Deep Graph Neural Networks

Yukuo Cen

GNN Center, Zhipu AI

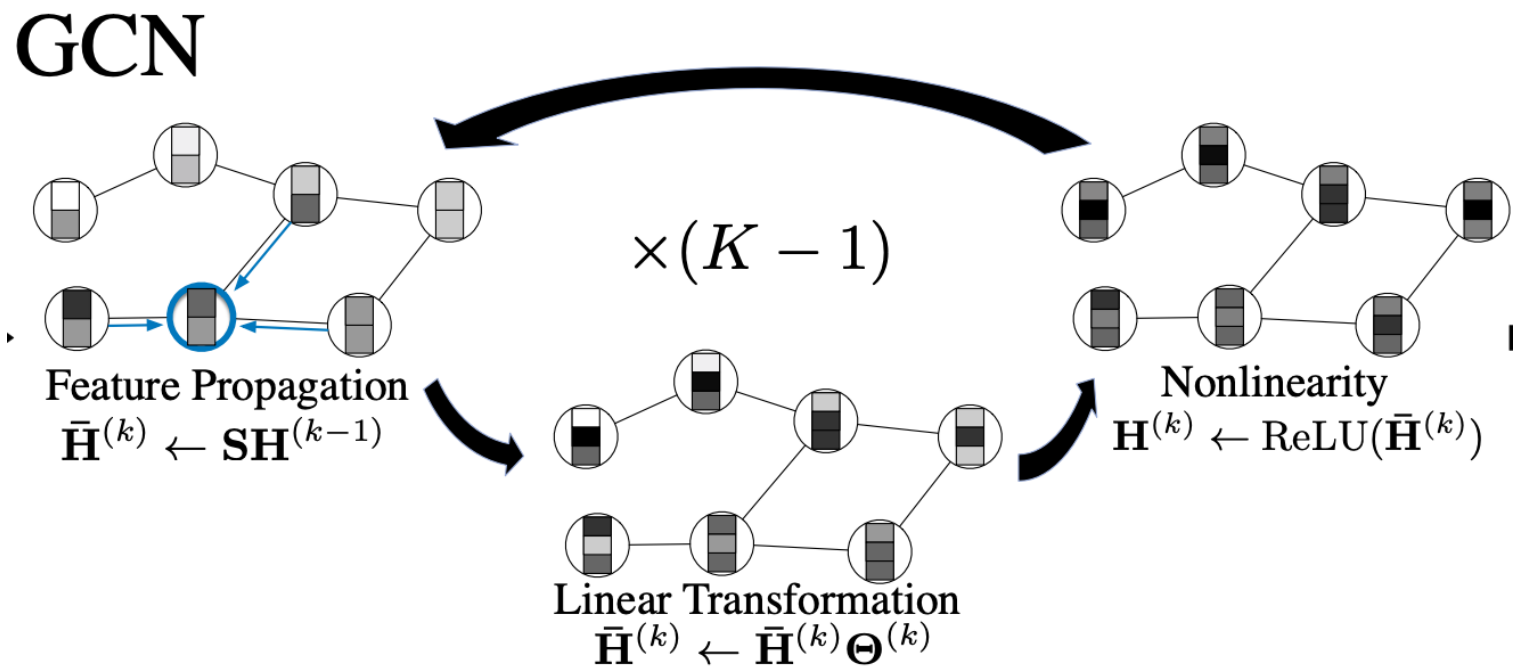
KEG, Tsinghua University

Advisors: Yuxiao Dong, Jie Tang

Course Link: https://cogdl.ai/gnn2022/

CogDL is publicly available at https://github.com/THUDM/cogdl

Background



Question: How to determine K? Do we need deeper GNNs?

Outline

- JKNet (ICML'18)
- PPNP (ICLR'19)
- DropEdge (ICLR'20)
- GCNII (ICML'20)
- DeeperGCN (Arxiv 2020)
- RevGNN (ICML'21)

JKNet (ICML'18)

- Observations:
  - 2-layer GCN performs well
  - GCNs with more layers do not perform well (although have access to more information)

- Questions:
  - Limitation of neighborhood aggregation?

Xu, Keyulu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation learning on graphs with jumping knowledge networks." In ICML'18.

