WWW'23 Tutorial

Self-supervised Learning and Pre-training on Graphs (GNNs)

Yukuo Cen, Yuxiao Dong, Jie Tang

Knowledge Engineering Group (KEG)

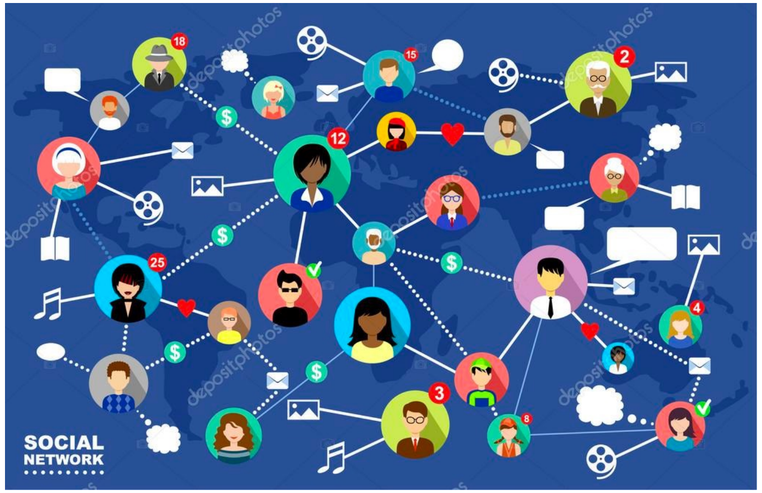
Department of Computer Science and Technology

Tsinghua University

Download the slides here

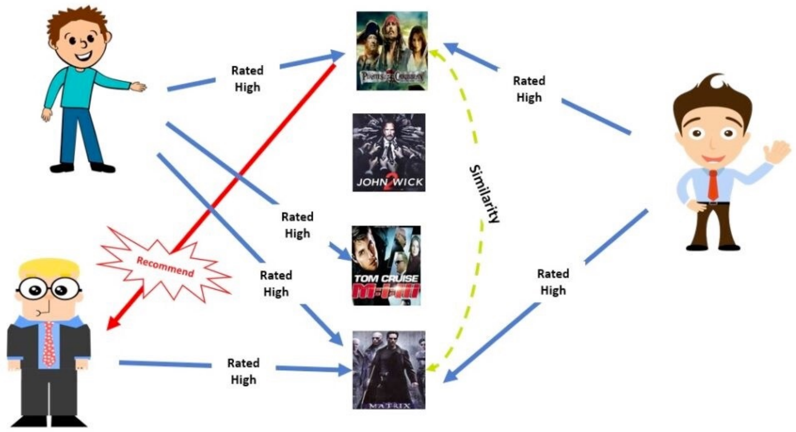
Graph

- Graph data exists everywhere



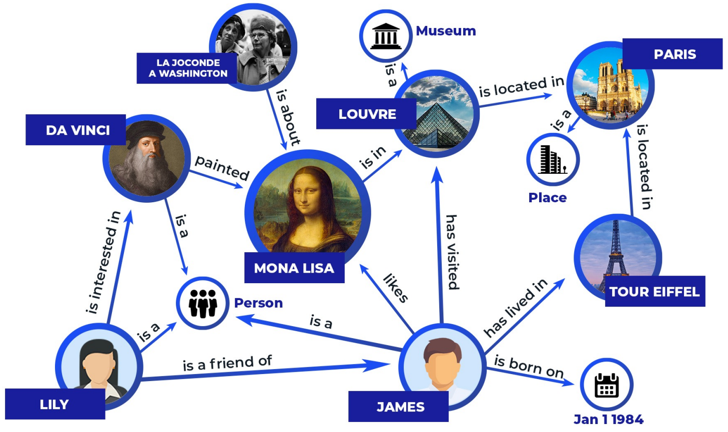
Social Network

- WeChat: 1.2 billion users
  61 billion links



Recommender System

- Alibaba: 2.3 billion trans. on 11/11



Knowledge Graph

- Wikidata: >1.4 billion triples

*"The number of **graph neural network** papers in this journal has grown as the field matures. We take a closer look at some of the **scientific applications**."*

1. The graph connection. *Nature Machine Intelligence* **4**, 187–188 (2022). https://doi.org/10.1038/s42256-022-00476-6

Machine Learning on Graphs

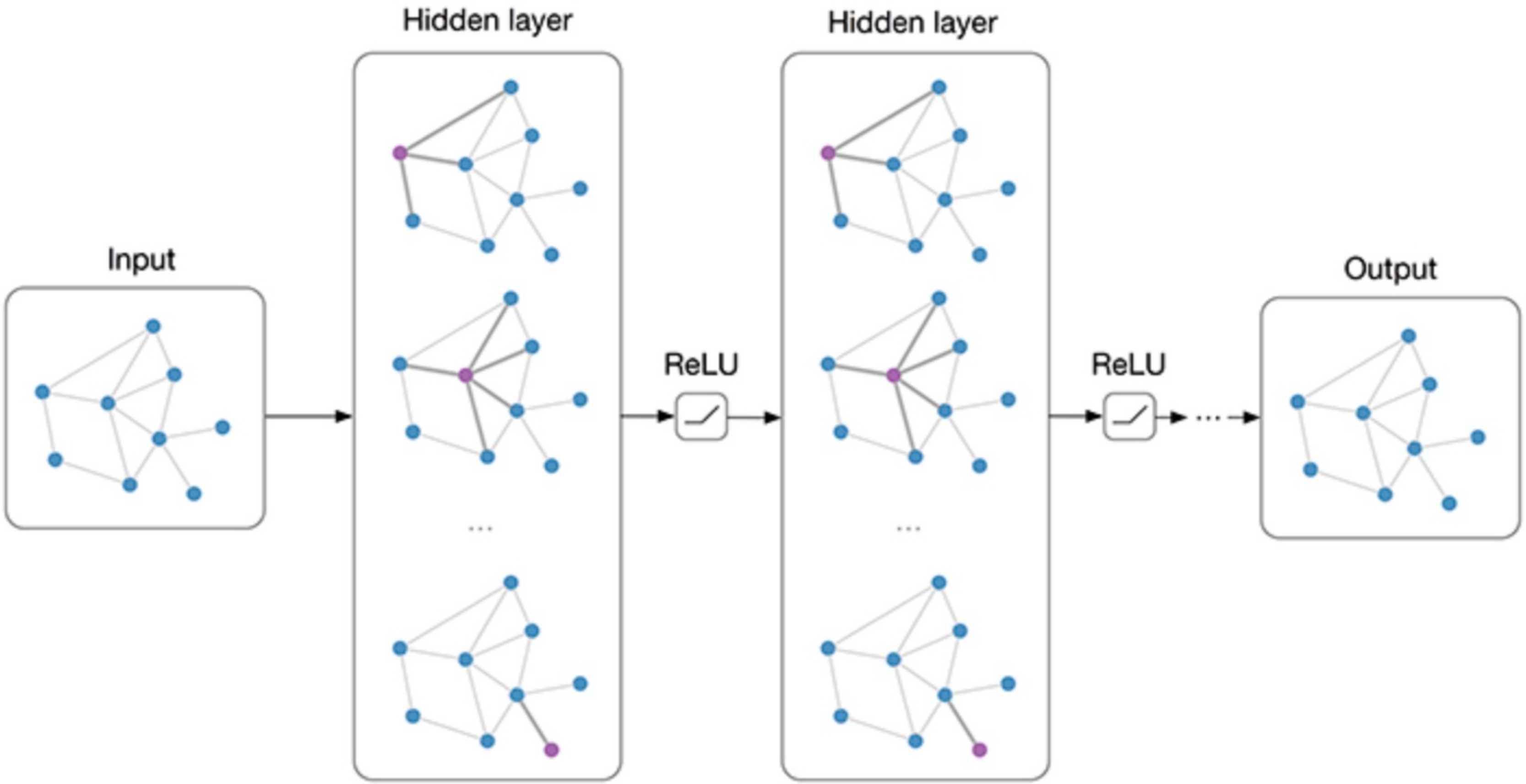
- ML tasks on Graphs:
  - Node classification
    - Predict a type of a given node
  - Link prediction
    - Predict whether two nodes are linked
  - Graph classification
    - Predict the properties of molecules
  - Community detection
    - Identify densely linked clusters of nodes

Learning on Graphs with Graph Neural Networks (GNNs)

- A question: Are you using GNNs?

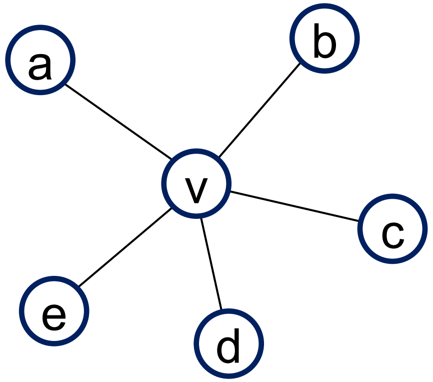
Graph Neural Networks

- Layer-wise propagation: $f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$



Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In ICLR '17.

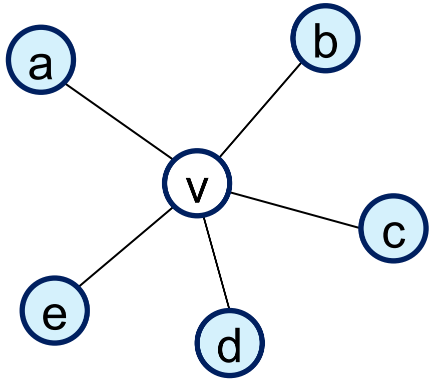
Graph Neural Networks



$$\boldsymbol{h}_v = f(\boldsymbol{h}_a, \boldsymbol{h}_b, \boldsymbol{h}_c, \boldsymbol{h}_d, \boldsymbol{h}_e)$$

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

GCN: Graph Convolutional Networks



parameters in layer k

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

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

node v's embedding at layer k

the neighbors of node v

1. Kipf et al. Semi-supervised Classification with Graph Convolutional Networks. ICLR 2017

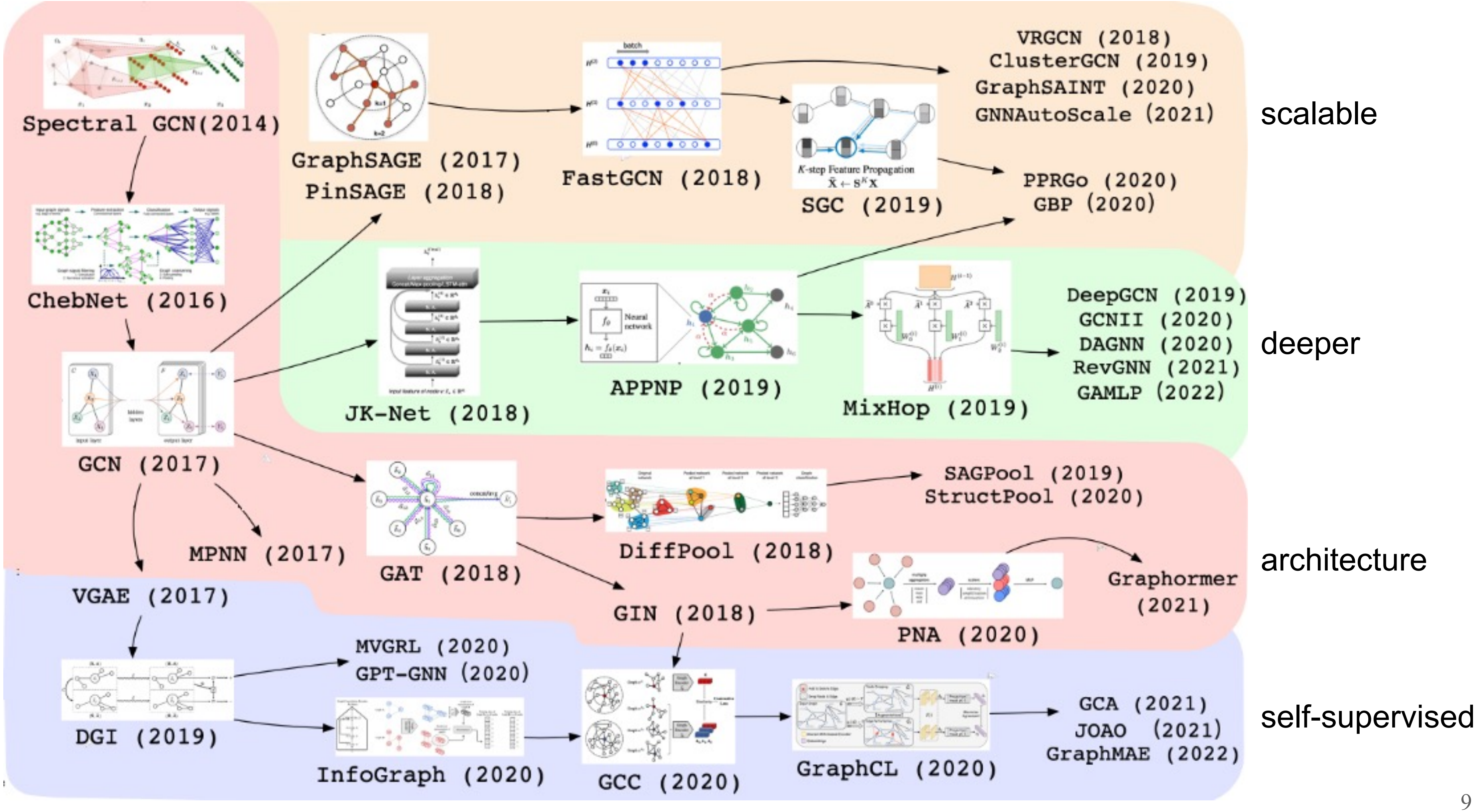
GCN Performance

- 2-layer GCN： $\boldsymbol{Z} = \mathrm{softmax}\left(\tilde{\boldsymbol{A}}\,\sigma\left(\tilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}_0\right)\boldsymbol{W}_1\right)$

Dataset	Type	Nodes	Edges	Classes	Features	Label rate
Citeseer	Citation network	3,327	4,732	6	3,703	0.036
Cora	Citation network	2,708	5,429	7	1,433	0.052
Pubmed	Citation network	19,717	44,338	3	500	0.003
NELL	Knowledge graph	65,755	266,144	210	5,414	0.001

Method	Citeseer	Cora	Pubmed	NELL
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)

GNN History


Spectral GCN(2014)
ChebNet (2016)
GCN (2017)
GraphSAGE (2017)
PinSAGE (2018)
FastGCN (2018)
SGC (2019)
VRGCN (2018)
ClusterGCN (2019)
GraphSAINT (2020)
GNNAutoScale (2021)
PPRGo (2020)
GBP (2020)
scalable
JK-Net (2018)
APPNP (2019)
MixHop (2019)
DeepGCN (2019)
GCNII (2020)
DAGNN (2020)
RevGNN (2021)
GAMLP (2022)
deeper
MPNN (2017)
GAT (2018)
DiffPool (2018)
SAGPool (2019)
StructPool (2020)
GIN (2018)
PNA (2020)
Graphormer (2021)
architecture
VGAE (2017)
DGI (2019)
MVGRL (2020)
GPT-GNN (2020)
InfoGraph (2020)
GCC (2020)
GraphCL (2020)
GCA (2021)
JOAO (2021)
GraphMAE (2022)
self-supervised

Do we really make big progress?

- Using “heterogeneous graph neural networks (HGNN)” as an example
- Unrobust results with biased setting on small data

	HAN [36]		GTN [43]			RSHN [45]			HetGNN [44]				MAGNN [12]	
Dataset	ACM		DBLP	ACM	IMDB	AIFB	MUTAG	BGS	MC (10%)		MC (30%)		DBLP	
Metric	Macro-F1	Micro-F1	Macro-F1	Macro-F1	Macro-F1	Accuracy	Accuracy	Accuracy	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1
model*	91.89	91.85	94.18	92.68	60.92	97.22	82.35	93.10	97.8	97.9	98.1	98.2	93.13	93.61
GCN*	89.31	89.45	87.30	91.60	56.89	-	-	-	-	-	-	-	88.00	88.51
GAT*	90.55	90.55	93.71	92.33	58.14	91.67	72.06	66.32	96.2	96.3	96.5	96.5	91.05	91.61
model														
GCN														
GAT														

We tested 12 HGNN algorithms

* With a fairly proper setting, the results are even reversed!

1. Lv et al. Are we really making much progress? Revisiting, benchmarking and refining the Heterogeneous Graph Neural Networks. KDD’21.

Challenges

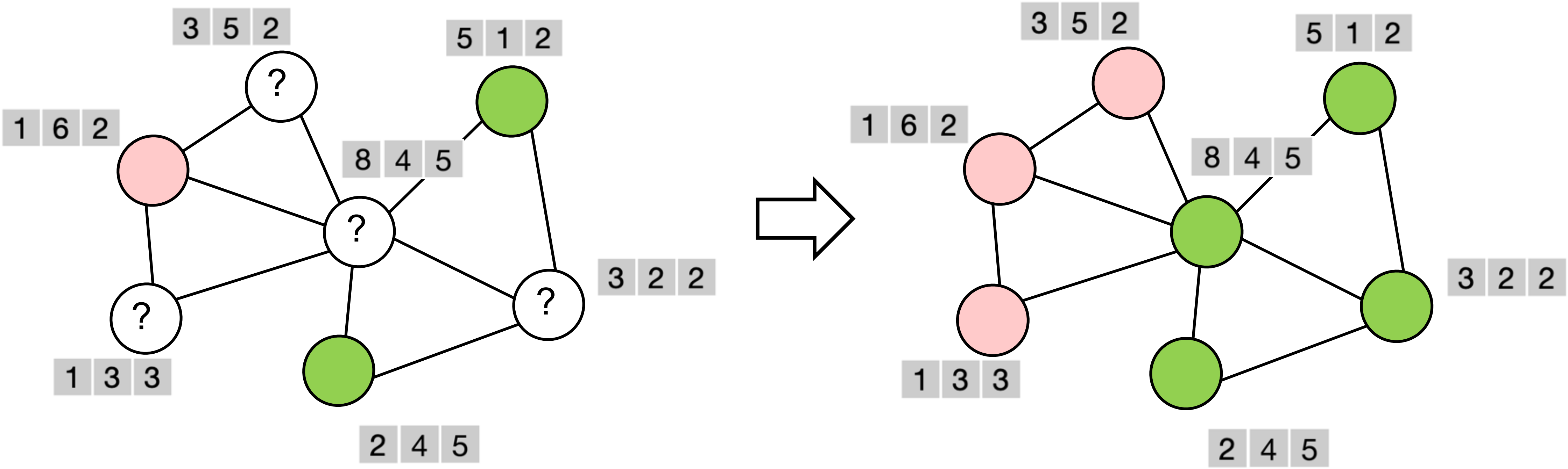
- Challenge 1: Robustness
- Challenge 2: Unlabeled data
- Challenge 3: Easy-to-use Tool

Overview

- Semi-supervised Learning on Graphs:
  - Training data: a small portion of labeled data + lots of unlabeled data
  - Robustness: consistency regularization for predictions of different views
- CogDL: A Comprehensive Library for Graphs (Easy-to-use)
- Contrastive Self-supervised Learning on Graphs:
  - Training data: all data is unlabeled
  - Contrasts the views generated from different augmentations
- Generative Self-supervised Learning on Graphs:
  - Training data: all data is unlabeled
  - Reconstruction of the input graph (graph structure, node features)

Semi-supervised Learning on Graphs

Semi-Supervised Learning on Graphs



Input: a partially labeled & attributed graph

Output: infer the labels of unlabeled nodes

Graph Neural Networks (GNN)

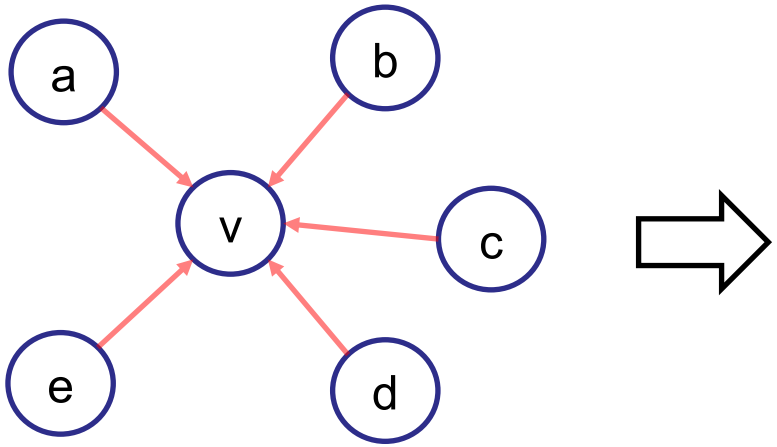
Message Passing:

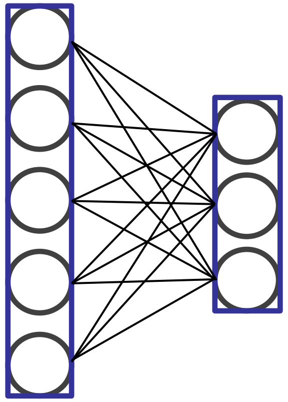
Representation vector of the $l + 1$-th layer

Activation function (e.g. ReLU)

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

Normalized adjacency matrix





$$\boldsymbol{H}^{l+1} = \sigma\left(\boldsymbol{W}^{l} \sum_{u \in N(v) \cup v} \frac{\boldsymbol{H}_{\boldsymbol{u}}^{\boldsymbol{l}}}{\sqrt{|N(u)||N(v)|}}\right)$$

Feature propagation

Non-linear transformation

Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In ICLR 2017

Over-fitting problem of GNNs

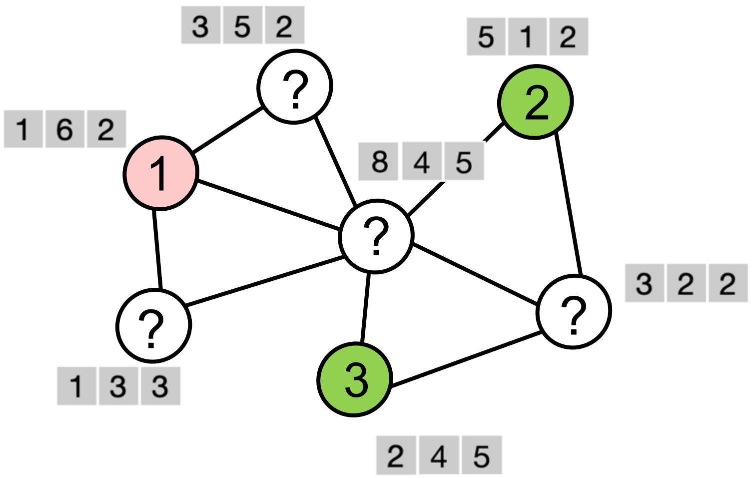
1. In GNNs, feature propagation is coupled with non-linear transformation. Increasing layers will introduce unnecessary parameters.

