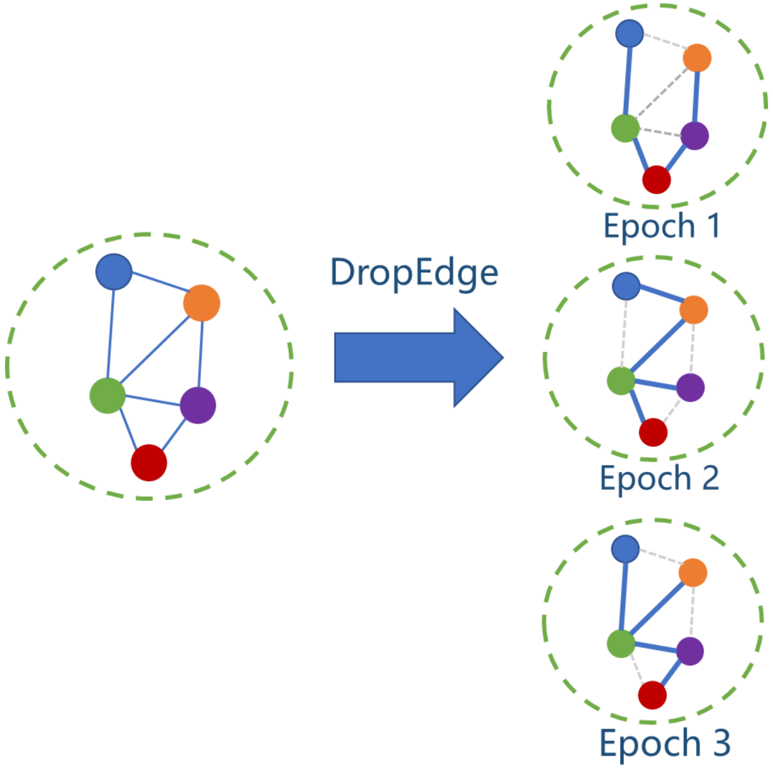


The relaxed smoothing layer only increases: $\hat{l}(\mathcal{M}, \epsilon) \leq \hat{l}(\mathcal{M}', \epsilon)$;

  - The dimension of subspace increases as the number of connected components increases



The information loss is decreased: $N - dim(\mathcal{M}) > N - dim(\mathcal{M}')$.



Rong, Yu, et al. “Dropedge: Towards deep graph convolutional networks on node classification.” ICLR *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)

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

Training Deeper GNNs

- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
- How to alleviate over-smoothing?
  - Deal with adjacency matrix
  - Deal with weights
- How to overcome training dynamics?

Alleviate Over-Smoothing by Weights

Relaxed ϵ-smoothing layer

$$\hat{l}(\mathcal{M},\epsilon) = \left\lceil \frac{\log\left(\frac{\epsilon}{d_{\mathcal{M}}(X)}\right)}{\log(\lambda_{m+1}\boxed{s_{max}})} \right\rceil$$

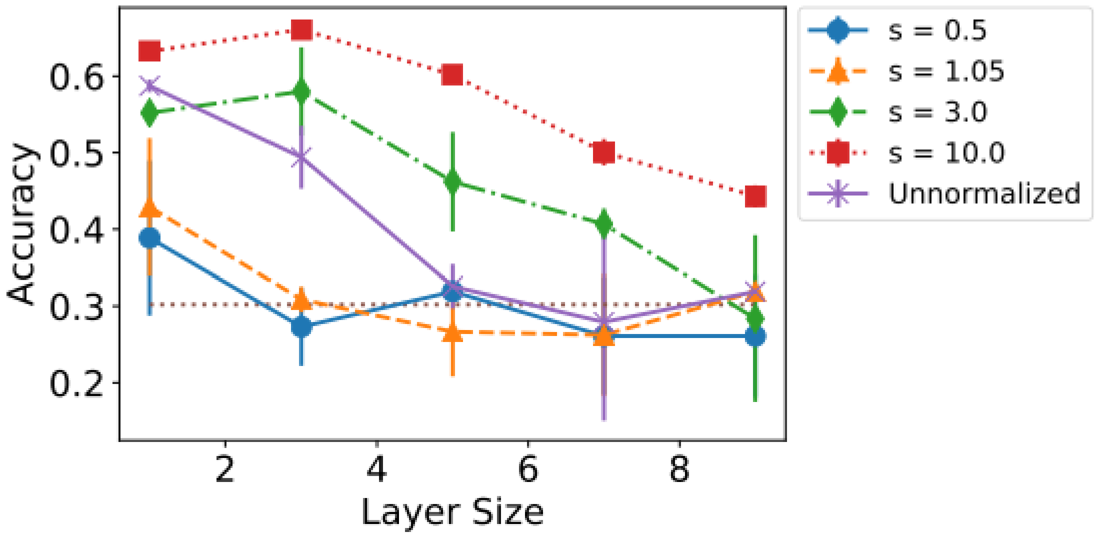
Weights

- Similarly, increasing s_{max} will increase the ϵ-smoothing layer. So how to increase s_{max}? 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_{max} as initial



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

Training Deeper GNNs

- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
  - **Over-smoothing (Graph Specific)**
  - Overfitting (Common)
  - Training dynamics (Common)
- How to alleviate over-smoothing?
- How to overcome training dynamics?
  - PairNorm
  - Shortcuts in Structures

Pair Norm: Center and Rescale

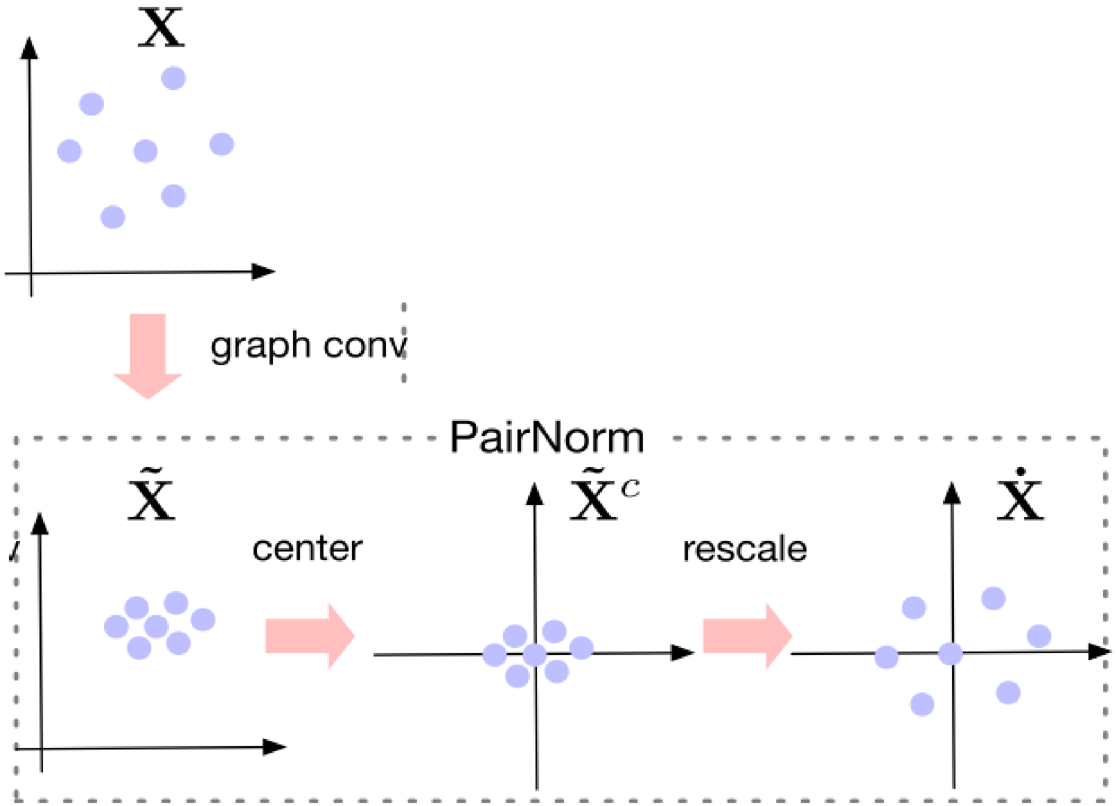
- PairNorm: Center and rescale (normalize) GCN outputs $\tilde{X} \coloneqq \mathrm{GCN}(A, X)$ to keep the **total pairwise squared distance** *unchanged*

Center

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

Rescale

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

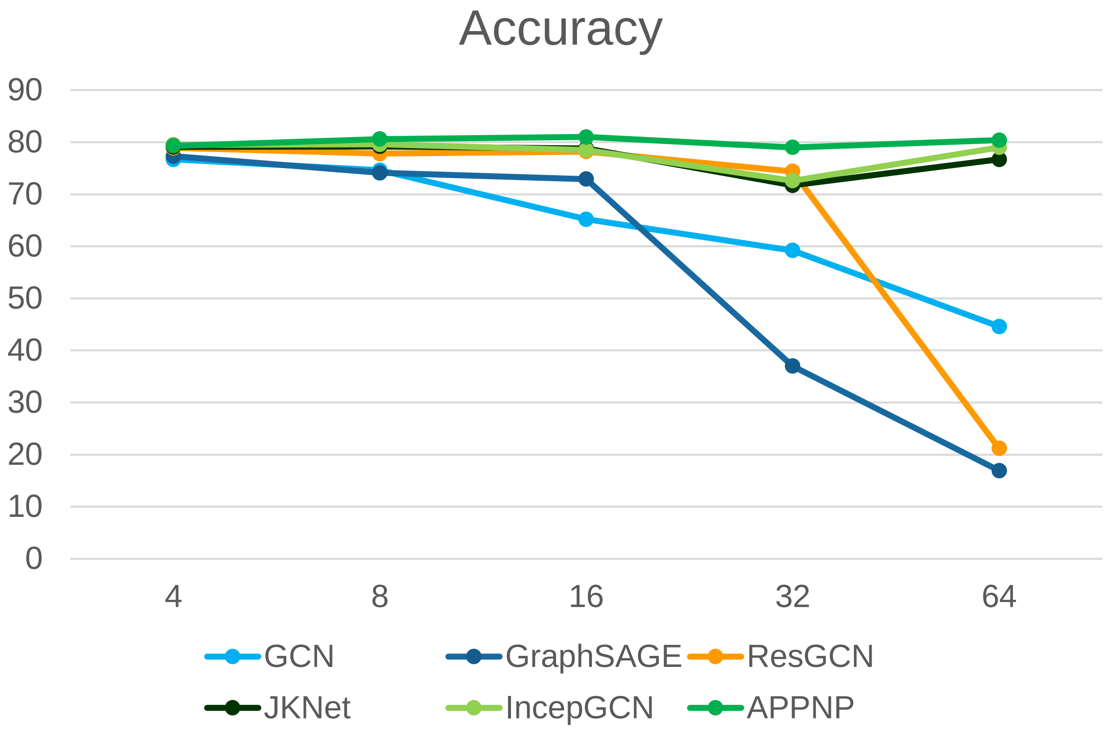


Zhao, Lingxiao, and Leman Akoglu. "PairNorm: Tackling Oversmoothing in GNNs." *ICLR. 2020.*

Training Deeper GNNs

- Why do we need deeper GNNs?
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
- How to alleviate over-smoothing?
- How to overcome training dynamics?
  - Pair norm
  - Shortcuts in Structures

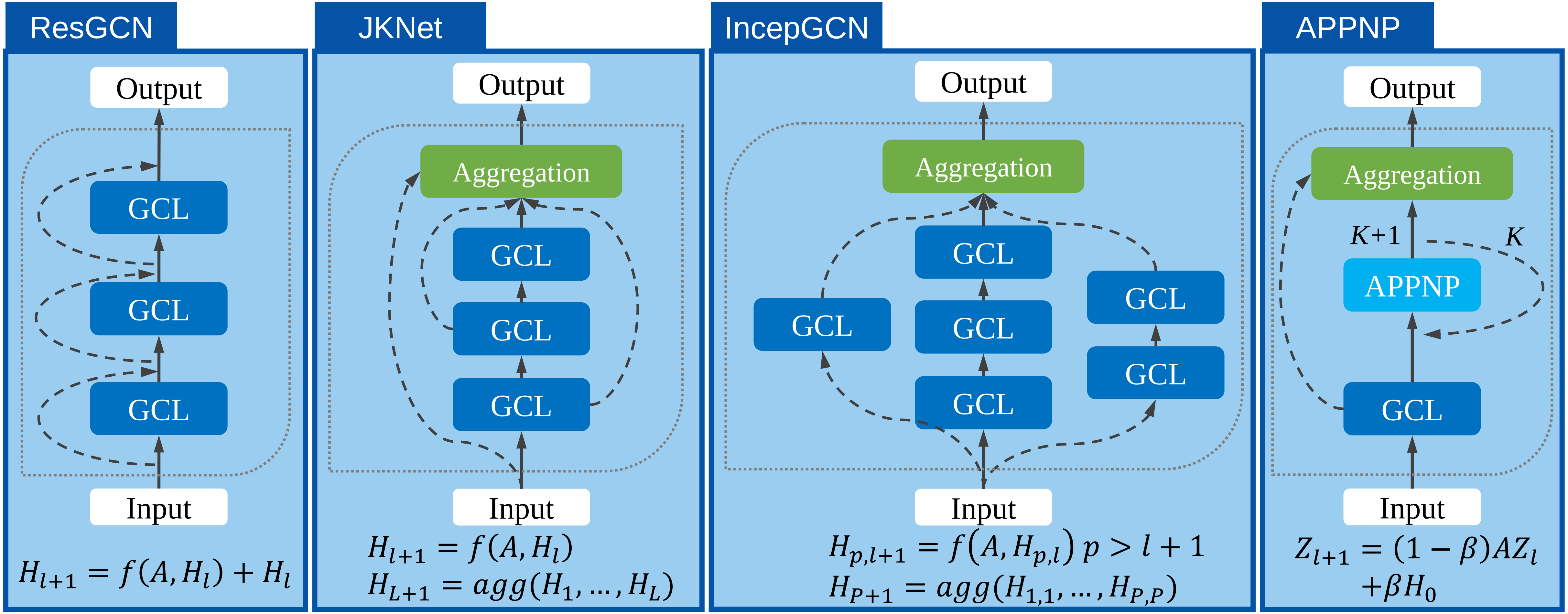
Structures with Shortcuts： Results



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**
JKNet	79.1	78.8	76.7
IncepGCN	79.5	78.5	79
APPNP	79.3	81.0	80.4

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

Shortcuts in Structures



ResGCN: $H_{l+1} = f(A, H_l) + H_l$

JKNet:

$$H_{l+1} = f(A, H_l)$$
$$H_{L+1} = agg(H_1, \dots, H_L)$$

IncepGCN:

$$H_{p,l+1} = f\left(A, H_{p,l}\right)\ p > l + 1$$
$$H_{P+1} = agg(H_{1,1}, \dots, H_{P,P})$$

APPNP:

$$Z_{l+1} = (1 - \beta)AZ_l + \beta H_0$$

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

GCN with Structures

General Case

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

Basic GCN

Non-linear GCN

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

GCN with bias

$$v = s_l \lambda_{m+1}$$
$$r = \frac{d_{\mathcal{M}}(b_{max})}{1 - v}$$

Different Structures

ResGCN

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

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

Training Deeper GNNs

- Why do we need deeper GNNs?
  - Deeper GNNs gain **expressive power** with larger receptive fields
- Can GNNs simply go deeper?
- What impedes GNNs to go deeper?
- How to alleviate over-smoothing?
- How to overcome training dynamics?

Training Deeper GNNs

- Why do we need deeper GNNs?
  - Deeper GNNs gain **expressive power** with larger receptive fields
- Can GNNs simply go deeper?
  - Not all. Some **underlying reasons** prevent GNNs from getting deeper
- What impedes GNNs to go deeper?
- How to alleviate over-smoothing?
- How to overcome training dynamics?

Training Deeper GNNs

- Why do we need deeper GNNs?
  - Deeper GNNs gain **expressive power** with larger receptive fields
- Can GNNs simply go deeper?
  - Not all. Some **underlying reasons** prevent GNNs from getting deeper
- What impedes GNNs to go deeper?
  - **Over-smoothing (Graph Specific)**
  - Overfitting (Common)
  - Training dynamics (Common)
- How to alleviate over-smoothing?
- How to overcome training dynamics?

Training Deeper GNNs

- Why do we need deeper GNNs?
  - Deeper GNNs gain **expressive power** with larger receptive fields
- Can GNNs simply go deeper?
  - Not all. Some **underlying reasons** prevent GNNs from getting deeper
- What impedes GNNs to go deeper?
  - **Over-smoothing (Graph Specific)**
  - Overfitting (Common)
  - Training dynamics (Common)
- How to alleviate over-smoothing?
  - Deal with adjacency matrix
  - Deal with weights
- How to overcome training dynamics?

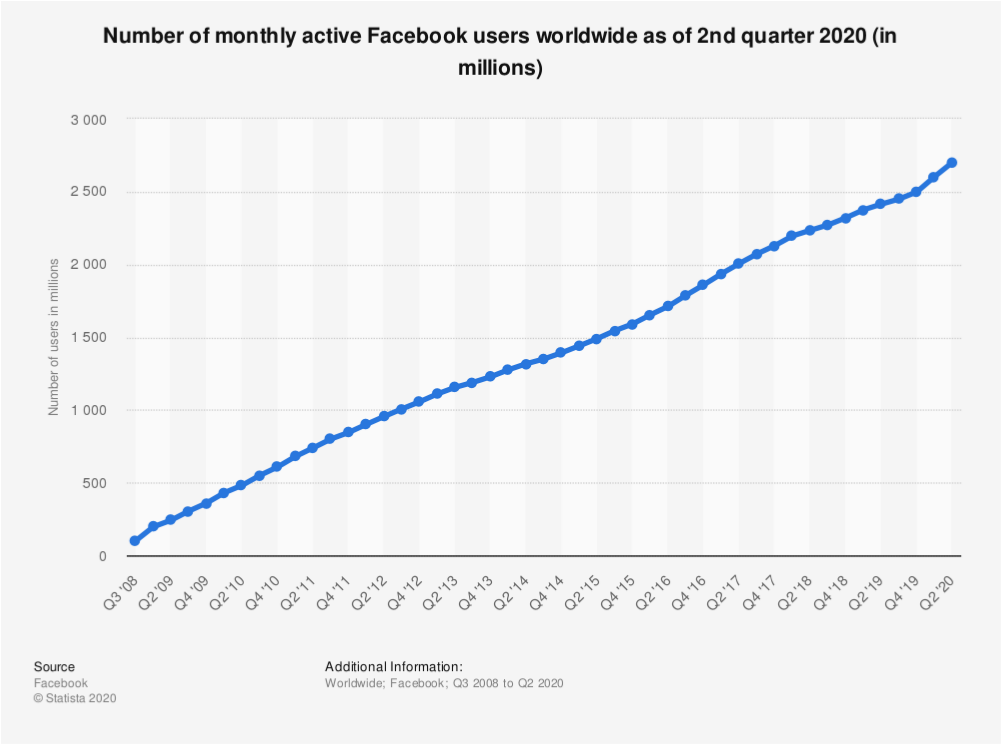
Training Deeper GNNs

- Why do we need deeper GNNs?
  - Deeper GNNs gain **expressive power** with larger receptive fields
- Can GNNs simply go deeper?
  - Not all. Some **underlying reasons** prevent GNNs from getting deeper
- What impedes GNNs to go deeper?
  - **Over-smoothing (Graph Specific)**
  - Overfitting (Common)
  - Training dynamics (Common)
- How to alleviate over-smoothing?
  - Deal with adjacency matrix
  - Deal with weights
- How to overcome training dynamics?
  - Pair norm
  - Shortcuts in Structures

Scalability of GNNs

Graph in the Real-world Can be Very Large

- Large scale:

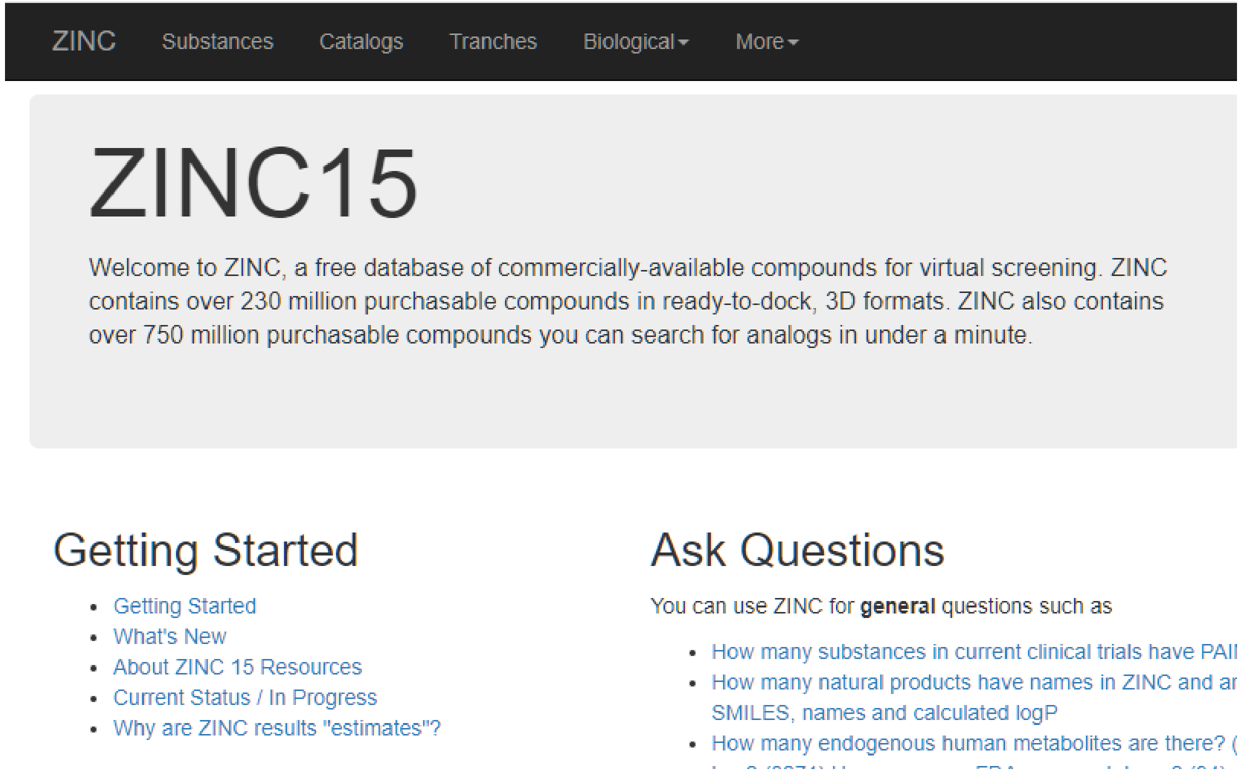


Number of monthly active Facebook users worldwide as of 2nd quarter 2020 (in millions)

Source
Facebook
© Statista 2020

Additional Information:
Worldwide; Facebook; Q3 2008 to Q2 2020

Large number:



ZINC15

Welcome to ZINC, a free database of commercially-available compounds for virtual screening. ZINC contains over 230 million purchasable compounds in ready-to-dock, 3D formats. ZINC also contains over 750 million purchasable compounds you can search for analogs in under a minute.

