Graph Neural Networks and Self-supervised Learning

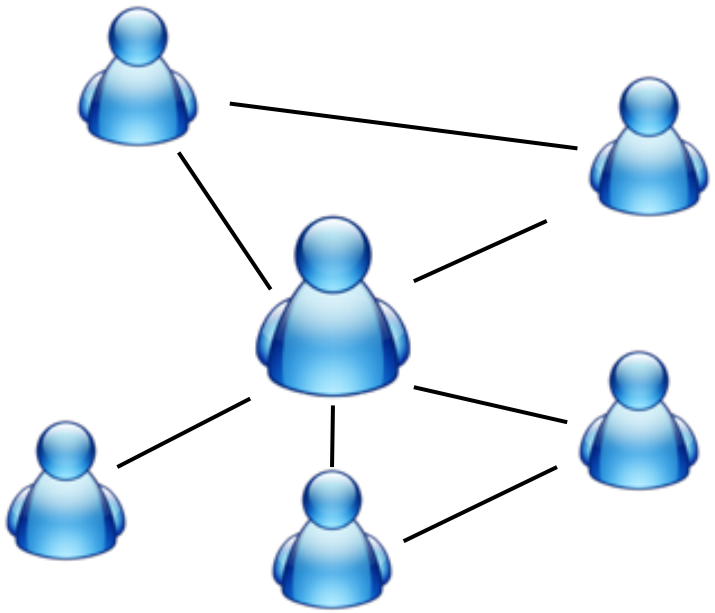
Jie Tang

Computer Science

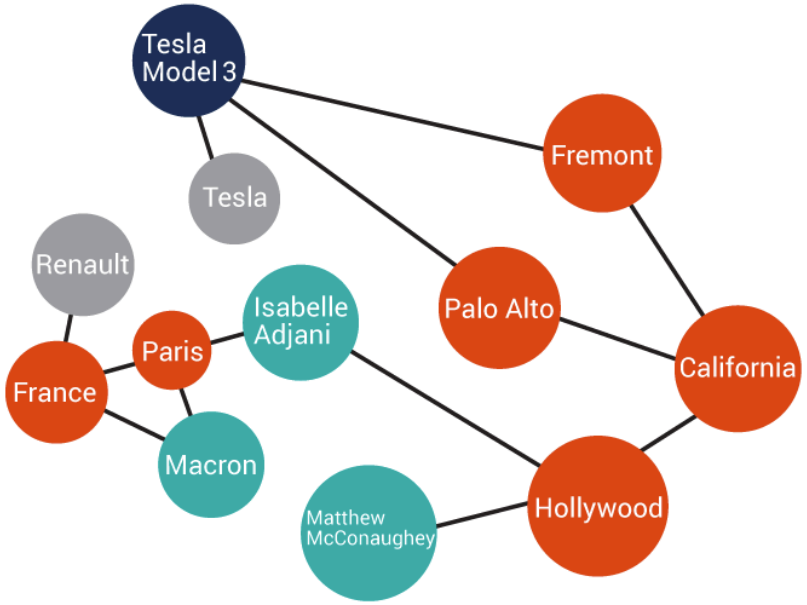
Tsinghua University

The slides can be downloaded at http://keg.cs.tsinghua.edu.cn/jietang

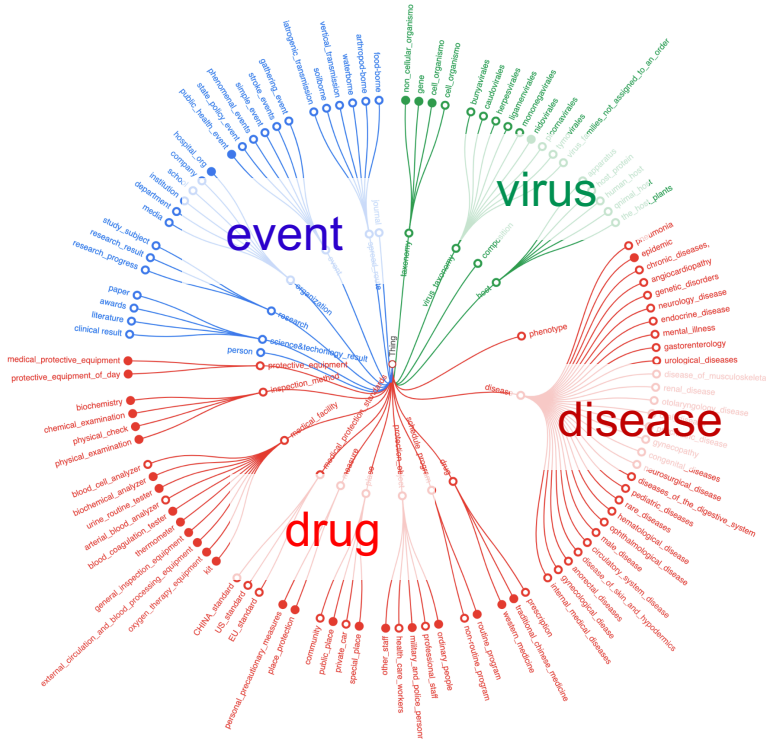
Networked Data



Social Network



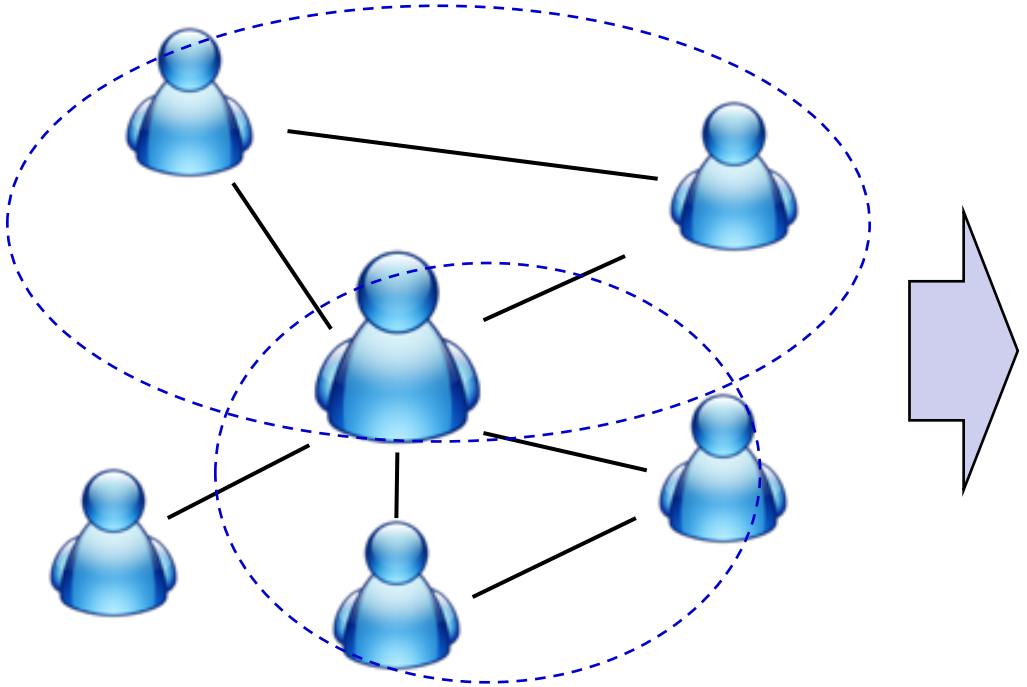
Knowledge Graph



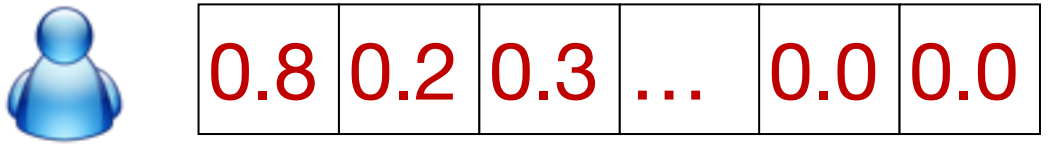
COVID Graph

Representation Learning on Networks

Representation Learning/
Graph Embedding

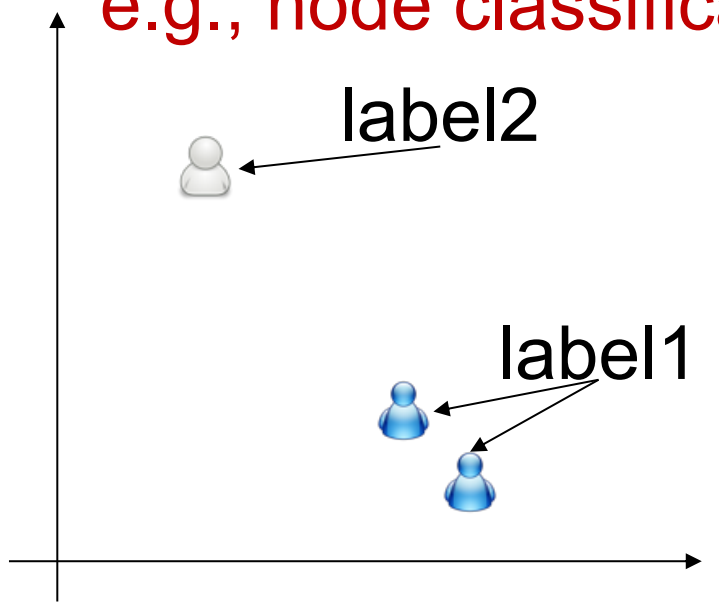


d-dimensional vector, $d << |V|$

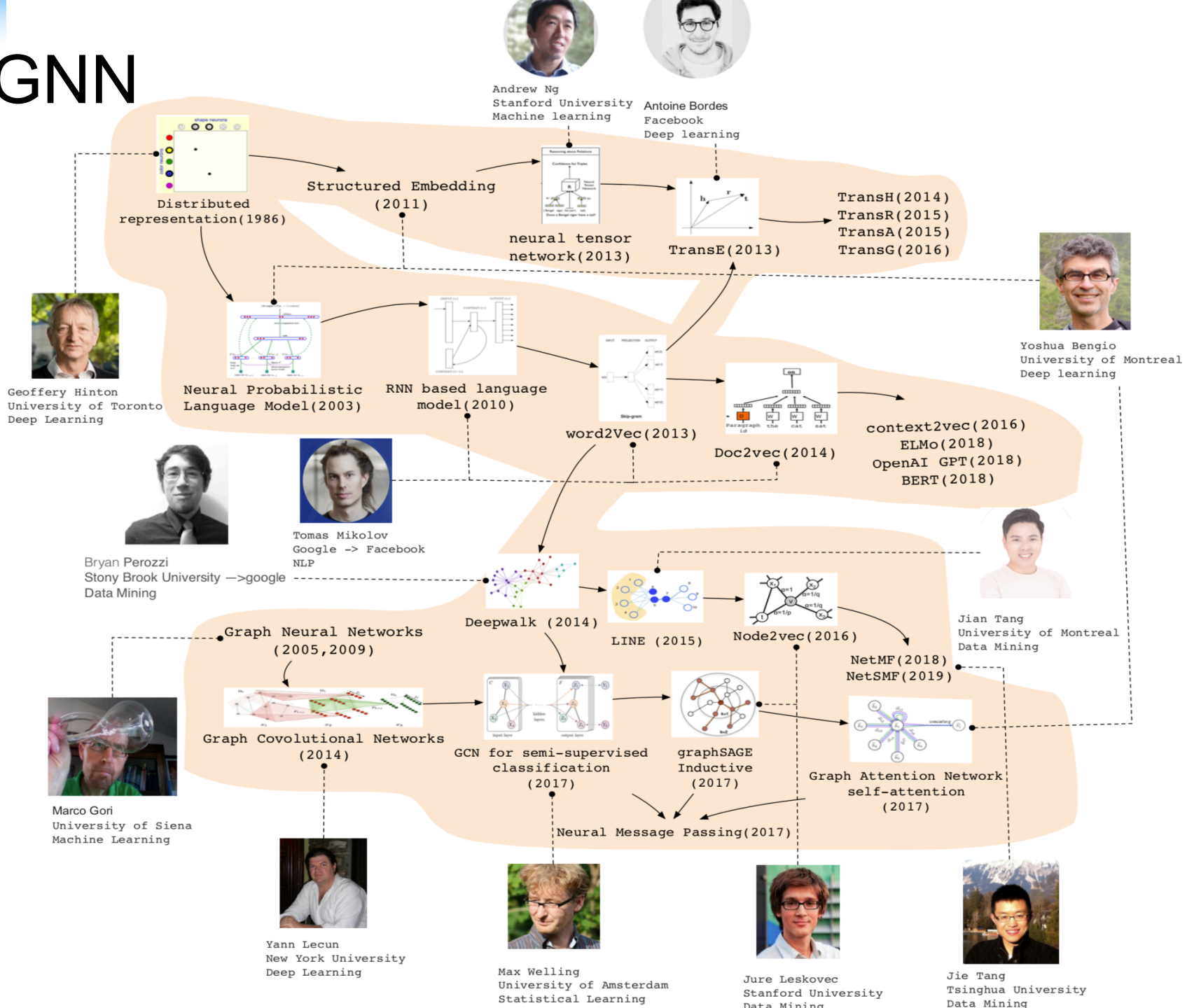


Users with the same label are located in closer

e.g., node classification



GRL: NE&GNN


Andrew Ng
Stanford University
Machine learning
Antoine Bordes
Facebook
Deep learning
Distributed representation(1986)
Structured Embedding (2011)
neural tensor network(2013)
TransE(2013)
TransH(2014)
TransR(2015)
TransA(2015)
TransG(2016)
Yoshua Bengio
University of Montreal
Deep learning
Geoffery Hinton
University of Toronto
Deep Learning
Neural Probabilistic Language Model(2003)
RNN based language model(2010)
word2Vec(2013)
Doc2vec(2014)
context2vec(2016)
ELMo(2018)
OpenAI GPT(2018)
BERT(2018)
Bryan Perozzi
Stony Brook University —>google
Data Mining
Tomas Mikolov
Google -> Facebook
NLP
Deepwalk (2014)
LINE (2015)
Node2vec(2016)
Jian Tang
University of Montreal
Data Mining
NetMF(2018)
NetSMF(2019)
Graph Neural Networks (2005,2009)
Graph Covolutional Networks (2014)
GCN for semi-supervised classification (2017)
graphSAGE Inductive (2017)
Graph Attention Network self-attention (2017)
Marco Gori
University of Siena
Machine Learning
Neural Message Passing(2017)
Yann Lecun
New York University
Deep Learning
Max Welling
University of Amsterdam
Statistical Learning
Jure Leskovec
Stanford University
Data Mining
Jie Tang
Tsinghua University
Data Mining