2. GNNs only adopt the supervised cross-entropy loss to guide the model training.

Training process:

Feature propagation is coupled with Non-linear transformation

$$H^{l+1} = \sigma\left(\hat{A}H^{(l)}W^{(l)}\right)$$



GNN

Loss function:

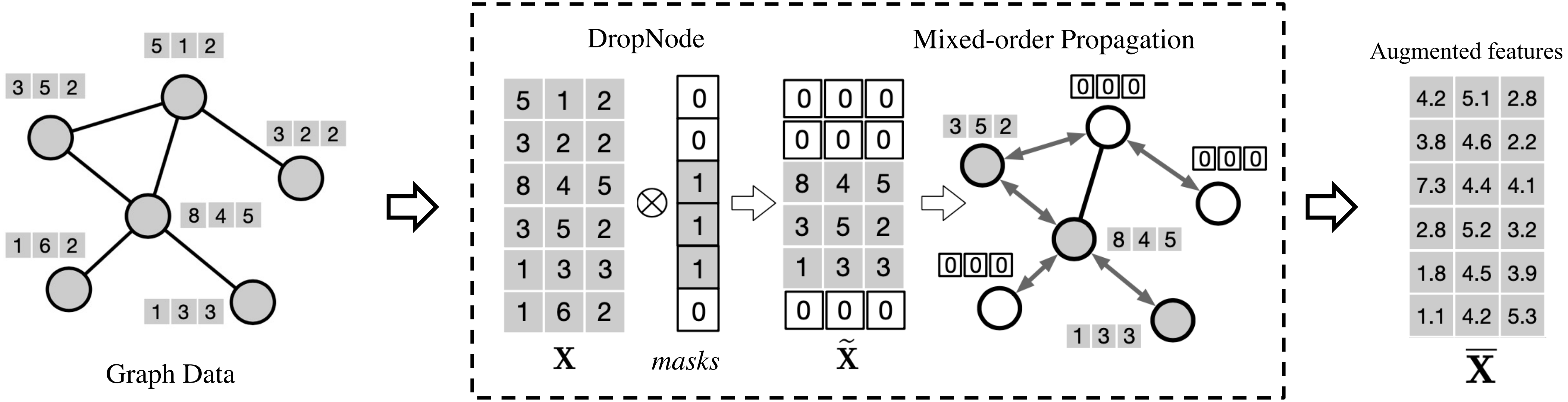
$$\boldsymbol{y}_1^{\mathrm{T}}\log(\hat{\boldsymbol{y}}_1) + \boldsymbol{y}_2^{\mathrm{T}}\log(\hat{\boldsymbol{y}}_2) + \boldsymbol{y}_3^{\mathrm{T}}\log(\hat{\boldsymbol{y}}_3)$$

Cannot fully leverage the unlabeled nodes

Graph Random Neural Network (GRAND)

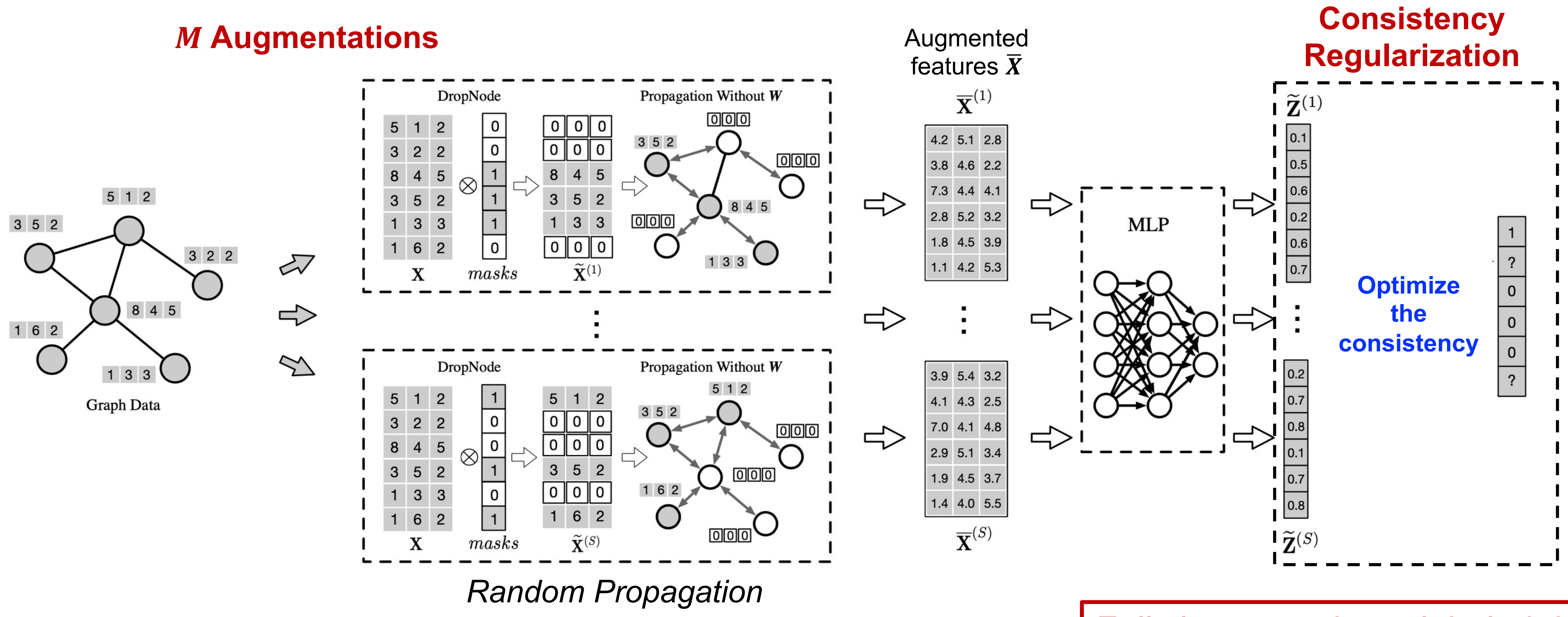
- Random Propagation (DropNode + Propagation):
  - Decouple the feature propagation from non-linear feature transformation.
  - Propagate feature with a mixed-order adjacency matrix: $\mathbf{\Pi} = \sum_{n=0}^{N} \frac{1}{N+1} \hat{\mathbf{A}}^n$
  - Use DropNode before feature propagation to randomly aggregate neighbors' features

Random Propagation as graph data augmentation



Feng W, Zhang J, Dong Y, et al. Graph random neural networks for semi-supervised learning on graphs[J]. In NeurIPS 2020.

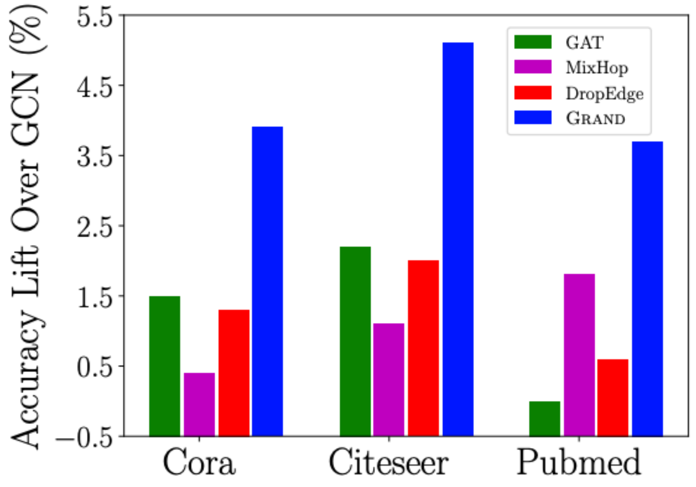
Graph Random Neural Network (GRAND) (cont.)



Fully leverage the unlabeled data

Performance of GRAND

	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**



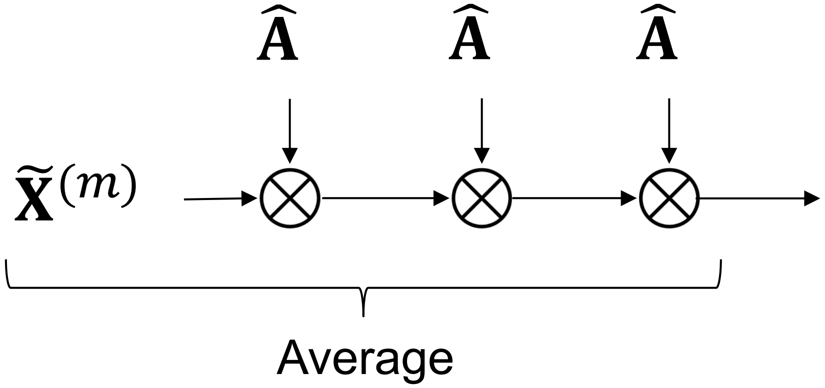
Instead of the marginal improvements by conventional GNN baselines over GCN, ***GRAND*** *achieves* ***much more significant*** *performance lift in* ***all three datasets***!

Scalability limitation of GRAND

- Random Propagation in GRAND:

$$\overline{\mathbf{X}}^{(m)} = \mathbf{\Pi}\, \widetilde{\mathbf{X}}^{(m)}, \qquad \mathbf{\Pi} = \sum_{n=0}^{N} \frac{1}{N+1} \hat{\mathbf{A}}^{n}$$

- $\overline{\mathbf{X}}^{(m)}$ is calculated with power iteration:



Weak Scalability:

- Time/Memory complexity: $O(|E| + |V|)$.
- Random propagation needs to be formed for multiple times at each epoch.

GRAND+: General Idea

- Mini-batch Radom Propagation:
  – Select a batch of nodes at each training step, and generate augmented features by

DropNode mask

$$\bar{\mathbf{X}}_s^{(m)} = \sum_{v \in \mathcal{N}_v^{\pi}} \boldsymbol{z}_v \cdot \mathbf{\Pi}(\mathrm{s}, \mathrm{v}) \cdot \mathbf{X}_v, \qquad \boldsymbol{z}_v \sim Bernoulli(1-\delta)$$

Non-zero elements in $\mathbf{\Pi}_s$

$$\mathbf{\Pi} = \sum_{n=0}^{N} \frac{1}{N+1} \widehat{\mathbf{A}}^n$$

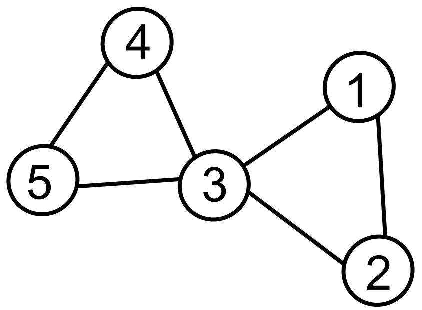
How to efficiently calculate the row vector $\mathbf{\Pi}_s$?

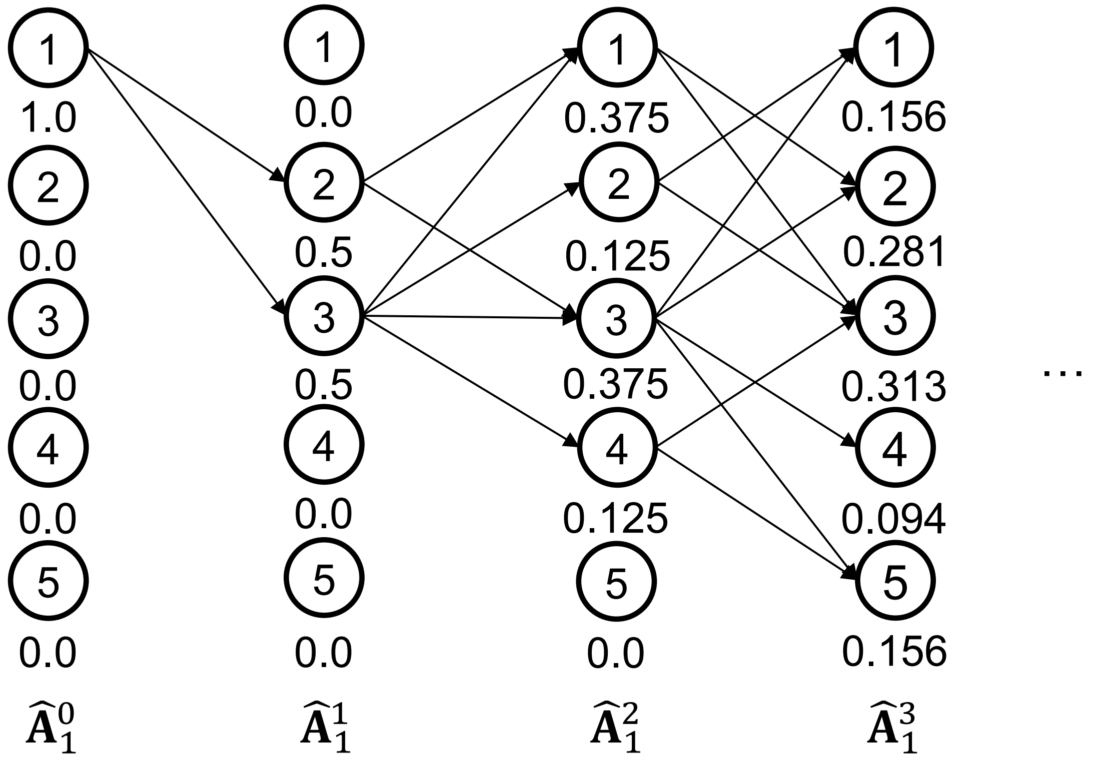
[1] Wenzheng Feng, et al. GRAND+: Scalable Graph Random Neural Networks. In WWW'22.

GRAND+：Matrix approximation

$\boldsymbol{\Pi} = \frac{1}{N+1} \sum_{n=0}^{N} \widehat{\mathbf{A}}^n$

$\widehat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}$ is random walk reverse transition matrix. $\mathbf{P}(s, \mathrm{v})$ indicates the random walk probability from s to v.





$$\boldsymbol{\Pi}_{\mathbf{1}} = \frac{1}{N+1} \sum_{n=0}^{N} \widehat{\mathbf{A}}_1^n$$

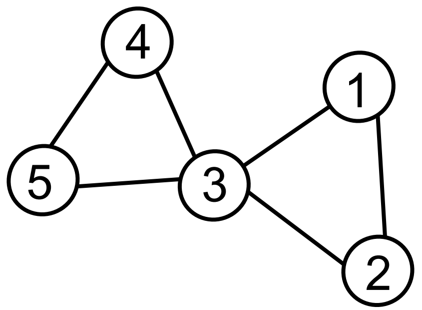
Random Walk Probability Diffusion

Complexity: $O(|E|)$ ☹

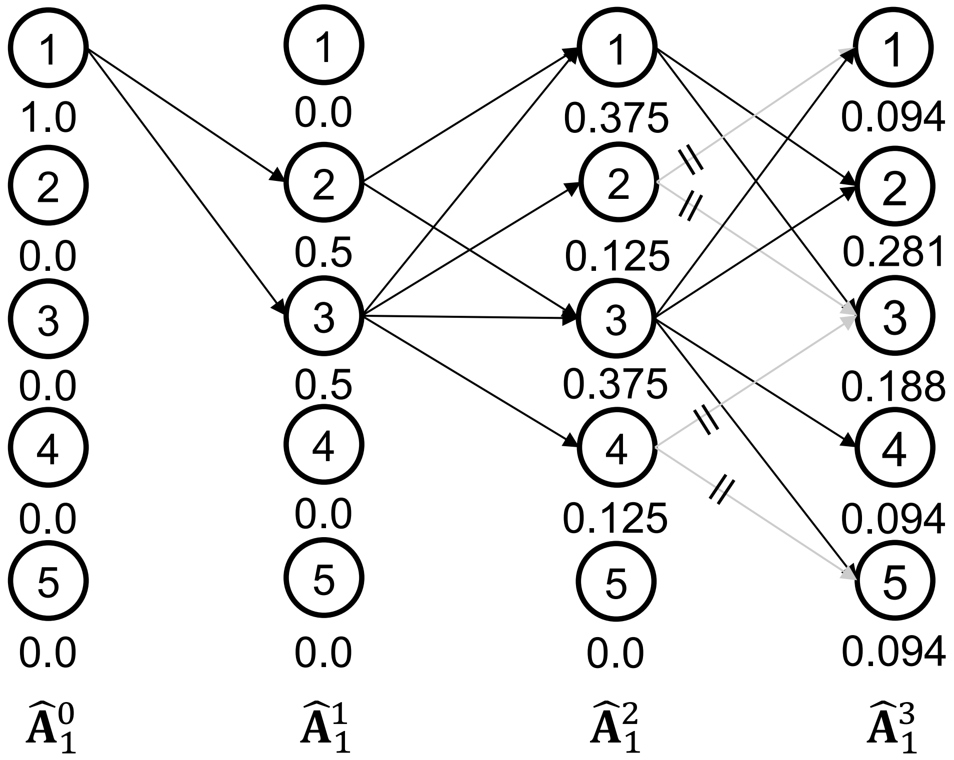
GRAND+：Matrix approximation

$\mathbf{\Pi} = \frac{1}{N+1}\sum_{n=0}^{N}\widehat{\mathbf{A}}^{n}$

$\widehat{\mathbf{A}} = \widetilde{\mathbf{D}}^{-1}\widetilde{\mathbf{A}}$ is random walk reverse transition matrix. $\mathbf{P}(s, \mathrm{v})$ indicates the random walk probability from s to v.



$$\widetilde{\mathbf{\Pi}}_{\mathbf{1}} = \frac{1}{N+1}\sum_{n=0}^{N}\widehat{\mathbf{A}}_{1}^{n}$$



$r_{max} = 0.07$

…

Generalized Forward Push (GFPush)

Error Bound：$T \cdot r_{max}$

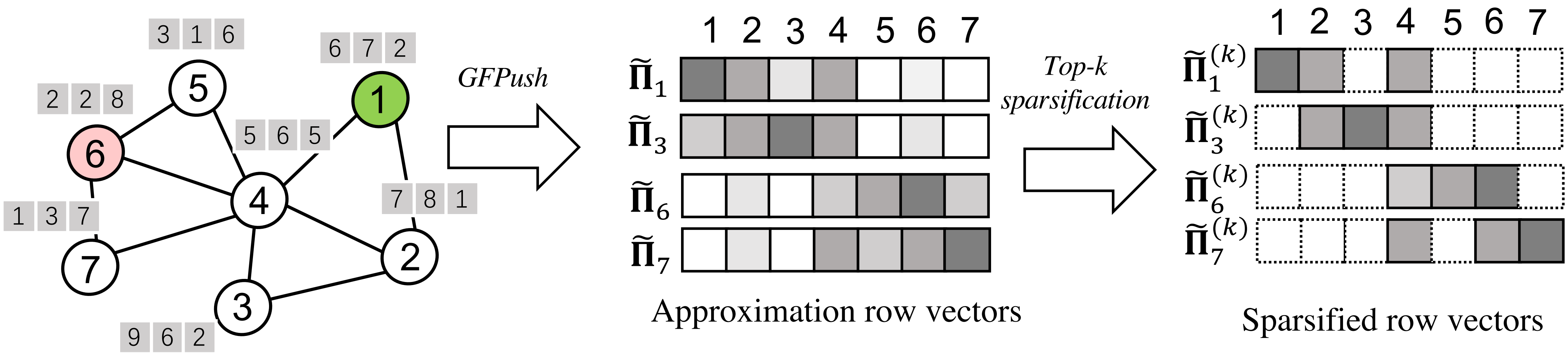
Time Complexity：T/r_{max}

Memory Complexity：T/r_{max}

GRAND+: Matrix approximation

- Approximation method:
  - GFPush: Generate an error bounded approximation $\tilde{\mathbf{\Pi}}_s$ for $\mathbf{\Pi}_s$.
  - Top-*k* sparsification: Truncate $\tilde{\mathbf{\Pi}}_s$ to make it only contains top-*k* elements.

THEOREM 1. *Algorithm 1 has $O(N/r_{max})$ time complexity and $O(N/r_{max})$ memory complexity, and returns $\tilde{\mathbf{\Pi}}_s$ as an approximation of $\mathbf{\Pi}_s$ with the L_1 error bound:* $\| \mathbf{\Pi}_s - \tilde{\mathbf{\Pi}}_s \|_1 \leq N \cdot (2|E| + |V|) \cdot r_{max}$.



GRAND+: Mini-batch Radom Propagation

- Mini-batch Radom Propagation with Approximation:

$$\overline{\mathbf{X}}_s^{(m)} = \sum_{v \in \mathcal{N}_v^{(k)}} \boldsymbol{z}_v \cdot \widetilde{\boldsymbol{\Pi}}^{(k)}(\mathrm{s}, \mathrm{v}) \cdot \mathbf{X}_v, \qquad \boldsymbol{z}_v \sim Bernoulli(1-\delta)$$

Non-zero elements in $\widetilde{\boldsymbol{\Pi}}_v^{(k)}$

- Prediction:

$$\hat{\mathbf{Y}}^{(m)} = \mathrm{MLP}(\overline{\mathbf{X}}_s^{(m)}, \Theta)$$

With batch size as b, the time complexity is $O(b \cdot k)$, which is independent of graph size

Scalability: Adopt GFPush to approximately calculate the propagation matrix， and adopt mini-batch method for model training

GRAND+: Confidence-aware Consistency Regularization

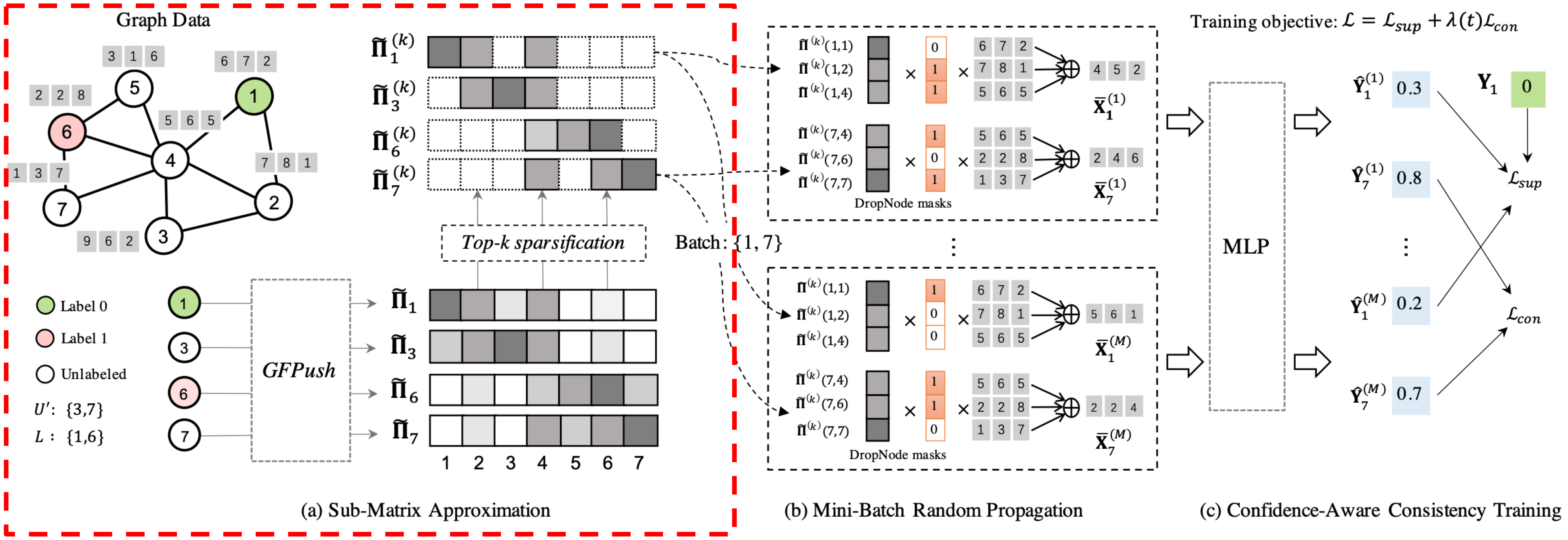
- Confidence-aware Consistency Loss：

$$\mathcal{L}_{con} = \frac{1}{b_u \cdot M} \sum_{s \in U_n} \mathbb{1}(\max(\overline{\mathbf{Y}}_s) \geq \gamma) \sum_{m=1}^{M} \mathcal{D}(\widetilde{\mathbf{Y}}_s, \hat{\mathbf{Y}}_s^{(m)}),$$

Confidence term: Filter out unlabeled nodes that have low confidence

Effectiveness: Further improving prediction performance

GRAND+ Architecture



Parallelization by OpenMP

GRAND+: Better scalability & generalization capability

Experiments

Table 2: Classification Accuracy (%) on Benchmarks.

Category	Method	Cora	Citeseer	Pubmed
Full-batch GNNs	GCN	81.5 ± 0.6	71.3 ± 0.4	79.1 ± 0.4
	GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3
	APPNP	84.1 ± 0.3	71.6 ± 0.5	79.7 ± 0.3
	GCNII	85.5 ± 0.5	73.4 ± 0.6	80.3 ± 0.4
	GRAND	85.4 ± 0.4	75.4 ± 0.4	82.7 ± 0.6
Scalable GNNs	FastGCN	81.4 ± 0.5	68.8 ± 0.9	77.6 ± 0.5
	GraphSAINT	81.3 ± 0.4	70.5 ± 0.4	78.2 ± 0.8
	SGC	81.0 ± 0.1	71.8 ± 0.1	79.0 ± 0.1
	GBP	83.9 ± 0.7	72.9 ± 0.5	80.6 ± 0.4
	PPRGo	82.4 ± 0.2	71.3 ± 0.3	80.0 ± 0.4
Our Methods	GRAND+ (P)	**85.8 ± 0.4**	**75.6 ± 0.4**	84.5 ± 1.1
	GRAND+ (A)	85.5 ± 0.4	75.5 ± 0.4	**85.0 ± 0.6**
	GRAND+ (S)	85.0 ± 0.5	74.4 ± 0.5	84.2 ± 0.6

Better generalization performance: Achieves 2.3% improvements over GRAND on Pubmed.

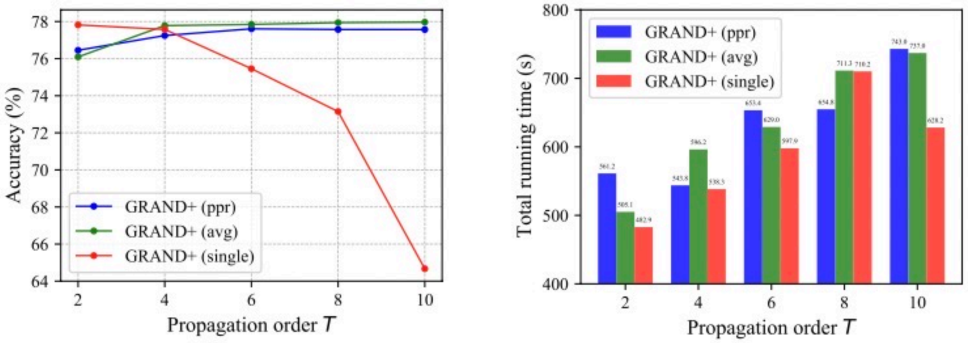
Experiments

Table 3: Accuracy (%) and Running Time (s) on Large Graphs.

