Advanced Topics in Scalable Learning

CSE 6392 Lecture 2 Graph Neural Network

Junzhou Huang, Ph.D.

Department of Computer Science and Engineering

UT Arlington

Administration

- **Course CSE6392**
  - What: Advanced Topics in Scalable Learning
  - When: Friday 1:00 ~ 3:50pm
  - Where: NH 109
  - Who: Junzhou Huang (Office ERB 650) jzhuang@uta.edu
  - Office Hour: FRIDAY 3:50 ~ 6:00pm and/or appointments
  - Webpage: http://ranger.uta.edu/~huang/teaching/CSE6392.htm

    (**Please check this page regularly**)
- **Lecturer**
  - PhD in CS from Rutgers, the State University of New Jersey
  - Research areas: machine learning , computer vision, medical image analysis and bioinformatics
- **GTA**
  - Qifeng Zhou (Office ERB 105B), qxz8706@mavs.uta.edu
  - Office hours: Friday 10:00am ~ 12:00pm and/or appointments

Assignment

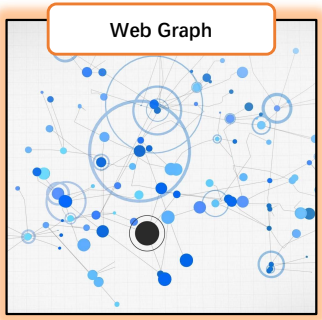
- **Paper Selection**
  - Each group has two members at most.
  - Each group will select at least one paper from the following paper list and then be scheduled to present their selected papers in our class.
  - You can choose any papers from the paper lists
  - Please talk to the lecturer if you prefer to select a paper out of the list
  - The selected paper has to be confirmed by the second week (before the second class)
  - GTA will set up the paper selection sheet
  - Different groups will present different papers

Grading

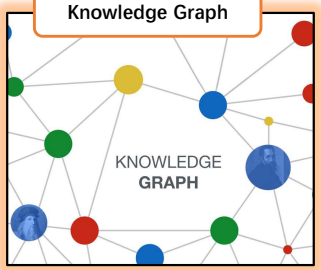
- **Distribution**
  - 30% Paper Presentation
  - 30% Slide Preparation
  - 30% Questions & Answering
  - 10% Class Participation
  - 100%
- **Attention**
  - No midterm or final exam for this course.
  - Please read the selected paper and prepare the final presentation as early as possible
  - This is research seminar course. Asking questions and discussion are highly encouraged
  - When missing a class due to unavoidable circumstances, PLEASE notify the instructor in advance with any notes/evidences

Graph is Everywhere

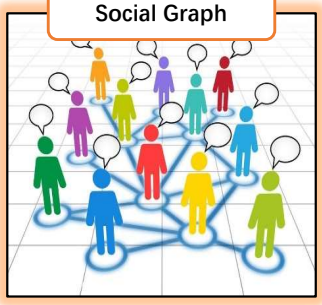
Web Graph



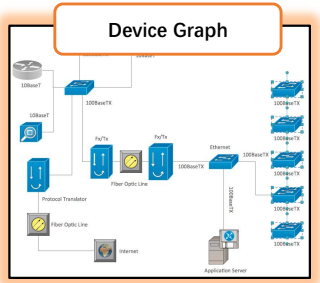
Knowledge Graph



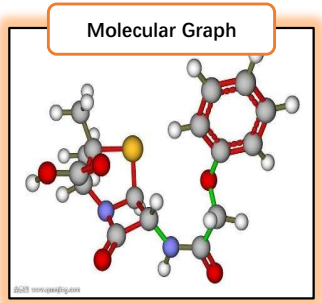
Social Graph



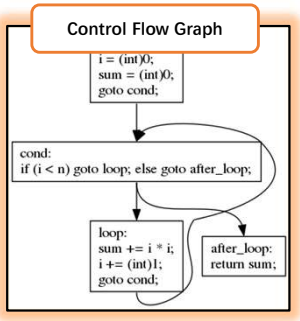
Device Graph



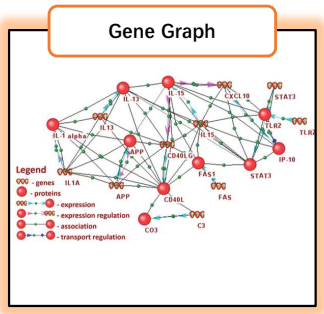
Molecular Graph



Control Flow Graph



Gene Graph

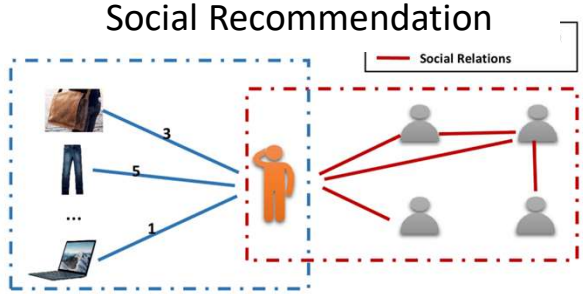


Brain Graph

Graph is Important

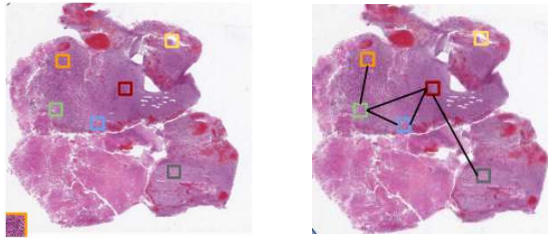
- Numerous real-world problems can be summarized as a set of tasks on graphs.

Social Recommendation



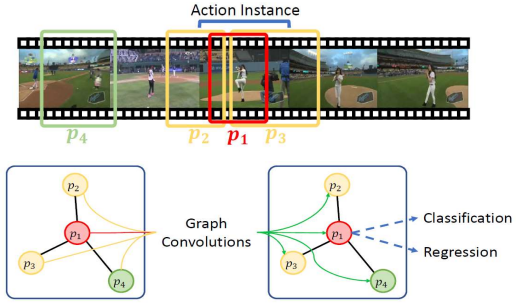
Fan W, Ma Y, Li Q, et al. 2019

Survival Analysis



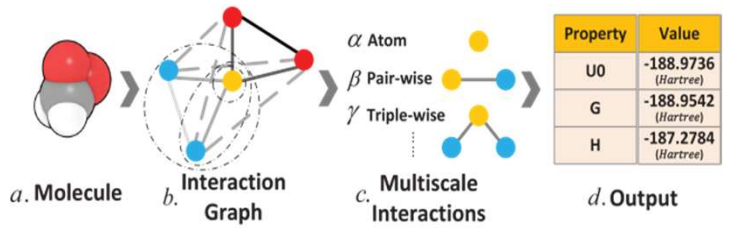
Li R, Yao J, Zhu X, et al. 2018

Temporal Action Localization



Zeng et al. 2019

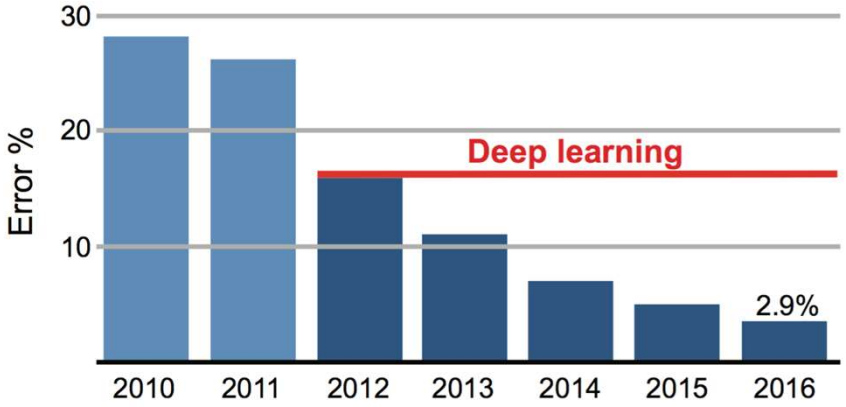
Molecular Property Prediction

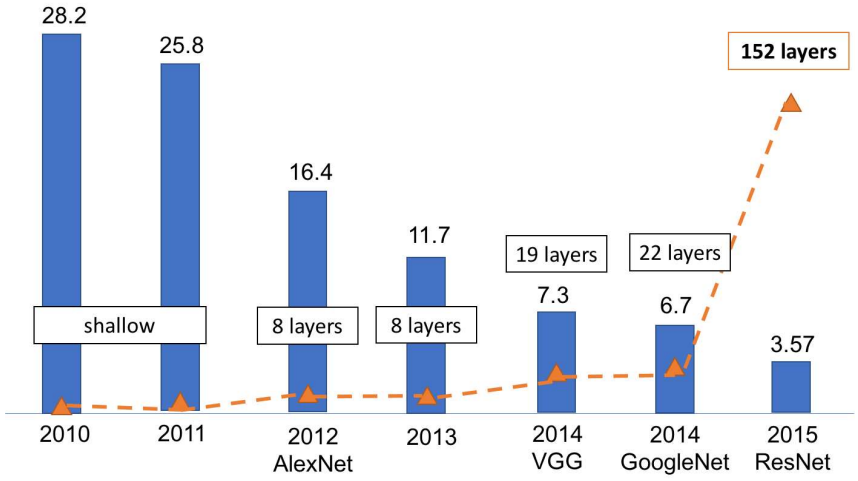


Lu C, Liu Q, Wang C, et al.

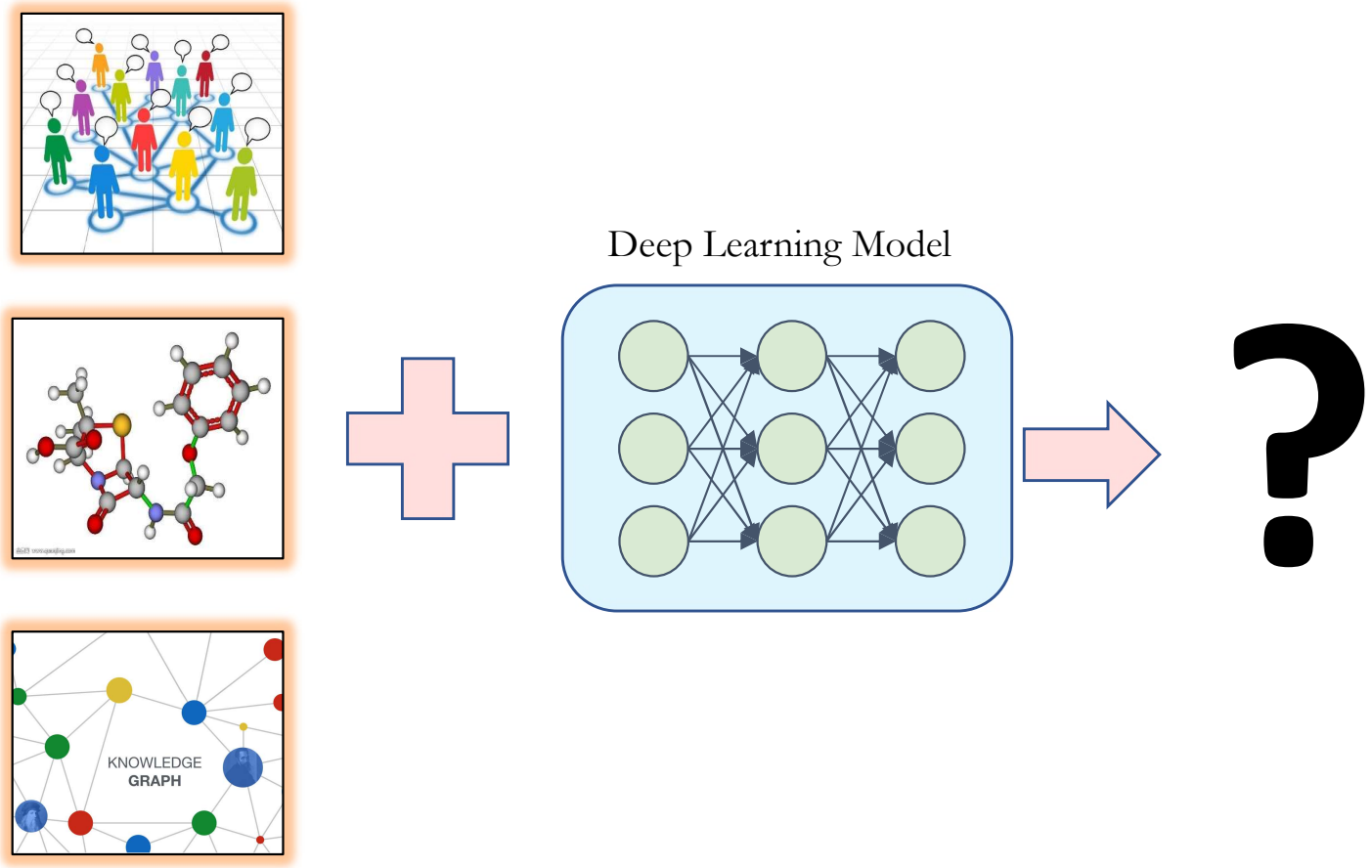
The Power of Deep Learning

IMAGENET

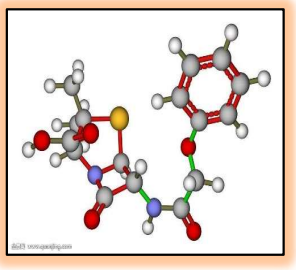


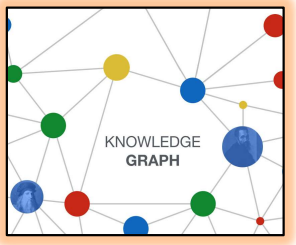


Deep Learning + Graphs = ?



Deep Graph Learning





Neural Network

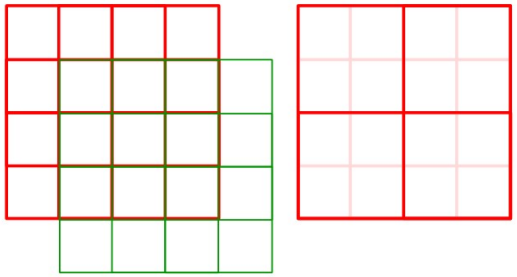
Graph/Node Representation

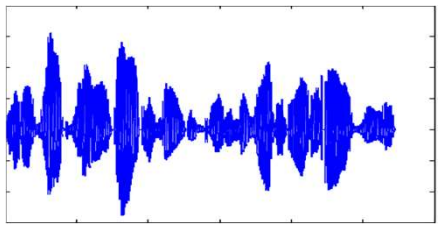
Applications

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

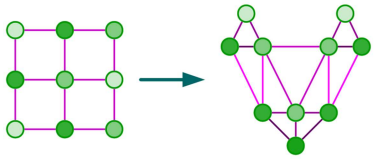
Make neural network model that can deal with graph data.

The Big Challenges

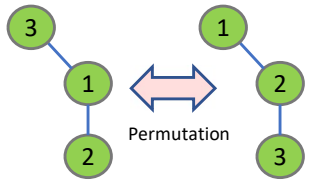




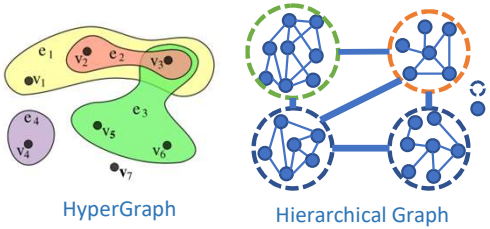
VS



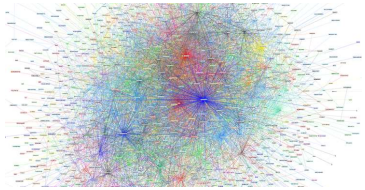
Irregular



Permutation Invariant



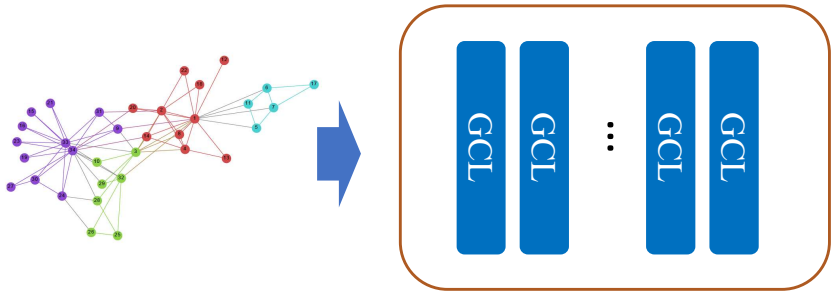
Complex variants



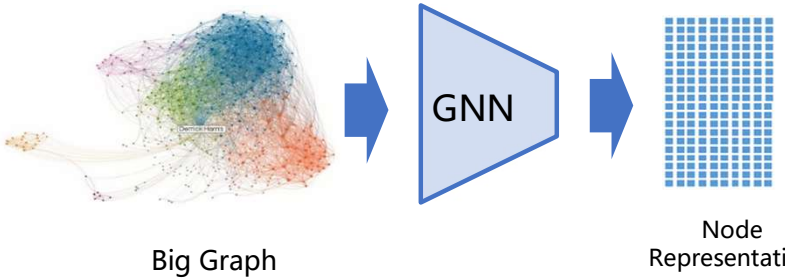
Large-scale instance

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

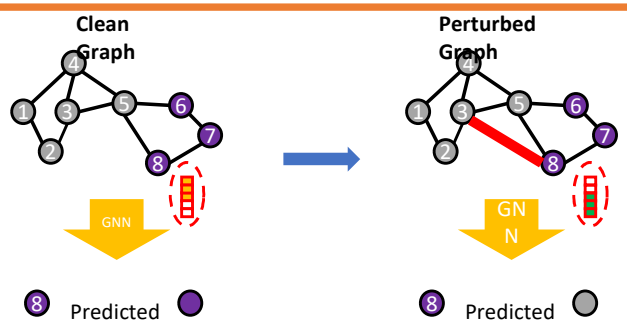
The Research Questions



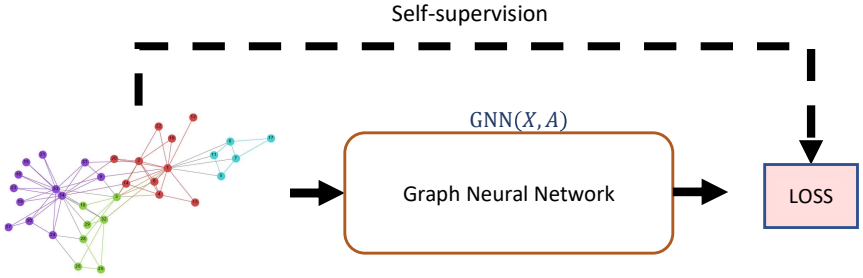
How to train the deep graph neural networks?



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



How to enhance the robustness of graph neural network ?



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

Overview

Foundations

- Preliminary
- The Brief History of Graph Neural Networks

Advances

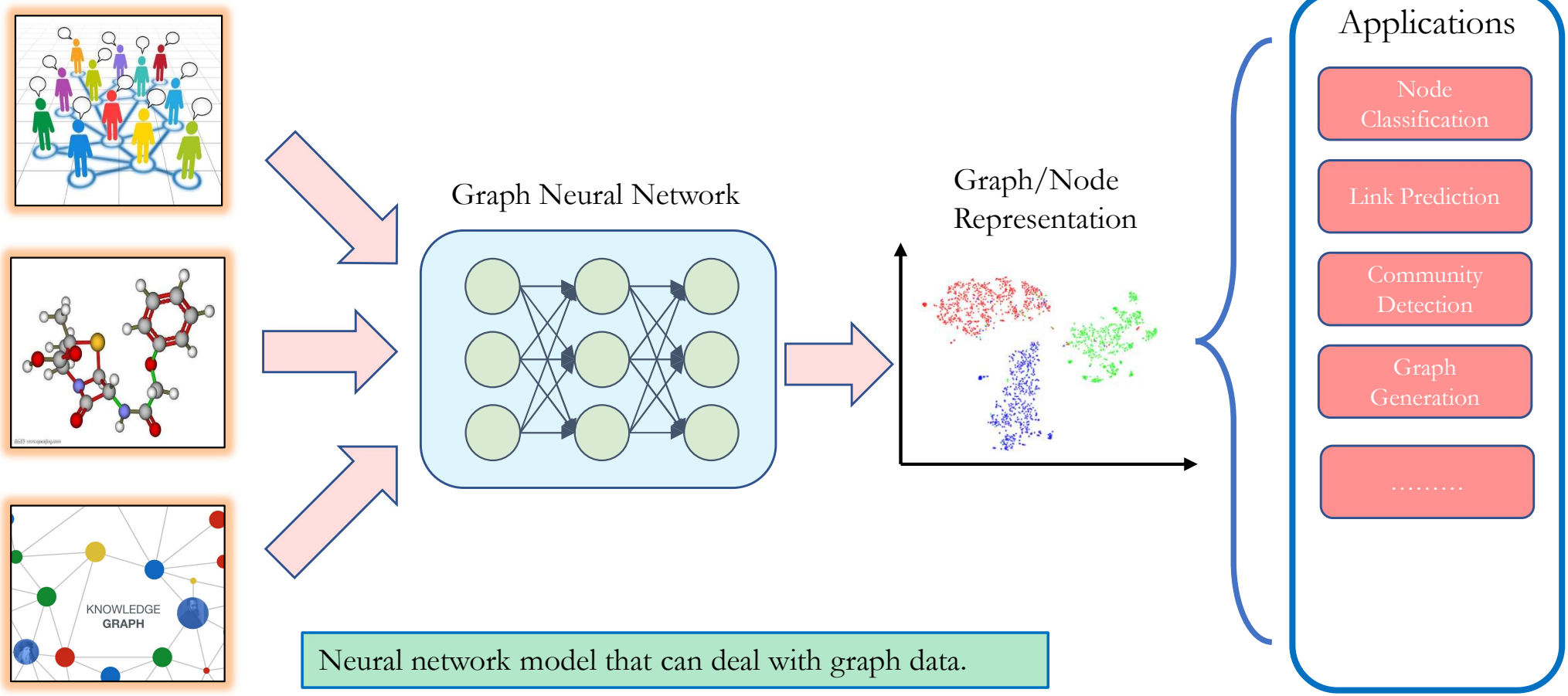
- Training Deep GNNs
- Scalability of GNNs
- Robustness of GNNs
- Self/Un-Supervised Learning of GNNs
- Other Advanced Topics

Applications

- Social Networks
- Medical Imaging

Preliminaries and Brief History of Graph Neural Networks

What is the Graph Neural Network?



Neural network model that can deal with graph data.