Getting Started

- Getting Started
- What's New
- About ZINC 15 Resources
- Current Status / In Progress
- Why are ZINC results "estimates"?

Ask Questions

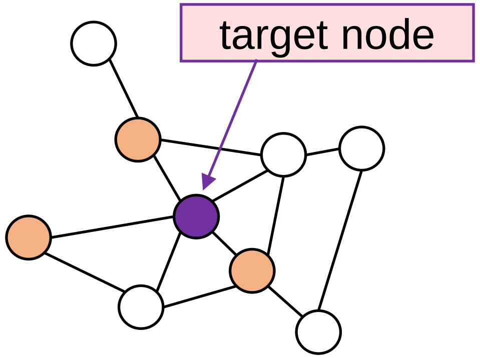
You can use ZINC for **general** questions such as

- How many substances in current clinical trials have PAI
- How many natural products have names in ZINC and ar SMILES, names and calculated logP
- How many endogenous human metabolites are there? (

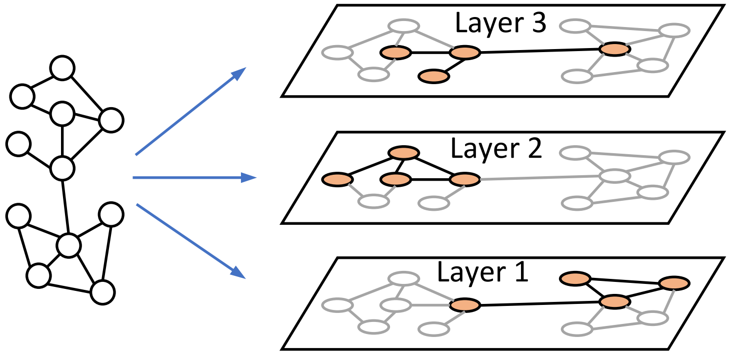
https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/
https://zinc15.docking.org/

Three Paradigms

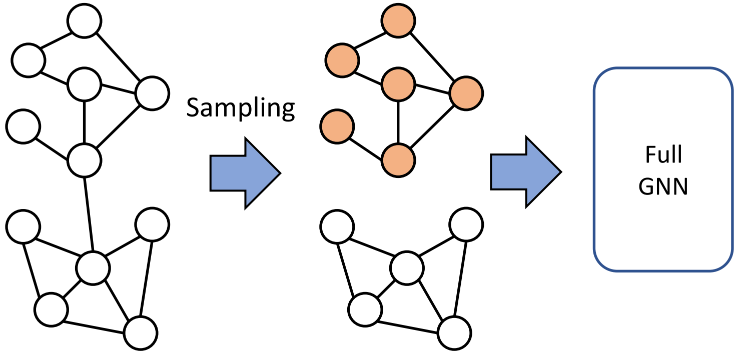
- Why the original GNN fails on large graph?
  - Large memory requirement.
  - Inefficient gradient update.
- Three paradigms toward large-scale GNN:



Node-wise Sampling



Layer-wise Sampling

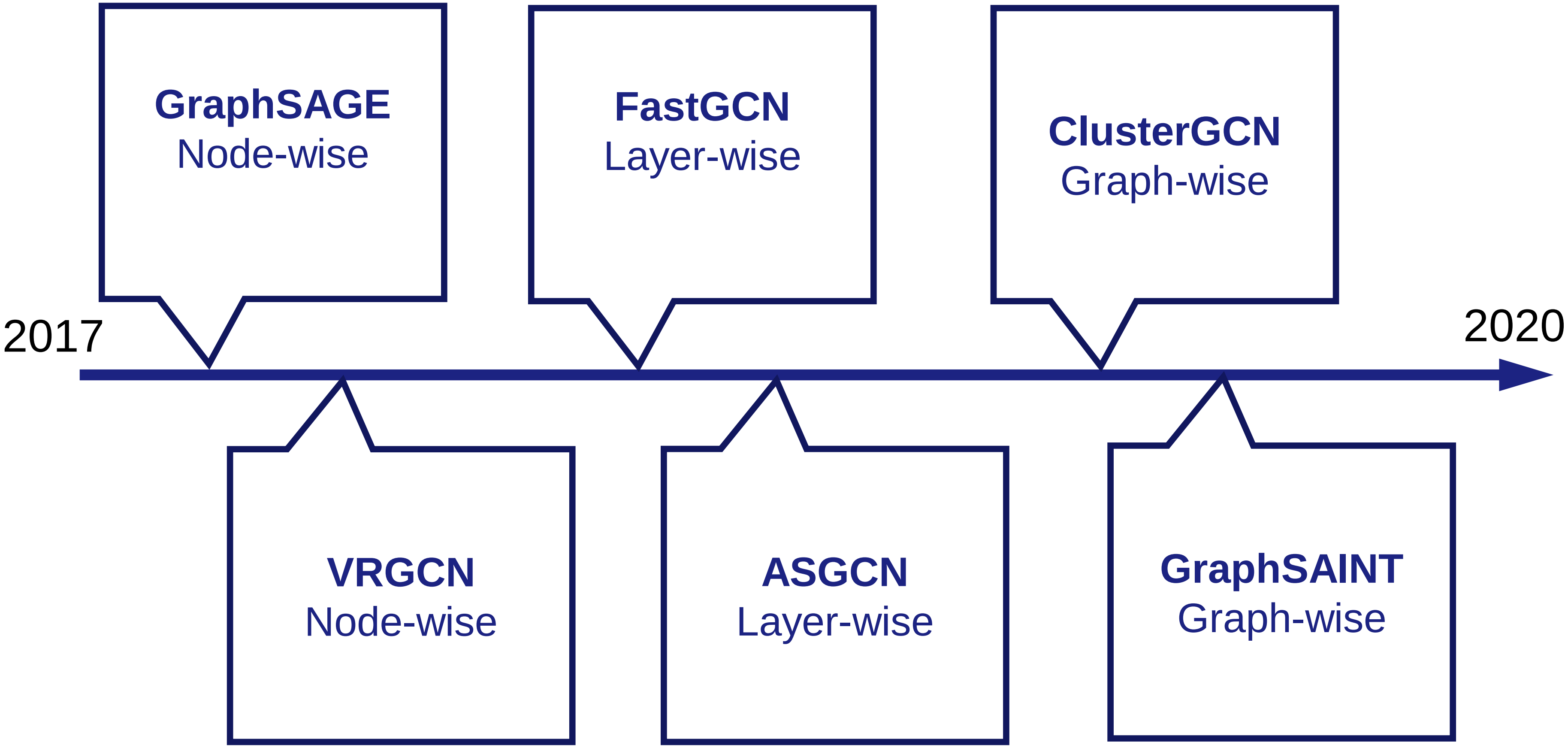


Graph-wise Sampling

Two issues towards the large-scale GNNs

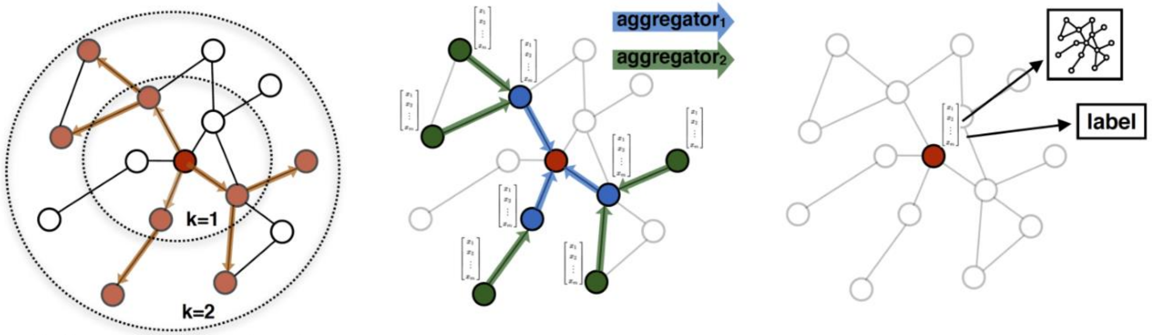
- How to design efficient sampling algorithm?
- How to guarantee the sampling quality?

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



1. Sample neighborhood

2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

- The architecture：

$$\mathbf{h}^k_{\mathcal{N}(v)} \leftarrow \boxed{\mathrm{AGGREGATE_k}}(\{\mathbf{h}_u^{k-1}, \forall \mathrm{u} \in \mathcal{N}(v)\})$$

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

Generalized Aggregator

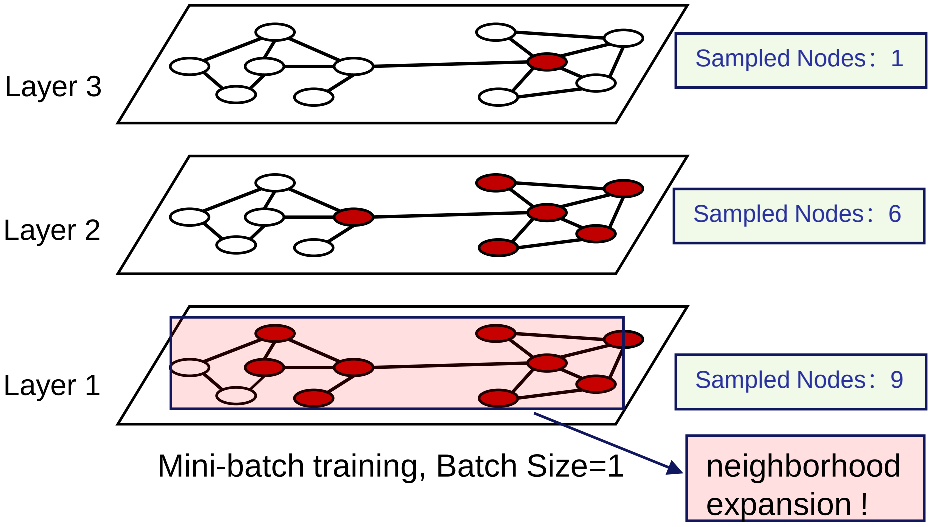
- Mean aggregator (GCN)
- Pooling aggregator
- LSTM aggregator
- .....

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



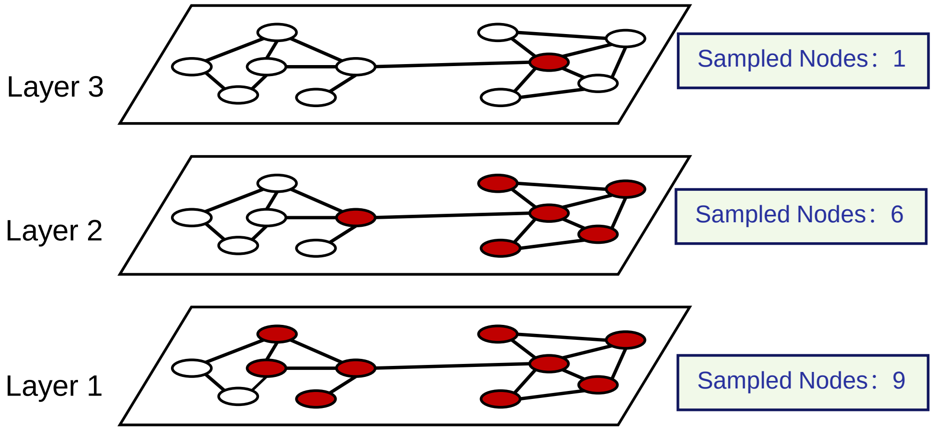
Mini-batch training, Batch Size=1

- 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

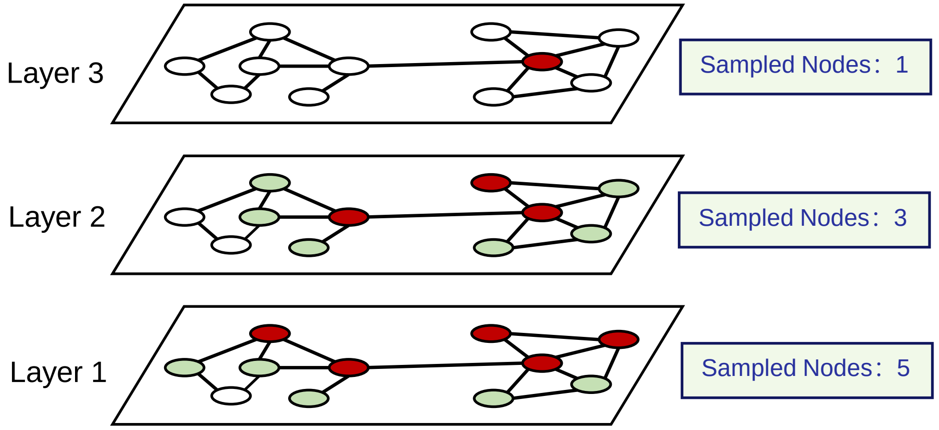
Mini-batch training



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."

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: NS Sampler

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

  - 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 $\widehat{\boldsymbol{n}}^{(l)}(\boldsymbol{u})$.**
- VR-GCN:

$$\boldsymbol{H}^{(l+1)} = \sigma\Big(\boxed{\widehat{\boldsymbol{P}}^{(l)}}\big(\boldsymbol{H}^{(l+1)} - \boxed{\bar{\boldsymbol{H}}^{(l)}}\big) + \boldsymbol{P}\bar{\boldsymbol{H}}^{(l)}\Big)\boldsymbol{W}^{(l)}$$

The sampled normalized adjacency matrix at layer l.

Historical hidden embedding

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

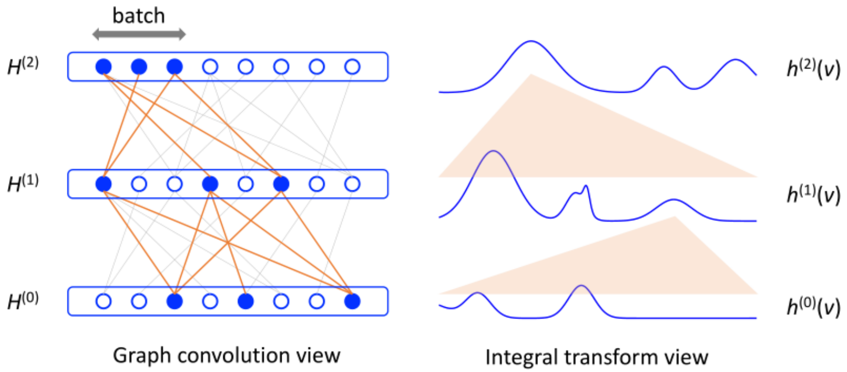
Chen, Jianfei, Jun Zhu, and Le Song. "Stochastic Training of Graph Convolutional Networks with Variance Reduction."

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

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

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

FastGCN: sampling fixed number of nodes at each layer.

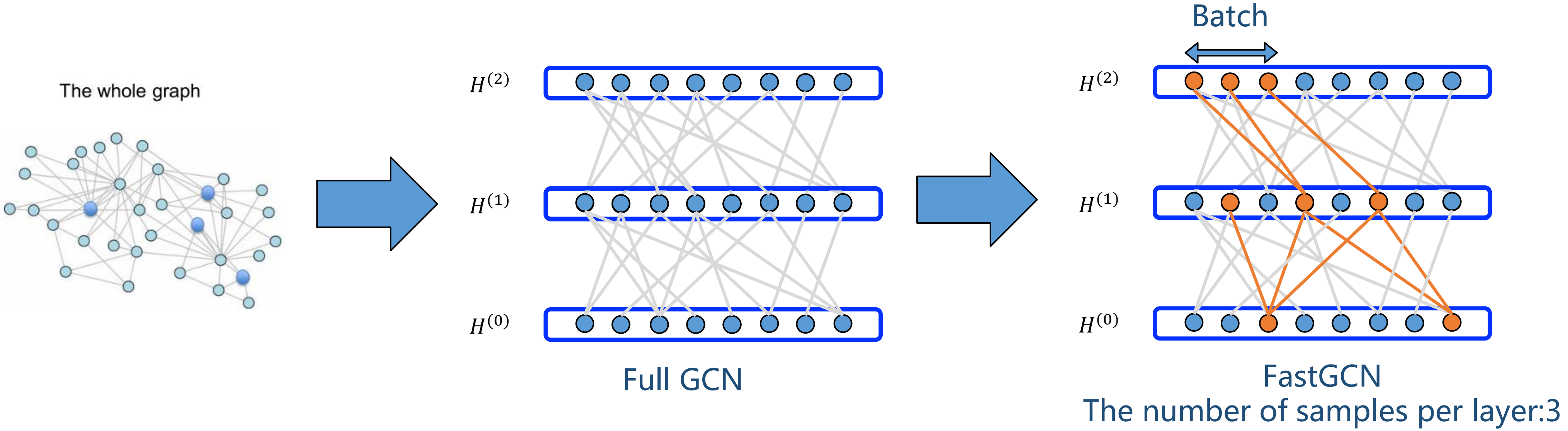
Chen, Jie, Tengfei Ma, and Cao Xiao. "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling."

Layer-wise Sampling: FastGCN

- Towards the variance reduction: importance sampling.

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

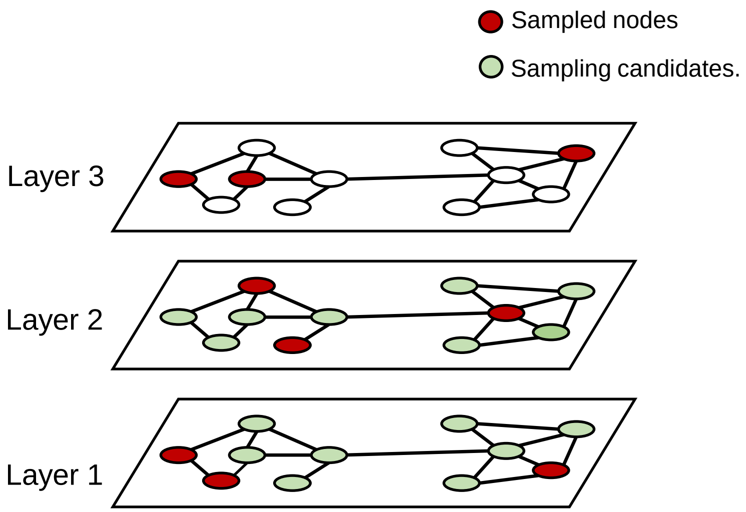
This sampling probability keeps the same for each layer.



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

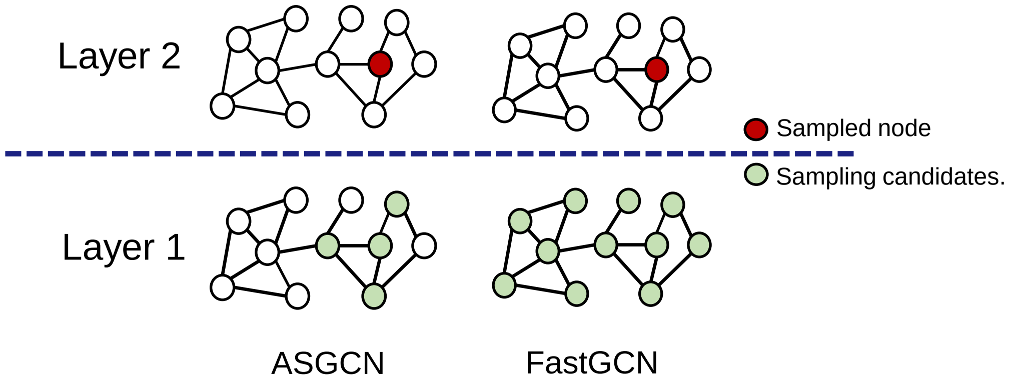
$x\,(u_j)$: the node feature of node u_j.
$p(u_j|v_i)$: the probability of sampling node u_j given node v_i.
$\hat{a}(v_i, u_j)$: The entry of node v_i and u_j in re-normalization of the adjacency matrix $\hat{A}$.
$\hat{\mu}_q(v_i)$: the output hidden embeddings of node v_i.

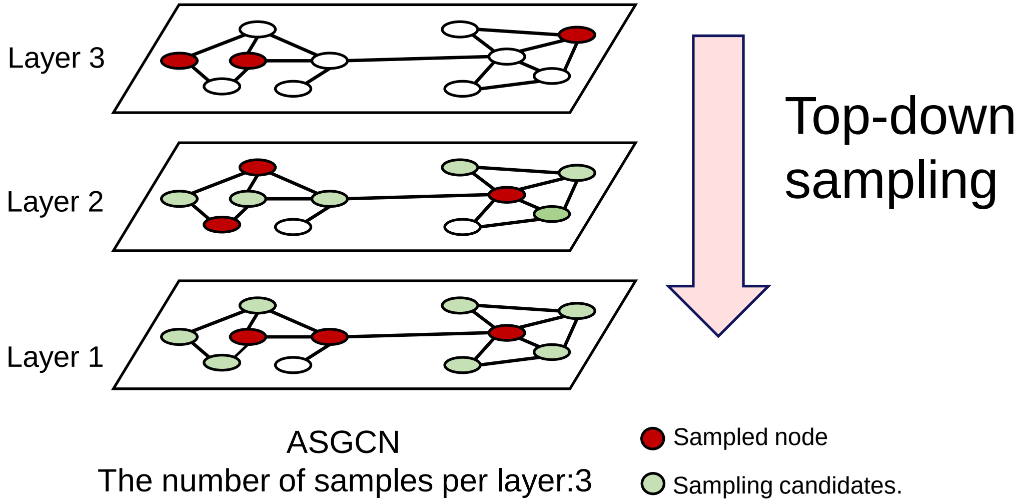
- Adaptive layer-wise sampling:
  - The sampling probability of lower layers depends on the upper ones.

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

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 $\hat{A}$.





ASGCN
The number of samples per layer:3

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

Layer-wise Sampling: ASGCN

- Parameterized for **explicit variance reduction**.

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

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

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

Can be estimated by the sampled instances.

$x\,(u_j)$: 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.

