Graph Representation Learning & Applications

GNNs & Pre-Training

Yuxiao Dong
https://ericdongyx.github.io/

Microsoft Research, Redmond

Joint Work with

Jiezhong Qiu
Tsinghua

Ziniu Hu
UCLA

Wenzheng Feng
Tsinghua University

Weihua Hu
Stanford University

Marinka Zitnik
Harvard University

Jie Tang
Tsinghua

Yizhou Sun
UCLA

Jure Leskovec
Stanford University

Hao Ma
Facebook AI

Kuansan Wang
Microsoft Research

Graph Representation Learning

Network Embedding ⇔ Matrix Factorization

Graph Neural Networks ⇨ GNN Pre-Training

Graph Data & Benchmarks

OPEN GRAPH BENCHMARK

https://alchemy.tencent.com/

Miner

Microsoft Academic

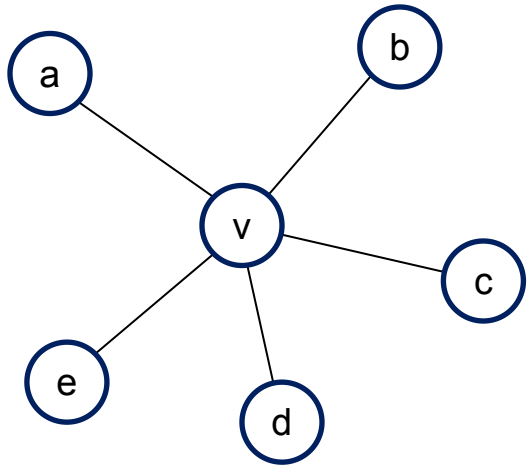
CogDL

github.com/thudm/cogdl

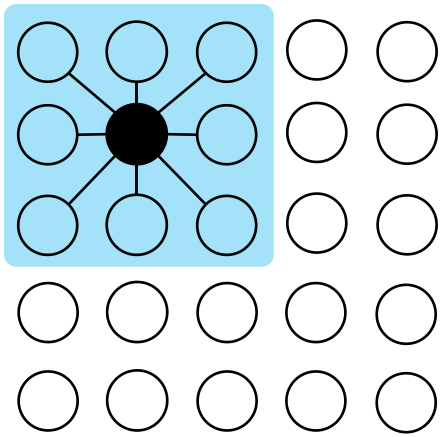
Graph Representation Learning

- **Brief introduction of GNNs**
- How to attend over heterogenous graphs?
- Do we really need message passing in GNNs?
- Can we pre-train for graph representations?

Graph Neural Networks



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



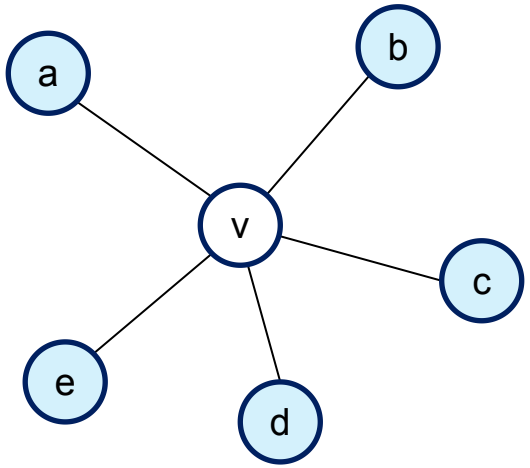
Graph Convolution **CNN**

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**

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

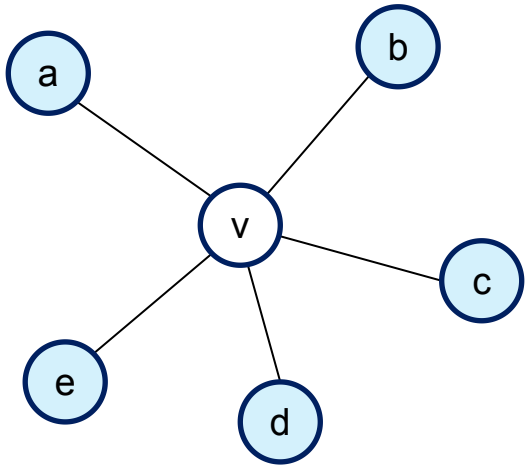
$$\boldsymbol{H}^k = \sigma\left(\hat{\boldsymbol{A}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right)$$

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



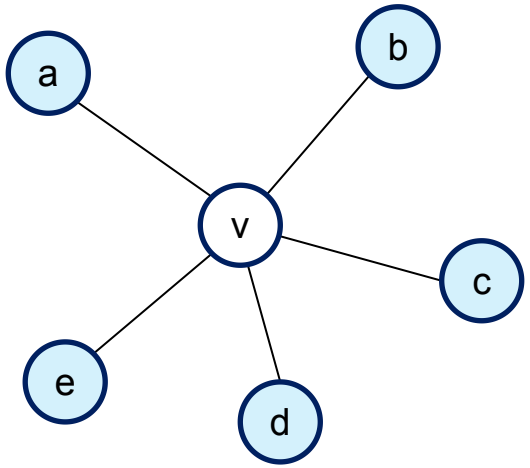
Aggregate from v's neighbors

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

Aggregate from itself

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

Graph Convolutional Networks

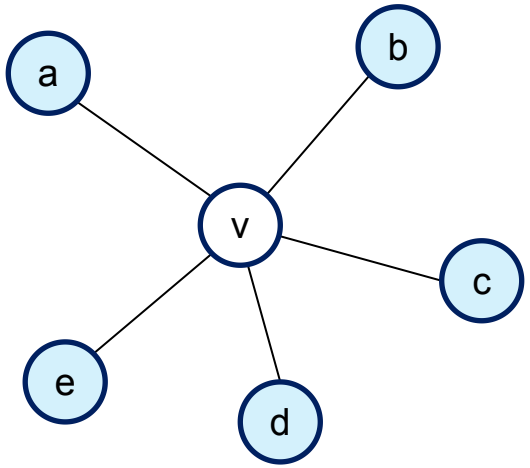


The same parameters for both its neighbors & itself

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

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

Graph Convolutional Networks



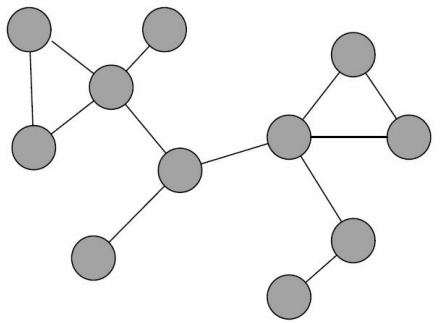
$$\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}$$

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

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

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

Graph Convolutional Networks



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

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

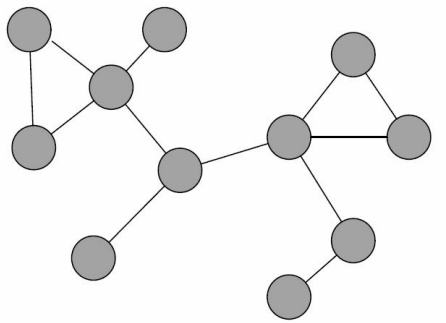
$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right) \Rightarrow \boldsymbol{Z} = \boldsymbol{H}^K$$

Input

Output

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

Graph Convolutional Networks



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

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

Input

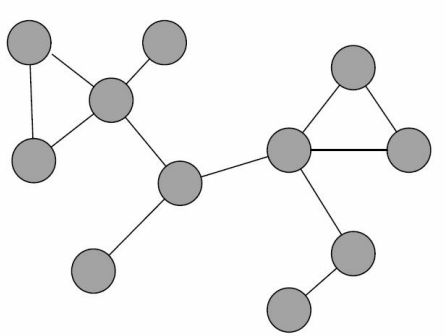
$$\Rightarrow \boldsymbol{H}^k = \sigma\left(\boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A}+\boldsymbol{I})\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{H}^{(k-1)}\boldsymbol{W}^{(k)}\right) \Rightarrow \boldsymbol{Z} = \boldsymbol{H}^K$$

Output

- 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 $\boldsymbol{Z}$

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

Graph Convolutional Networks



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

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

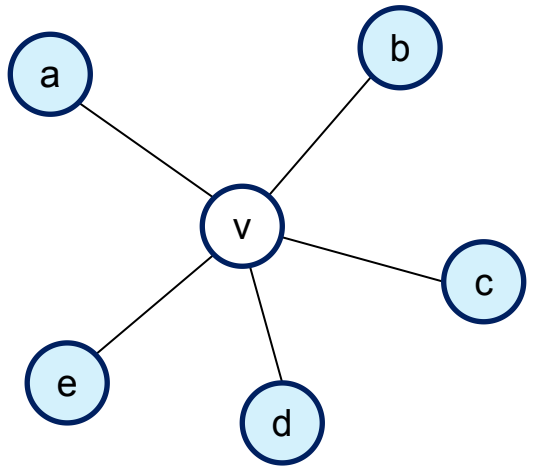
Input

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

Output

- Benefits: Parameter sharing for all nodes
  - #parameters is subline in |V|
  - Enable inductive learning for new nodes

GraphSage



GCN

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

GraphSage

Instead of summation, it concatenates neighbor & self embeddings

$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}^k \cdot \mathrm{AGG}(\{\boldsymbol{h}_u^{k-1}, \forall u \in N(v)\}), \boldsymbol{B}^k \boldsymbol{h}_v^{k-1}])$$

Generalized aggregation: any differentiable function that maps set of vectors to a single vector

Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017

GraphSage

$$\boldsymbol{h}_v^k = \sigma([\boldsymbol{A}^k \cdot \mathrm{AGG}(\{\boldsymbol{h}_u^{k-1}, \forall u \in N(v)\}), \boldsymbol{B}^k \boldsymbol{h}_v^{k-1}])$$

- Mean:

$$\mathrm{AGG} = \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}$$

- Pool
  - Transform neighbor vectors and apply symmetric vector function.

element-wise mean/max

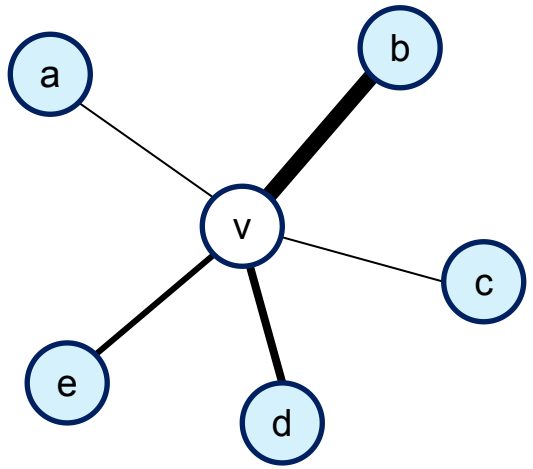
$$\mathrm{AGG} = \gamma\left(\{\mathbf{Q}\mathbf{h}_u^{k-1}, \forall u \in N(v)\}\right)$$

- LSTM:
  - Apply LSTM to random permutation of neighbors.

$$\mathrm{AGG} = \mathrm{LSTM}\left([\mathbf{h}_u^{k-1}, \forall u \in \pi(N(v))]\right)$$

Hamilton et al. Inductive Representation Learning on Large Graphs. NIPS 2017
Slide snipping from “Hamiltion & Tang, AAAI 2019 Tutorial on Graph Representation Learning”

Graph Attention



GCN

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

Aggregate info from neighborhood via the normalized Laplacian matrix

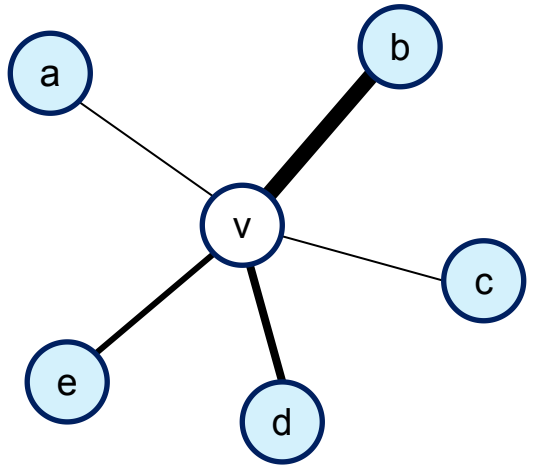
Graph Attention

$$\boldsymbol{h}_v^k = \sigma(\sum_{u \in \mathrm{N}(v) \cup v} \alpha_{v,u} \boldsymbol{W}^k \boldsymbol{h}_u^{k-1})$$

Aggregate info from neighborhood via the learned attention

Velickovic et al. Graph Attention Networks. ICLR 2018

Graph Attention

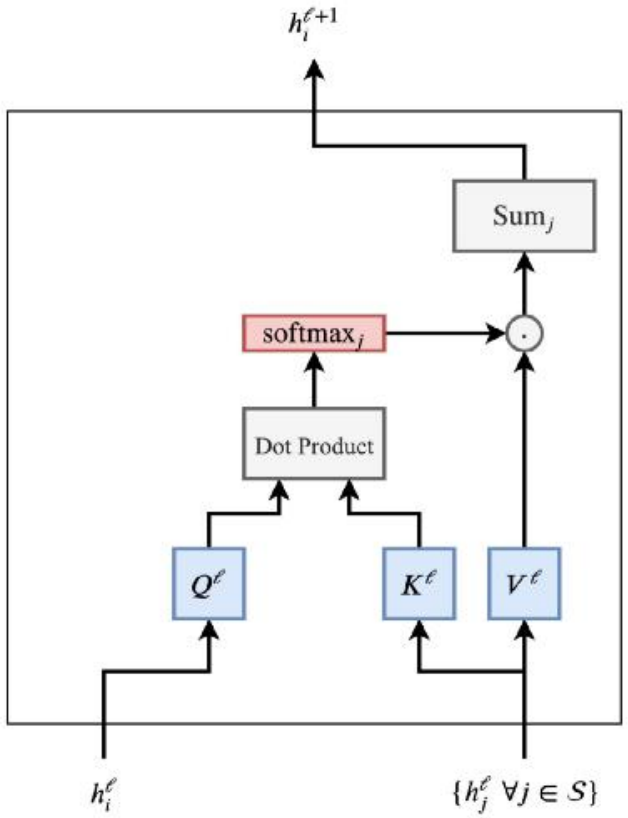


$$\alpha_{v,u} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_u]\right)\right)}{\sum_{u' \in N(v) \cup \{v\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{Q}\mathbf{h}_v, \mathbf{Q}\mathbf{h}_{u'}]\right)\right)}$$

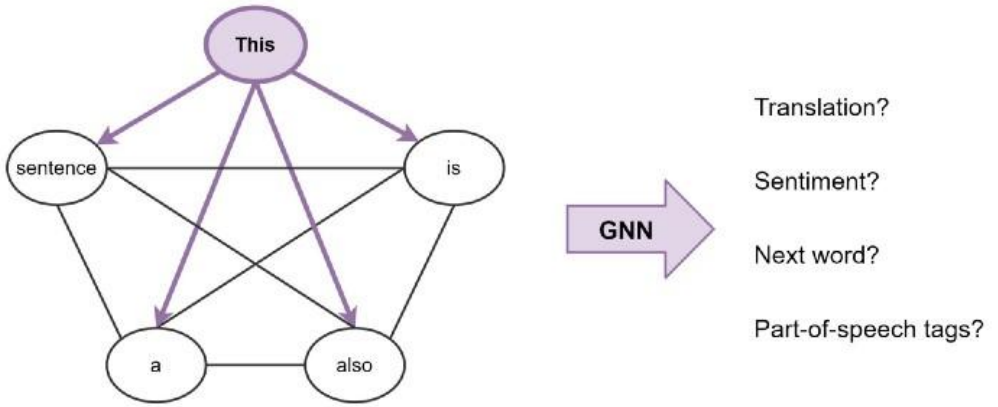
many ways to define attention!

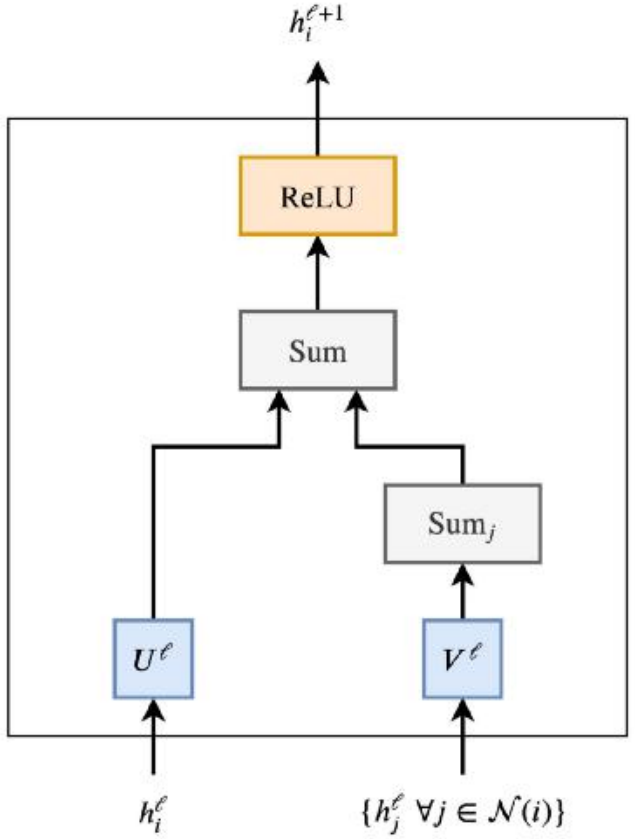
Velickovic et al. Graph Attention Networks. ICLR 2018

Transformer as GNNs



Transformer



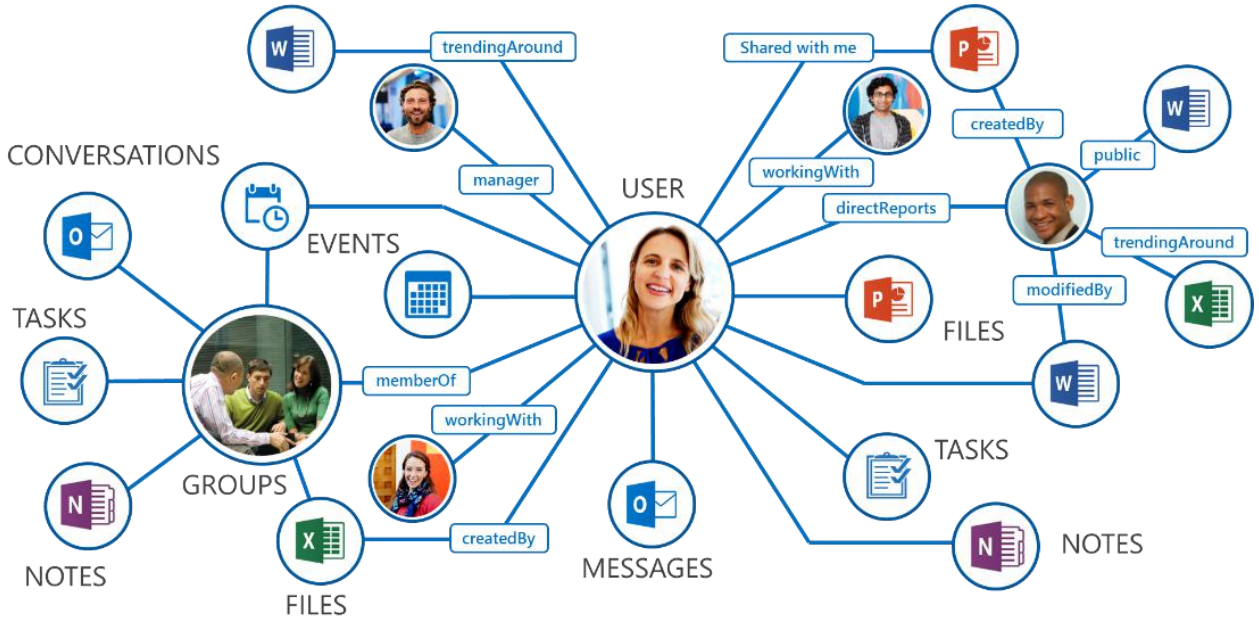


Graph Neural Nets

Figure credit: Chaitanya Joshi

Attention over Heterogeneous Graphs?