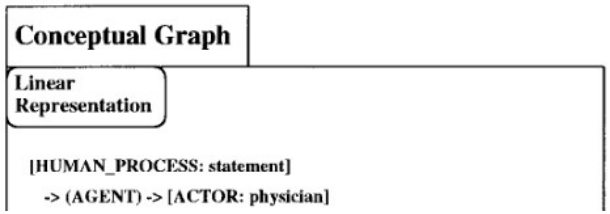
Graph Neural Network is not a New Thing

Sperduti, Alessandro and Starita, Antonina. 1997

Supervised Neural Networks for the Classification of Structures

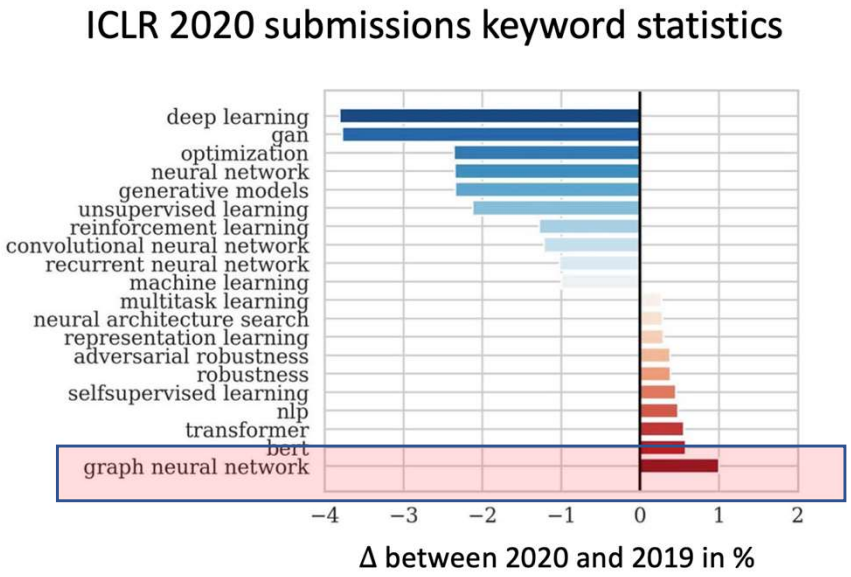
Alessandro Sperduti and Antonina Starita, *Member, IEEE*

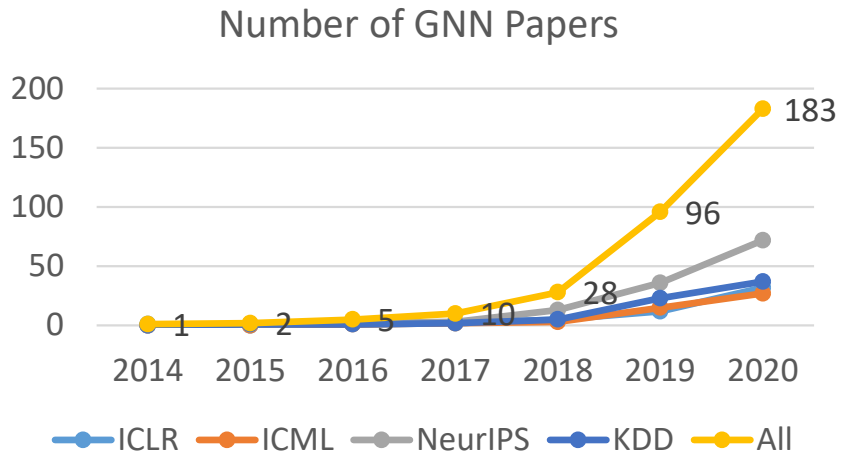
***Abstract*—Until now neural networks have been used for classifying unstructured patterns and sequences. However, standard neural networks and statistical methods are usually believed to be inadequate when dealing with complex structures because of their feature-based approach. In fact, feature-based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



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

A Rapidly Growing Area

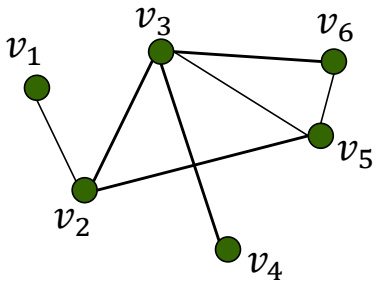




https://github.com/shaohua0116/ICLR2020-OpenReviewData

Preliminaries of Graph Learning

A topological graph



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

which has n nodes and m edges

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

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

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

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

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

Degree matrix

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

$-$

Adjacency matrix

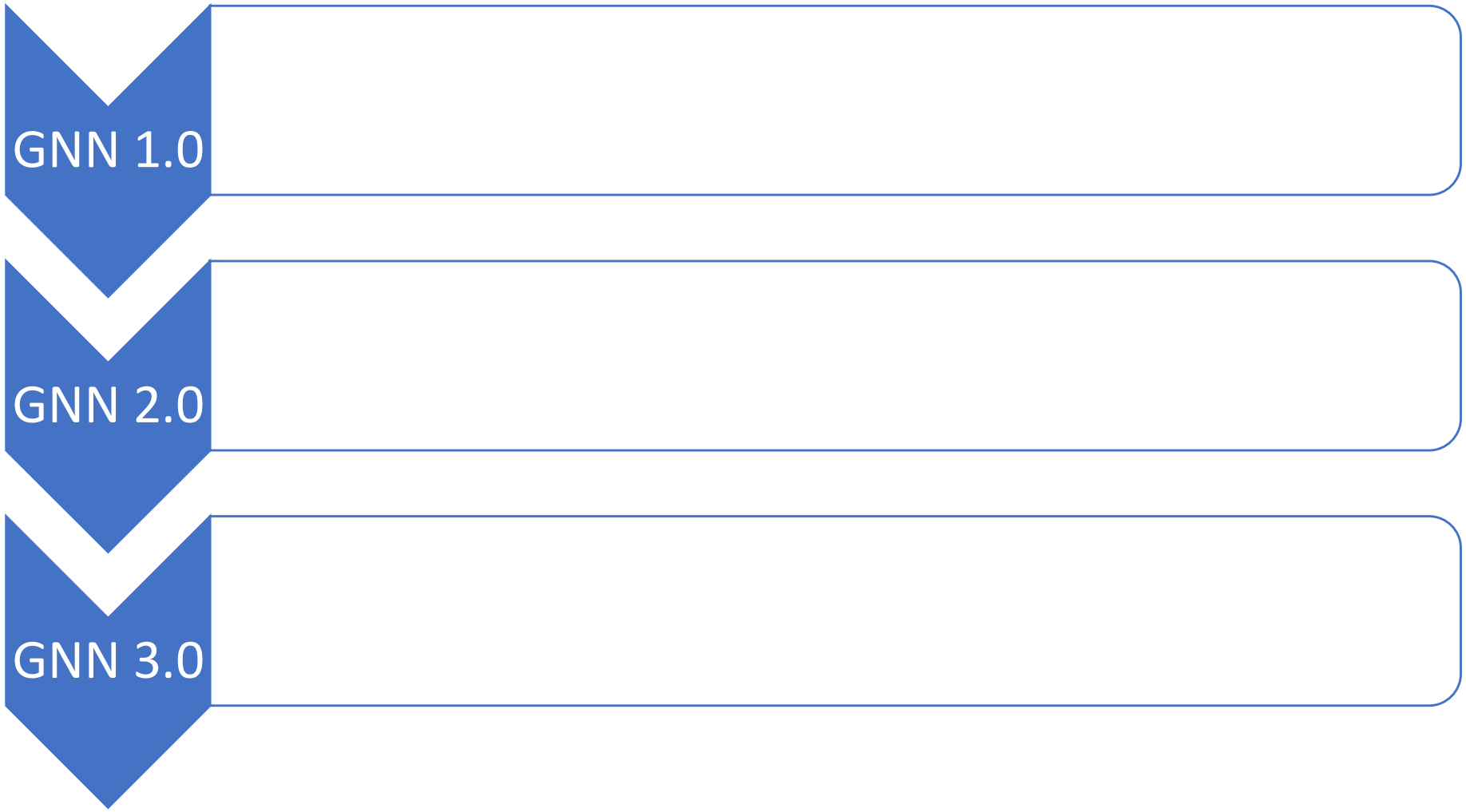
$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

$=$

Laplacian matrix

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

The Model of Graph Neural Networks



The Model of Graph Neural Networks

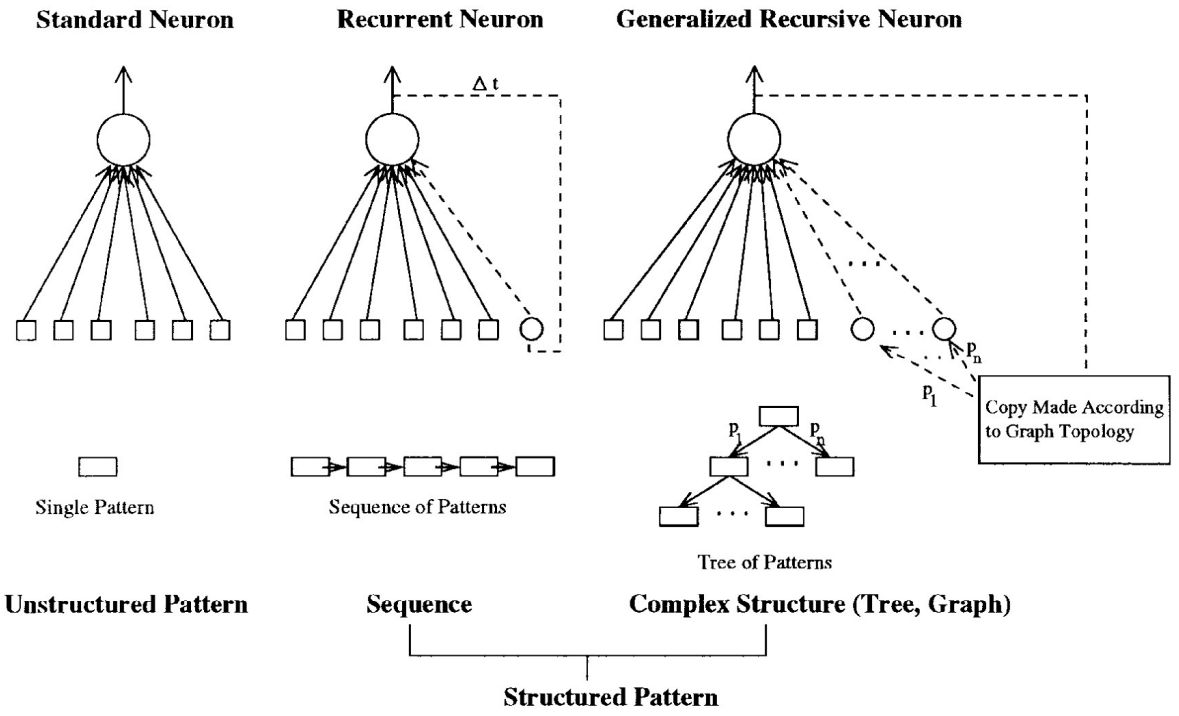
GNN 1.0

- Understanding GNN as RNN

GNN 2.0

GNN 3.0

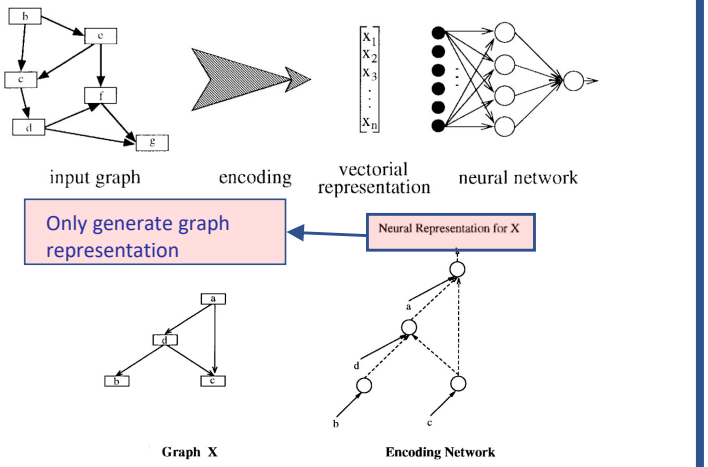
GNN 1.0: Understanding GNN as RNN



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

Sperduti, Alessandro, and Antonina Starita. 1997

GNN 1.0: Understanding GNN as RNN



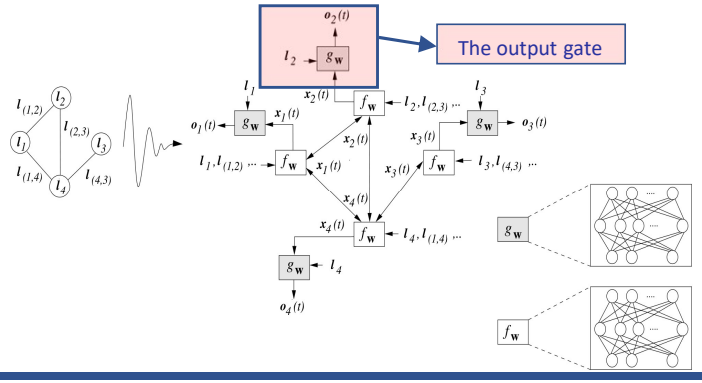
Before 2000

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.



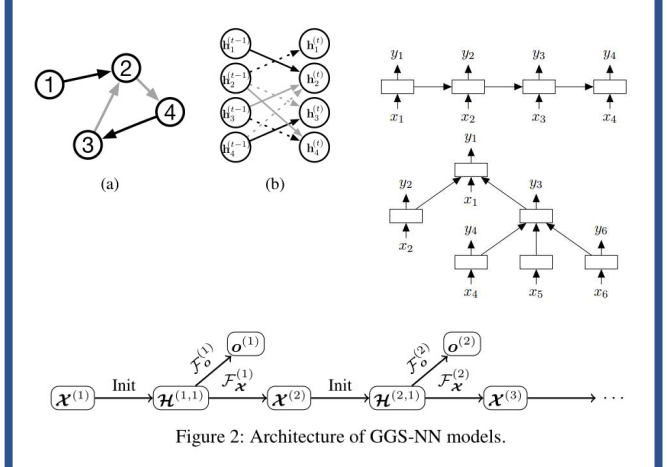


Figure 2: Architecture of GGS-NN models.

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

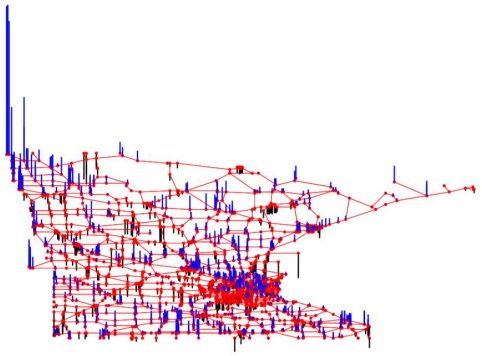
GNN 2.0

- Understanding GNN as Convolution

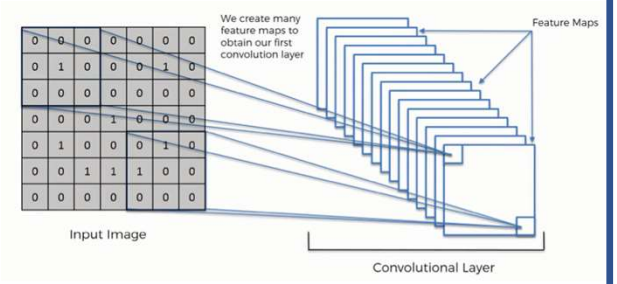
GNN 3.0

GNN 2.0: Understanding GNN as Convolution

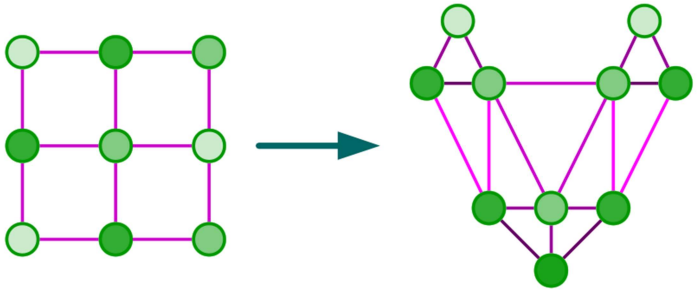
Graph Signal Processing



Convolutional Neural Networks

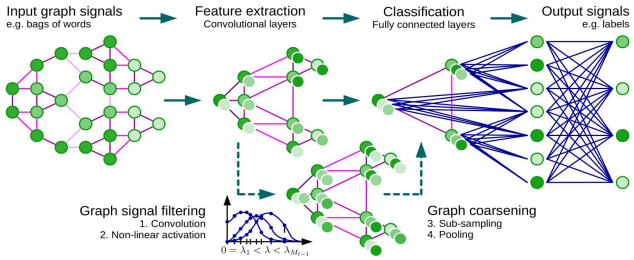


- How to perform the convolutions on graphs?
  - Irregular structures.
  - Weighted edges.
  - No orientation or ordering (in general).



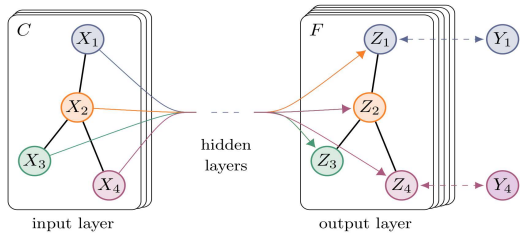
GNN 2.0: Understanding GNN as Convolution

ChebNet (NeurIPS 2016) [2]



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

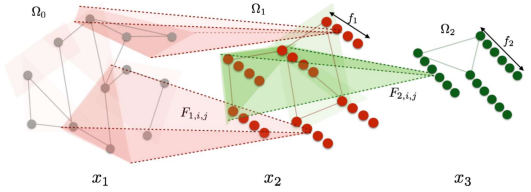
Graph Convolutional Network (ICLR 2017)



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

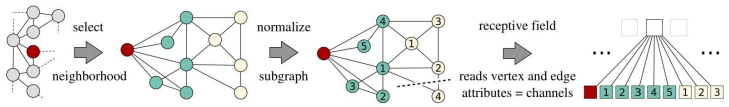
- Approximate 1-order Chebyshev polynomial the in spatial domain.
- Layer-wise convolution to extend receptive field.
- **The practical convolutional model for graphs.**

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)



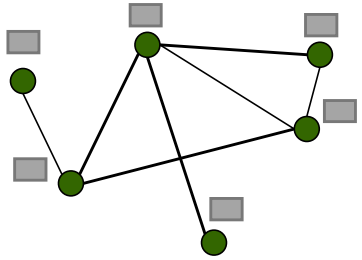
- Neighborhood sampling to construct receptive field.

[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

Graph Signal Processing

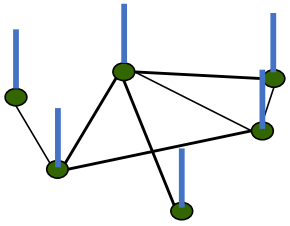
Graph signal:

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

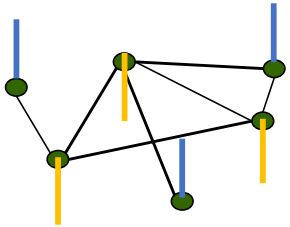


"**Frequency**" or "**Smoothness**" of the signal h

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



Low frequency graph signal



High frequency graph signal

Eigen decomposition of graph Laplacian

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

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

eigenvalues sorted non-decreasingly:

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

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

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

Graph Convolution: Spectral domain → Spatial domain

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

Spectral

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

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

ChebNet (NeurIPS 2016): parameterize with Chebshev polynomials:

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

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

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

Spatial

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

The Brief History of Graph Neural Networks

GNN 1.0

- Understanding GNN as RNN

GNN 2.0

- Understanding GNN as Convolution

GNN 3.0

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

GNN 3.0: Variants of Convolutions

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

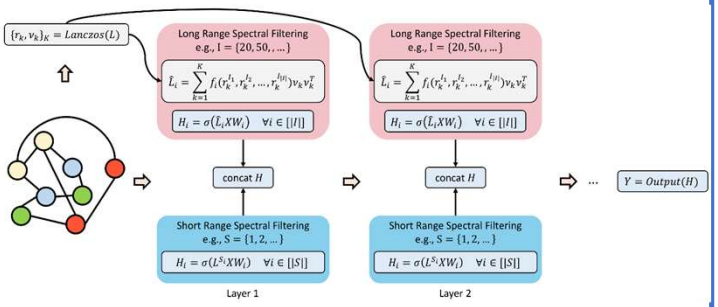
Lanczos Network [3]	Graph Wavelet Neural Network [1]	Hyperbolic GCN [2]

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

GNN 3.0: Variants of Convolutions

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

Lanczos Network [3]



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

Graph Wavelet Neural Network [1]

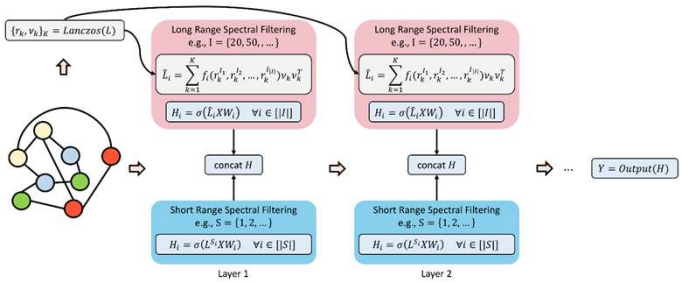
Hyperbolic GCN [2]

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

GNN 3.0: Variants of Convolutions

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

Lanczos Network [3]



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

Graph Wavelet Neural Network [1]

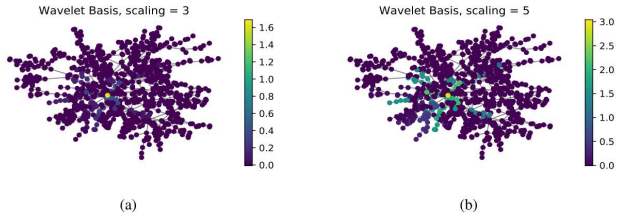


Figure 1: Wavelets on an example graph at (a) small scale and (b) large scale.

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

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

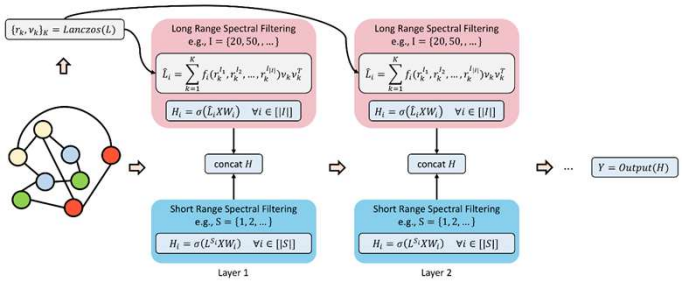
Hyperbolic GCN [2]

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

GNN 3.0: Variants of Convolutions

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

Lanczos Network [3]



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

Graph Wavelet Neural Network [1]

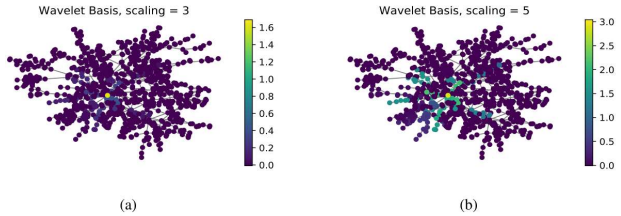
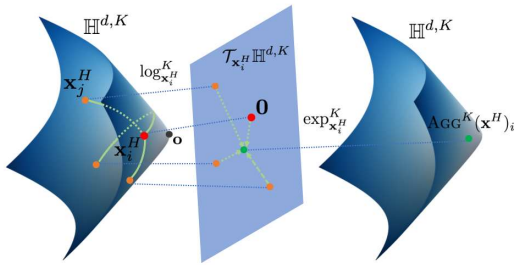


Figure 1: Wavelets on an example graph at (a) small scale and (b) large scale.

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

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

Hyperbolic GCN [2]



Construct the GCN in hyperbolic space.

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

- Hyperbolic feature transform.

$$\boldsymbol{h}_i^{(l+1),H} = (\boldsymbol{W}^{(l+1)} \otimes^{K_l} \boldsymbol{h}_i^{(l),H}) \oplus^{K_l} \boldsymbol{b}^{(l+1)}$$

- Attention-based hyperbolic aggregation.