JKNet (ICML'18)

- Analysis: connections between influence distributions and random walk distribution:

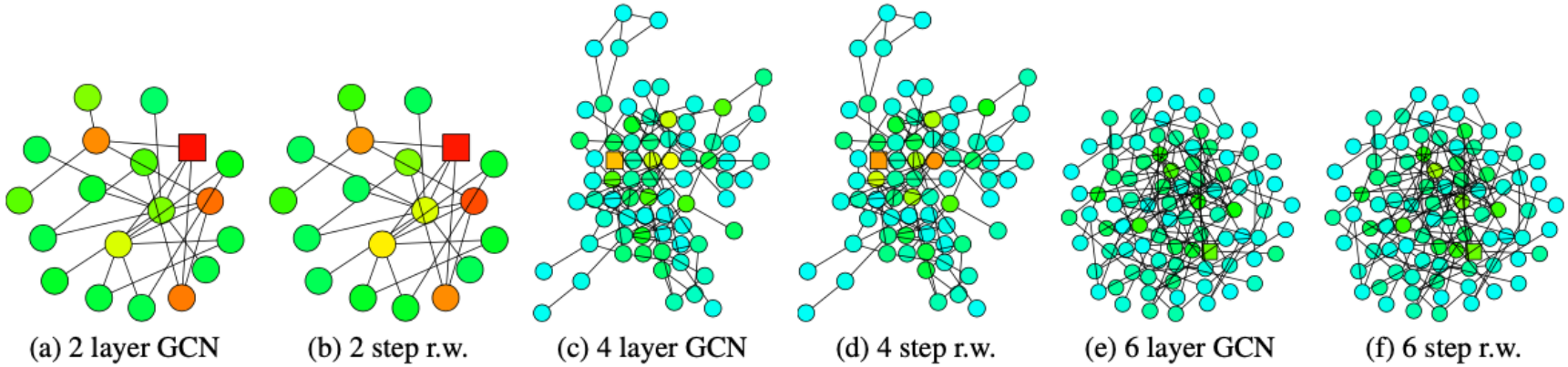


Figure 2. Influence distributions of GCNs and random walk distributions starting at the square node

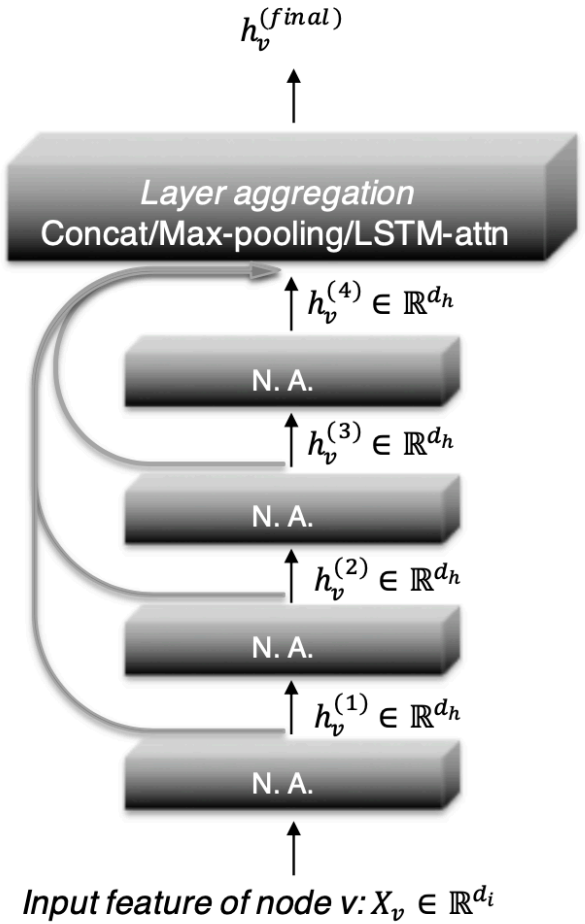
- Hard to determine propagation step!
- Layer aggregation!

Xu, Keyulu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "Representation learning on graphs with jumping knowledge networks." In ICML'18.

JKNet Results

- Layer-aggregation Mechanism
  - Concatenation
  - Max-pooling
  - LSTM-attention

Model	Citeseer	Model	Cora
GCN (2)	77.3 (1.3)	GCN (2)	88.2 (0.7)
GAT (2)	76.2 (0.8)	GAT (3)	87.7 (0.3)
JK-MaxPool (1)	77.7 (0.5)	JK-Maxpool (6)	**89.6** (0.5)
JK-Concat (1)	**78.3** (0.8)	JK-Concat (6)	89.1 (1.1)
JK-LSTM (2)	74.7 (0.9)	JK-LSTM (1)	85.8 (1.0)



PPNP (ICLR'19)

- Inspired by JKNet:
  - connections between influence distribution and random walk distribution