heterogeneous academic graph



heterogeneous office graph

Heterogeneous Graph Transformer

- **Heterogeneous Graph Transformer (HGT)**
  - Define unique parameters for each type of nodes & edges
  - Leverage meta relation of each edge to parameterize attention & message passing weights

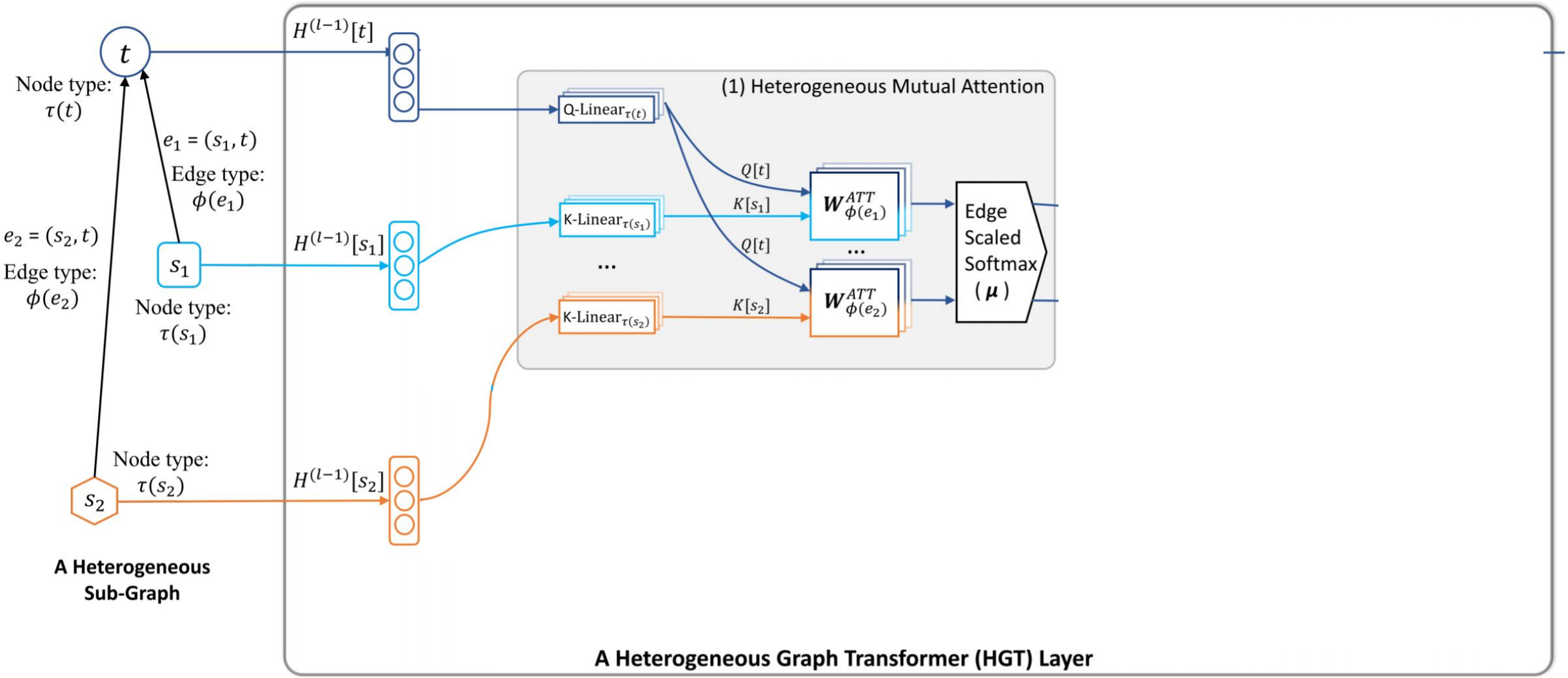
- meta relation of an edge $e = (s, t)$

$$\langle \tau(s), \phi(e), \tau(t) \rangle$$

- <author, write, paper>

Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Heterogeneous Graph Transformer (HGT)



Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Heterogeneous Graph Transformer (HGT)

meta relation of $e = (s, t)$

$\langle \tau(s), \phi(e), \tau(t) \rangle$

- **heterogeneous mutual attention**

$$\textbf{Attention}_{HGT}(s,e,t) = \underset{\forall s \in N(t)}{\text{Softmax}}\Big(\Big\|_{i \in [1,h]} ATT\text{-}head^i(s,e,t) \Big) \quad (3)$$

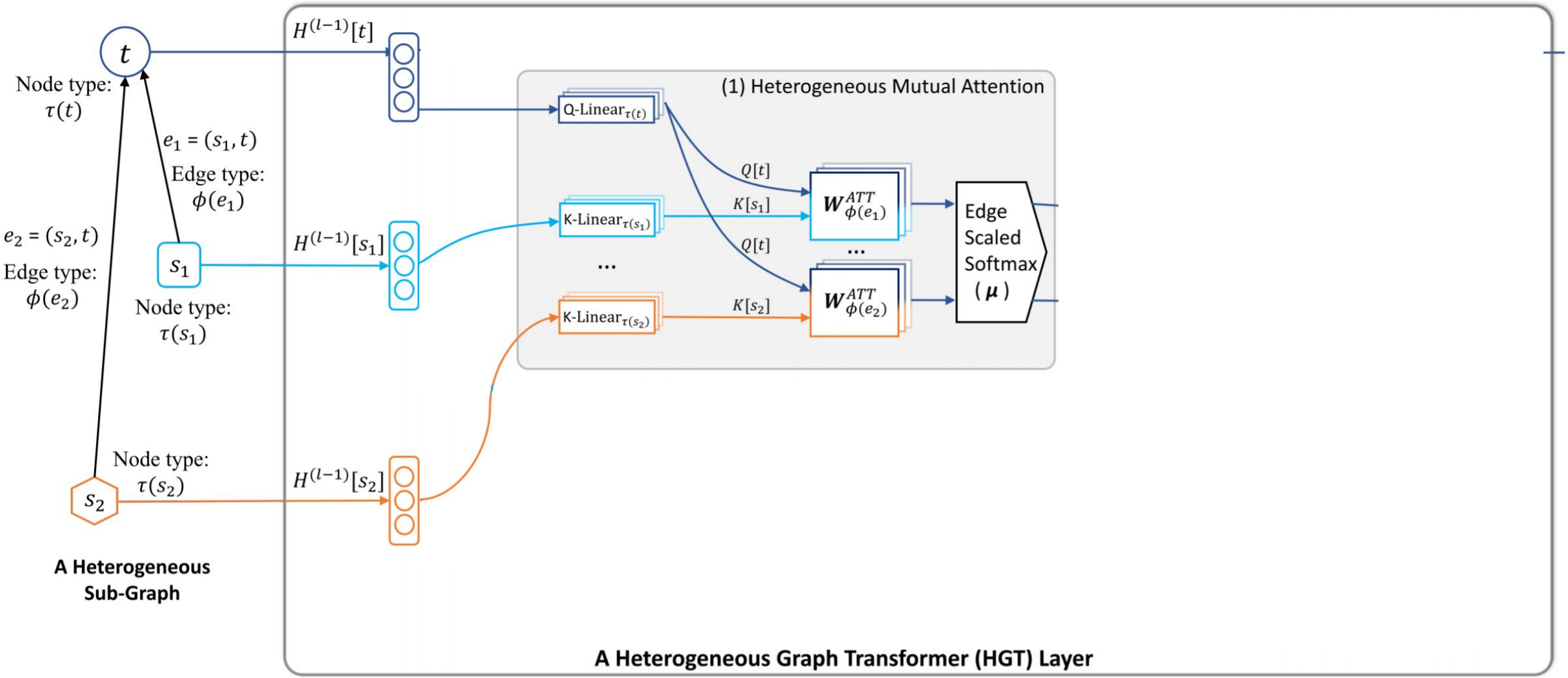
$$ATT\text{-}head^i(s,e,t) = \Big(K^i(s)\, W^{ATT}_{\phi(e)}\, Q^i(t)^T \Big) \cdot \frac{\mu_{\langle \tau(s), \phi(e), \tau(t) \rangle}}{\sqrt{d}}$$

$$K^i(s) = \text{K-Linear}^i_{\tau(s)}\Big(H^{(l-1)}[s] \Big)$$

$$Q^i(t) = \text{Q-Linear}^i_{\tau(t)}\Big(H^{(l-1)}[t] \Big)$$

Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Heterogeneous Graph Transformer (HGT)



Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Heterogeneous Graph Transformer (HGT)

meta relation of $e = (s, t)$

$\langle \tau(s), \phi(e), \tau(t) \rangle$

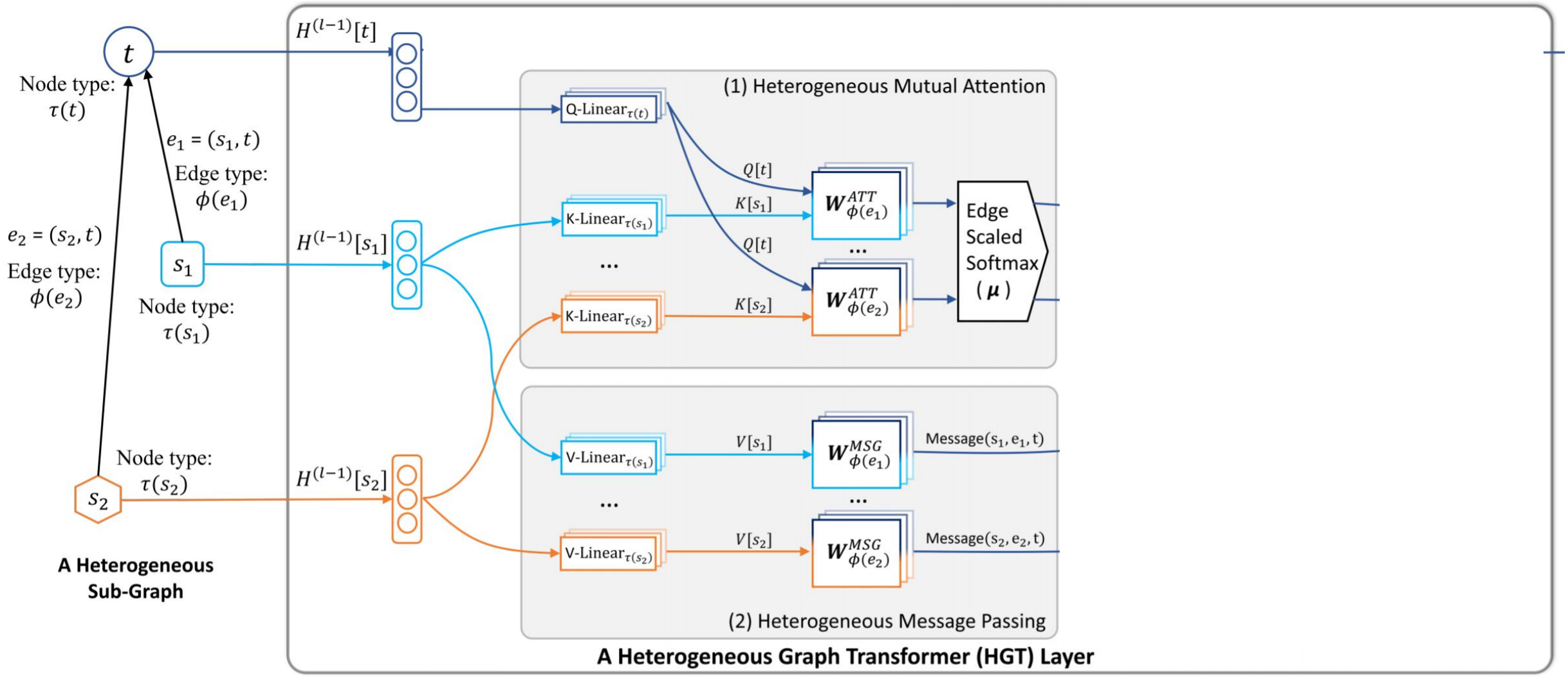
- **heterogeneous message passing**

$$\textbf{Message}_{HGT}(s, e, t) = \Big\Vert_{i \in [1,h]} MSG\text{-}head^i(s, e, t)$$

$$MSG\text{-}head^i(s, e, t) = \text{V-Linear}^i_{\tau(s)}\left(H^{(l-1)}[s]\right) W^{MSG}_{\phi(e)}$$

Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Heterogeneous Graph Transformer (HGT)



Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Heterogeneous Graph Transformer (HGT)

meta relation of $e = (s, t)$

$\langle \tau(s), \phi(e), \tau(t) \rangle$

- **Target specific aggregation**

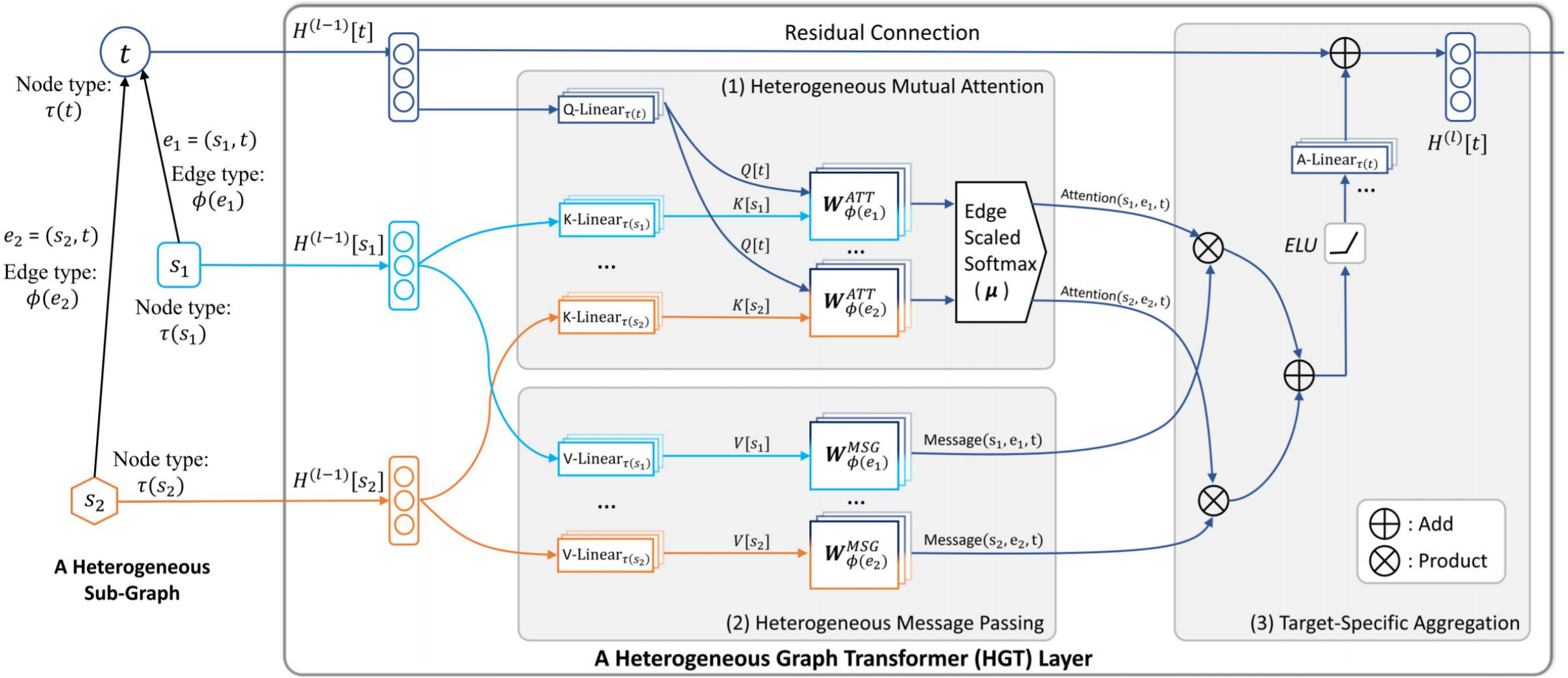
$$\widetilde{H}^{(l)}[t] = \bigoplus_{\forall s \in N(t)} \left(\textbf{Attention}_{HGT}(s, e, t) \cdot \textbf{Message}_{HGT}(s, e, t) \right)$$

$$H^{(l)}[t] = \text{A-Linear}_{\tau(t)}\left(\sigma\left(\widetilde{H}^{(l)}[t] \right) \right) + H^{(l-1)}[t]$$

$$\textbf{Attention}_{HGT}(s, e, t) = \underset{\forall s \in N(t)}{\text{Softmax}}\left(\Big\Vert_{i \in [1, h]} ATT\text{-}head^{i}(s, e, t) \right)$$

Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Heterogeneous Graph Transformer



Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Graph Dynamics

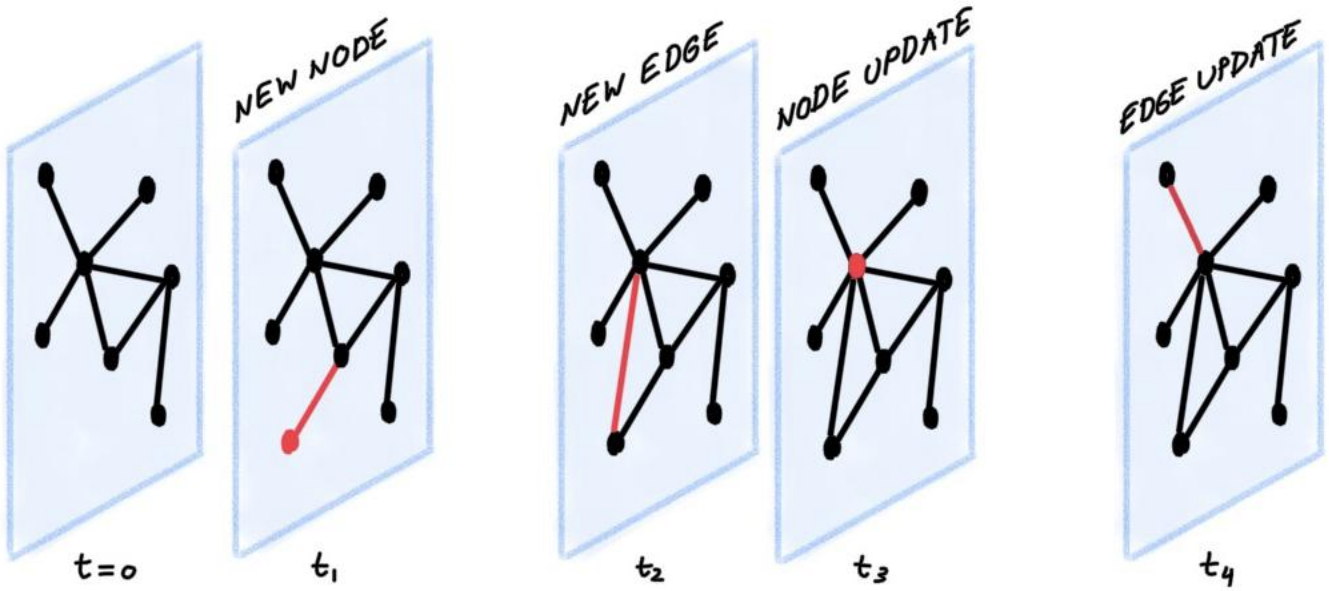
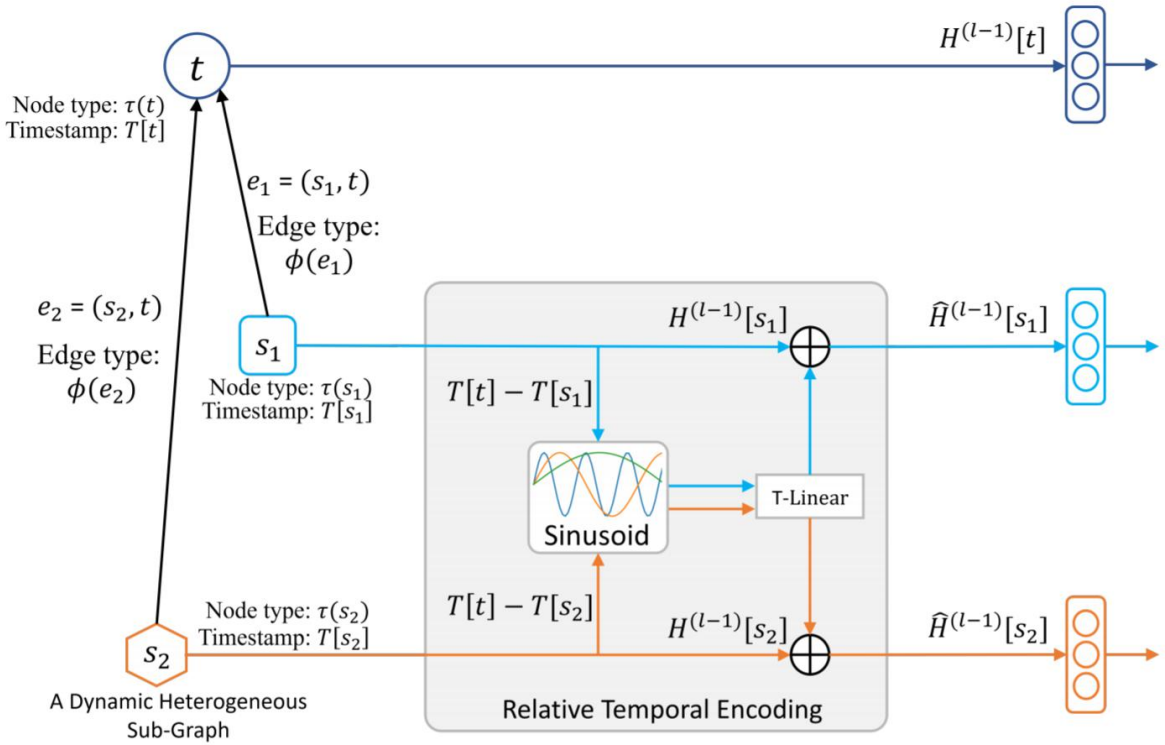