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

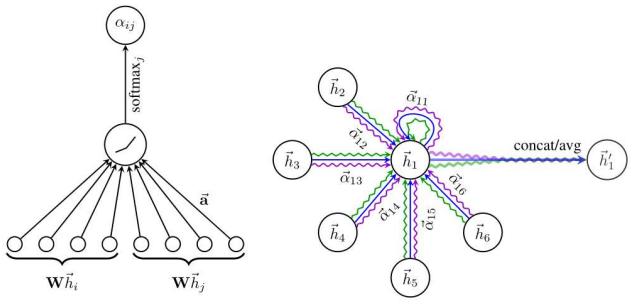
[1] Xu, Bingbing, et al. 2018 [2] Chami, Ines, et al. 2019 [3] Liao, Renjie, et al. 2019

GNN 3.0: GNN with Attention

$$h_i^{(l+1)} = \sigma(\sum_{j \in N(v_i)} S_{i,j} W^{(l)} h_j^{(l)}) \qquad S = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$$

Fixed during training

Graph Attention Network [1]



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

$$h_i^{(l+1)} = \sigma(\sum_{j \in N(v_i)} a_{ij} W^{(l)} h_j^{(l)})$$

$$a_{ij} = \exp(\frac{\sigma\left(\alpha^{\mathrm{T}}[Wh_i || Wh_j]\right)}{\sum_{k \in N(v_i)} \alpha^{\mathrm{T}}[Wh_i || Wh_k]})$$

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

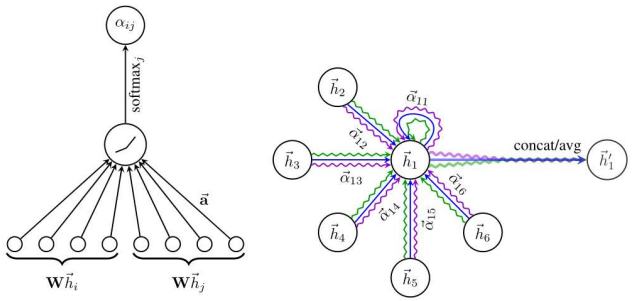
The original form:

$$h_i^{(l+1)} = \sigma(\sum_{j \in N(v_i)} S_{i,j} W^{(l)} h_j^{(l)})$$

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

Fixed during training

Graph Attention Network [1]

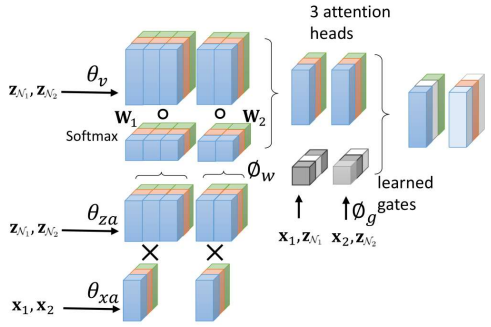


Replace the fixed aggregation weight a_{ij} to the learnable self-attention.

$$h_i^{(l+1)} = \sigma(\sum_{j \in N(v_i)} a_{ij} W^{(l)} h_j^{(l)})$$

$$a_{ij} = \exp(\frac{\sigma\left(\alpha^{\mathrm{T}}[Wh_i || Wh_j]\right)}{\sum_{k \in N(v_i)} \alpha^{\mathrm{T}}[Wh_i || Wh_k]})$$

Gated Attention Networks [2]



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

$$h_i^{(l+1)} = \sigma(\sum_{k=1}^{K} g_i^k \sum_{j \in N(v_i)} a_{ij} W^{(l)} h_j^{(l)})$$

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

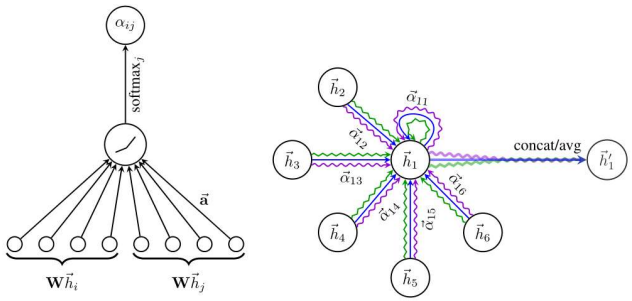
The original form:

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in N(v_i)} S_{i,j} W^{(l)} h_j^{(l)}\right)$$

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

Fixed during training

Graph Attention Network [1]

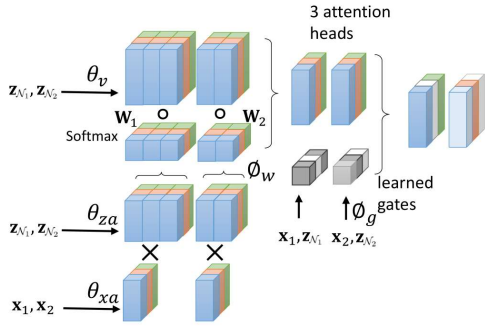


Replace the fixed aggregation weight a_{ij} to the learnable self-attention.

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in N(v_i)} a_{ij} W^{(l)} h_j^{(l)}\right)$$

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

Gated Attention Networks [2]

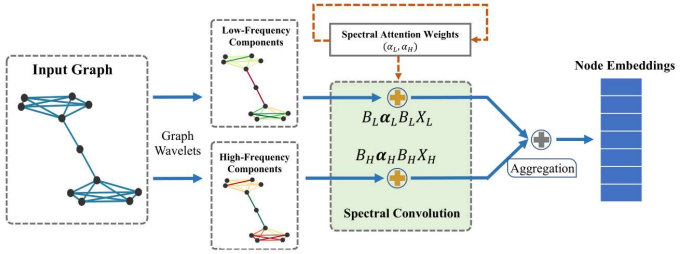


Add a learnable gate g_i^k to model the importance for each head.

$$h_i^{(l+1)} = \sigma\left(\sum_{k=1}^{K} g_i^k \sum_{j \in N(v_i)} a_{ij} W^{(l)} h_j^{(l)}\right)$$

K is the number of heads.

Spectral Graph Attention Network [3]



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

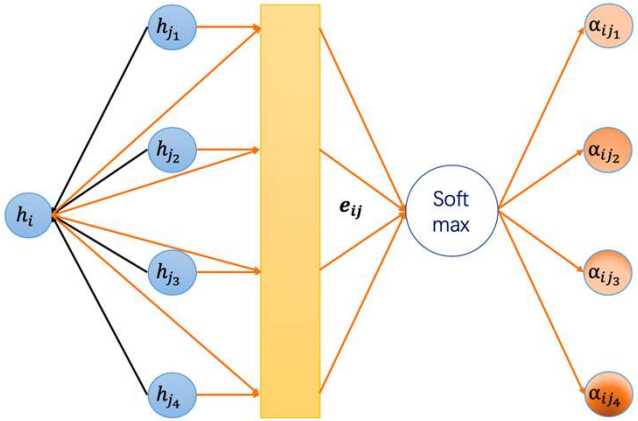
$$H^{(l+1)} = \sigma\left(\mathrm{AGG}\left(\boldsymbol{B_L} a_L \boldsymbol{B_L} \boldsymbol{H}^{(l)}, \boldsymbol{B_H} a_H \boldsymbol{B_H} \boldsymbol{H}^{(l)}\right) \boldsymbol{W}^{(l)}\right)$$

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

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

Graph Attention Network in Detail

Single head attention



$$h_i^{(l+1)} = \sigma(\sum_{j \in N(v_i)} \alpha_{ij} W \; h_j^{(l)})$$

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

$$e_{ij} = \text{LeakyReLU}(a^T \; (Wh_i \,||Wh_j)$$

Multi-head attention

Enrich the model capacity and stabilize the learning process

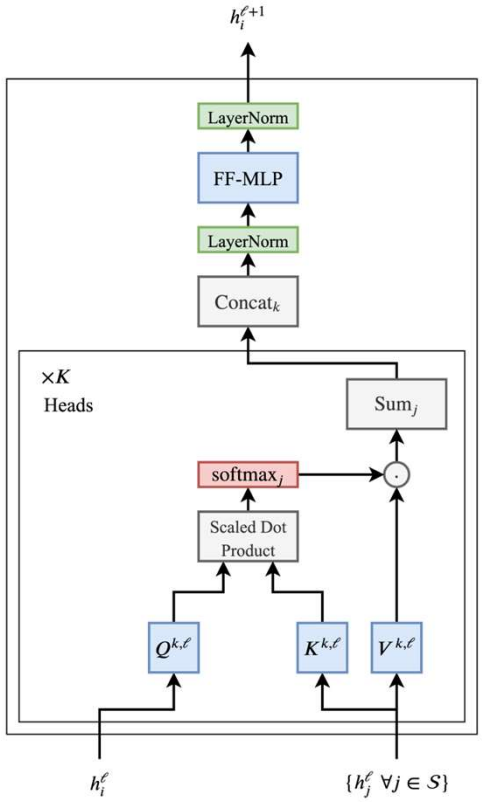
Each head has its own parameters
and their outputs can be merged in two ways:

- ❑ Concatenation
- ❑ Average

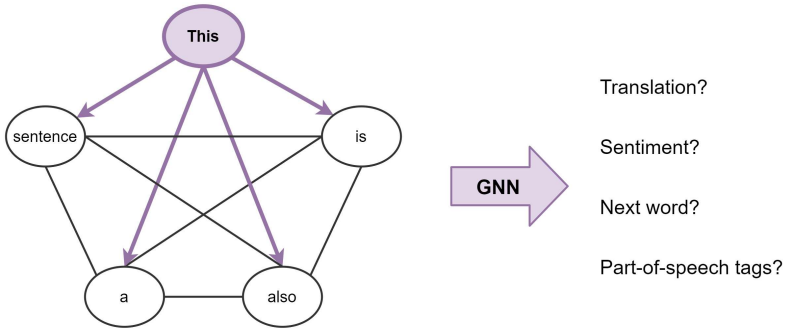
Attention weights learnt for the Cora dataset

Transformers as GNNs with Multi-head Attention

One layer of the multi-head QKV attention



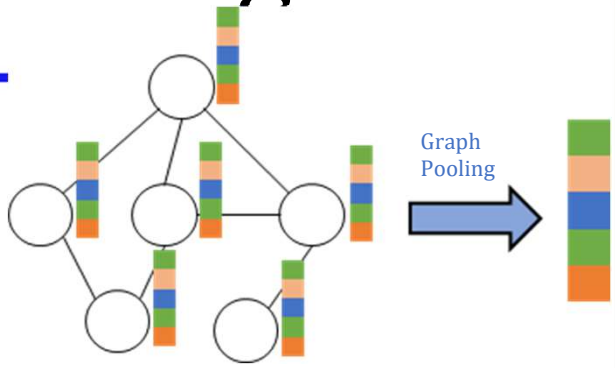
Transformer takes input sequence as a complete graph.



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

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

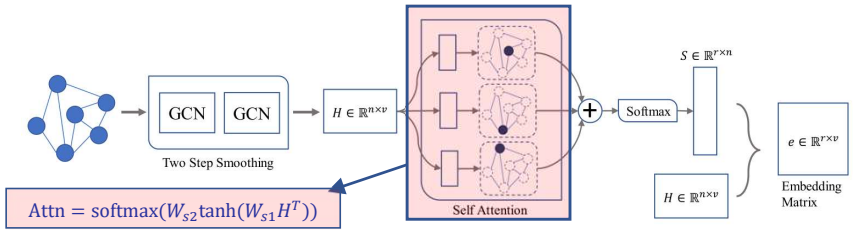
GNN 3.0: GNN with Graph Pooling



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

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

Self-Attentive Graph Embedding (SAGE)

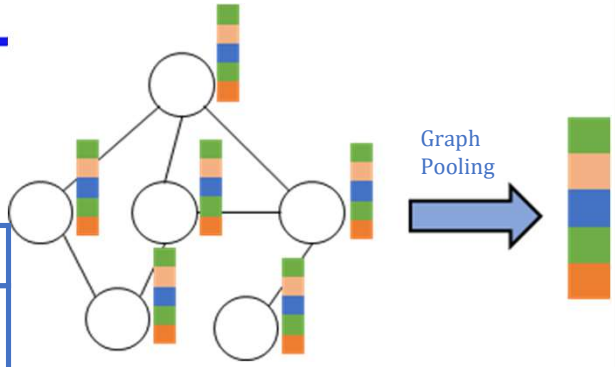


Introduce the self-attention mechanism to model the node importance during the pooling.

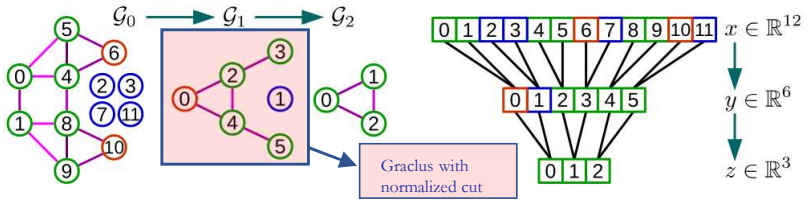
Li, Jia, et al. "Semi-supervised graph classification: A hierarchical graph perspective."

GNN 3.0: GNN with Graph Pooling

Hierarchical Pooling

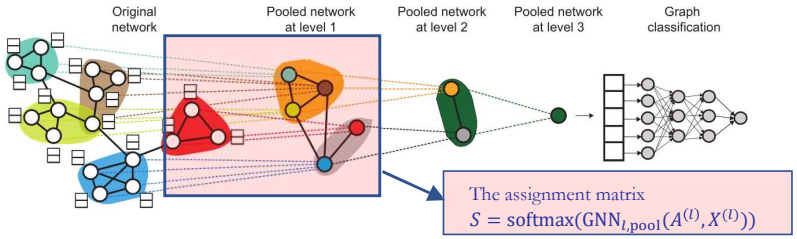


Graph Coarsening by Graph Cut [1]



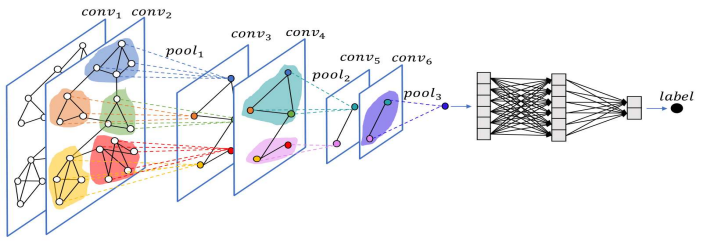
Graph Pooling with pre-defined subgraph by graph cut algorithm.

Differentiable Graph Pooling (DIFFPOOL)[2]



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

EigenPooling [3]



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

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

GNN Implementation: Message Passing Framework

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

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

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

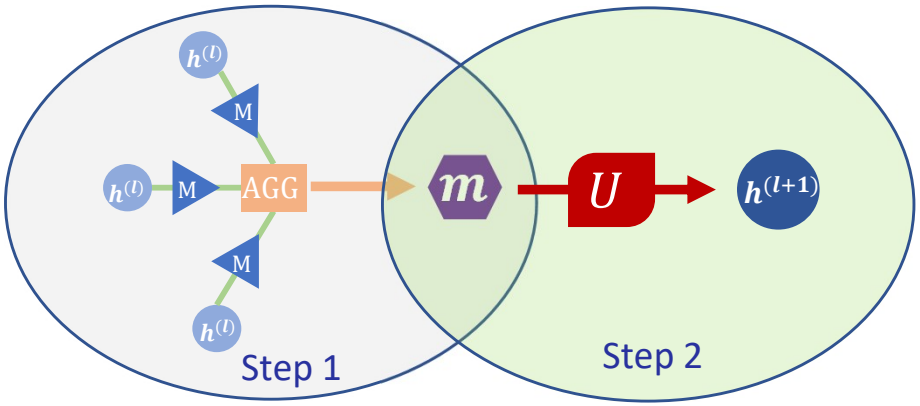
The neighborhood set of node. E.g. 1-hop neighbors.

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

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

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

The state update function.

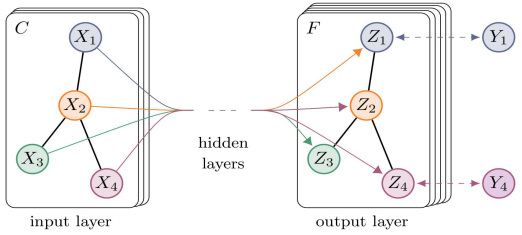


Gilmer, Justin, et al. "Neural Message Passing for Quantum Chemistry." ICML. 2017.

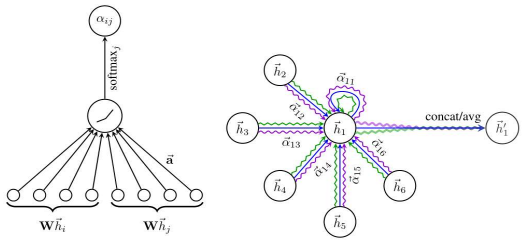
Examples of Message Passing Realizations

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

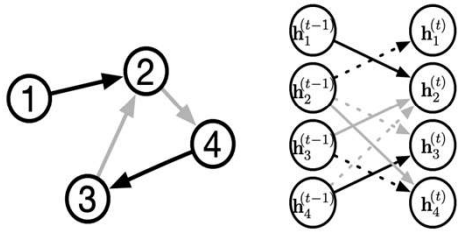
GCN (ICLR 2017)



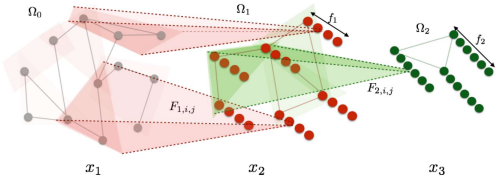
Graph Attention Network (ICLR 2018)



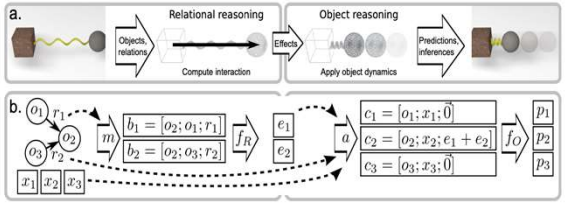
Gated GNNs (ICLR 2016)



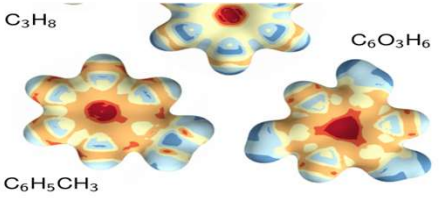
Deep Locally Connected Networks (ICLR 2014)



Interaction Networks (NeurIPS 2016)



Deep Tensor Neural Networks (Nature Communications 2017)

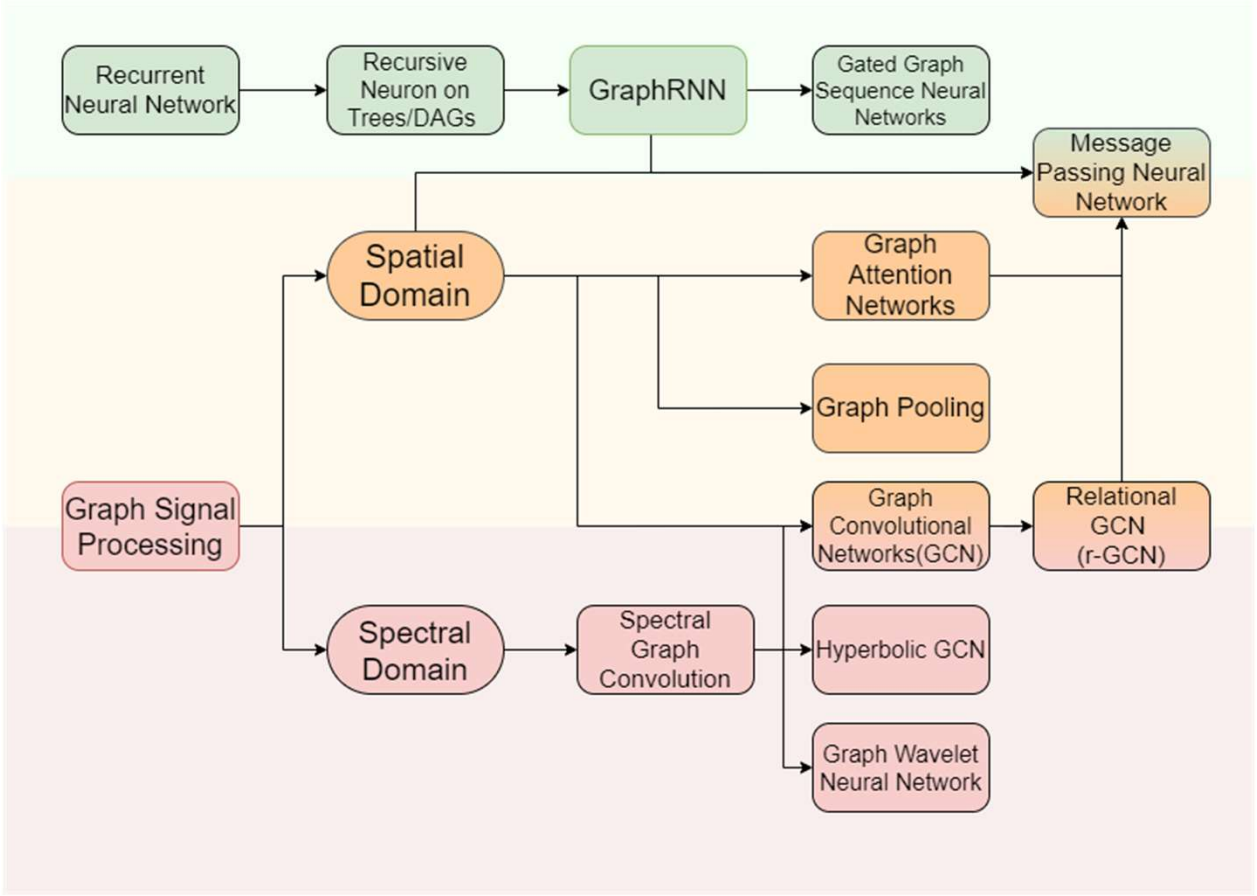


Wang, Minjie, et al. "Deep graph library: A graph-centric, highly-performant package for graph neural networks." *arXiv preprint arXiv:1909.01315* (2019).