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

Layer-wise Sampling: ASGCN

- Self-dependent shared attention:

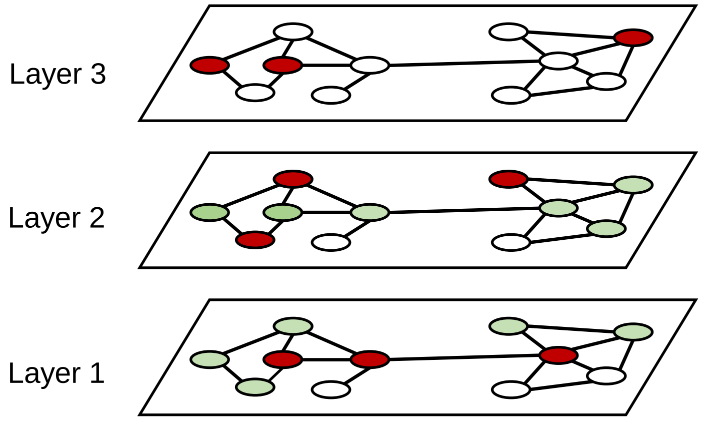
$$a(x(v_i), x(u_j)) \quad = \quad \frac{1}{n}\mathrm{ReLu}(W_1 g(x(v_i)) + W_2 g(x(u_j)))$$

- Preserving second-order proximities by skip connections:

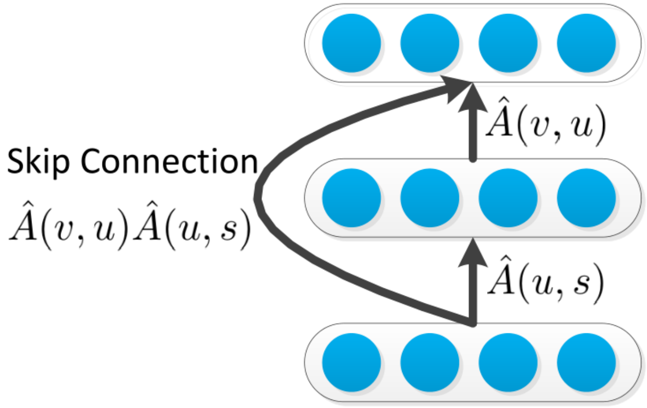
$$h^{(l+1)}_{skip}(v_i) \quad = \quad \sum\nolimits_{j=1}^{n} \hat{a}_{skip}(v_i, s_j) h^{(l-1)}(s_j) W^{(l-1)}_{skip}, \quad i = 1, \cdots, n,$$

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

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



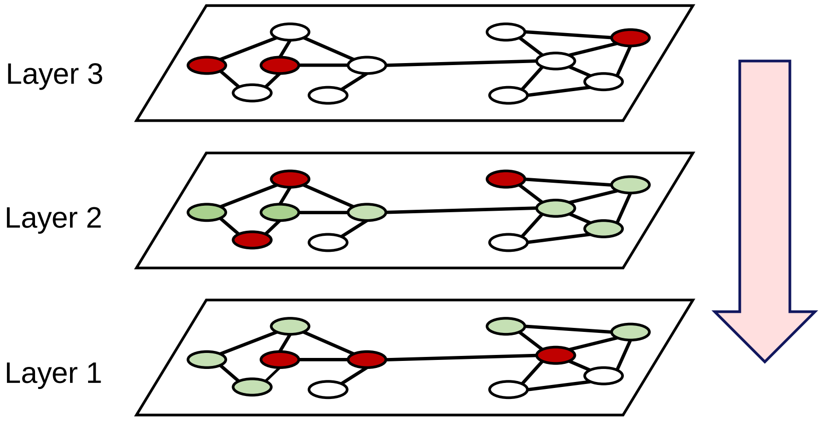
ASGCN



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."

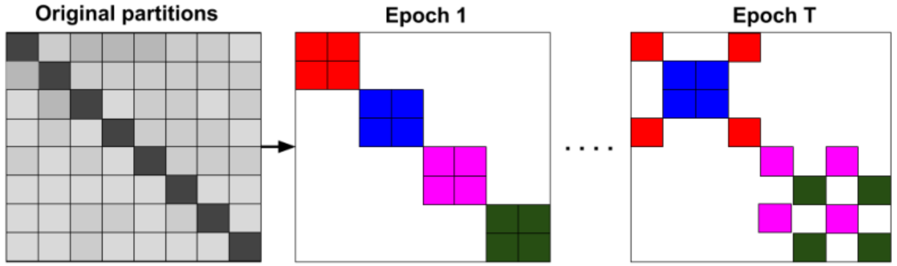
Graph-wise Sampling: ClusterGCN

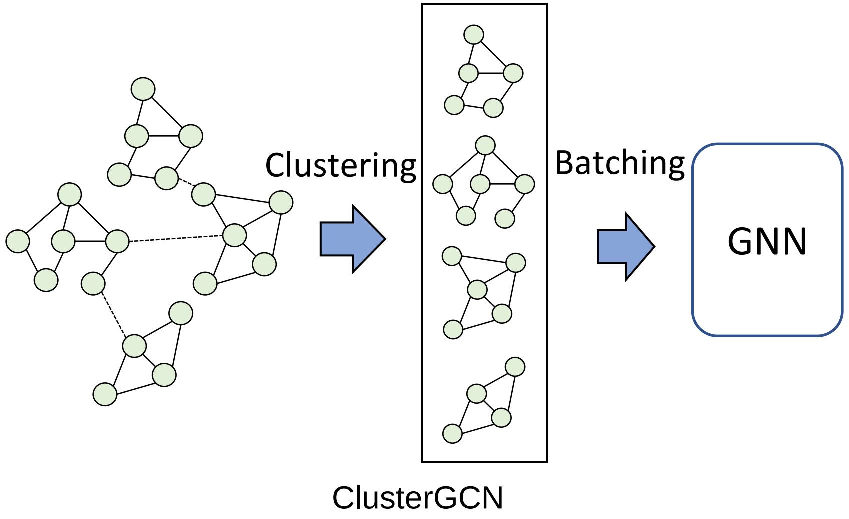
- Extract small clusters based efficient clustering algorithms.

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

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

- Random batching at the subgraph level.

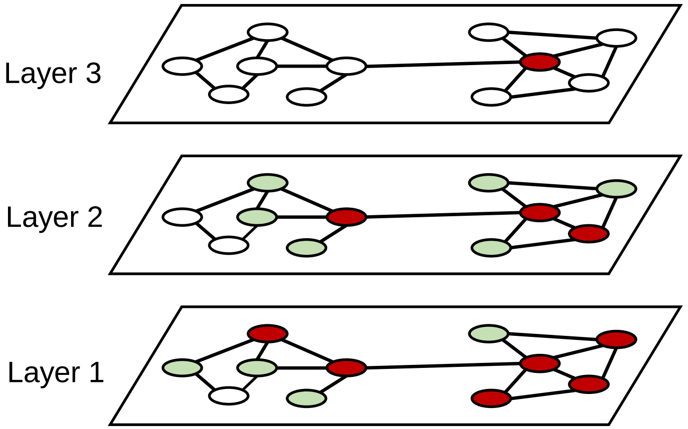




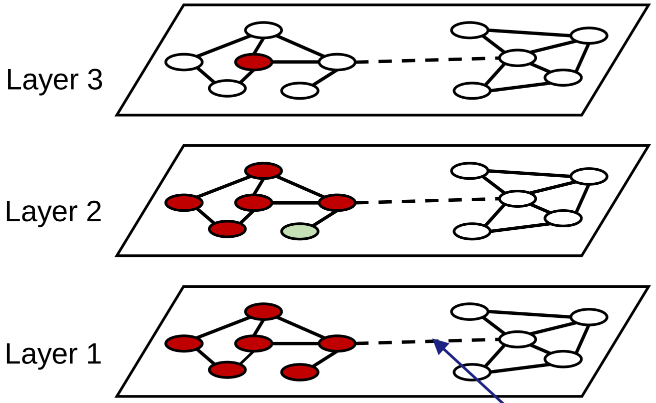
Chiang, Wei-Lin, et al. "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks."

Graph-wise Sampling: ClusterGCN

- Neighbor expansion control.



Fix-size neighbor sampling S=2



ClusterGCN

Only sample the nodes in the clusters

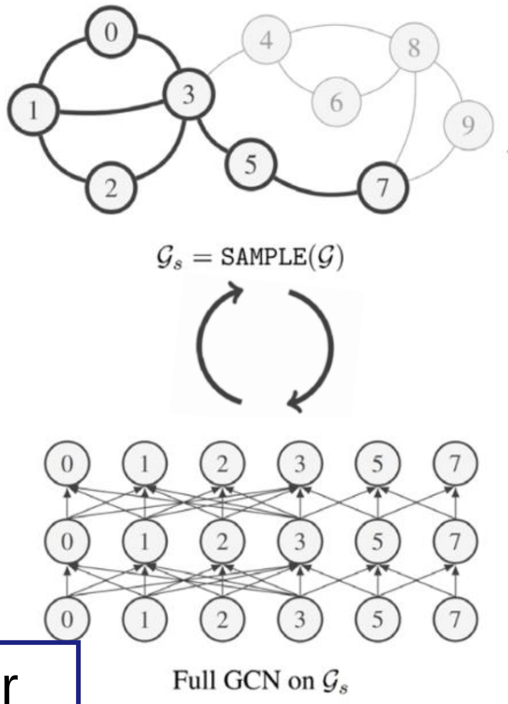
Chiang, Wei-Lin, et al. "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks."

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.

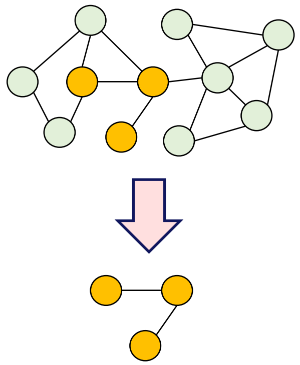
Chiang, Wei-Lin, et al. "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks."

Graph-wise Sampling: GraphSAINT


0
1
2
3
4
5
6
7
8
9
$\mathcal{G}_s = \texttt{SAMPLE}(\mathcal{G})$
Full GCN on $\mathcal{G}_s$

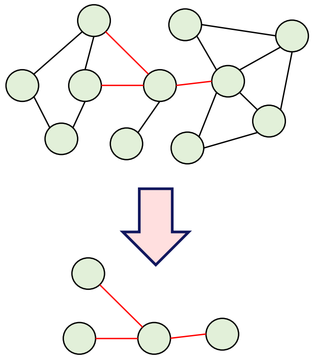
- Directly sample a subgraph for mini-batch training according to subgraph sampler.
- Sampler construction

Node sampler



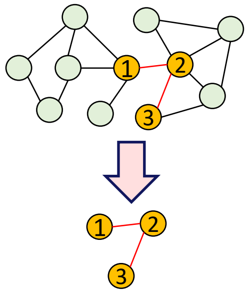
Uniformly sample nodes.

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.

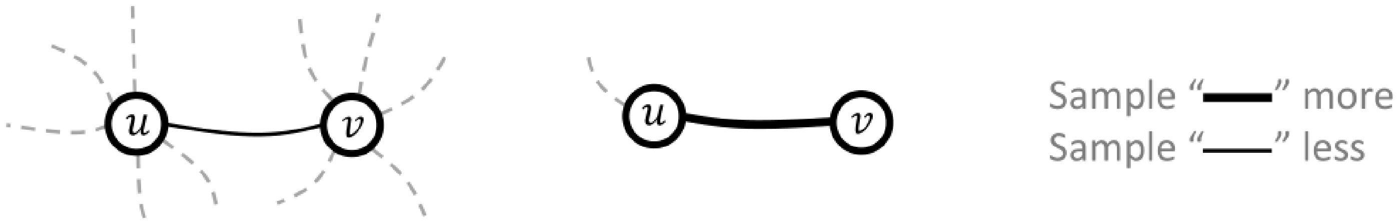
Zeng, Hanqing, et al. "GraphSAINT: Graph Sampling Based Inductive Learning Method."

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.

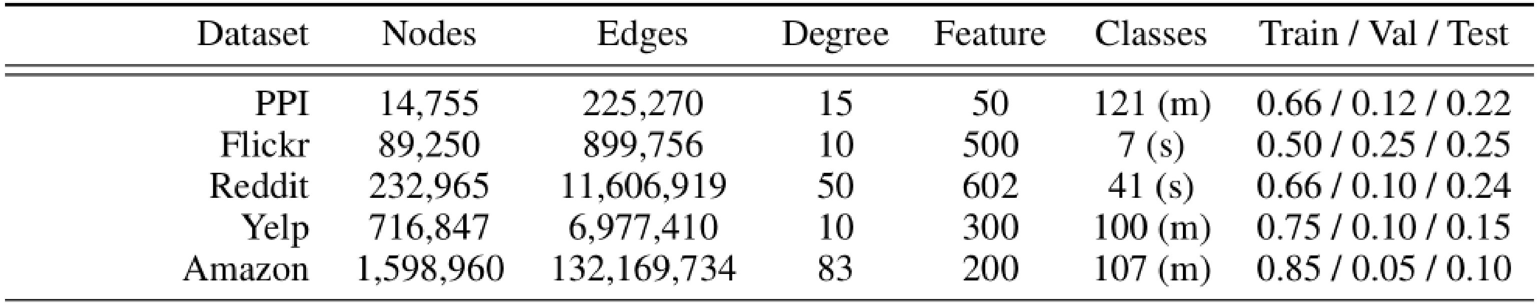
- How to reduce the sampling variance?
  - The optimal edge probability for variance minimization:

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



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

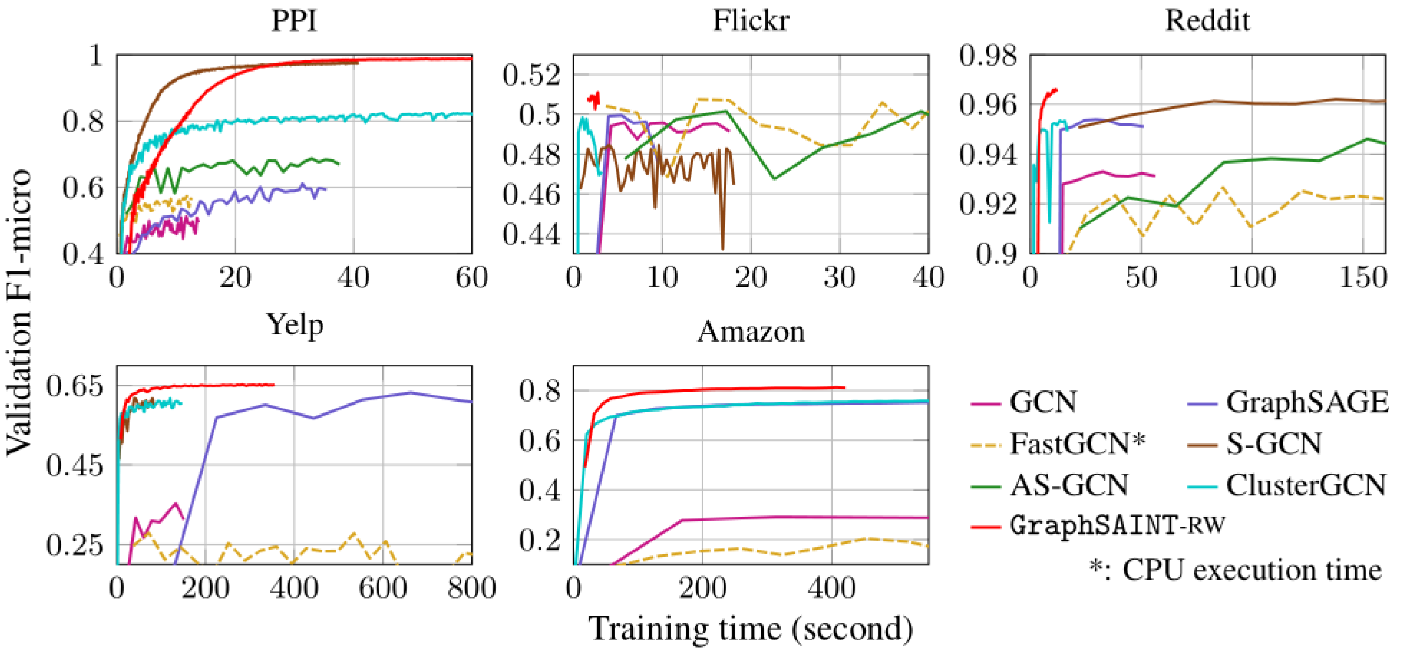
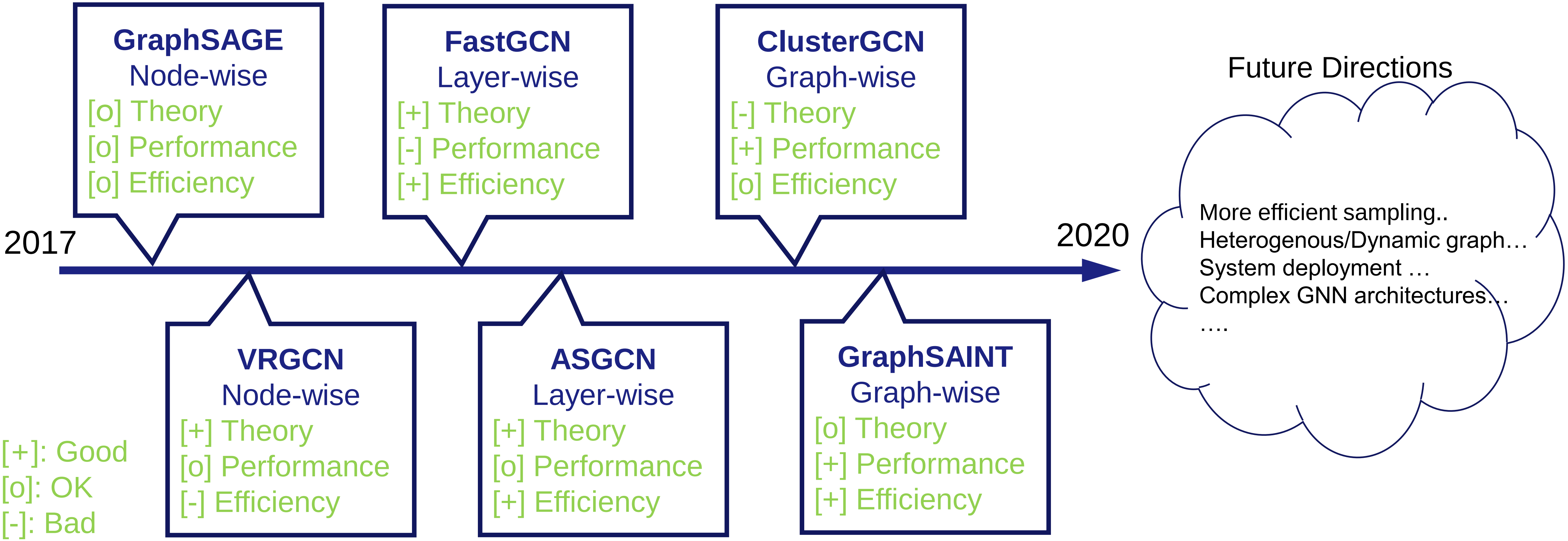


Figure 2: Convergence curves of 2-layer models on GraphSAINT and baselines

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

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

Summary



2017

GraphSAGE
Node-wise
[o] Theory
[o] Performance
[o] Efficiency

VRGCN
Node-wise
[+] Theory
[o] Performance
[-] Efficiency

FastGCN
Layer-wise
[+] Theory
[-] Performance
[+] Efficiency

ASGCN
Layer-wise
[+] Theory
[o] Performance
[+] Efficiency

ClusterGCN
Graph-wise
[-] Theory
[+] Performance
[o] Efficiency

GraphSAINT
Graph-wise
[o] Theory
[+] Performance
[+] Efficiency

2020

Future Directions

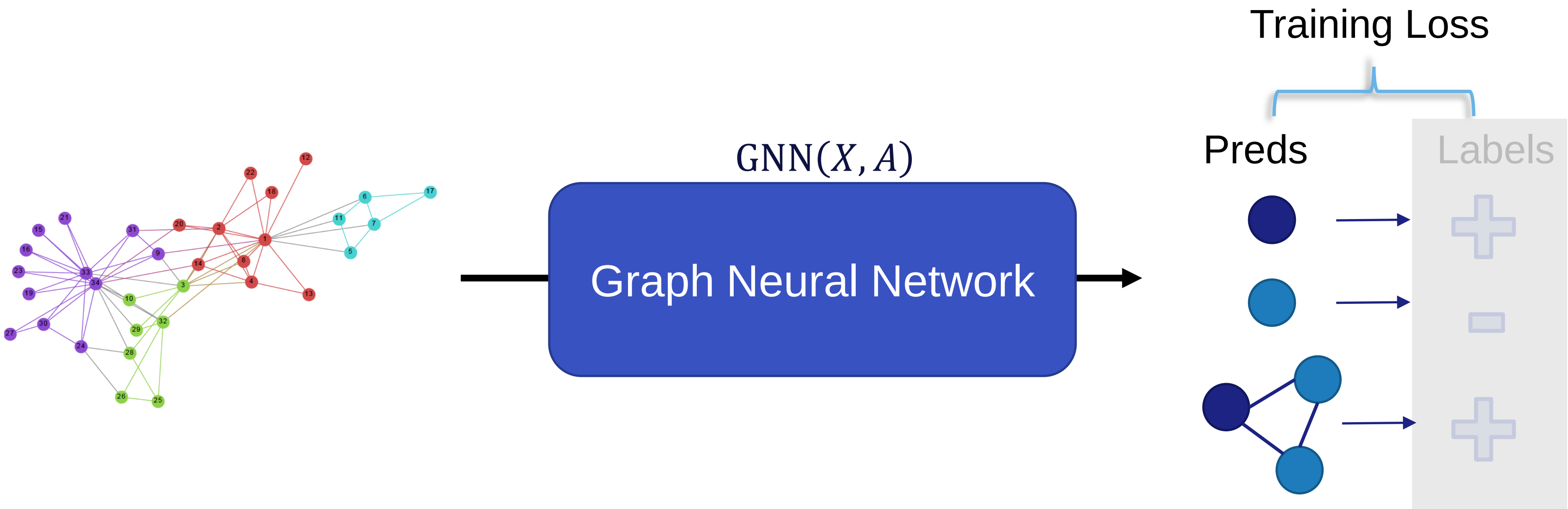
More efficient sampling..
Heterogenous/Dynamic graph…
System deployment …
Complex GNN architectures…
….

[+]: Good
[o]: OK
[-]: Bad

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

Self/Un-Supervised Learning of GNNs

What we discussed before are supervised



Training Loss

$\mathrm{GNN}(X, A)$

Graph Neural Network

Preds

Labels

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

Existing Self-Supervised GNNs

	Node-Classification	Graph-Classification
Predictive Methods	VGAE [1], EP-B [2], GraphSAGE [3]	N-gram Graph [4], PreGNN [5] , GCC [6], GROVER [7]
Information-based Methods	DGI [8], GMI [9]	InfoGraph [10]

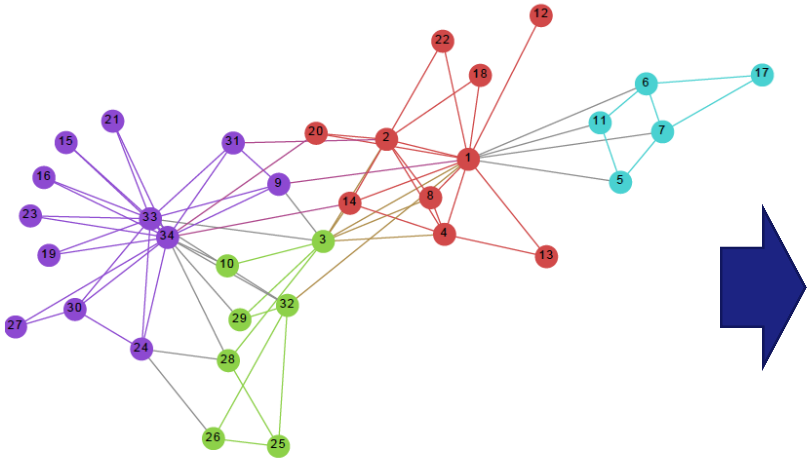
[1] Kipf & Welling 2016; [2] Durán & Niepert 2017; [3] Hamilton et al. 2017;
[4] Liu et al. 2019; [5] Hu et al. 2020; [6] Qiu et al. 2020; [7] Rong et al. 2020;
[8] Veliˇckovi´c et al. 2019; [9] Peng et al. 2020; [10] Sun et al. 2020

“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!