GRL: NE&GNN

Network Embedding ⇔ Matrix Factorization

Graph Neural Networks ⇨ GNN Pre-Training

Graph Data & Benchmarks

OPEN GRAPH BENCHMARK

https://alchemy.tencent.com/

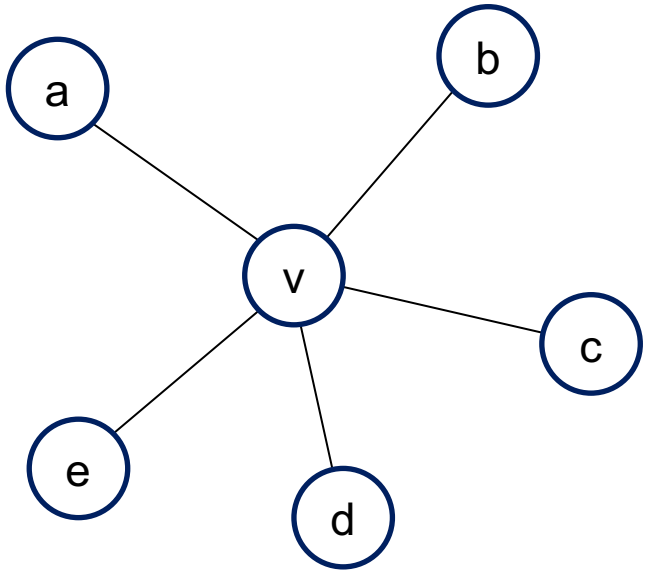
Miner

Microsoft Academic

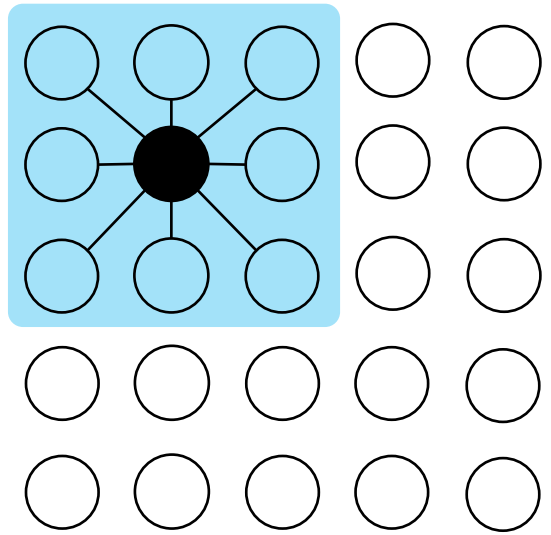
CogDL

https://github.com/thudm/cogdl

Graph Neural Networks



1. Choose neighborhood
2. Determine the order of selected neighbors
3. Parameter sharing



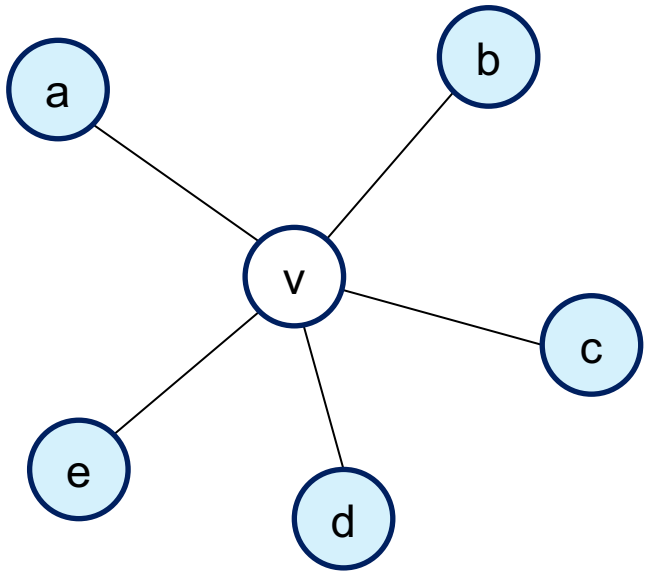
Graph Convolution **CNN**

Neighborhood Aggregation:

- Aggregate neighbor information and pass into a neural network
- It can be viewed as a center-surround filter in CNN---graph convolutions!

1. Niepert et al. Learning Convolutional Neural Networks for Graphs. In ICML 2016
2. Defferrard et al. Convolutional Neural Networks on Graphs with Fast Locailzied Spectral Filtering. In NIPS 2016
3. Huawei Shen. Graph Neural Networks. A video talk (in Chinese), June 2020. https://dl.ccf.org.cn/audioVideo/detail.html?id=4966039790962688&_ack=1

Graph Convolutional Networks



Non-linear activation function (e.g., ReLU)

parameters in layer k

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}^k \sum_{u \in N(v) \cup v} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}}\right)$$

node v's embedding at layer k

the neighbors of node v

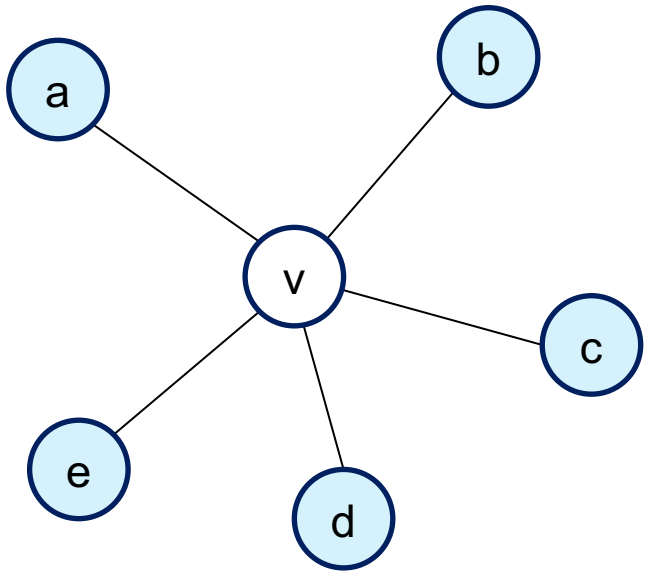
$$\boldsymbol{H}^k = \sigma\left(\widehat{\boldsymbol{A}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right)$$

normalized Laplacian matrix

Aggregate info from neighborhood via the normalized Laplacian matrix

1. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

Graph Convolutional Networks



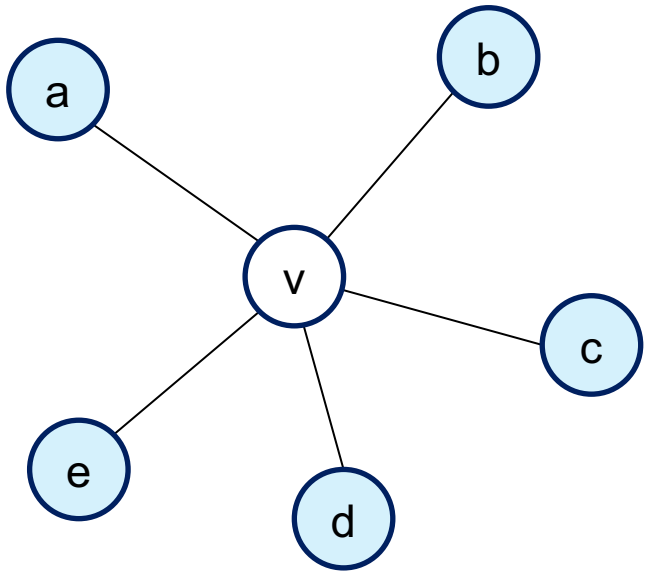
Aggregate from v's neighbors

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v)} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}^k \sum_{v} \frac{\boldsymbol{h}_v^{k-1}}{\sqrt{|N(v)||N(v)|}})$$

Aggregate from itself

1. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. ICLR 2017

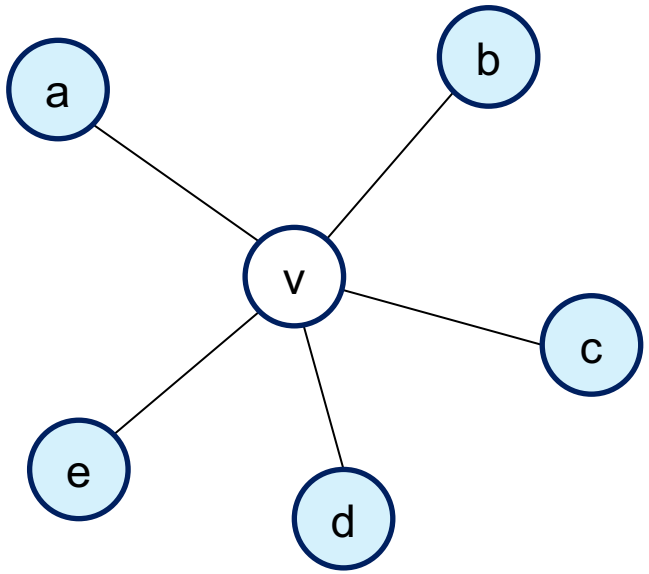
Graph Convolutional Networks



The same parameters for both its neighbors & itself

$$\boldsymbol{h}_v^k = \sigma\left(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v)} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}^k \sum_{v} \frac{\boldsymbol{h}_v^{k-1}}{\sqrt{|N(v)||N(v)|}}\right)$$

Graph Convolutional Networks

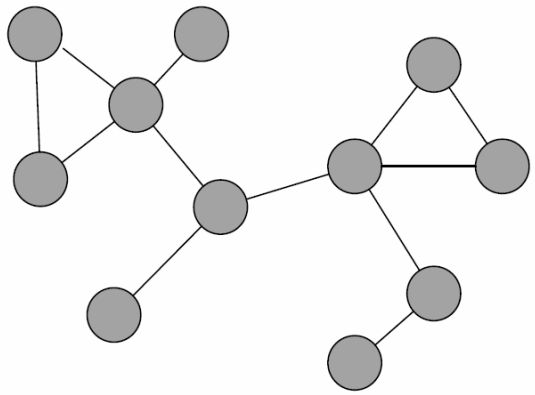


$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}$$

$$\boldsymbol{h}_v^k = \sigma(\boldsymbol{W}^k \sum_{u \in \mathrm{N}(v)} \frac{\boldsymbol{h}_u^{k-1}}{\sqrt{|N(u)||N(v)|}} + \boldsymbol{W}^k \sum_{v} \frac{\boldsymbol{h}_v^{k-1}}{\sqrt{|N(v)||N(v)|}})$$

$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{I}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}$$

Graph Convolutional Networks



$G = (V, E, \boldsymbol{A})$

$\boldsymbol{H}^0 = \boldsymbol{X}$

Input

$$\boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right)$$

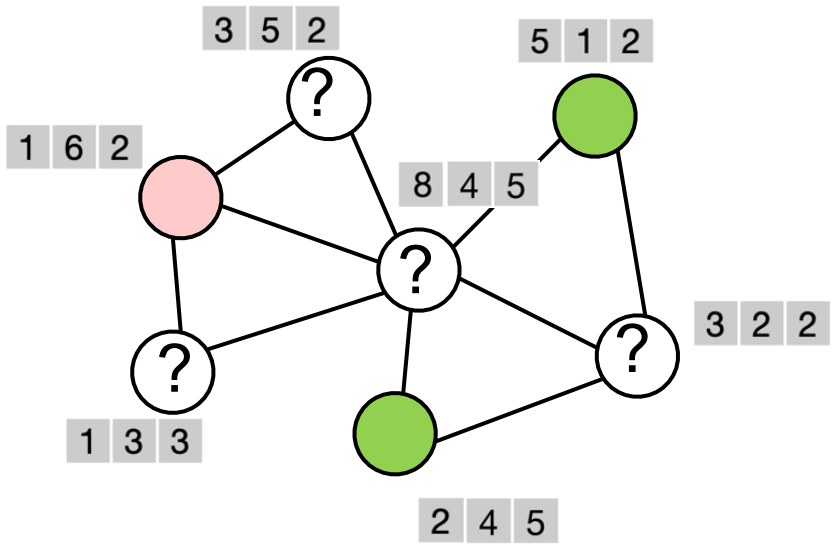
$$\Rightarrow \boldsymbol{Z} = \boldsymbol{H}^K$$

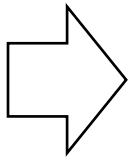
- Model training
  - The common setting is to have an end to end training framework with a supervised task
  - That is, define a loss function over $\boldsymbol{Z}$

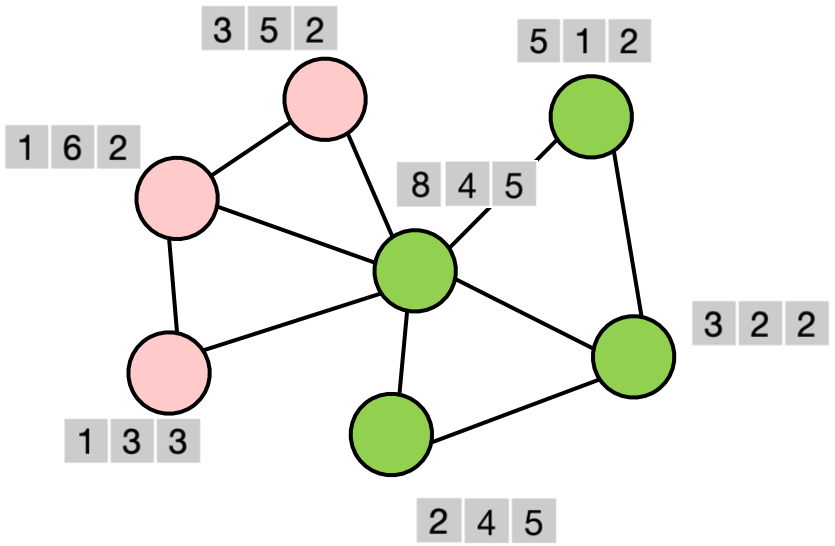
Revisiting Semi-Supervised Learning on Graphs

- W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS'20. https://arxiv.org/abs/2005.11079
- Code & data for Grand: https://github.com/Grand20/grand