Method	AMiner-CS		Reddit		Amazon2M		MAG.	
	Acc	RT	Acc	RT	Acc	RT	Acc	RT
GRAND	53.1±1.1	750	OOM	–	OOM	–	OOM	–
FastGCN	48.9±1.6	69	89.6±0.6	158	72.9±1.0	239	64.3±5.6	4220
GraphSAINT	51.8±1.3	39	92.1±0.5	39	75.9±1.3	189	75.0±1.7	6009
SGC	50.2±1.2	9	92.5±0.2	31	74.9±0.5	69	–	>24h
GBP	52.7±1.7	21	88.7±1.1	370	70.1±0.9	280	–	>24h
PPRGo	51.2±1.4	11	91.3±0.2	233	67.6±0.5	160	72.9±1.1	434
GRAND+ (P)	53.9±1.8	17	93.3±0.2	183	75.6±0.7	188	77.6±1.2	653
GRAND+ (A)	54.2±1.7	14	**93.5±0.2**	174	75.9±0.7	136	**80.0±1.1**	737
GRAND+ (S)	**54.2±1.6**	10	92.8±0.2	62	**76.2±0.6**	80	77.8±0.9	483

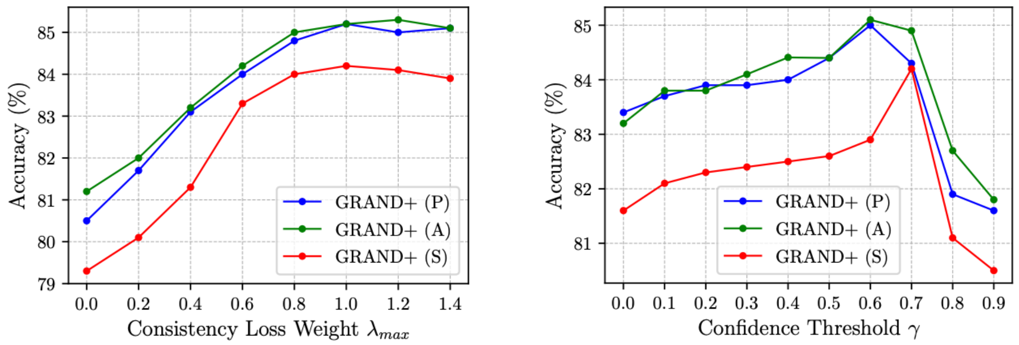
Scalability:
- 40 times faster than GRAND on Aminer-CS.
- 8 times faster than FastGCN on MAG.
- 12 times faster than GraphSAINT on MAG.
- Achieves comparable running time and 4.9% improvement than PPRGo on MAG.

Parameter Analysis



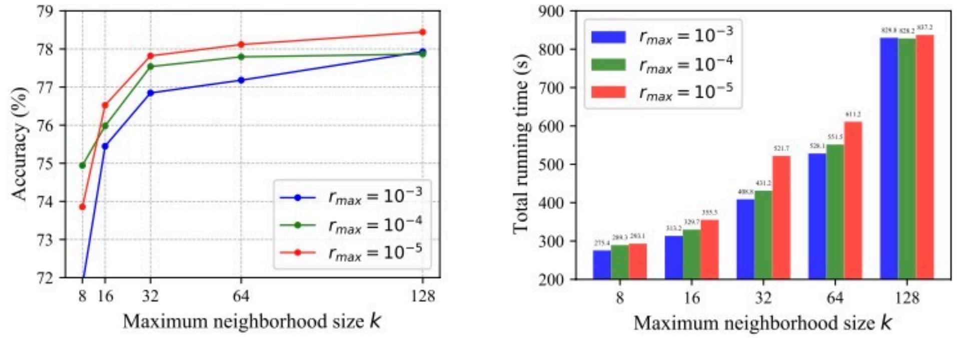
(a) Classification accuracy. (b) Total running time.

Figure 5: Effects of propagation order T on MAG-Scholar-C.



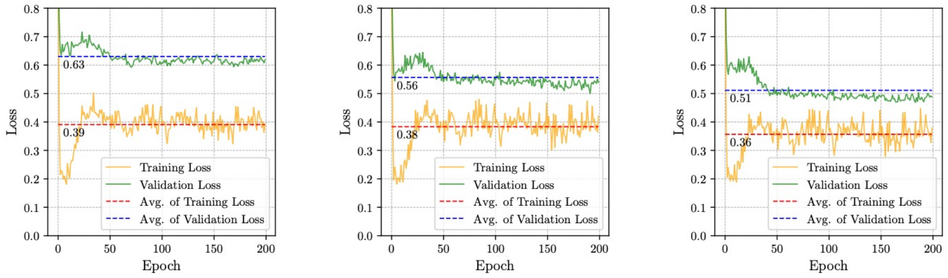
(a) Accuracy w.r.t. λ_{max}. (b) Accuracy w.r.t. γ.

Figure 2: Effects of λ_{max} and γ on Pubmed.



(a) Classification accuracy. (b) Total running time.

Figure 4: GRAND+ w.r.t. k and r_{max} on MAG-Scholar-C.



(a) $\lambda_{max} = 0.0$ (b) $\lambda_{max} = 1.0, \gamma = 0.0$ (c) $\lambda_{max} = 1.0, \gamma = 0.6$

Figure 3: Training and Validation Losses on Pubmed.

CogDL: A Comprehensive Library for Graph Deep Learning

Access the code here

GitHub: https://github.com/THUDM/cogdl , 1400+ stars as of April 2023.

CogDL – Overview

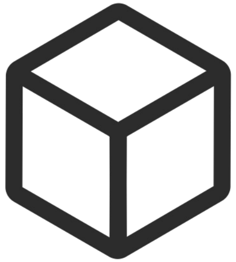
Vision

CogDL aims at providing researchers and practitioners with easy-to-use APIs, reproducible results, and high efficiency for most graph tasks and applications.

Philosophy

Easy-to-use

Reproducibility



Efficiency

[1] Yukuo Cen, Zhenyu Hou, Yan Wang, Qibin Chen, Yizhen Luo, Zhongming Yu, Hengrui Zhang, Xingcheng Yao, Aohan Zeng, Shiguang Guo, Yuxiao Dong, Yang Yang, Peng Zhang, Guohao Dai, Yu Wang, Chang Zhou, Hongxia Yang, Jie Tang. CogDL: A Comprehensive Library for Graph Deep Learning. In WWW'23.

CogDL - A Unified GNN Trainer

- Design a unified Trainer for GNN training

CogDL

```
model = GCN(...)
data = CoraDataset()[0]
mw = ModelWrapper(model, ...)
dw = DataWrapper(data, ...)
trainer = Trainer(epochs=100)
result = trainer.run(mw, dw)
```

PyG

```
model = GCN(...)
data = Planetoid(name="Cora")[0]
optimizer = torch.optim.Adam(...)
for epoch in range(100):
    pred = model(data.x, data.edge_index)
    labels = data.y
    mask = data.train_mask
    loss = F.nll_loss(pred[mask],
                      labels[mask])
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    val_acc = evaluate(model, data)
    if val_acc > best_val_acc:
        best_val_acc = val_acc
        best_model = deepcopy(model)
result = test(best_model, data)
```

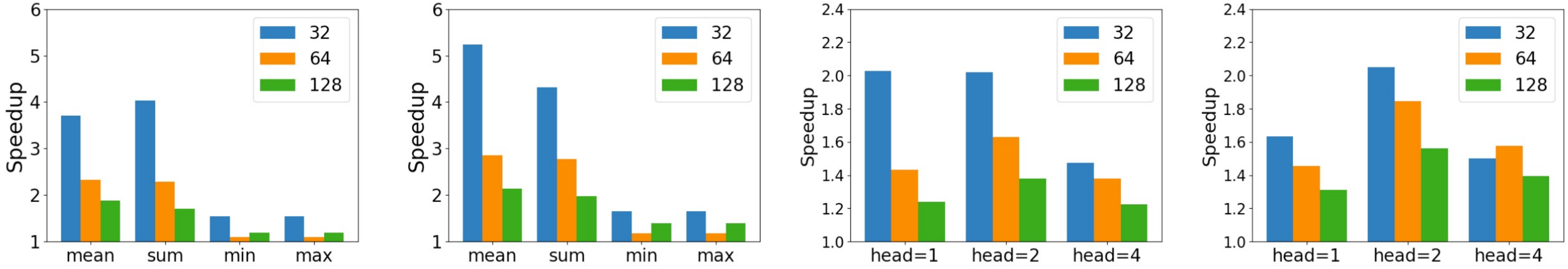
DGL

```
model = GCN(...)
data = CoraGraphDataset()[0]
optimizer = torch.optim.Adam(...)
for epoch in range(100):
    pred = model(data)
    labels = data.ndata['label']
    mask = data.ndata['train_mask']
    loss = F.nll_loss(pred[mask],
                      labels[mask])
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    val_acc = evaluate(model, data)
    if val_acc > best_val_acc:
        best_val_acc = val_acc
        best_model = deepcopy(model)
result = test(best_model, data)
```

CogDL – Operator Acceleration

- Acceleration for SpMM-like operators
  - Design a more balanced strategy for GPU parallel computation
  - 1.17x~5.24x Speedup on Reddit/Yelp datasets compared to DGL

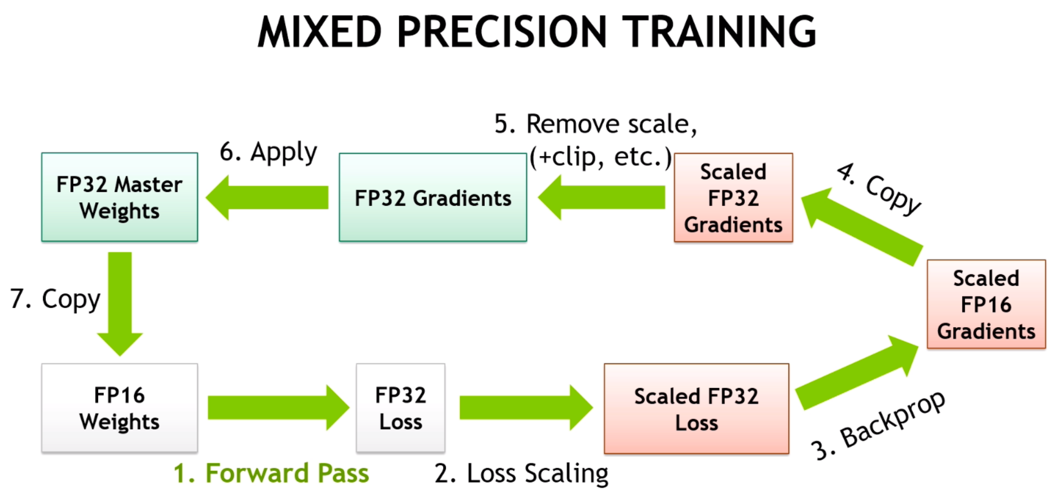
```
workload_balance = balancer(graph) # preprocessing
out_feat = GSpMM(workload_balance, graph, in_feat, reduce_op,
      compute_op)
```



(a) GSpMM on Reddit (b) GSpMM on Yelp (c) multi-head SpMM on Reddit (d) multi-head SpMM on Yelp

CogDL - Mixed Precision Training

- Support mixed precision training:
  - 1.44x~2.02x speedups due to half-precision (FP16) computation



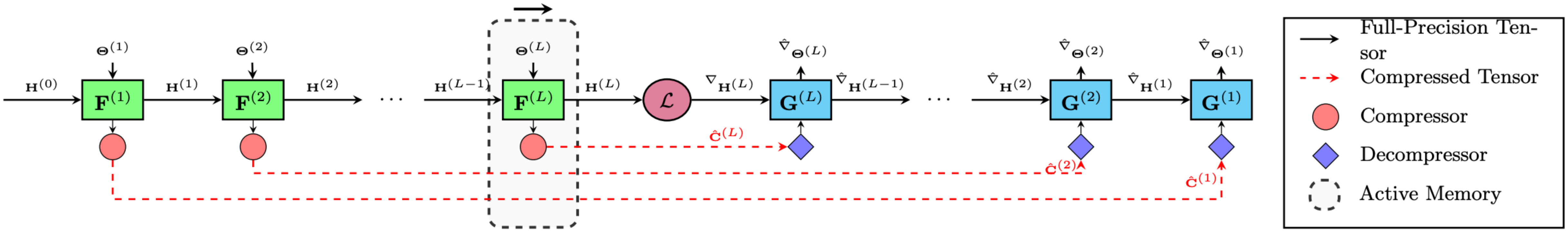
2-layer GCNs on the Reddit dataset（single / mixed precision）

	GPU Memory	Accuracy	Training Speed	
			2080 Ti	3090
Single precision training	5,567 MB	95.44 ± 0.07	2.20 it/s	3.93 it/s
Mixed precison training	4,046 MB	95.35 ± 0.08	3.17 it/s	7.97 it/s

1. Image Credit: https://developer.nvidia.com/blog/video-mixed-precision-techniques-tensor-cores-deep-learning/

CogDL - Activation Compressed Training

- Support activation compressed training:
  - reduce 6.4x~16x training memory footprints



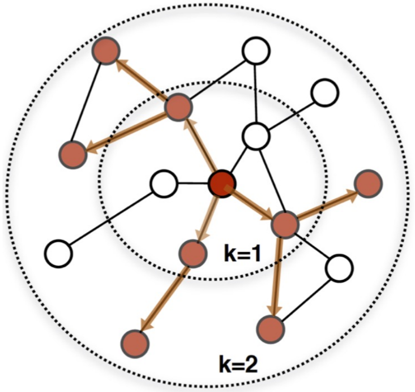
2-layer GCNs on three datasets: accuracy (%), act memory(MB)

	origin (32-bit)	+actnn (4-bit)	+actnn (3-bit)	+actnn (2-bit)
Flickr	51.17 ± 0.19 (288)	51.08 ± 0.18 (37)	51.14 ± 0.18 (26)	51.20 ± 0.18 (18)
Reddit	95.33 ± 0.07 (1532)	95.32 ± 0.07 (194)	95.31 ± 0.07 (158)	95.34 ± 0.06 (112)
Yelp	39.86 ± 0.94 (4963)	40.06 ± 0.74 (773)	40.21 ± 0.82 (665)	39.89 ± 1.45 (551)

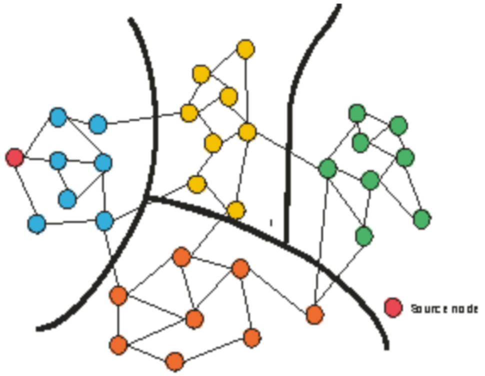
1. Xiaoxuan Liu, Lianmin Zheng, Dequan Wang, **Yukuo Cen**, Weize Chen, Xu Han, Jianfei Chen, Zhiyuan Liu, Jie Tang, Joey Gonzalez, Michael Mahoney, and Alvin Cheung. GACT: Activation Compressed Training for Generic Network Architectures. In ICML'22.

CogDL – Training on Large-scale Graphs

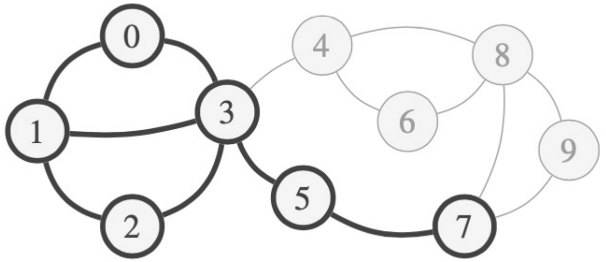
- Full-batch training on large-scale graphs is unaffordable
- Training GNNs via mini-batch sampling
- Usage: python scripts/train.py --model gcn --dataset reddit --dw cluster_dw



Neighbor Sampling
(NeurIPS ’17)



ClusterGCN
(KDD ’19)



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

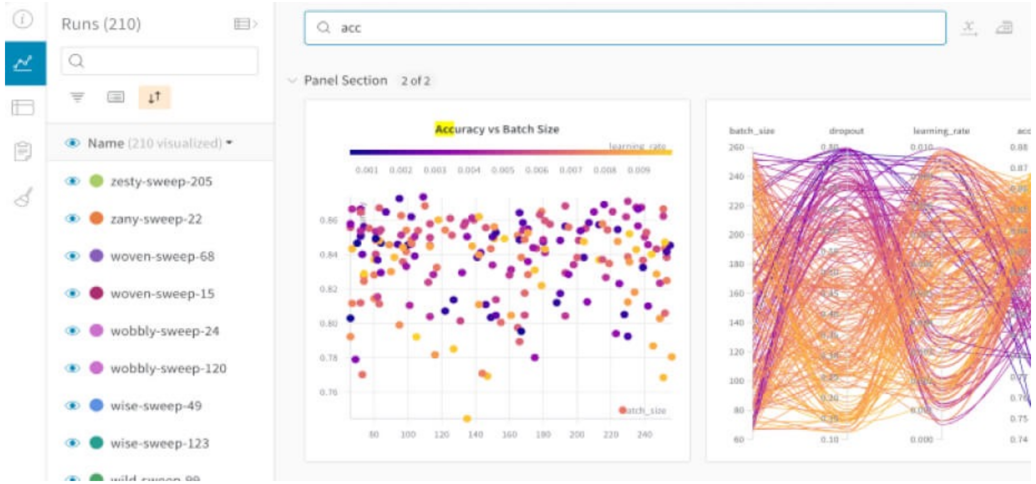
GraphSAINT
(ICLR ’20)

1. Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In NeurIPS ’17.
2. Wei-Lin Chiang, Xuanqing Liu, Si Si, Yang Li, Samy Bengio, and Cho-Jui Hsieh. Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks. In KDD ’19.
3. Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In ICLR ’20.

CogDL – Experiment Management

- Hyper-parameter Search
  - Integrate optuna for users to enable hyper-parameter search
- Experiment Management
  - Support Tensorboard and WandB for logging and debugging

```
def func(trial):
  return {
    "lr": trial.suggest_categorical("lr",[0.01, 0.1]),
    "hidden_size": trial.suggest_categorical("hidden_size", [32, 64, 128]),
    "dropout": trial.suggest_uniform("dropout", 0.1, 0.8),
  }
experiment(dataset="cora", model="gcn", search_space=func) # hyper-parameter search
```



1.Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In KDD'19.
2. Lukas Biewald. 2020. Experiment Tracking with Weights and Biases. https: //www.wandb.com/

CogDL – Benchmarks

- Easy-to-reproduce benchmarks
  - Traditional graph tasks such as network embedding
  - Frontier graph benchmarks such as IGB[1], GRB[2], HGB[3]

Method	PPI (50%)	Wikipedia (50%)	Blogcatalog (50%)	DBLP (5%)	Flickr (5%)	*Reproducible*
NetMF [37]	23.73 ± 0.22	57.42 ± 0.56	42.47 ± 0.35	56.72 ± 0.14	36.27 ± 0.17	*Yes*
ProNE [66]	24.60 ± 0.39	56.06 ± 0.48	41.16 ± 0.26	56.85 ± 0.28	36.56 ± 0.11	*Yes*
NetSMF [36]	23.88 ± 0.35	53.81 ± 0.58	40.62 ± 0.35	59.76 ± 0.41	35.49 ± 0.07	*Yes*
Node2vec [17]	20.67 ± 0.54	54.59 ± 0.51	40.16 ± 0.29	57.36 ± 0.39	36.13 ± 0.13	*Yes*
LINE [45]	21.82 ± 0.56	52.46 ± 0.26	38.06 ± 0.39	49.78 ± 0.37	31.61 ± 0.09	*Yes*
DeepWalk [35]	20.74 ± 0.40	49.53 ± 0.54	40.48 ± 0.47	57.54 ± 0.32	36.09 ± 0.10	*Yes*
SpectralClustering [48]	22.48 ± 0.30	49.35 ± 0.34	41.41 ± 0.34	43.68 ± 0.58	33.09 ± 0.07	*Yes*
Hope [33]	21.43 ± 0.32	54.04 ± 0.47	33.99 ± 0.35	56.15 ± 0.22	28.97 ± 0.19	*Yes*
GraRep [5]	20.60 ± 0.34	54.37 ± 0.40	33.48 ± 0.30	52.76 ± 0.42	31.83 ± 0.12	*Yes*

1. Ziang Li, Ming Ding, Weikai Li, Zihan Wang, Ziyu Zeng, Yukuo Cen, and Jie Tang. Rethinking the Setting of Semi-supervised Learning on Graphs. In IJCAI'22.
2. Qinkai Zheng, Xu Zou, Yuxiao Dong, Yukuo Cen, Da Yin, Jiarong Xu, Yang Yang, and Jie Tang. Graph Robustness Benchmark: Benchmarking the Adversarial Robustness of Graph Machine Learning. In NeurIPS'21 D&B.
3. Qingsong Lv, Ming Ding, Qiang Liu, Yuxiang Chen, Wenzheng Feng, Siming He, Chang Zhou, Jian-guo Jiang, Yuxiao Dong, and Jie Tang. Are we really making much progress? Revisiting, benchmarking and refining the Heterogeneous Graph Neural Networks. In KDD'21.

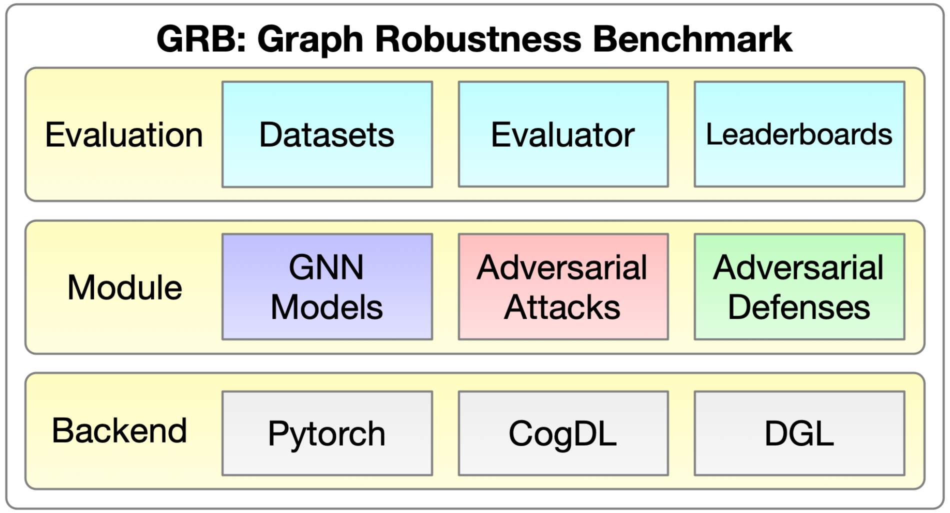
CogDL – GRB

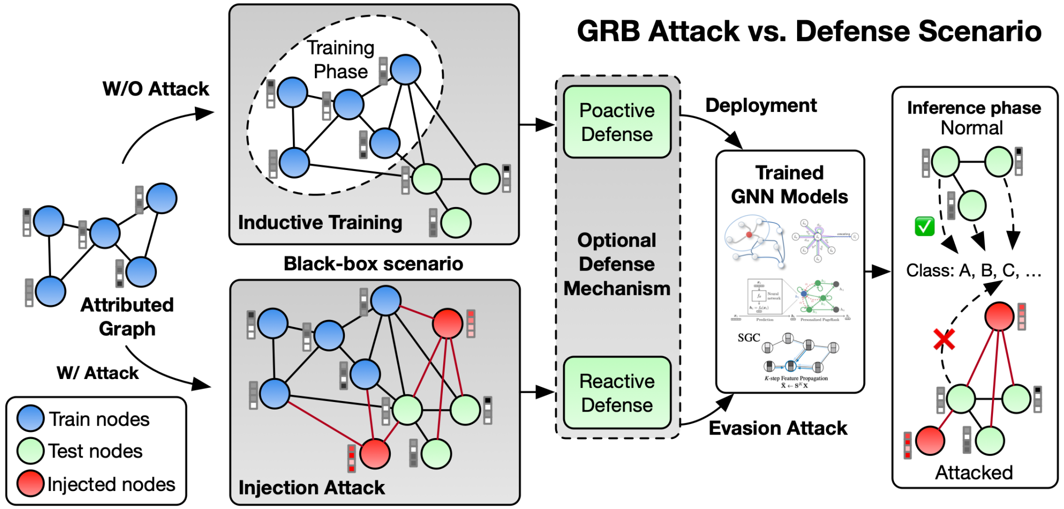
Background:

Recently, works have proved that adversarial attacks can threat the *robustness* of graph ML models in various tasks.

Problems:

1. Ill-defined threat model in previous works.
2. Absence of unified and standard evaluation approach.





Example of GRB evaluation scenario

Solution: **Graph Robustness Benchmark (GRB)**

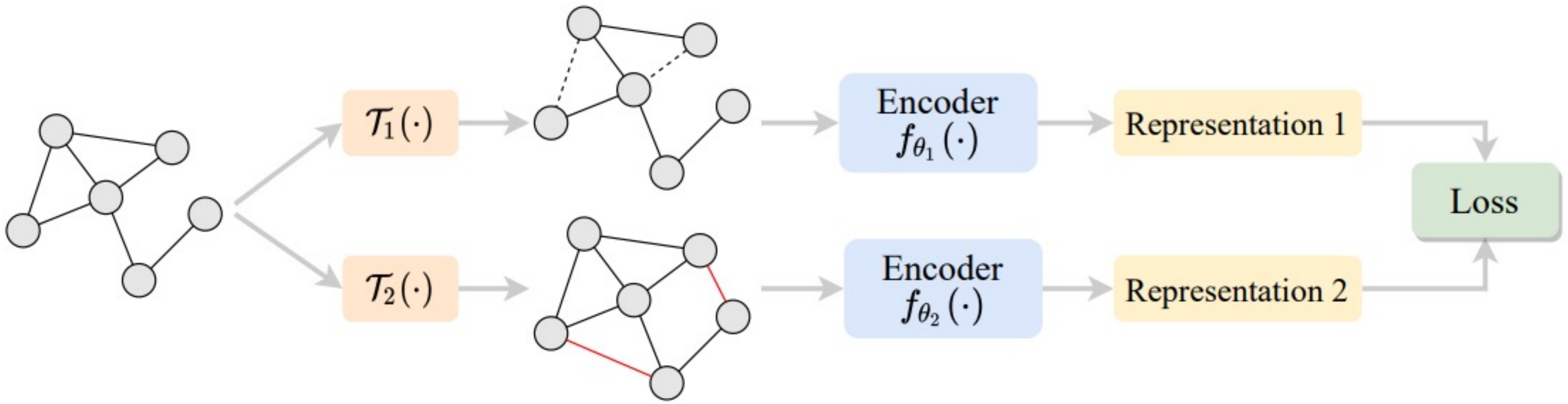
Scalable, *general*, *unified*, and *reproducible* benchmark on *adversarial robustness* of graph ML models, which facilitates fair comparisons among various attacks & defenses and promotes future research in this field.