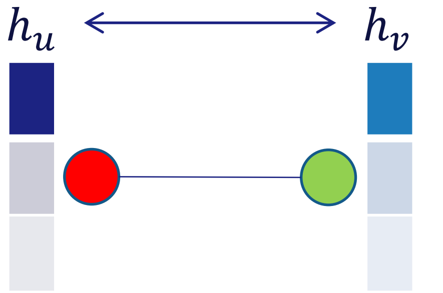
Node classification

- Two typical ways to formulate training loss



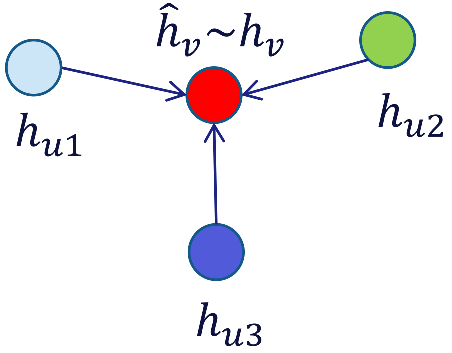
I. Enforcing Adjacent Similarity

VGAE, GraphSAGE



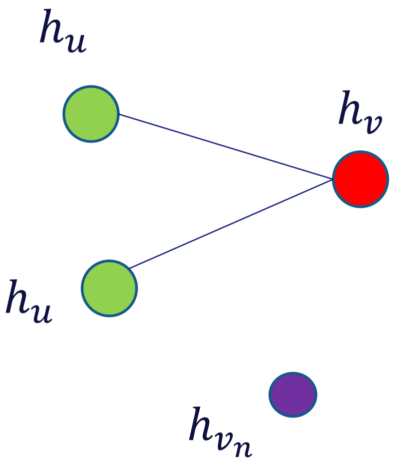
II. Reconstruction from Neighbors

EP-B



I Enforcing Adjacent Similarity

- GraphSAGE (Hamilton et al. 2017)



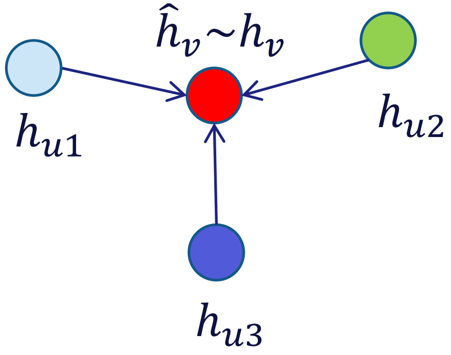
Enforcing nearby nodes to have similar representations, while enforcing disparate nodes to be distinct:

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

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):

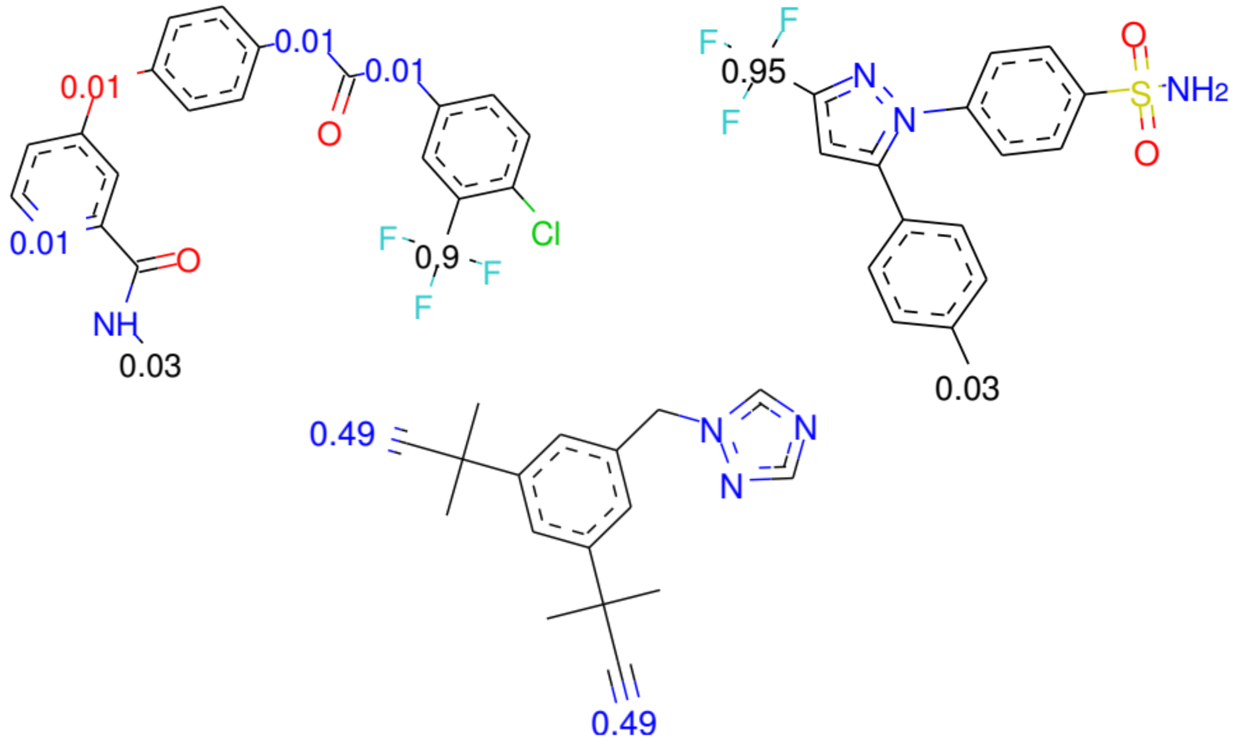
$$\boxed{\min \sum_{u \in V \setminus \{v\}} [\gamma + \underbrace{d(\tilde{\boldsymbol{h}}_v, h_v)}_{\text{Positive Samples}} - \underbrace{d(\tilde{\boldsymbol{h}}_v, h_u)}_{\text{Negative Samples}}]_+}$$

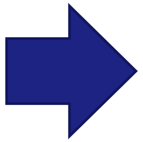
Hinge loss

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.

How about graph classification/regression?





Toxicity?

Solubility?

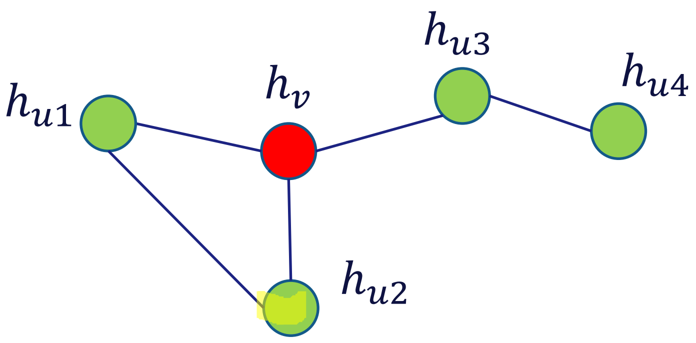
…

Yu Rong, Wenbing Huang, Tingyang Xu, Hong Cheng, Junzhou Huang 2020

N-Gram Graph

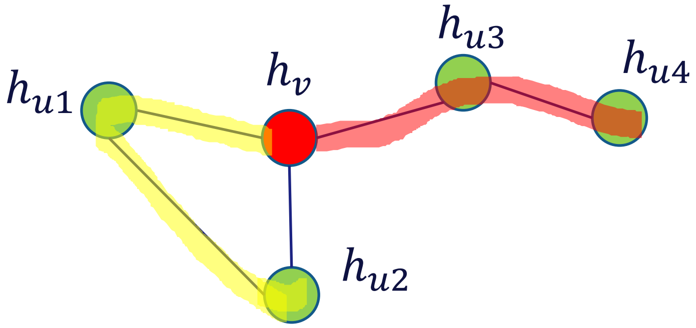
- (Liu et al. 2019)

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: $F = [f_{(1)}, \dots, f_{(T)}]$

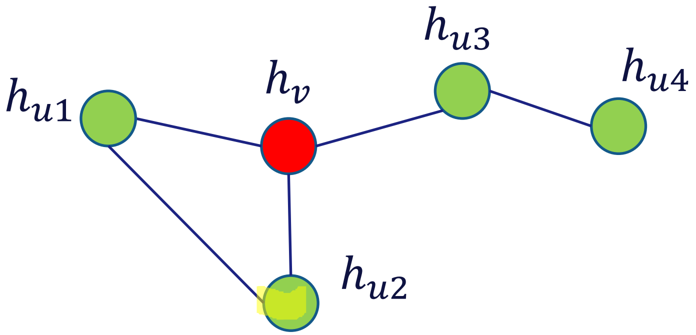
Equivalent to a GNN that needs no training

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

PreGNN: node- and graph-level pretraining

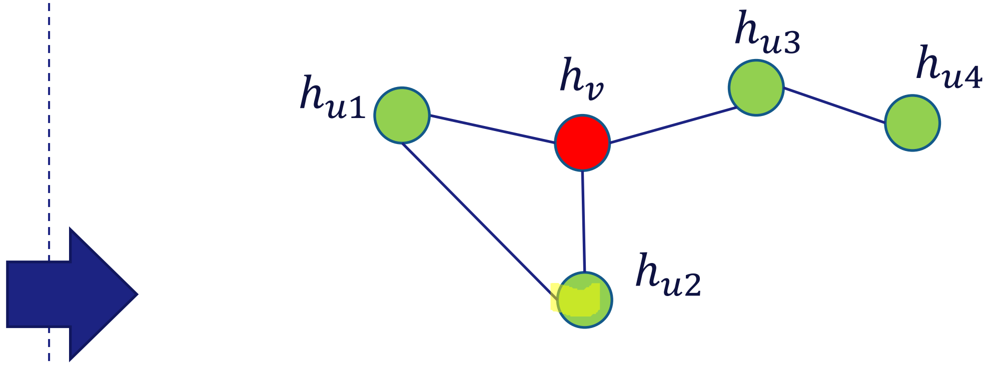
- (Hu et al. 2020)

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:

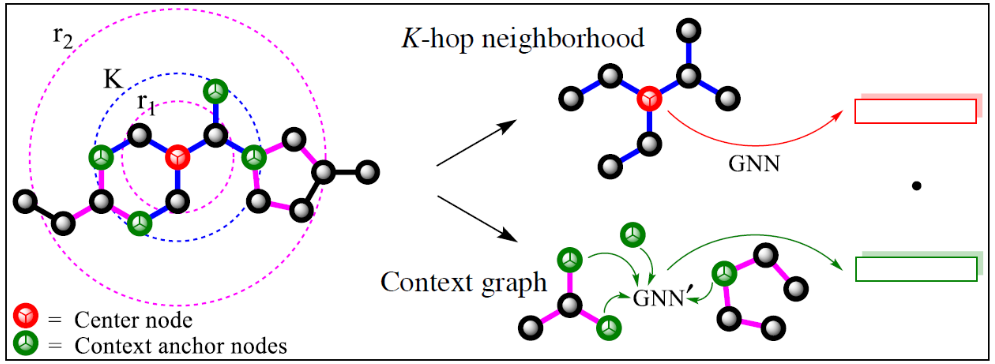
$$\boxed{\min -\log\left(\sigma\left(({h_v}^K)^T {C_v}^G\right)\right) - I(v \neq v')log(1-\sigma(({h_v}^K)^T {C_{v'}}^{G'}))}$$

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

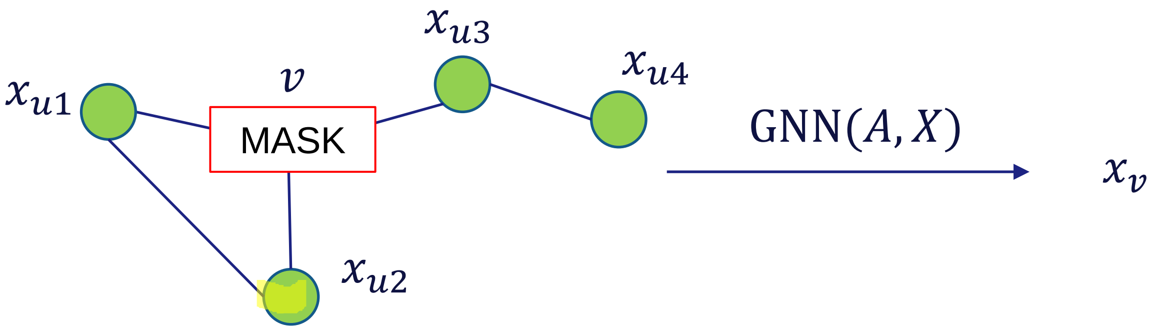
Hu et al. Strategies for Pre-training Graph Neural Networks.

PreGNN

- (Hu et al. 2020)

Stage I: Node Representation

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



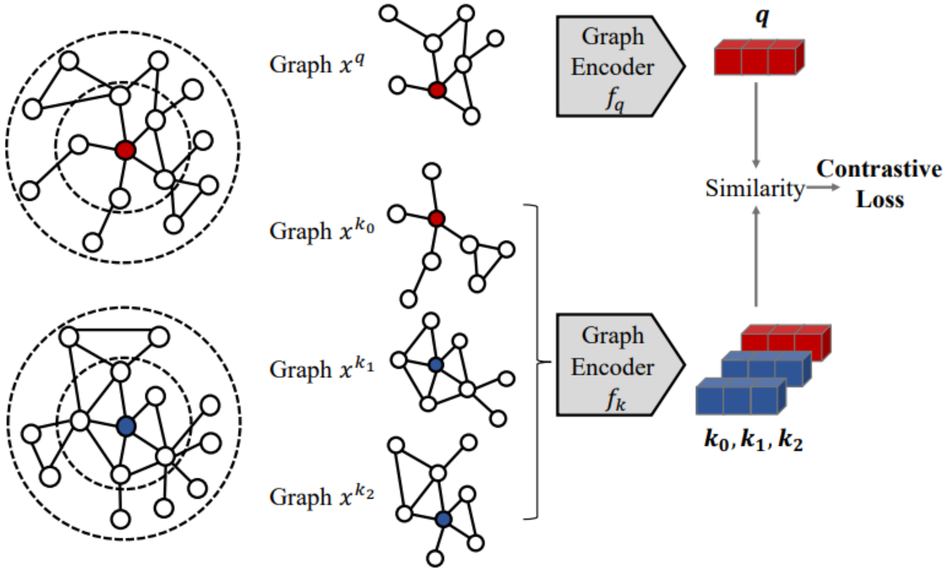
Context Prediction
Attribute Masking

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:



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

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

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.

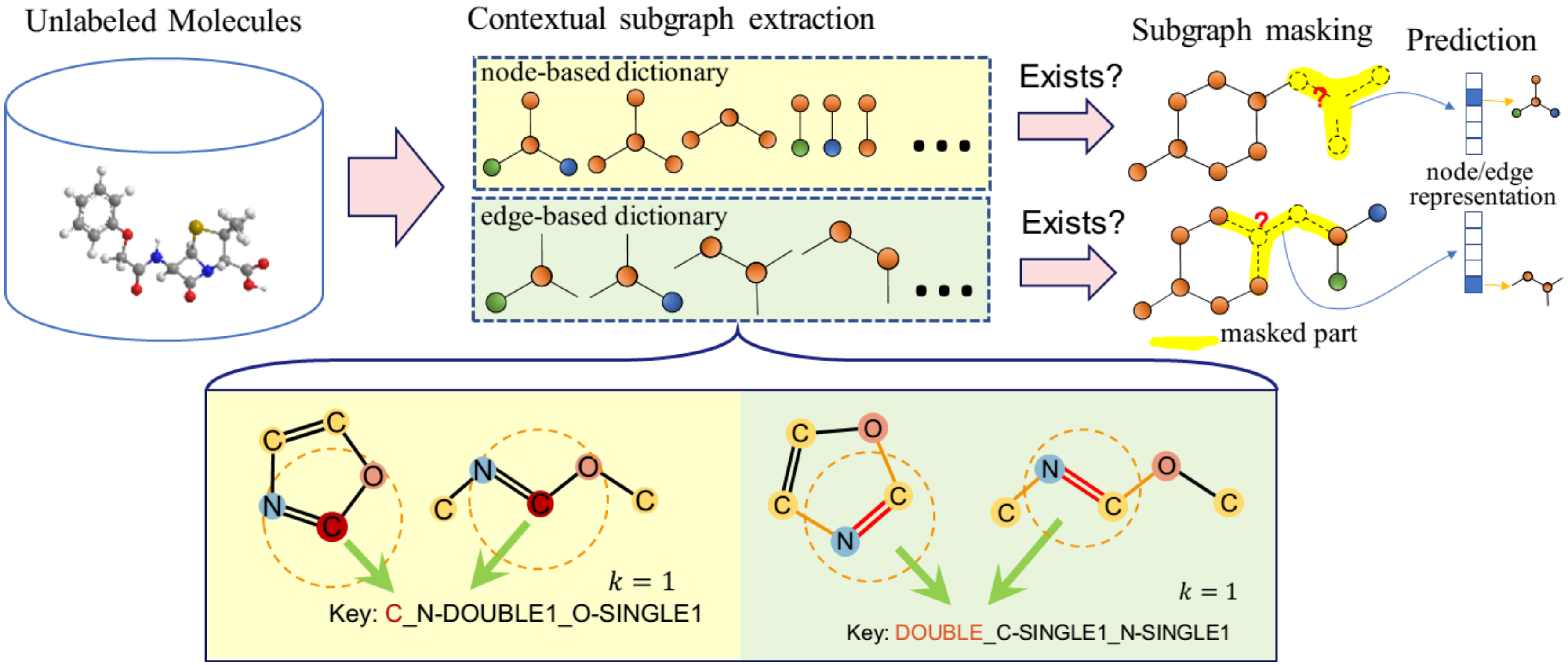
GROVER (Rong et al. 2020)

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

GROVER

- (Rong et al. 2020)

Stage I: Node/Edge-level pretraining



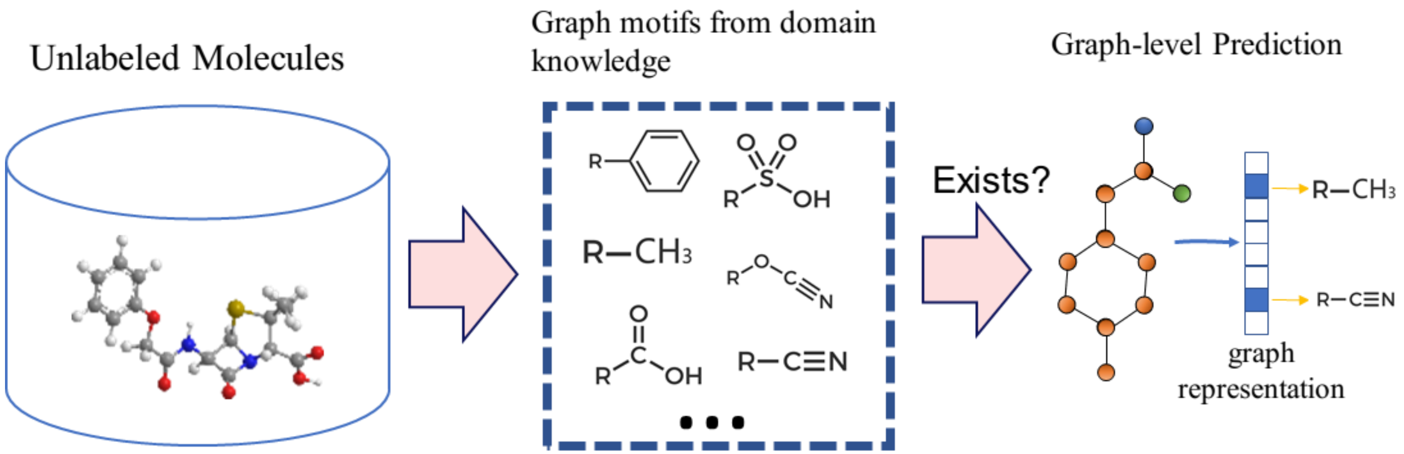
Predicting node/edge contexts instead of node labels can better capture local structures (multi-label)

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

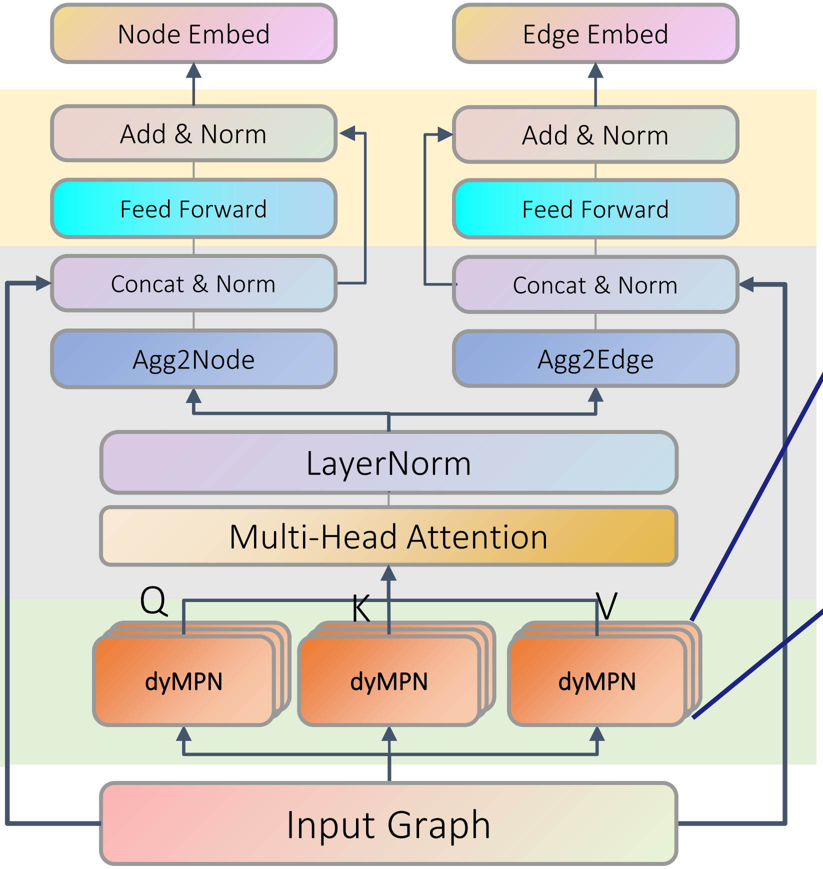
- One more thing: GTransformer

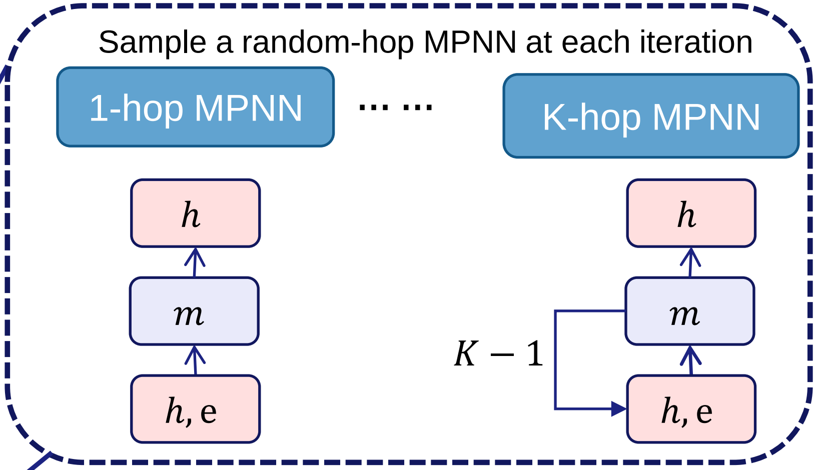
We build a more expressive and transformer-alike model: GTransformer

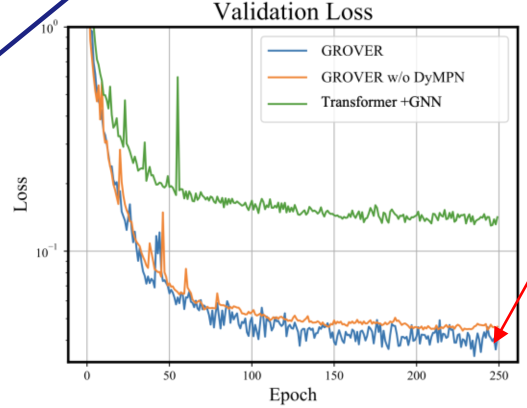
- Output for both node embedding and edge embedding.