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

Summary

@change advanced topics



Advanced topics

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

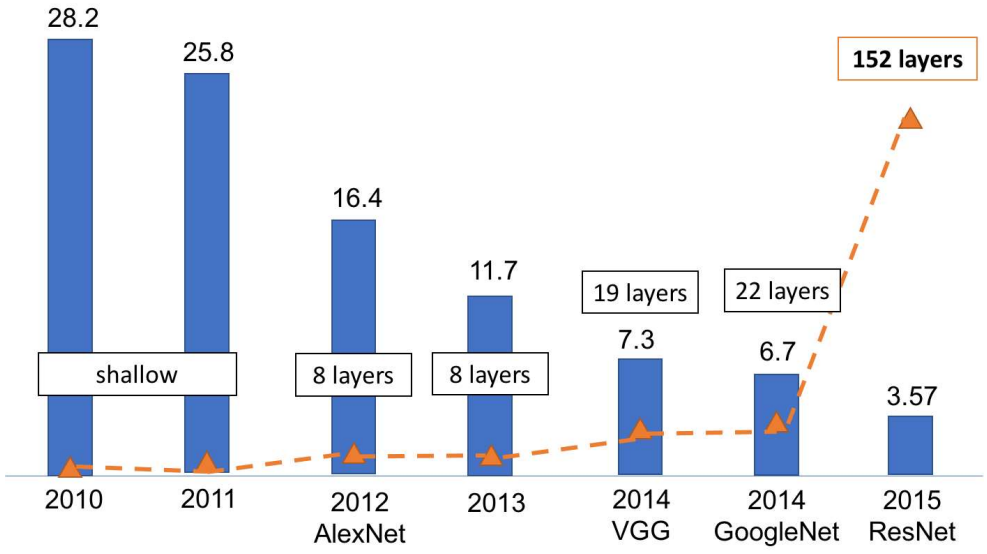
Training Deep GNNs

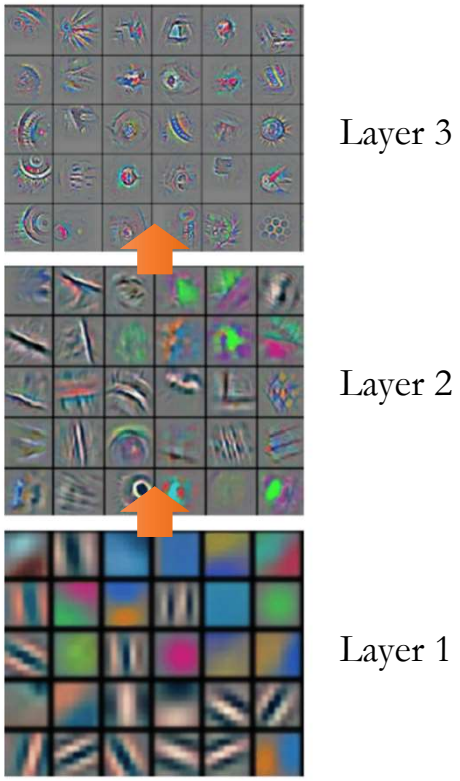
Training Deep GNNs

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

The Power of Deep DNNs

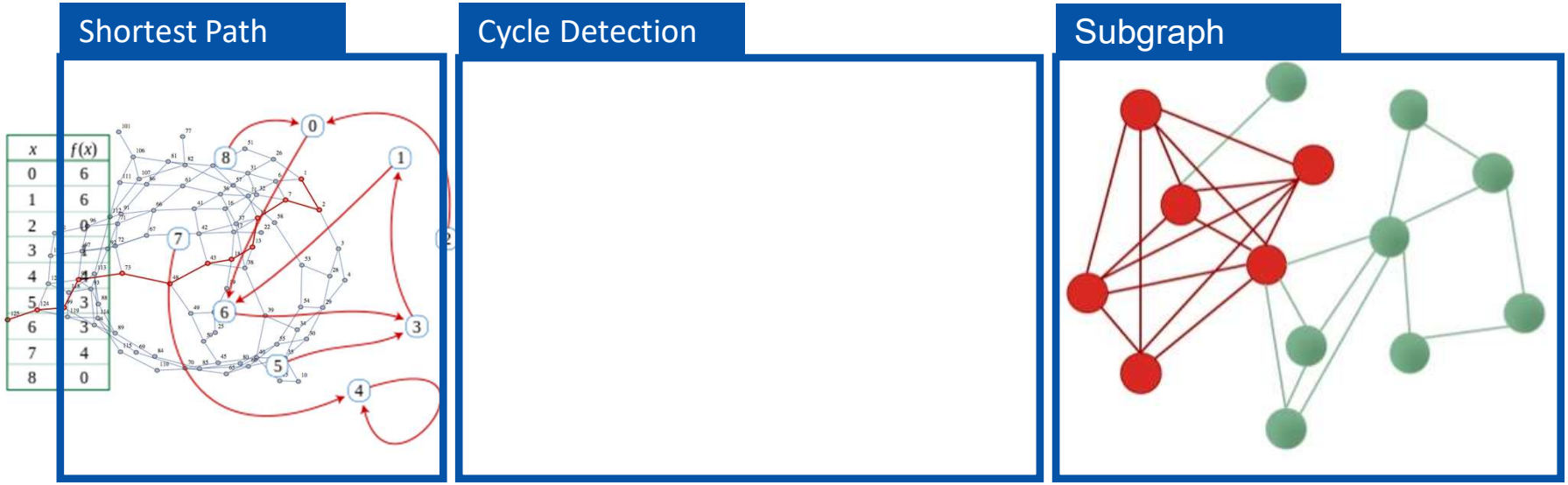
- Unprecedented success of deep DNNs in computer vision
- Deep DNNs enable larger receptive fields
- Deep DNNs enable more expressivity





The Power of Deep GNNs

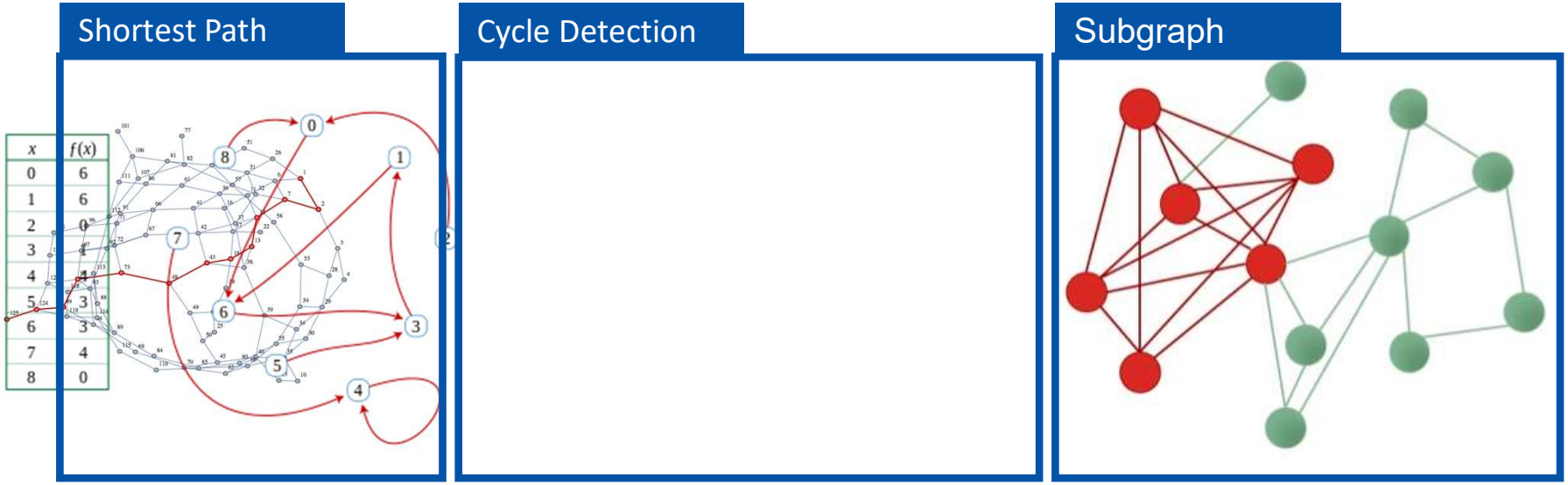
- Do GNNs need deep structures to enable larger receptive fields, too?
- What limits the expressive power of GNNs?
  - The depth d
  - The width w
- GNNs significantly lose their power when *capacity*, dw, is restricted



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

The Power of Deep GNNs

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



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

The Power of Deep GNNs

- The boundary of capacity for different problems

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

(Loukas, ICLR'20)

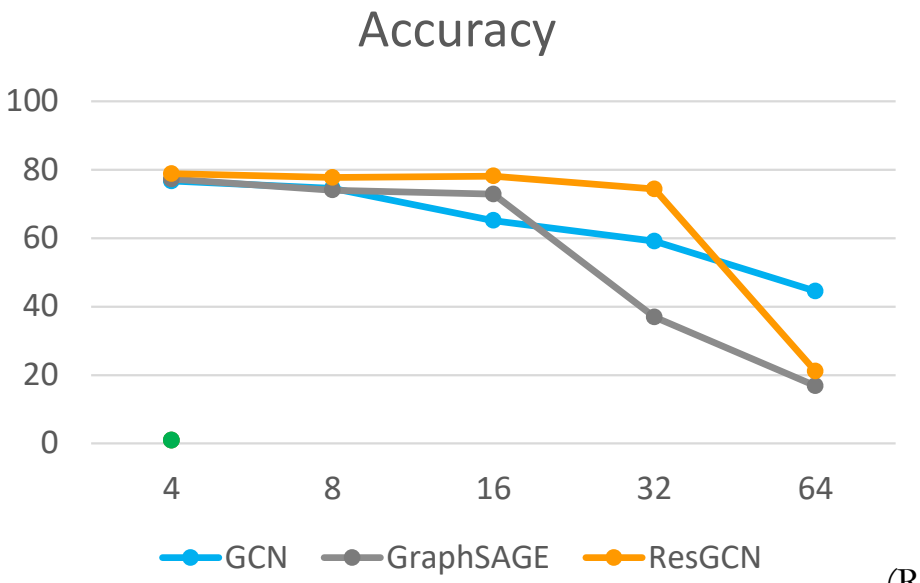
Loukas, Andreas. "What graph neural networks cannot learn: depth vs width." *International Conference on Learning Representations. 2020.*

Training Deep GNNs

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

GNNs are Shallow

- But can they really go deep? Not all



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

(Rong et al, ICLR'20)

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

Training deep GNNs

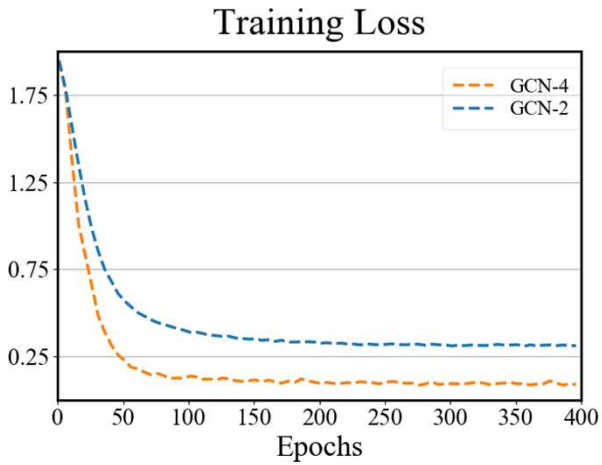
- Why do we need deep GNNs?
- Can GNNs simply go deep?
- What impedes GNNs to go deep?
  - Overfitting (Common)
  - Training dynamics (Common)
  - Over-smoothing (Graph Specific)
- How to make GNNs deep?

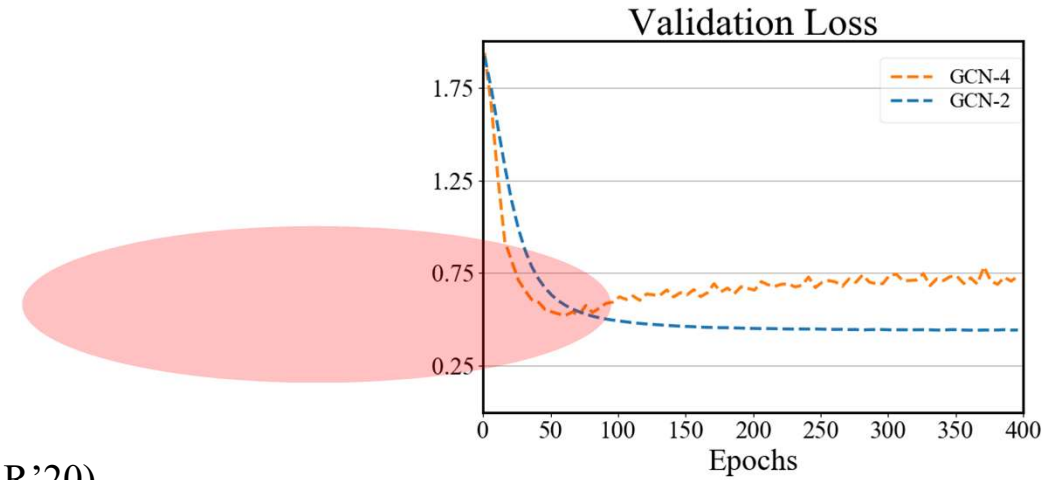
Training deep GNNs

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

Overfitting

- GNNs suffer from Overfitting





(Rong et al, ICLR'20)

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

$$O(dh^2)$$

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

Training deep GNNs

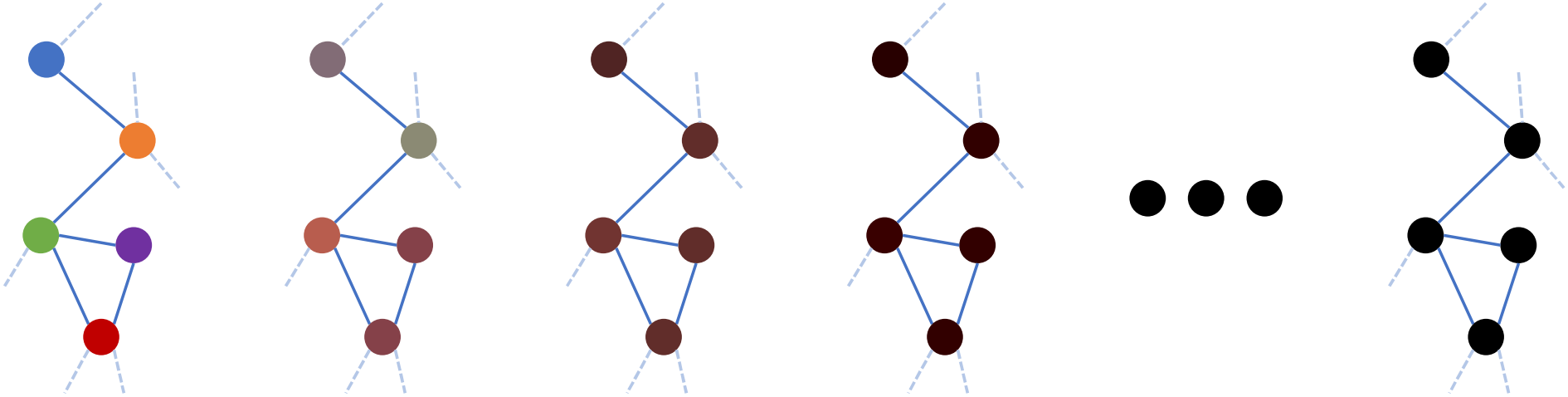
- Why do we need deep GNNs?
- Can GNNs simply go deep?
- What impedes GNNs to go deep?
  - Overfitting (Common)
  - **Training dynamics (Common)**
  - Over-smoothing (Graph Specific)
- How to make GNNs deep?

Training dynamics

l-layers gradient

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

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



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

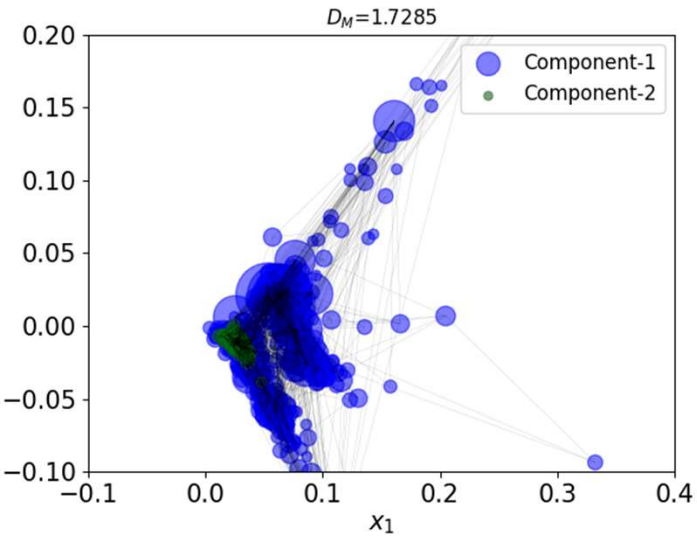
Training deep GNNs

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

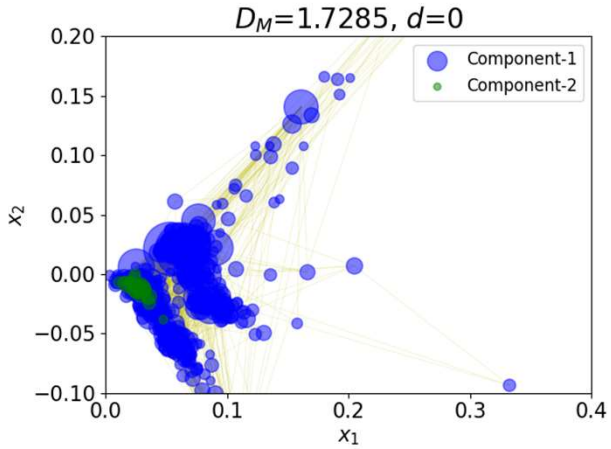
Over-Smoothing

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

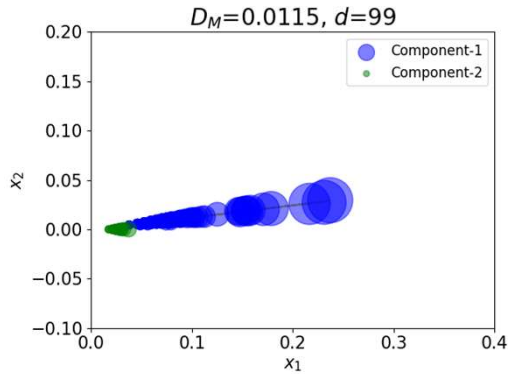
    (Li et al. AAAI'18; Chen et al. AAAI'20; Oono et al. ICLR'20; Rong et al. ICLR'20; Huang et al. arXiv'20)



(Huang et al. arXiv'20)



Initial

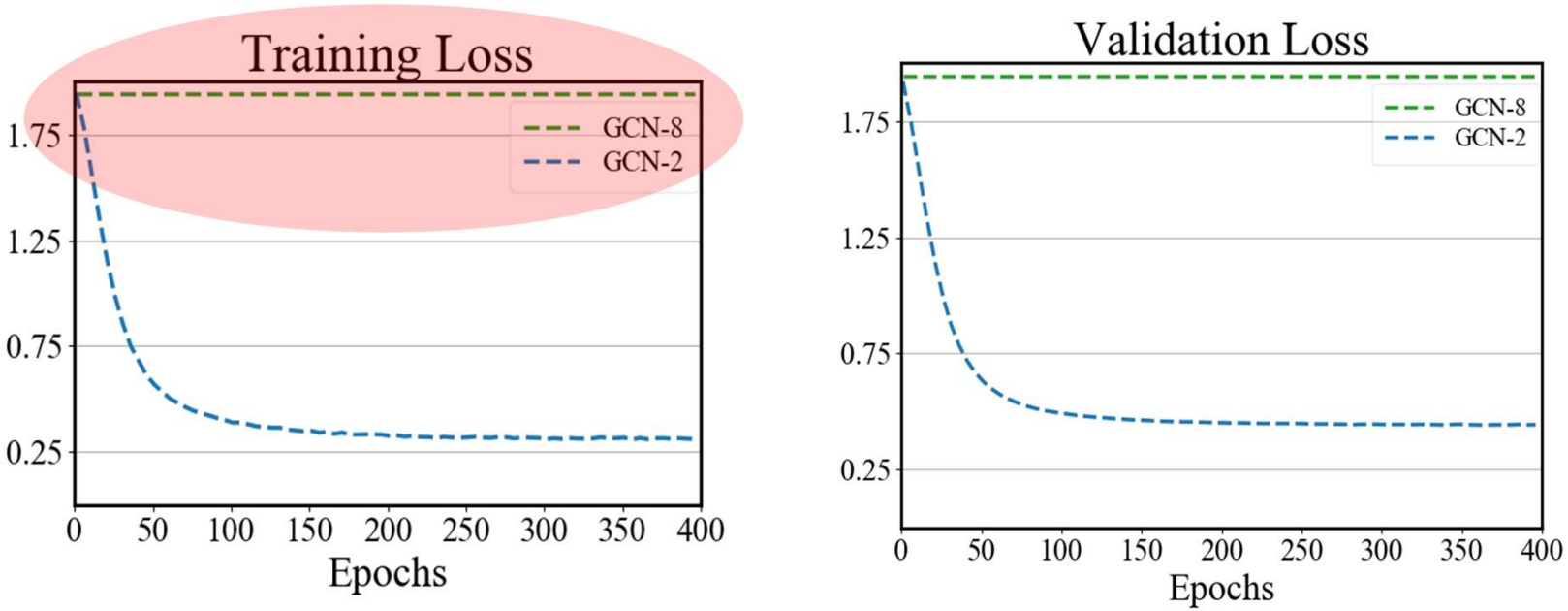


99 layers

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

Over-Smoothing

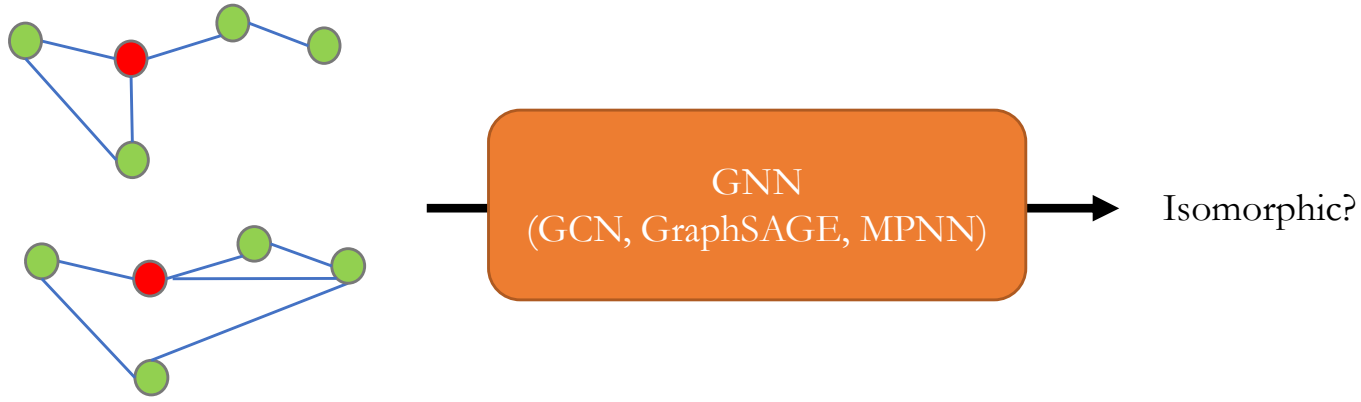
- Over-smoothing also impedes training.