Semi-Supervised Learning on Graphs







Input: a partially labeled & attributed graph

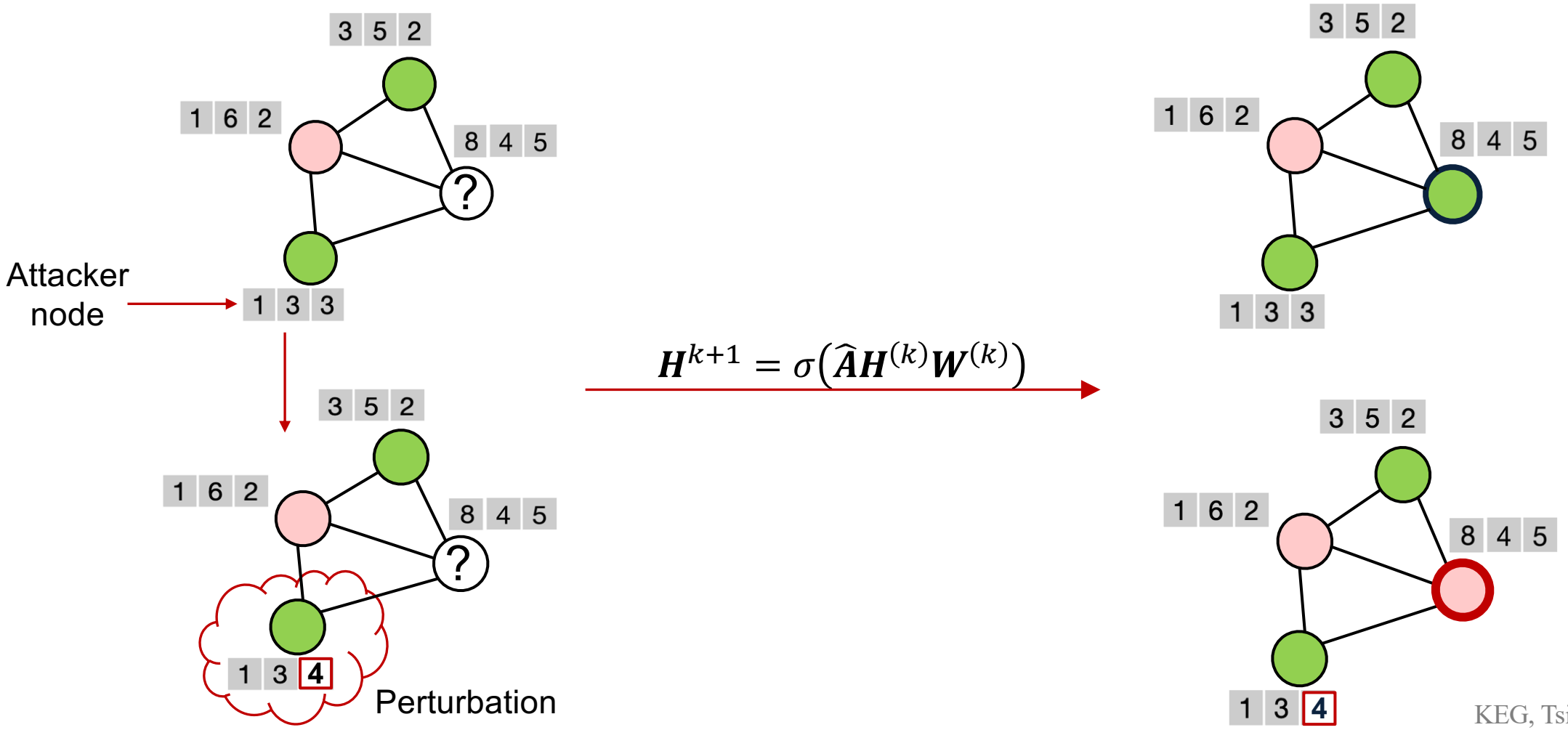
Output: infer the labels of unlabeled nodes

Graph Neural Networks

$$\boldsymbol{H}^{k+1} = \sigma\left(\widehat{\boldsymbol{A}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\right)$$

a deterministic propagation

1. Each node is highly dependent with its neighbors, making GNNs **non-robust** to noises



Graph Neural Networks

$$\boldsymbol{H}^{k+1} = \sigma\left(\widehat{\boldsymbol{A}}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\right)$$

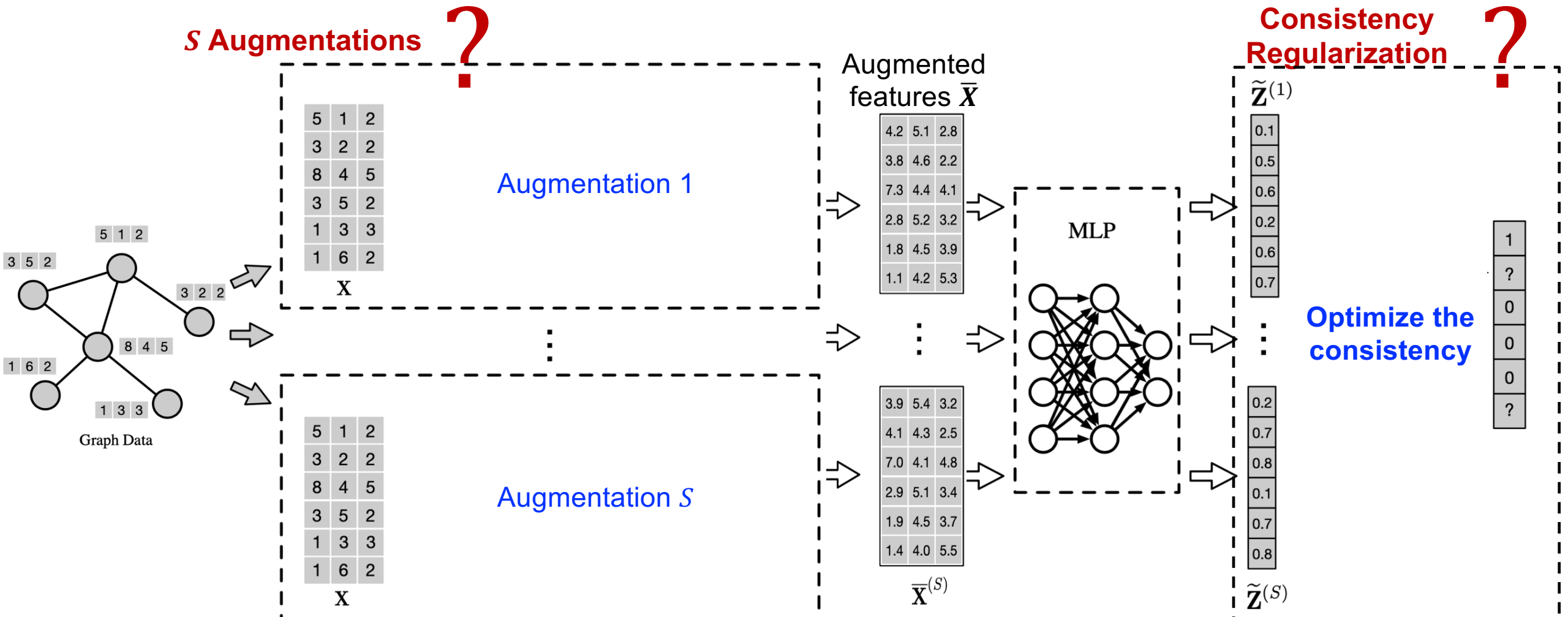
feature **propagation**
is coupled with
non-linear transformation

1. Each node is highly dependent with its neighbors, making GNNs **non-robust** to noises

2. Stacking many GNN layers may cause **over-fitting** & **over-smoothing.**

Graph Random Neural Networks (GRAND)

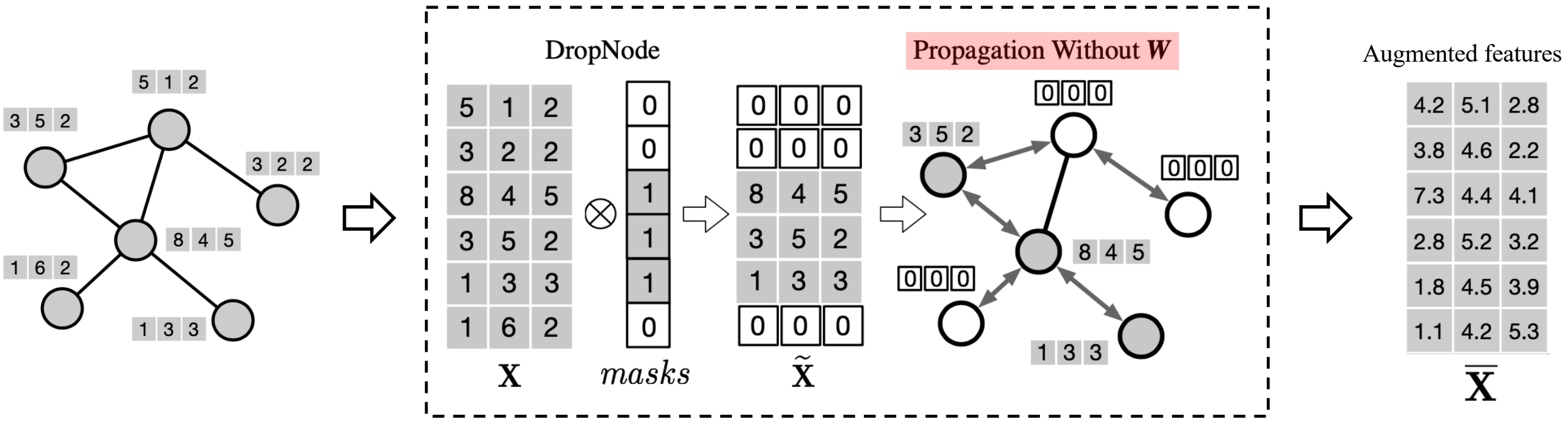
- Consistency Regularized Training:
  - Generates S data augmentations of the graph
  - Optimizing the consistency among S augmentations of the graph.



GRAND: Random Propagation for Graph Data Augmentation

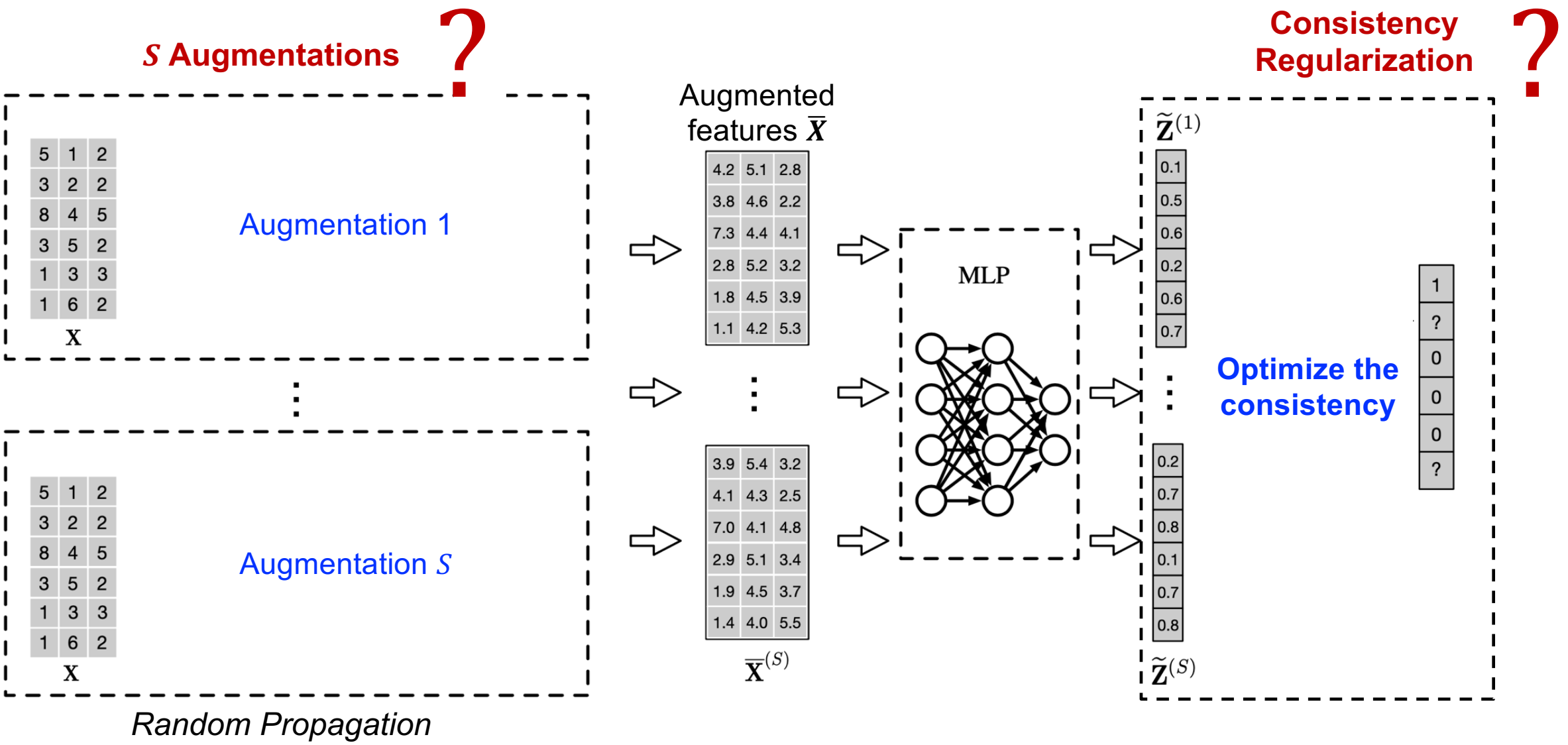
- **Random Propagation** (DropNode + Propagation):
  - Each node is enabled to be not sensitive to specific neighborhoods.
  - Decouple feature propagation from feature transformation.