- 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.







Better generalization ability

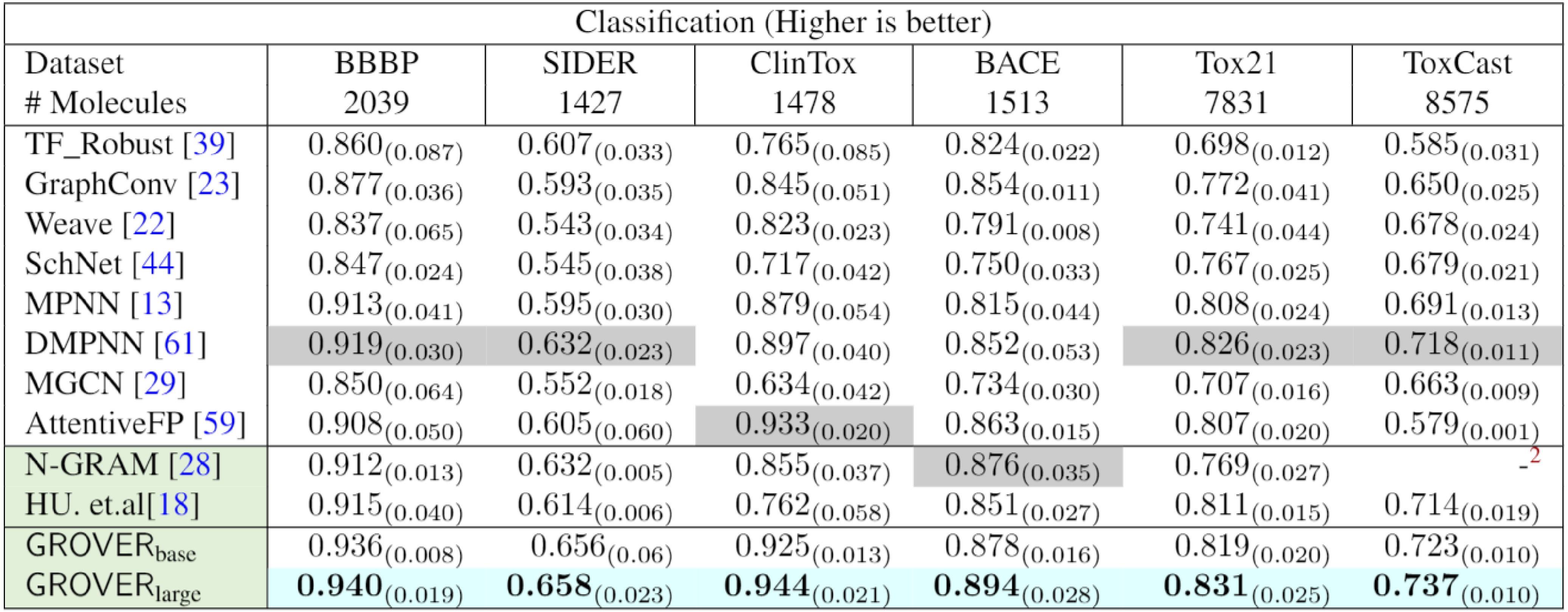
Rong et al. GROVER: Self-supervised Message Passing Transformer on Large-scale Molecular Data.

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	BBBP	SIDER	ClinTox	BACE	Tox21	ToxCast
# Molecules	2039	1427	1478	1513	7831	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$_{(0.019)}$
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$_{(0.010)}$
GROVER$_{\text{large}}$	**0.940**$_{(0.019)}$	**0.658**$_{(0.023)}$	**0.944**$_{(0.021)}$	**0.894**$_{(0.028)}$	**0.831**$_{(0.025)}$	**0.737**$_{(0.010)}$

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

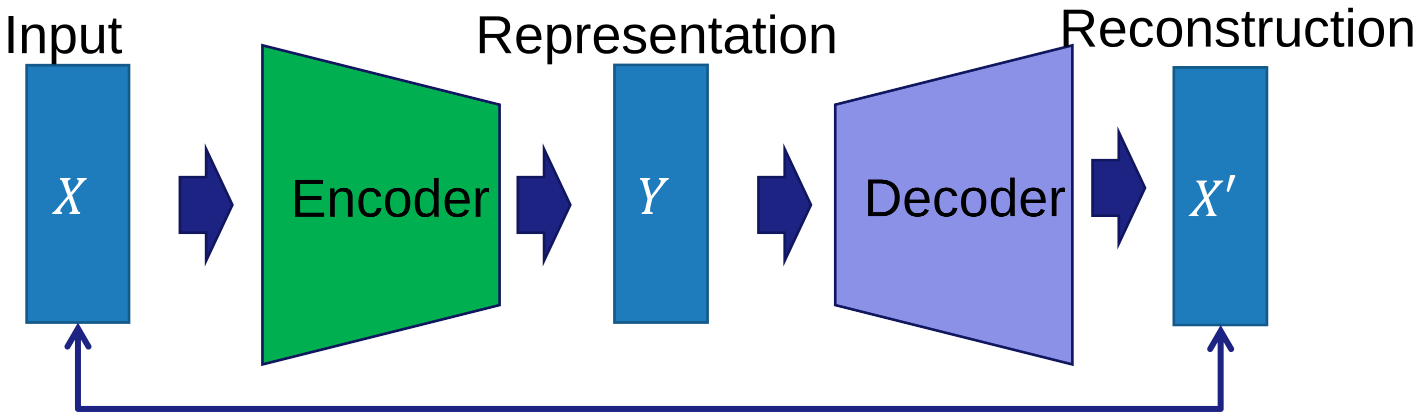
Existing Self-supervised models

	Node-Classification	Graph-Classification
Predictive Methods	VGAE [1], EP-B [2], GraphSAGE [3]	N-gram Graph [4], PreGNN [5], GCC [6], GROVER [7]
Information-based Methods	DGI [8], GMI [9]	InfoGraph [10]

[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

What makes a good representation?

Auto-Encoder (AE)



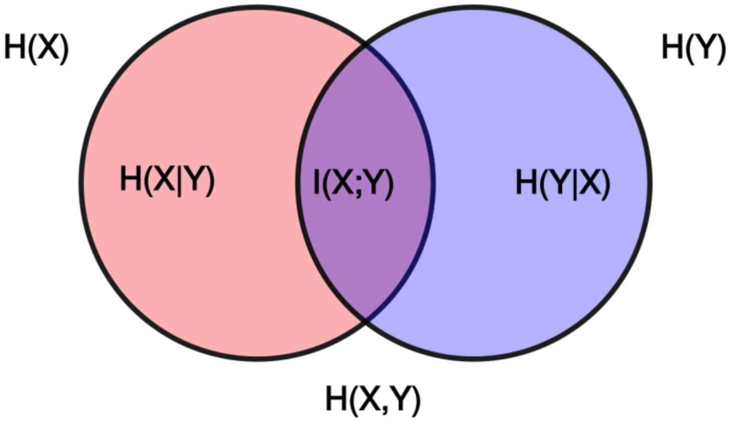
“One natural criterion that we may expect any good representation to meet, at least to some degree, is to **retain a significant amount of information about the input**.” by Vincent et al. 2010

Hinton & Salakhutdinov 2006; Vincent et al. 2010

What makes a good representation?

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

$$I(X;Y) = D_{KL}(p(X)p(Y)||p(X,Y)) \\ = \underbrace{H(X)}_{\text{Entropy}} - \underbrace{H(X|Y)}_{\text{Conditional Entropy}}$$



- $0 \leq I(X;Y) \leq H(X)$ 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. $H(X|Y) = 0$

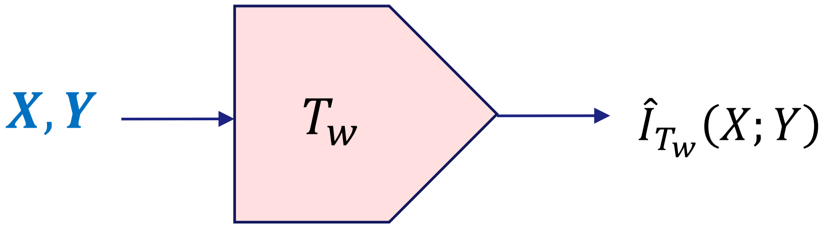
AE is a lower bound of MI

(Hjelm et al. 2019)

$$\boxed{\underbrace{\boxed{I(X;Y)}}_{\text{Mutual Information}} = H(X) - H(X|Y) \geq H(X) - \underbrace{\boxed{R(X|Y)}}_{\text{Reconstruction error}}}$$

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)



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

MINE (Belghazi et al., 2018):

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

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

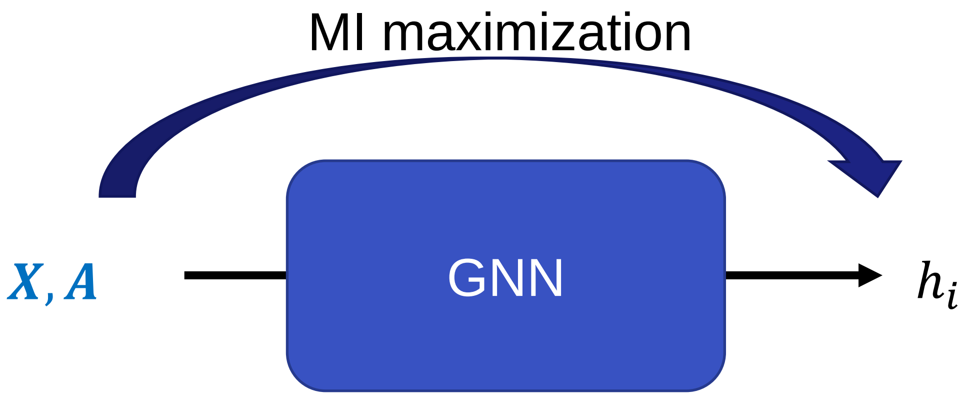
$$I^{\mathrm{JSD}}(X;Y) \triangleq E_{p(X,Y)}\left[\log \sigma\left(T_w(x,y)\right)\right] + E_{p(X)p(Y)}[\log(1-\sigma\left(T_w(x,y)\right))]$$

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

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

Deep Graph Infomax (DGI)

- (Velickovic et al. 2019)



The JSD MI estimator is applied：

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

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

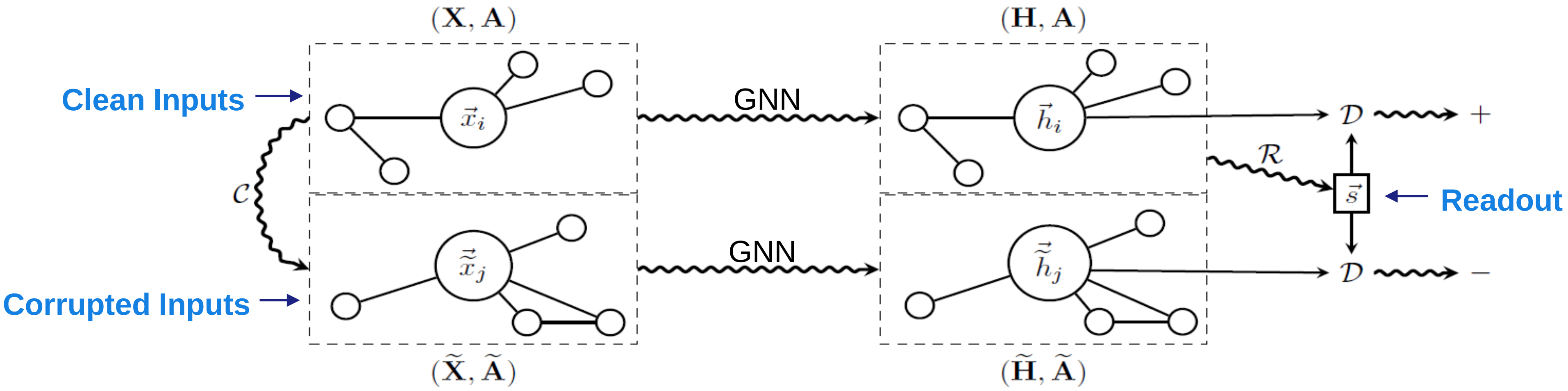
Deep Graph Infomax (DGI)

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

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

$$\Downarrow$$

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



Deep Graph Infomax (DGI)

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

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

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

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

- Some issues in DGI

  - Computing MI requires the injectivity of readout function

Indirect

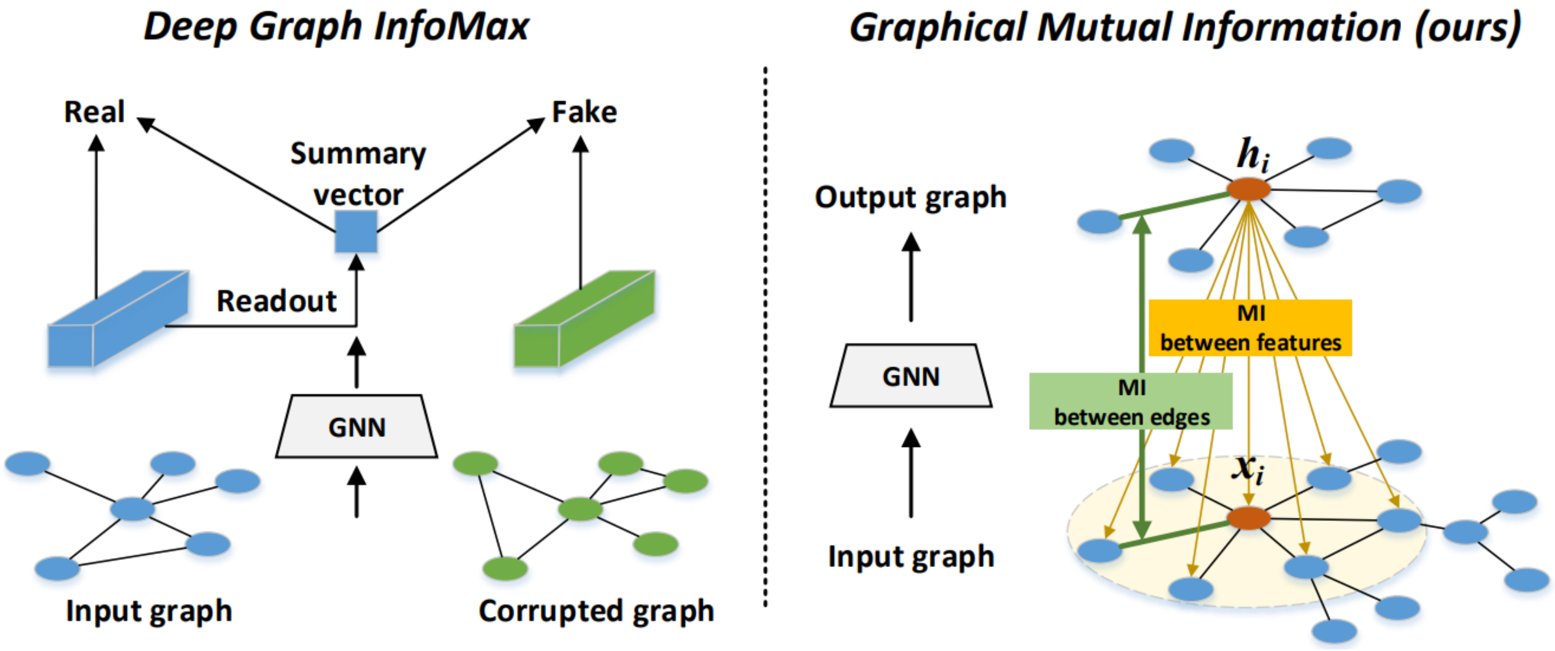
  - It resorts to graph corruption to generate negative samples

Inefficient

  - Distinct encoders and corruption functions for different tasks

GMI: Graphical Mutual Information

- (Peng et al. 2020)



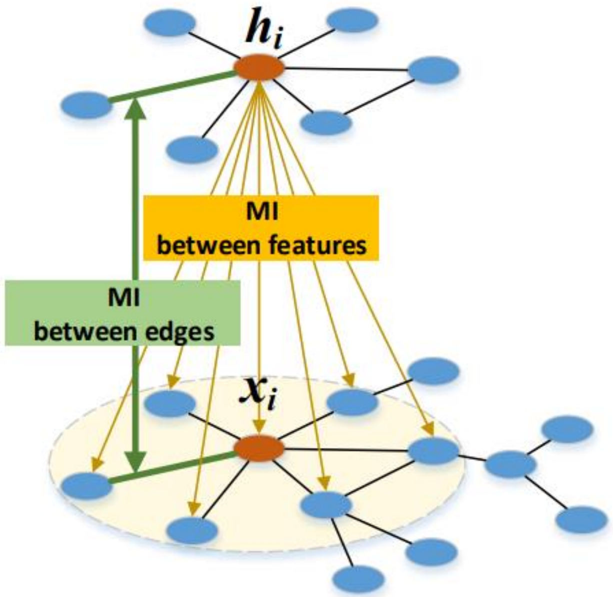
The basic idea of GMI is to compute the MI directly.

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

GMI: Graphical Mutual Information

We define that,

$$I(X,A;h_i) \approx \underbrace{\boxed{I(X;h_i)}}_{\text{Feature MI}} + \underbrace{\boxed{\sum_{j\in N(i)} I(\sigma({h_i}^T h_j);A_{ij})}}_{\text{Topology MI}}$$



- ➢ 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.

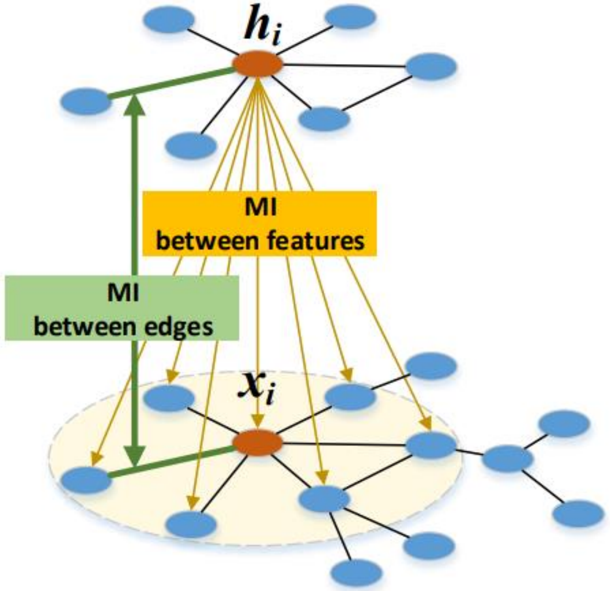
Peng et al. Graph Representation Learning via Graphical Mutual Information Maximization.

GMI: Graphical Mutual Information

- (Peng et al. 2020)

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

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



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

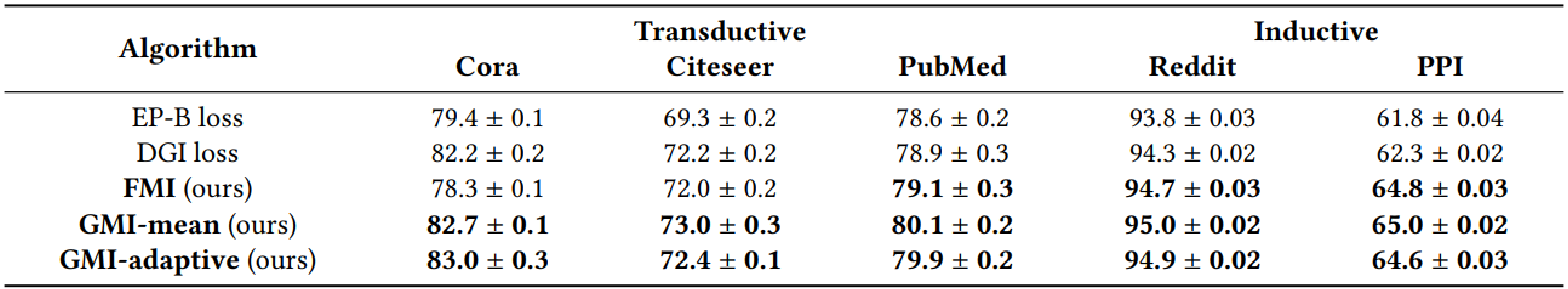
We then apply the JSD MI estimator to compute $I(x_j;\ h_i)$ and $I(\sigma({h_i}^T h_j); A_{ij})$

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**

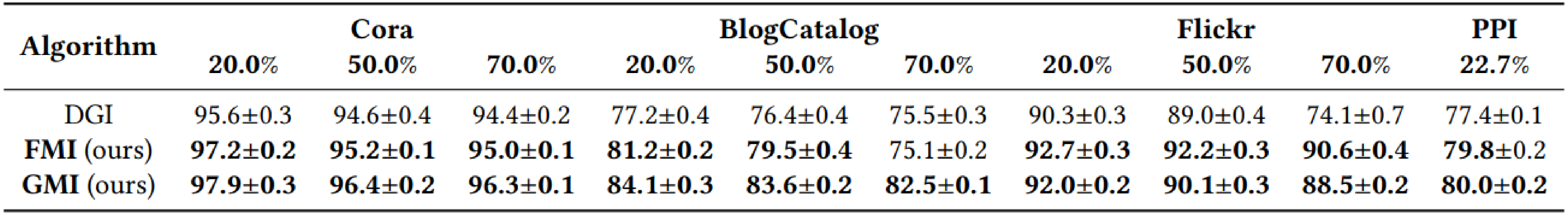
Codes: https://github.com/zpeng27/GMI

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.