Graph Robustness Benchmark: Rethinking and Benchmarking Adversarial Robustness of Graph Neural Networks
Qinkai Zheng, Xu Zou, Yuxiao Dong, Yukuo Cen, Jie Tang

Contrastive Learning on Graphs

Paradigm of Graph Contrastive Learning

- The graph contrastive learning
  - 1) generates two views based on different augmentations
  - 2) encodes the graphs of two views
  - 3) construct the self-supervised signal via contrast

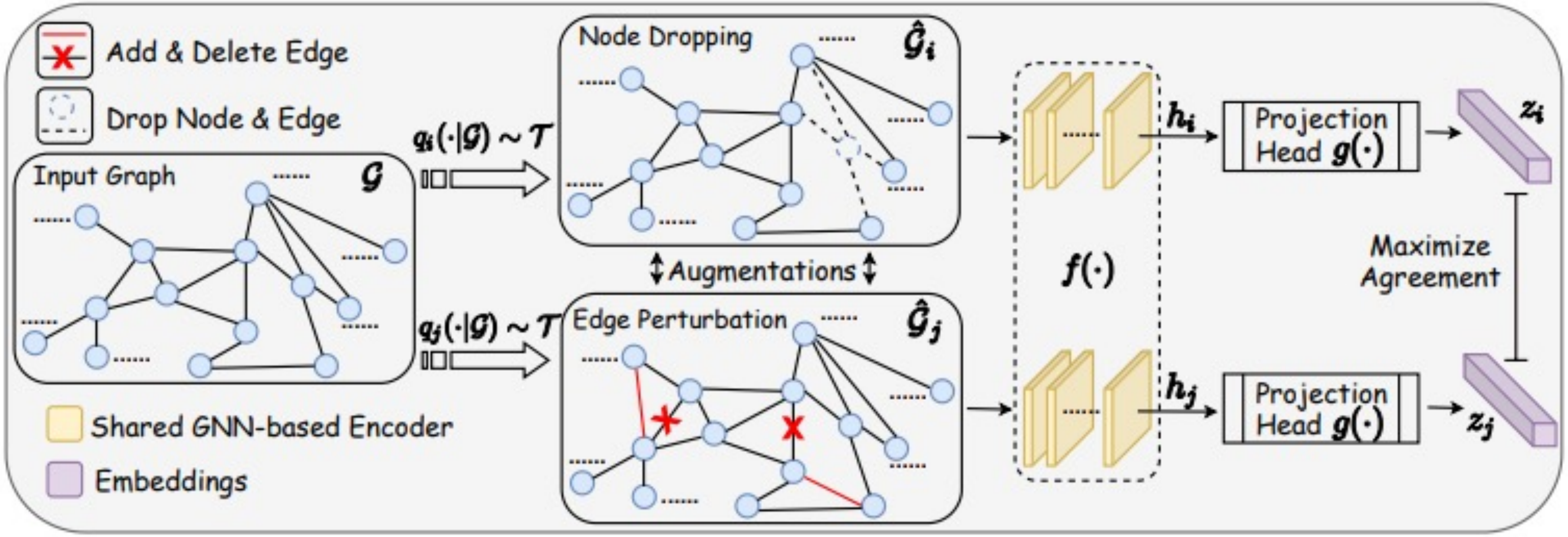


[1] Wu, Lirong, et al. "Self-supervised learning on graphs: Contrastive, generative, or predictive." IEEE Transactions on Knowledge and Data Engineering (2021).

Graph Contrastive Learning with Augmentations

GraphCL: a contrastive learning method with augmentations

- (1) propose different graph augmentation strategies
- (2) use projection heads for the graph encoding
- (3) maximizing the consistency $\ell_n = -\log \frac{\exp(\mathrm{sim}(z_{n,i}, z_{n,j})/\tau)}{\sum_{n'=1, n' \neq n}^{N} \exp(\mathrm{sim}(z_{n,i}, z_{n',j})/\tau)}$



[1] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, Yang Shen. Graph Contrastive Learning with Augmentations. NeurIPS 2020.

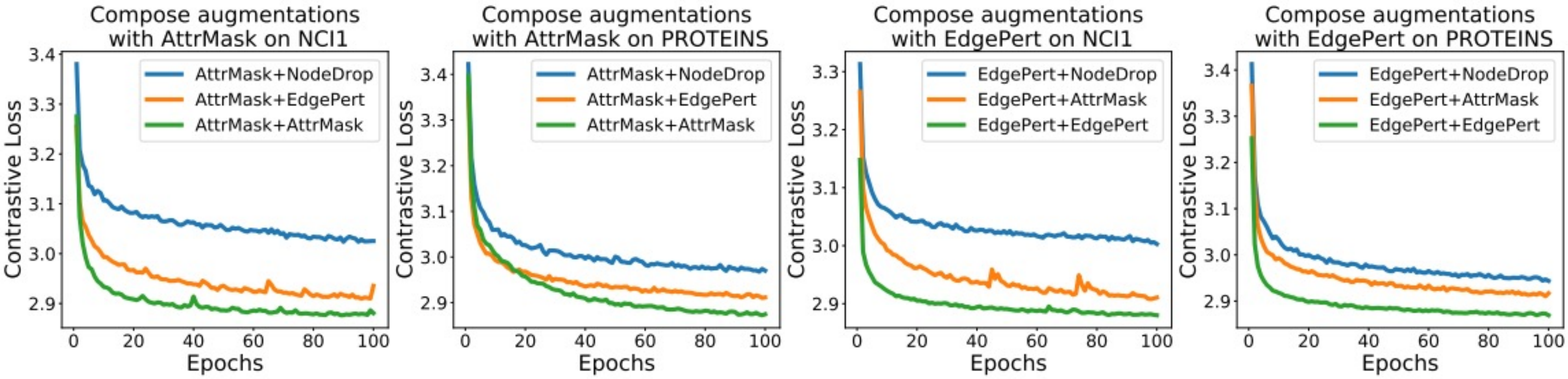
Data Augmentation for Graphs

- Focus on three categories of graphs:
  - Biochemical molecules, social networks, image super-pixel graphs
- Propose four graph-level augmentations

Data augmentation	**Type**	**Underlying Prior**
Node dropping	Nodes, edges	Vertex missing does not alter semantics.
Edge perturbation	Edges	Semantic robustness against connectivity variations.
Attribute masking	Nodes	Semantic robustness against losing partial attributes.
Subgraph	Nodes, edges	Local structure can hint the full semantics.

Role of Data Augmentation

- Contrastive loss curves for different augmentation pairs
- Using the same type gets better performance



Experimental Results of GraphCL

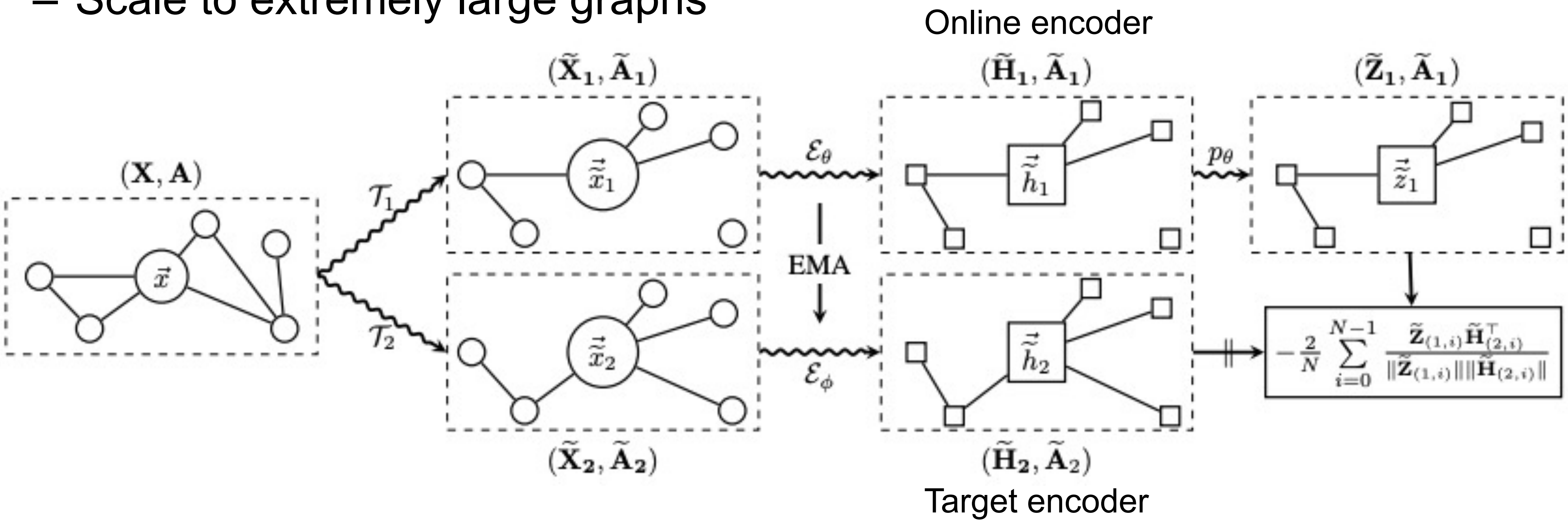
- Unsupervised representation learning setting

Datasets	Category	Graph Num.	Avg. Node	Avg. Degree
NCI1	Biochemical Molecules	4110	29.87	1.08
PROTEINS	Biochemical Molecules	1113	39.06	1.86
COLLAB	Social Networks	5000	74.49	32.99
RDT-B	Social Networks	2000	429.63	1.15

Dataset	NCI1	PROTEINS	DD	MUTAG	COLLAB	RDT-B	RDT-M5K	IMDB-B
GL	-	-	-	81.66±2.11	-	77.34±0.18	41.01±0.17	65.87±0.98
WL	80.01±0.50	72.92±0.56	-	80.72±3.00	-	68.82±0.41	46.06±0.21	72.30±3.44
DGK	80.31±0.46	73.30±0.82	-	87.44±2.72	-	78.04±0.39	41.27±0.18	66.96±0.56
node2vec	54.89±1.61	57.49±3.57	-	72.63±10.20	-	-	-	-
sub2vec	52.84±1.47	53.03±5.55	-	61.05±15.80	-	71.48±0.41	36.68±0.42	55.26±1.54
graph2vec	73.22±1.81	73.30±2.05	-	83.15±9.25	-	75.78±1.03	47.86±0.26	71.10±0.54
InfoGraph	76.20±1.06	74.44±0.31	72.85±1.78	89.01±1.13	70.65±1.13	82.50±1.42	53.46±1.03	73.03±0.87
GraphCL	77.87±0.41	74.39±0.45	78.62±0.40	86.80±1.34	71.36±1.15	89.53±0.84	55.99±0.28	71.14±0.44

Bootstrapped Graph Latents (BGRL)

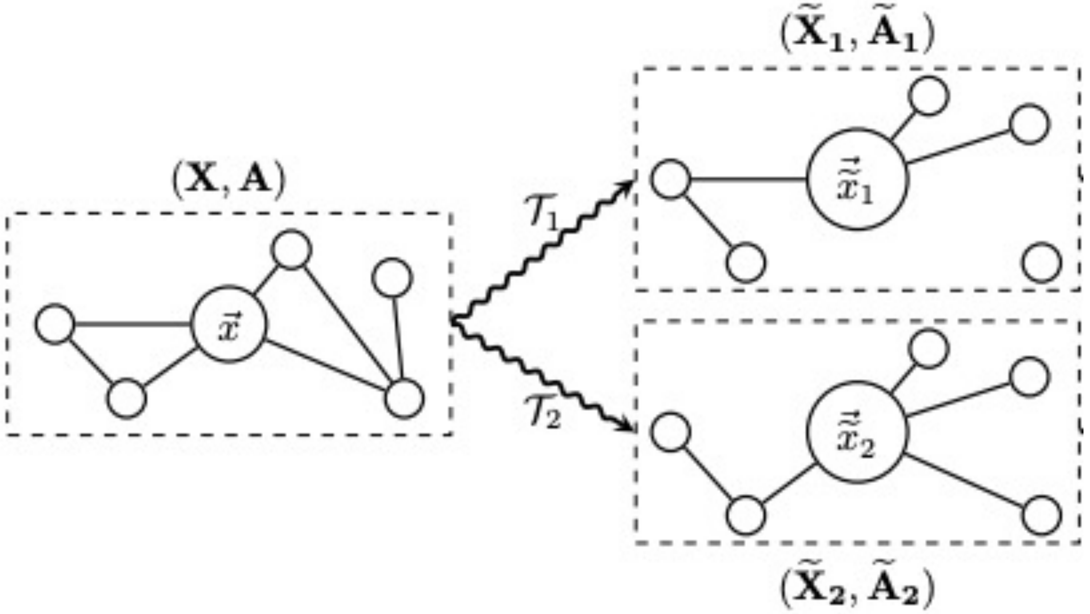
- The characteristics of BGRL:
  - Simple graph augmentation
  - Not requiring negative samples
  - Scale to extremely large graphs



Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L. Dyer, Rémi Munos, Petar Veličković, Michal Valko. Large-Scale Representation Learning on Graphs via Bootstrapping. ICLR 2022.

Graph Augmentation of BGRL

- Applying stochastic graph augmentation functions
  - random node feature masking: Bernoulli distribution $\mathcal{B}(1 - p_f)$
  - random edge masking: Bernoulli distribution $\mathcal{B}(1 - p_e)$
- With fixed hyperparameters for each graph



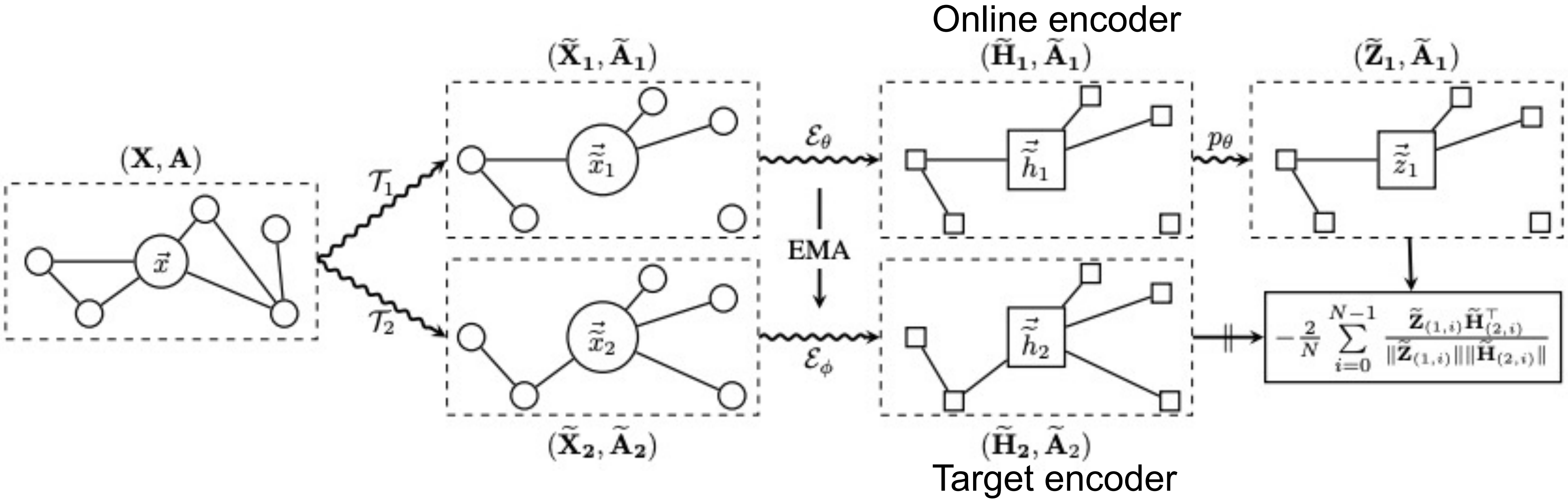
BGRL Update Step

- Update the online encoder via gradient descent

$$\ell(\theta, \phi) = -\frac{2}{N} \sum_{i=0}^{N-1} \frac{\widetilde{\mathbf{Z}}_{(1,i)} \widetilde{\mathbf{H}}_{(2,i)}^\top}{\|\widetilde{\mathbf{Z}}_{(1,i)}\| \|\widetilde{\mathbf{H}}_{(2,i)}\|}$$

- Update the target encoder via EMA

$$\phi \leftarrow \tau\phi + (1-\tau)\theta$$



Experimental Results of BGRL

	Task	Nodes	Edges	Features	Classes
WikiCS	Transductive	11,701	216,123	300	10
Amazon Computers	Transductive	13,752	245,861	767	10
Amazon Photos	Transductive	7,650	119,081	745	8
Coauthor CS	Transductive	18,333	81,894	6,805	15
Coauthor Physics	Transductive	34,493	247,962	8,415	5

	WikiCS	Am. Comp.	Am. Photos	Co.CS	Co.Phy
Raw features	71.98 ± 0.00	73.81 ± 0.00	78.53 ± 0.00	90.37 ± 0.00	93.58 ± 0.00
`DeepWalk`	74.35 ± 0.06	85.68 ± 0.06	89.44 ± 0.11	84.61 ± 0.22	91.77 ± 0.15
`DeepWalk` + feat.	77.21 ± 0.03	86.28 ± 0.07	90.05 ± 0.08	87.70 ± 0.04	94.90 ± 0.09
`DGI`	75.35 ± 0.14	83.95 ± 0.47	91.61 ± 0.22	92.15 ± 0.63	94.51 ± 0.52
`GMI`	74.85 ± 0.08	82.21 ± 0.31	90.68 ± 0.17	OOM	OOM
`MVGRL`	77.52 ± 0.08	87.52 ± 0.11	91.74 ± 0.07	92.11 ± 0.12	95.33 ± 0.03
`Random-Init`*	78.95 ± 0.58	86.46 ± 0.38	92.08 ± 0.48	91.64 ± 0.29	93.71 ± 0.29
`GRACE`*	$\mathbf{80.14 \pm 0.48}$	89.53 ± 0.35	92.78 ± 0.45	91.12 ± 0.20	OOM
`BGRL`*	79.98 ± 0.10	$\mathbf{90.34 \pm 0.19}$	$\mathbf{93.17 \pm 0.30}$	$\mathbf{93.31 \pm 0.13}$	$\mathbf{95.73 \pm 0.05}$
`GCA`	*78.35 ± 0.05*	*88.94 ± 0.15*	*92.53 ± 0.16*	*93.10 ± 0.01*	*95.73 ± 0.03*
Supervised GCN	*77.19 ± 0.12*	*86.51 ± 0.54*	*92.42 ± 0.22*	*93.03 ± 0.31*	*95.65 ± 0.16*

Experimental Results of BGRL (cont.)

- Comparison with SOTA methods on ogbn-arxiv and PPI datasets

	Validation	Test
MLP	57.65 ± 0.12	55.50 ± 0.23
node2vec	71.29 ± 0.13	70.07 ± 0.13
Random-Init*	69.90 ± 0.11	68.94 ± 0.15
DGI*	71.26 ± 0.11	70.34 ± 0.16
GRACE full-graph*	OOM	OOM
GRACE-SUBSAMPLING ($k = 2$)*	60.49 ± 3.72	60.24 ± 4.06
GRACE-SUBSAMPLING ($k = 8$)*	71.30 ± 0.17	70.33 ± 0.18
GRACE-SUBSAMPLING ($k = 32$)*	72.18 ± 0.16	71.18 ± 0.16
GRACE-SUBSAMPLING ($k = 2048$)*	$\mathbf{72.61 \pm 0.15}$	$\mathbf{71.51 \pm 0.11}$
BGRL*	$\mathbf{72.53 \pm 0.09}$	$\mathbf{71.64 \pm 0.12}$
Supervised GCN	*73.00 ± 0.17*	*71.74 ± 0.29*

ogbn-arxiv

	PPI
Raw features	42.20
DGI	63.80 ± 0.20
GMI	65.00 ± 0.02
Random-Init	62.60 ± 0.20
GRACE MeanPooling encoder*	69.66 ± 0.15
BGRL MeanPooling encoder*	69.41 ± 0.15
GRACE GAT encoder*	69.71 ± 0.17
BGRL GAT encoder*	$\mathbf{70.49 \pm 0.05}$
Supervised MeanPooling	*96.90 ± 0.20*
Supervised GAT	*97.30 ± 0.20*

PPI

Experiment on MAG240M

- MAG240M: over 240 million nodes (of which 1.4 million are labeled) and 1.7 billion edges

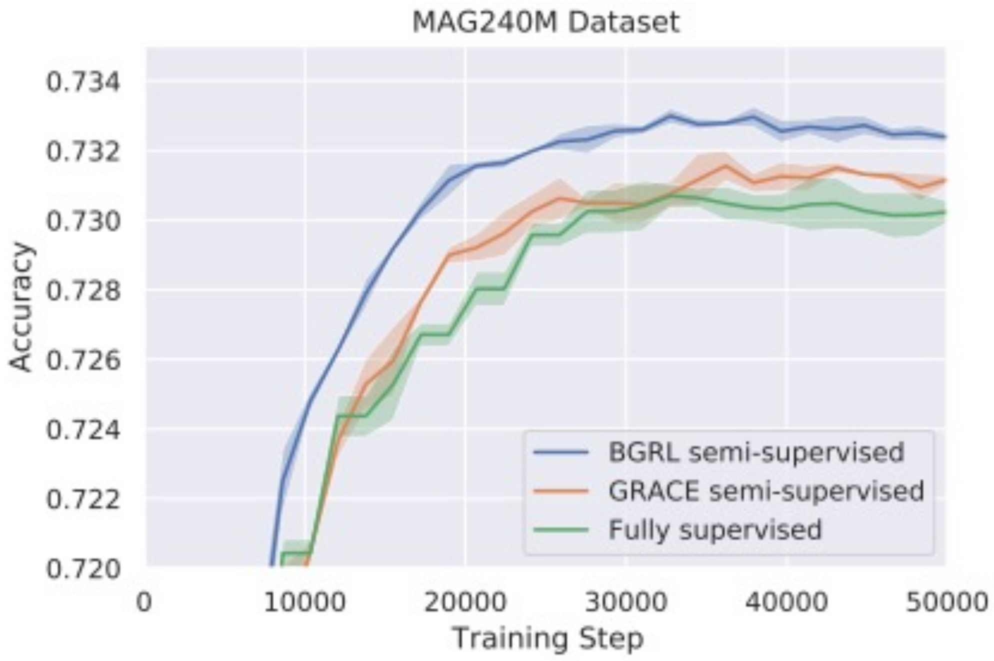


Figure 4: Performance on MAG240M using BGRL or GRACE-SUBSAMPLING as an auxiliary signal, averaged over 5 seeds and run for 50k steps.