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

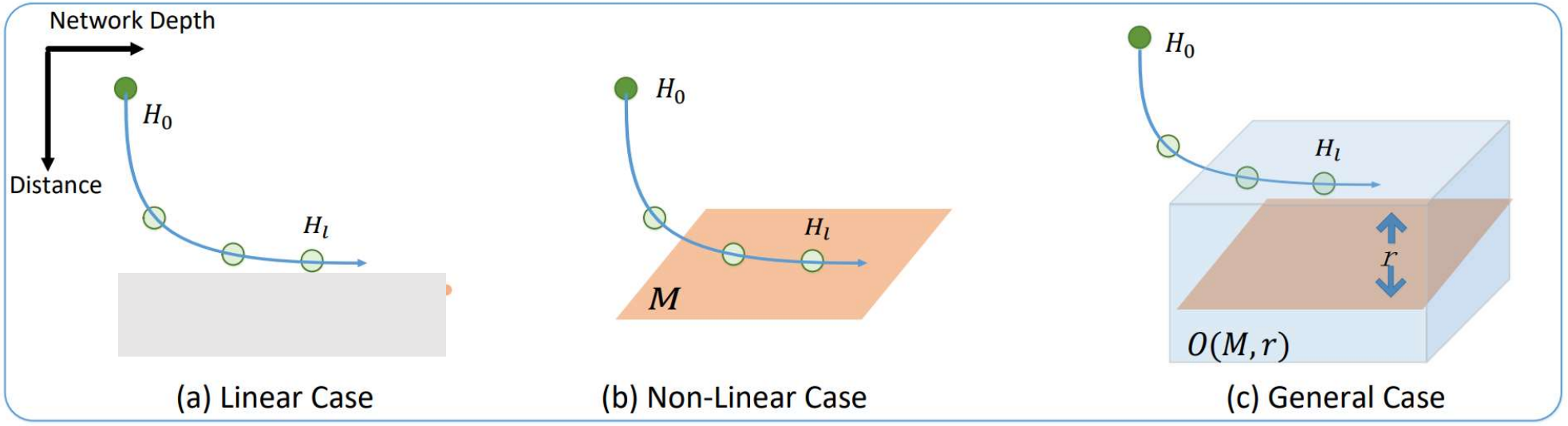
Over-Smoothing

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



Over-Smoothing

- When GCNs fail?
  - With linear activation
  - With ReLU activation
  - *W*ith ReLU and bias



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

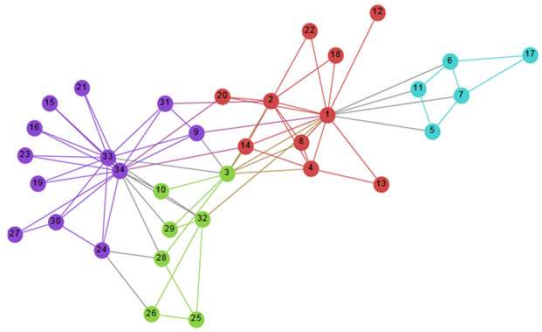
Over-Smoothing of Linear GCN

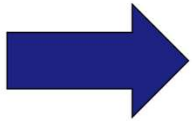
- When GCNs fail?
  - With linear activation

l-step Random Walk

Probability of walking

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





Random Walks on Graph

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

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

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

Over-Smoothing of Linear GCN

- When GCNs fail?
  - With linear activation

l-step Random Walk

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

l-layer GCNs

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

Learnable Probability

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

Over-Smoothing of Linear GCN

- When GCNs fail?
  - With linear activation

l-step Random Walk

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

l-layer GCNs

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

Eigen decomposition

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

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

Over-Smoothing of Linear GCN

- Rewrite eigen decomposition

Eigen decomposition

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

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

Over-Smoothing of Linear GCN

- Rewrite eigen decomposition

Eigen decomposition

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

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

Over-Smoothing of Linear GCN

- Rewrite eigen decomposition

Eigen decomposition

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

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

Eigenvalues

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

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

Convergence

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

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

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

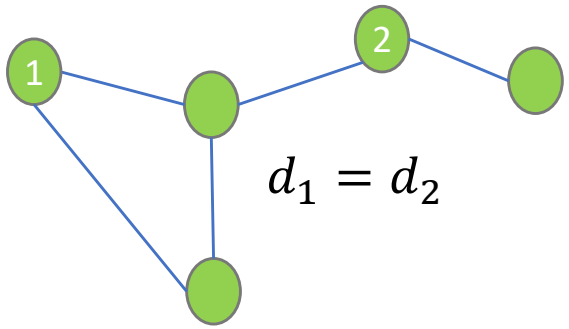
Over-Smoothing of Linear GCN

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

Convergence

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

Node 1 is **indistinguishable** with node 2



degree

Y

Node 1	$d_1^{\frac{1}{2}} z_1$	$d_1^{\frac{1}{2}} z_2$	$d_1^{\frac{1}{2}} z_3$	…
Node 2	$d_2^{\frac{1}{2}} z_1$	$d_2^{\frac{1}{2}} z_2$	$d_2^{\frac{1}{2}} z_3$	

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

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

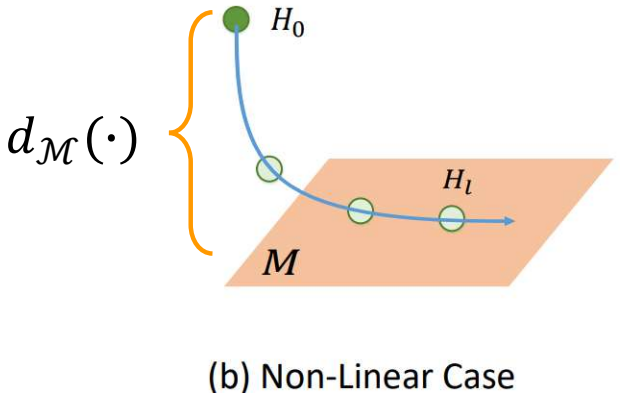
Over-Smoothing of Non-Linear GCN

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

$\mathcal{M}$ subspcae

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

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



(b) Non-Linear Case

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

Over-Smoothing of Non-Linear GCN

Convergence of $\tilde{A}$

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

Convergence

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

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

Over-Smoothing of Non-Linear GCN

Convergence of $\tilde{A}$

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

Convergence of W

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

Convergence

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

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

Over-Smoothing of Non-Linear GCN

Convergence of $\tilde{A}$

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

Convergence of W

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

Convergence of ReLU

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

Convergence

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

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

Over-Smoothing of GCNs with bias

- With ReLU and bias

GCNs with bias

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

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

Over-Smoothing of GCNs with bias

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

GCNs with bias

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

Convergence of bias

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

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

Over-Smoothing of GCNs with bias

- With ReLU and bias

GCNs with bias

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

Convergence of bias

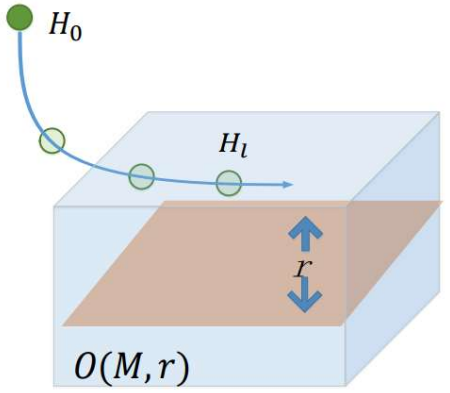
$$d_{\mathcal{M}}(H_{l+1}) \leq \lambda_{m+1} s d_{\mathcal{M}}(H_l) + d_{\mathcal{M}}(b)$$

GCN with bias

H_l converges to a certain sub-cube:

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

(Huang et al. arXiv'20)



(c) General Case

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

Summary

Linear GCN

H_l converges to a certain point that can be exactly derived

Non-linear GCN

H_l converges to a certain point within a certain subspace

GCN with bias

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

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

Other methods to measure over-smoothing

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

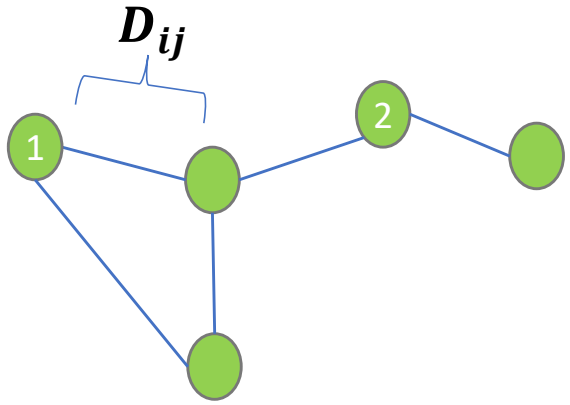
Pair Distance

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

The i-th row of the output

Mean Average Distance

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



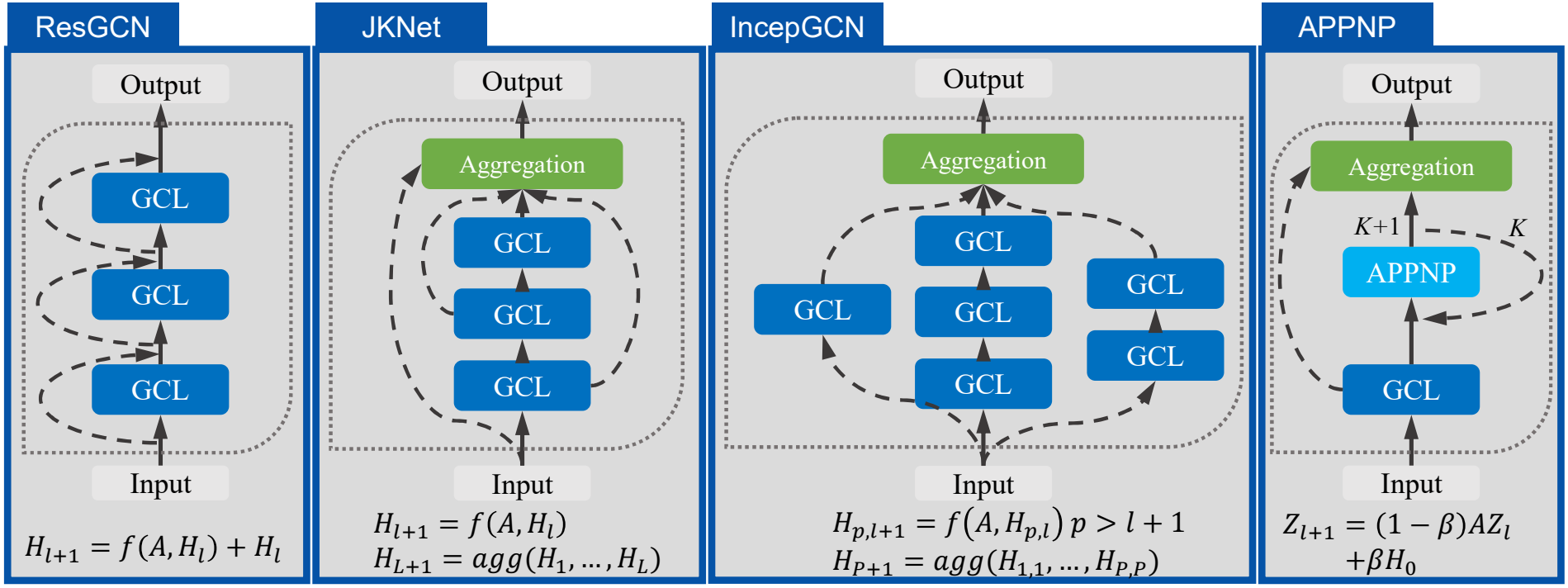
- Other metrics see PairNorm (Zhao et al., 2020); GroupNorm (Zhou et al., 2020)

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

Training deep GNNs

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

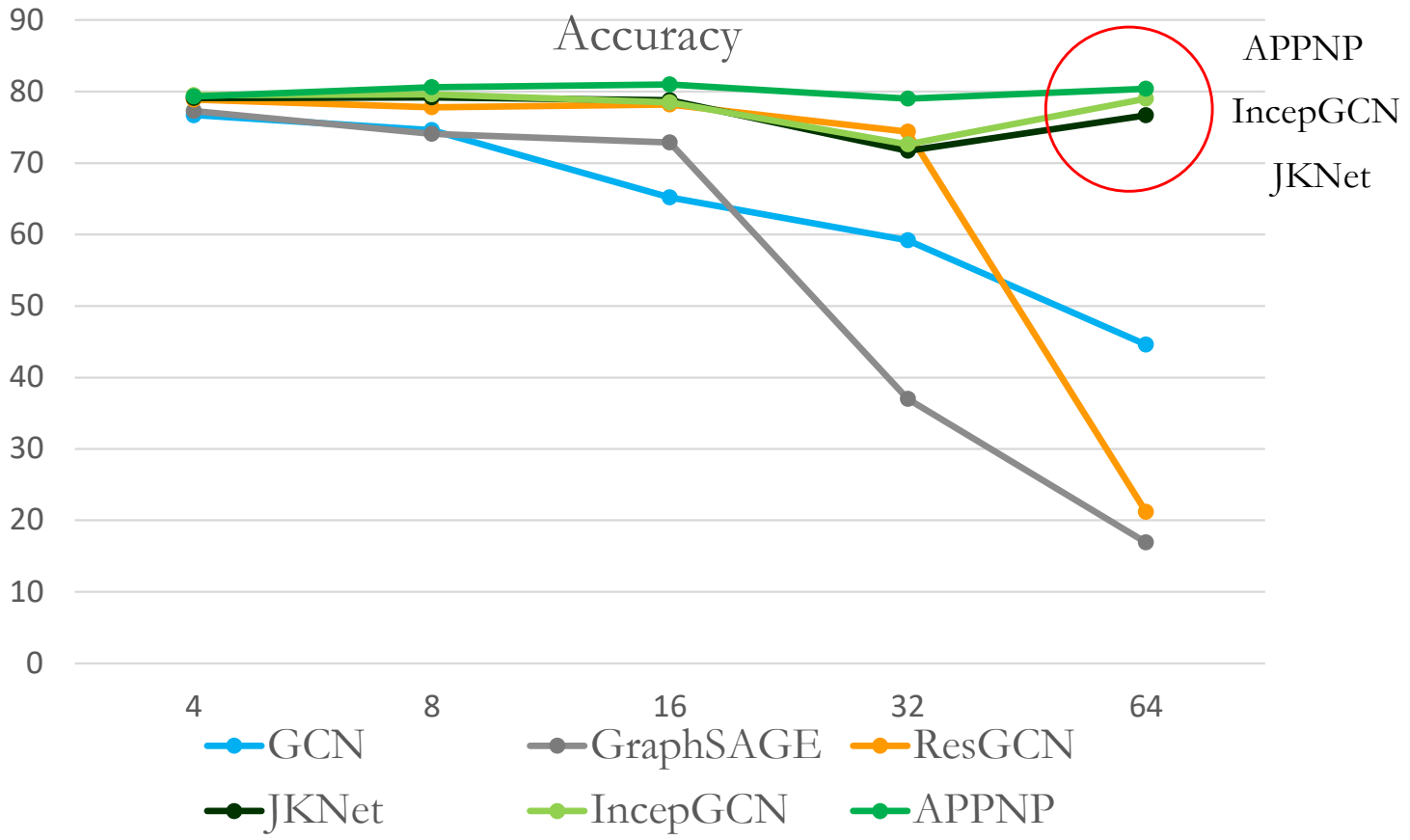
Shortcuts in Structures



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

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

Shortcuts in Structures



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

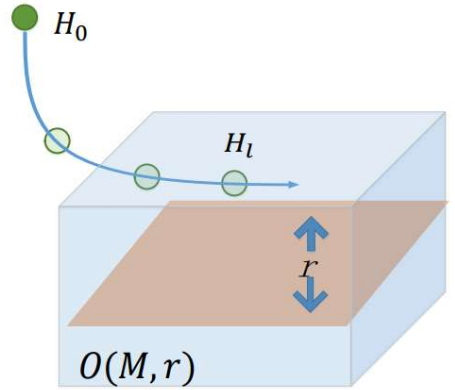
Shortcuts in Structures

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

GCN with bias

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

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



(c) General Case

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

Shortcuts in Structures

General Case

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

Basic GCN

Generic GCN

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

GCN with bias

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

Different Structures

ResGCN

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

APPNP

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

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

Shortcuts in Structures

General Case

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

Basic GCN

Generic GCN

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

GCN with bias

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

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

Different Structures

ResGCN

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

APPNP

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

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

Shortcuts in Structures

General Case

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

Basic GCN

Generic GCN

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

GCN with bias

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

Different Structures

ResGCN

v is increased

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

APPNP

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

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

Shortcuts in Structures

General Case

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

Basic GCN

Generic GCN

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

GCN with bias

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

Different Structures

ResGCN

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

APPNP

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

v is decreased

r is nonzero

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

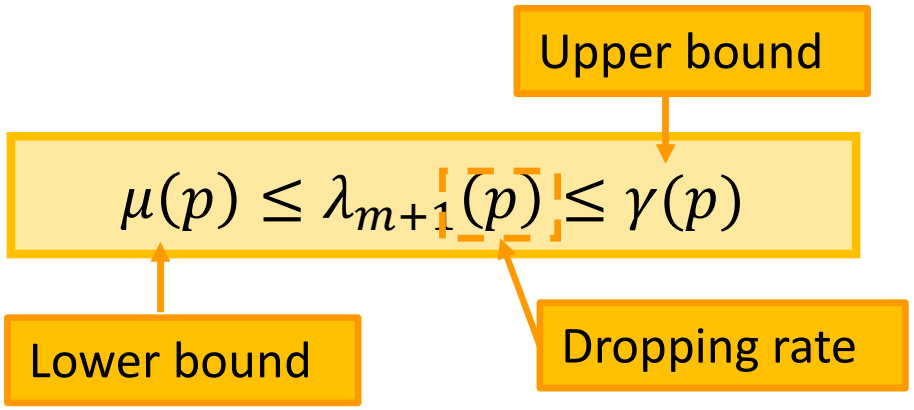
Over-smoothing Layer

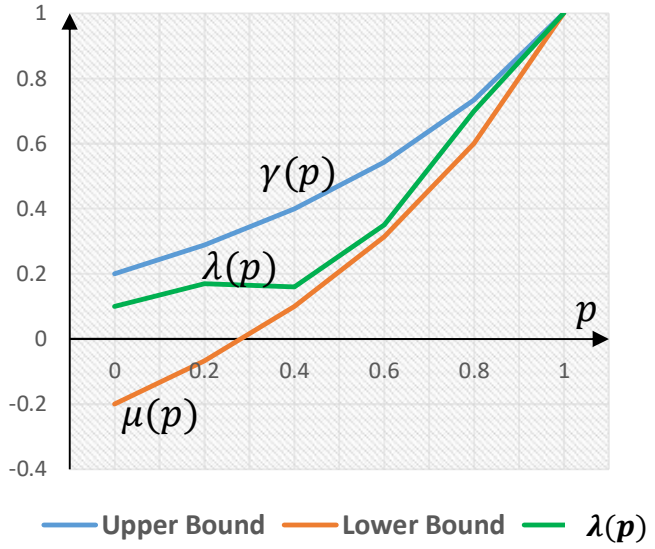
- For all cases, the over-smoothing speed v^{-1} is controlled by $\boldsymbol{\lambda_{m+1}}$ (the second-biggest eigenvalue of normalized adjacency matrix)

- So how to increase λ_{m+1}?

Alleviate Over-Smoothing by DropEdge

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





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

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

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

Alleviate Over-Smoothing by DropEdge

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

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

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

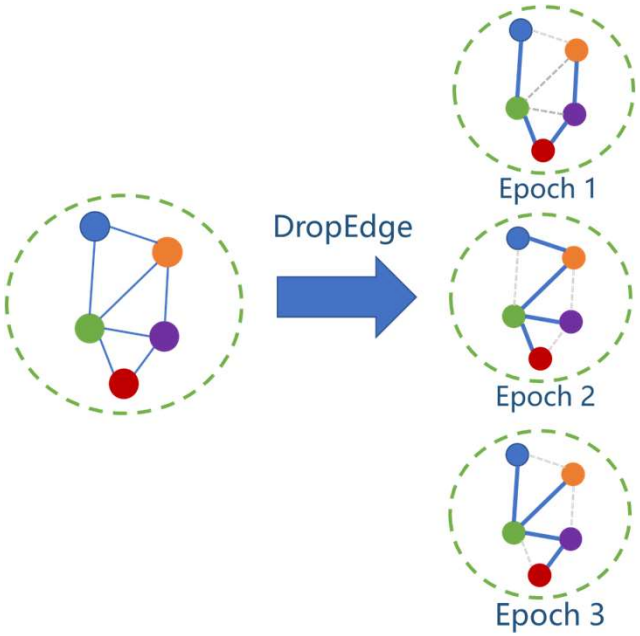
(Rong et al., 2020)

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

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

Alleviate Over-Smoothing by DropEdge

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



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

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

Alleviate Over-Smoothing by Adjacency Matrix

- DropEdge results

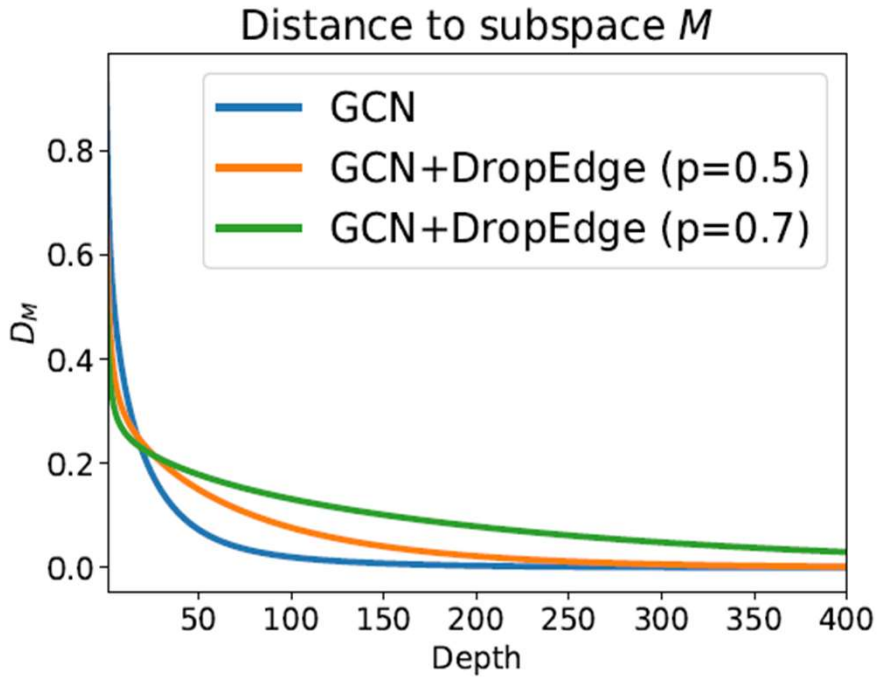
Citeseer	4 layers	DropEdges	16 layers	DropEdges	64 layers	DropEdges
GCN	76.7	79.2(+2.5)	65.2	76.8(+11.6)	44.6	45.6(+1.0)
ResGCN	78.9	78.8(-0.1)	78.2	79.4(+1.2)	21.2	75.3(+54.1)
JKNet	79.1	80.2(+1.1)	78.8	80.1(+1.3)	76.7	80.0(+3.3)
IncepGCN	79.5	79.9(+0.4)	78.5	80.2(+1.7)	79.0	79.9(+0.9)
GraphSAGE	77.3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

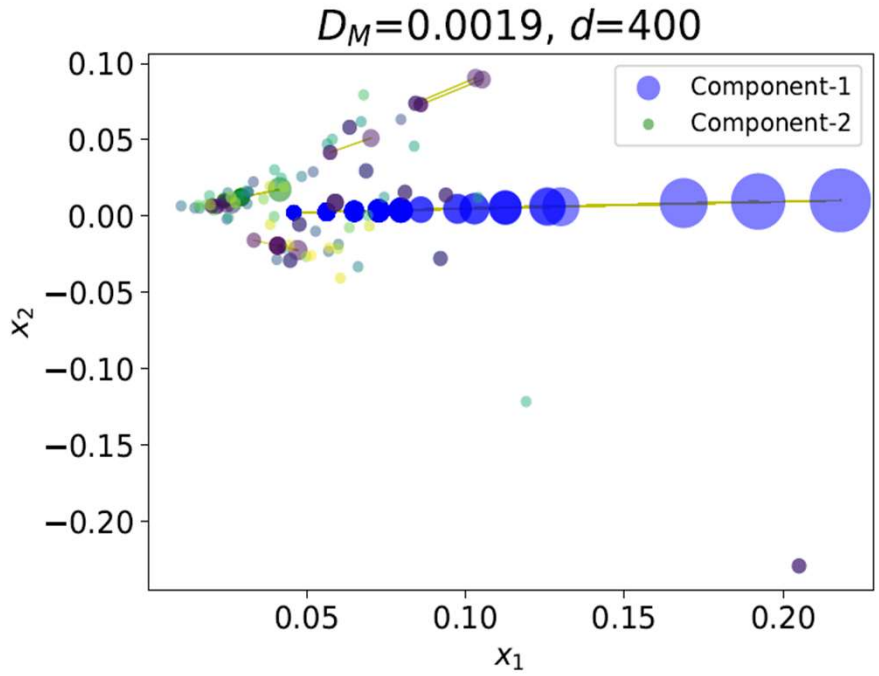
Rong, Yu, et al. "Dropedge: Towards deep graph convolutional networks on node classification." ICLR *2020*.