Figure Credit: Michael Bronstein

Common strategy: Slice the dynamic graph into multiple timestamps

Relative Temporal Encoding (RTE) in HGT



$$\widehat{H}^{(l-1)}[s] = H^{(l-1)}[s] + RTE\big(\Delta T(t,s)\big)$$

$$RTE\big(\Delta T(t,s)\big) = \text{T-Linear}\Big(Base(\Delta T_{t,s})\Big)$$

$$Base\big(\Delta T(t,s), 2i\big) = sin\Big(\Delta T_{t,s}/10000^{\frac{2i}{d}}\Big)$$

$$Base\big(\Delta T(t,s), 2i+1\big) = cos\Big(\Delta T_{t,s}/10000^{\frac{2i+1}{d}}\Big)$$

- Maintain all edges in different timestamps

Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Large-Scale Graphs: HGSampling in HGT

Algorithm 1 Heterogeneous Mini-Batch Graph Sampling

Require: Adjacency matrix A for each $\langle \tau(s), \phi(e), \tau(t) \rangle$ relation pair; Output node Set OS; Sample number n per node type; Sample depth L.

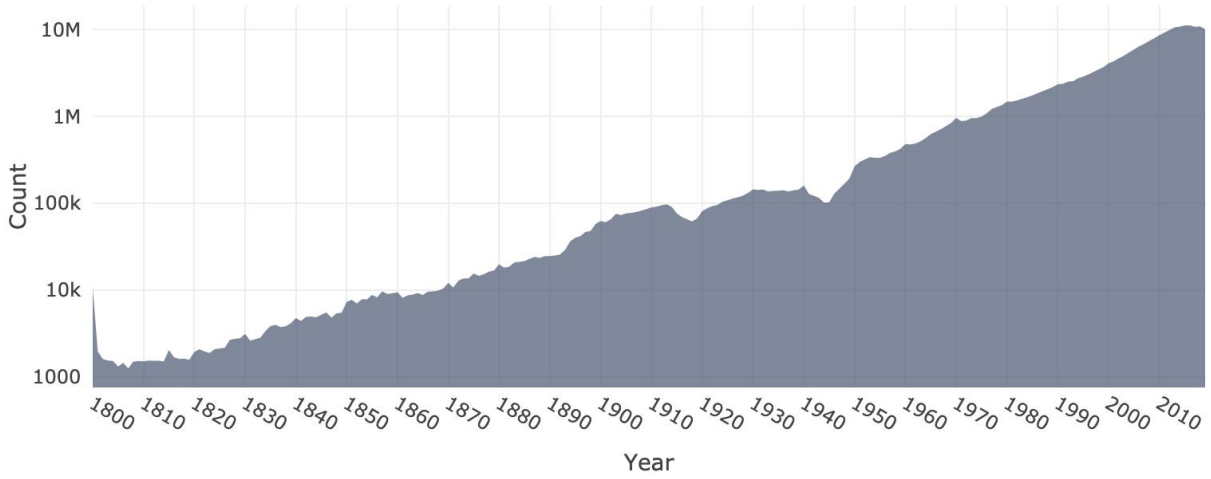
Ensure: Sampled node set NS; Sampled adjacency matrix $\hat{A}$.

$NS \leftarrow OS$ // Initialize sampled node set as output node set.
Initialize an empty Budget B storing nodes for each node type with normalized degree.
for $t \in NS$ **do**
  Add-In-Budget(B, t, A, NS) // Add neighbors of t to B.
end for
for $l \leftarrow 1$ to L **do**
  for source node type $\tau \in B$ **do**
    for source node $s \in B[\tau]$ **do**
      $prob^{(l-1)}[\tau][s] \leftarrow \frac{B[\tau][s]^2}{\|B[\tau]\|_2^2}$ // Calculate sampling probability for each source node s of node type τ.
    end for
    Sample n nodes $\{t_i\}_{i=1}^{n}$ from $B[\tau]$ using $prob^{(l-1)}[\tau]$.
    for $t \in \{t_i\}_{i=1}^{n}$ **do**
      $OS[\tau].add(t)$ // Add node t into Output node set.
      Add-In-Budget(B, t, A, NS) // Add neighbors of t to B.
      $B[\tau].pop(t)$ // Remove sampled node t from Budget.
    end for
  end for
end for
Reconstruct the sampled adjacency matrix $\hat{A}$ among the sampled nodes OS from A.
return OS and $\hat{A}$;

Experiments

Microsoft Academic Graph

236,963,398 Publications	740,048 Topics	48,876 Journals
240,667,697 Authors	4,467 Conferences	25,759 Institutions



HGT

- Infer paper field
- Infer paper venue
- Name disambiguation
- Fake paper detection
- …

***HGSampling**: Heterogeneous Mini-Batch Graph Sampling algorithm*

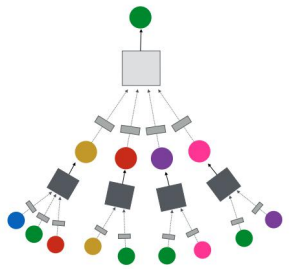


Figure Credit: Rex Ying

Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Results

GNN Models		GCN [7]	RGCN [12]	GAT [21]	HetGNN [25]	HAN [22]	$HGT_{noHeter}$	HGT_{noTime}	HGT
# of Parameters		1.69M	8.80M	1.69M	8.41M	9.45M	3.12M	7.44M	8.20M
Paper–Field (L1)	NDCG	.558±.141	.563±.128	.601±.103	.615±.084	.617±.096	.674±.086	.702±.089	**.735±.084**
	MRR	.513±.136	.526±.105	.587±.096	.595±.076	.604±.092	.652±.078	.676±.082	**.713±.081**
Paper–Field (L2)	NDCG	.241±.074	.258±.046	.276±.049	.271±.062	.281±.051	.301±.046	.307±.052	**.332±.048**
	MRR	.192±.067	.206±.052	.228±.045	.231±.053	.242±.049	.257±.058	.260±.064	**.276±.071**
Paper–Venue	NDCG	.303±.066	.354±.051	.461±.057	.447±.071	.478±.062	.515±.059	.538±.064	**.551±.062**
	MRR	.114±.070	.198±.047	.244±.052	.226±.059	.269±.067	.295±.060	.322±.048	**.334±.061**
Author Disambiguation	NDCG	.730±.064	.742±.057	.785±.063	.792±.052	.810±.049	.834±.058	.849±.066	**.857±.054**
	MRR	.762±.042	.786±.048	.843±.044	.852±.058	.876±.056	.903±.041	.911±.043	**.918±.048**

HGT offers ~9−21% improvements over existing (heterogeneous) GNNs

Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020**

Case Study

Experiments done in **2019**!

Venue	Time	Top−5 Most Similar Venues
WWW	2000	SIGMOD, VLDB, NSDI, GLOBECOM, SIGIR
	2010	GLOBECOM, KDD, CIKM, SIGIR, SIGMOD
	2020	KDD, GLOBECOM, SIGIR, WSDM, SIGMOD
KDD	2000	SIGMOD, ICDE, ICDM, CIKM, VLDB
	2010	ICDE, WWW, NeurIPS, SIGMOD, ICML
	2020	NeurIPS, SIGMOD, WWW, AAAI, EMNLP
NeurIPS	2000	ICCV, ICML, ECCV, AAAI, CVPR
	2010	ICML, CVPR, ACL, KDD, AAAI
	2020	ICML, CVPR, ICLR, ICCV, ACL

DB + Networking + IR

⇩

DM + Networking + IR + DB

DB + DM

⇩

ML + DB + Web + AI + NLP!!!

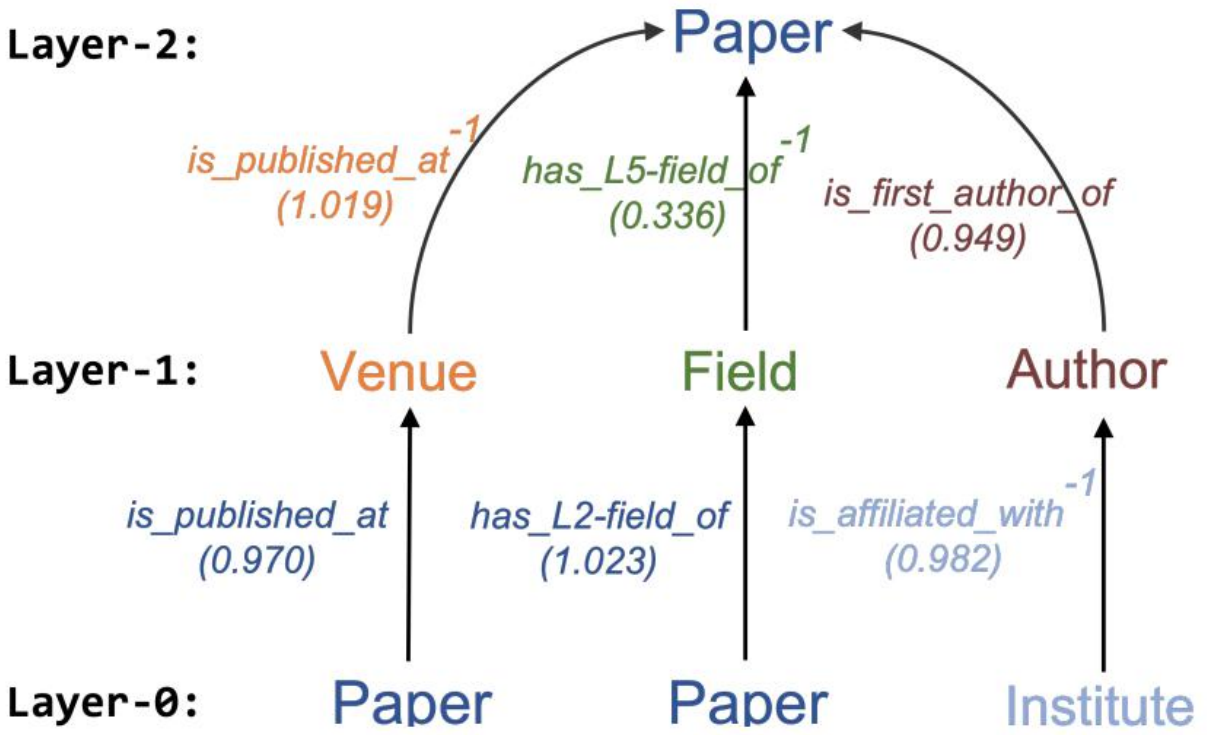
CV + ML + AI

⇩

ML + CV + DL + NLP

Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

What is the Best Part of HGT?



Learn meta-paths & their weights implicitly and automatically!

Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

Heterogeneous Graph Transformer

- To handle graph *heterogeneity*
- To avoid *manual meta-path design, so end-to-end*
  - parametrize heterogeneous attentions based on meta relations

- To handle graph *dynamics*
  - *relative temporal encoding*

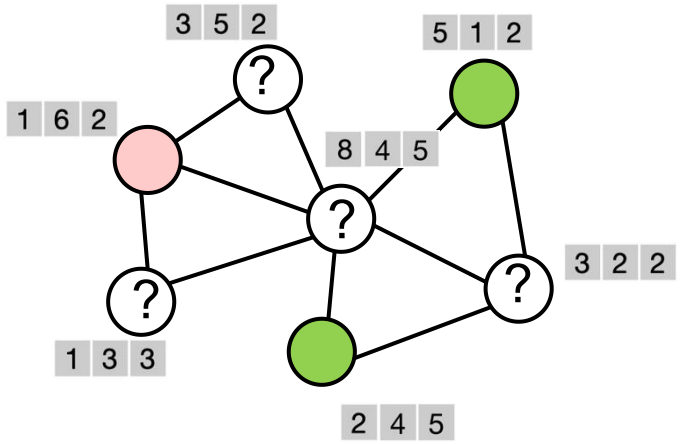
- To handle *large-scale* graph
  - *heterogeneous mini-batch graph sampling*

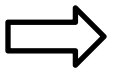
Hu, Dong, Wang, Sun. Heterogeneous Graph Transformer. **WWW 2020.**

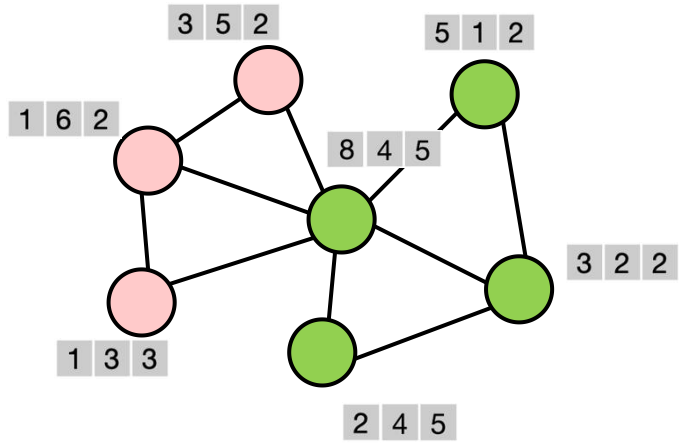
Graph Representation Learning

- Brief introduction of GNNs
- How to attend over heterogenous graphs?
- **Do we really need message passing in GNNs?**
- Can we pre-train for graph representations?

Semi-Supervised Learning on Graphs







Input: a partially labeled & attributed graph

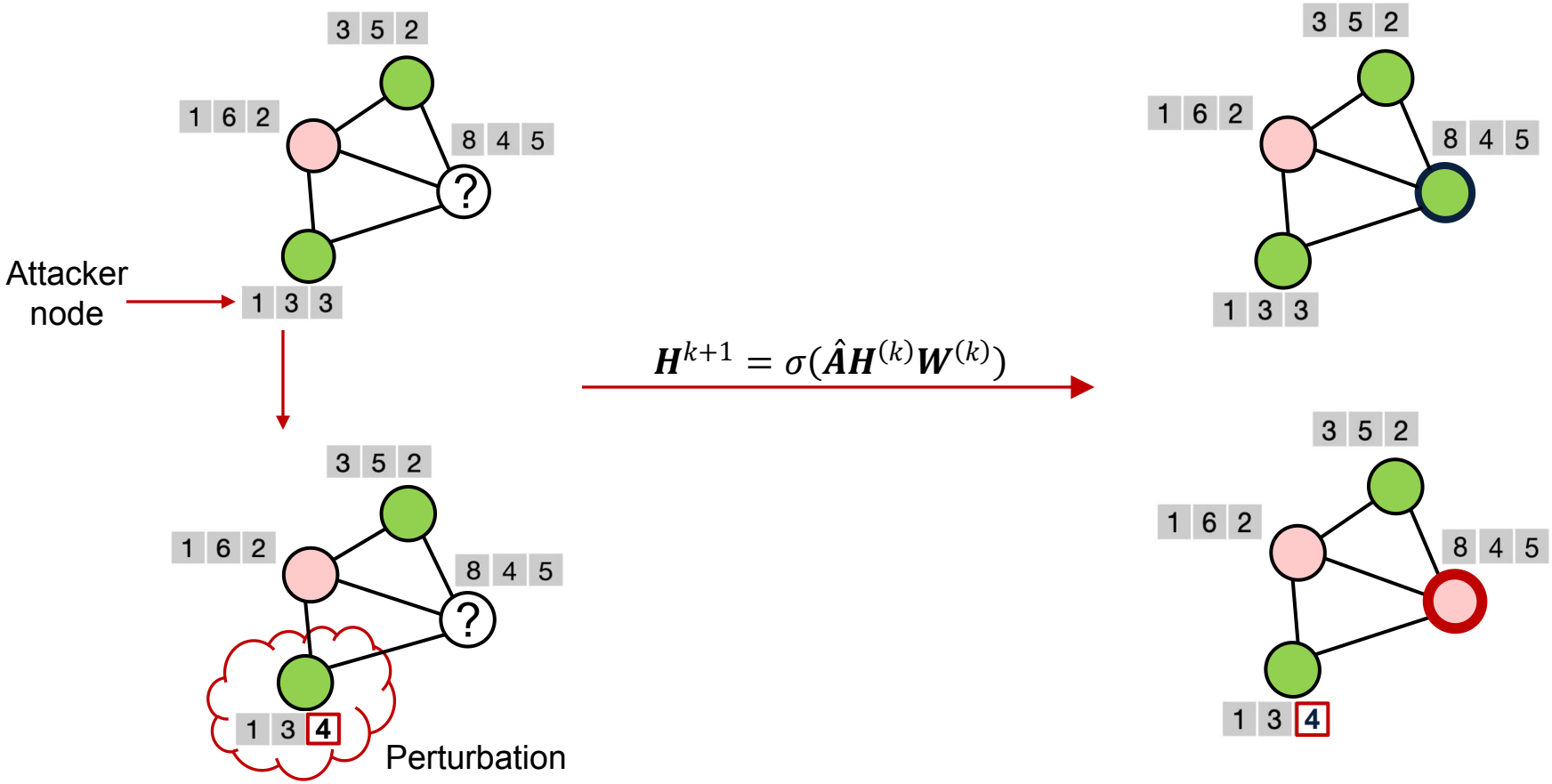
Output: infer the labels of unlabeled nodes

Graph Neural Networks

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

a deterministic propagation

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



Graph Neural Networks

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

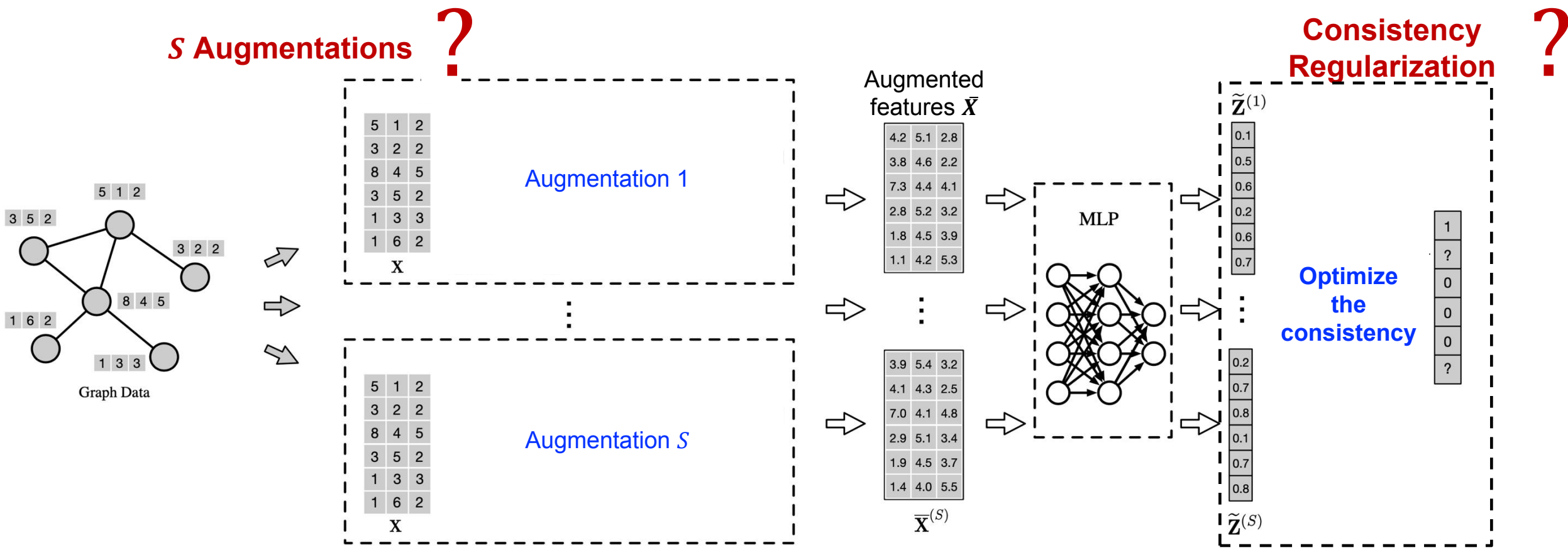
feature **propagation**
is coupled with
non-linear transformation

1. Each node is highly dependent with its neighborhoods, making GNNs **non-robust** to noises
2. Stacking many GNN layers may cause **over-fitting** & **over-smoothing.**

- Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI'18.*
- Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. IEEE Transactions on Neural Networks, 2009.
- David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. NeurIPS'19.