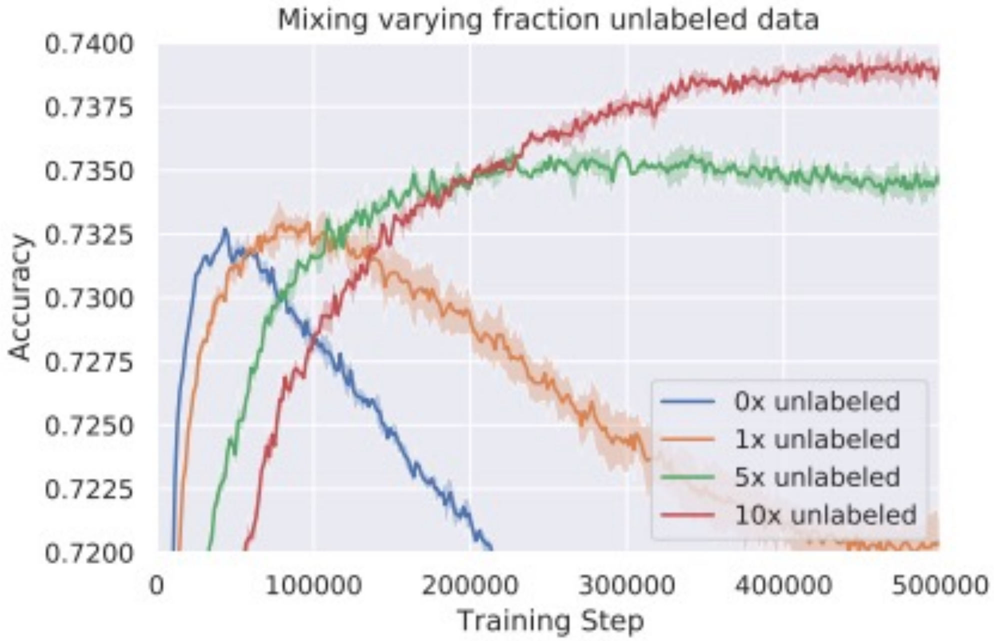
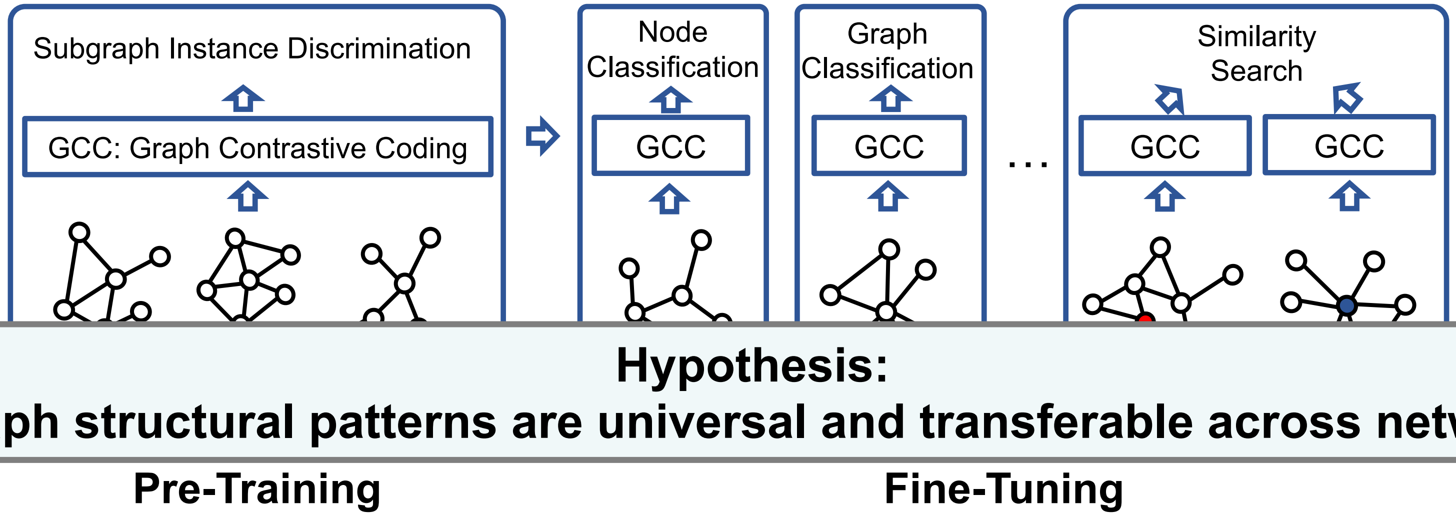


Figure 5: Mixing varying amounts of unlabeled data for representation learning with BGRL, averaged over 5 seeds and run for 500k steps.

Graph Contrastive Coding (GCC)

- Problem:
  - Learn a function f that maps a vertex to a low-dimensional vector
  - **Structural similarity**: map vertices with similar local network topologies close in the vector space
  - **Transferability**: compatible with vertices and graphs from various sources, even unseen during training time.



Hypothesis:
Graph structural patterns are universal and transferable across networks.

[1] Jiezhong Qiu et al. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In KDD'20.

GCC Pre-training

- **Pre-training Task: Instance** Discrimination
- **InfoNCE Loss:** output **instance representations** that are capable of capturing the **similarities** between instances

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

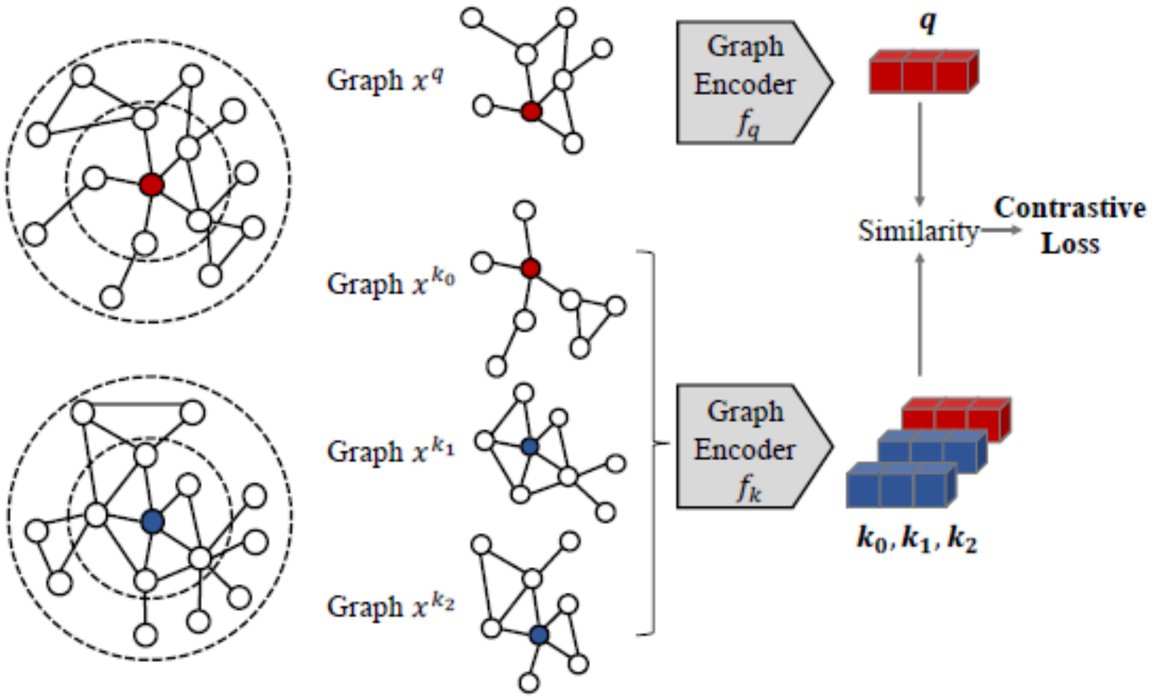
- query instance x^q
- query $\boldsymbol{q}$ (embedding of x^q), i.e., $\boldsymbol{q} = f(x^q)$
- dictionary of keys $\{\boldsymbol{k}_0, \boldsymbol{k}_1, \cdots, \boldsymbol{k}_K\}$
- key $\boldsymbol{k} = f(x^k)$

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

GCC Pre-training

- **Q1**: How to define **instances** in graphs?
- **Q2**: How to define **(dis) similar instance**?
- **Q3**: What are the proper **encoders**?

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



GCC Pre-training: Learning Algorithms

- Optimizing Contrastive Loss
  - Encoded query $\boldsymbol{q}$
  - $K + 1$ encoded keys $\{\boldsymbol{k}_0, \cdots, \boldsymbol{k}_K\}$

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

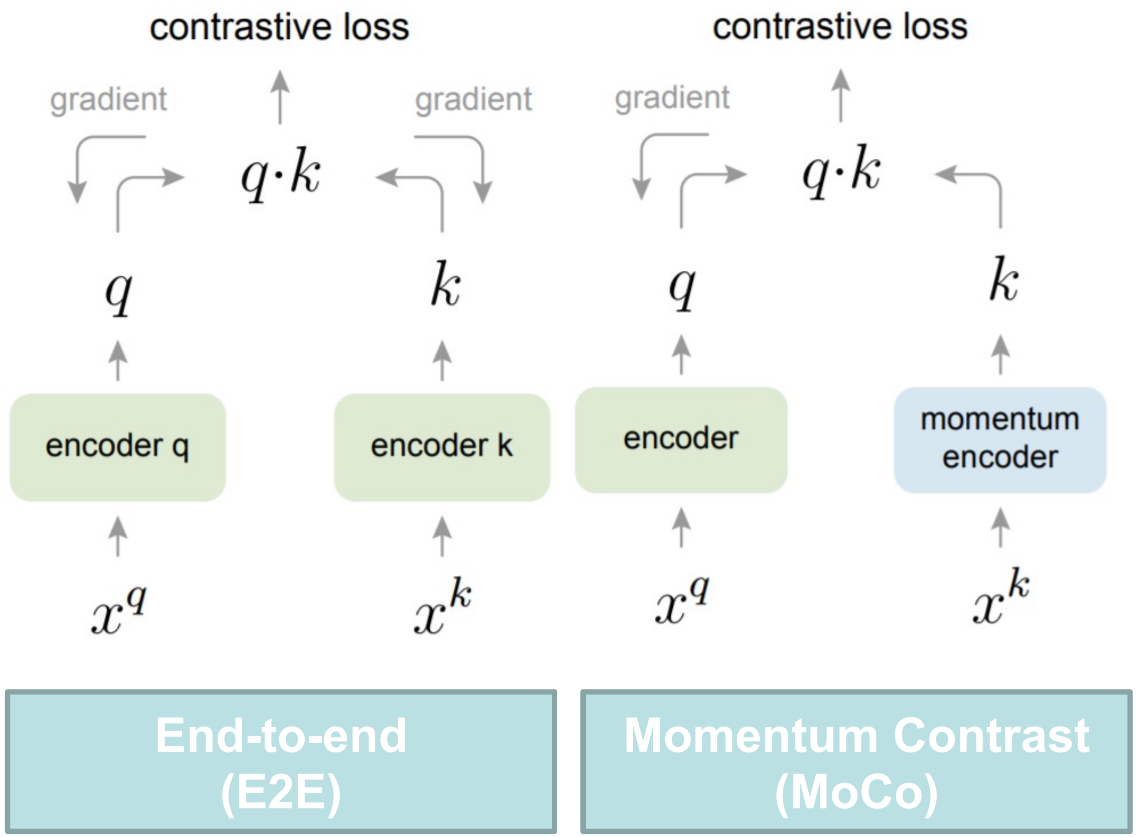
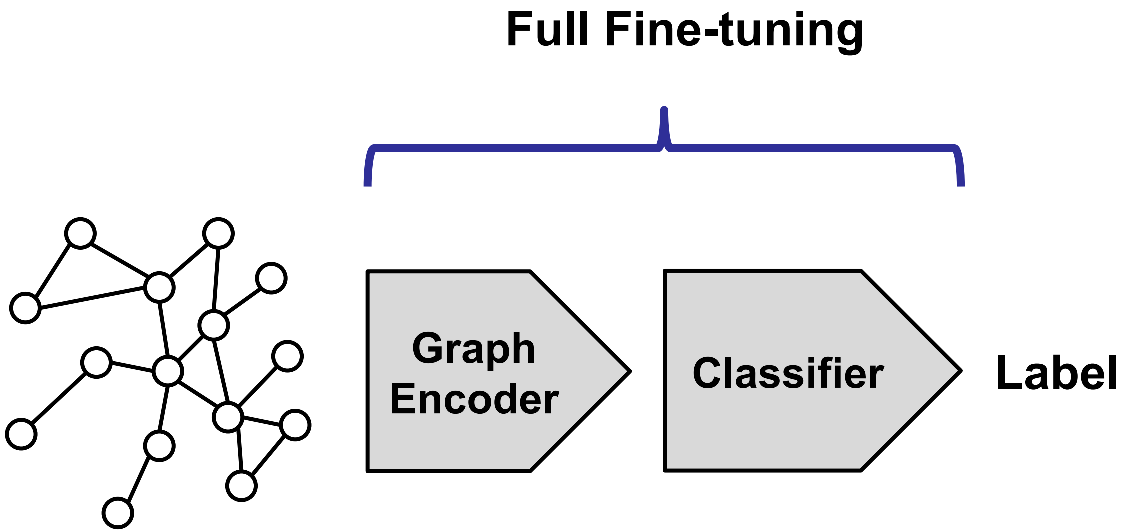


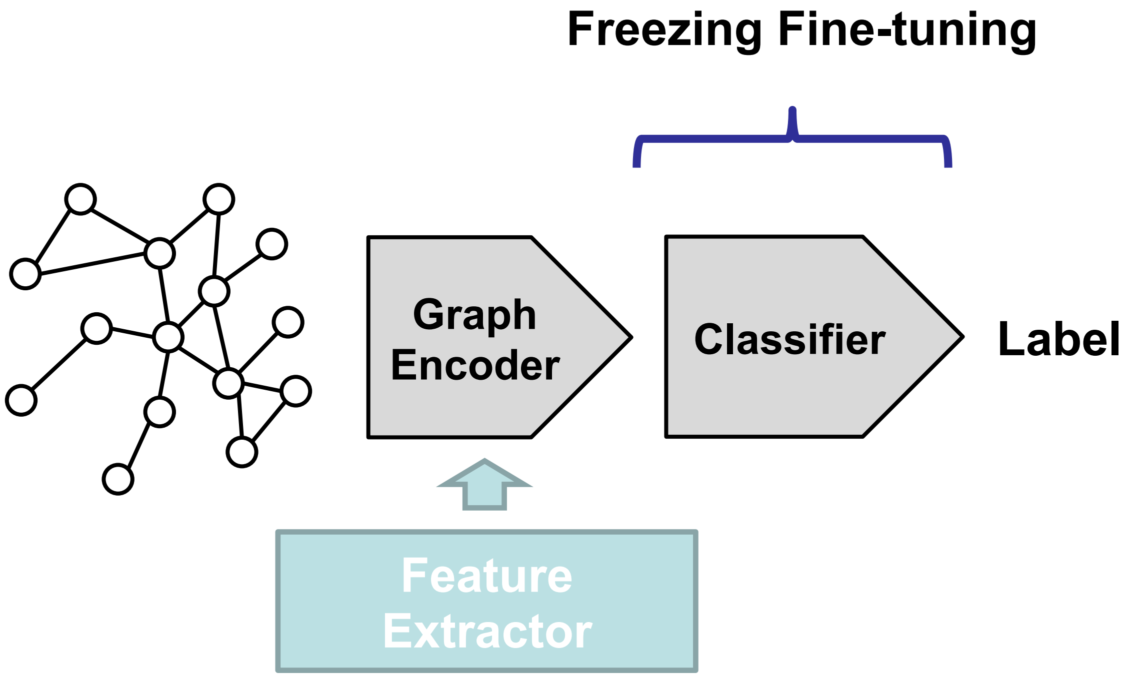
Figure Credit: Momentum Contrast for Unsupervised Visual Representation Learning (arxiv.org/abs/1911.05722)

GCC Fine-tuning: Full v.s. Freezing

Full fine-tuning



Freezing fine-tuning

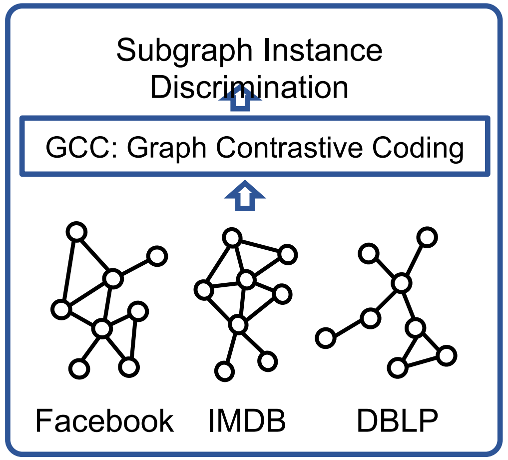


GCC Pre-Training / Fine-tuning

- Six real-world information networks for pre-training.

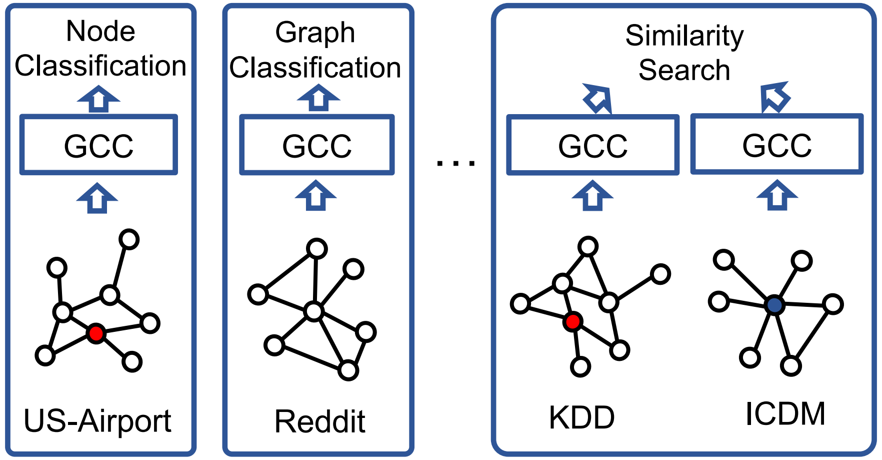
Table 1: Datasets for pre-training, sorted by number of vertices.

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



Pre-Training

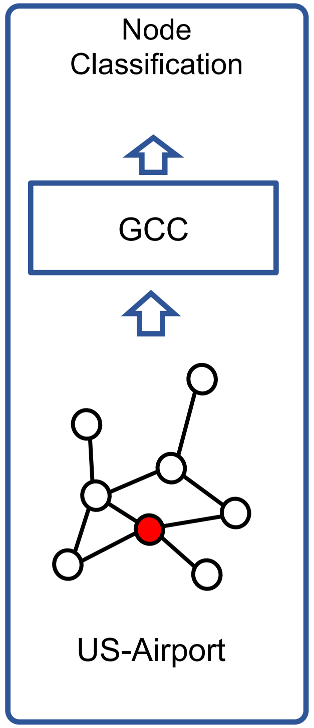
- Fine-tuning Tasks:
  - Node classification
  - Graph classification
  - Top-k Similarity search



Fine-Tuning

Task 1: Node Classification

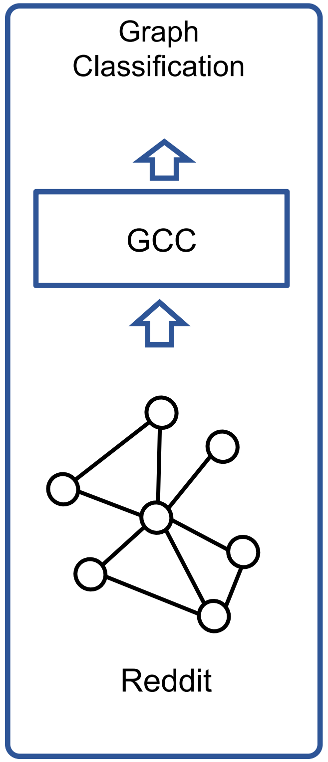
- Setup
  - US-Airport
  - AMiner academic graph



Datasets	US-Airport	H-index
$\lvert V \rvert$	1,190	5,000
$\lvert E \rvert$	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**

Task 2: Graph Classification

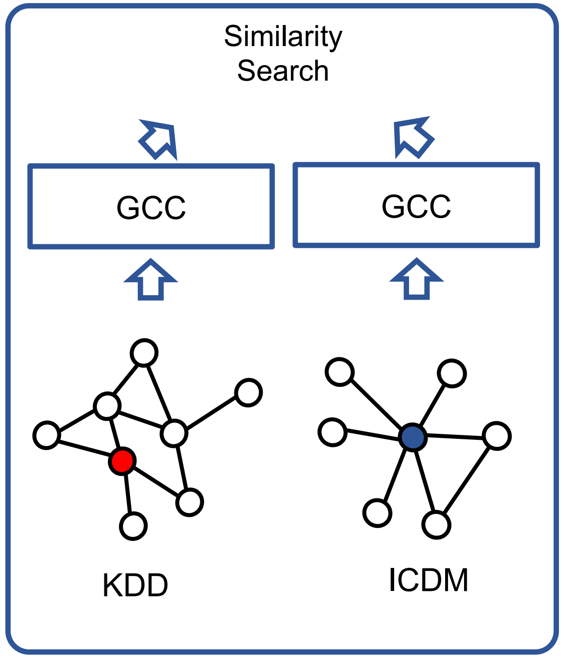
- Setup
  - COLLAB, RDT-B, RDT-M, & IMDB-B, IMDB-M



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

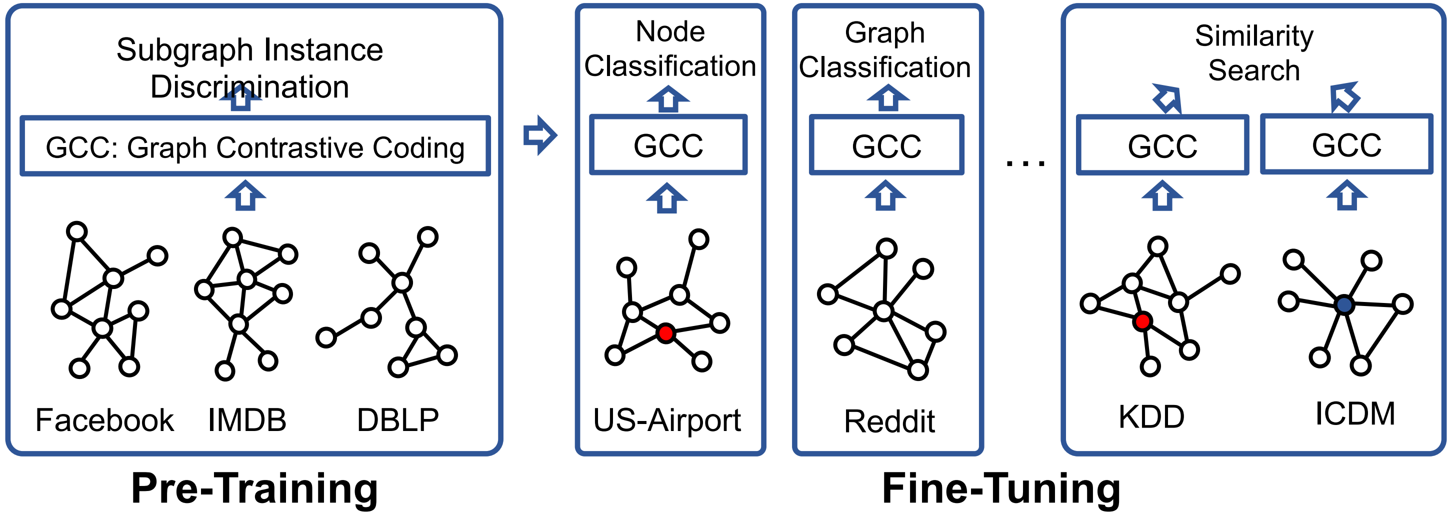
Task 3: Top-k Similarity Search

- 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

Summary of GCC



- We study the pre-training of GNN with the goal of characterizing and transferring structural representations in social and information networks.
- We present Graph Contrastive Coding, which is a graph-based contrastive learning framework to pre-train GNN.
- The pre-trained GNN achieves competitive performance to its supervised trained-from-scratch counterparts in 3 graph learning tasks on 10 graph datasets.

Generative Learning on Graphs

Web-Scale Graphs

Academic Graph

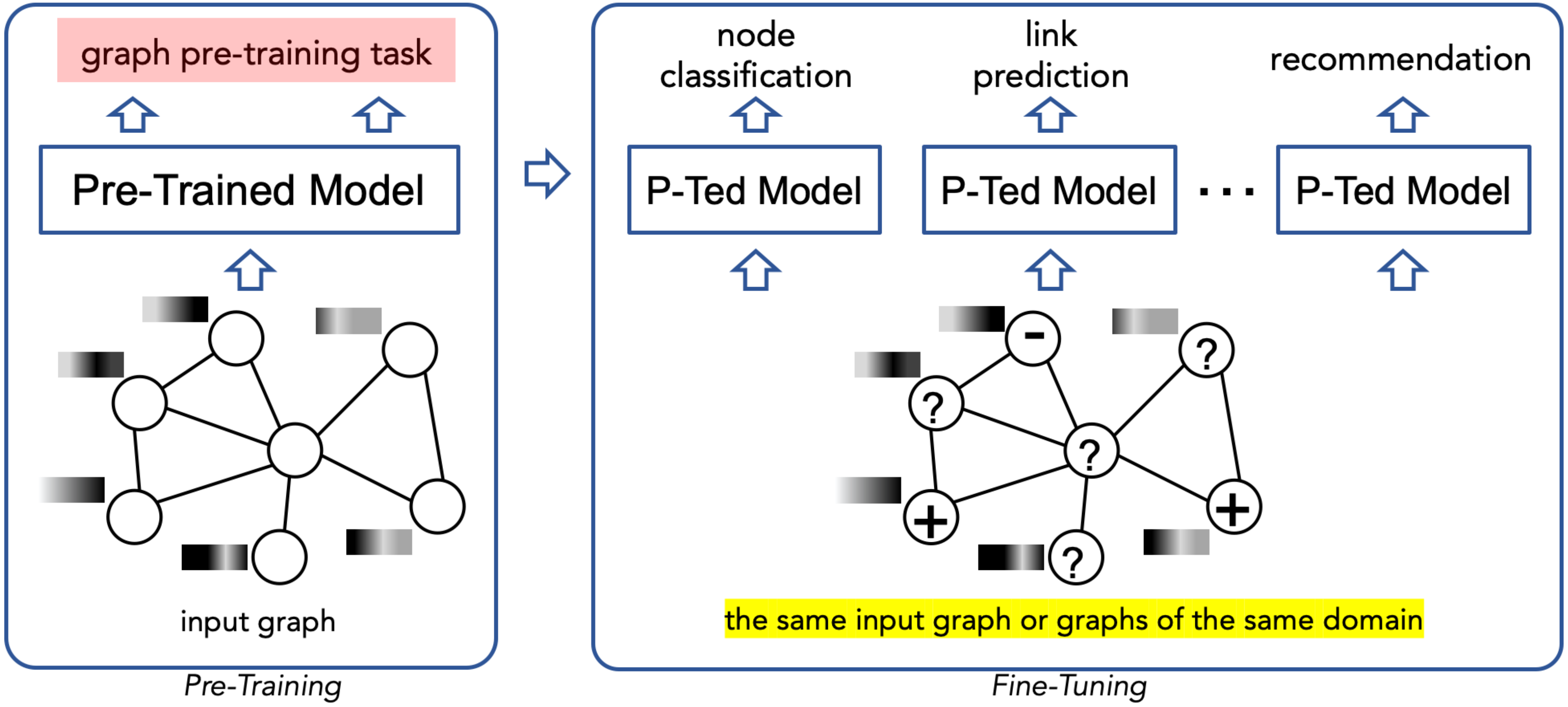
Microsoft Office Graph

LinkedIn Economic Graph

Facebook Entity Graph

Figure Credit: Microsoft/LinkedIn/Facebook

GNN Pre-Training

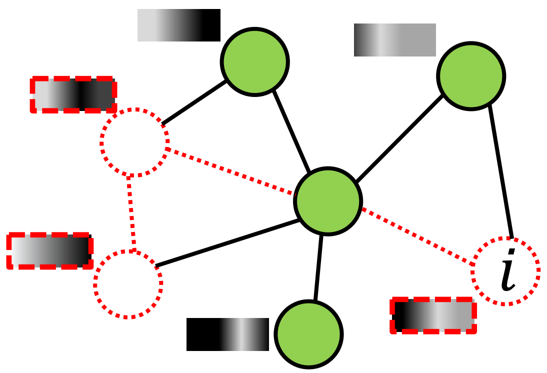


1.Ziniu Hu et al. GPT-GNN: Generative Pre-Training of Graph Neural Networks. **KDD** 2020.