Random Propagation for Augmentation



$$\overline{\mathbf{X}} = \overline{\mathbf{A}}\tilde{\mathbf{X}}$$

Graph Random Neural Networks (GRAND)



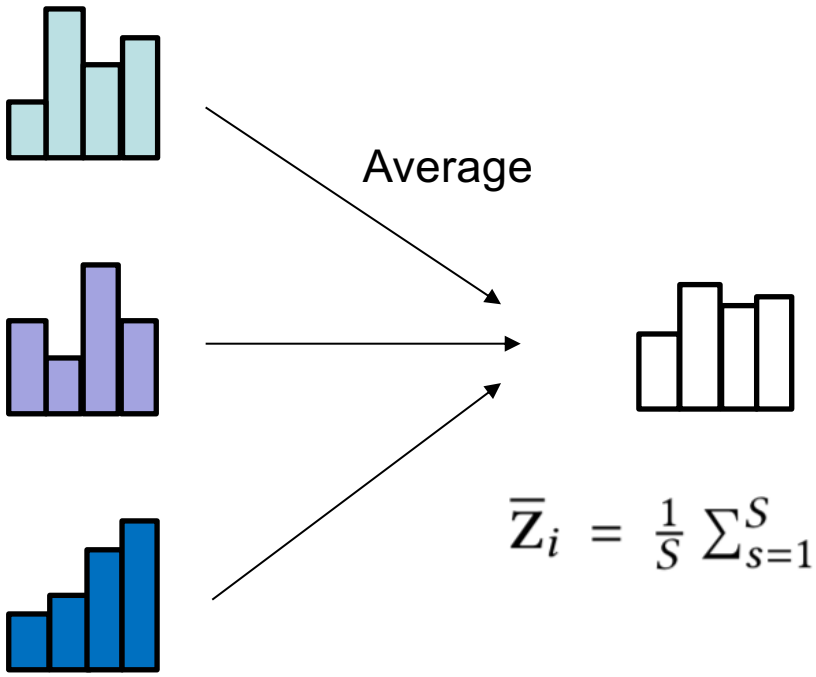
GRAND: Consistency Regularization

$$\mathcal{L}_{sup} = -\frac{1}{S}\sum_{s=1}^{S}\sum_{i=0}^{m-1}\mathbf{Y}_i^{\top}\log\widetilde{\mathbf{Z}}_i^{(s)}$$

$$\mathcal{L}_{con} = \frac{1}{S}\sum_{s=1}^{S}\sum_{i=0}^{n-1}\mathcal{D}(\overline{\mathbf{Z}}_i', \widetilde{\mathbf{Z}}_i^{(s)})$$

$$\mathcal{L} = \mathcal{L}_{sup} + \lambda\mathcal{L}_{con}$$

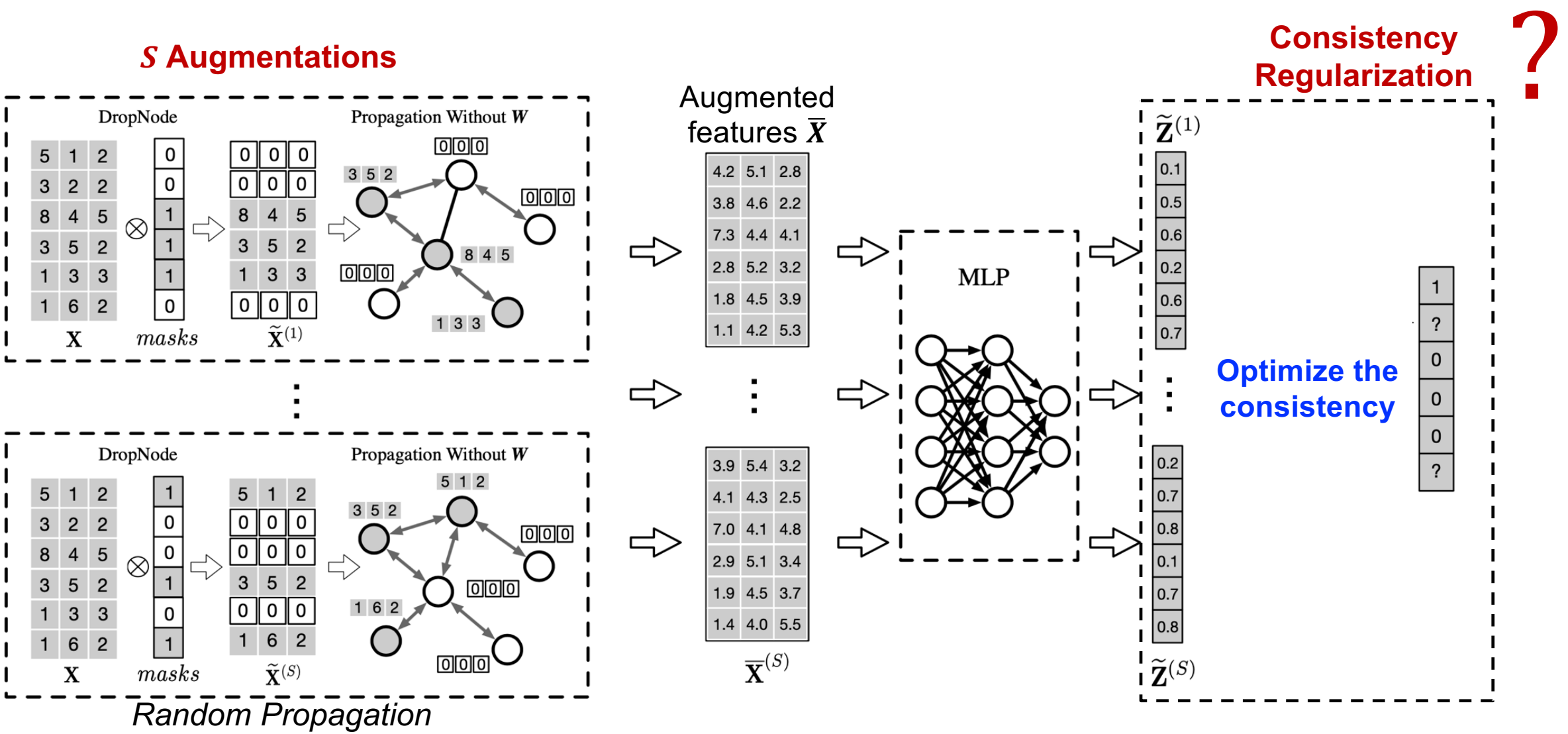
Distributions of a node after augmentations



$$\overline{\mathbf{Z}}_i = \frac{1}{S}\sum_{s=1}^{S}\widetilde{\mathbf{Z}}_i^{(s)}$$

$$\overline{\mathbf{Z}}'_{ik} = \overline{\mathbf{Z}}_{ik}^{\frac{1}{T}} \Big/ \sum_{j=0}^{C-1}\overline{\mathbf{Z}}_{ij}^{\frac{1}{T}}$$

Graph Random Neural Networks (GRAND)



Consistency Regularized Training Algorithm

Input:

Adjacency matrix $\hat{\mathbf{A}}$, feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, times of augmentations in each epoch S, DropNode probability δ.

Output:

Prediction $\mathbf{Z}$.

while not convergence **do**
  for $s = 1 : S$ **do**
    Apply DropNode via Algorithm 1: $\widetilde{\mathbf{X}}^{(s)} \sim \text{DropNode}(\mathbf{X}, \delta)$.
    Perform propagation: $\overline{\mathbf{X}}^{(s)} = \frac{1}{K+1} \sum_{k=0}^{K} \hat{\mathbf{A}}^k \widetilde{\mathbf{X}}^{(s)}$.
    Predict class distribution using MLP: $\widetilde{\mathbf{Z}}^{(s)} = P(\mathbf{Y}|\overline{\mathbf{X}}^{(s)}; \Theta)$.
  end for
  Compute supervised classification loss $\mathcal{L}_{sup}$ via Eq. 4 and consistency regularization loss via Eq. 6.
  Update the parameters Θ by gradients descending:

$$\nabla_{\Theta} \mathcal{L}_{sup} + \lambda \mathcal{L}_{con}$$

end while
Output prediction $\mathbf{Z}$ via Eq. 8.

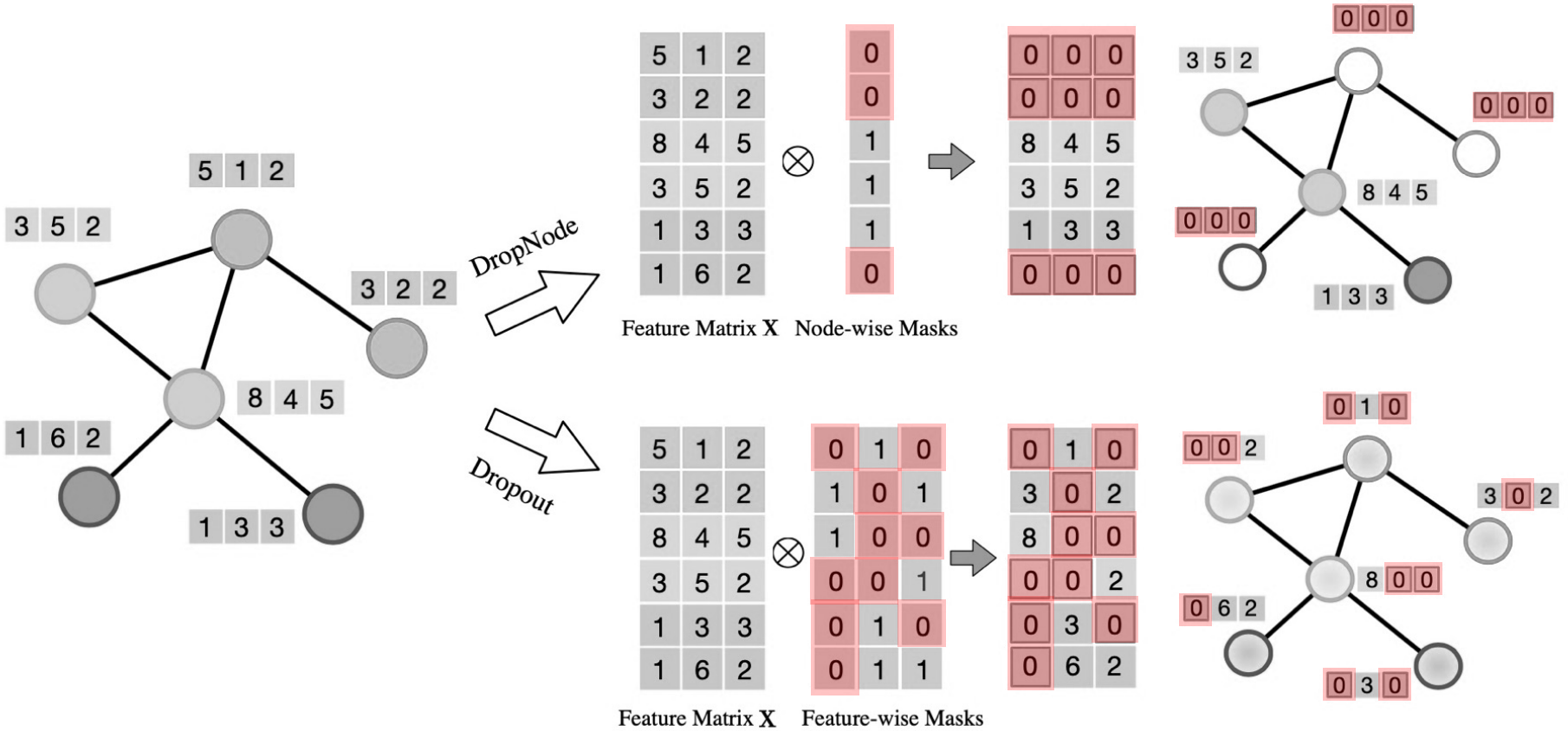
Generate S Augmentations

Consistency Regularization

- W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS'20. https://arxiv.org/abs/2005.11079
- Code & data for Grand: https://github.com/Grand20/grand

GRAND: DropNode vs Dropout

- Dropout drops each element in $\boldsymbol{X}$ independently
- DropNode drops the entire features of selected nodes, i.e., the row vectors of $\boldsymbol{X}$, randomly
  - Theoretically, Dropout is an adaptive L_2 regularization.



Graph Random Neural Networks (GRAND)

- With Consistency Regularization Loss:
  - Random propagation can enforce the consistency of the classification confidence between each node and its all multi-hop neighborhoods.

$$\mathbb{E}_\epsilon\left(\mathcal{L}_{con}\right) \approx \mathcal{R}^c(\mathbf{W}) = \sum_{i=0}^{n-1} z_i^2(1-z_i)^2 \mathrm{Var}_\epsilon\left(\overline{\mathbf{A}}_i\widetilde{\mathbf{X}}\cdot\mathbf{W}\right)$$

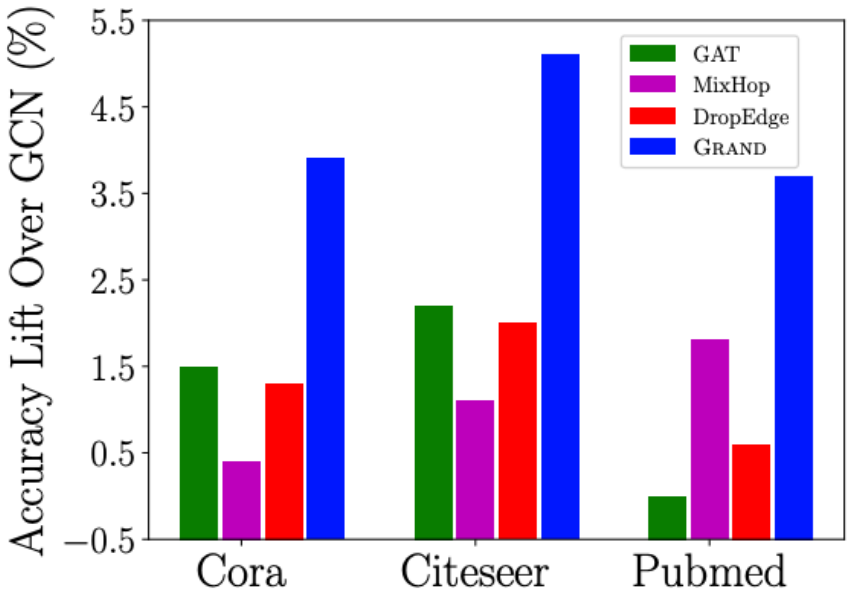
$$\mathcal{R}^c_{DN}(\mathbf{W}) = \frac{\delta}{1-\delta}\sum_{j=0}^{n-1}\left[(\mathbf{X}_j\cdot\mathbf{W})^2\sum_{i=0}^{n-1}(\overline{\mathbf{A}}_{ij})^2 z_i^2(1-z_i)^2\right]$$

$$\mathcal{R}^c_{Do}(\mathbf{W}) = \frac{\delta}{1-\delta}\sum_{h=0}^{d-1}\mathbf{W}_h^2\sum_{j=0}^{n-1}\left[\mathbf{X}_{jh}^2\sum_{i=0}^{n-1} z_i^2(1-z_i)^2(\overline{\mathbf{A}}_{ij})^2\right]$$

- With Supervised Cross-Entropy Loss:
  - Random propagation can enforce the consistency of the classification confidence between each node and its labeled multi-hop neighborhoods.

Results

	Method	Cora	Citeseer	Pubmed
GCNs	GCN [19]	81.5	70.3	79.0
	GAT [32]	83.0±0.7	72.5±0.7	79.0±0.3
	APPNP [20]	83.8±0.3	71.6± 0.5	79.7 ± 0.3
	Graph U-Net [11]	84.4±0.6	73.2±0.5	79.6±0.2
	SGC [36]	81.0 ±0.0	71.9 ± 0.1	78.9 ± 0.0
	MixHop [1]	81.9± 0.4	71.4±0.8	80.8±0.6
	GMNN [28]	83.7	72.9	81.8
	GraphNAS [12]	84.2±1.0	73.1±0.9	79.6±0.4
Sampling GCNs	GraphSAGE [16]	78.9±0.8	67.4±0.7	77.8±0.6
	FastGCN [7]	81.4±0.5	68.8±0.9	77.6±0.5
Reg. GCNs	VBAT [10]	83.6±0.5	74.0±0.6	79.9±0.4
	G^3NN [24]	82.5±0.2	74.4±0.3	77.9 ±0.4
	GraphMix [33]	83.9±0.6	74.5±0.6	81.0±0.6
	DropEdge [29]	82.8	72.3	79.6
	GRAND	**85.4±0.4**	**75.4±0.4**	**82.7±0.6**



Instead of the marginal improvements by conventional GNN baselines over GCN, ***GRAND*** *achieves* ***much more significant*** *performance lift in* ***all three datasets***!