Graph Random Neural Networks (GRAND)

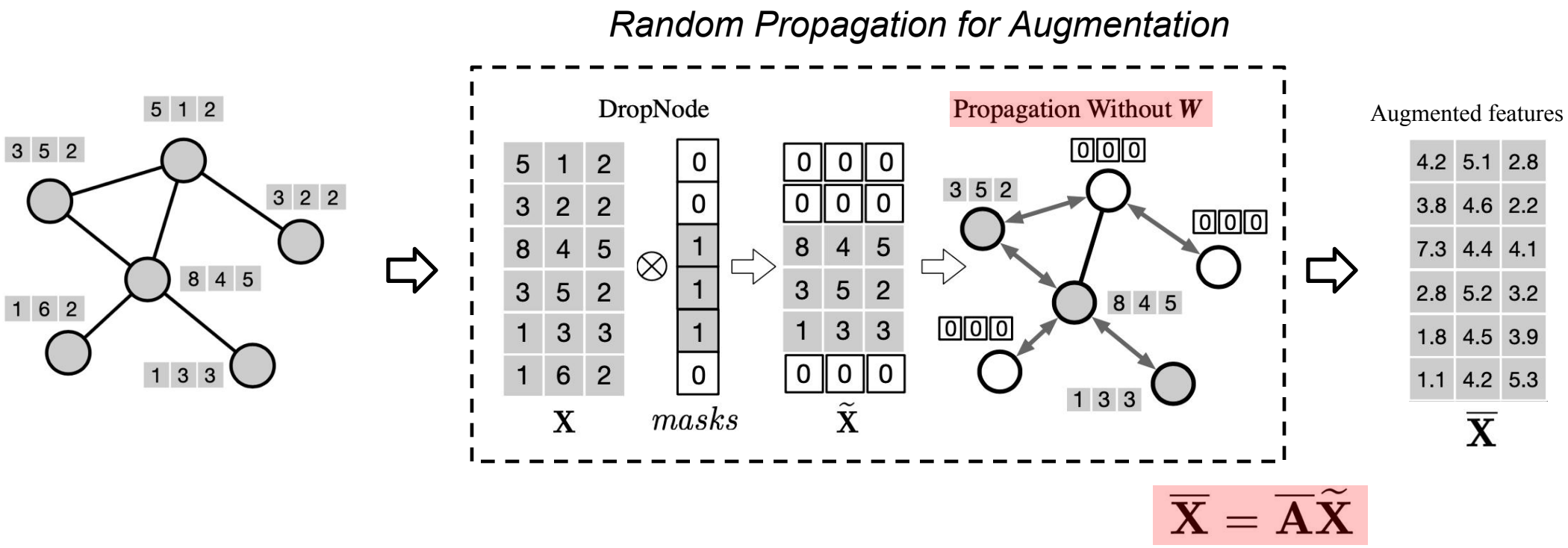
- Consistency Regularized Training:
  - Generates S data augmentations of the graph
  - Optimizing the consistency among S augmentations of the graph.



- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

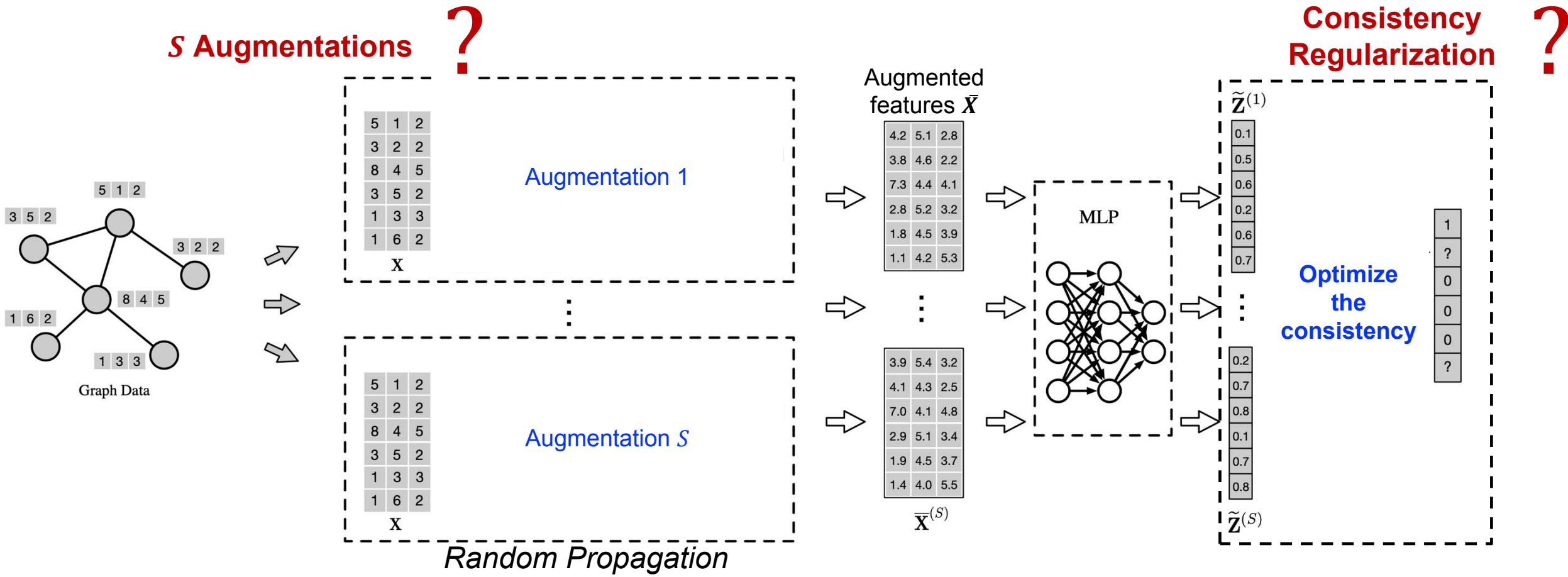
GRAND: Random Propagation for Graph Data Augmentation

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



- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

Graph Random Neural Networks (GRAND)



- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

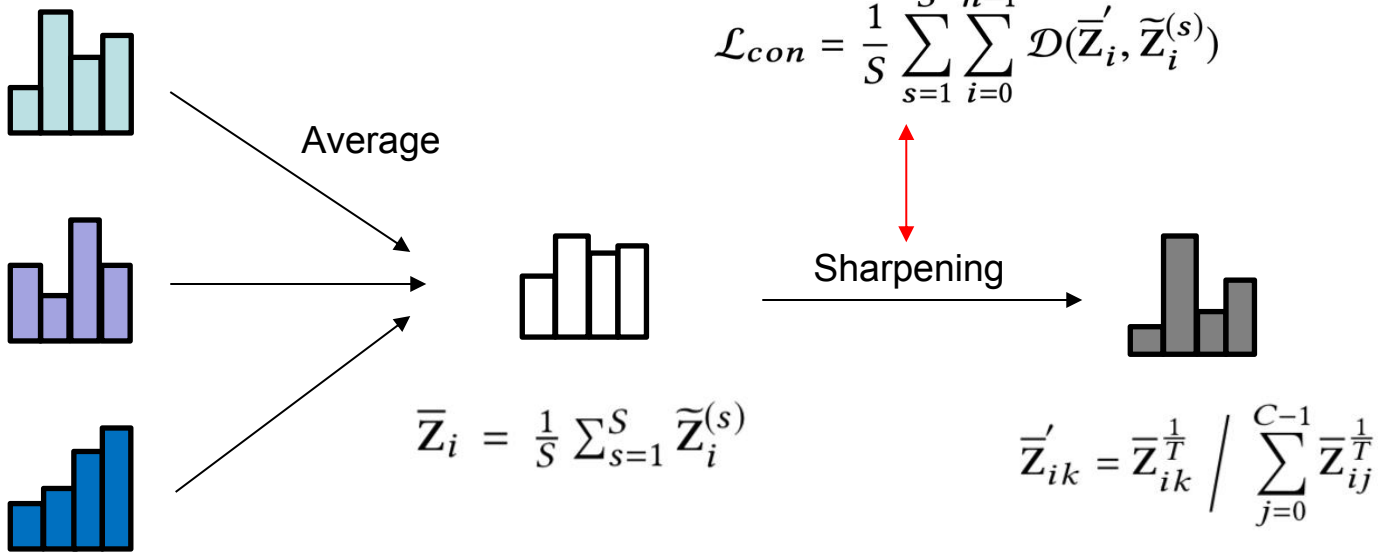
GRAND: Consistency Regularization

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

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

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

Distributions of a node after augmentations

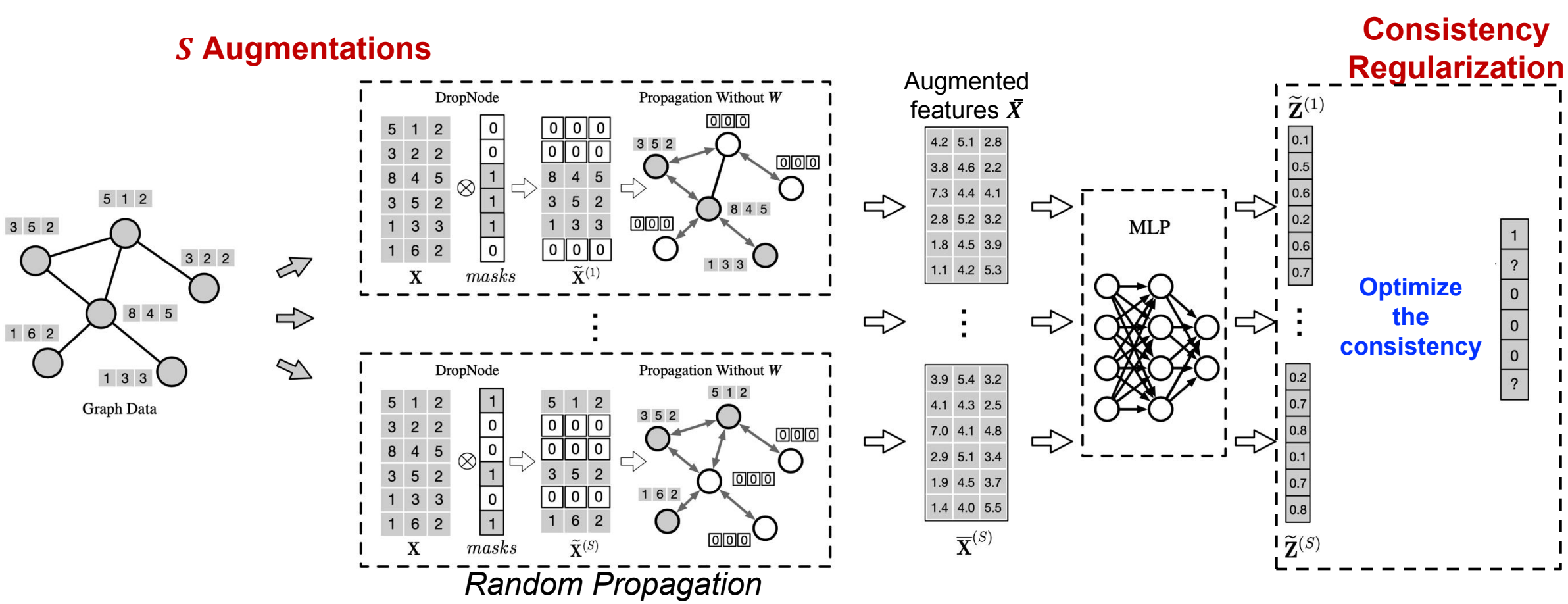


$$\overline{\mathbf{Z}}_i = \frac{1}{S}\sum_{s=1}^{S}\widetilde{\mathbf{Z}}_i^{(s)}$$

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

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

Graph Random Neural Networks (GRAND)



- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

Graph Random Neural Networks (GRAND)

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.

Consistency Regularized Training Algorithm

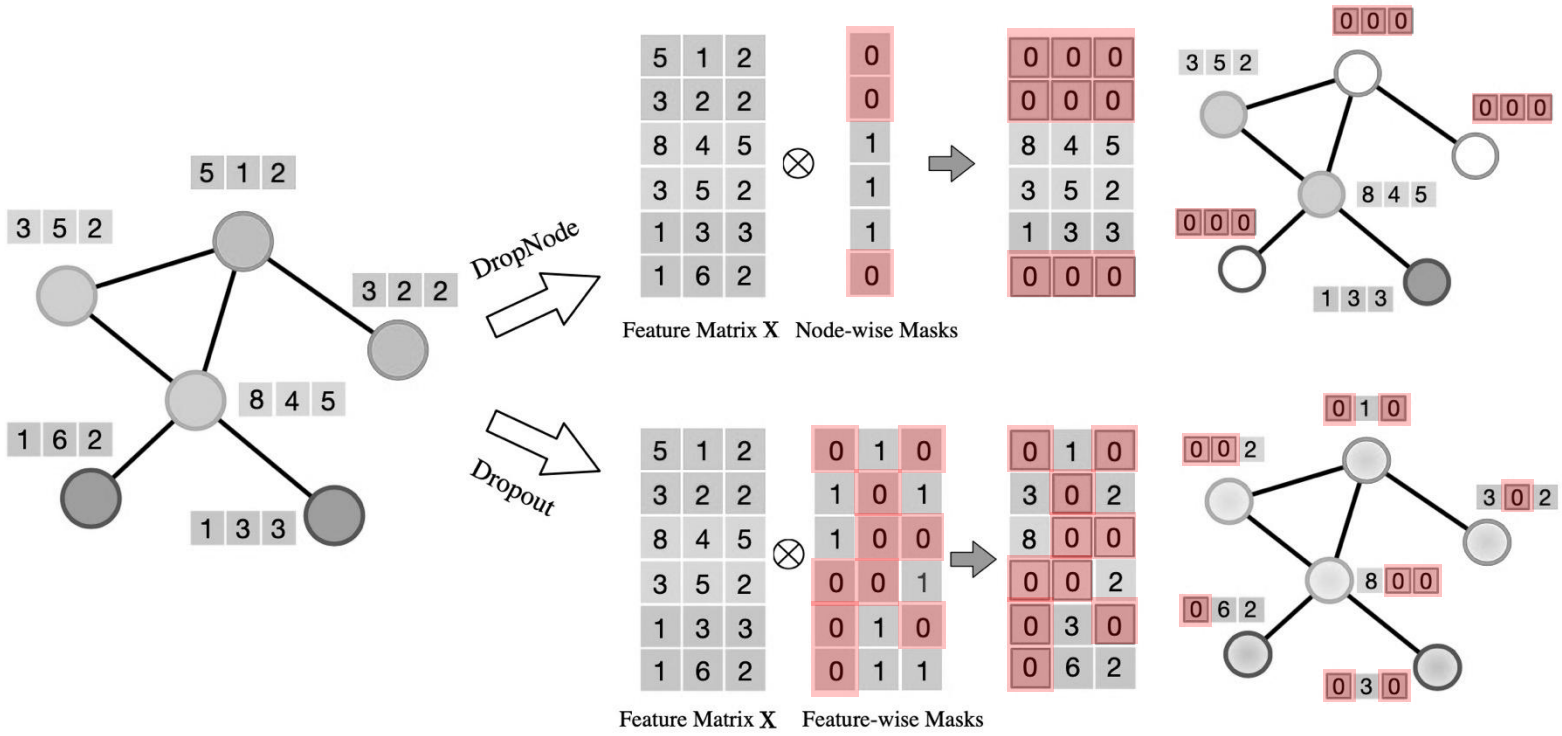
Generate S Augmentations

Consistency Regularization

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- 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.



- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

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.