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

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

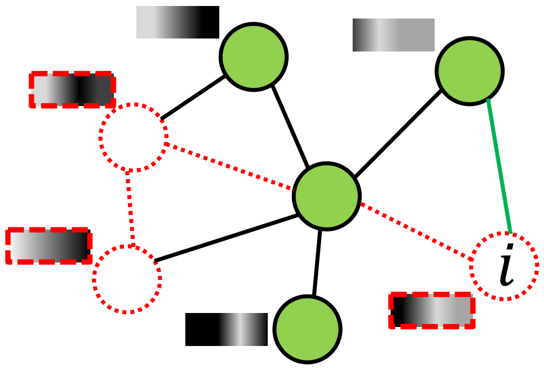
?

Lose the dependency between X_i and E_i

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

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

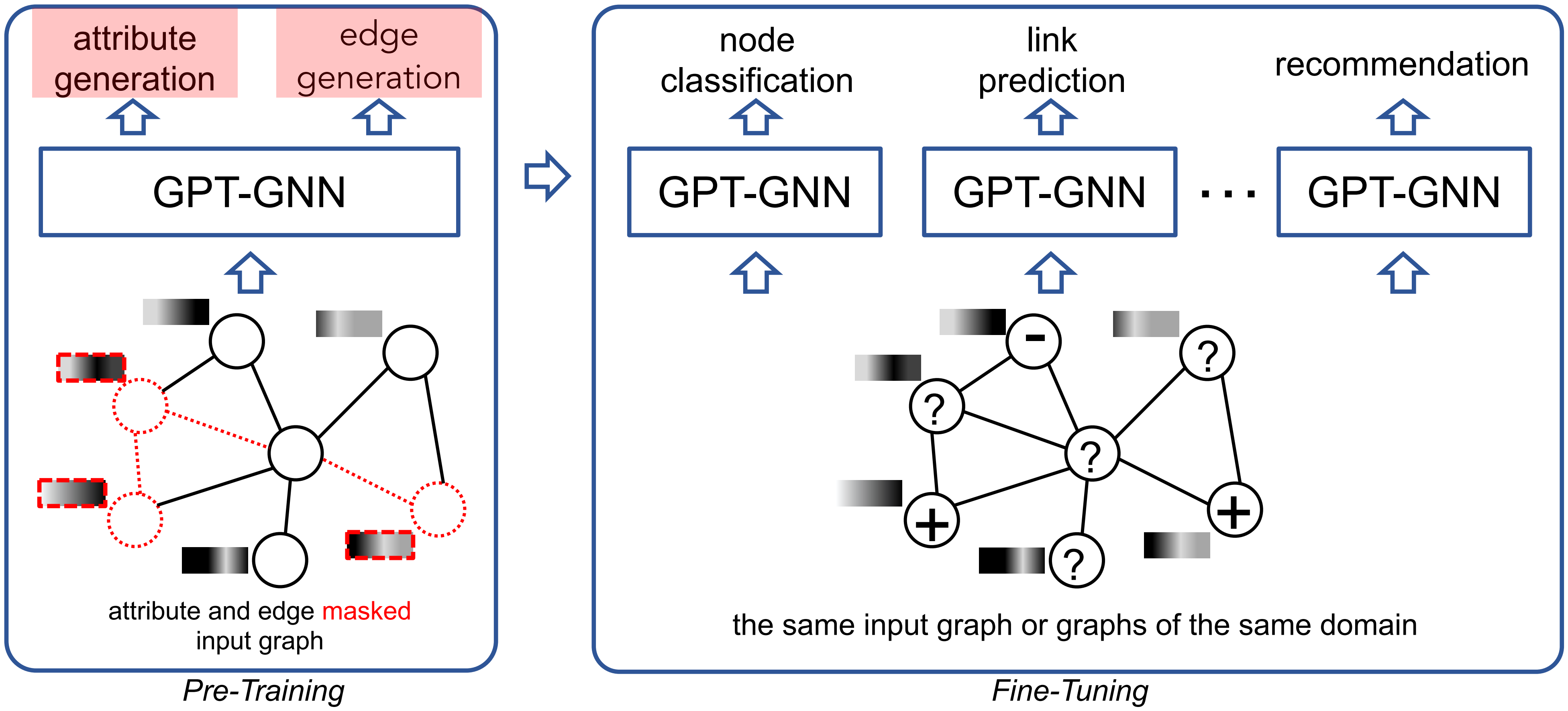


attribute and edge masked
input graph

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

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

GPT-GNN: Generative Pre-Training of GNNs



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.Ziniu Hu et al. Heterogeneous Graph Transformer. **WWW** 2020 (most cited in WWW'20).

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

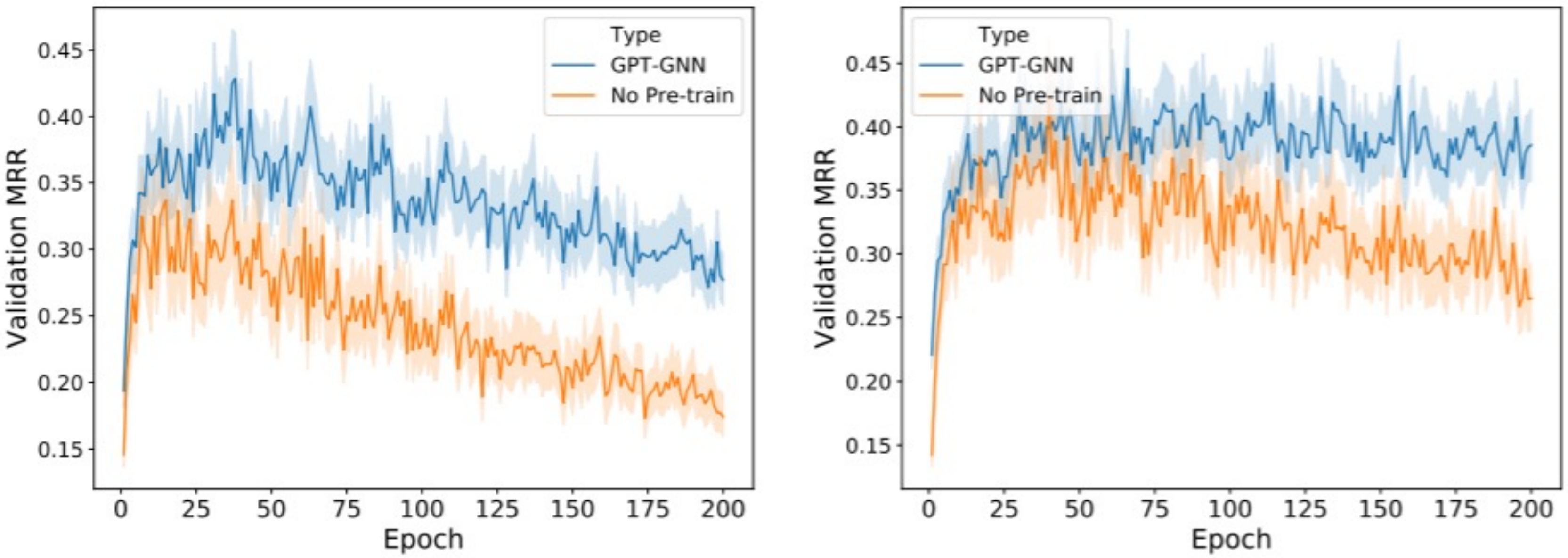
	Pre-Train	Fine-Tune
Tasks:	• Attribute Generation • Edge Generation	• Inferring the topic of each paper • Inferring the venue of each paper • Author name disambiguation

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.

The Promise of Graph Pre-Training!

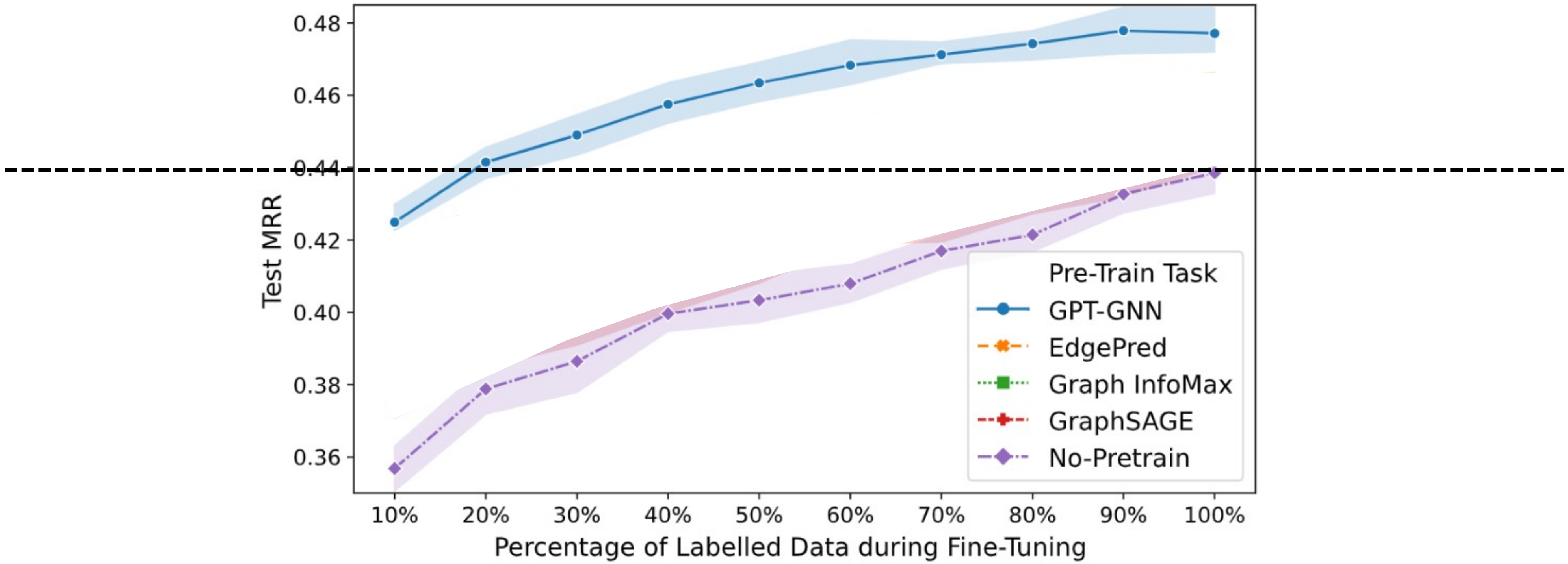


(a) Data Percentage: 10% (b) Data Percentage: 20%

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 with **100%** training data
VS
The pre-trained GNN model with **10-20%** training data

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

Powering the Microsoft Office Graph

One enterprise graph (monthly)

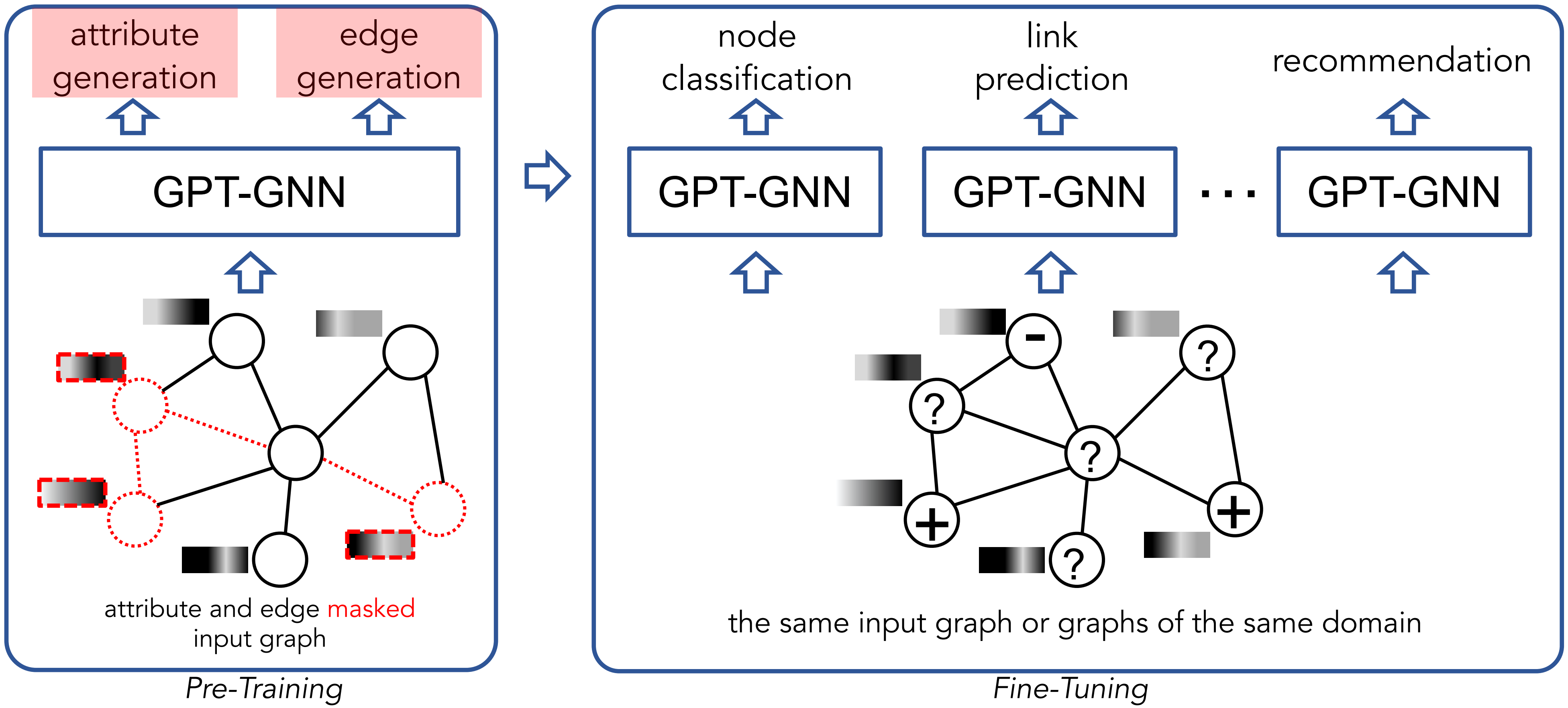
- 1.6 billion entities
  - 7 types of entities
- 7.8 trillion edges

Anomaly detection on Microsoft Office Graph

	Prec.	Recall	F1	Accu.
GraphSage	+0.00	+0.09	+0.06	+0.03
Graph Attention	+0.01	+0.11	+0.08	+0.03
HGT	+0.01	+0.30	+0.19	+0.07

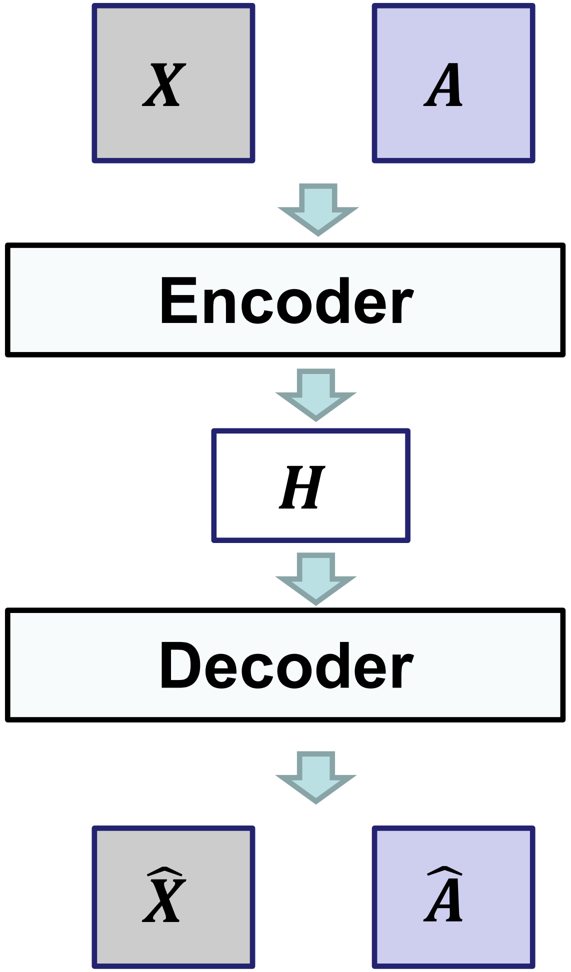
Pre-trained HGT on one enterprise ⇨ Other enterprise customers w/o data access

GPT-GNN: Generative Pre-Training of GNNs



Graph AutoEncoder

- $G = (V, A, X)$
  - $A \in \{0,1\}^{N\times N}$: adjacency matrix
  - $X \in \mathbb{R}^{N\times d}$: node features
- Encoding
  - $H = f_E(A, X)$
- Decoding
  - $G' = f_D(A, H)$
- Reconstruction objectives
  - graph structure (link)
  - node features

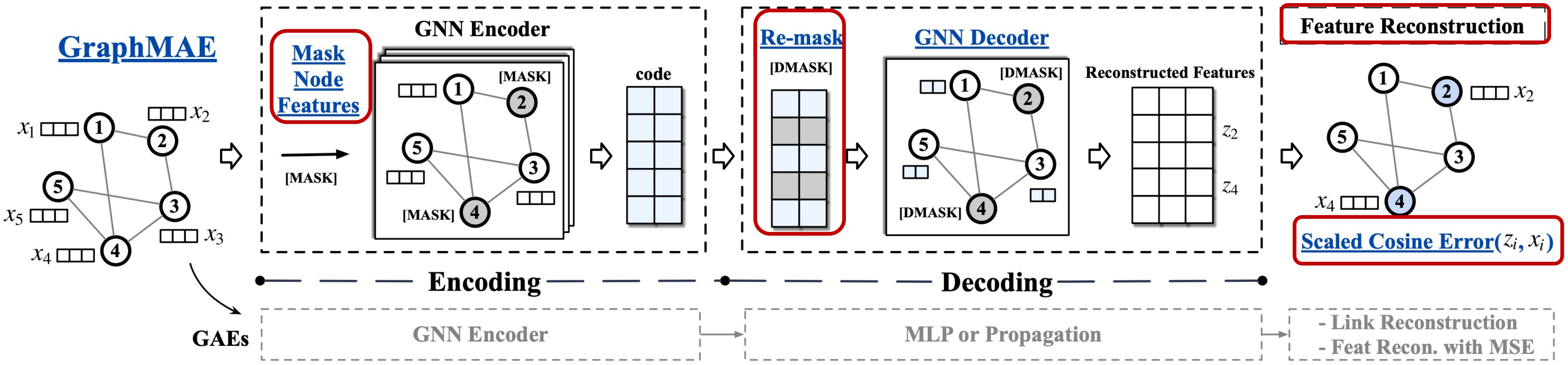


Reconstruction Target **Decoding Strategy**

Methods	Feat. Loss	AE	No Struc.	Mask Feat.	GNN Decoder	Re-mask Dec.	Space
VGAE [20]	n/a	✓	-	-	-	-	$O(N^2)$
ARVGA [26]	n/a	✓	-	-	-	-	$O(N^2)$
MGAE [42]	MSE	✓	-	✓	-	-	$O(N)$
GALA [27]	MSE	✓	✓	-	✓	-	$O(N)$
GATE [31]	MSE	✓	-	-	✓	-	$O(N)$
AttrMask [16]	CE	✓	✓	✓	-	-	$O(N)$
GPT-GNN [17]	MSE	-	-	✓	-	-	$O(N)$
AGE [3]	n/a	✓	-	-	-	-	$O(N^2)$
NodeProp [18]	MSE	✓	✓	✓	-	-	$O(N)$

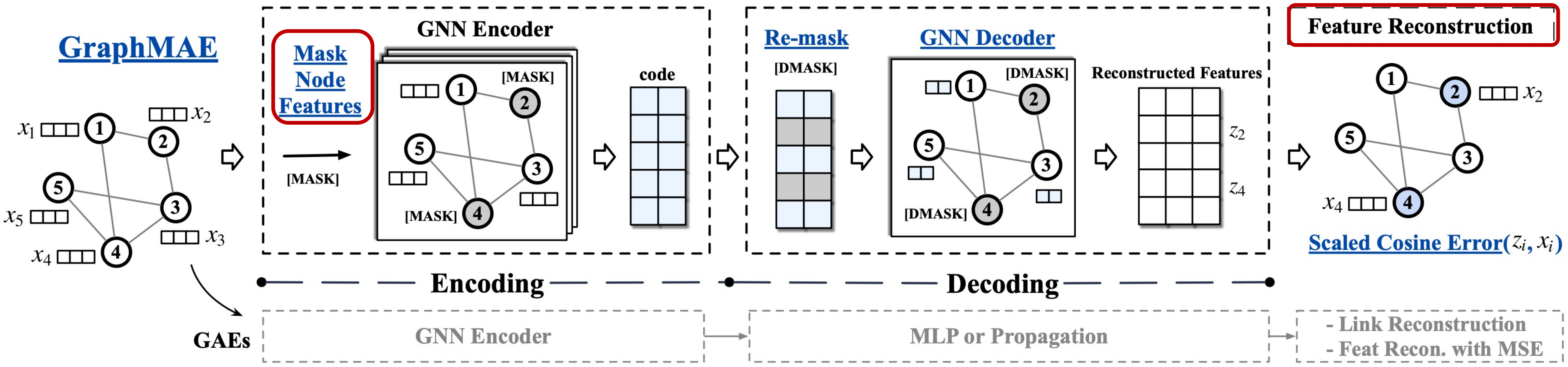
Error Function **Reconstruction Method**

GraphMAE



GraphMAE
Mask Node Features
GNN Encoder
[MASK]
code
Re-mask
[DMASK]
GNN Decoder
Reconstructed Features
Feature Reconstruction
Scaled Cosine Error(z_i, x_i)
Encoding
Decoding
GAEs
GNN Encoder
MLP or Propagation
- Link Reconstruction
- Feat Recon. with MSE

Masked Feature Reconstruction

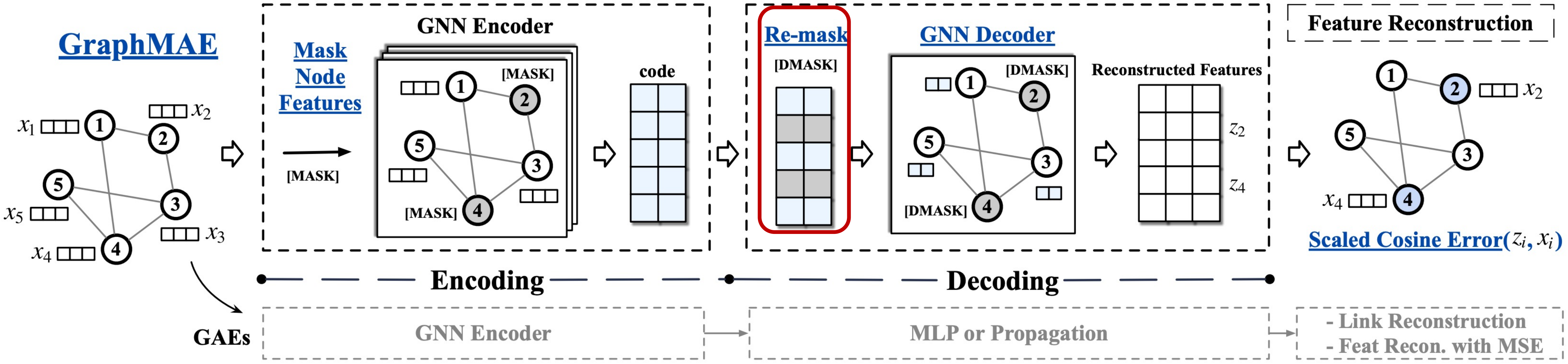


– Feature construction as the learning objective
– Masked feature reconstruction
  1. Sample a subset of nodes $\widetilde{V} \subset V$
  2. Replace node feature with [MASK]

$$\widetilde{\boldsymbol{x}}_i = \begin{cases} \boldsymbol{x}_{[M]} & v_i \in \widetilde{\mathcal{V}} \\ \boldsymbol{x}_i & v_i \notin \widetilde{\mathcal{V}} \end{cases}$$