- W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS'20. https://arxiv.org/abs/2005.11079
- Code & data for Grand: https://github.com/Grand20/grand

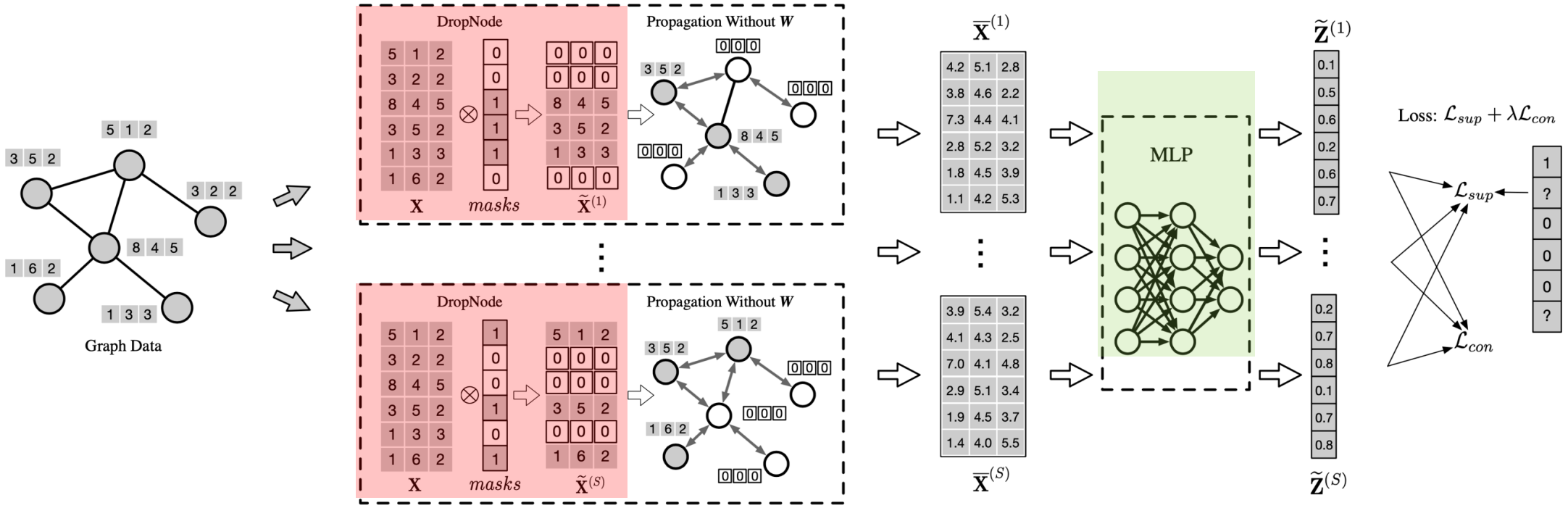
Results

Table 5: Results on large datasets.

Method	Cora Full	Coauthor CS	Coauthor Physics	Amazon Computer	Amazon Photo	Citation CS
GCN	62.2 ± 0.6	91.1 ± 0.5	92.8 ± 1.0	82.6 ± 2.4	91.2 ± 1.2	49.9 ± 2.0
GAT	51.9 ± 1.5	90.5 ± 0.6	92.5 ± 0.9	78.0 ± 19.0	85.7 ± 20.3	49.6 ± 1.7
GRAND	**63.5 ±0.6**	**92.9 ± 0.5**	**94.6 ± 0.5**	**85.7 ± 1.8**	**92.5 ± 1.7**	**52.8 ± 1.2**

More experiments on larger graph datasets

Results



GRAND_dropout	84.9±0.4	75.0±0.3	81.7±1.0
GRAND_GCN	84.5±0.3	74.2±0.3	80.0±0.3
GRAND_GAT	84.3±0.4	73.2± 0.4	79.2±0.6
GRAND	**85.4±0.4**	**75.4±0.4**	**82.7±0.6**

Evaluation of the design choices in GRAND

Results

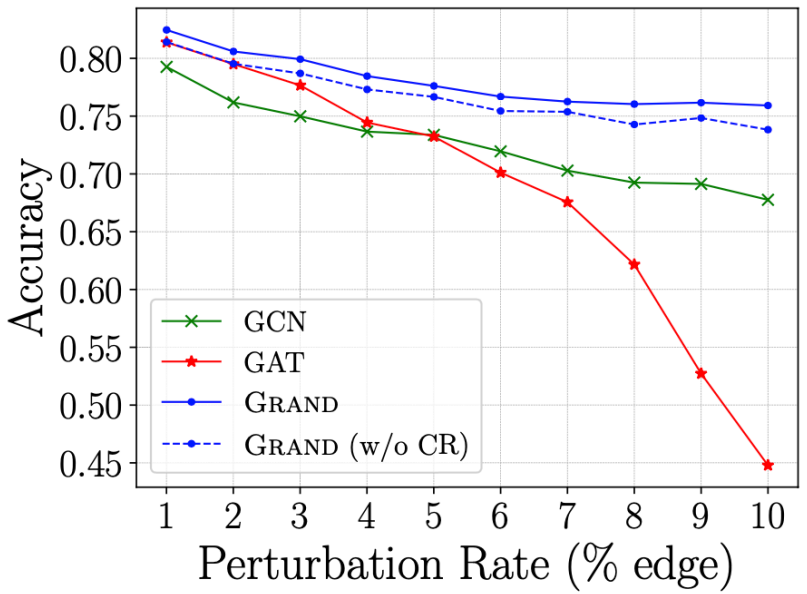
Method	Cora	Citeseer	Pubmed
GCN [19]	81.5	70.3	79.0
GAT [32]	83.0±0.7	72.5±0.7	79.0±0.3
APPNP [20]	83.8±0.3	71.6± 0.5	79.7 ± 0.3
Graph U-Net [11]	84.4±0.6	73.2±0.5	79.6±0.2
SGC [36]	81.0 ±0.0	71.9 ± 0.1	78.9 ± 0.0
MixHop [1]	81.9± 0.4	71.4±0.8	80.8±0.6
GMNN [28]	83.7	72.9	81.8
GraphNAS [12]	84.2±1.0	73.1±0.9	79.6±0.4
DropEdge [29]	82.8	72.3	79.6
w/o CR	84.4±0.5	73.1±0.6	80.9±0.8
w/o mDN	84.7±0.4	74.8±0.4	81.0±1.1
w/o sharpening	84.6±0.4	72.2±0.6	81.6±0.8
w/o CR & DN	83.2±0.5	70.3±0.6	78.5±1.4

Ablation Study

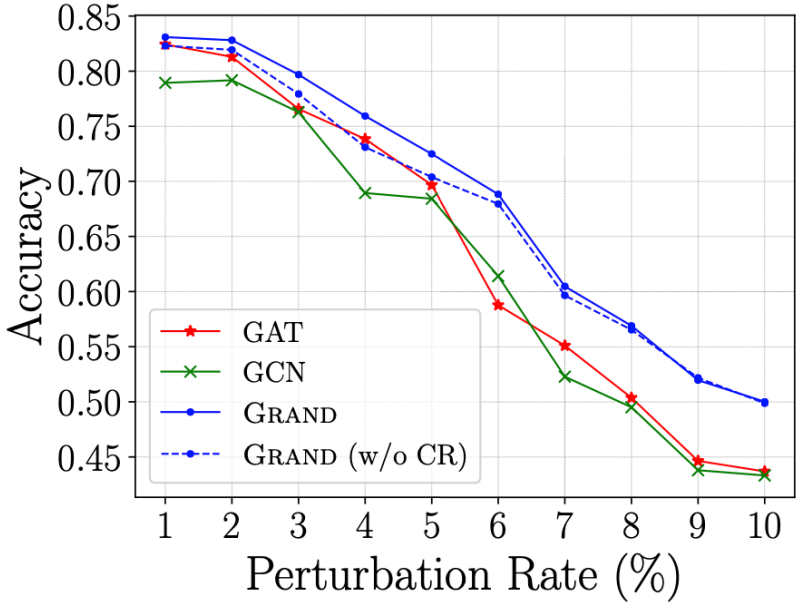
1. Each of the designed components contributes to the success of GRAND.
2. GRAND w/o consistency regularization outperforms almost *all 8 non-regularization based GCNs* & *DropEdge*

Random Propagation
vs.
Feature Propagation & Non-Linear Transformation

Results



(a) Random Attack



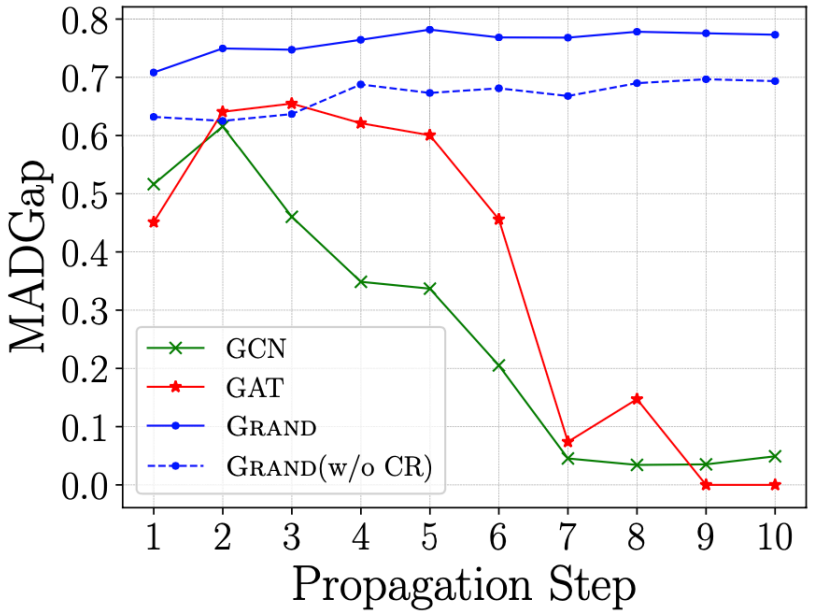
(b) Metattack

Robustness

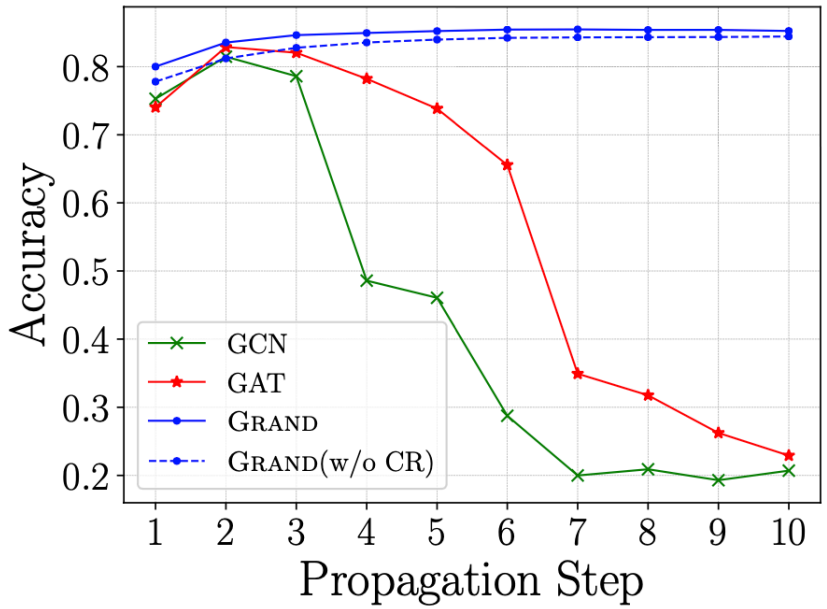
1. GRAND (with or w/o) consistency regularization is more robust than GCN and GAT.

Random Propagation vs. Feature Propagation & Non-Linear Transformation

Results



(a) MADGap



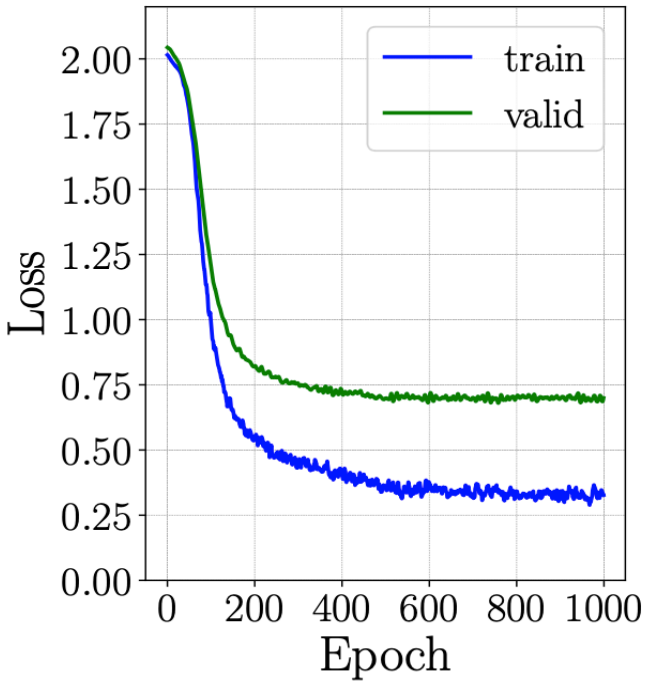
(b) Classification Results

Over-Smoothing

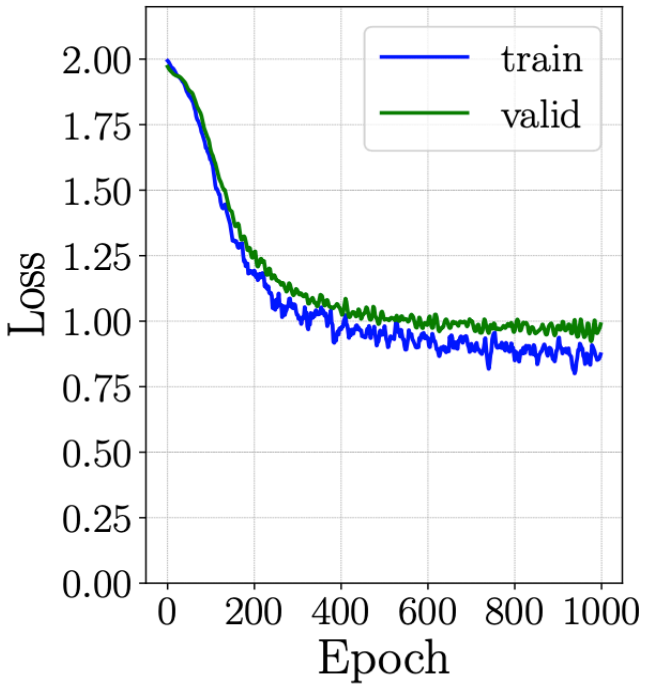
1. GRAND is powerful to relieve over-smoothing, while GCN & GAT are vulnerable to it