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

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

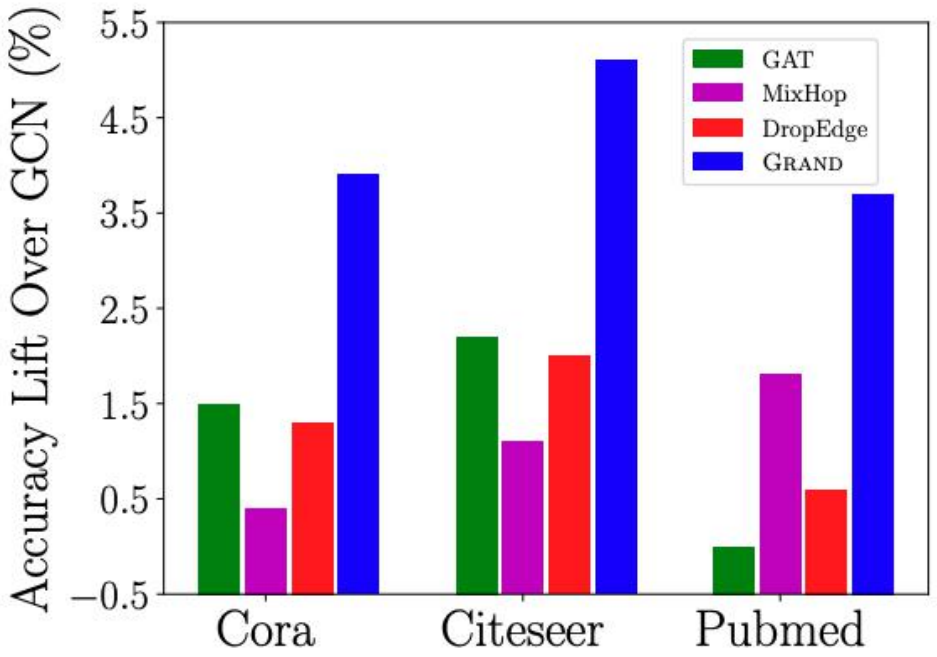
$$\mathcal{R}^c_{Do}(\mathbf{W}) = \frac{\delta}{1-\delta}\sum_{h=0}^{d-1}\mathbf{W}_h^2\sum_{j=0}^{n-1}\left[\mathbf{X}_{jh}^2\sum_{i=0}^{n-1} z_i^2(1-z_i)^2(\overline{\mathbf{A}}_{ij})^2\right]$$

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

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

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
Regularization 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**



- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

Results

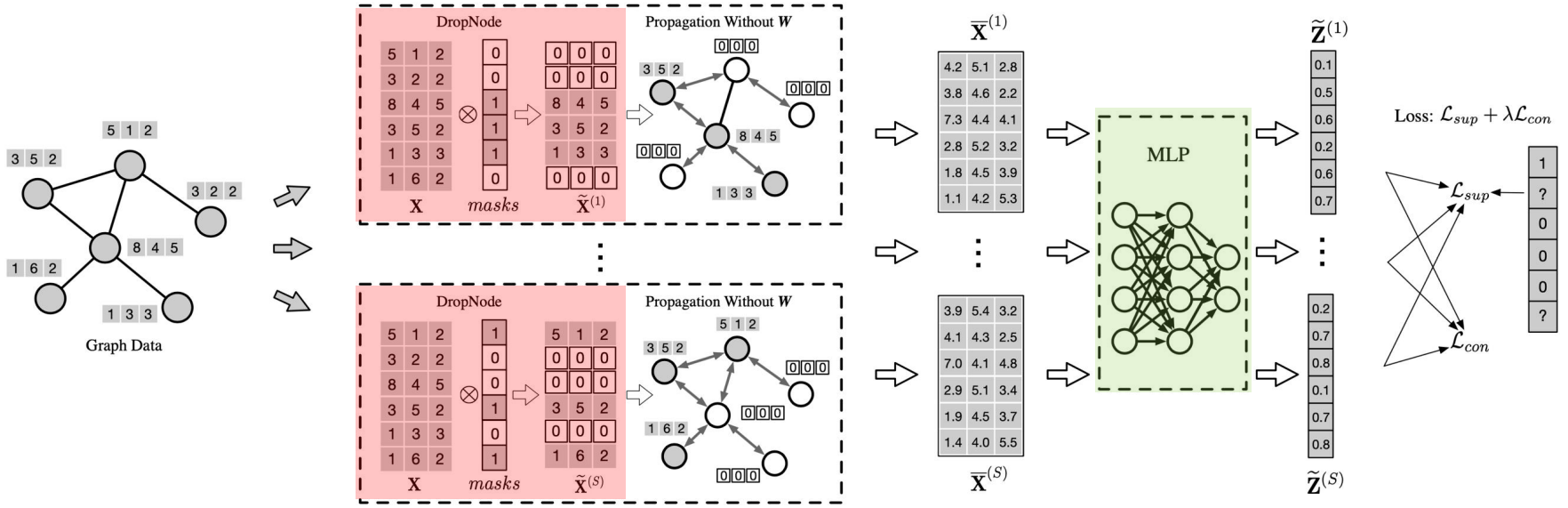
Table 5: Results on large datasets.

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

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

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

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/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

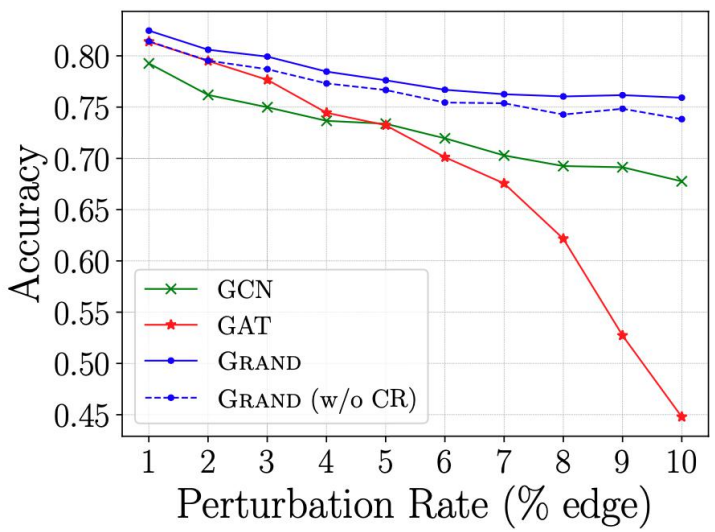
Ablation Study

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*

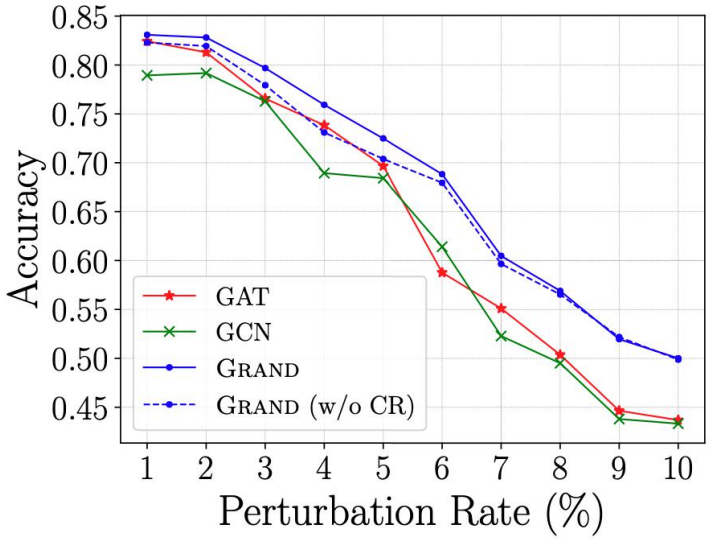
Random Propagation vs. Feature Propagation & Non-Linear Transformation

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

Results



(a) Random Attack



(b) Metattack

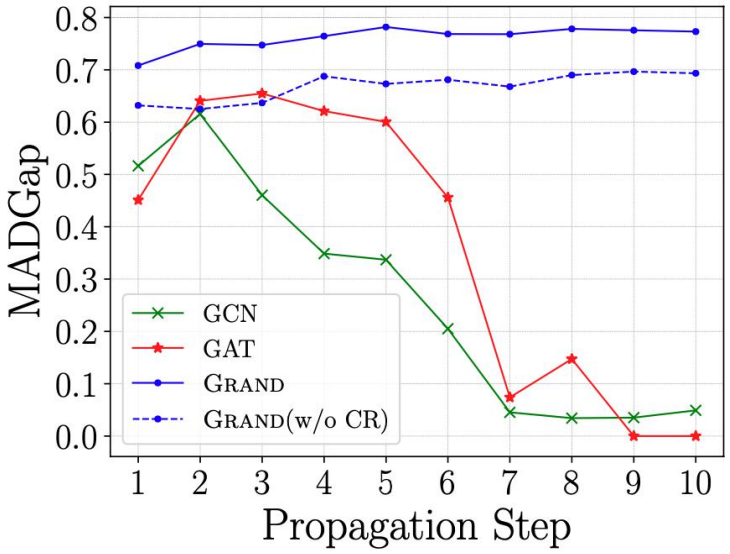
Robustness

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

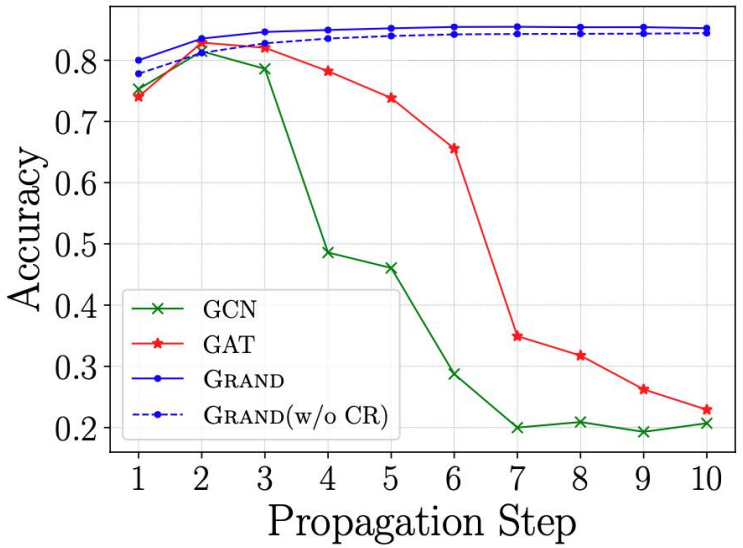
Random Propagation vs. Feature Propagation & Non-Linear Transformation

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

Results



(a) MADGap



(b) Classification Results

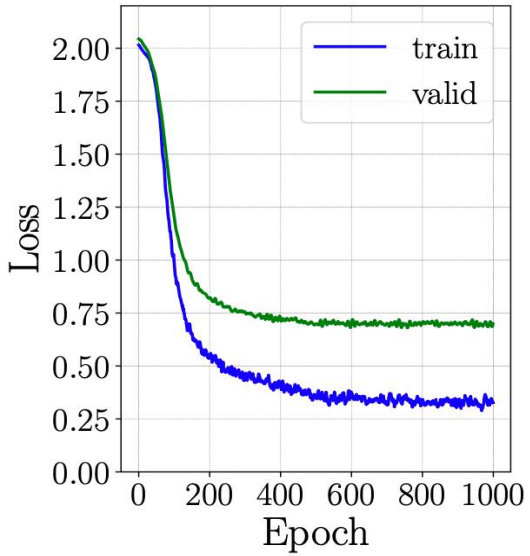
Over-Smoothing

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

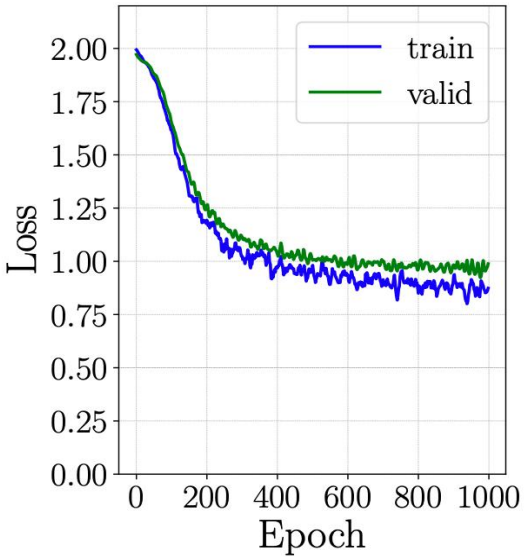
Random Propagation vs. Feature Propagation & Non-Linear Transformation

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

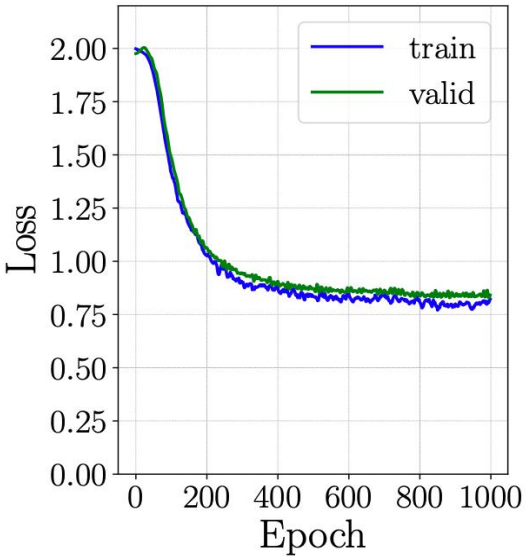
Results



(a) Without CR and RP



(b) Without CR



(c) GRAND

Generalization

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

- Feng et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. NeurIPS 2020
- Code & data for Grand: https://github.com/Grand20/grand

Graph Representation Learning

- Brief introduction of GNNs
- How to attend over heterogenous graphs?
- Do we really need message passing in GNNs?
- **Can we pre-train for graph representations?**

Remaining Challenges in Graph Learning

- Common issues in graph research
  - Usually expensive and even infeasible to access sufficient labeled data
  - Sometimes need to handle out-of-distribution predictions
- Solution: graph pre-training?
  - Great success in language & image pre-training, e.g., ELMO, BERT, Roberta, MoCo…

BERT for NLP

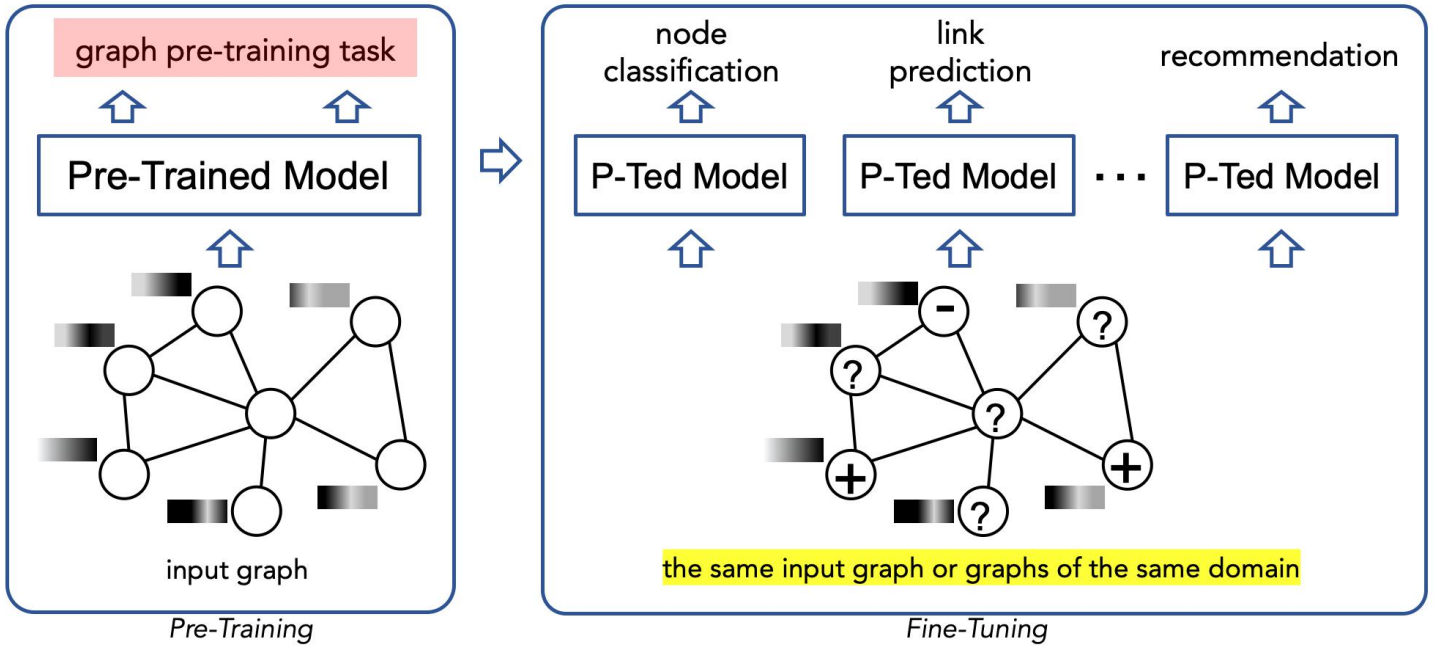
- Model level: Transformer
- Pre-training Task: MLM & NSP

Pre-training for Graphs

- Model level: graph neural nets?
- Pre-training Task: ?

GNN Pre-Training

- One graph pre-training setting:
  - To pre-train from large-scale attributed graphs
  - To fine-tune for unseen tasks on the same graph or graphs of the same domain.



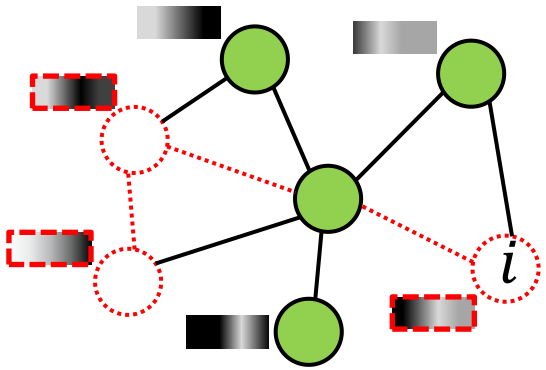
- How to do this?
  - Model level: GNNs, such as HGT
  - Pre-training task: **self-supervised** tasks on graphs?

1.Hu, Dong, Wang, Chang, Sun. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.

GPT-GNN: Generative Pre-Training of GNNs

- Model the graph distribution $p(G;\theta)$ by learning to reconstruct the input graph.
  - Factorize the graph likelihood into two terms:
    - Attribute Generation
    - Edge Generation

$$\log p_\theta(X,E) = \sum_{i=1}^{|\mathcal{V}|} \log p_\theta(X_i, E_i \mid X_{<i}, E_{<i}).$$



attribute and edge masked input graph

$$p_\theta(X_i, E_i | X_{<i}, E_{<i})$$
$$= p_\theta(X_i | X_{<i}, E_{<i}) \cdot p_\theta(E_i | X_{<i}, E_{<i})$$

?

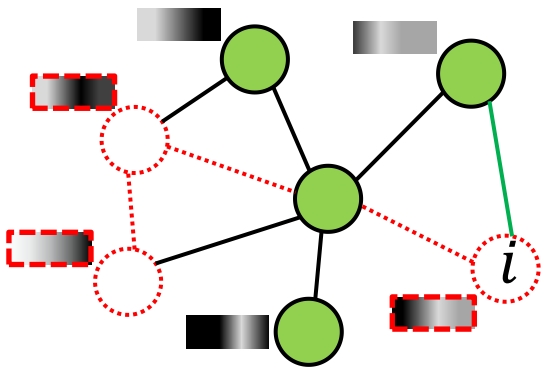
Lose the dependency between X_i and E_i

1.Hu, Dong, Wang, Chang, Sun. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.