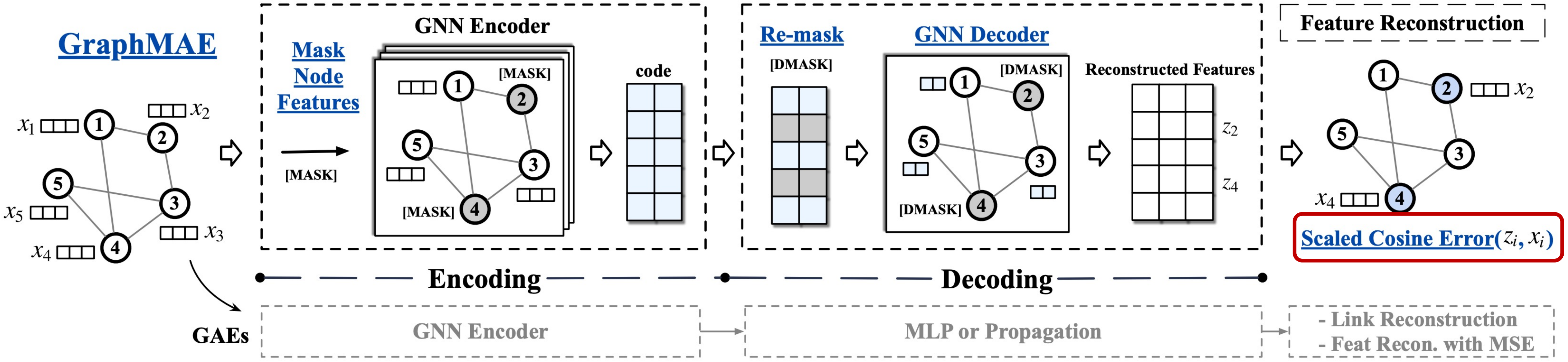
- $H = f_E(A, X)$

GNNs as Decoder with Re-Mask Decoding



- Use a GNN as the decoder
  - A more expressive decoder helps reconstruct low informative features
- Re-mask node features before decoder
  - Re-mask the “masked” nodes
- $\widetilde{H} = \mathrm{Remask}(H),\ Z = f_D(A, \widetilde{H})$ $\quad \tilde{\boldsymbol{h}}_i = \begin{cases} \boldsymbol{h}_{[M]} & v_i \in \widetilde{\mathcal{V}} \\ \boldsymbol{h}_i & v_i \notin \widetilde{\mathcal{V}} \end{cases}$

Scaled Cosine Error as the Criterion



- MSE fails, especially for continuous features
  - *Sensitivity & low selectivity*

$$L_{MSE} = \frac{1}{|\tilde{V}|} \sum_{v_i \in \tilde{V}} (x_i - z_i)^2$$

- Scaled cosine error as the criterion
  - *Cosine error & scaled coefficient*

$$\mathcal{L}_{\text{SCE}} = \frac{1}{|\widetilde{\mathcal{V}}|} \sum_{v_i \in \widetilde{\mathcal{V}}} \left(1 - \frac{\boldsymbol{x}_i^T \boldsymbol{z}_i}{\|\boldsymbol{x}_i\| \cdot \|\boldsymbol{z}_i\|}\right)^{\gamma}, \ \gamma \geq 1,$$

Reconstruction Target

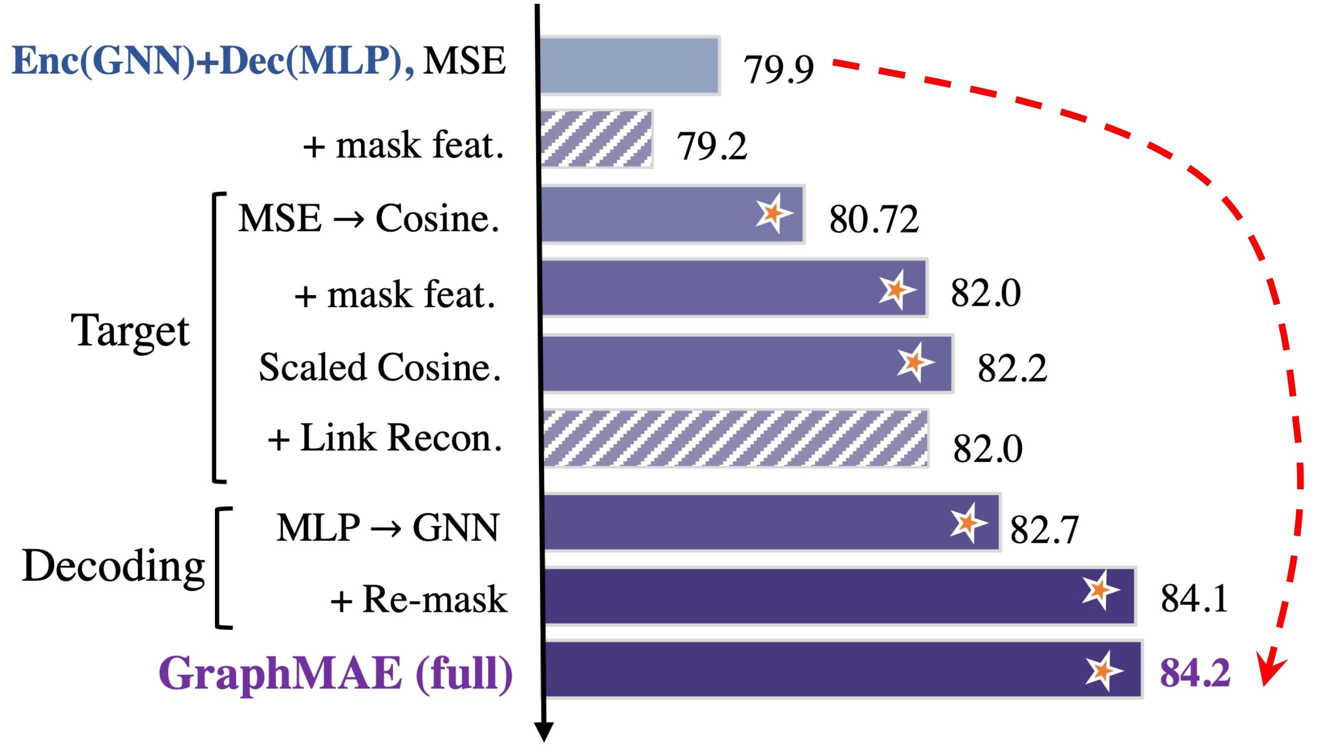
Decoding Strategy

Methods	Feat. Loss	AE	No Struc.	Mask Feat.	GNN Decoder	Re-mask Dec.	Space
VGAE [20]	n/a	✓	-	-	-	-	$O(N^2)$
ARVGA [26]	n/a	✓	-	-	-	-	$O(N^2)$
MGAE [42]	MSE	✓	-	✓	-	-	$O(N)$
GALA [27]	MSE	✓	✓	-	✓	-	$O(N)$
GATE [31]	MSE	✓	-	-	✓	-	$O(N)$
AttrMask [16]	CE	✓	✓	✓	-	-	$O(N)$
GPT-GNN [17]	MSE	-	-	✓	-	-	$O(N)$
AGE [3]	n/a	✓	-	-	-	-	$O(N^2)$
NodeProp [18]	MSE	✓	✓	✓	-	-	$O(N)$
GraphMAE	SCE	✓	✓	✓	✓	✓	$O(N)$

Error Function

Reconstruction Method

GraphMAE



(b) The effect of GraphMAE designs on the performance on Cora dataset.

Table 4: Ablation studies of decoder type, re-mask and reconstruction criterion on node- and graph-level benchmarks.

	Dataset	Node-Level			Graph-Level	
		Cora	PubMed	Arxiv	MUTAG	IMDB-B
COMP.	GraphMAE	84.2	81.1	71.75	88.19	75.52
	w/o mask	79.7	77.9	70.97	82.58	74.42
	w/o re-mask	82.7	80.0	71.61	86.29	74.42
	w/ MSE	79.1	73.1	67.44	86.30	74.04
Decoder	MLP	82.2	80.4	71.54	87.16	73.94
	GCN	81.3	79.1	71.59	87.78	74.54
	GIN	81.8	80.2	71.41	88.19	75.52
	GAT	84.2	81.1	71.75	86.27	74.04

Downstream Tasks

Node Classification

Table 1: Experiment results in unsupervised representation learning for node classification. We report Micro-F1(%) score for PPI and accuracy(%) for the other datasets.

	Dataset	Cora	CiteSeer	PubMed	Ogbn-arxiv	PPI	Reddit
Supervised	GCN	81.5	70.3	79.0	71.74±0.29	75.7±0.1	95.3±0.1
	GAT	83.0±0.7	72.5±0.7	79.0±0.3	72.10±0.13	97.30±0.20	96.0±0.1
Self-supervised	GAE	71.5±0.4	65.8±0.4	72.1±0.5	-	-	-
	GPT-GNN	80.1±1.0	68.4±1.6	76.3±0.8	-	-	-
	GATE	83.2±0.6	71.8±0.8	80.9±0.3	-	-	-
	DGI	82.3±0.6	71.8±0.7	76.8±0.6	70.34±0.16	63.80±0.20	94.0±0.10
	MVGRL	83.5±0.4	73.3±0.5	80.1±0.7	-	-	-
	GRACE[1]	81.9±0.4	71.2±0.5	80.6±0.4	71.51±0.11	69.71±0.17	94.72±0.04
	BGRL[1]	82.7±0.6	71.1±0.8	79.6±0.5	71.64±0.12	73.63±0.16	94.22±0.03
	InfoGCL	83.5±0.3	**73.5±0.4**	79.1±0.2	-	-	-
	CCA-SSG[1]	84.0±0.4	73.1±0.3	81.0±0.4	71.24±0.20	73.34±0.17	95.07±0.02
	GraphMAE	**84.2±0.4**	73.4±0.4	**81.1±0.4**	**71.75±0.17**	**74.50±0.29**	**96.01±0.08**

Graph Classification

Table 2: Experiment results in unsupervised representation learning for graph classification. We report accuracy(%) for all datasets.

	Dataset	IMDB-B	IMDB-M	PROTEINS	COLLAB	MUTAG	REDDIT-B	NCI1
Supervised	GIN	75.1±5.1	52.3±2.8	76.2±2.8	80.2±1.9	89.4±5.6	92.4±2.5	82.7±1.7
	DiffPool	72.6±3.9	-	75.1±3.5	78.9±2.3	85.0±10.3	92.1±2.6	-
Graph Kernels	WL	72.30±3.44	46.95±0.46	72.92±0.56	-	80.72±3.00	68.82±0.41	80.31±0.46
	DGK	66.96±0.56	44.55±0.52	73.30±0.82	-	87.44±2.72	78.04±0.39	80.31±0.46
Self-supervised	graph2vec	71.10±0.54	50.44±0.87	73.30±2.05	-	83.15±9.25	75.78±1.03	73.22±1.81
	Infograph	73.03±0.87	49.69±0.53	74.44±0.31	70.65±1.13	89.01±1.13	82.50±1.42	76.20±1.06
	GraphCL	71.14±0.44	48.58±0.67	74.39±0.45	71.36±1.15	86.80±1.34	89.53±0.84	77.87±0.41
	JOAO	70.21±3.08	49.20±0.77	74.55±0.41	69.50±0.36	87.35±1.02	85.29±1.35	78.07±0.47
	GCC	72.0	49.4	-	78.9	-	**89.8**	-
	MVGRL	74.20±0.70	51.20±0.50	-	-	89.70±1.10	84.50±0.60	-
	InfoGCL	75.10±0.90	51.40±0.80	-	80.00±1.30	**91.20±1.30**	-	80.20±0.60
	GraphMAE	**75.52±0.66**	**51.63±0.52**	**75.30±0.39**	**80.32±0.46**	88.19±1.26	88.01±0.19	**80.40±0.30**

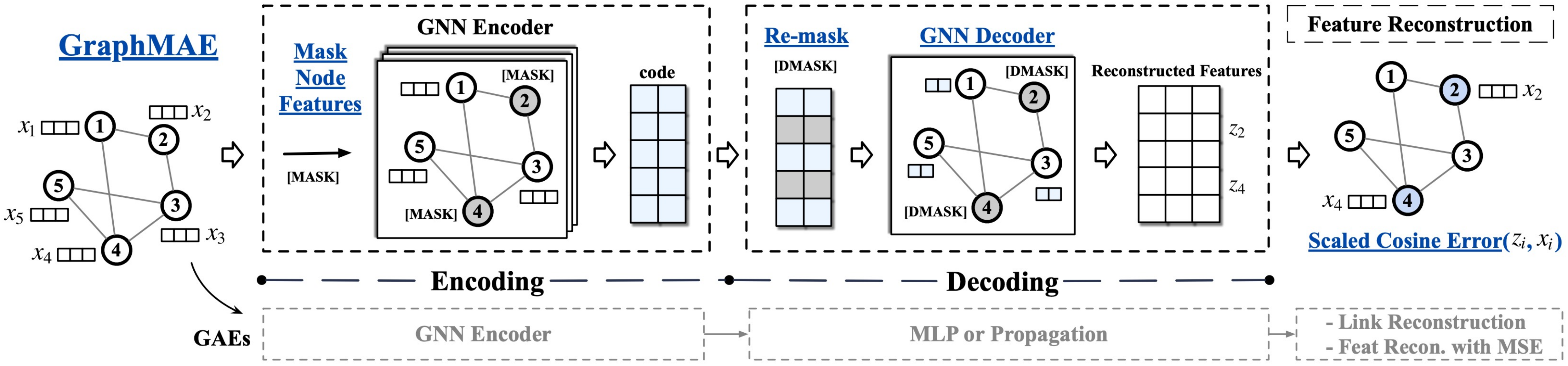
Transfer Learning

Table 3: Experiment results in transfer learning on molecular property prediction benchmarks. The model is first pre-trained on ZINC15 and then finetuned on the following datasets. We report ROC-AUC(%) scores.

	BBBP	Tox21	ToxCast	SIDER	ClinTox	MUV	HIV	BACE	Avg.
No-pretrain	65.5±1.8	74.3±0.5	63.3±1.5	57.2±0.7	58.2±2.8	71.7±2.3	75.4±1.5	70.0±2.5	67.0
ContextPred	64.3±2.8	75.7±0.7	63.9±0.6	60.9±0.6	65.9±3.8	75.8±1.7	77.3±1.0	79.6±1.2	70.4
AttrMasking	64.3±2.8	**76.7±0.4**	**64.2±0.5**	61.0±0.7	71.8±4.1	74.7±1.4	77.2±1.1	79.3±1.6	71.1
Infomax	68.8 ±0.8	75.3 ±0.5	62.7 ±0.4	58.4 ±0.8	69.9±3.0	75.3 ±2.5	76.0 ±0.7	75.9 ±1.6	70.3
GraphCL	69.7±0.7	73.9±0.7	62.4±0.6	60.5±0.9	76.0±2.7	69.8±2.7	**78.5±1.2**	75.4±1.4	70.8
JOAO	70.2±1.0	75.0±0.3	62.9±0.5	60.0±0.8	81.3±2.5	71.7±1.4	76.7±1.2	77.3±0.5	71.9
GraphLoG	**72.5±0.8**	75.7±0.5	63.5±0.7	**61.2±1.1**	76.7±3.3	76.0±1.1	77.8±0.8	**83.5±1.2**	73.4
GraphMAE	72.0±0.6	75.5±0.6	64.1±0.3	60.3±1.1	**82.3±1.2**	**76.3±2.4**	77.2±1.0	83.1±0.9	**73.8**

Code: https://github.com/THUDM/GraphMAE

Summary of GraphMAE



1. Generative SSL on Graphs vs. Contrastive Learning on Graphs
2. Identify the common issues in current graph autoencoders
3. Present a simple masked graph autoencoder—**GraphMAE**

Code: https://github.com/THUDM/GraphMAE

Reflection & Motivation

- Problems in masked-feature-prediction
  – more *sensitive to the discriminability of input features.*

	Cora raw → *w/ PCA*	PubMed raw → *w/ PCA*
Supervised	83.0 → 82.3 (↓ 0.7)	78.0 → 77.0 (↓ 1.0)
GraphMAE	84.2 → 82.6 (↓ 1.6)	81.1 → 78.9 (↓ 2.2)
GraphMAE2	84.5 → 83.5 (↓ 1.0)	81.4 → 80.1 (↓ 1.3)

- *raw* : the original node features
- *w/ PCA* : the input features are reduced to 50-dimensional vectors using PCA

***Resolution*: imposing regularization on target reconstruction**

GraphMAE2

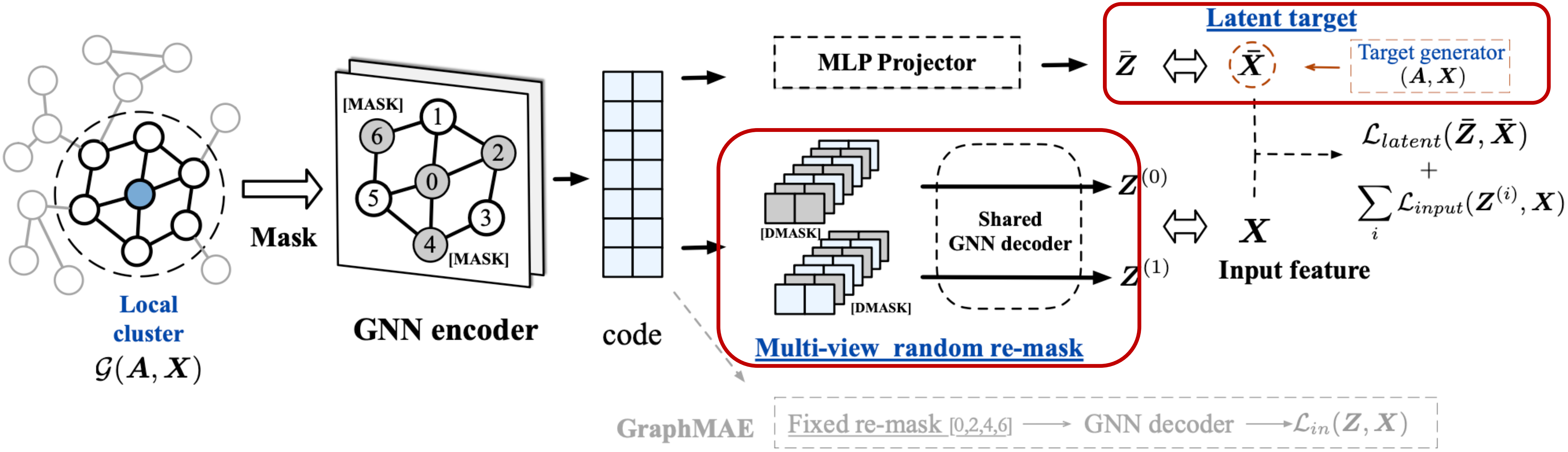
A Decoding-Enhanced Masked Self-Supervised Graph Learner.

10:00-10:10 AM

Thursday, May 4, 2023

@Classroom 107

GraphMAE2 Framework



- Multi-view random re-mask decoding
- Latent representation prediction
- Scaling to large-scale graphs with local clustering

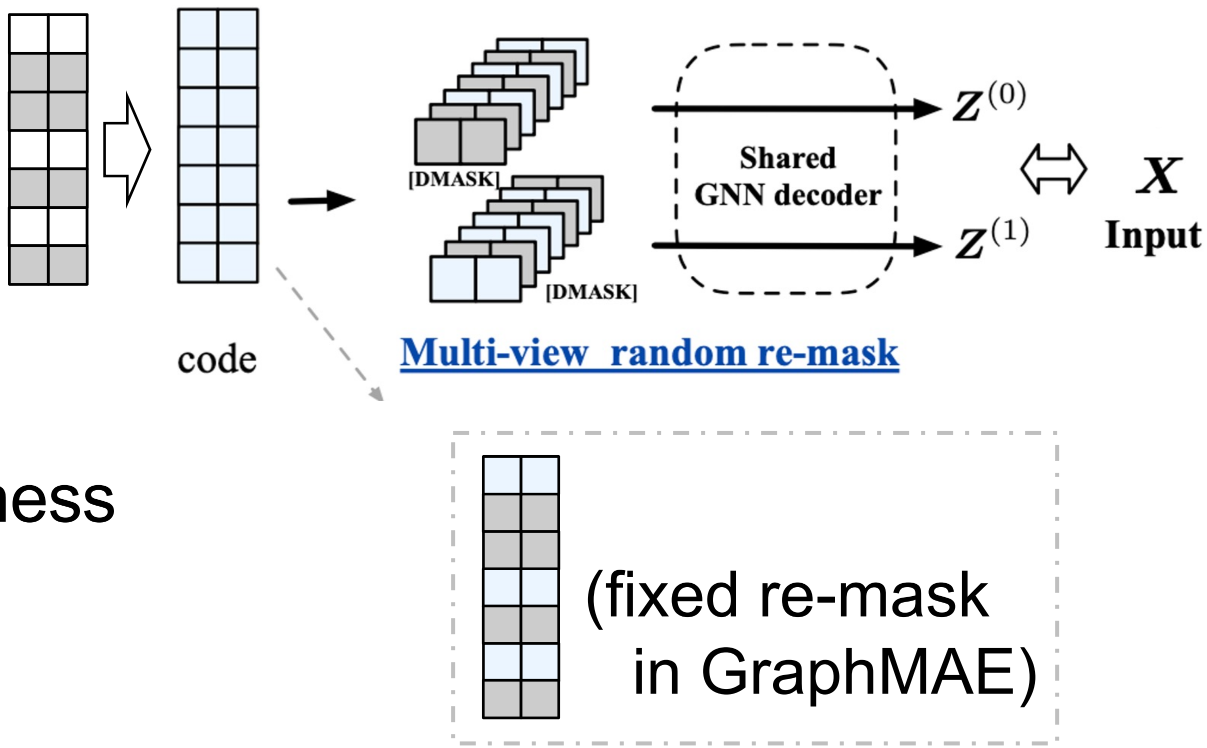
Multi-View Random Re-Mask Decoding

– Avoid representation overfitting to input features

– **Randomly** re-mask representations/code

- $\widetilde{H} = \mathrm{Remask}(H),\ Z = f_D(A, \widetilde{H})$

$$\widetilde{\boldsymbol{h}}_i = \begin{cases} \boldsymbol{h}_{[M]} & v_i \in \overline{\mathcal{V}} \\ \boldsymbol{h}_i & v_i \notin \overline{\mathcal{V}} \end{cases}$$



– **Multiple** re-masking

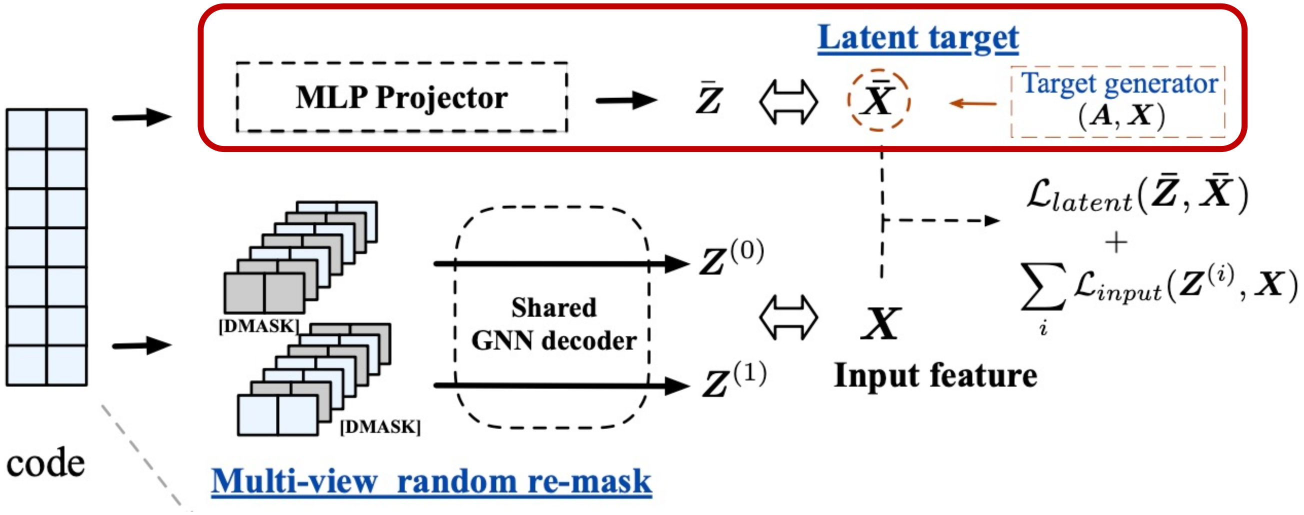
- *K*-different randomly re-masking
- better generalization and effectiveness

$$\mathcal{L}_{input} = \frac{1}{|\widetilde{\mathcal{V}}|} \sum_{j=1}^{K} \sum_{v_i \in \widetilde{\mathcal{V}}} \left(1 - \frac{\boldsymbol{x}_i^\top \boldsymbol{z}_i^{(j)}}{\|\boldsymbol{x}_i\| \cdot \|\boldsymbol{z}_i^{(j)}\|}\right)^{\gamma}$$