Random Propagation vs. Feature Propagation & Non-Linear Transformation

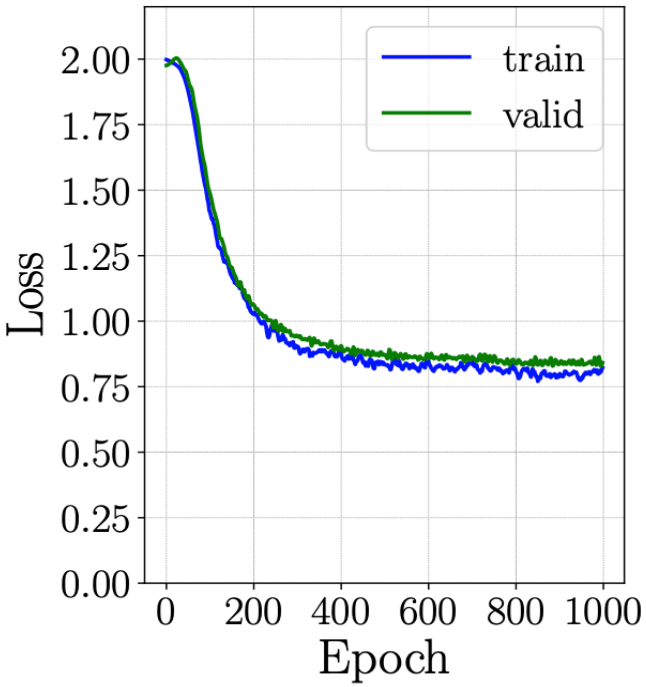
Results



(a) Without CR and RP



(b) Without CR



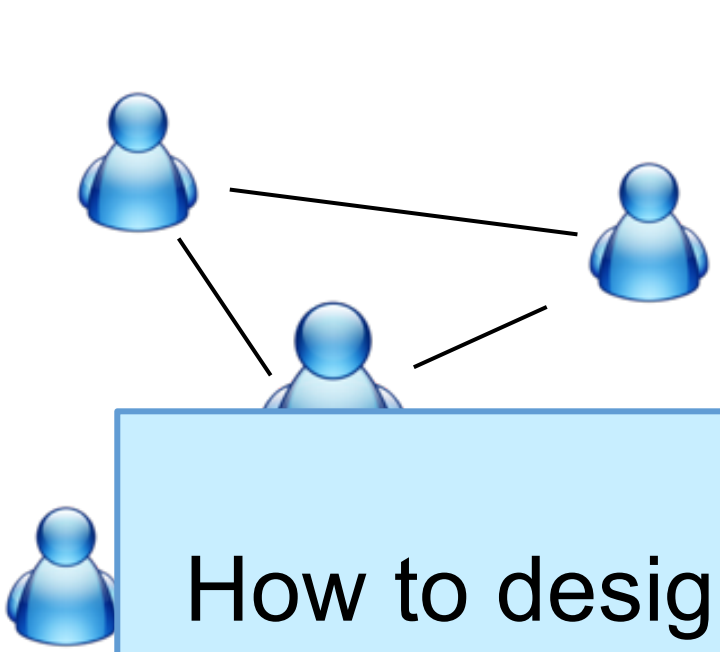
(c) GRAND

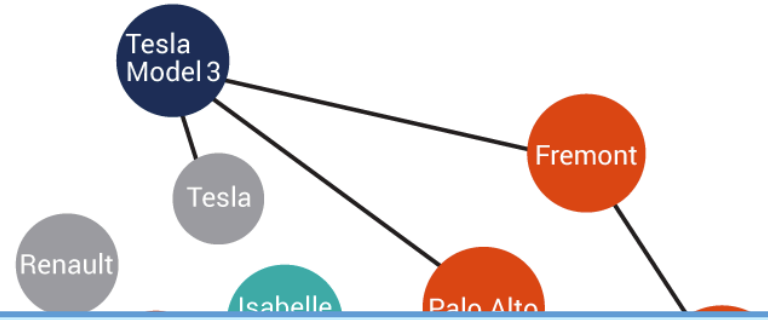
Generalization

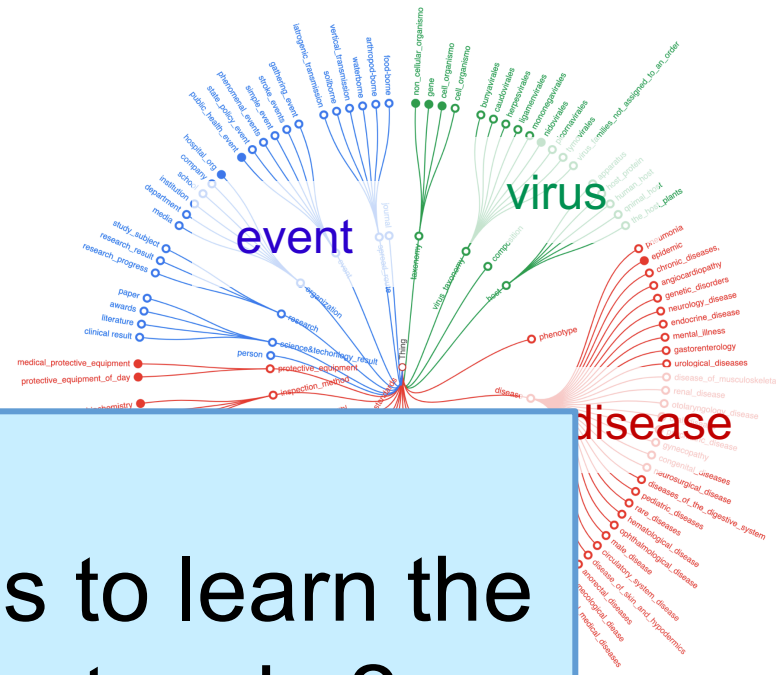
1. Both the random propagation and consistency regularization improve GRAND's generalization capability

GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training

Networked Data







Question:

How to design machine learning models to learn the universal structural patterns across networks?

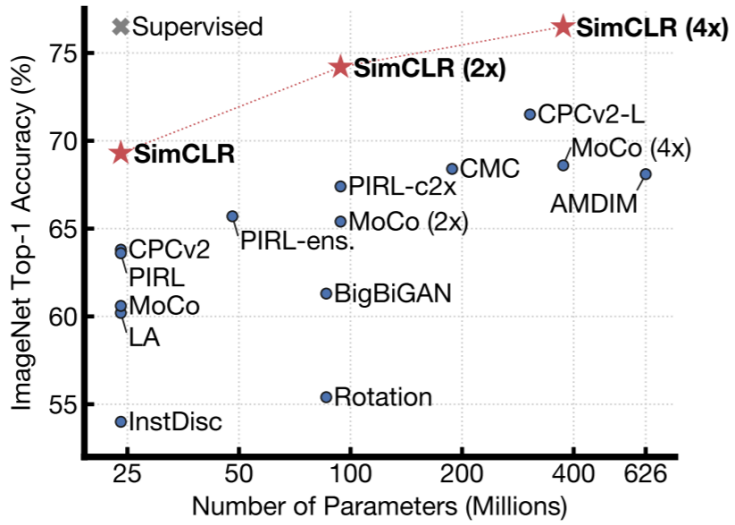
Social Network

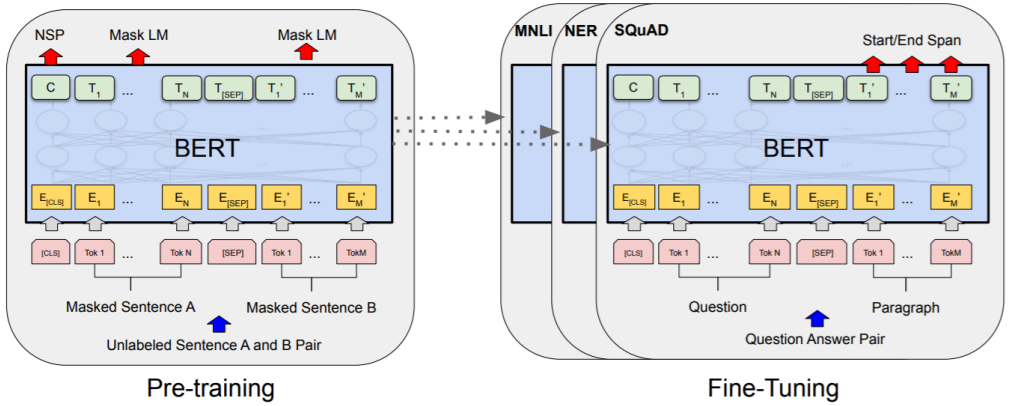
Knowledge Graph

COVID Graph

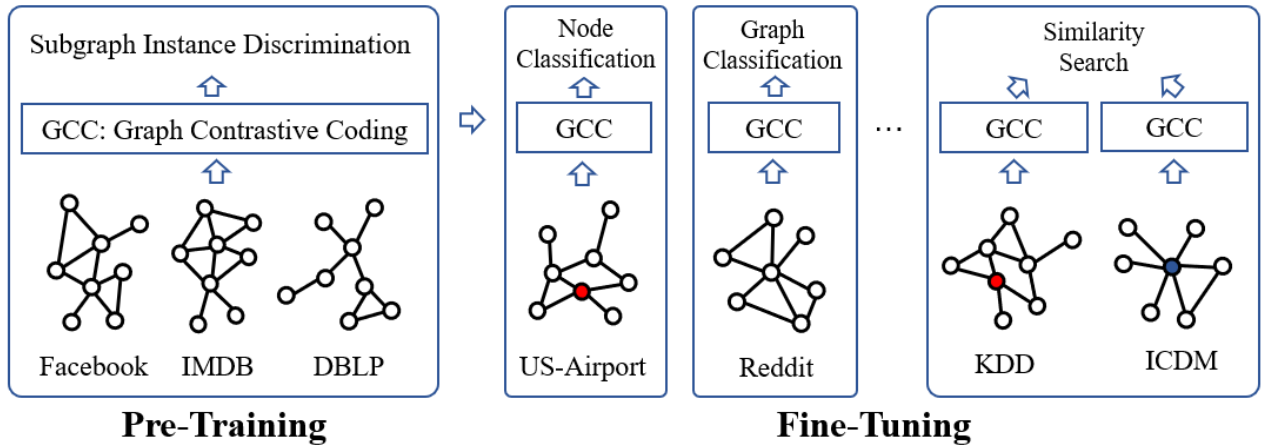
Pre-training and Fine-tuning

CV: MoCo, SimCLR





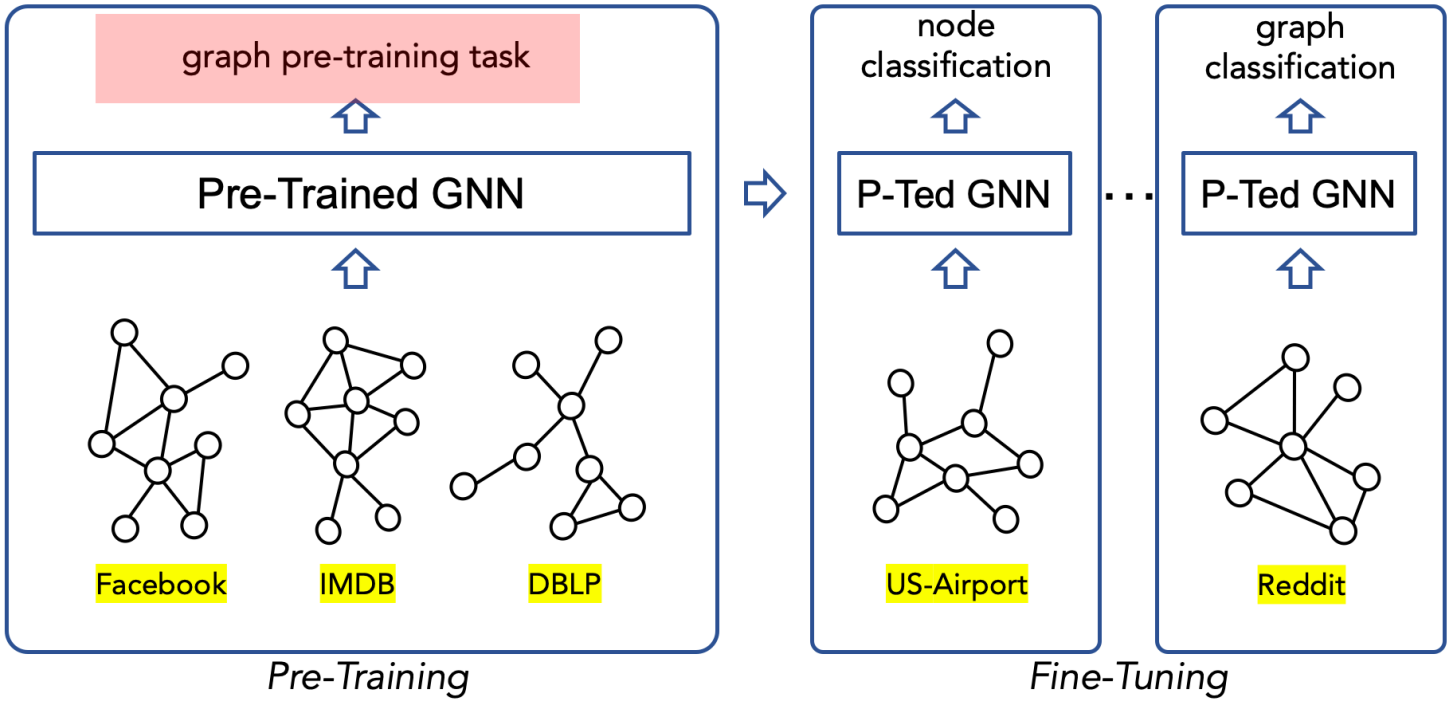
NLP: BERT



Graph Learning
GCC

GNN Pre-Training

- Graph pre-training setting:
  - To pre-train from some graphs
  - To fine-tune for unseen tasks on unseen graphs



- How to do this?
  - Model level: GNNs?
  - Pre-training task: **self-supervised** tasks across graphs?

GNN Pre-Training across Networks

- What are the requirements?
  - **structural similarity**, it maps vertices with similar local network topologies close to each other in the vector space
  - **transferability**, it is compatible with vertices and graphs unseen by the pre-training algorithm

1. Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2. Code & Data for GCC: https://github.com/THUDM/GCC

GNN Pre-Training across Networks

- The Idea: Contrastive learning
  - **pre-training task**: instance discrimination
  - **InfoNCE objective**: output instance representations that are capable of capturing the similarities between instances

$$\mathcal{L} = -\log \frac{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}_+ / \tau\right)}{\sum_{i=0}^{K} \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_i / \tau\right)}$$

- query instance x^q
- query $\boldsymbol{q}$ (embedding of x^q), i.e., $\boldsymbol{q} = f(x^q)$
- dictionary of keys $\{\boldsymbol{k}_0, \boldsymbol{k}_1, \cdots, \boldsymbol{k}_K\}$
- key $\boldsymbol{k} = f(x^k)$

- Contrastive learning for graphs?
  - Q1: How to define instances in graphs?
  - Q2: How to define (dis) similar instance pairs in and across graphs?
  - Q3: What are the proper graph encoders?