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

Alleviate Over-Smoothing by Adjacency Matrix

- DropEdge results





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

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

Alleviate Over-Smoothing by Weights

Convergence speed

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

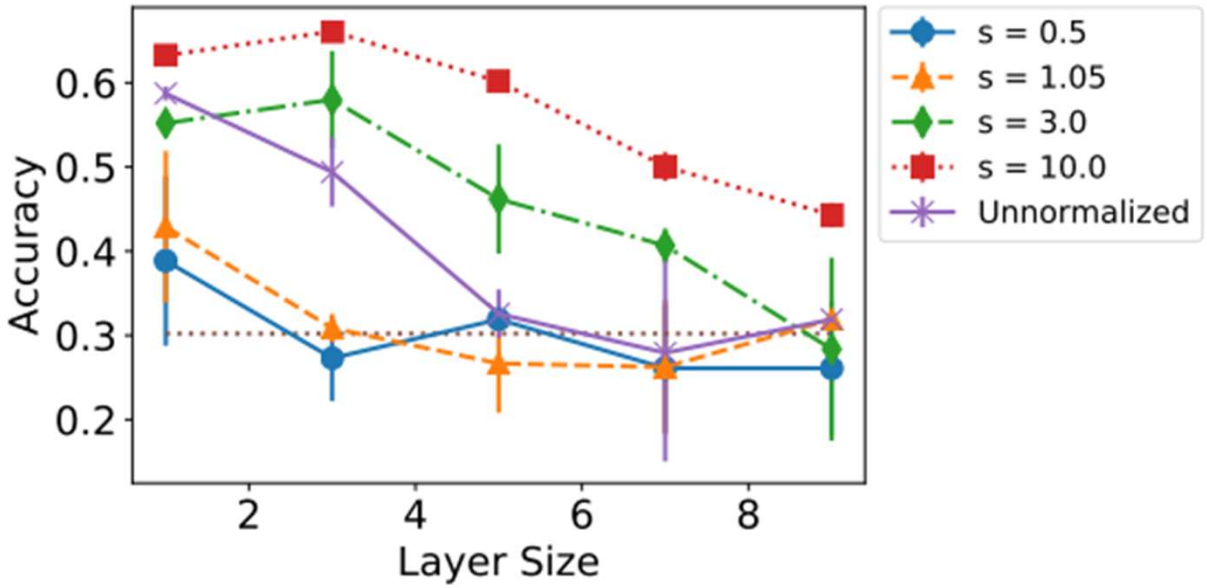
Weights

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

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

Alleviate Over-Smoothing by Weights

- Try different s as initial



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

Training deep GNNs

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

Pair Norm: Center and Rescale

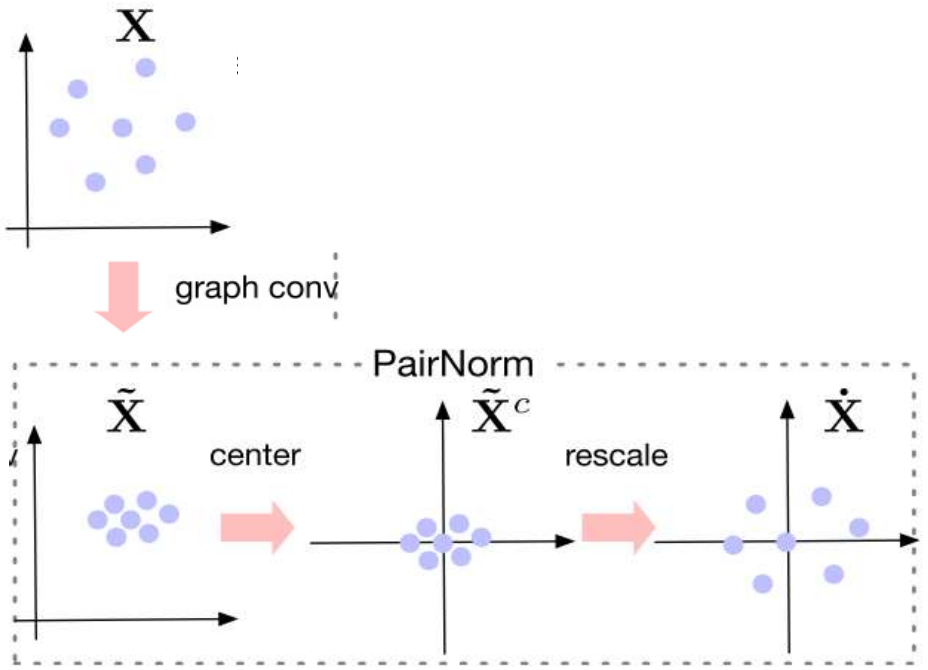
- PairNorm: Center and rescale (normalize) GCN outputs $\tilde{X} := \mathrm{GCN}(A, X)$ to keep the **total pairwise squared distance** *unchanged*

Center

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

Rescale

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

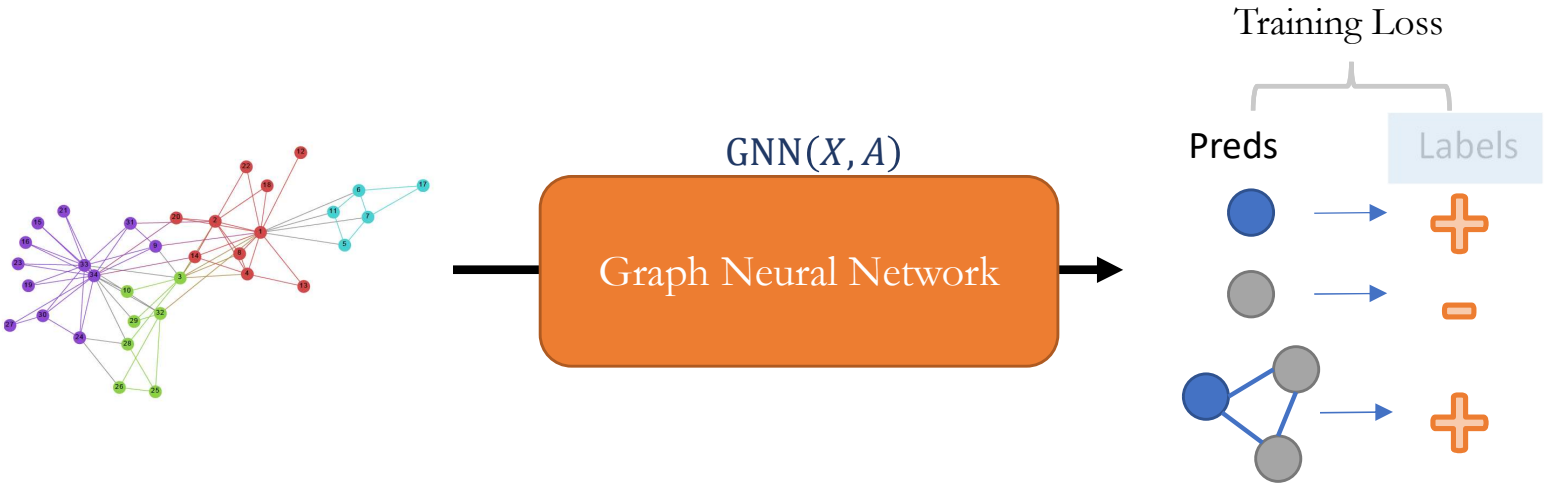


- See also GroupNorm (Zhou et al., 2020)

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

Self/Un-Supervised Learning of GNNs

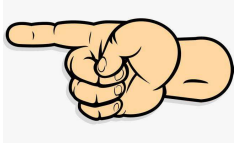
What we discussed before are supervised



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

Existing Self-Supervised GNNs

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



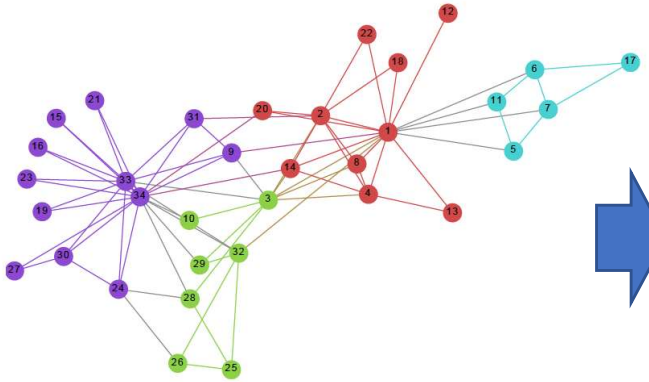
[1] Kipf & Welling 2016; [2] Durán & Niepert 2017; [3] Hamilton et al. 2017;
[4] Liu et al. 2019; [5] Hu et al. 2020; [6] Qiu et al. 2020; [7] Rong et al. 2020;
[8] Veli˘ckovi´c et al. 2019; [9] Peng et al. 2020; [10] Sun et al. 2020
[11] Peng, Zhen, et al. 2020 [12] Hwang, Dasol, et al. 2020
[13] Pan,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

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

Graphs are highly structured!

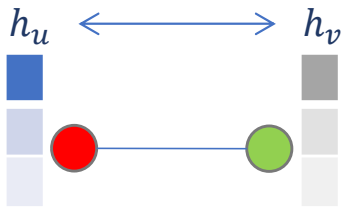
Node Classification

- Two typical ways to formulate training loss



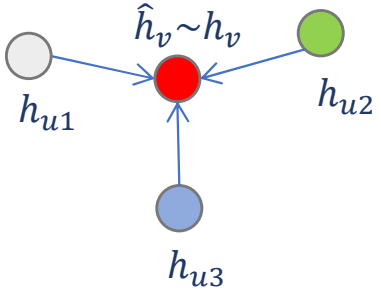
I. Enforcing Adjacent Similarity

GraphSAGE



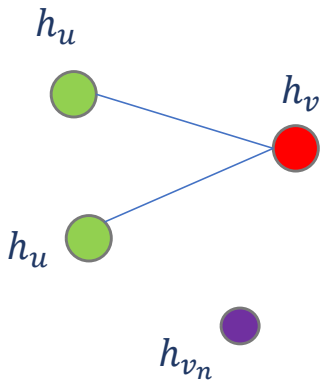
II. Reconstruction from Neighbors

EP-B



I Enforcing Adjacent Similarity

- GraphSAGE (Hamilton et al. 2017)



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

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

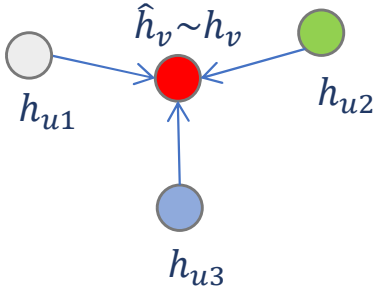
Positive Samples

Negative Samples

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

II Reconstruction from neighbors

- EP-B (Durán & Niepert, 2017)



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

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

Hinge loss

Positive Samples

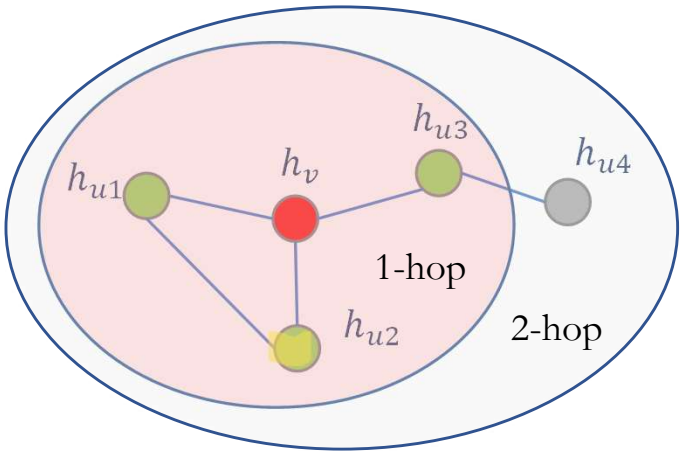
Negative Samples

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

Durán & Niepert. Learning Graph Representations with Embedding Propagation.

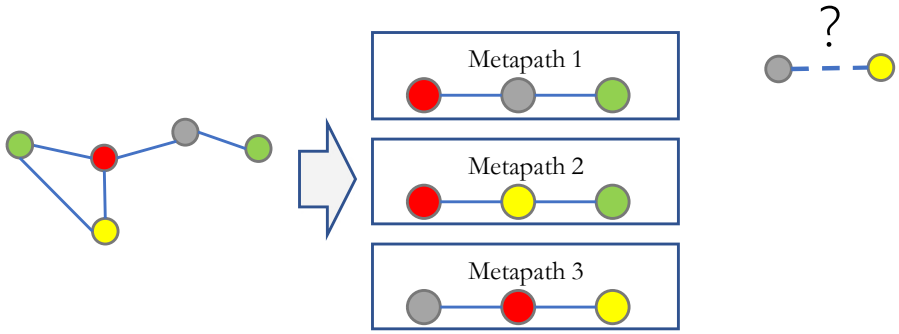
Link/Metapath Prediction

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



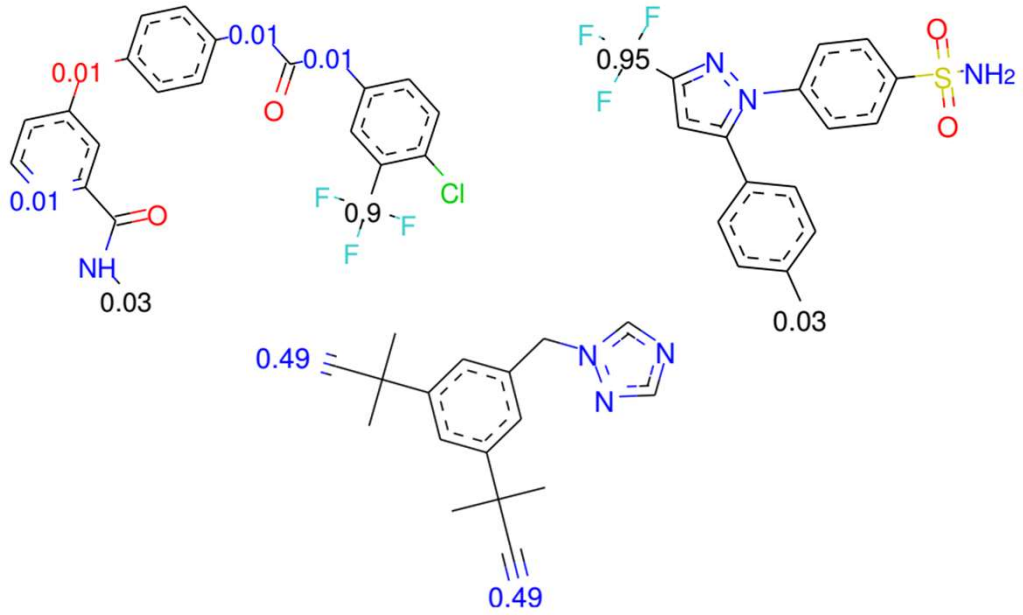
Predict hop counts (K-hop connectivity)

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



Predict the type of meta path between two nodes

How about graph classification/regression?



Toxicity?

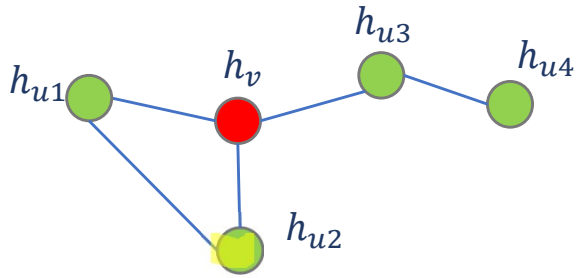
Solubility?

…

N-Gram Graph

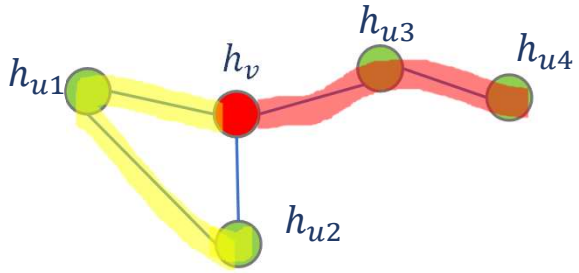
- (Liu et al. 2019)

Stage I: Node Representation



First learn node representations by CBoW-like pipeline

Stage II: Graph Representation



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

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

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

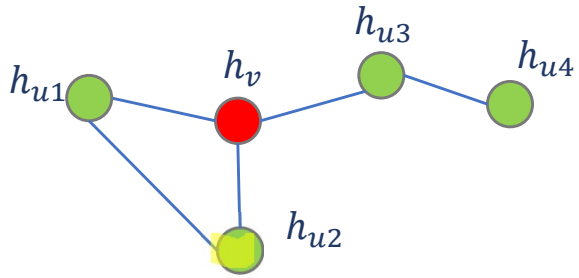
Equivalent to a GNN that needs no training

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

PreGNN: Node- and Graph-level Pretraining

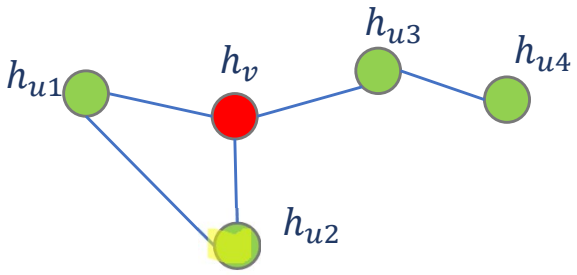
- (Hu et al. 2020)

Stage I: Node Representation



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

Stage II: Graph Representation



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

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

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

Both node- and graph- level training are crucial!

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

PreGNN

- (Hu et al. 2020)

Stage I: Node Representation

Enforcing node representation to be similar to its contextual structures:

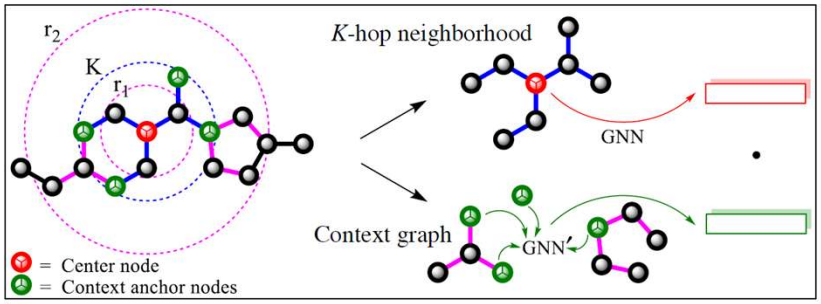
$$\min -\log\left(\sigma\left((h_v{}^K)^T C_v{}^G\right)\right) - I(v \neq v')log(1-\sigma((h_v{}^K)^T C_{v'}{}^{G'}))$$

Positive Samples

Negative Samples

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

Context Prediction
Attribute Masking



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

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

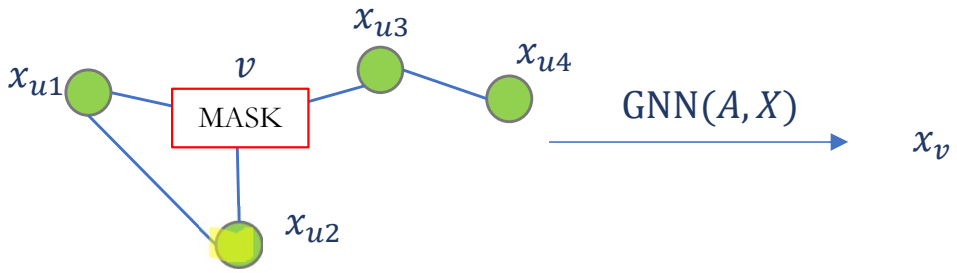
Hu et al. Strategies for Pre-training Graph Neural Networks.

PreGNN

- (Hu et al. 2020)

Stage I: Node Representation

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



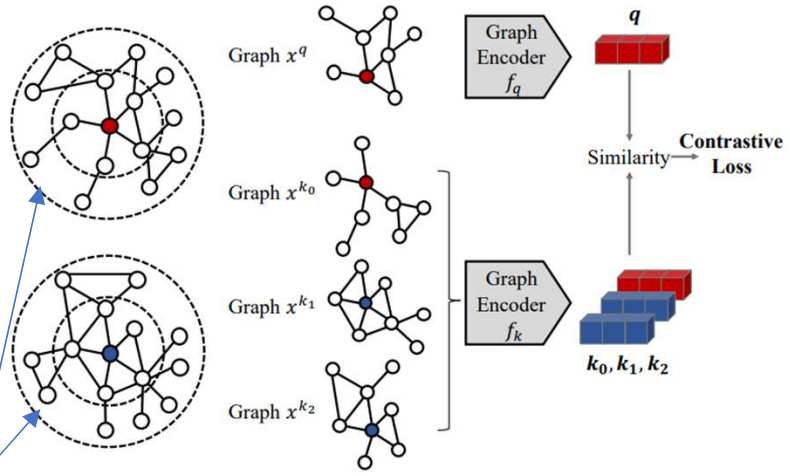
Context Prediction
Attribute Masking

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

GCC: Contrastive learning

- (Qiu et al. 2020)

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



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

InfoNCE

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

r-ego network

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

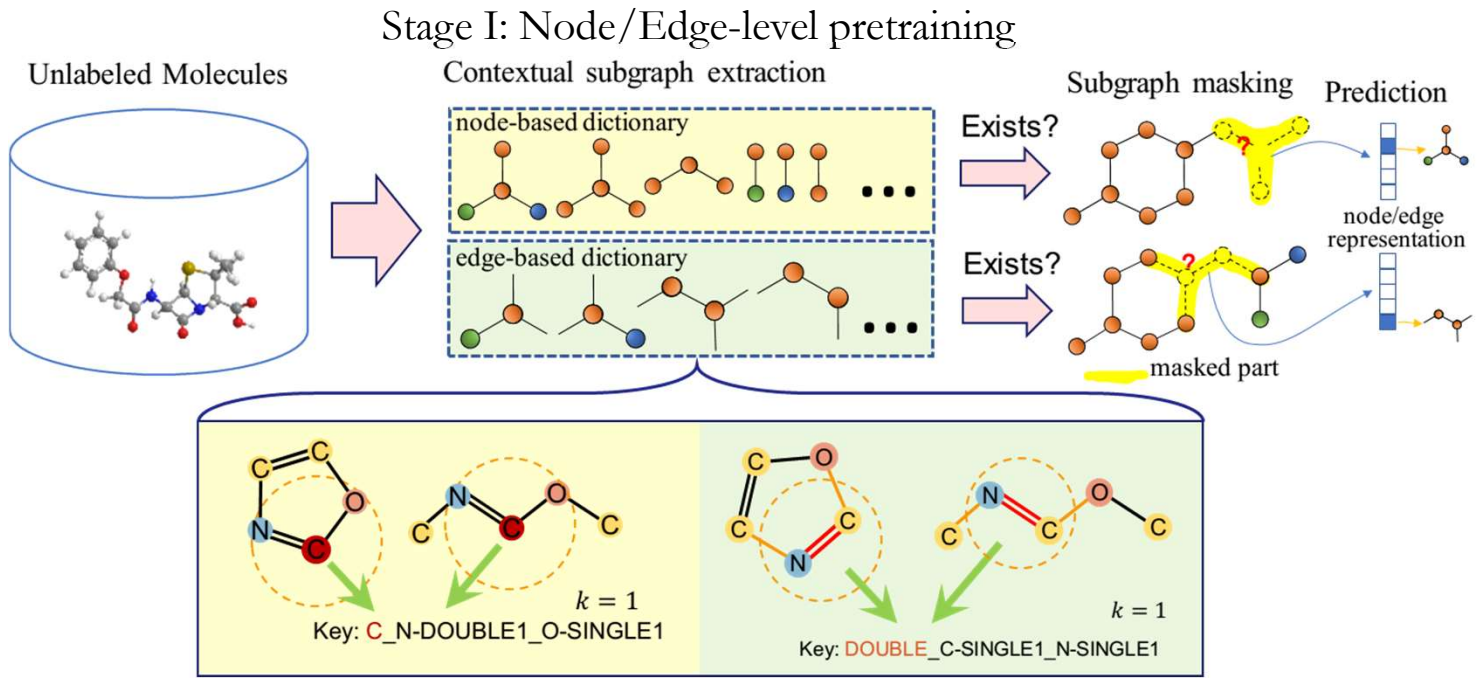
Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training.