Summarization

- Node classification
  - EP-B, GraphSAGE
  - DGI， GMI
- Graph classification
  - N-gram Graph, PreGNN, GCC, Grover
  - InfoGraph

Sun et al. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization, ICLR 2020

Applications

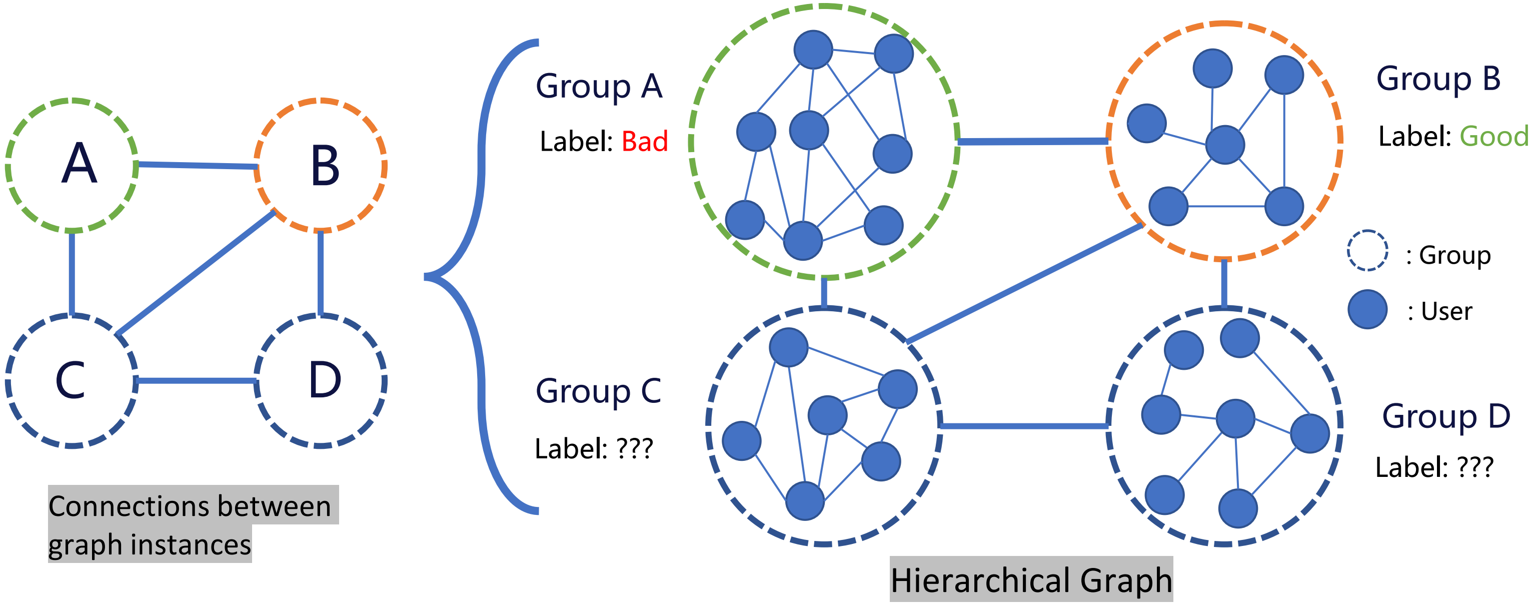
GNN in Social Networks

GNN in Social Networks

- "Semi-supervised graph classification: A hierarchical graph perspective." **WWW 2019**
- "Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks." **AAAI 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.

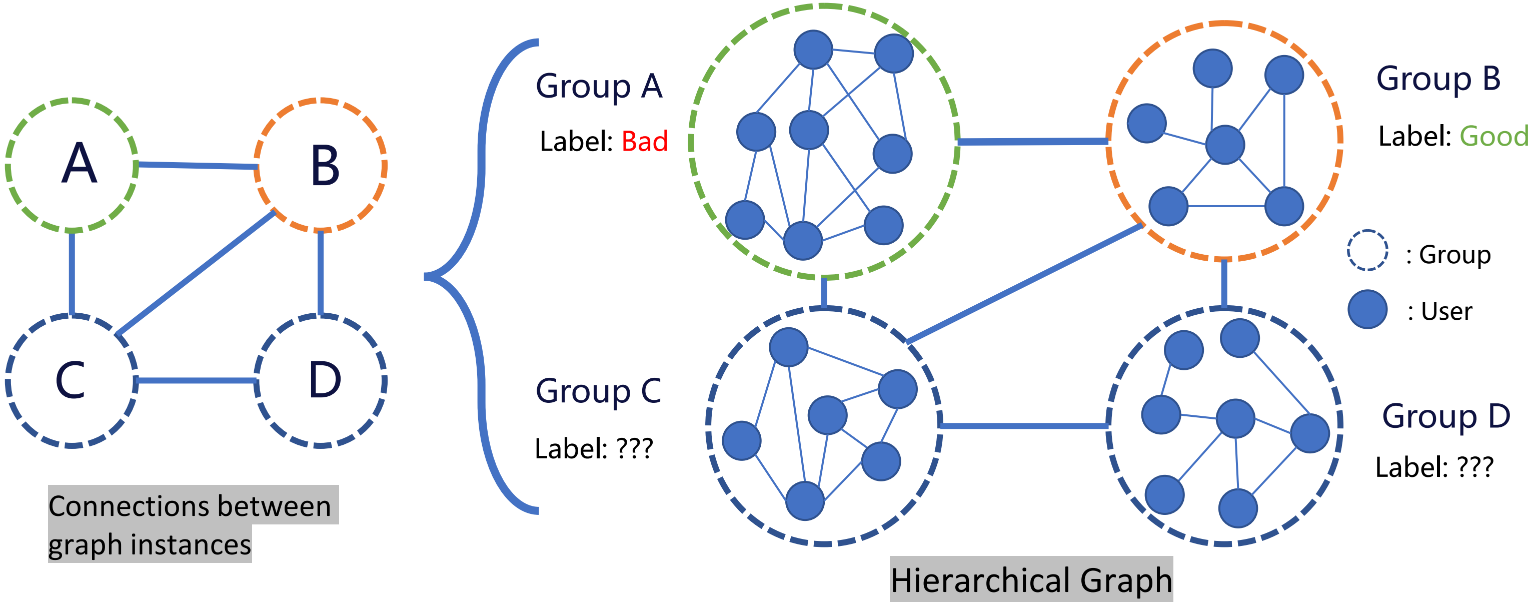


- **The Problem:** predicts the class label of graph instances in a hierarchical graph.

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?



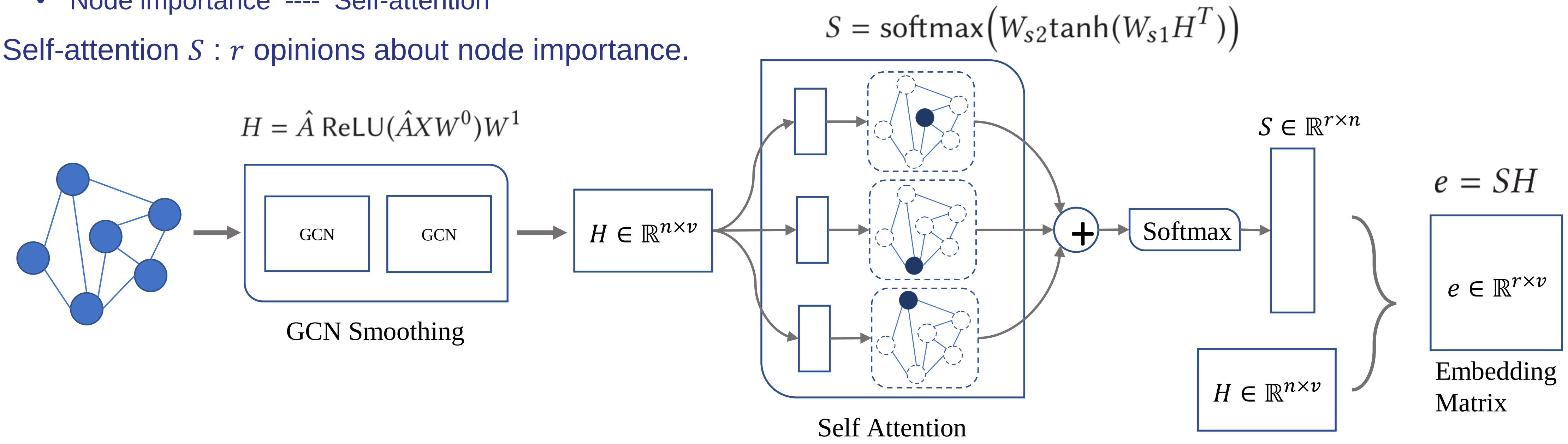
Connections between graph instances

Hierarchical Graph

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

Self-Attentive Graph Embedding

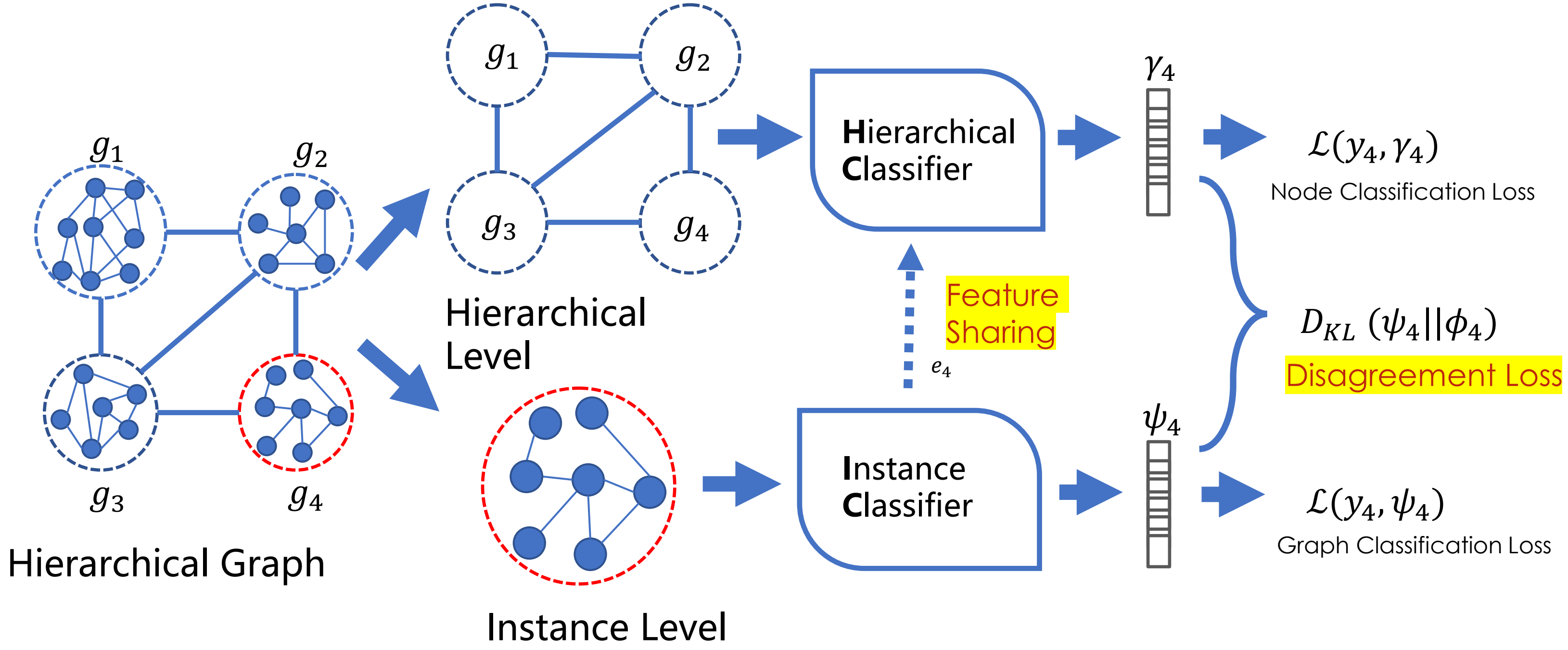
- How to represent the graphs with arbitrary size into a fixed-length vector?
- Graph representation learning at different level:
  - Node Level: $G(V,E) \rightarrow H^{n\times v}$
  - Graph Level: $G(V,E) \rightarrow e^{v}$
- **SAGE: Self-Attentive Graph Embedding**
  - Size invariance ---- Self-attention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


$S = \mathrm{softmax}\left(W_{s2}\tanh(W_{s1}H^T)\right)$
$H = \hat{A}\,\mathrm{ReLU}(\hat{A}XW^0)W^1$
GCN
GCN
GCN Smoothing
$H \in \mathbb{R}^{n\times v}$
Self Attention
Softmax
$S \in \mathbb{R}^{r\times n}$
$H \in \mathbb{R}^{n\times v}$
$e = SH$
$e \in \mathbb{R}^{r\times v}$
Embedding Matrix

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:

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

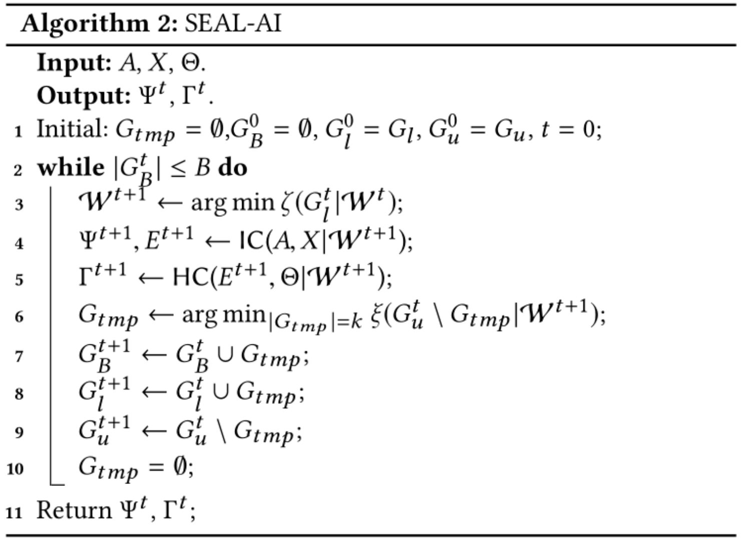
The disagreement loss:

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

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

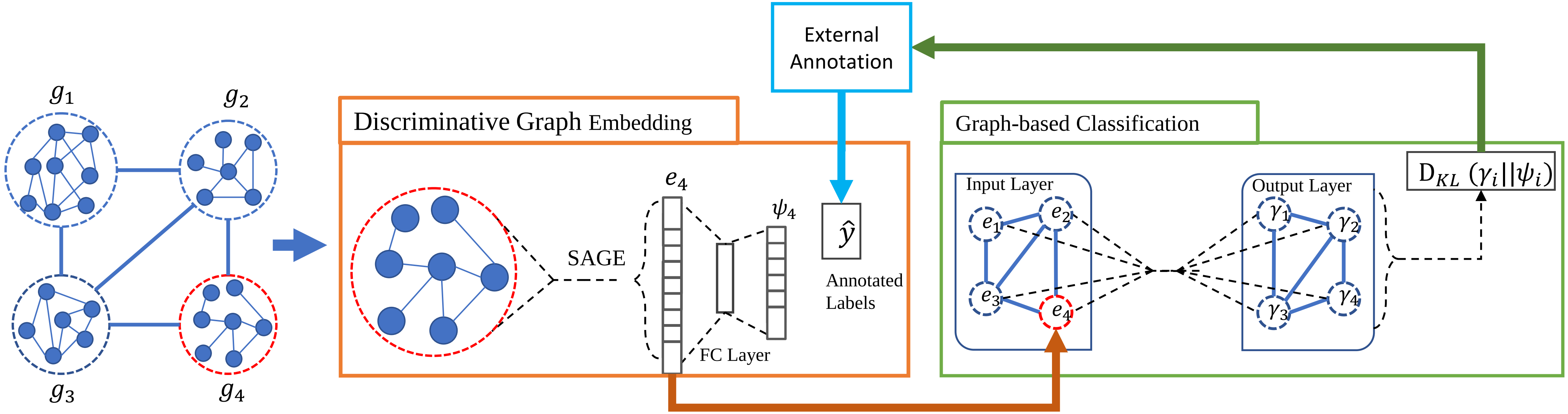
One More Thing: SEAL-AI/CI

- How to deal with the small mount of class label?
- Semi-supervised Graph Classification (SEAL)
    - **Active Learning Setting (SEAL-AI):** Choose the instances with large disagreement loss to annotate.



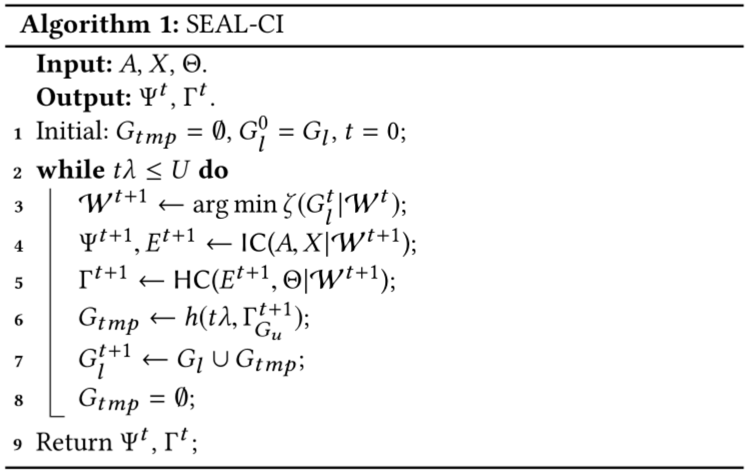
Algorithm 2: SEAL-AI

```
Input: $A, X, \Theta$.
Output: $\Psi^t, \Gamma^t$.
Initial: $G_{tmp} = \emptyset, G_B^0 = \emptyset, G_l^0 = G_l, G_u^0 = G_u, t = 0$;
while $|G_B^t| \leq B$ do
    $\mathcal{W}^{t+1} \leftarrow \arg\min \zeta(G_l^t | \mathcal{W}^t)$;
    $\Psi^{t+1}, E^{t+1} \leftarrow \mathsf{IC}(A, X | \mathcal{W}^{t+1})$;
    $\Gamma^{t+1} \leftarrow \mathsf{HC}(E^{t+1}, \Theta | \mathcal{W}^{t+1})$;
    $G_{tmp} \leftarrow \arg\min_{|G_{tmp}|=k} \xi(G_u^t \setminus G_{tmp} | \mathcal{W}^{t+1})$;
    $G_B^{t+1} \leftarrow G_B^t \cup G_{tmp}$;
    $G_l^{t+1} \leftarrow G_l^t \cup G_{tmp}$;
    $G_u^{t+1} \leftarrow G_u^t \setminus G_{tmp}$;
    $G_{tmp} = \emptyset$;
Return $\Psi^t, \Gamma^t$;
```



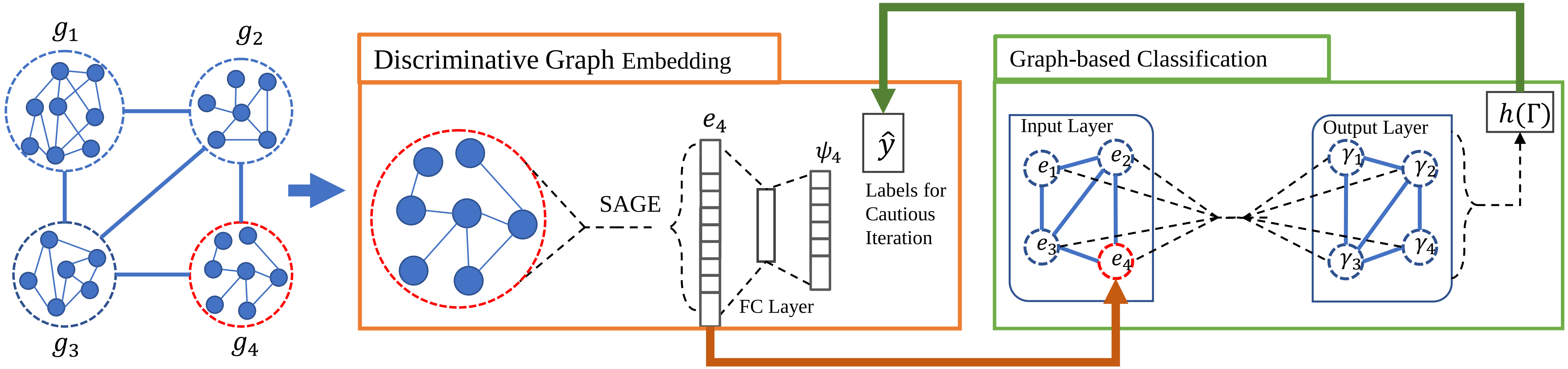
Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective."

SEAL-AI/CI



```
Algorithm 1: SEAL-CI
Input: $A, X, \Theta$.
Output: $\Psi^t, \Gamma^t$.
Initial: $G_{tmp} = \emptyset, G_l^0 = G_l, t = 0$;
while $t\lambda \leq U$ do
    $\mathcal{W}^{t+1} \leftarrow \arg\min \zeta(G_l^t | \mathcal{W}^t)$;
    $\Psi^{t+1}, E^{t+1} \leftarrow \mathsf{IC}(A, X | \mathcal{W}^{t+1})$;
    $\Gamma^{t+1} \leftarrow \mathsf{HC}(E^{t+1}, \Theta | \mathcal{W}^{t+1})$;
    $G_{tmp} \leftarrow h(t\lambda, \Gamma_{G_u}^{t+1})$;
    $G_l^{t+1} \leftarrow G_l \cup G_{tmp}$;
    $G_{tmp} = \emptyset$;
Return $\Psi^t, \Gamma^t$;
```

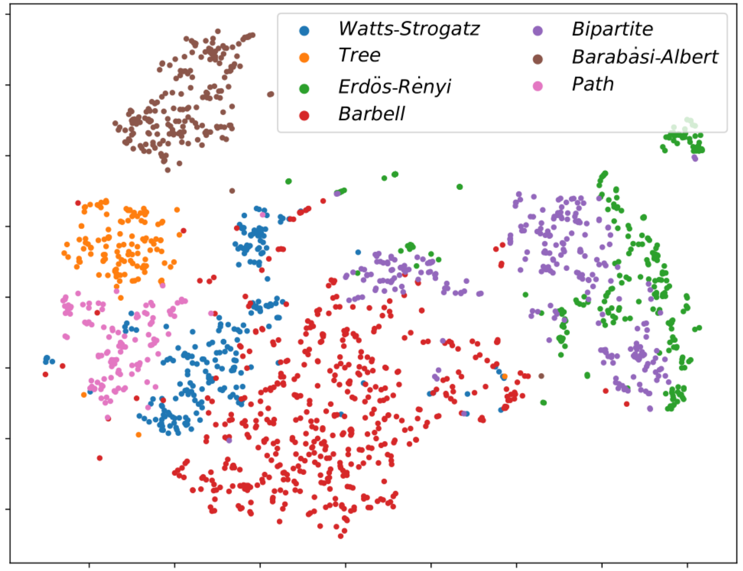
- How to deal with the small mount of class label?
- Semi-supervised Graph Classification (SEAL)
  - **Cautious Iteration Setting (SEAL-CI):** Choose the instances with maximum predicted probability and annotate them with the predicted class label.