GPT-GNN: Generative Pre-Training of GNNs

- Model the graph distribution $p(G;\theta)$ by learning to reconstruct the input graph.
  - Factorize the graph likelihood into two terms:
    - Attribute Generation: given observed edges, generate node attributes
    - Edge Generation: given observed edges and generated attributes, generate masked edges

$$\log p_\theta(X,E) = \sum_{i=1}^{|\mathcal{V}|} \log p_\theta(X_i, E_i \mid X_{<i}, E_{<i}).$$

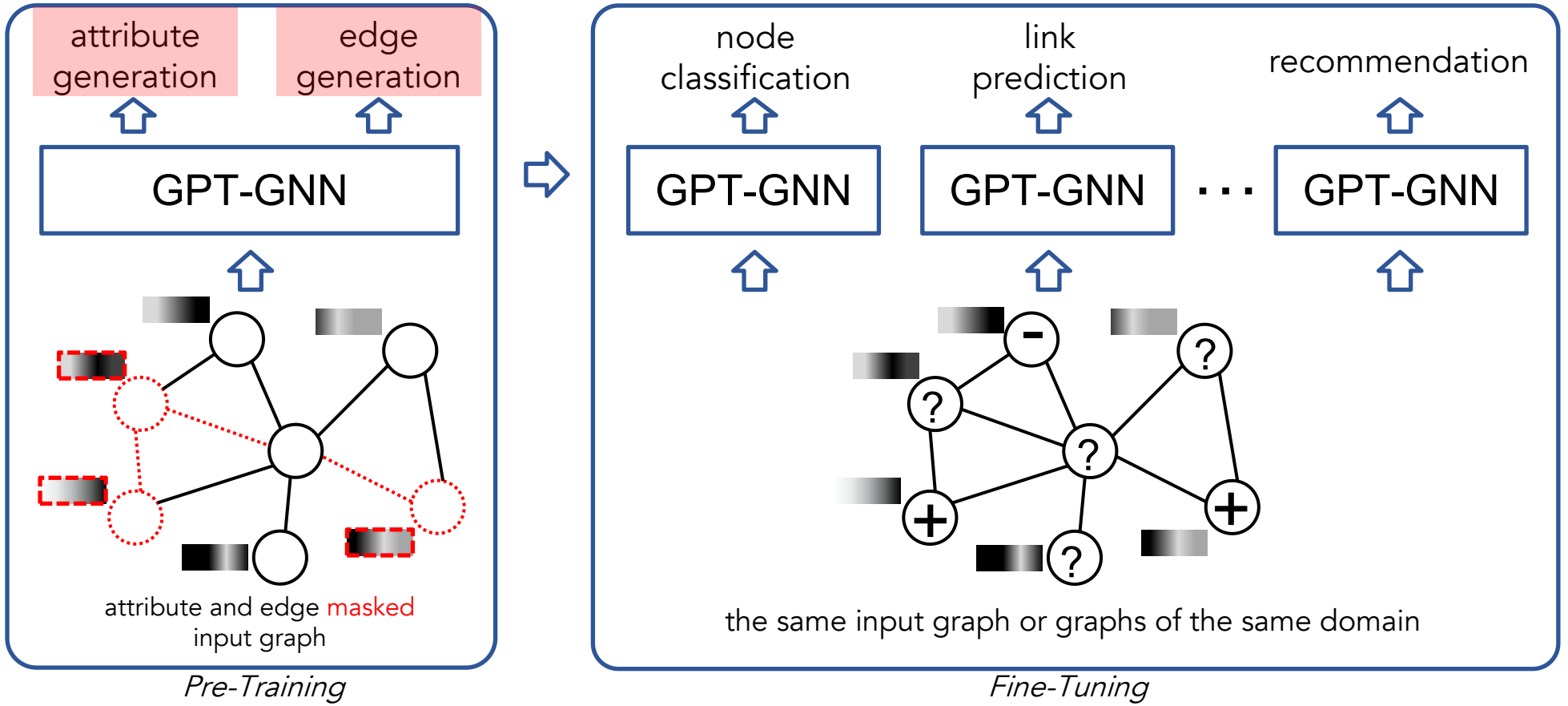


attribute and edge masked input graph

$$\begin{aligned}
&p_\theta(X_i, E_i \mid X_{<i}, E_{<i}) \\
=&\sum\nolimits_o p_\theta(X_i, E_{i,\neg o} \mid E_{i,o}, X_{<i}, E_{<i}) \cdot p_\theta(E_{i,o} \mid X_{<i}, E_{<i}) \\
=&\mathbb{E}_o\left[p_\theta(X_i, E_{i,\neg o} \mid E_{i,o}, X_{<i}, E_{<i})\right] \\
=&\mathbb{E}_o\Big[\underbrace{p_\theta(X_i \mid E_{i,o}, X_{<i}, E_{<i})}_{\text{1) generate attributes}} \cdot \underbrace{p_\theta(E_{i,\neg o} \mid E_{i,o}, X_{\leq i}, E_{<i})}_{\text{2) generate edges}}\Big].
\end{aligned}$$

1.Hu, Dong, Wang, Chang, Sun. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.

GPT-GNN: Generative Pre-Training of GNNs



1.Hu, Dong, Wang, Chang, Sun. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.

GPT-GNN: Generative Pre-Training of GNNs

- Data: Microsoft Academic Graph

	Pre-Train	Fine-Tune
Tasks:	• Attribute Generation • Edge Generation	• Inferring the topic of each paper • Inferring the venue of each paper • Author name disambiguation

Base GNN model: Heterogeneous Graph Transformer (HGT)

1.Hu, Dong, Wang, Chang, Sun. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.

GPT-GNN: Generative Pre-Training of GNNs

- Data: Microsoft Academic Graph

	Pre-Train	Fine-Tune
No Transfer:	CS Academic Graph	CS Academic Graph
Field Transfer:	Med, Bio, Physics…	CS Academic Graph
Time Transfer:	CS before 2014	CS after 2014
Time + Field Transfer:	Med, Bio, Physics… before 2014	CS after 2014

1.Hu, Dong, Wang, Chang, Sun. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.

GPT-GNN: Generative Pre-Training of GNNs

	Downstream Dataset	OAG		
	Evaluation Task	Paper–Field	Paper–Venue	Author ND
	No Pre-train	.346±.149	.598±.122	.813±.105
Field Transfer	GAE	.403±.114	.626±.093	.836±.084
	GraphSAGE (unsp.)	.368±.125	.609±.096	.818±.092
	Graph Infomax	.387±.112	.612±.097	.827±.084
	GPT-GNN (Attr)	.396±.118	.623±.105	.834±.086
	GPT-GNN (Edge)	.413±.109	.635±.096	.842±.093
	GPT-GNN	**.420±.107**	**.641±.098**	**.848±.102**
Time Transfer	GAE	.384±.117	.619±.101	.828±.095
	GraphSAGE (unsp.)	.352±.121	.601±.105	.815±.093
	Graph Infomax	.369±.116	.606±.102	.821±.089
	GPT-GNN (Attr)	.374±.114	.614±.098	.826±.089
	GPT-GNN (Edge)	.397±.105	.629±.102	.836±.088
	GPT-GNN	**.405±.108**	**.635±.101**	**.840±.093**
Time + Field Transfer	GAE	.371±.124	.611±.108	.821±.102
	GraphSAGE (unsp.)	.349±.130	.602±.118	.812±.097
	Graph Infomax	.360±.121	.600±.102	.815±.093
	GPT-GNN (Attr)	.364±.115	.609±.103	.824±.094
	— (w/o node separation)	.347±.128	.601±.102	.813±.108
	GPT-GNN (Edge)	.390±.116	.622±.104	.830±.105
	— (w/o adaptive queue)	.376±.121	.617±.115	.828±.104
	GPT-GNN	**.397±.112**	**.628±.108**	**.833±.102**

- **All pre-training frameworks** help the performance of GNNs
  - GAE, GraphSage, Graph Infomax
  - GPT-GNN
- **GPT-GNN helps the most** by achieving a relative performance gain of 9.1% over the base model without pre-training
- **Both self-supervised tasks in GPT-GNN** help the pre-training framework
  - Attribute generation
  - Edge generation

1.Hu, Dong, Wang, Chang, Sun. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.

GPT-GNN: Generative Pre-Training of GNNs

- Data: Microsoft Academic Graph

	Pre-Train	Fine-Tune
Tasks:	• Attribute Generation • Edge Generation	• Inferring the topic of each paper • Inferring the venue of each paper • Author name disambiguation

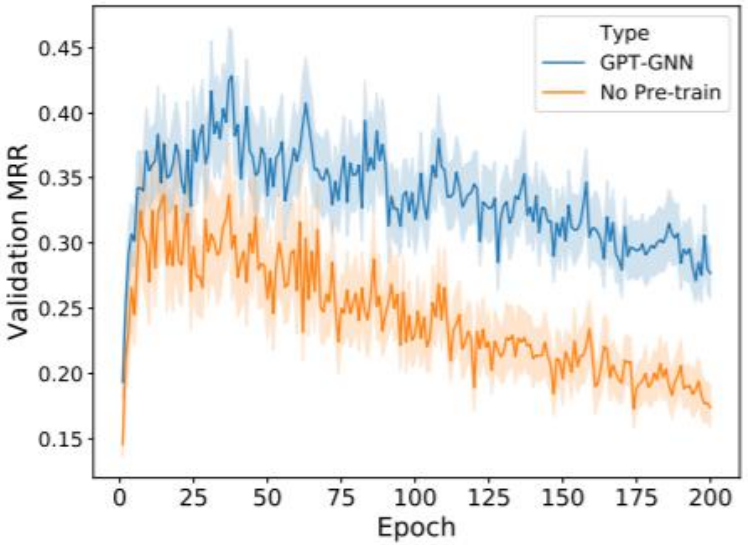
Base GNN model: Heterogeneous Graph Transformer (HGT)

Model	HGT	GCN	GAT	RGCN	HAN
No Pre-train	**.346**	.327	.318	.296	.332
GPT-GNN	**.420**	.359	.382	.351	.406
Relative Gain	21.4%	9.8%	20.1%	18.9%	22.3%

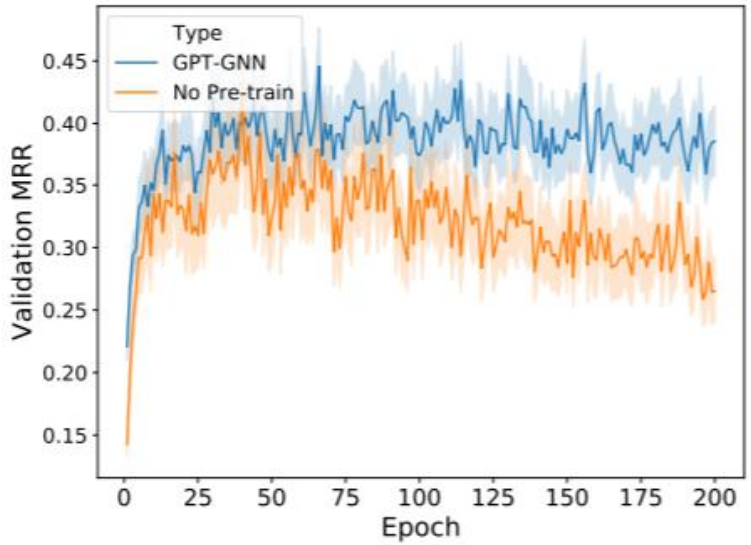
1.Hu, Dong, Wang, Chang, Sun. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.

Downstream Dataset	OAG (citation)	Reddit
No Pre-train	.281±.087	.873±.036
GAE	.296±.095	.885±.039
GraphSAGE (unsp.)	.287±.093	.880±.042
Graph Infomax	.291±.086	.877±.034
GPT-GNN	**.309±.081**	**.896±.028**

The Promise of Generative Pre-training



(a) Data Percentage: 10%

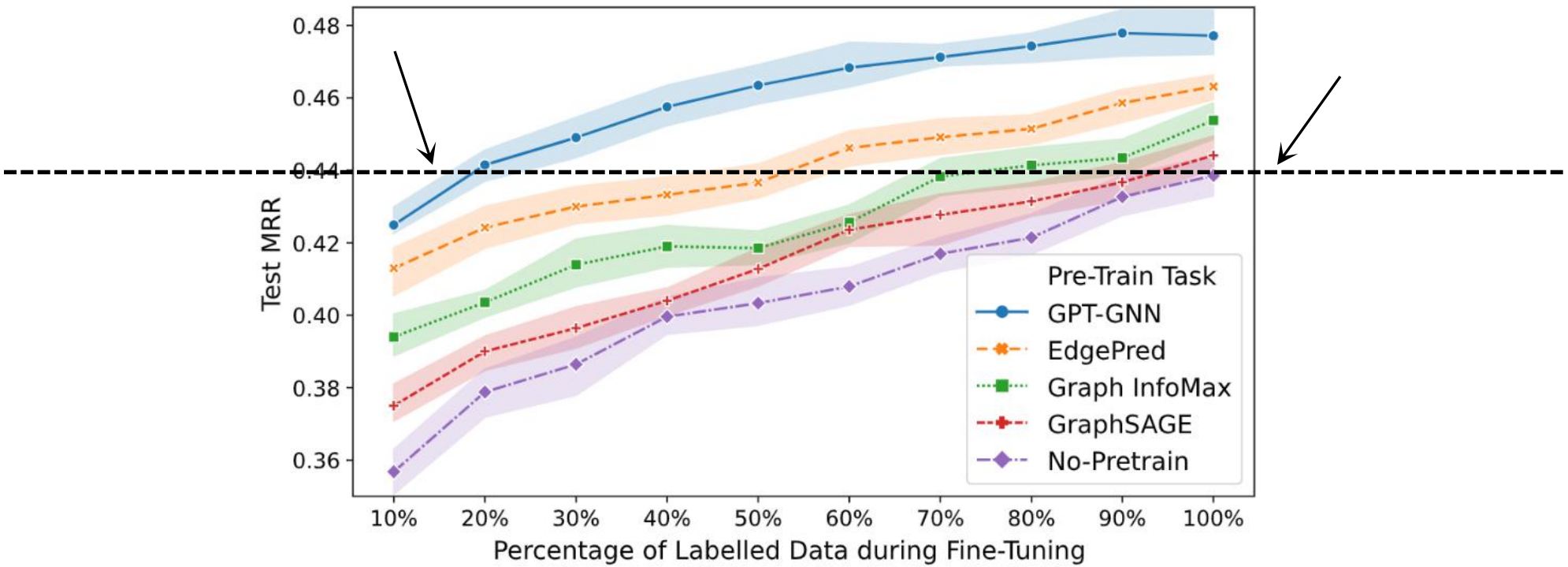


(b) Data Percentage: 20%

Predict Paper Title	GroundTruth Paper Title
person recognition system using automatic probabilistic classification	person re-identification by probabilistic relative distance comparison
a novel framework using spectrum sensing in wireless systems	a secure collaborative spectrum sensing strategy in cyber physical systems
a efficient evaluation of a distributed data storage service storage	an empirical analysis of a large scale mobile cloud storage service
parameter control in wireless sensor networks networks networks	optimal parameter estimation under controlled communication over sensor networks
a experimental system for for to the analysis of graphics	an interactive computer graphics approach to surface representation

The Promise of Graph Pre-Training!

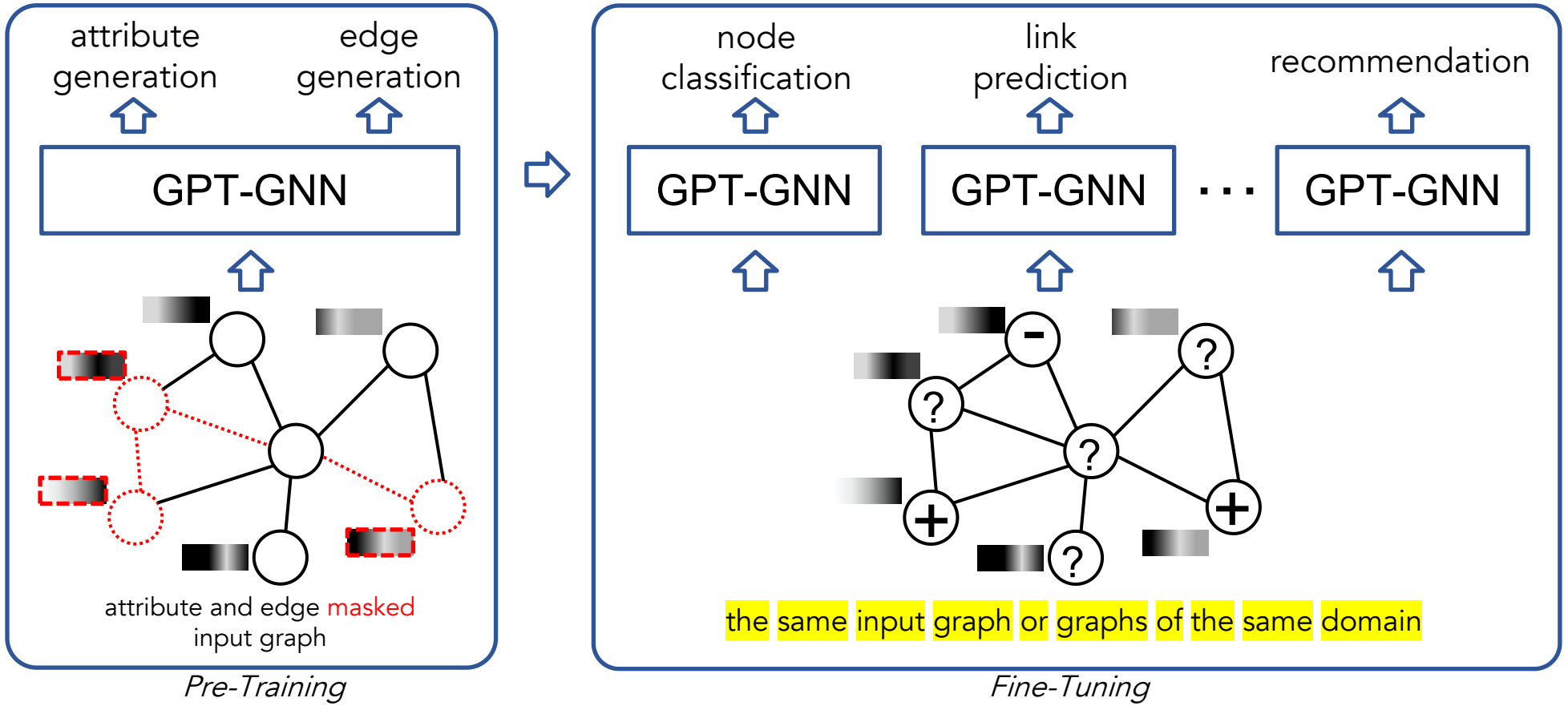
- During fine-turning



The GNN model **w/o** pre-training by using **100%** training data
VS
The GNN model **with** pre-training by using **10-20%** training data

1.Hu, Dong, Wang, Chang, Sun. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.