GROVER (Rong et al. 2020)

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

GROVER

- (Rong et al. 2020)

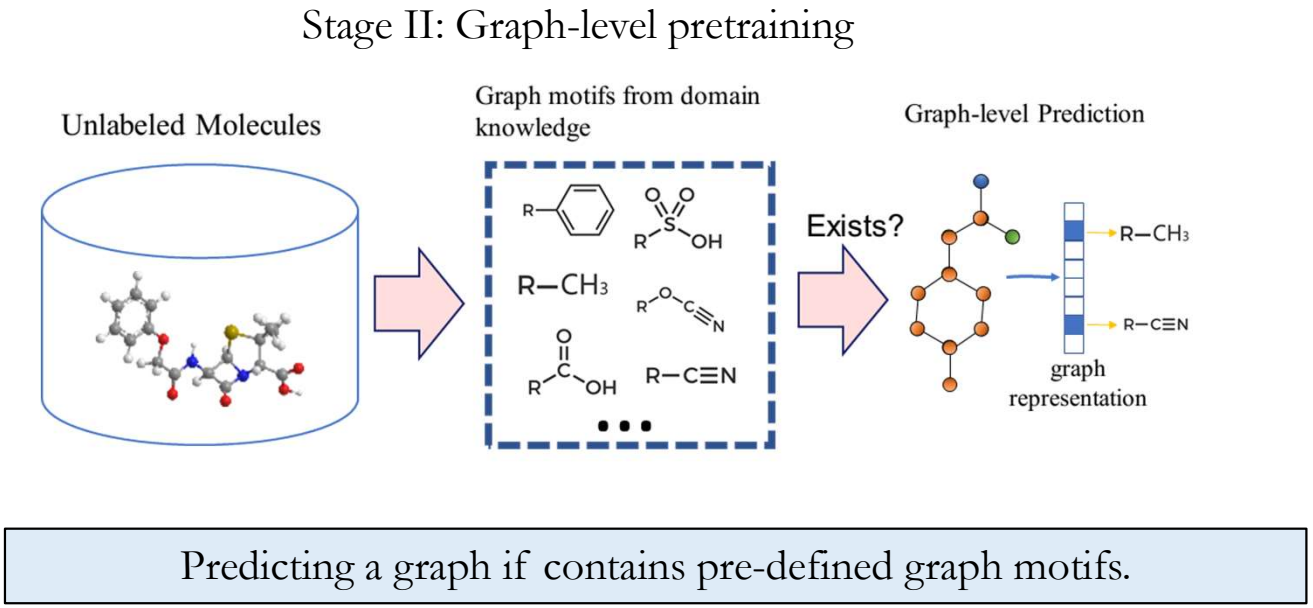


Predicting node/edge contexts instead of node labels can better capture local structures (multi-label)

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

GROVER

- (Rong et al. 2020)



Predicting a graph if contains pre-defined graph motifs.

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

GROVER

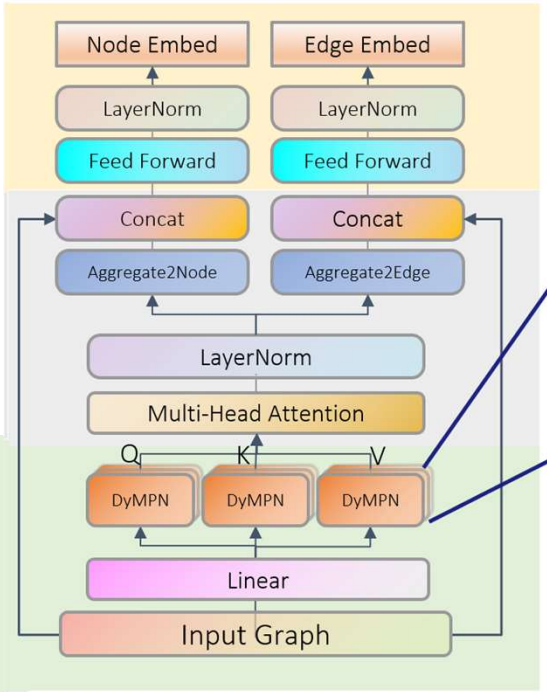
- One more thing: GTransformer

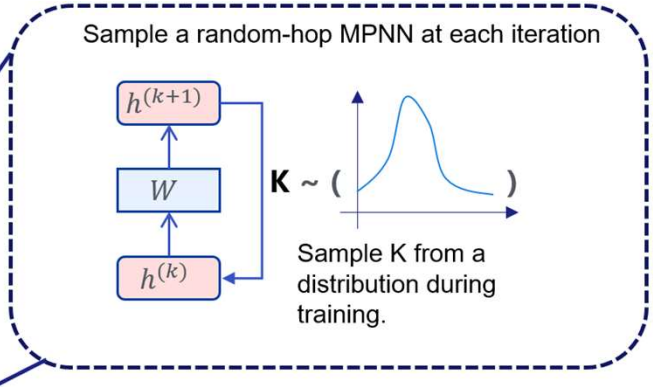
We build a more expressive and transformer-alike model: GTransformer

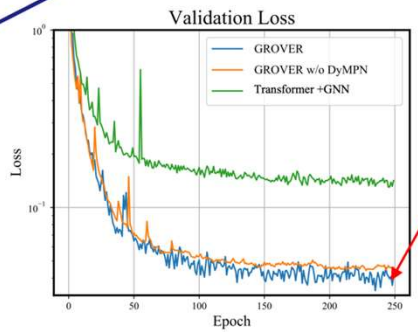
- Output for both node embedding and edge embedding.

- 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

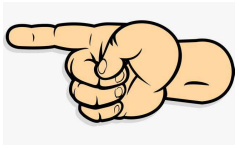
Molecular classification

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

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

Existing Self-Supervised GNNs

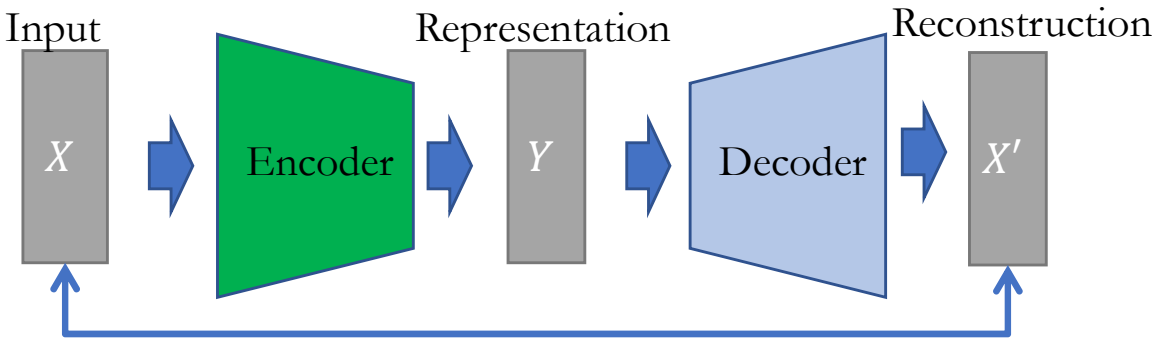
	Node-Classification	Link/Metapath Prediction	Graph-Classification	Graph Reconstruction
Predictive Methods	EP-B [2], GraphSAGE [3], GROVER [7]	S^2GRL[11] SELAR[12]	N-gram Graph [4], PreGNN [5] , GROVER [7] GCC [6]	
Information-based Methods	DGI [8], GMI [9]		InfoGraph [10]	VGAE [1] VRVGA[13] SIG-VAE[14]



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

What makes a good representation?

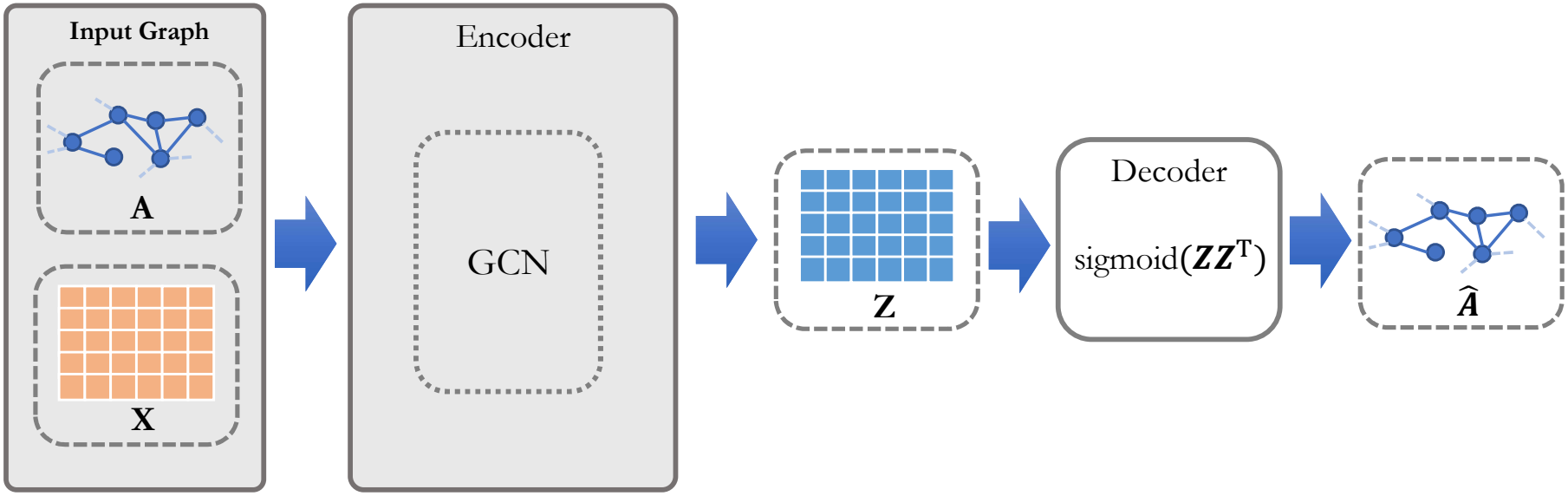
Auto-Encoder (AE)



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

Hinton & Salakhutdinov 2006; Vincent et al. 2010

Graph Auto-Encoders (VGAE)



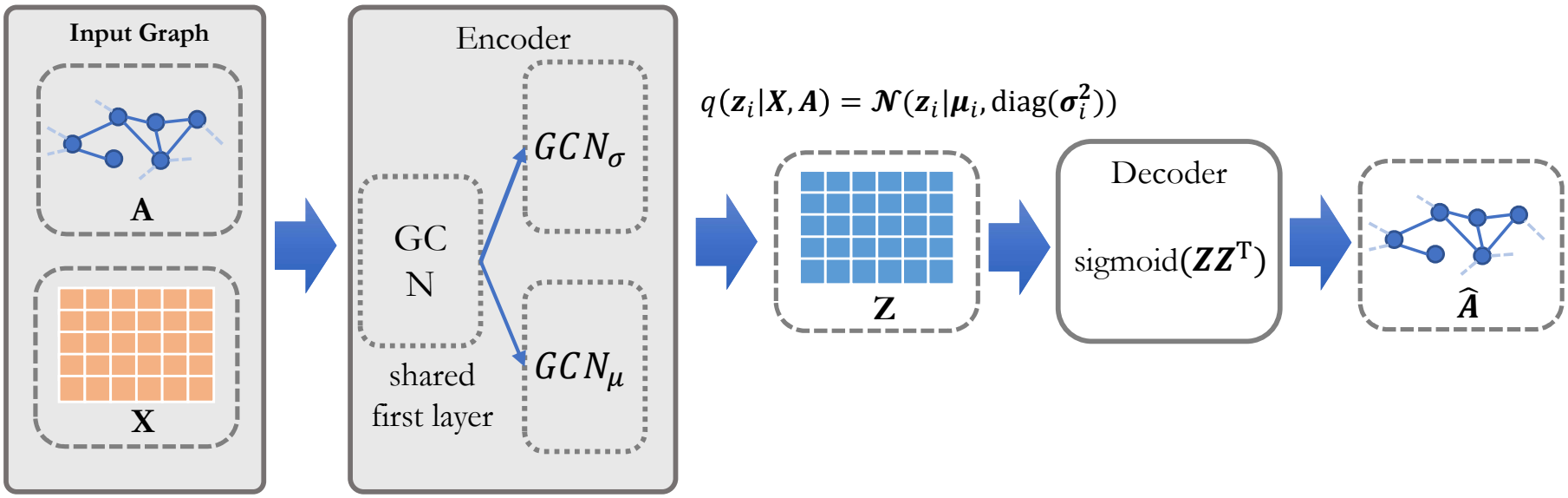
The overall loss:

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

The reconstruction loss.

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

Variational Graph Auto-Encoders (VGAE)



The overall loss:

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

The reconstruction loss.

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

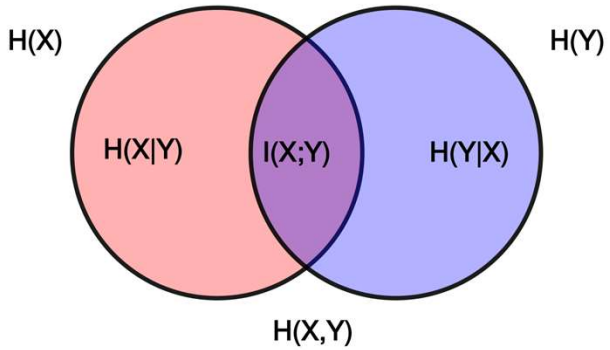
Kipf, T. N., & Welling, M. (2016).

What makes a good representation?

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

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

Entropy Conditional Entropy



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

AE is a lower bound of MI

(Hjelm et al. 2019)

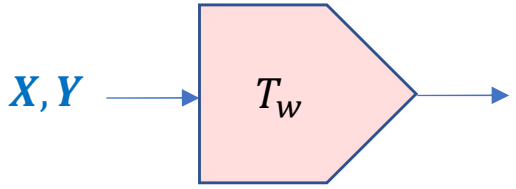
$$I(X;Y) = H(X) - H(X|Y) \geq H(X) - R(X|Y)$$

Mutual Information

Reconstruction error

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

Estimating/Maximizing MI (Hjelm et al. 2019)



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

Maximize Lower bound of MI

MINE (Belghazi et al., 2018):

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

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

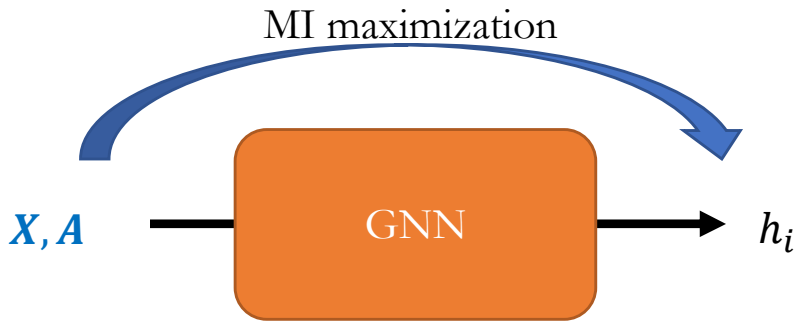
$$I^{\text{JSD}}(X;Y) \triangleq E_{p(X,Y)}\left[\log \sigma\left(T_w(x,y)\right)\right] + E_{p(X)p(Y)}[\log(1-\sigma\left(T_w(x,y)\right))]$$

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

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

Deep Graph Infomax (DGI)

- (Velickovic et al. 2019)



The JSD MI estimator is applied:

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

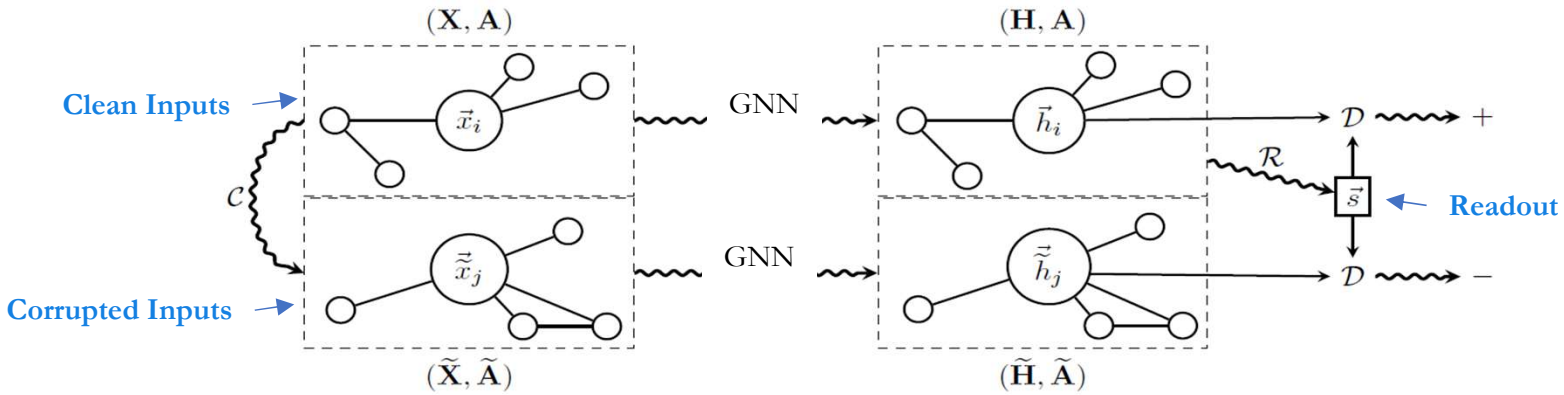
$h_i = \text{GNN}(X, A)$ $\tilde{h}_i$ negative sample

Deep Graph Infomax (DGI)

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

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

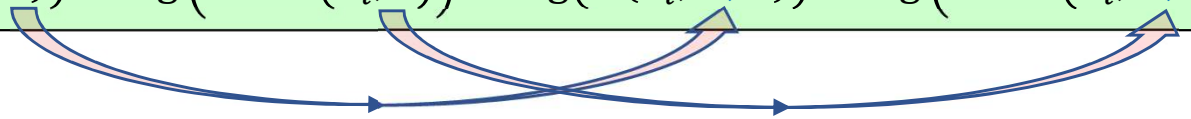
$$\max_{\text{GNN}} I(X, A; h_i) = \max_{\text{GNN}} \log(D(h_i; \boldsymbol{s})) + \log(1 - D(\tilde{h}_i; \boldsymbol{s}))$$



Deep Graph Infomax (DGI)

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

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



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

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

- Some issues in DGI

  - Computing MI requires the injectivity of readout function

  - It resorts to graph corruption to generate negative samples

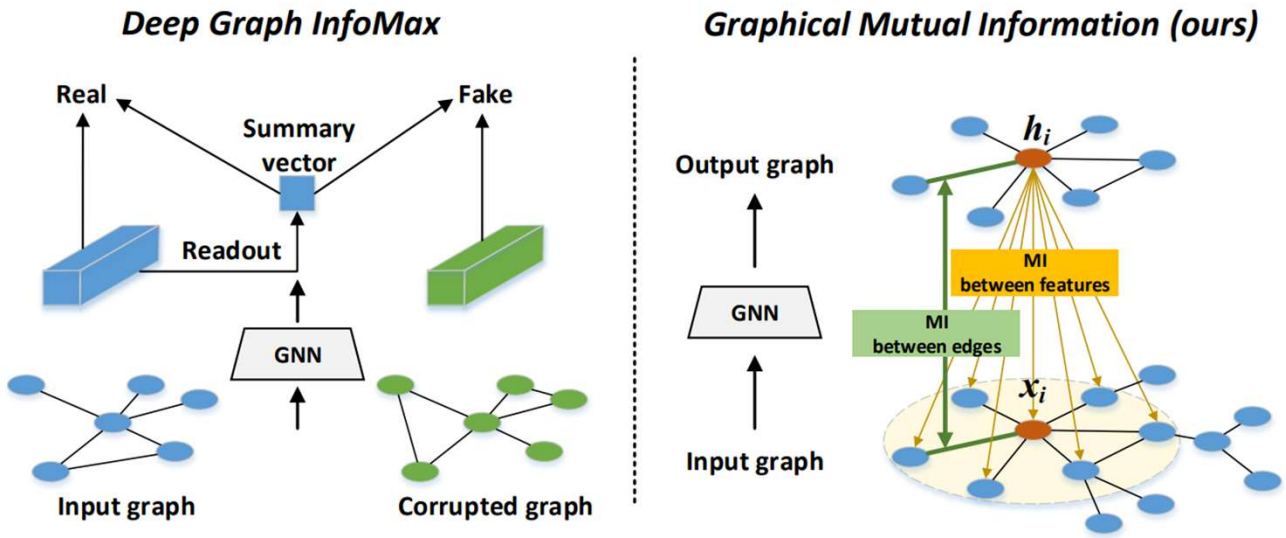
  - Distinct encoders and corruption functions for different tasks

Indirect

Inefficient

GMI: Graphical Mutual Information

- (Peng et al. 2020)



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

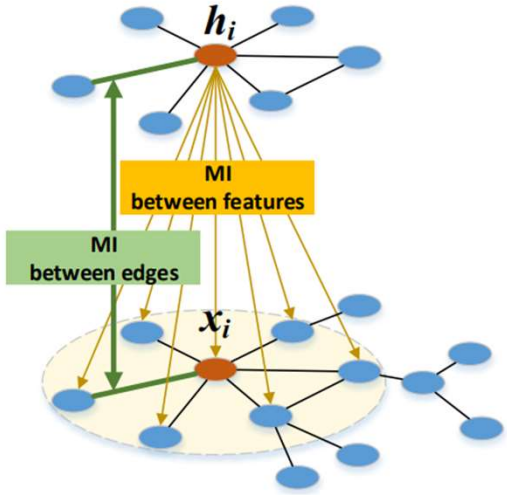
Peng et al. Graph Representation Learning via Graphical Mutual Information Maximization.

GMI: Graphical Mutual Information

We define that,

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

Feature MI Topology MI



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

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

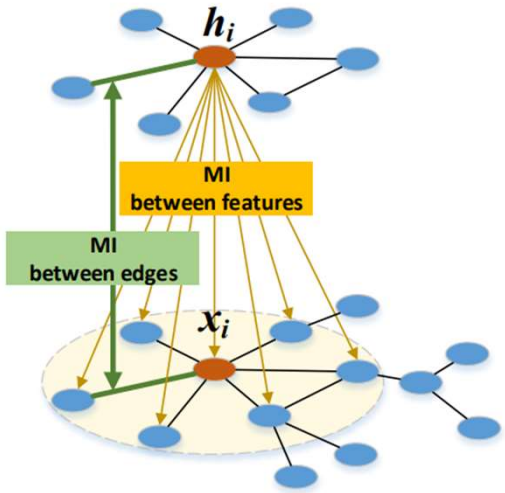
Peng et al. Graph Representation Learning via Graphical Mutual Information Maximization.