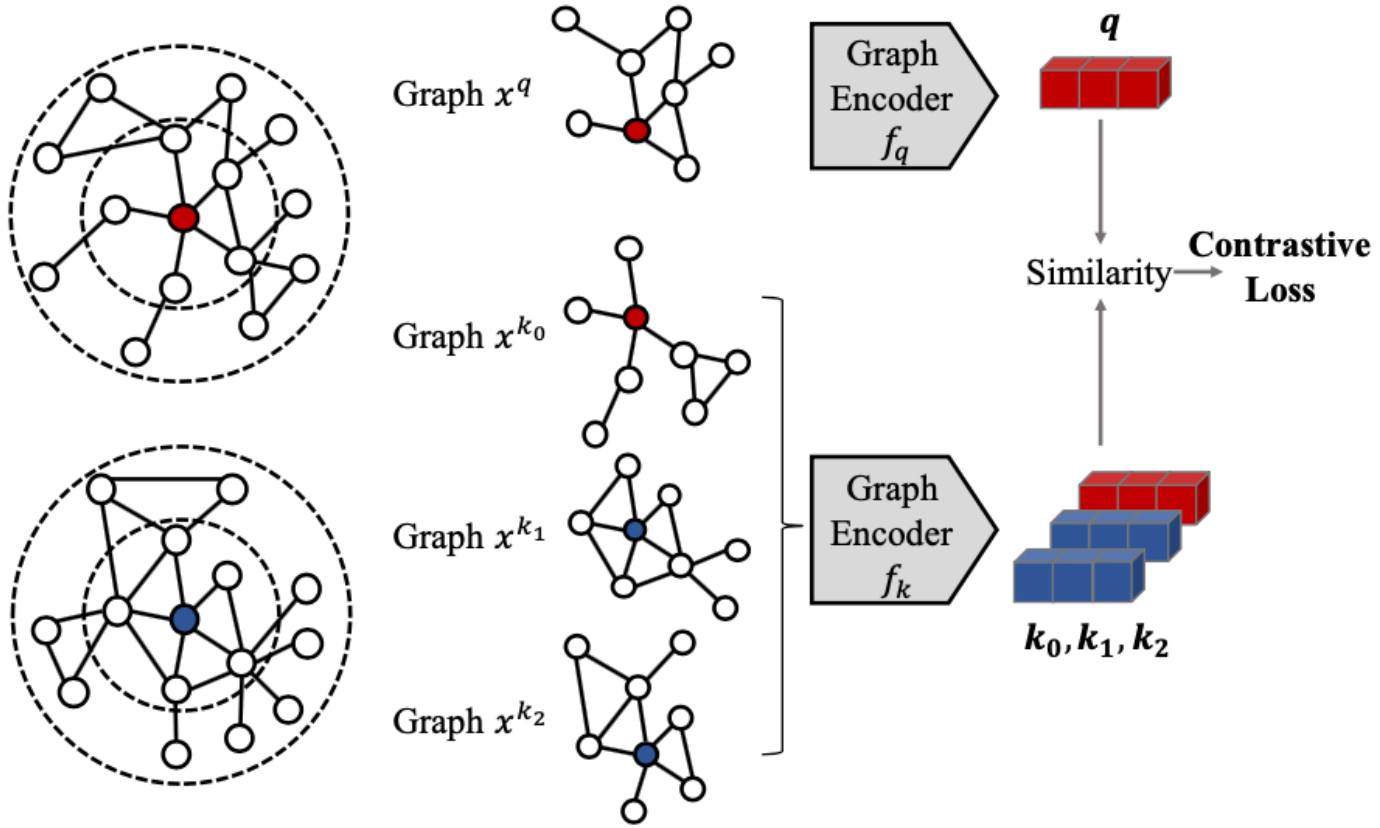
1. Z. Wu, Y. Xiong, S. X Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In CVPR '18.
2. K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In CVPR '20.

Graph Contrastive Coding (GCC)

- Contrastive learning for graphs
  - Q1: How to define instances in graphs?
  - Q2: How to define (dis) similar instance pairs in and across graphs?
  - Q3: What are the proper graph

$$\mathcal{L} = -\log \frac{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}_+ / \tau\right)}{\sum_{i=0}^{K} \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_i / \tau\right)}$$

Subgraph instance discrimination



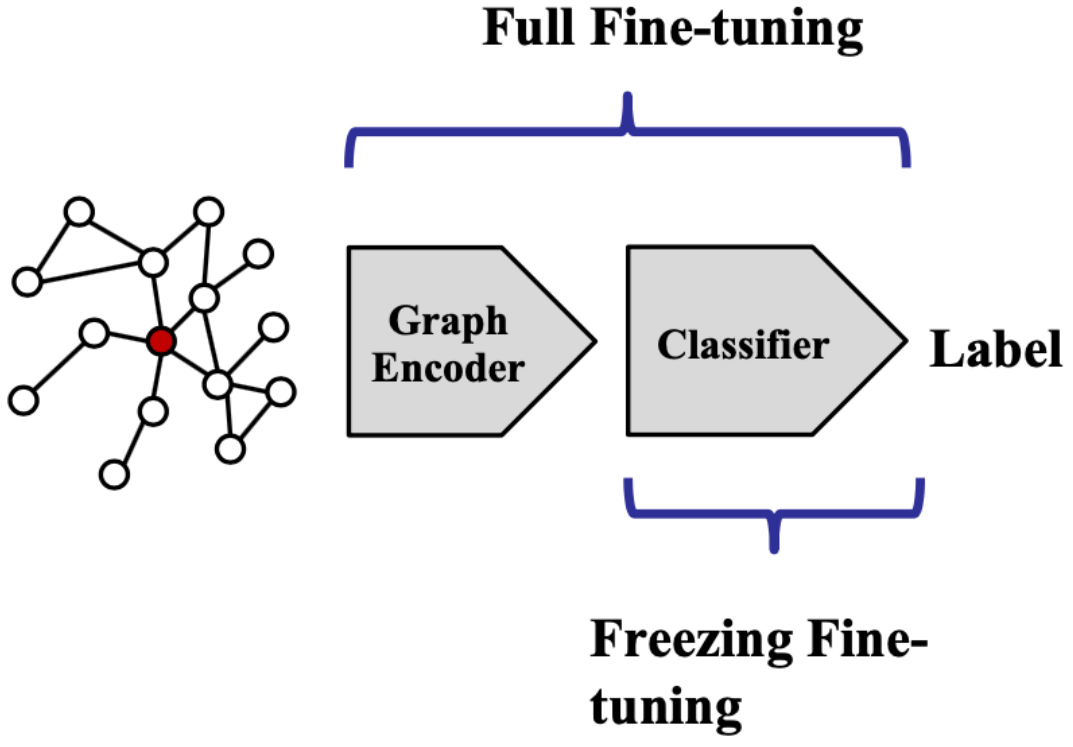
GCC Pre-Training / Fine-Tuning

- pre-train on six graphs

Code & Data for GCC:
https://github.com/THUDM/GCC

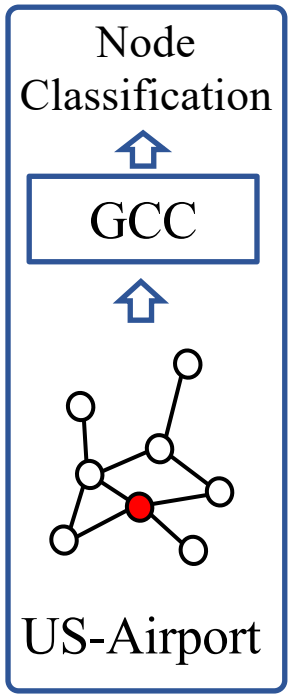
Dataset	Academia	DBLP (SNAP)	DBLP (NetRep)	IMDB	Facebook	LiveJournal
$\|V\|$	137,969	317,080	540,486	896,305	3,097,165	4,843,953
$\|E\|$	739,384	2,099,732	30,491,458	7,564,894	47,334,788	85,691,368

- fine-tune on **different** graphs
  - US-Airport & AMiner academic graph
    - Node classification
  - COLLAB, RDT-B, RDT-M, & IMDB-B/M
    - Graph classification
  - AMiner academic graph
    - Similarity search
- The base GNN
  - Graph Isomorphism Network (GIN)



Result 1: Node Classification

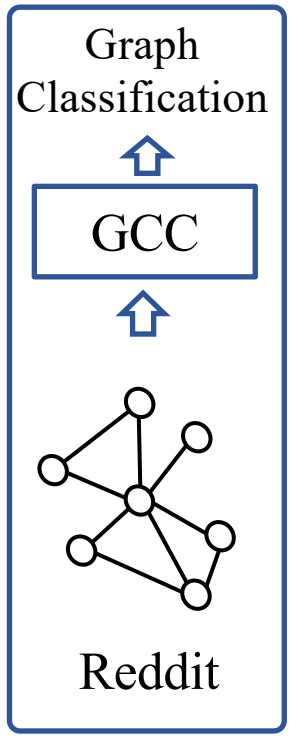
- Setup
  - US-Airport
  - AMiner academic graph



Datasets	US-Airport	H-index
$\|V\|$	1,190	5,000
$\|E\|$	13,599	44,020
ProNE	62.3	69.1
GraphWave	60.2	70.3
Struc2vec	**66.2**	> 1 Day
GCC (E2E, freeze)	64.8	**78.3**
GCC (MoCo, freeze)	65.6	75.2
GCC (rand, full)	64.2	76.9
GCC (E2E, full)	**68.3**	80.5
GCC (MoCo, full)	67.2	**80.6**

Result 2: Graph Classification

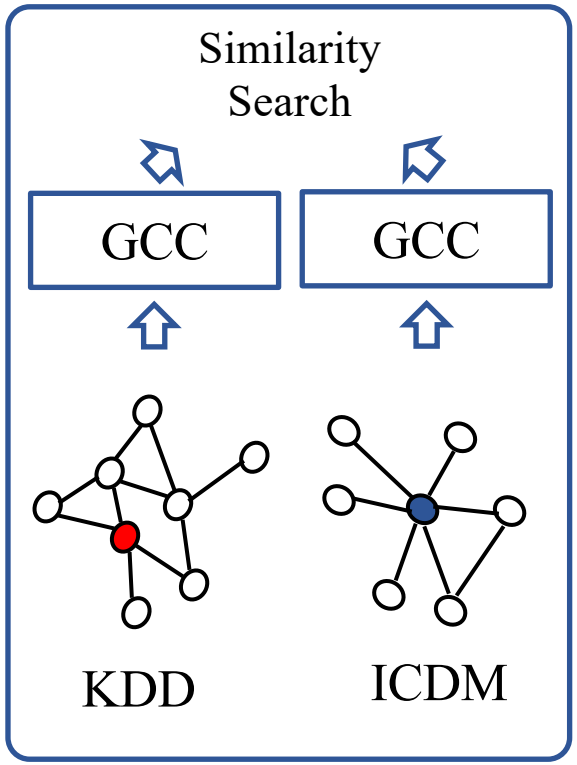
- Setup
  - COLLAB, RDT-B, RDT-M, & IMDB-B, IMDB-M



Datasets	IMDB-B	IMDB-M	COLLAB	RDT-B	RDT-M
# graphs	1,000	1,500	5,000	2,000	5,000
# classes	2	3	3	2	5
Avg. # nodes	19.8	13.0	74.5	429.6	508.5
DGK	67.0	44.6	73.1	78.0	41.3
graph2vec	71.1	50.4	–	75.8	47.9
InfoGraph	**73.0**	**49.7**	–	82.5	53.5
GCC (E2E, freeze)	71.7	49.3	74.7	87.5	52.6
GCC (MoCo, freeze)	72.0	49.4	**78.9**	**89.8**	**53.7**
DGCNN	70.0	47.8	73.7	–	–
GIN	**75.6**	**51.5**	80.2	**89.4**	**54.5**
GCC (rand, full)	**75.6**	50.9	79.4	87.8	52.1
GCC (E2E, full)	70.8	48.5	79.0	86.4	47.4
GCC (MoCo, full)	73.8	50.3	**81.1**	87.6	53.0

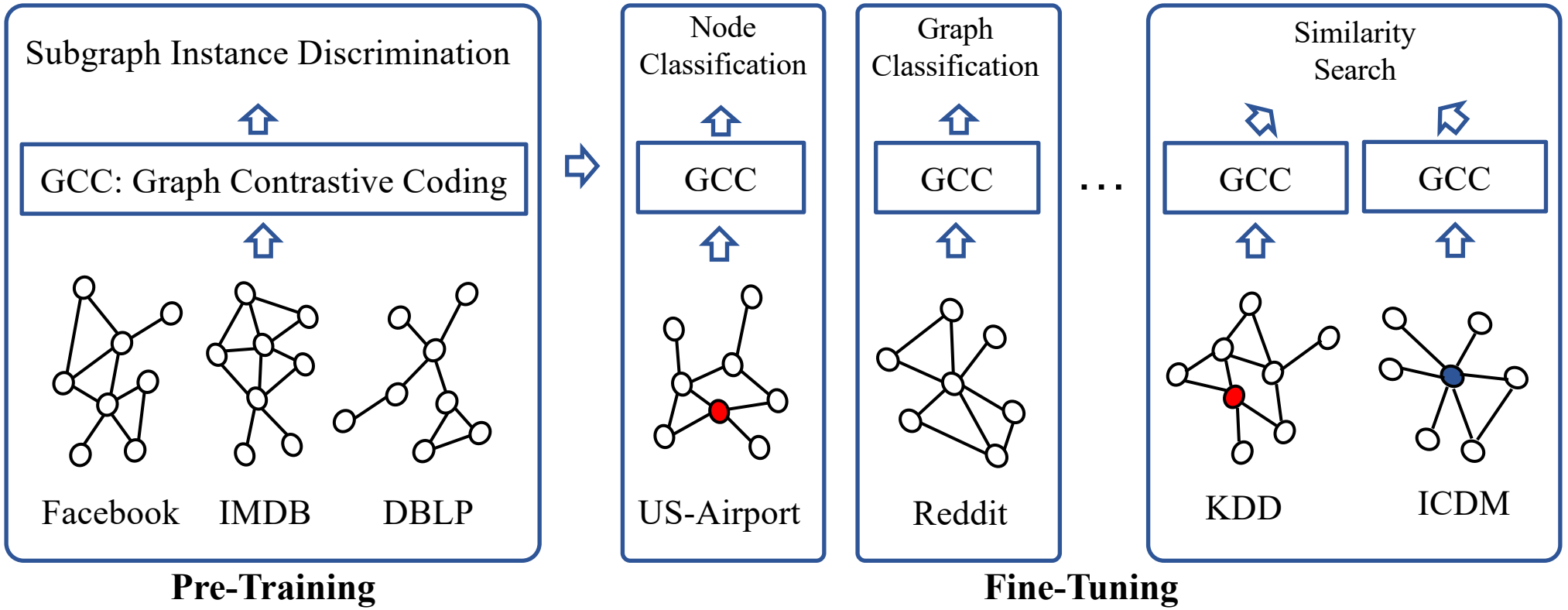
Result 3: Top-k Similarity Search

- Setup
  - AMiner academic graph



	KDD-ICDM		SIGIR-CIKM		SIGMOD-ICDE	
$\|V\|$	2,867	2,607	2,851	3,548	2,616	2,559
$\|E\|$	7,637	4,774	6,354	7,076	8,304	6,668
# groud truth		697		874		898
k	20	40	20	40	20	40
Random	0.0198	0.0566	0.0223	0.0447	0.0221	0.0521
RolX	0.0779	0.1288	0.0548	0.0984	0.0776	0.1309
Panther++	0.0892	0.1558	**0.0782**	0.1185	0.0921	0.1320
GraphWave	0.0846	**0.1693**	0.0549	0.0995	**0.0947**	**0.1470**
GCC (E2E)	**0.1047**	0.1564	0.0549	**0.1247**	0.0835	0.1336
GCC (MoCo)	0.0904	0.1521	0.0652	0.1178	0.0846	0.1425