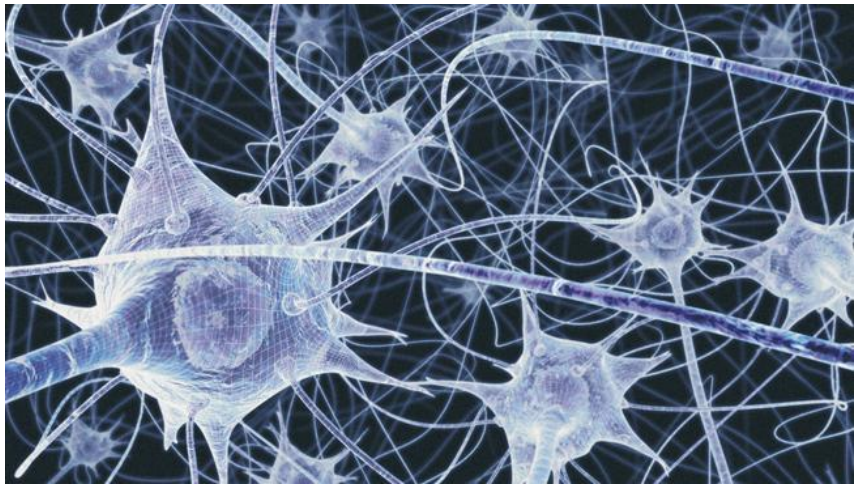
GNN Pre-Training on the “Same” Networks



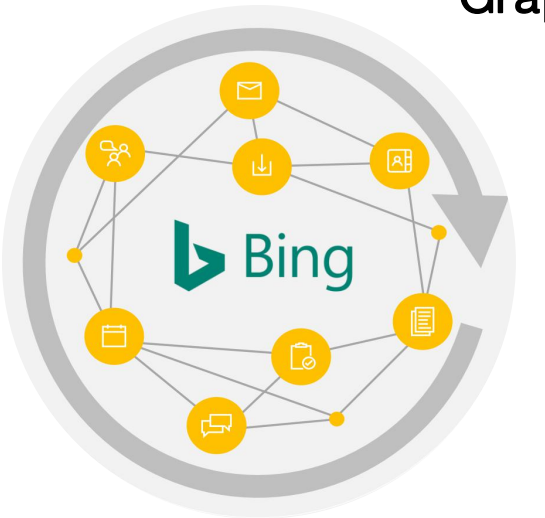
1.Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2.Code & data for GPT-GNN: https://github.com/acbull/GPT-GNN

Graphs

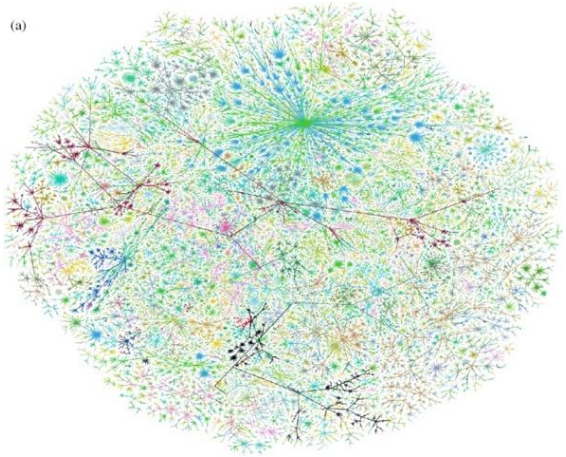
Office/Social Graph



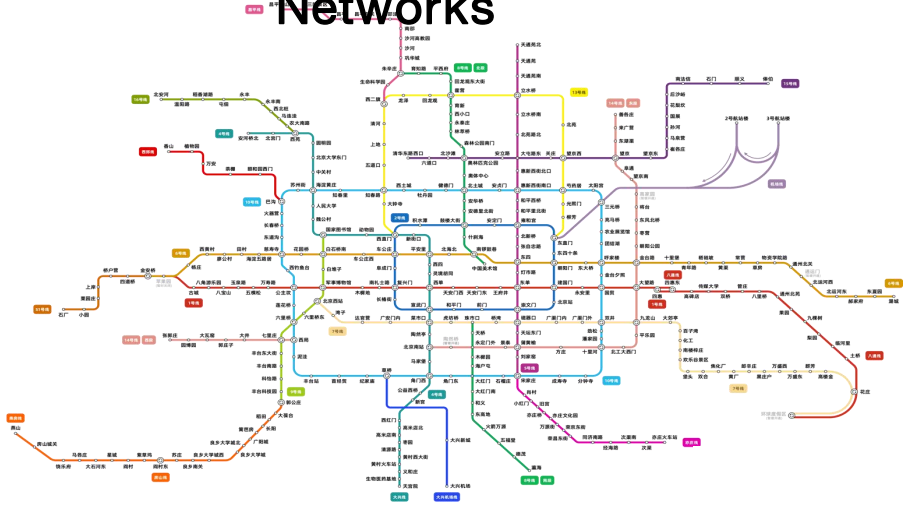
Biological Neural Networks



Knowledge Graph



Internet

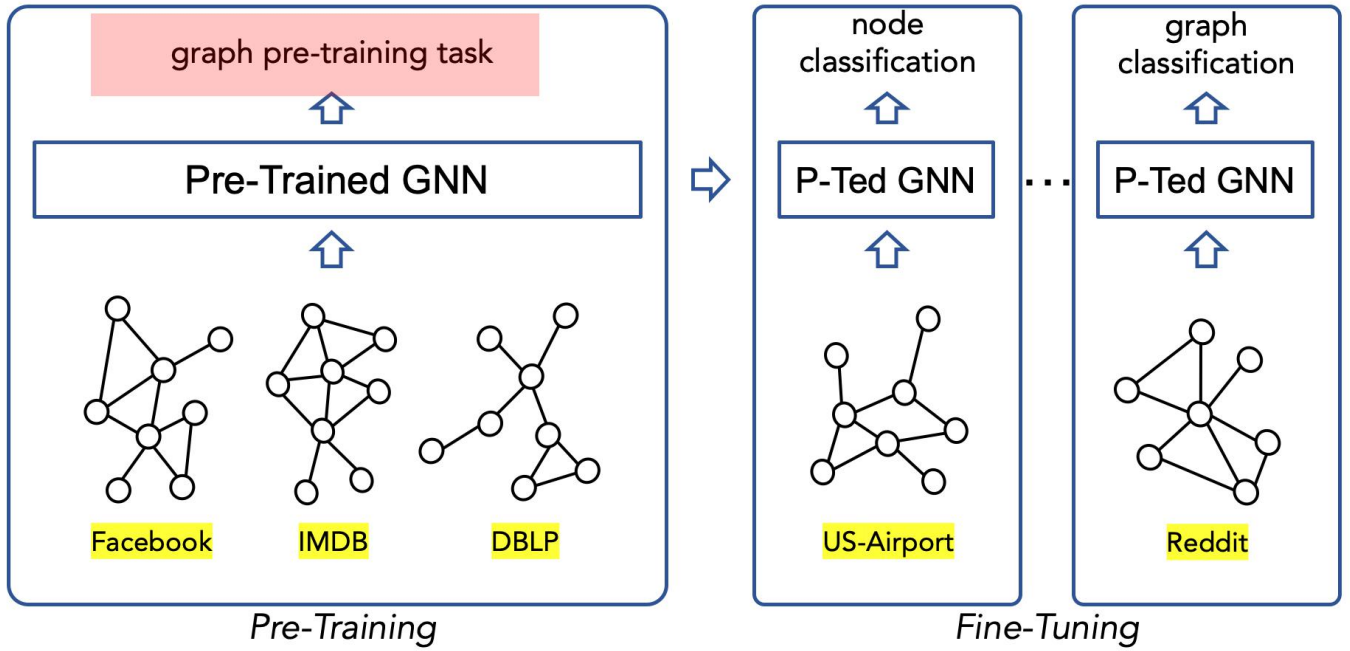


Transportation

figure credit: Web

GNN Pre-Training

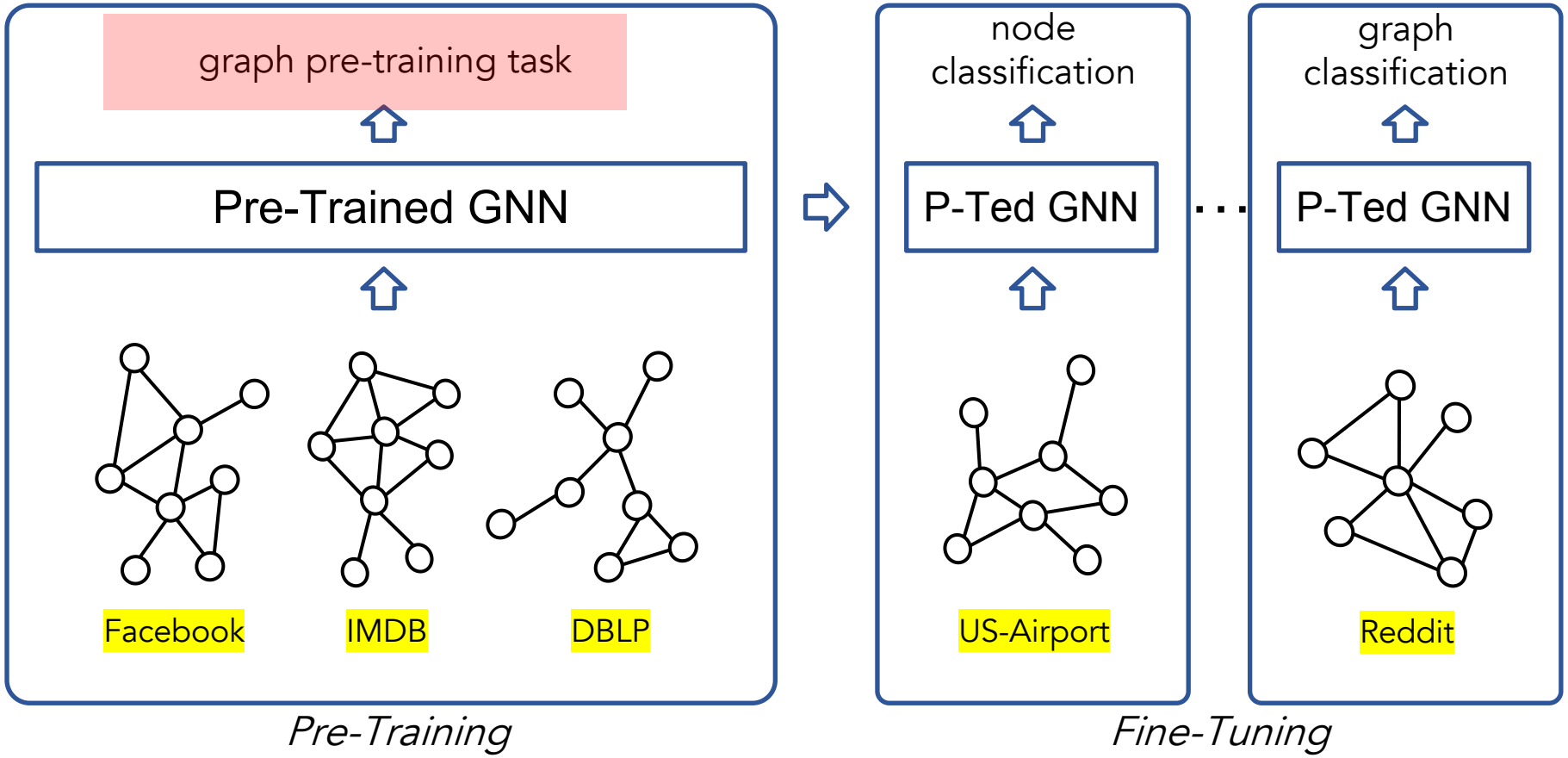
- The SECOND graph pre-training setting:
  - To pre-train from some graphs
  - To fine-tune for unseen tasks on unseen graphs



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

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



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

- 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?

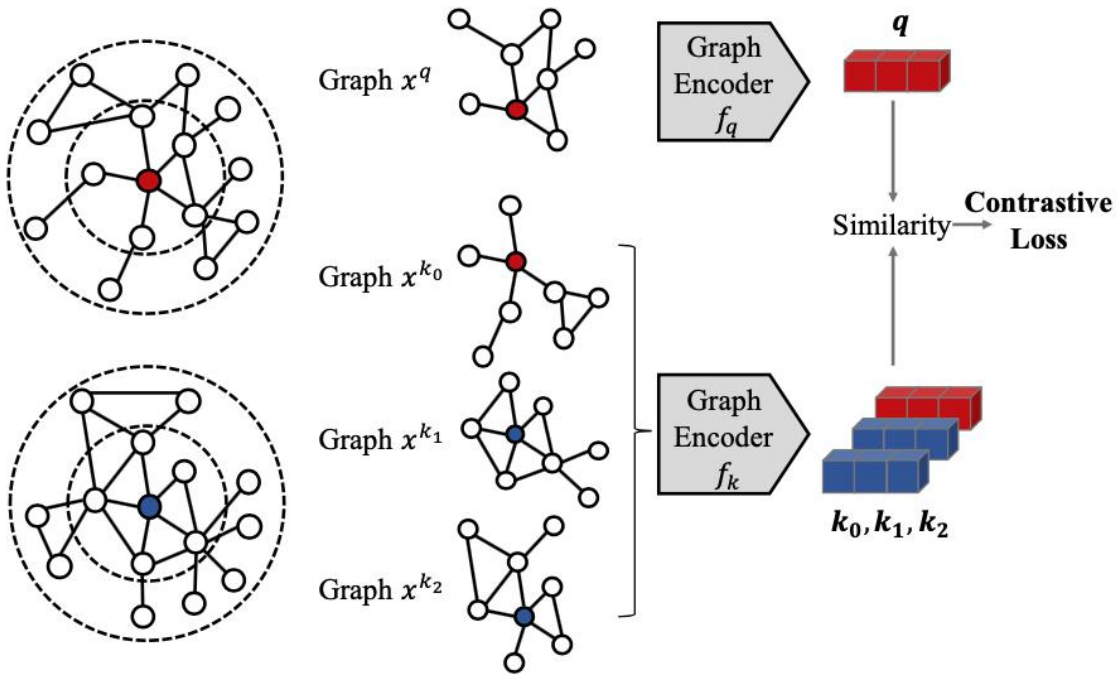
1. Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In CVPR '18.
2. Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In CVPR '20.

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 encoders?

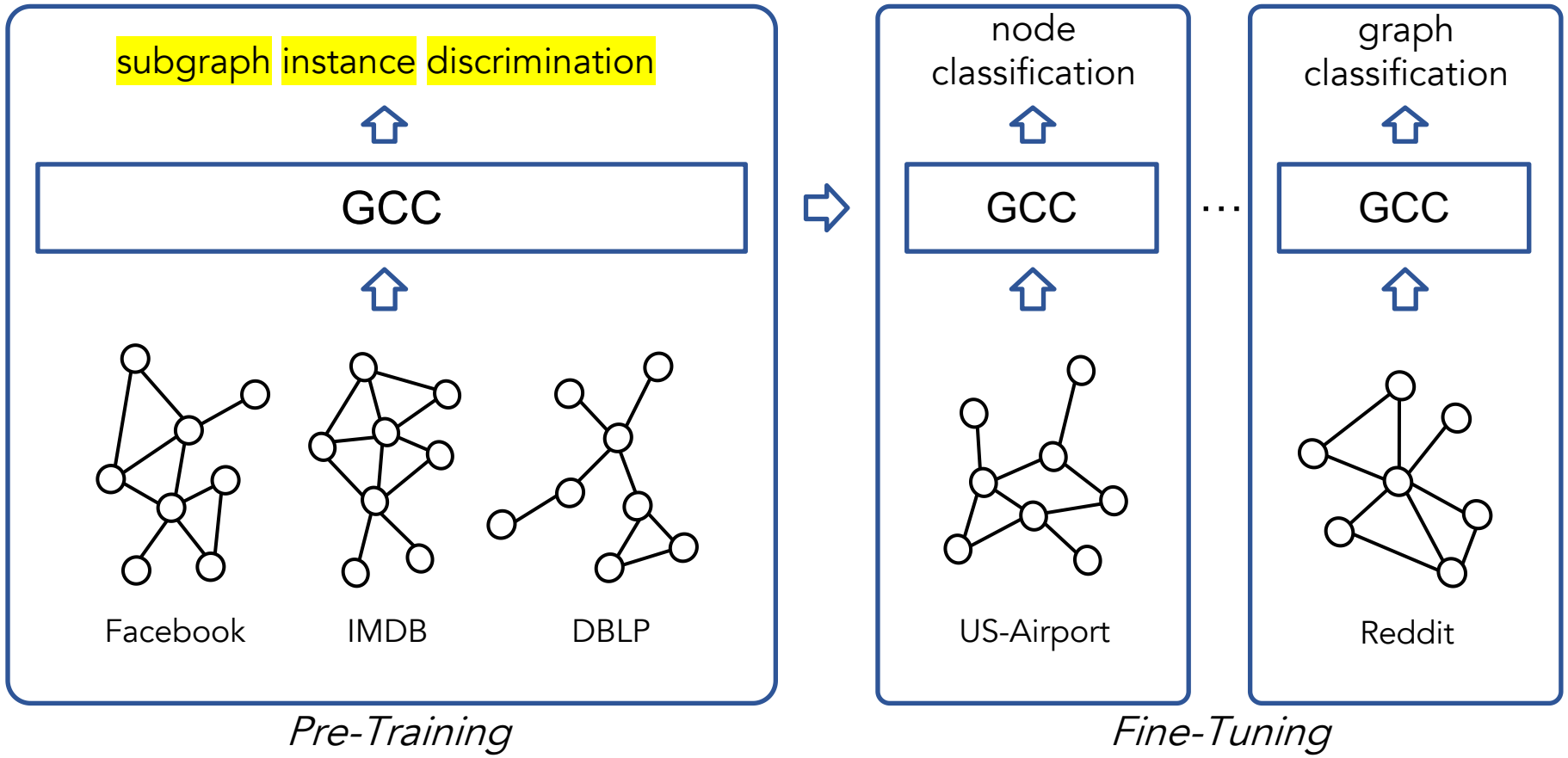
$$\mathcal{L} = -\log \frac{\exp\left(\boldsymbol{q}^\top \boldsymbol{k}_+ / \tau\right)}{\sum_{i=0}^{K} \exp\left(\boldsymbol{q}^\top \boldsymbol{k}_i / \tau\right)}$$

Subgraph instance discrimination



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

Graph Contrastive Coding (GCC)



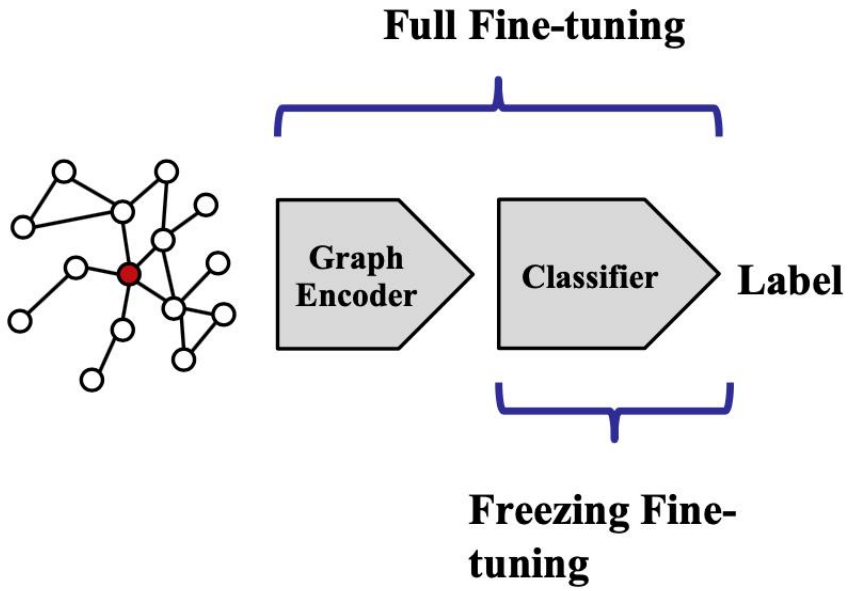
1.Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
2.Code & Data for GCC: https://github.com/THUDM/GCC

GCC Pre-Training / Fine-Tuning

- pre-train on six graphs

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, IMDB-M
    - Graph classification
  - AMiner academic graph
    - Similarity search
- The base GNN
  - Graph Isomorphism Network (GIN)



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

Results

Node Classification

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**

Similarity Search

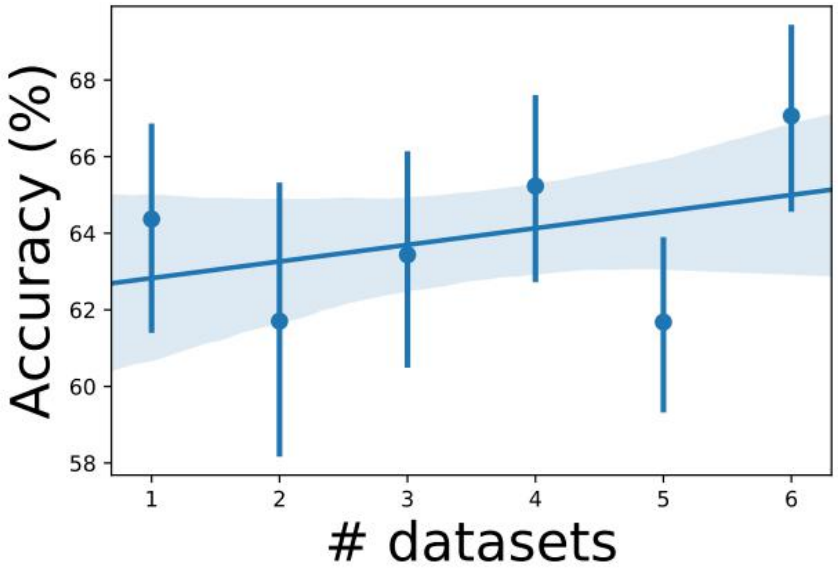
	KDD-ICDM		SIGIR-CIKM		SIGMOD-ICDE	
$\|V\|$	2,867	2,607	2,851	3,548	2,616	2,559
$\|E\|$	7,637	4,774	6,354	7,076	8,304	6,668
# groud truth		697		874		898
k	20	40	20	40	20	40
Random	0.0198	0.0566	0.0223	0.0447	0.0221	0.0521
RolX	0.0779	0.1288	0.0548	0.0984	0.0776	0.1309
Panther++	0.0892	0.1558	**0.0782**	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