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

Performance of SAGE

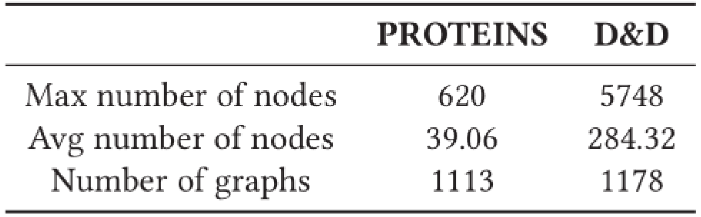
- Performance of synthesized graphs



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

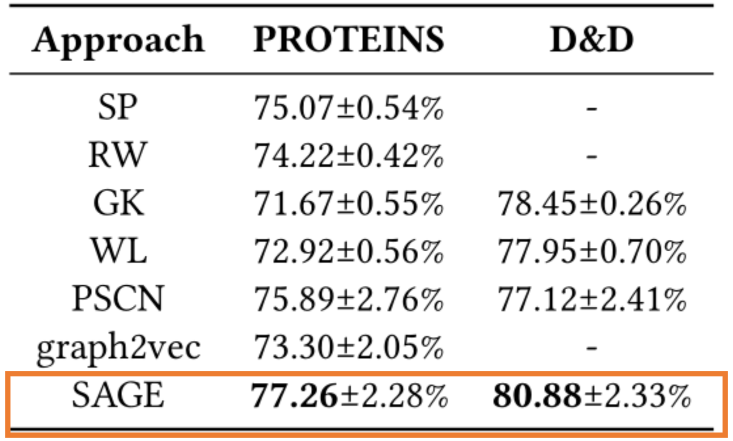
- Performance on the protein classification task:

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

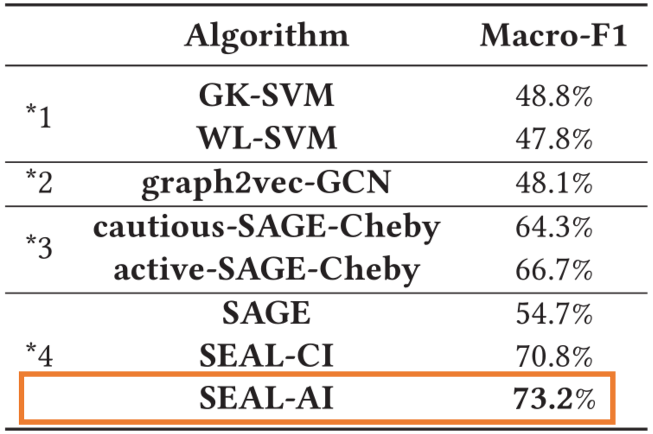


Approach	PROTEINS	D&D
SP	75.07±0.54%	-
RW	74.22±0.42%	-
GK	71.67±0.55%	78.45±0.26%
WL	72.92±0.56%	77.95±0.70%
PSCN	75.89±2.76%	77.12±2.41%
graph2vec	73.30±2.05%	-
SAGE	**77.26**±2.28%	**80.88**±2.33%

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

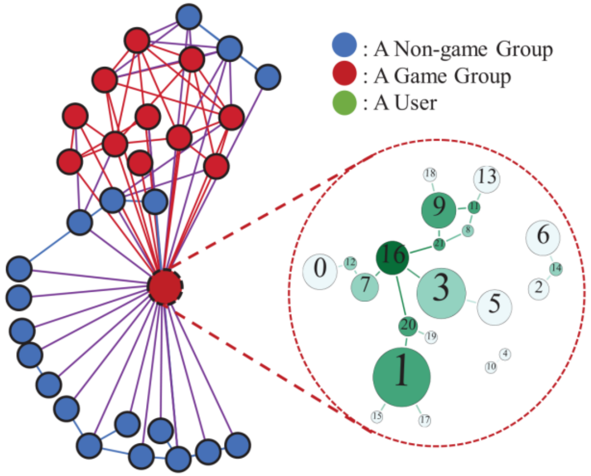
Performance of SEAL

- Real Datasets
    - User-Group Data
    - # of users: 18M
    - # of groups: 37K
    - Target: Predicting group label.



	Algorithm	Macro-F1
*1	GK-SVM	48.8%
	WL-SVM	47.8%
*2	graph2vec-GCN	48.1%
*3	cautious-SAGE-Cheby	64.3%
	active-SAGE-Cheby	66.7%
*4	SAGE	54.7%
	SEAL-CI	70.8%
	SEAL-AI	73.2%

- The visualization of the hierarchical graph for a "game" group.



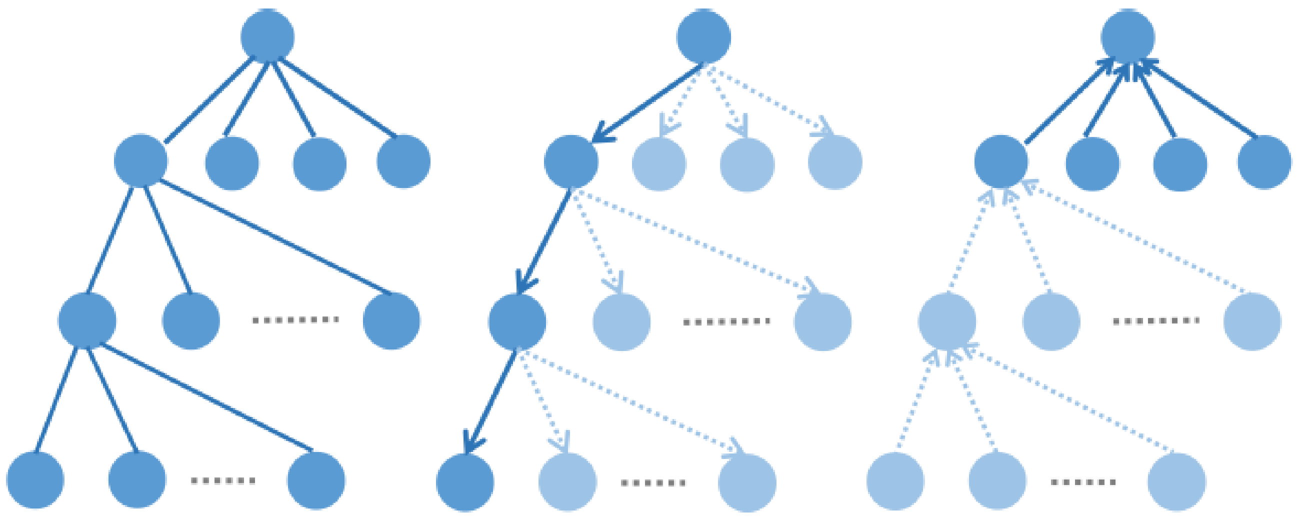
Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective."

GNN in Social Networks

- "Semi-supervised graph classification: A hierarchical graph perspective." **WWW 2019**
- "Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks." **AAAI 2020**

Bi-GCN: Rumor Detection

- Rumor tree: rumors spread like a tree in the social network. Rumor has two major characteristics:
  - **Propagation**: deep spread **along a relationship chain**
  - **Dispersion**: widespread **across a social community**

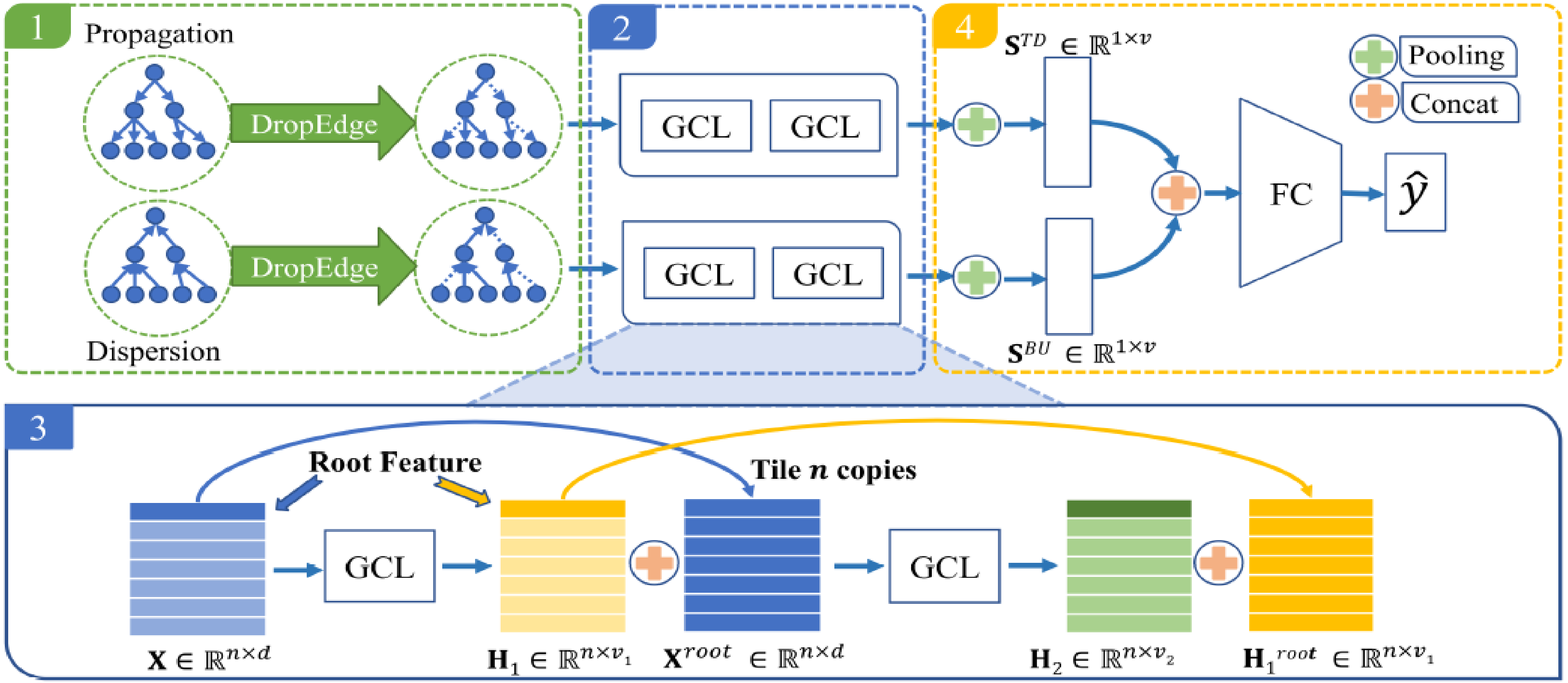


(a) Rumor Tree (b) Propagation (c) Dispersion

Bian, Tian, et al. "Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks." AAAI 2020.

Bi-GCN: Rumor Detection

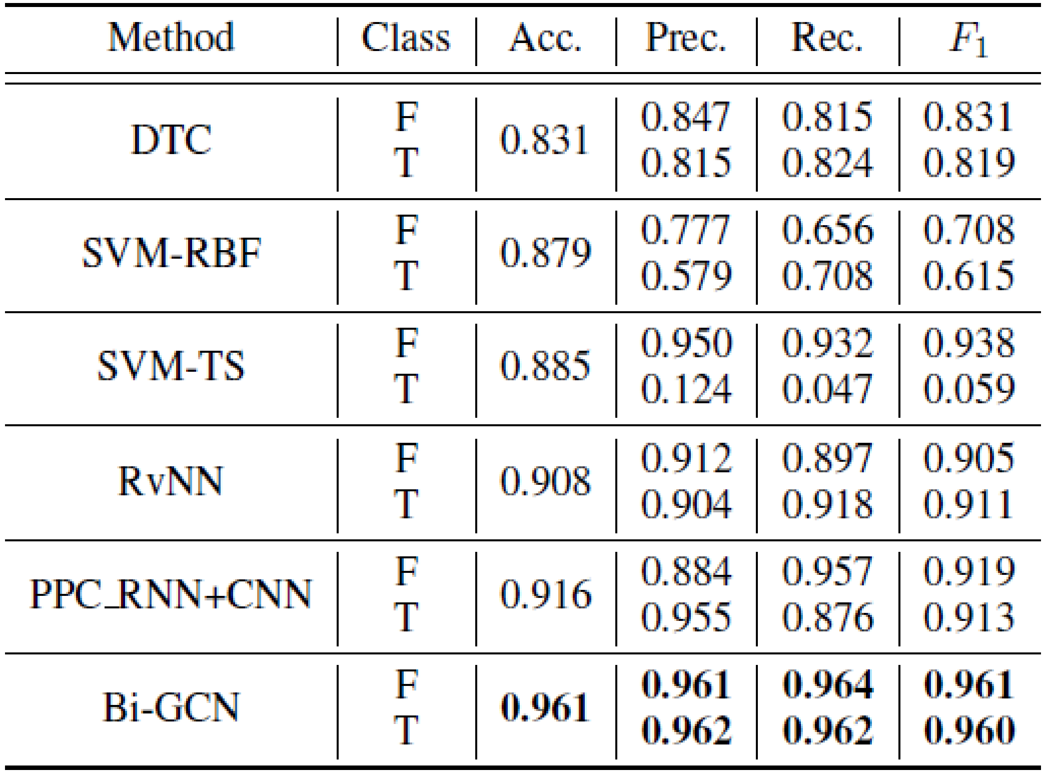
- Bi-directed GCN:
  1. Construct directed **Propagation** and **Dispersion** graphs for rumors
  2. Calculate high-level node representations via **GCNs**
  3. Concatenate **root features** to enhance the performance
  4. Classify rumors from **Representations** of propagation and dispersion


1
Propagation
DropEdge
DropEdge
Dispersion
2
GCL
GCL
GCL
GCL
4
$\mathbf{S}^{TD} \in \mathbb{R}^{1\times v}$
$\mathbf{S}^{BU} \in \mathbb{R}^{1\times v}$
Pooling
Concat
FC
$\hat{y}$
3
Root Feature
Tile n copies
GCL
GCL
$\mathbf{X} \in \mathbb{R}^{n\times d}$
$\mathbf{H}_1 \in \mathbb{R}^{n\times v_1}$
$\mathbf{X}^{root} \in \mathbb{R}^{n\times d}$
$\mathbf{H}_2 \in \mathbb{R}^{n\times v_2}$
$\mathbf{H}_1{}^{root} \in \mathbb{R}^{n\times v_1}$

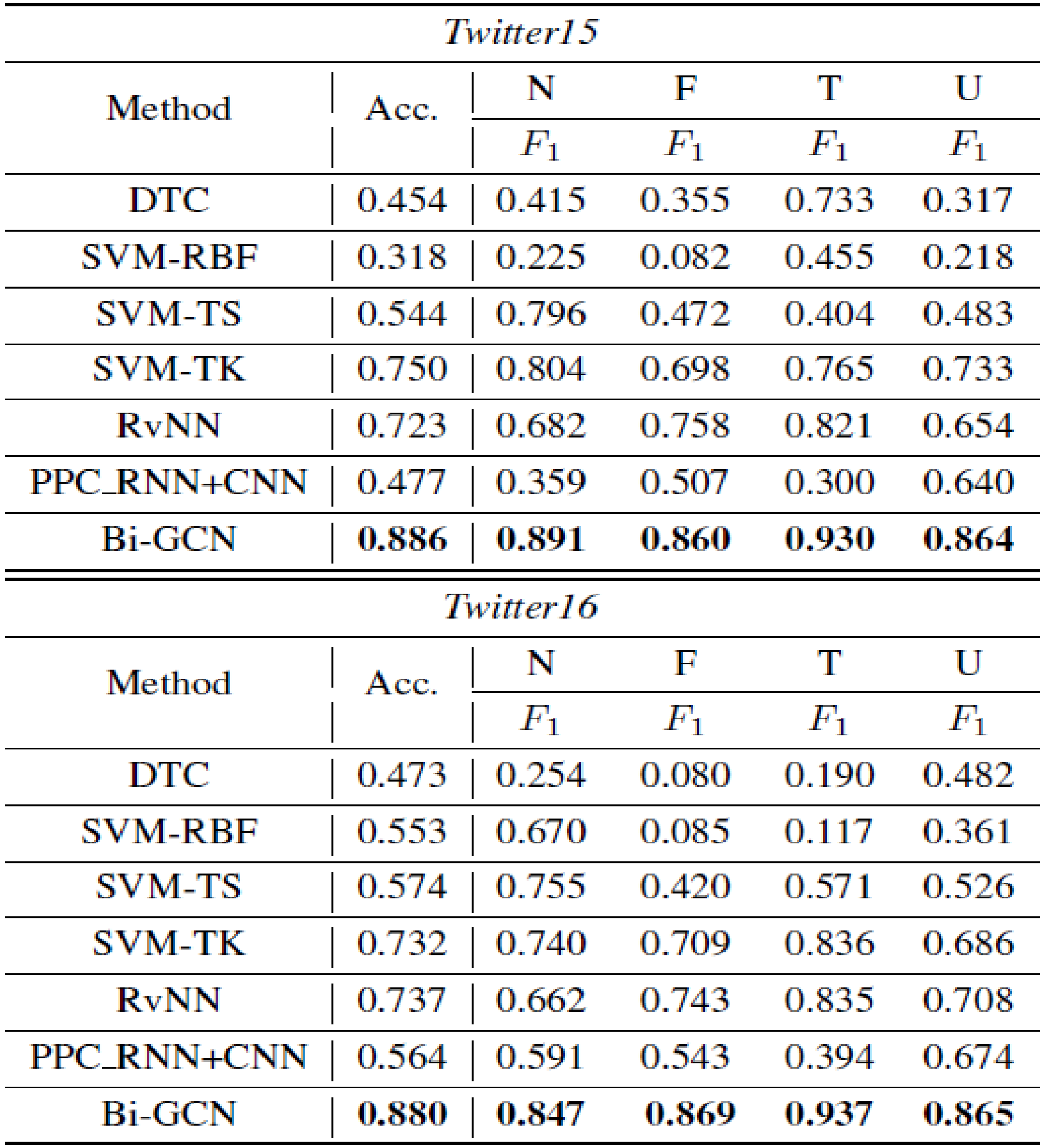
Bian, Tian, et al. "Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks." AAAI 2020.

Bi-GCN: Rumor Detection: results

- We tested on 3 datasets:
  - Twitter15
  - Twitter16
  - Weibo



Method	Class	Acc.	Prec.	Rec.	F_1
DTC	F T	0.831	0.847 0.815	0.815 0.824	0.831 0.819
SVM-RBF	F T	0.879	0.777 0.579	0.656 0.708	0.708 0.615
SVM-TS	F T	0.885	0.950 0.124	0.932 0.047	0.938 0.059
RvNN	F T	0.908	0.912 0.904	0.897 0.918	0.905 0.911
PPC_RNN+CNN	F T	0.916	0.884 0.955	0.957 0.876	0.919 0.913
Bi-GCN	F T	**0.961**	**0.961** **0.962**	**0.964** **0.962**	**0.961** **0.960**



Twitter15

Method	Acc.	N	F	T	U
		F_1	F_1	F_1	F_1
DTC	0.454	0.415	0.355	0.733	0.317
SVM-RBF	0.318	0.225	0.082	0.455	0.218
SVM-TS	0.544	0.796	0.472	0.404	0.483
SVM-TK	0.750	0.804	0.698	0.765	0.733
RvNN	0.723	0.682	0.758	0.821	0.654
PPC_RNN+CNN	0.477	0.359	0.507	0.300	0.640
Bi-GCN	**0.886**	**0.891**	**0.860**	**0.930**	**0.864**

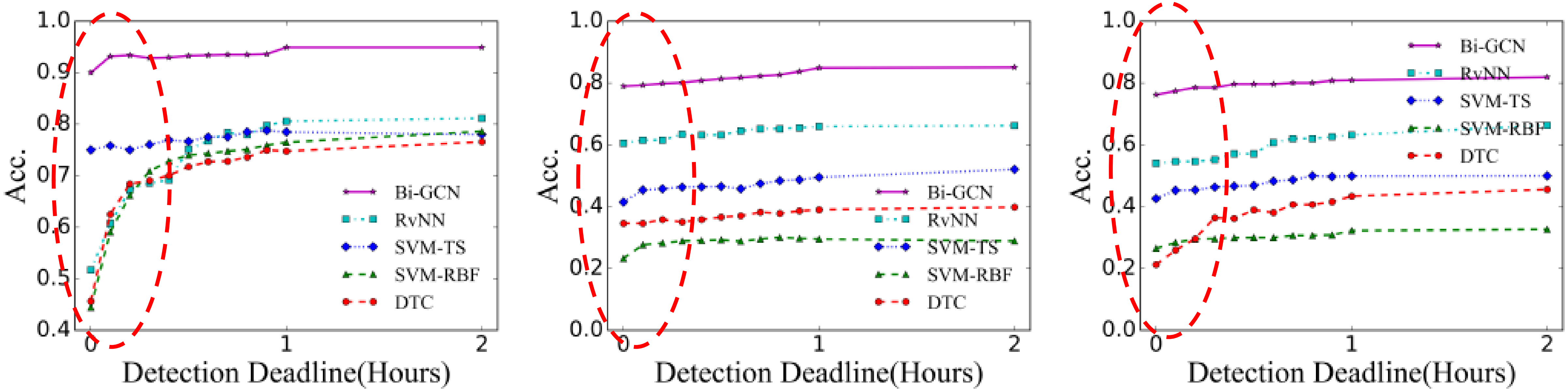
Twitter16

Method	Acc.	N	F	T	U
		F_1	F_1	F_1	F_1
DTC	0.473	0.254	0.080	0.190	0.482
SVM-RBF	0.553	0.670	0.085	0.117	0.361
SVM-TS	0.574	0.755	0.420	0.571	0.526
SVM-TK	0.732	0.740	0.709	0.836	0.686
RvNN	0.737	0.662	0.743	0.835	0.708
PPC_RNN+CNN	0.564	0.591	0.543	0.394	0.674
Bi-GCN	**0.880**	**0.847**	**0.869**	**0.937**	**0.865**

Bian, Tian, et al. "Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks." AAAI 2020.

Bi-GCN: Rumor Detection: early detection

- Early detection of rumors



(a) *Weibo* dataset (b) *Twitter15* dataset (c) *Twitter16* dataset

Github: https://github.com/TianBian95/BiGCN

Bian, Tian, et al. "Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks." AAAI 2020.