GNN Pre-Training



1. Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2. Code & Data for GCC: https://github.com/THUDM/GCC

GRL: NE&GNN

Network Embedding ⇔ Matrix Factorization

Graph Neural Networks ⇨ GNN Pre-Training

Graph Data & Benchmarks

OPEN GRAPH BENCHMARK

https://alchemy.tencent.com/

Microsoft Academic

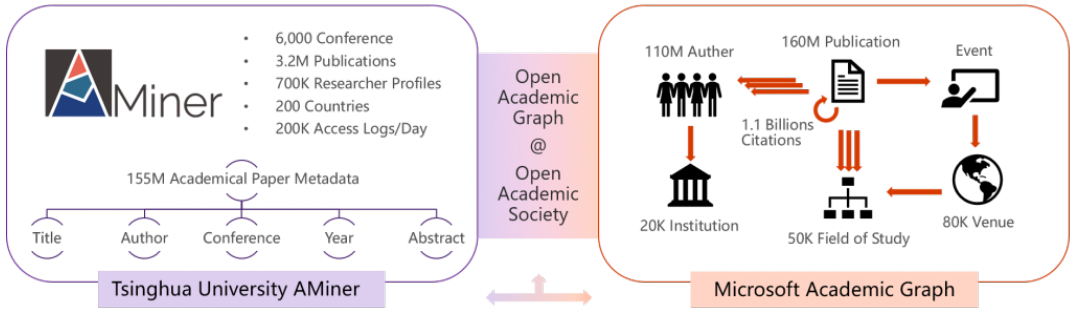
AMiner

CogDL

https://github.com/thudm/cogdl

OAG: Open Academic Graph

https://www.openacademic.ai/oag/



Data set	#Pairs/Venues	Date
Linking relations	29,841	2018.12
AMiner venues	69,397	2018.07
MAG venues	52,678	2018.11

Table 1: statistics of OAG venue data

Data set	#Pairs/Papers	Date
Linking relations	91,137,597	2018.12
AMiner papers	172,209,563	2019.01
MAG papers	208,915,369	2018.11

Table 2: statistics of OAG paper data

Data set	#Pairs/Authors	Date
Linking relations	1,717,680	2019.01
AMiner authors	113,171,945	2018.07
MAG authors	253,144,301	2018.11

Open Academic Graph

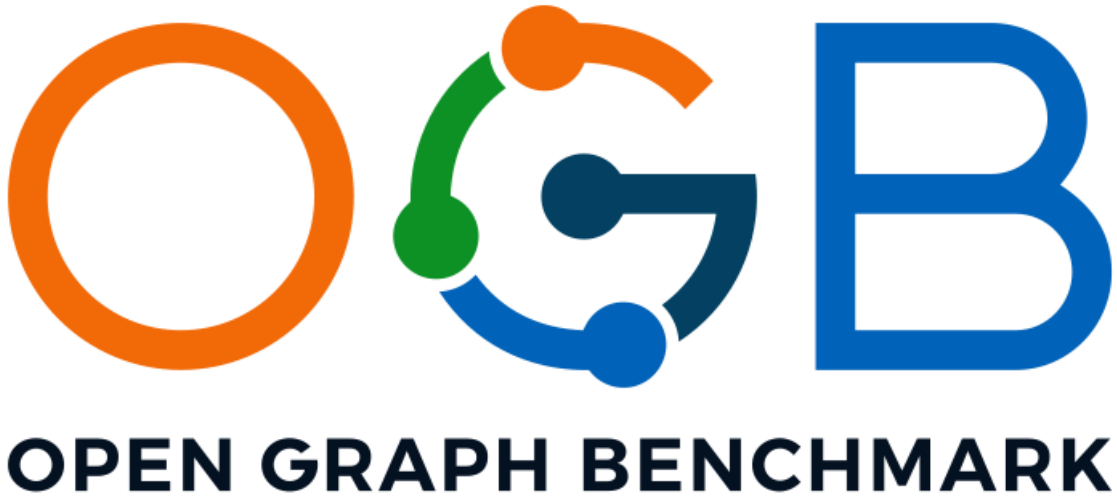
Open Academic Graph (OAG) is a large knowledge graph unifying two billion-scale academic graphs: Microsoft Academic Graph (MAG) and AMiner. In mid 2017, we published OAG v1, which contains 166,192,182 papers from MAG and 154,771,162 papers from AMiner (see below) and generated 64,639,608 linking (matching) relations between the two graphs. This time, in OAG v2, author, venue and newer publication data and the corresponding matchings are available.

Overview of OAG v2

The statistics of OAG v2 is listed as the three tables below. The two large graphs are both evolving and we take MAG November 2018 snapshot and AMiner July 2018 or January 2019 snapshot for this version.

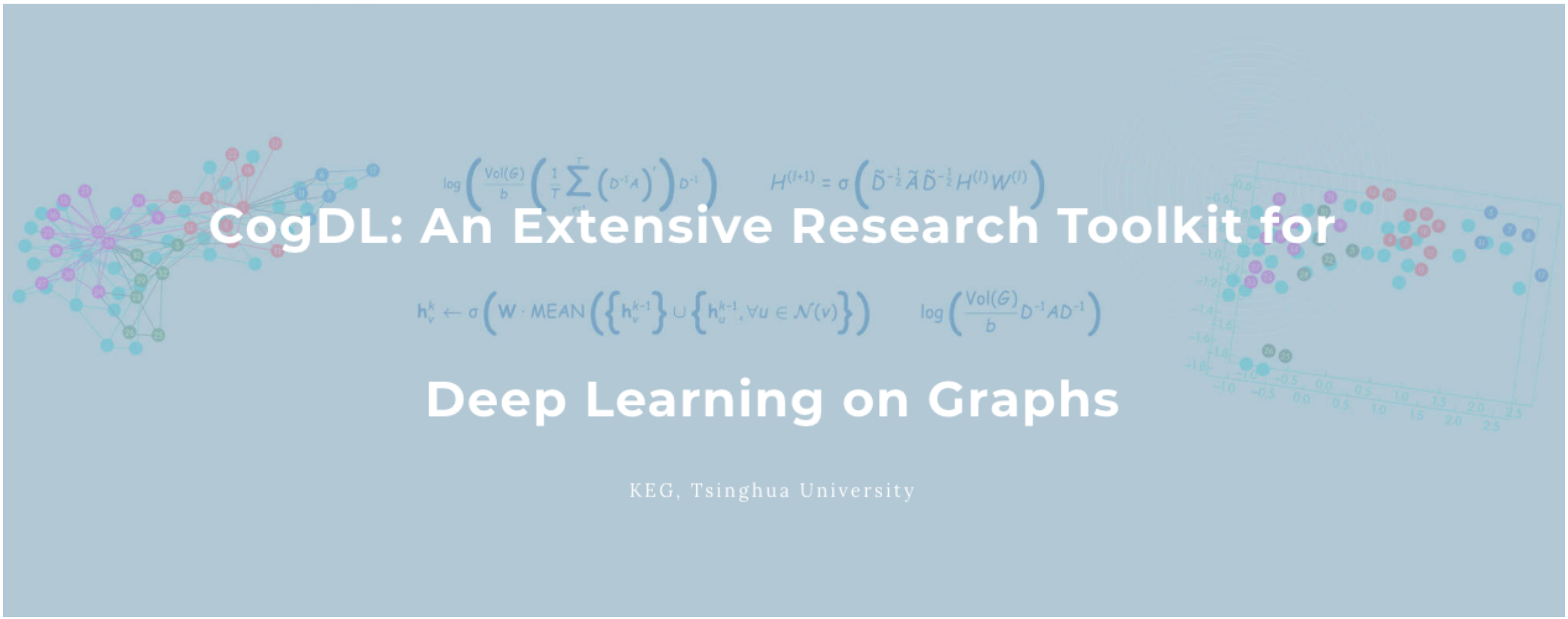
Open Graph Benchmark

- Large-scale, realistic, and diverse benchmark datasets for graph ML.



Paper: https://arxiv.org/abs/2005.00687
Webpage: https://ogb.stanford.edu/
Github: https://github.com/snap-stanford/ogb

CogDL


CogDL: An Extensive Research Toolkit for
Deep Learning on Graphs
KEG, Tsinghua University

Algorithms

Unsupervised Node Representation Learning

Algorithm	Directed	Weight	Shallow network	Matrix factorization	Sampling	Reproducibility
DeepWalk			✓			✓
LINE	✓	✓	✓		✓	✓
Node2vec	✓	✓	✓		✓	✓
SDNE	✓	✓	✓			✓
DNGR	✓	✓	✓			
HOPE	✓	✓		✓		✓
GraRep	✓	✓		✓		
NetMF	✓	✓		✓		✓
NetSMF		✓		✓	✓	✓
ProNE	✓	✓		✓		✓

Algorithms

Semi-supervised Node Representation Learning

Algorithm	Weight	Sampling	Attention	Inductive	Reproducibility
Graph U-Net	✓	✓			✓
MixHop	✓				✓
Dr-GAT			✓	✓	✓
GAT			✓	✓	✓
DGI	✓	✓		✓	✓
GCN	✓			✓	✓
GraphSAGE	✓	✓		✓	✓
Chebyshev	✓			✓	✓

Algorithms

Node Representation Learning for Heterogeneous Graphs

Algorithm	Multi-Node	Multi-Edge	Attribute	Supervised	MetaPath	Reproducibility
GATNE	✓	✓	✓		✓	✓
Metapath2vec	✓				✓	✓
PTE	✓					✓
Hin2vec	✓				✓	✓
GTN	✓		✓	✓	✓	✓
HAN	✓		✓	✓	✓	✓

Algorithms

Graph-level Representation Learning

Algorithm	Node feature	Unsupervised	Graph kernel	Shallow network	Reproducibility
Infograph	✓	✓			✓
Diffpool	✓				✓
Graph2Vec		✓	✓	✓	✓
Sortpool	✓				✓
GIN	✓				✓
PATCHY_SAN	✓		✓		✓
DGCNN	✓				✓
DGK		✓	✓	✓	

GRL: NE&GNN

Network Embedding ⇔ Matrix Factorization

Graph Neural Networks ⇨ GNN Pre-Training

Graph Data & Benchmarks

OPEN GRAPH BENCHMARK

https://alchemy.tencent.com/

Microsoft Academic

AMiner

CogDL

https://github.com/thudm/cogdl

Related Publications

For more, check http://keg.cs.tsinghua.edu.cn/jietang

- Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. **NeurIPS'20**.
- Ming Ding, Chang Zhou, Hongxia Yang, and Jie Tang. CogLTX: Applying BERT to Long Texts. **NeurIPS'20**.
- Jiezhong Qiu, Chi Wang, Ben Liao, Richard Peng, and Jie Tang. Concentration Bounds for Co-occurrence Matrices of Markov Chains. **NeurIPS'20**.
- Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. Self-supervised Learning: Generative or Contrastive. https://arxiv.org/pdf/2006.08218.pdf
- Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: Graph Contrastive Coding for Structural Graph Representation Pre-Training. **KDD'20**.
- Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. Understanding Negative Sampling in Graph Representation Learning. **KDD'20**.
- Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. Controllable Multi-Interest Framework for Recommendation. **KDD'20**.
- Yuxiao Dong, Ziniu Hu, Kuansan Wang, Yizhou Sun and Jie Tang. Heterogeneous Network Representation Learning. **IJCAI'20**.
- Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive Graph for Multi-Hop Reading Comprehension at Scale. **ACL'19**.
- Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. ProNE: Fast and Scalable Network Representation Learning. **IJCAI'19**.
- Yukuo Cen, Xu Zou, Jianwei Zhang, Hongxia Yang, Jingren Zhou and Jie Tang. Representation Learning for Attributed Multiplex Heterogeneous Network. **KDD'19**.
- Fanjin Zhang, Xiao Liu, Jie Tang, Yuxiao Dong, Peiran Yao, Jie Zhang, Xiaotao Gu, Yan Wang, Bin Shao, Rui Li, and Kuansan Wang. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. **KDD'19**.
- Qibin Chen, Junyang Lin, Yichang Zhang, Hongxia Yang, Jingren Zhou and Jie Tang. Towards Knowledge-Based Personalized Product Description Generation in E-commerce. **KDD'19**.
- Yifeng Zhao, Xiangwei Wang, Hongxia Yang, Le Song, and Jie Tang. Large Scale Evolving Graphs with Burst Detection. **IJCAI'19**.
- Yu Han, Jie Tang, and Qian Chen. Network Embedding under Partial Monitoring for Evolving Networks. **IJCAI'19**.
- Yifeng Zhao, Xiangwei Wang, Hongxia Yang, Le Song, and Jie Tang. Large Scale Evolving Graphs with Burst Detection. **IJCAI'19**.
- Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Chi Wang, Kuansan Wang, and Jie Tang. NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization. **WWW'19**.
- Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. DeepInf: Modeling Influence Locality in Large Social Networks. **KDD'18**.
- Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. **WSDM'18**.
- Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: Extraction and Mining of Academic Social Networks. **KDD'08**.

Thank you !

Collaborators:

Jie Zhang, Ming Ding, Jiezhong Qiu, Qibin Chen, Yifeng Zhao, Yukuo Cen, Yu Han, Fanjin Zhang, Xu Zou, Yan Wang, et al. (**THU**)

Yuxiao Dong, Kuansan Wang (**Microsoft**)

Hongxiao Yang, Chang Zhou, Le Song, Jingren Zhou, et al. (**Alibaba**)

Jie Tang, KEG, Tsinghua U

Download all data & Codes

http://keg.cs.tsinghua.edu.cn/jietang
https://keg.cs.tsinghua.edu.cn/cogdl/
https://github.com/THUDM