Graph Classification

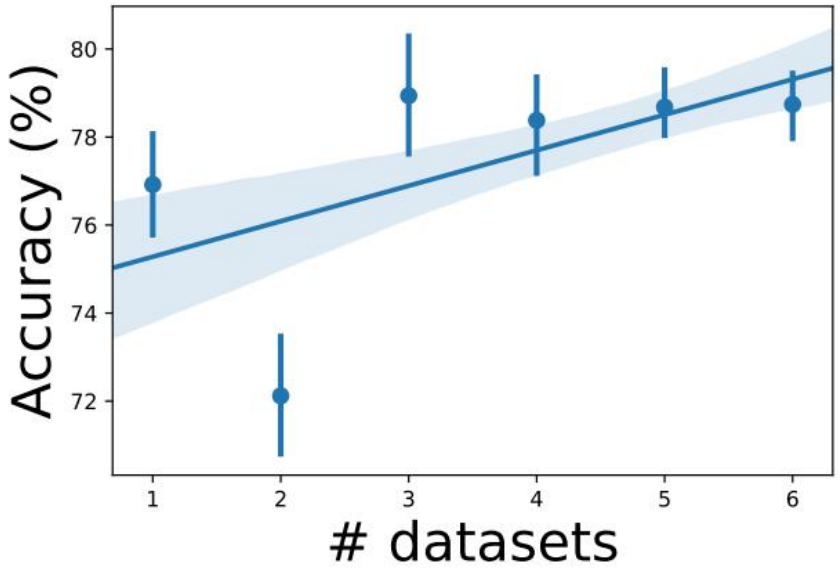
Datasets	IMDB-B	IMDB-M	COLLAB	RDT-B	RDT-M
# graphs	1,000	1,500	5,000	2,000	5,000
# classes	2	3	3	2	5
Avg. # nodes	19.8	13.0	74.5	429.6	508.5
DGK	67.0	44.6	73.1	78.0	41.3
graph2vec	71.1	50.4	–	75.8	47.9
InfoGraph	**73.0**	**49.7**	–	82.5	53.5
GCC (E2E, freeze)	71.7	49.3	74.7	87.5	52.6
GCC (MoCo, freeze)	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

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

Results



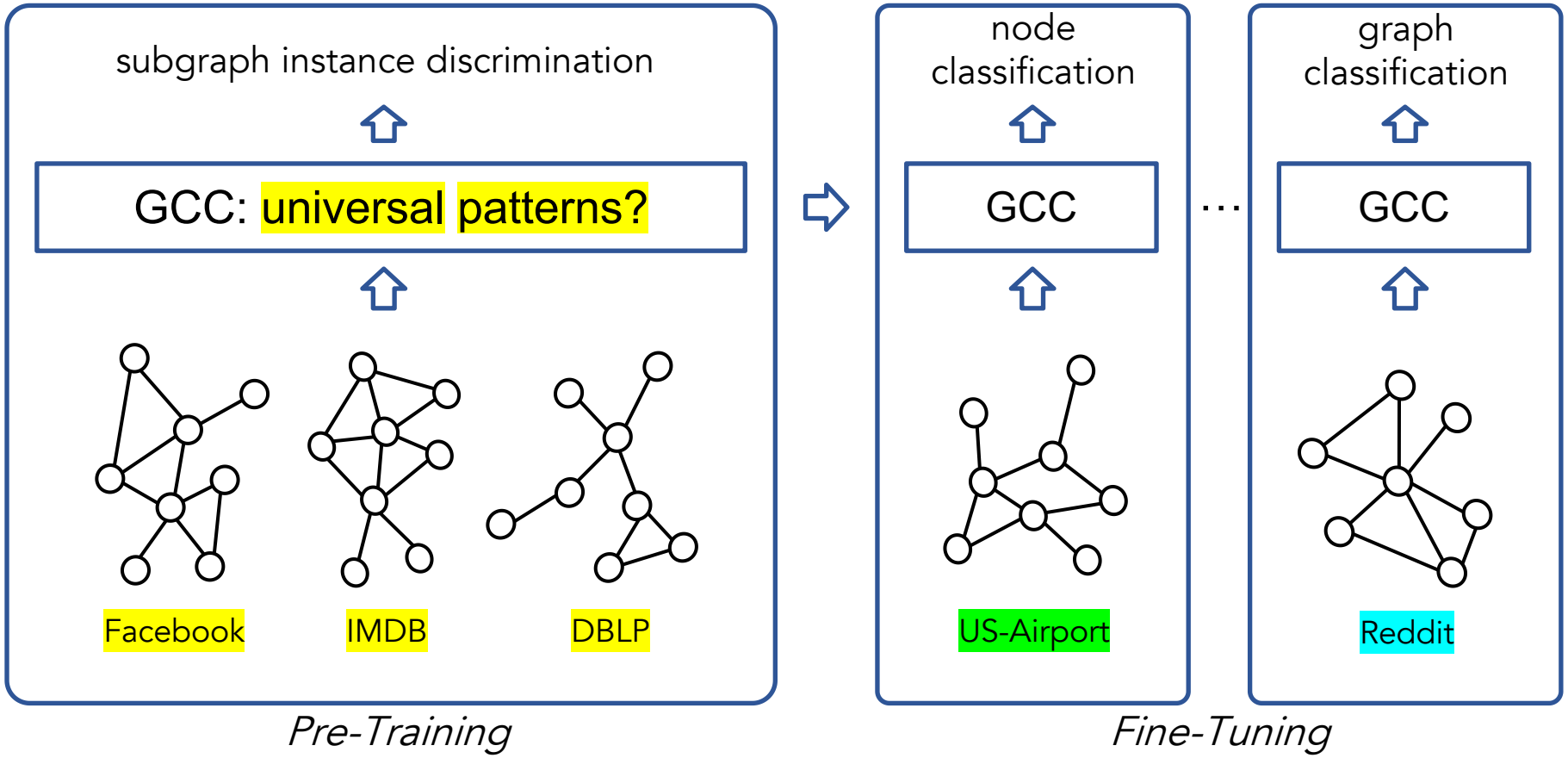
(a) **US-Airport:** $y = 0.4344x + 62.394$



(b) **COLLAB:** $y = 0.8065x + 74.4737$

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

Results



Does the pre-training of GNNs learn the **universal structural patterns** across networks?

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

Graph Representation Learning

- Brief introduction of GNNs
- How to attend over heterogenous graphs?
- Do we really need message passing in GNNs?
- Can we pre-train for graph representations?

Graph Representation Learning

Network Embedding ⇔ Matrix Factorization

Graph Neural Networks ⇨ GNN Pre-Training

Graph Data & Benchmarks

OPEN GRAPH BENCHMARK

https://alchemy.tencent.com/

Miner

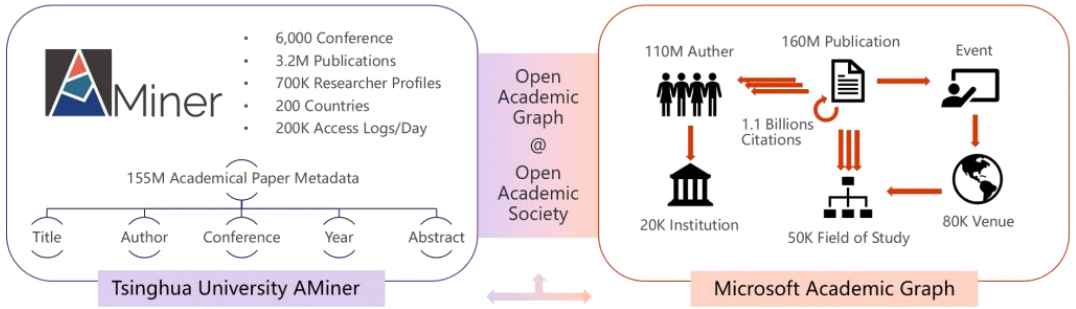
Microsoft Academic

CogDL

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.

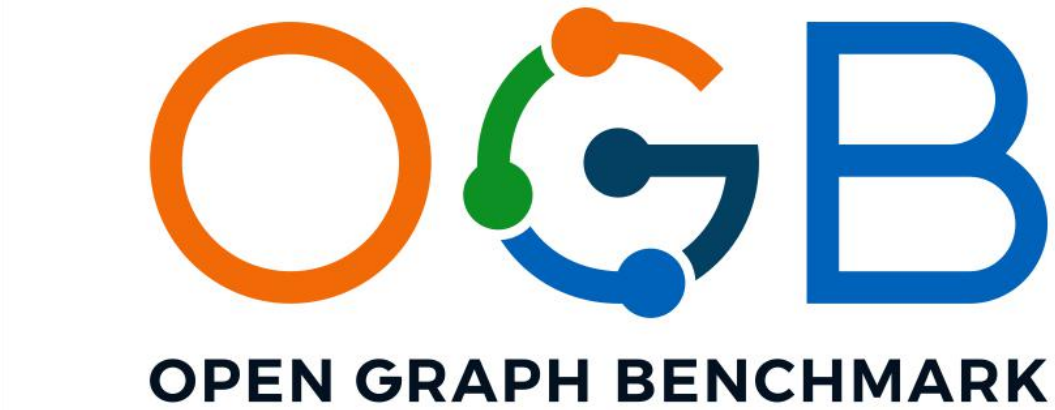
Overview of OAG v2

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.

1. Zhang, et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. In KDD'19.

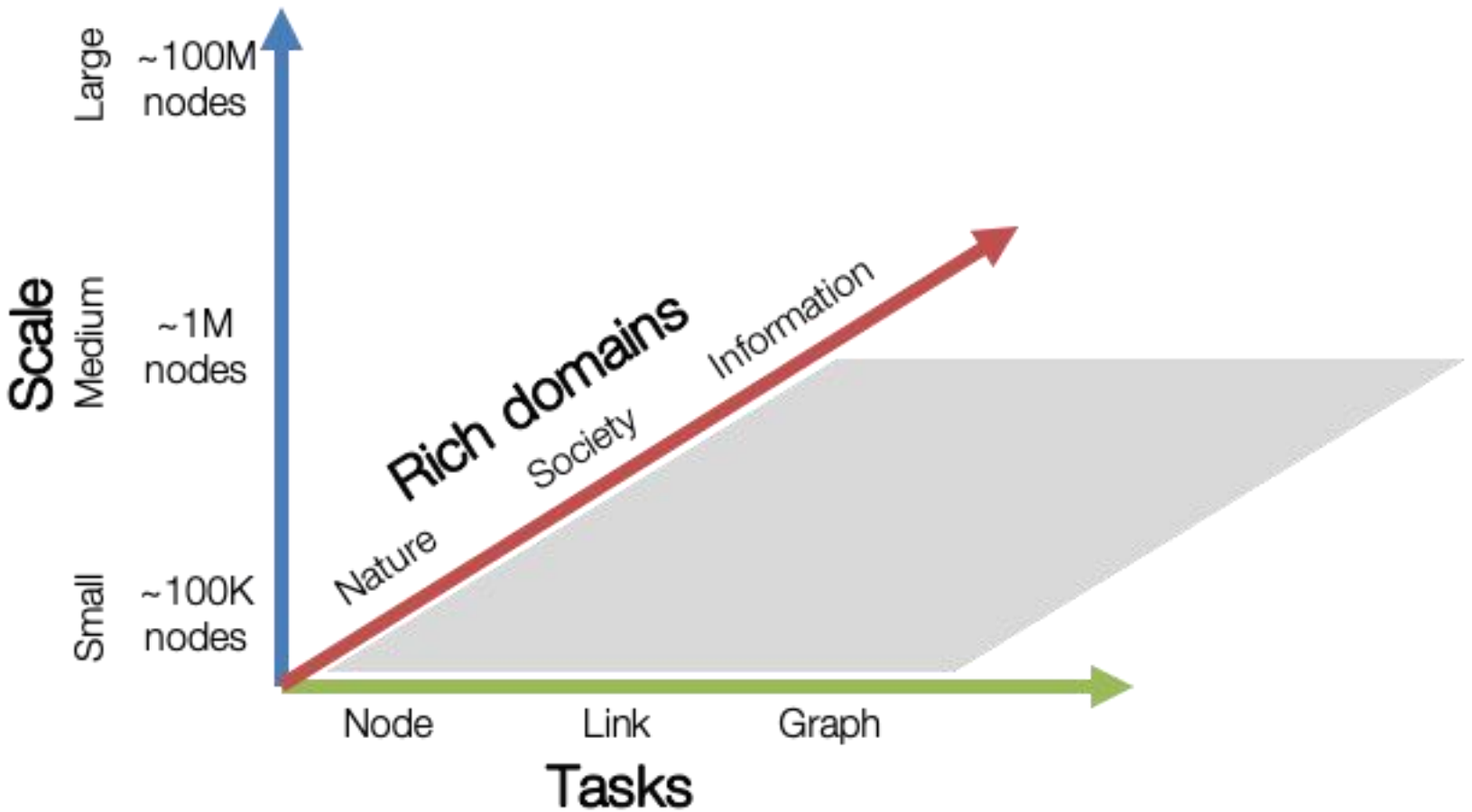
Open Graph Benchmark

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



Paper: https://arxiv.org/abs/2005.00687, NeurIPS 2020
Leaderboards: https://ogb.stanford.edu/

OGB Datasets are Diverse



Open Graph Benchmark

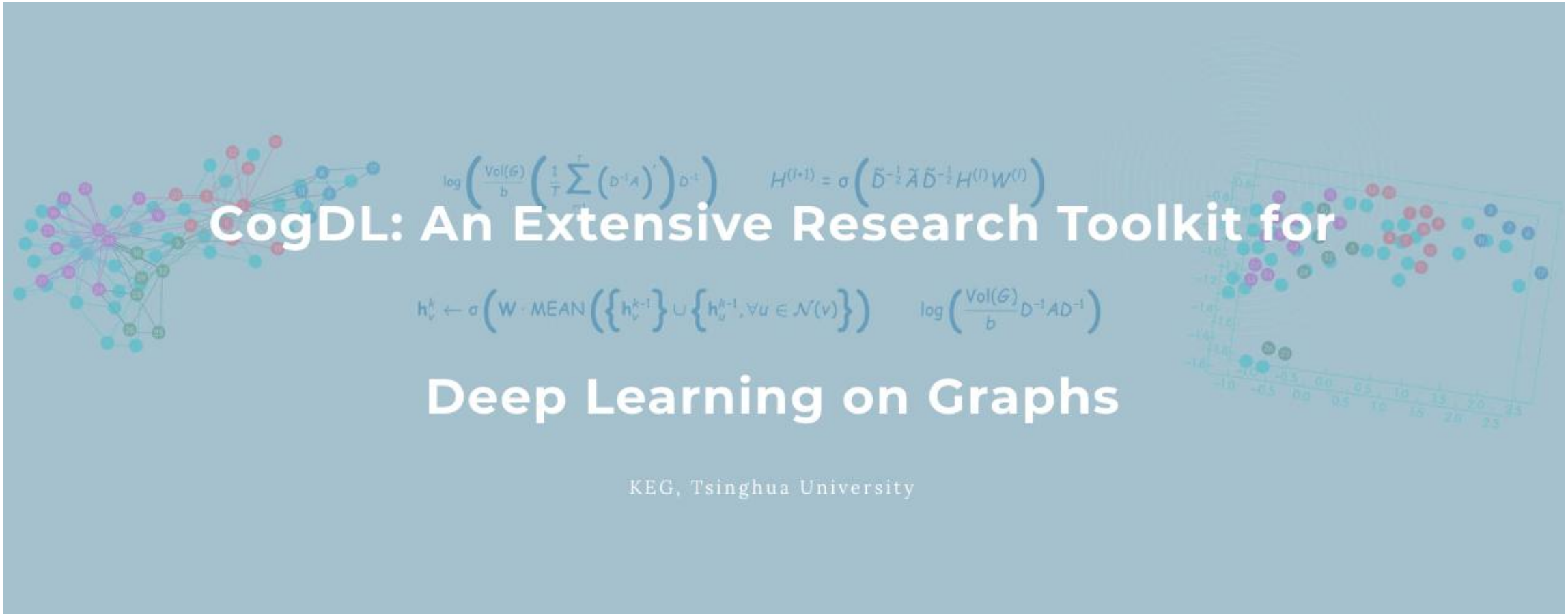
Resource for graph ML problems

We envision OGB to be:

- Common, community-driven platform for graph ML research
- Teaching resource

Slides from Weihua Hu, Stanford University

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		✓	✓	✓	

Graph Representation Learning

Network Embedding ⇔ Matrix Factorization

Graph Neural Networks ⇨ GNN Pre-Training

Graph Data & Benchmarks

OPEN GRAPH BENCHMARK

https://alchemy.tencent.com/

AMiner

Microsoft Academic

CogDL

github.com/thudm/cogdl

References

1. Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.
2. Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. **KDD** 2020.
3. Weihua Hu et al. Open Graph Benchmark: Datasets for Machine Learning on Graphs. **NeurIPS** 2020.
4. Feng et al. Graph Random Neural Networks. **NeurIPS** 2020.
5. Ziniu Hu et al. Heterogeneous Graph Transformer. **WWW** 2020.
6. Kuansan Wang et al. Microsoft Academic Graph: When experts are not enough. Quantitative Science Studies 1 (1), 396-413, 2020.
7. Yuxiao Dong et al. Heterogeneous Network Representation Learning. **IJCAI** 2020.
8. Jiezhong Qiu et al. *NetSMF: Large-Scale Network Embedding as Sparse Matrix Factorization*. **WWW** 2019.
9. Jie Zhang et al. *ProNE: Fast and Scalable Network Representation Learning*. **IJCAI** 2019.
10. Fanjing Zhang et al. OAG: Toward Linking Large-scale Heterogeneous Entity Graphs. ACM KDD 2019.
11. Xian Wu et al. Neural Tensor Decomposition. WSDM 2019.
12. Jiezhong Qiu et al. *Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec.* **WSDM** 2018.
13. Yuxiao Dong et al. *metapath2vec: Scalable Representation Learning for Heterogeneous Networks*. **KDD** 2017.
14. Perozzi et al. DeepWalk: Online learning of social representations. In **KDD**' 14.
15. Tang et al. LINE: Large scale information network embedding. In **WWW'15**.
16. Grover and Leskovec. node2vec: Scalable feature learning for networks. In **KDD'16**.
17. Harris, Z. (1954). Distributional structure. ***Word***, 10(23): 146-162.
18. Kipf et al. Semisupervised Classification with Graph Convolutional Networks. **ICLR** 2017
19. Velickovic et al. Graph Attention Networks. **ICLR** 2018
20. Hamilton et al. Inductive Representation Learning on Large Graphs. **NeurIPS** 2017
21. Defferrard et al. Convolutional Neural Networks on Graphs with Fast Locailzied Spectral Filtering. In **NeurIPS** 2016
22. Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. **NAACL-HLT** 2019.
23. Justin Gilmer, et al. Neural message passing for quantum chemistry. arXiv: 2017.
24. Kaiming He, et al. Momentum contrast for unsupervised visual representation learning. arXiv: 2019
25. Tomas Mikolov et al. Distributed representations of words and phrases and their compositionality. **NeurIPS** 2013.
26. Petar Velickovic et al. Deep Graph Infomax. In **ICLR** 19.
27. Zhen Yang et al. Understanding Negative Sampling in Graph Representation Learning. **KDD** 2020.

Thank You!