Applications

GNN in Medical Imaging

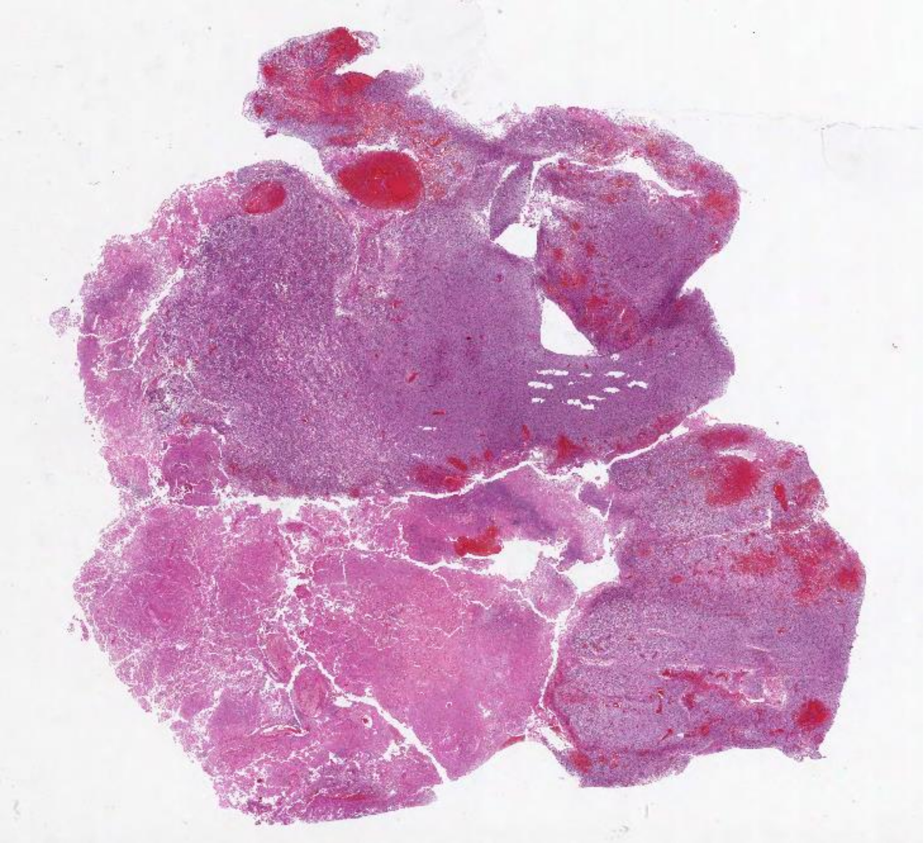
GNN in Medical Imaging

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

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

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

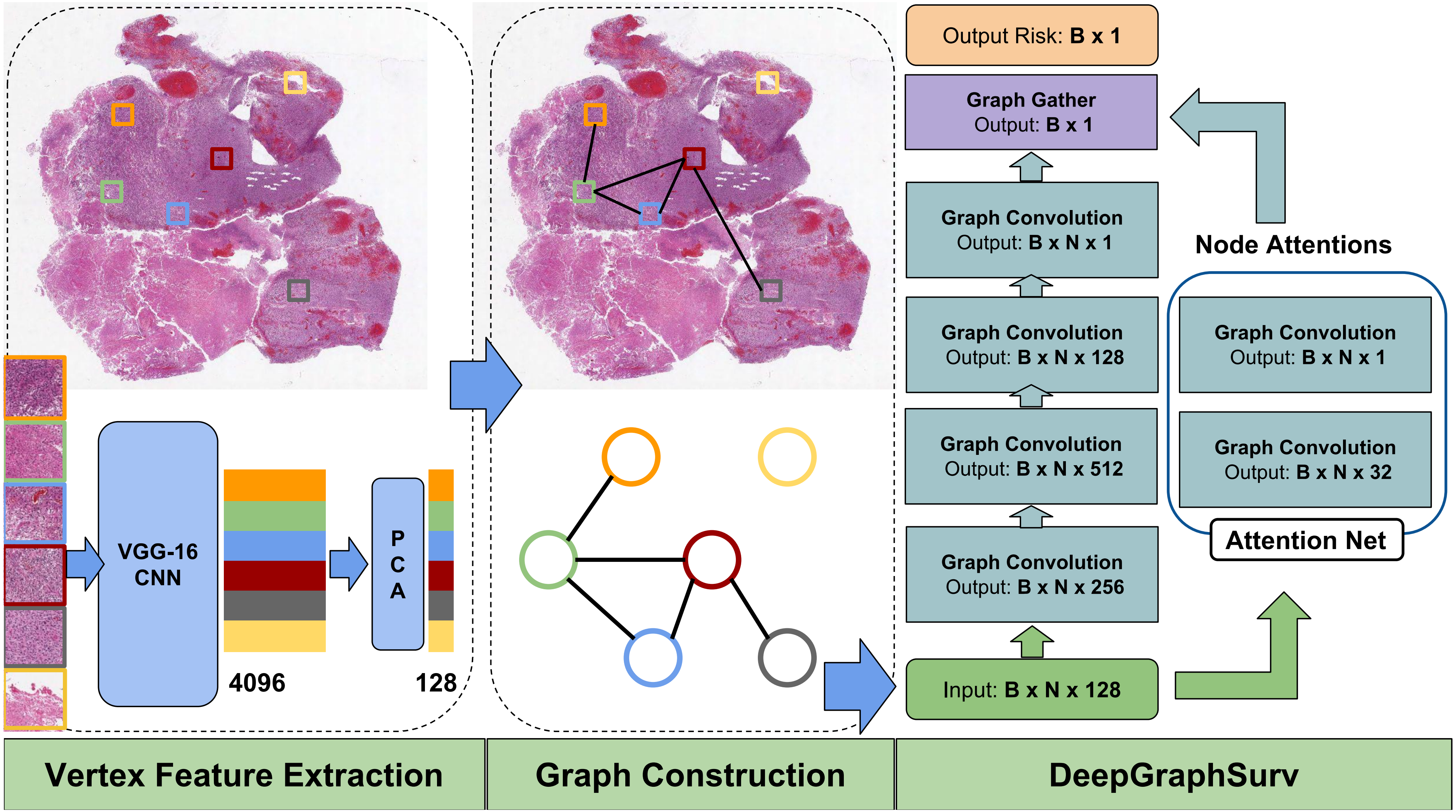
$$\theta_j = \exp(X_j \cdot \beta)$$

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

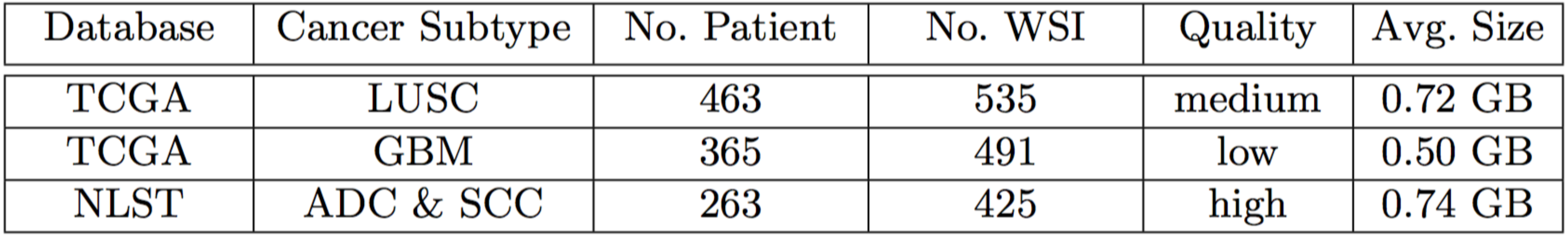
- **Join likelihood of all subjects:** $L(\beta) = \prod_{i:C_i=1} L_i(\beta)$

- **Log likelihood as object function:**

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


VGG-16 CNN
4096
P C A
128
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
Vertex Feature Extraction
Graph Construction
DeepGraphSurv

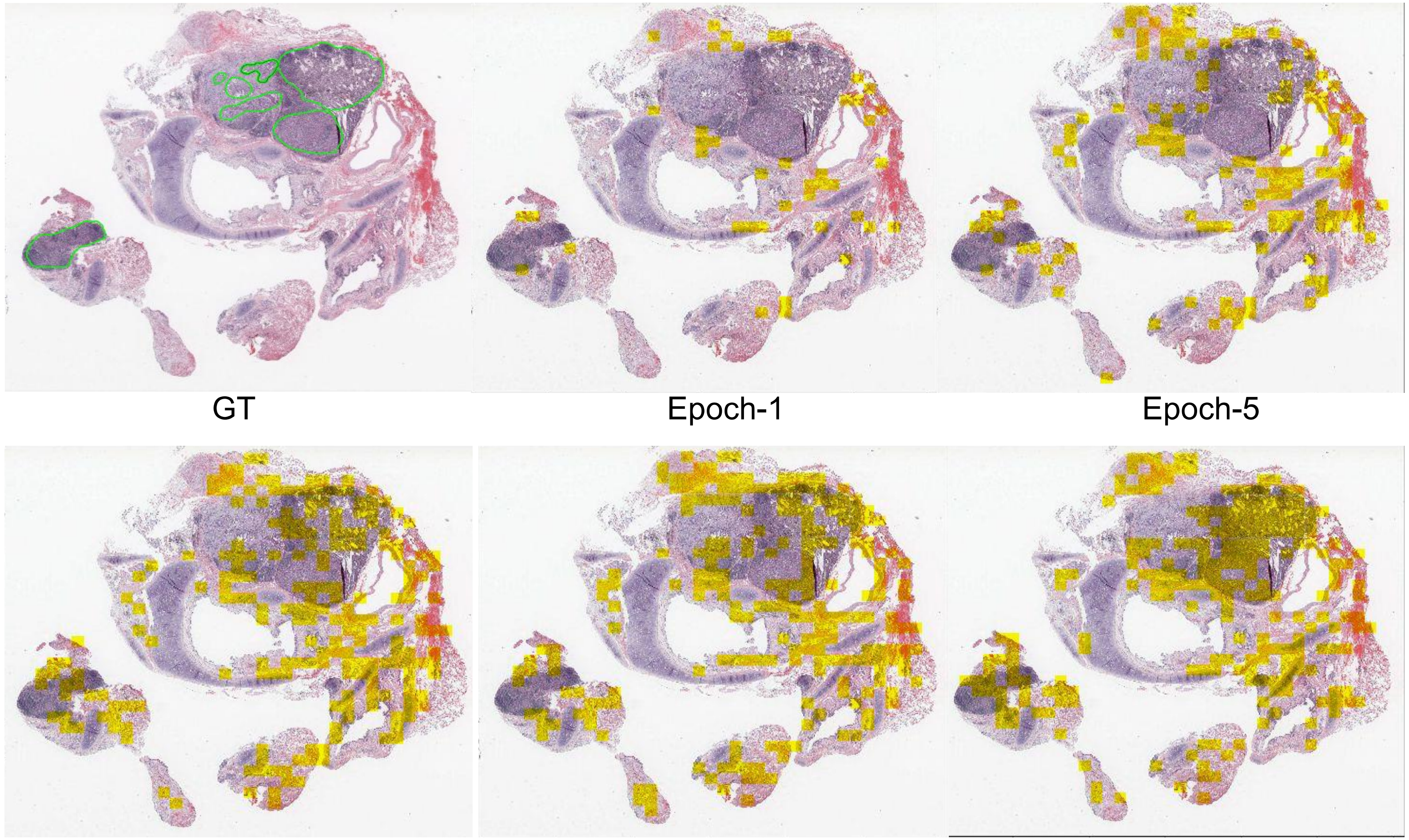
- **Pathological Images and Patient Survival Time and Label**
  - TCGA, The Cancer Genome Atlas
  - NLST, National Lung Screening Trials



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))



GT

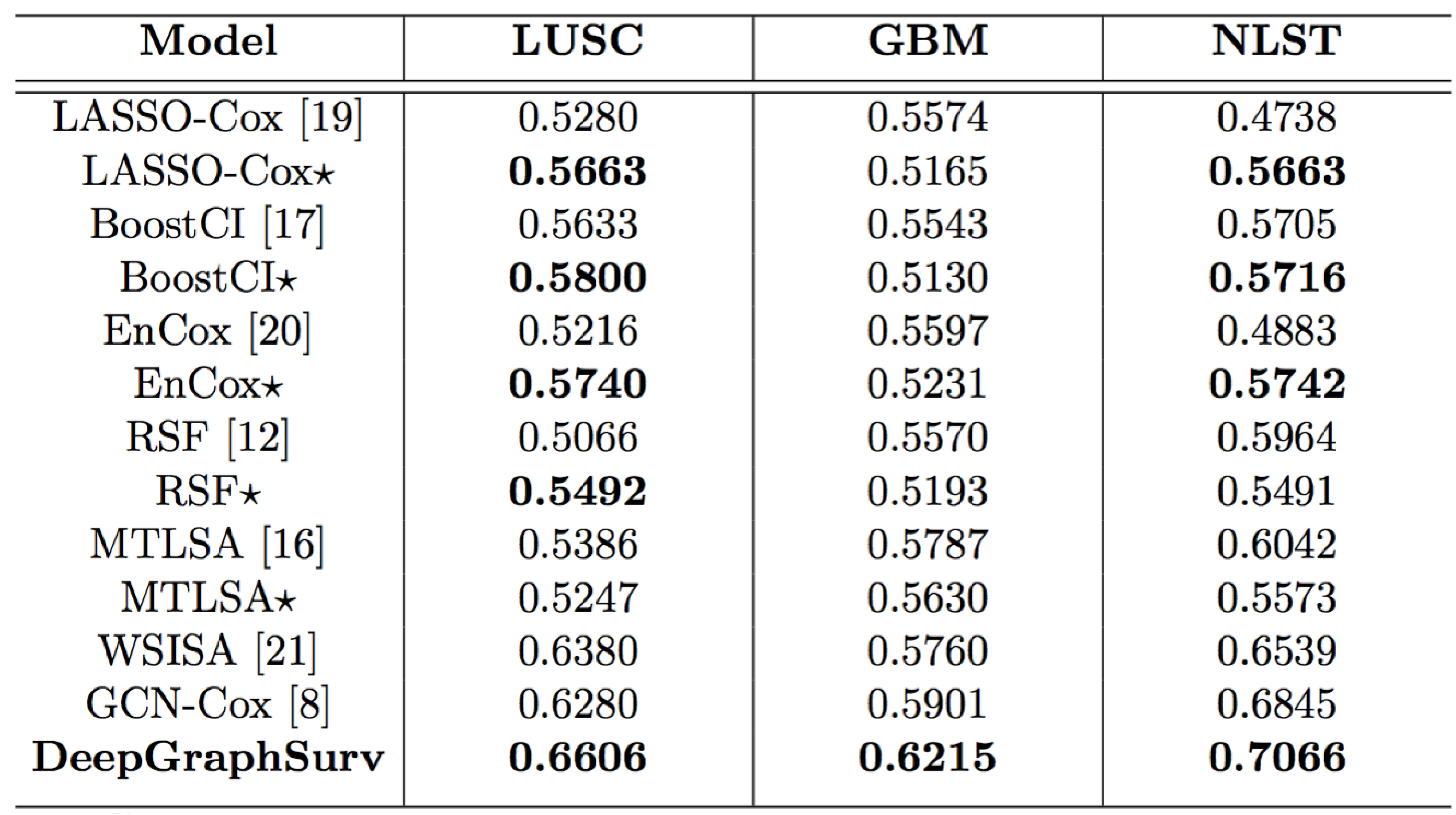
Epoch-1

Epoch-5

Epoch-10

Epoch-15

Epoch-40

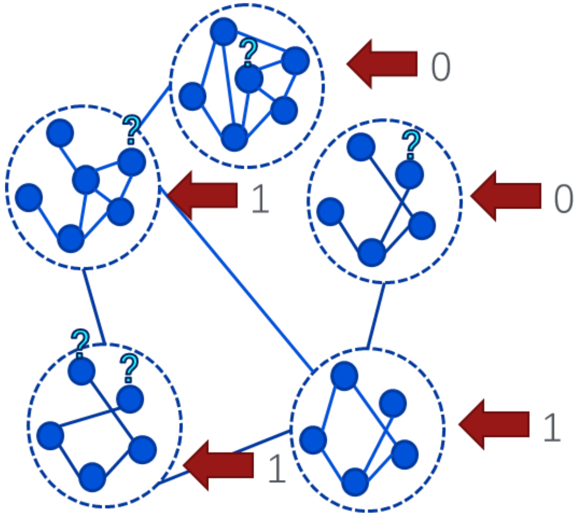


Model	LUSC	GBM	NLST
LASSO-Cox [19]	0.5280	0.5574	0.4738
LASSO-Cox⋆	**0.5663**	0.5165	**0.5663**
BoostCI [17]	0.5633	0.5543	0.5705
BoostCI⋆	**0.5800**	0.5130	**0.5716**
EnCox [20]	0.5216	0.5597	0.4883
EnCox⋆	**0.5740**	0.5231	**0.5742**
RSF [12]	0.5066	0.5570	0.5964
RSF⋆	**0.5492**	0.5193	0.5491
MTLSA [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**

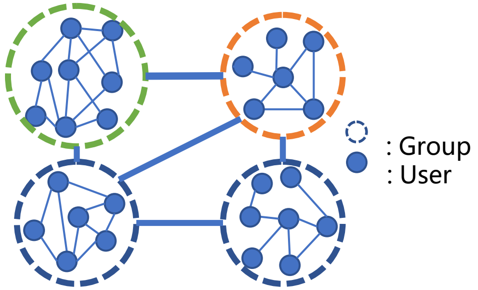
* Use our graph features for the survival model.

Future Directions

Graph Multi-instance Learning



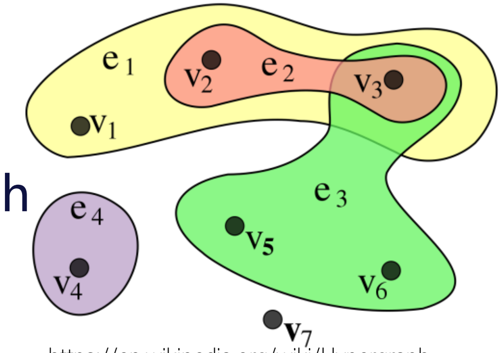
Graph Defense



Hierarchical Graph

Graph Attack

Hypergraph

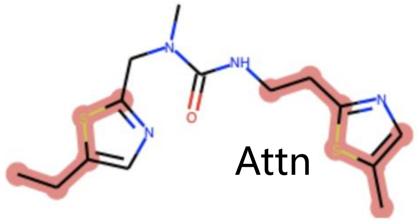


https://en.wikipedia.org/wiki/Hypergraph

Inverse Graph Identification

"Inverse Graph Identification: Can We Identify Node Labels Given Graph Labels?." *arXiv preprint arXiv:2007.05970* (2020).

Subgraph Recognition



Future Directions

Bibliography

Bibliography

- Sperduti, Alessandro, and Antonina Starita. "Supervised neural networks for the classification of structures." *IEEE Transactions on Neural Networks* 8.3 (1997): 714-735.
- 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.
- Scarselli, Franco, et al. "The graph neural network model." *IEEE Transactions on Neural Networks* 20.1 (2008): 61-80.
- Li, Yujia, et al. "Gated graph sequence neural networks." arXiv preprint arXiv:1511.05493 (2015).
- Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." Advances in neural information processing systems. 2016.
- 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).
- Bruna, Joan, et al. "Spectral networks and locally connected networks on graphs." arXiv preprint arXiv:1312.6203 (2013).
- Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov. "Learning convolutional neural networks for graphs." International conference on machine learning. 2016.
- Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." arXiv preprint arXiv:1609.02907 (2016).
- Xu, Bingbing, et al. "Graph Wavelet Neural Network." International Conference on Learning Representations. 2018.
- Chami, Ines, et al. "Hyperbolic graph convolutional neural networks." Advances in neural information processing systems. 2019.
- Liao, Renjie, et al. "LanczosNet: Multi-Scale Deep Graph Convolutional Networks." International Conference on Learning Representations. 2018.
- Veličković, Petar, et al. "Graph Attention Networks." International Conference on Learning Representations. 2018.
- Zhang, Jiani, et al. "Gaan: Gated attention networks for learning on large and spatiotemporal graphs." arXiv preprint arXiv:1803.07294 (2018).
- Chang, Heng, et al. "Spectral Graph Attention Network." arXiv preprint arXiv:2003.07450 (2020).
- Ying, Zhitao, et al. "Hierarchical graph representation learning with differentiable pooling." Advances in neural information processing systems. 2018.
- Ma, Yao, et al. "Graph convolutional networks with eigenpooling." Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 2019.
- Atwood, James, and Don Towsley. "Diffusion-convolutional neural networks." Advances in neural information processing systems. 2016.
- Abu-El-Haija, Sami, et al. "MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing." International Conference on Machine Learning. 2019.
- Klicpera, Johannes, Aleksandar Bojchevski, and Stephan Günnemann. "Predict then Propagate: Graph Neural Networks meet Personalized PageRank." International Conference on Learning Representations. 2018.
- Gilmer, Justin, et al. "Neural Message Passing for Quantum Chemistry." ICML. 2017.
- Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective." The World Wide Web Conference. 2019.

Bibliography

- Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." Advances in neural information processing systems. 2017.
- 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.
- Huang, Wenbing, et al. "Adaptive sampling towards fast graph representation learning." Advances in neural information processing systems. 2018.
- 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.
- Zeng, Hanqing, et al. "GraphSAINT: Graph Sampling Based Inductive Learning Method." International Conference on Learning Representations. 2020.
- Morris et al. Weisfeiler and Leman Go Neural Higher-order Graph Neural Networks. AAAI, 2019.
- Xu et al. How Powerful Are Graph Neural Networks. ICLR, 2019.
- Maron et al. Invariant and Equivariant Graph Networks. ICLR, 2019.
- Maron et al. On the Universality of Invariant Networks. ICML, 2019.
- Maron et al. Provably Powerful Graph Networks, NeurIPS. 2019.
- Dehmamy et al. Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology. NeurIPS, 2019.
- Sato et al. Approximation Ratios of Graph Neural Networks for Combinatorial Problems. NeurIPS, 2019.
- Loukas, What graph neural networks cannot learn: depth vs width. ICLR, 2020.
- Garg et al. Generalization and Representational Limits of Graph Neural Networks. ICML, 2020.
- Shervashidze et al. Weisfeiler-Lehman Graph Kernels. JRML, 2011.
- Cybenko, G. Approximation by superpositions of a sigmoidal function. Mathematics of control, signals and systems, 2(4):303–314, 1989.
- Hornik, K. Approximation capabilities of multilayer feedforward networks. Neural networks, 4(2):251–257, 1991.
- Chen et al. On the equivalence between graph isomorphism testing and function approximation with GNNs. NeurIPS, 2019.
- Kipf & Welling. Variational Graph Auto-Encoders. arXiv, 2016.
- Durán & Niepert. Learning Graph Representations with Embedding Propagation. NeurIPS, 2017.
- Liu et al. N-Gram Graph: Simple Unsupervised Representation for Graphs, with Applications to Molecules. NeurIPS, 2019.
- Hu et al. Strategies for Pre-training Graph Neural Networks. ICLR, 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.
- Veličković et al. Deep Graph Infomax. ICLR, 2019.
- Peng et al. Graph Representation Learning via Graphical Mutual Information Maximization. WWW, 2020.
- Sun et al. InfoGraph: Unsupervised and Semi-supervised Graph-Level Representation Learning via Mutual Information Maximization. ICLR, 2020.

Bibliography

- Hinton, G. E., & Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. Science, 2006.
- Vincent et al. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. JMLR, 2010.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. NeurIPS, 2018.
- Belghazi et al. Mine: mutual information neural estimation. ICML, 2018.
- Nowozin et al. f-gan: Training generative neural samplers using variational divergence minimization. NeurIPS, 2016.
- Hjelm et al. Learning deep representations by mutual information estimation and maximization. ICLR, 2019.
- Wang, X., & Gupta, A. Videos as Space-Time Region Graphs. ECCV, 2018.
- Yan et al. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. AAAI, 2018.
- Zeng, et al. Graph convolutional networks for temporal action localization. ICCV, 2019
- Bian, Tian, et al. "Rumor Detection on Social Media with Bi-Directional Graph Convolutional Networks." AAAI 2020.