GMI: Graphical Mutual Information

- (Peng et al. 2020)

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

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



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

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

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

GMI: Graphical Mutual Information

- Node Classification

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

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

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

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

GMI: Graphical Mutual Information

- Link Prediction

We use an universal backbone (GCN) for all tasks

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

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

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

Summary

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

Applications

- **GNN in Social Networks**

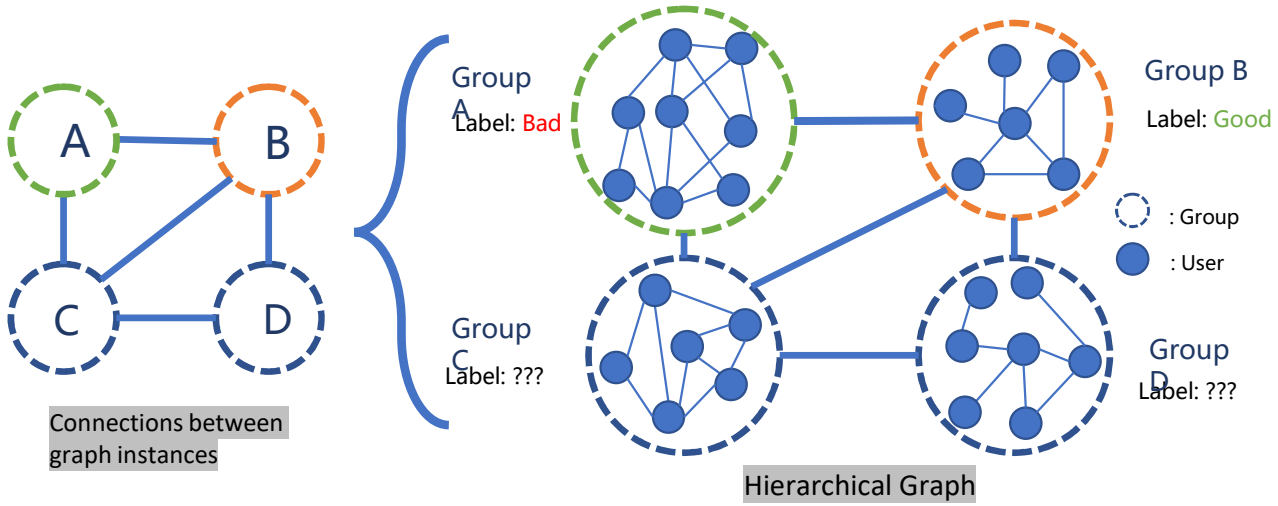
GNN in Social Networks

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

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

Hierarchical Graph Classification

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

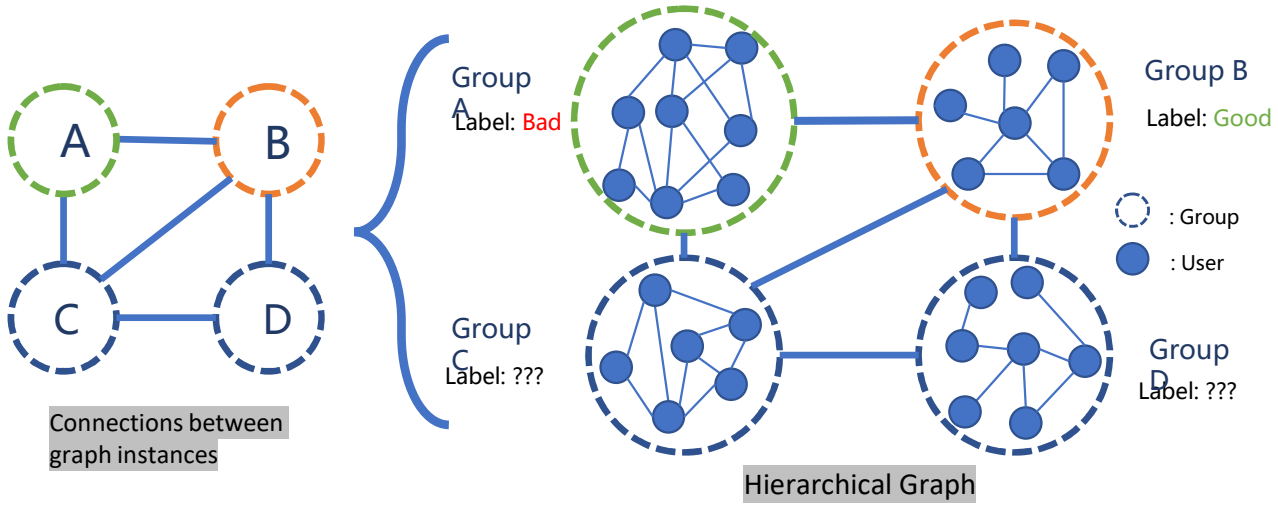


- **The Problem:** predicts the class label of graph instances in a hierarchical graph.

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

Hierarchical Graph Classification

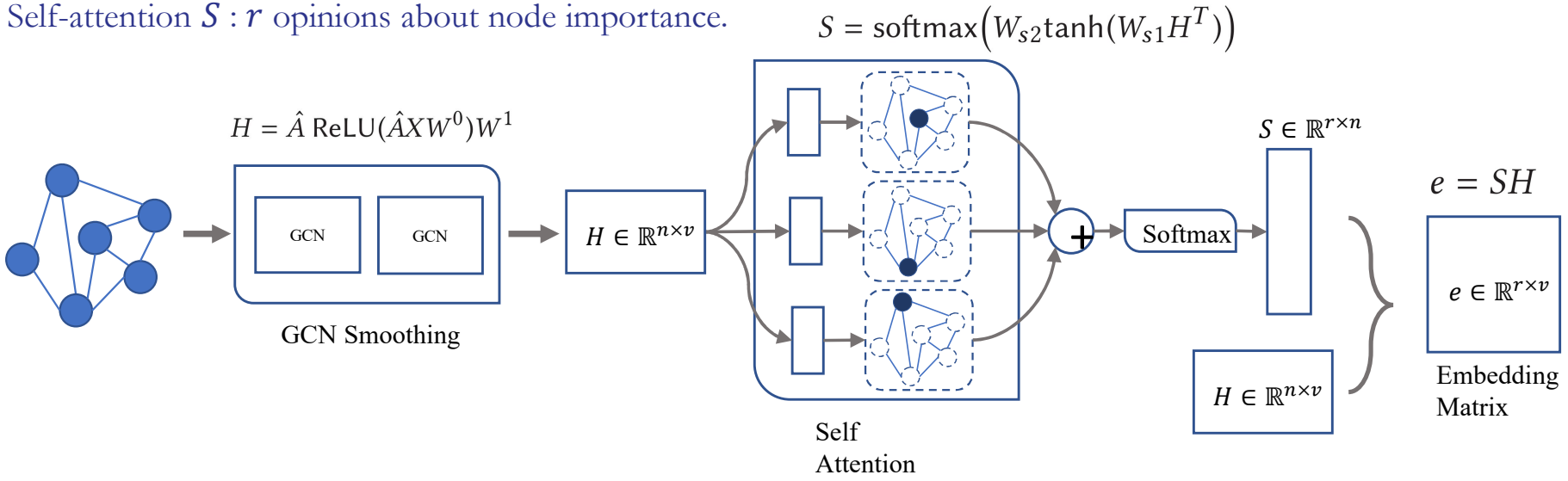
- **The Problem:** predicts the class label of graph instances in a hierarchical graph.
- **Challenges:**
  - How to represent the graphs with arbitrary size into a fixed-length vector?
  - How to incorporate the information of instance level and hierarchical level?



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

Graph Instance Level: Self-Attentive Graph Embedding

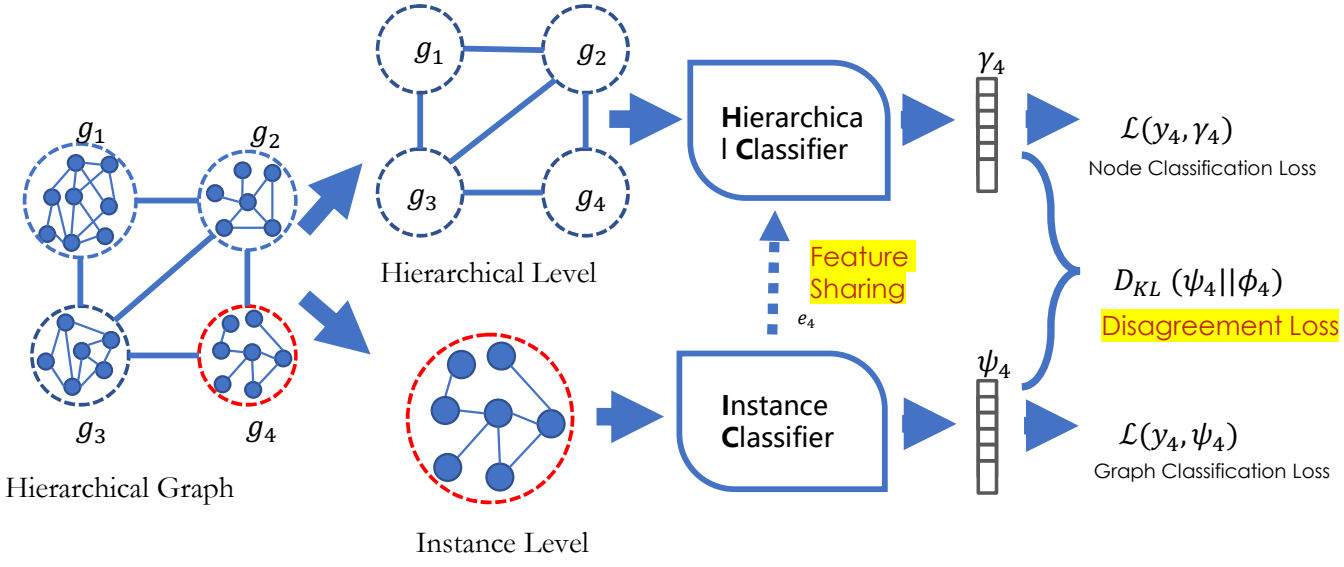
- How to represent the graphs with arbitrary size into a fixed-length vector?
- Graph representation learning at different level:
  - Node Level: $G(V,E) \rightarrow H^{n \times v}$
  - Graph Level: $G(V,E) \rightarrow e^{v}$
- **SAGE: Self-Attentive Graph Embedding**
  - Size invariance ---- Self-attention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



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

The Unified Model

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



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

The supervised loss:

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

The disagreement loss:

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

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

Applications

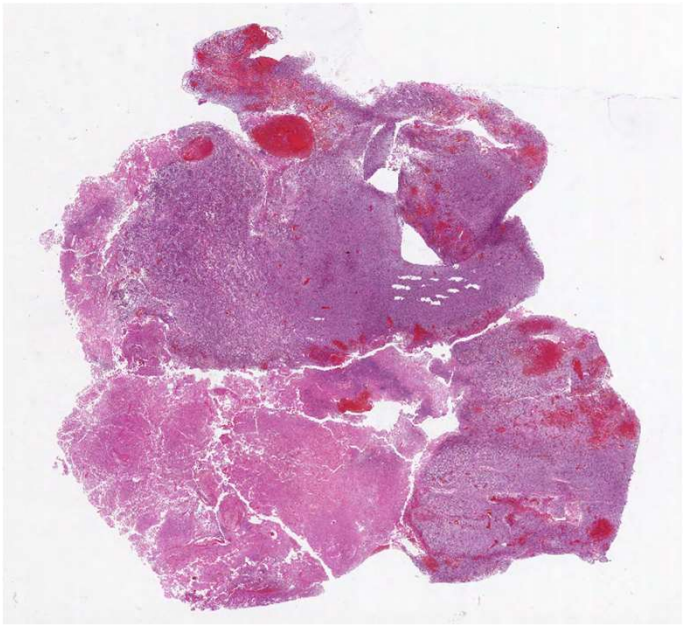
- **GNN in Medical Imaging**

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

GNN in Medical Imaging

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



GNN in Medical Imaging

- **Cox proportional hazard function**

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

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

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

where, Y is the observation time.

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

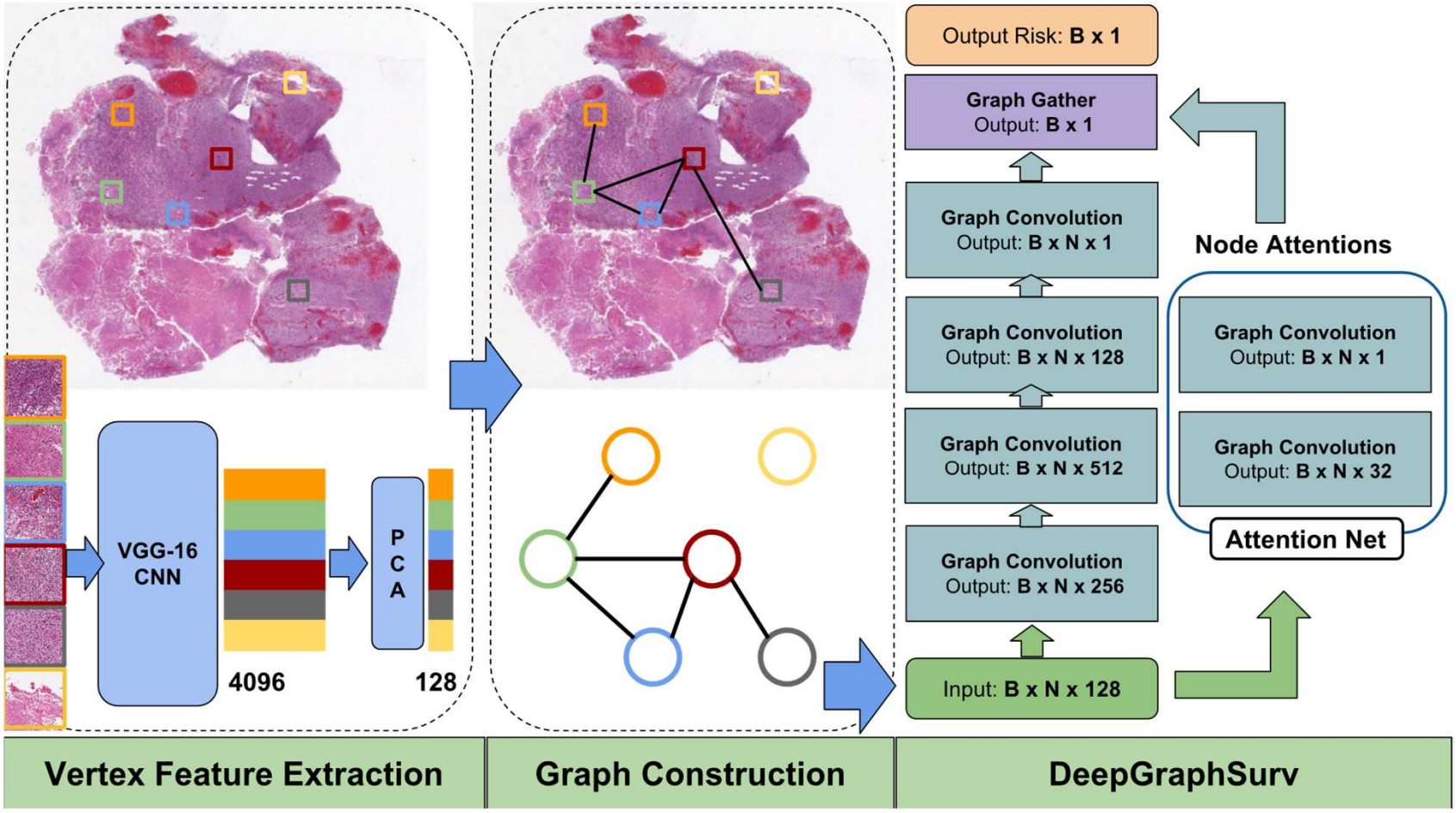
- **Join likelihood of all subjects:**

$$L(\beta) = \prod_{i:C_i=1} L_i(\beta)$$

- **Log likelihood as object function:**

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

GNN in Medical Imaging



GNN in Medical Imaging

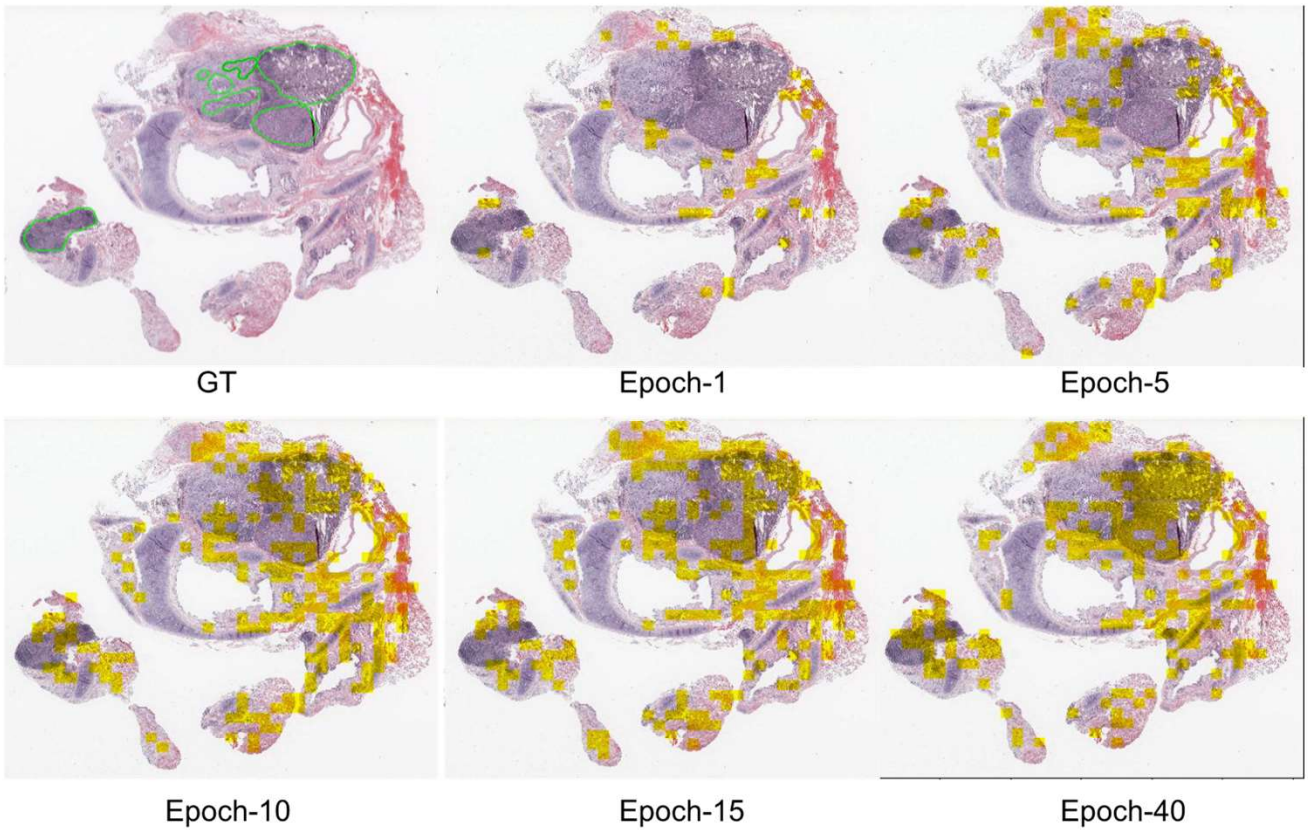
- **Pathological Images and Patient Survival Time and Label**
  - TCGA, The Cancer Genome Atlas
  - NLST, National Lung Screening Trials

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.

GNN in Medical Imaging

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

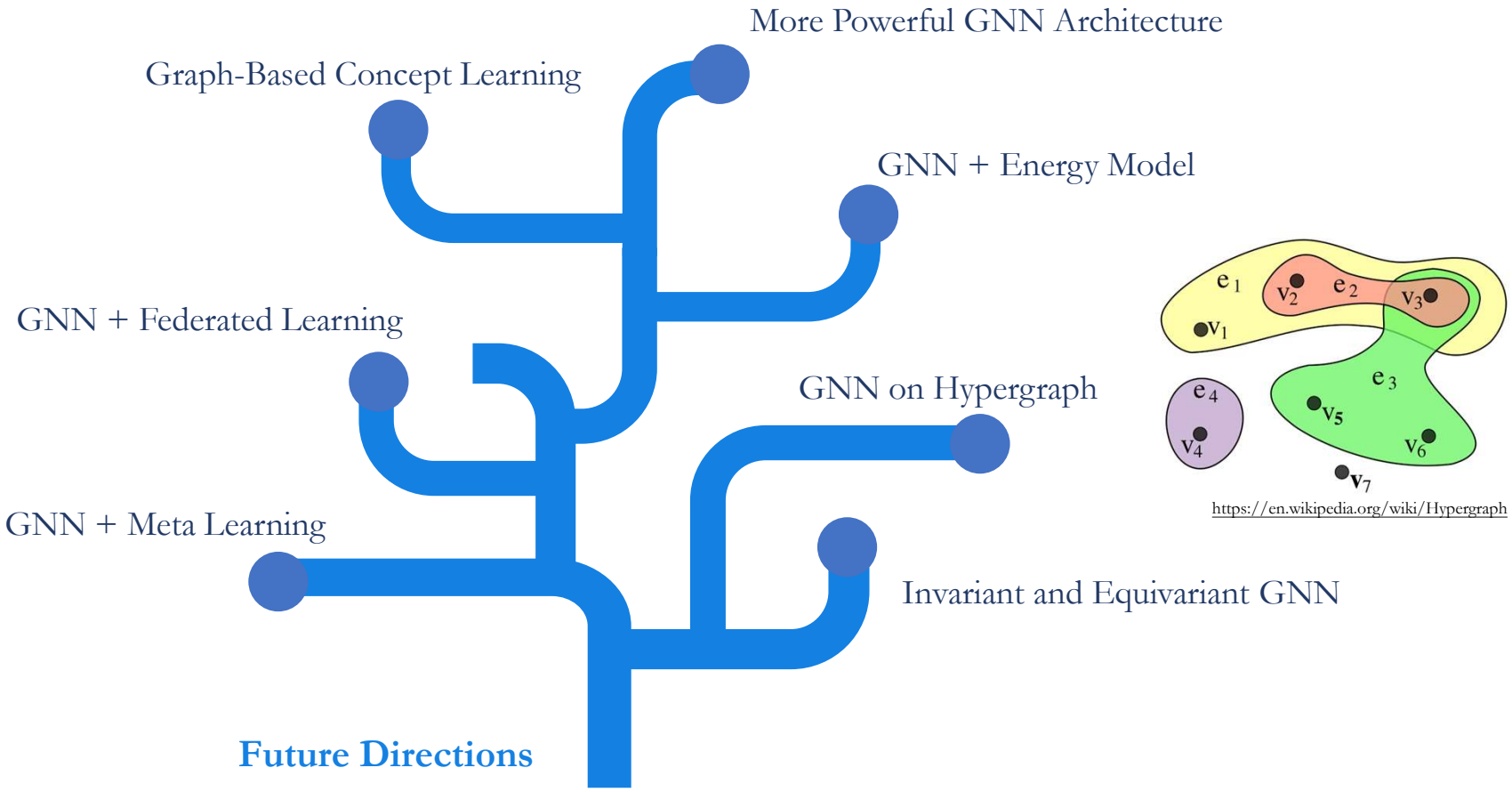


GNN in Medical Imaging

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



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.
- Junchi, Yu, et al. “Graph Information Bottleneck for Subgraph Recognition.”, ICLR 2021.