Latent Representation Prediction

- Additional informative prediction target
  - Minimally affected by input features & and GNN as a denoiser
- Predicting **masked latent representations**
  - A (*momentum*) target generator $f_{target}(\cdot|\xi)$
  - Prediction: $\bar{Z} = f_E(\mathrm{mask}(G); \theta)$
  - Latent target: $\bar{X} = f_{target}(G; \xi)$
    - $\xi \leftarrow \tau \cdot \xi + (1 - \tau) \cdot \theta$

$$\mathcal{L}_{latent} = \frac{1}{N} \sum_{i}^{N} \left(1 - \frac{\bar{\boldsymbol{z}}_i^\top \bar{\boldsymbol{x}}_i}{\|\bar{\boldsymbol{z}}\| \cdot \|\bar{\boldsymbol{x}}\|}\right)^\gamma$$



Experiments

- Embedding evaluation with linear probing
- On large-scale graphs

Datasets	#Nodes	#Edges
Cora	2,485	5,069
Citeseer	2,110	3,668
Pubmed	19,717	44,324
ogbn-Arxiv	169,343	1,166,243
ogbn-Products	2,449,029	61,859,140
MAG-Scholar-F	12,403,930	358,010,024
ogbn-Papers100M	111,059,956	1,615,685,872

Code: https://github.com/THUDM/GraphMAE2

Linear Probing

- **Setting: training a linear classifier**
- Results:
  1. GraphMAE2 consistently outperforms all baselines
  2. *Random-Init* models can achieve decent results

	Arxiv	Products	MAG	Papers100M
MLP	55.50±0.23	61.06±0.08	39.11±0.21	47.24±0.31
SGC	66.92±0.08	74.87±0.25	54.68±0.23	63.29±0.19
Random-Init	68.14±0.02	74.04±0.06	56.57±0.03	61.55±0.12
CCA-SSG	68.57±0.02	75.27±0.05	51.55±0.03	55.67±0.15
GRACE	69.34±0.01	79.47±0.59	57.39±0.02	61.21±0.12
BGRL	70.51±0.03	78.59±0.02	57.57±0.01	62.18±0.15
GGD[1]	-	75.70±0.40	-	63.50±0.50
GraphMAE	71.03±0.02	78.89±0.01	58.75±0.03	62.54±0.09
GraphMAE2	**71.89**±0.03	**81.59**±0.02	**59.24**±0.01	**64.89**±0.04

	Cora	CiteSeer	PubMed
GCN	81.5	70.3	79.0
GAT	83.0±0.7	72.5±0.7	79.0±0.3
GAE	71.5±0.4	65.8±0.4	72.1±0.5
DGI	82.3±0.6	71.8±0.7	76.8±0.6
MVGRL	83.5±0.4	73.3±0.5	80.1±0.7
GRACE	81.9±0.4	71.2±0.5	80.6±0.4
BGRL	82.7±0.6	71.1±0.8	79.6±0.5
InfoGCL	83.5±0.3	**73.5**±0.4	79.1±0.2
CCA-SSG	84.0±0.4	73.1±0.3	81.0±0.4
GGD	83.9±0.4	73.0±0.6	81.3±0.8
GraphMAE	84.2±0.4	73.4±0.4	81.1±0.4
GraphMAE2	**84.5**±0.6	73.4±0.3	**81.4**±0.5

Code: https://github.com/THUDM/GraphMAE2

Ablation Studies

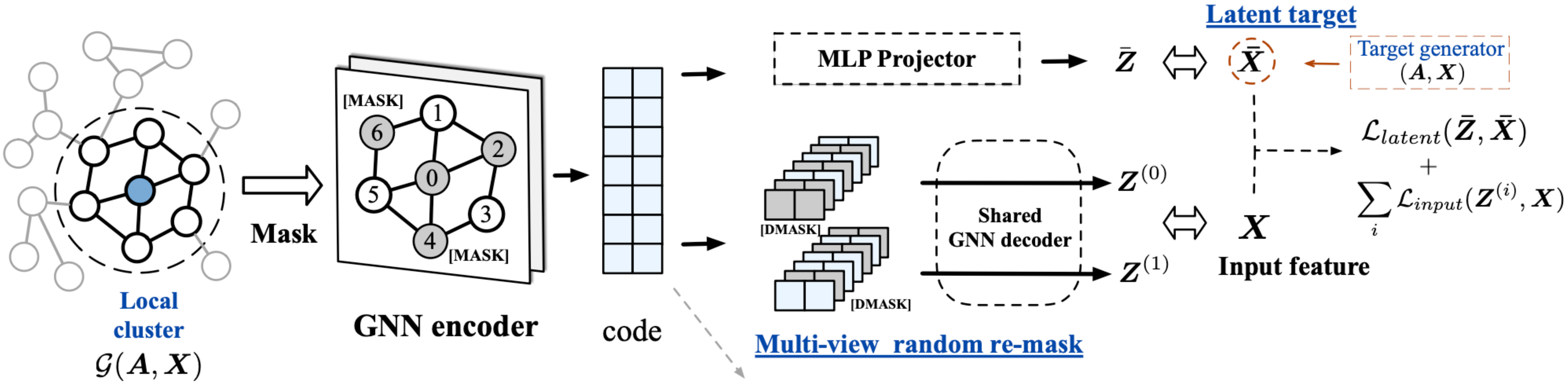
Component Ablation of GraphMAE2

- GraphMAE2 surpasses all baselines with the same sampling strategy
- Using local clusters brings further improvement

	Strategy	Products	MAG	Papers100M
GRACE	*SAINT*	79.47±0.59	57.39±0.02	61.21±0.12
BGRL	*SAINT*	78.59±0.02	57.57±0.01	62.18±0.15
GraphMAE2	*SAINT*	80.96±0.03	58.75±0.03	64.21±0.11
GraphMAE2	*Cluster*	79.35±0.05	58.05±0.02	63.77±0.11
GraphMAE2	*LC*	81.59±0.02	59.24±0.01	64.89±0.12

Code: https://github.com/THUDM/GraphMAE2

GraphMAE2 Summary



- Analyze the problem in masked feature prediction
- Present GraphMAE2 with improved decoding strategies
- GraphMAE2 achieves promising performance in large-scale graphs

GraphMAE2: https://github.com/THUDM/GraphMAE2
GraphMAE: https://github.com/THUDM/GraphMAE

Generative Learning on Graphs

GPT-GNN
(KDD’20)

⇨

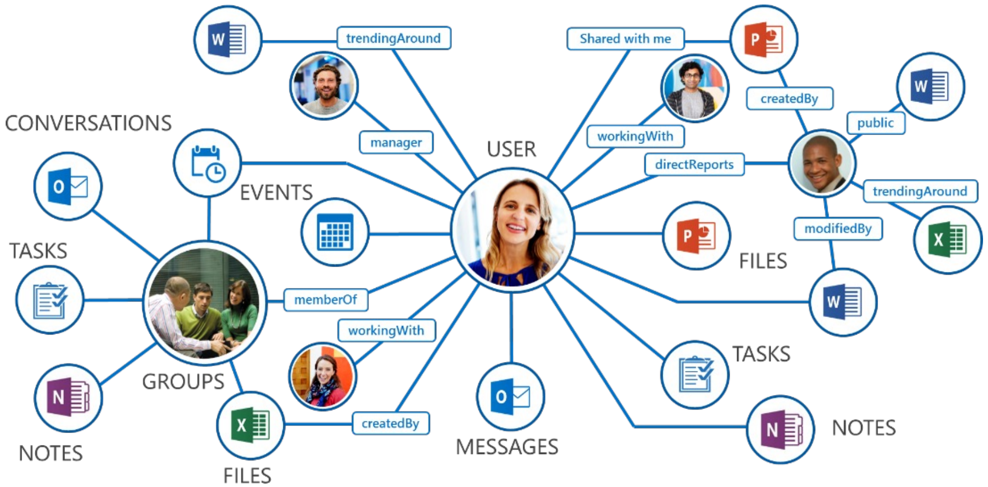
GraphMAE
(KDD’22)

⇨

GraphMAE2
(WWW’23)

Pre-Train Graphs with *Language/Image/Knowledge*

Academic Graph



Microsoft Office Graph

LinkedIn Economic Graph

Facebook Entity Graph

Figure Credit: Microsoft/LinkedIn/Facebook

Neural Symbolic Reasoning

Diabetic diet
Ketoacidosis
Blurred vision
Polyphagia
Kidney disease
Autoimmunity
Polyuria
Insulin
Insulin resistance
Nephropathy
Polydipsia
Diabetes
Weight loss
Glycosuria
Insulin glargine
Streptozotocin
Edmonton protocol
Diabetic retinopathy
Cytokine
Insulin glulisine
Insulin lispro
Albuminuria
Regular insulin
Glucagon
Pramlintide
Insulin aspart
Pregnancy
Physical exercise
Pancreas transplantation
Insulin degludec
Autoimmune disease
Microalbuminuria
Miglitol
Muscle cramp
Acarbose

Cause

Symptom

Treatment

Neural Symbolic Reasoning

Nephropathy
Autoimmunity
Insulin resistance
Pregnancy
Diabetic retinopathy
Kidney disease

Cause

Muscle cramp
Blurred vision
Glycosuria
Polyphagia
Polydipsia
Polyuria
Weight loss
Diabetes
Ketoacidosis
Albuminuria
Autoimmune disease
Microalbuminuria

Symptom

Physical exercise
Insulin
Insulin aspart
Streptozotocin
Pramlintide
Insulin degludec
Cytokine
Glucagon
Acarbose
Insulin glargine
Miglitol
Pancreas transplantation
Diabetic diet
Insulin lispro
Insulin glulisine
Regular insulin
Edmonton protocol

Treatment

GNNs vs. LLMs

GNN:

GPT-GNN (KDD'20) ⇨ GraphMAE (KDD'22) ⇨ GraphMAE2 (WWW'23)

LLM:

GLM-130B — An Open Bilingual Pre-Trained Model

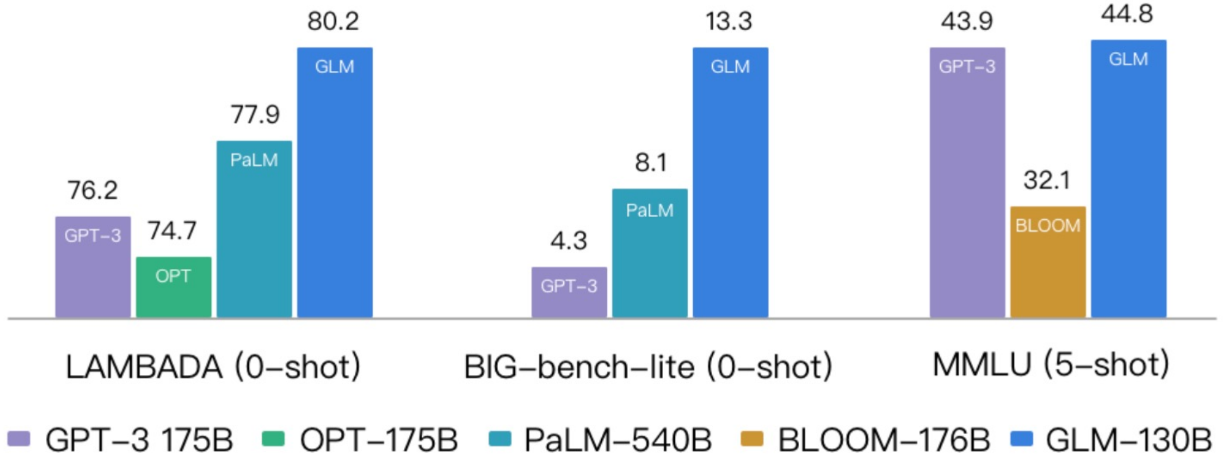
CodeGeeX — A Multilingual Code Generation Model

ChatGLM

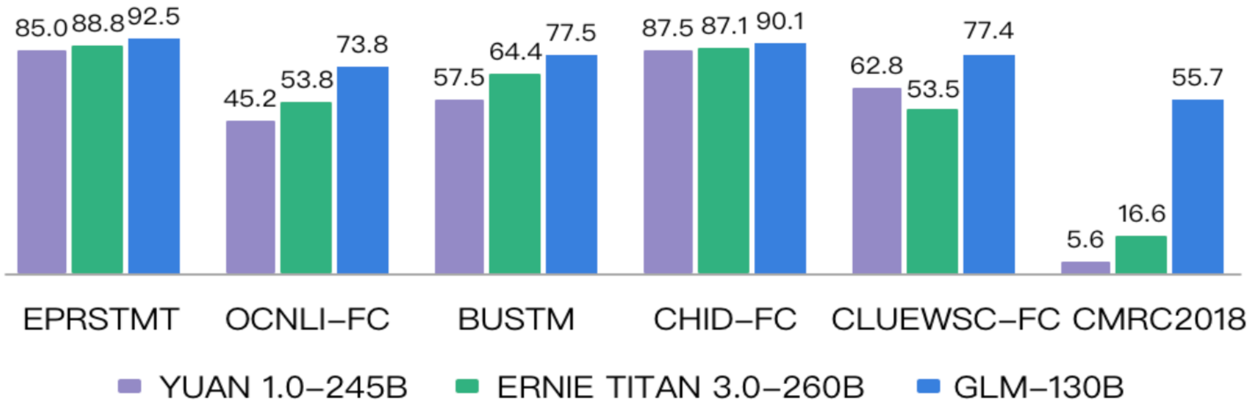
https://github.com/THUDM

GLM-130B

English: better than GPT-3/OPT/PaLM on MMLU, LAMBADA, BIG-bench-lite



Chinese: better than ERNIE 260B & YUAN 245B



Since Aug., 2022, requests from ~1000 orgs in 69 countries

- Google
- Microsoft
- Facebook
- Stanford
- MIT
- UC Berkely
- CMU
- Harvard
- Princeton
- Yale
- Cornell
- UIUC
- Cambridge
- Oxford

- Huawei
- Alibaba
- Tencent
- Baidu
- Meituan
- Bytedance
- Didi
- Xiaoice
- Xiaodu
- Xiaomi
- Xiaopeng
- Youdao
- Face++
- Ping An Cap

- Peking U.
- Zhejiang U.
- Shanghai JT U.
- Fudan U.
- USTC
- U of CAS
- Wuhan U.
- Naikai U.
- Hongkong U.
- CUHK
- HKUST
- BAAI
- Zhejiang Lab
- Shanghai AI Lab

GLM-130B

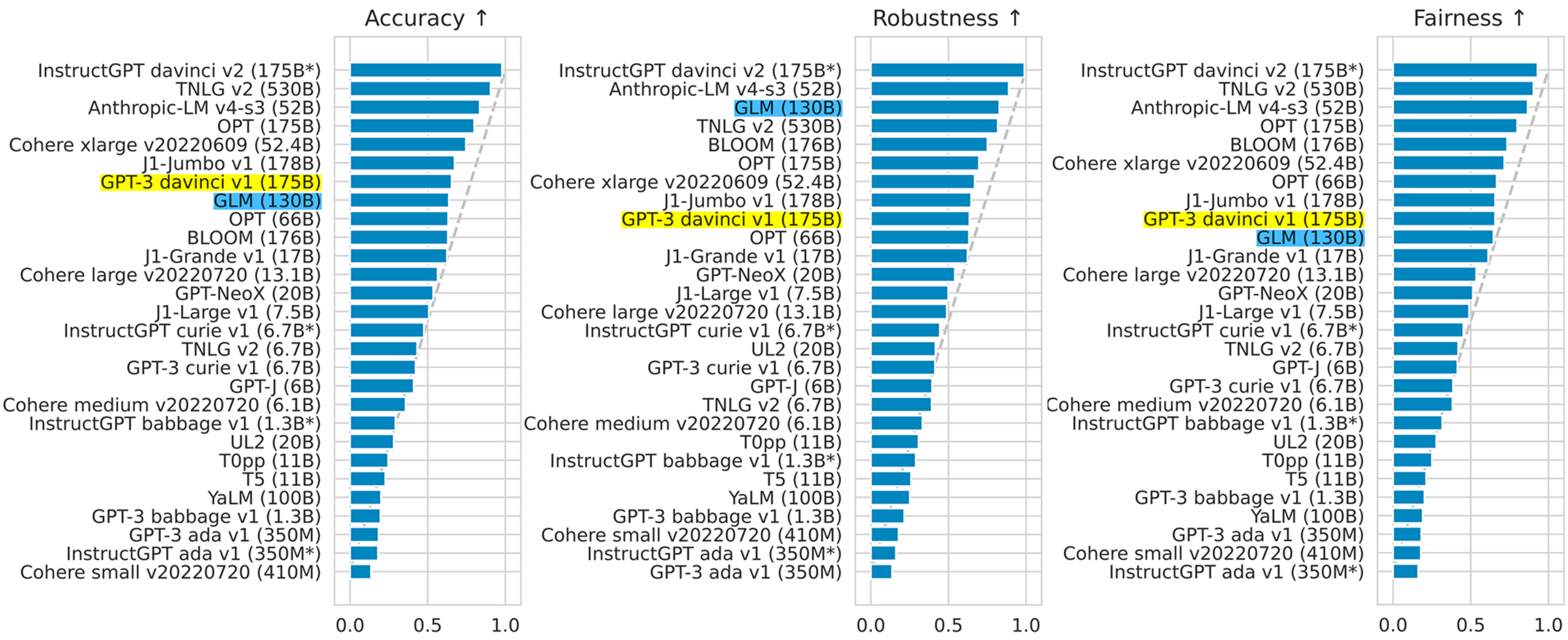
The only model from academia was covered by Stanford's HELM

Model	Model Creator	Modality	# Parameters	Tokenizer	Window Size	Access	Total Tokens	Total Queries	Total Cost
J1-Jumbo v1 (178B)	AI21 Labs	Text	178B	AI21	2047	limited	327,443,515	591,384	$10,926
J1-Grande v1 (17B)	AI21 Labs	Text	17B	AI21	2047	limited	326,815,150	591,384	$2,973
J1-Large v1 (7.5B)	AI21 Labs								
Anthropic-LM v4-s3 (52B)	Anthropic								
BLOOM (176B)	BigScience								
T0++ (11B)	BigScience								
Cohere xlarge v20220609 (52.4B)	Cohere								
Cohere large v20220720 (13.1B)[58]	Cohere								
Cohere medium v20220720 (6.1B)	Cohere								
Cohere small v20220720 (410M)[59]	Cohere								
GPT-J (6B)	EleutherAI								
GPT-NeoX (20B)	EleutherAI								
T5 (11B)	Google								
UL2 (20B)	Google								
OPT (66B)	Meta								
OPT (175B)	Meta								
TNLG v2 (6.7B)	Microsoft/NVIDIA								
TNLG v2 (530B)	Microsoft/NVIDIA								
GPT-3 davinci v1 (175B)	OpenAI								
GPT-3 curie v1 (6.7B)	OpenAI								
GPT-3 babbage v1 (1.3B)	OpenAI	Text	1.3B	GPT-2	2048	limited	422,123,900	606,253	$211
GPT-3 ada v1 (350M)	OpenAI	Text	350M	GPT-2	2048	limited	422,635,705	604,253	$169
InstructGPT davinci v2 (175B*)	OpenAI	Text	175B*	GPT-2	4000	limited	466,872,228	599,815	$9,337
InstructGPT curie v1 (6.7B*)	OpenAI	Text	6.7B*	GPT-2	2048	limited	420,004,477	606,253	$840
InstructGPT babbage v1 (1.3B*)	OpenAI	Text	1.3B*	GPT-2	2048	limited	419,036,038	604,253	$210
InstructGPT ada v1 (350M*)	OpenAI	Text	350M*	GPT-2	2048	limited	418,915,281	604,253	$168
Codex davinci v2	OpenAI	Code	Unknown	GPT-2	4000	limited	46,272,590	57,051	$925
Codex cushman v1	OpenAI	Code	Unknown	GPT-2	2048	limited	42,659,399	59,751	$85
GLM (130B)	Tsinghua University	Text	130B	ICE	2048	open	375,474,243	406,072	2,100 GPU hours
YaLM (100B)	Yandex	Text	100B	Yandex	2048	open	378,607,292	405,093	2,200 GPU hours

https://crfm.stanford.edu/helm, **2023.0308**

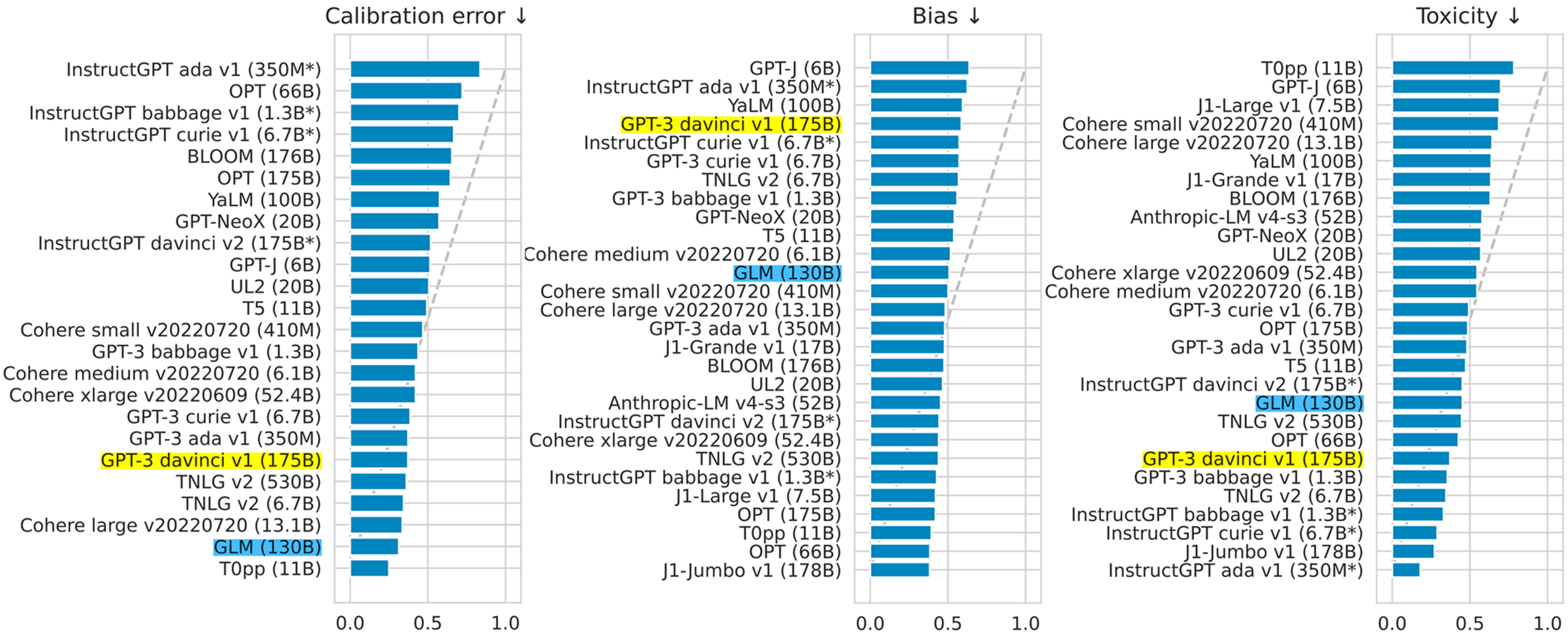
1.Liang et al., Holistic Evaluation of Language Models. arXiv: **2211.09110**

GLM-130B



1.Liang et al., Holistic Evaluation of Language Models. arXiv: **2211.09110**

GLM-130B



1.Liang et al., Holistic Evaluation of Language Models. arXiv: **2211.09110**

chatglm.cn

我们一直在路上 博客 内测邀请码

ChatGLM Alpha 内测

提出你的想法

ChatGLM

√‾

收养一只流浪猫好还是买一只...

2022年世界杯冠军是谁

...

试试这些例子：

列出一些年夜饭好意头的菜肴以及其寓意。

帮我写一篇人工智能课程的教案，1000字。

怎么修改huggingface transformers的model cache位置?

当前模型限制：

可能会生成不正确的信息

可能会产生有害说明或有偏见的内容

暂时不擅长逻辑类回答，如数学和编程类问题

Alpha/beta test from Mar. 14, 2023

An Open **ChatGLM-6B**

- Mar. 14, 2023, open-soured model
- Mar. 16, 2023, **#1** on GitHub Trending
- Mar. 18-30,   **#1** on Hugging Face Trending
- Apr. 30, 2023, 21.8k stars in GitHub

  1M downloads in HF

THUDM/chatglm-6b
Updated 3 days ago • ↓ 910k • ♥ 1.76k

THUDM/chatglm-6b-int4
Updated 3 days ago • ↓ 91.9k • ♡ 199

THUDM/chatglm-6b-int8
Updated 3 days ago • ↓ 14.1k • ♡ 12

https://huggingface.co/THUDM

Screenshots on Apr. 30, 2023

Thank you !

ChatGLM-6B Public

ChatGLM-6B: An Open Bilingual Dialogue Language Model | 开源双语对话语言模型

Python ☆ 21.9k 2.6k

GLM-130B Public

GLM-130B: An Open Bilingual Pre-Trained Model (ICLR 2023)

Python ☆ 5.1k 365

CodeGeeX Public

CodeGeeX: An Open Multilingual Code Generation Model

Python ☆ 4.8k 316

CogView Public

Text-to-Image generation. The repo for NeurIPS 2021 paper "CogView: Mastering Text-to-Image Generation via Transformers".

Python ☆ 1.4k 163

CogVideo Public

Text-to-video generation. The repo for ICLR2023 paper "CogVideo: Large-scale Pretraining for Text-to-Video Generation via Transformers"

Python ☆ 2.8k 286

cogdl Public

CogDL: A Comprehensive Library for Graph Deep Learning (WWW 2023)

Python ☆ 1.4k 300

slides

https://github.com/THUDM

@thukeg