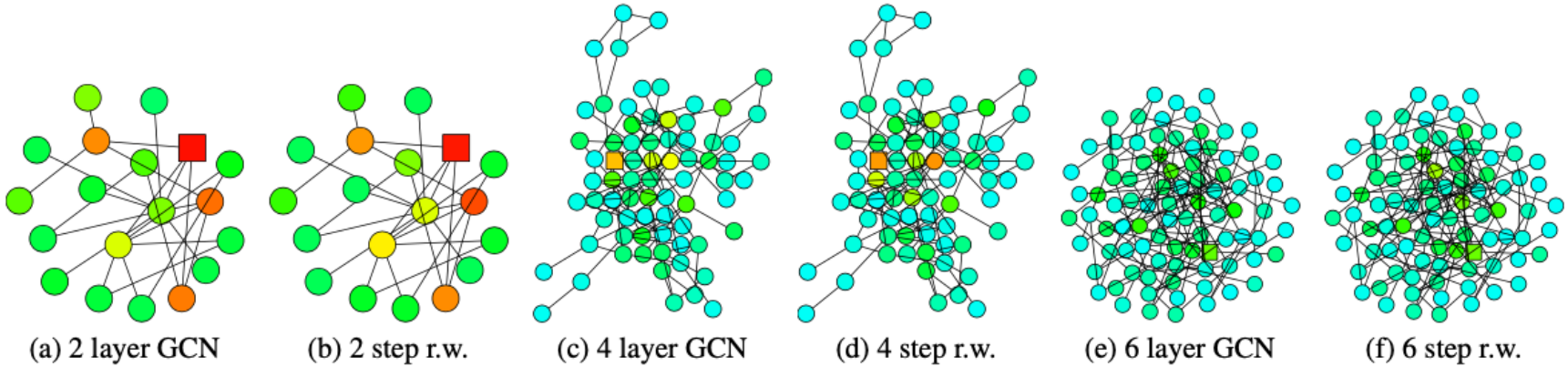


Figure 2. Influence distributions of GCNs and random walk distributions starting at the square node

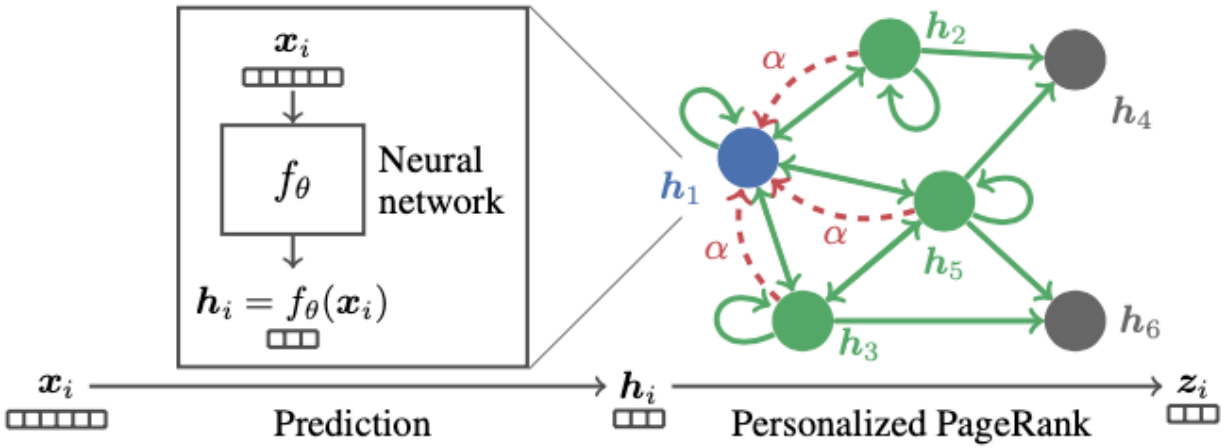
- Decouple the propagation and transformation!

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. In ICLR'19.

PPNP (ICLR'19)

- Personalized PageRank (PPR): $\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x) = (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x)+\alpha\boldsymbol{i}_x$
- By solving the equation, we obtain: $\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x) = \alpha\left(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\right)^{-1}\boldsymbol{i}_x$
- Personalized propagation of neural predictions (PPNP):
  - generate **predictions** based on its own features and then **propagate** them via PPR:

$$\boldsymbol{Z}_{\text{PPNP}} = \text{softmax}\left(\alpha\left(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\right)^{-1}\boldsymbol{H}\right), \qquad \boldsymbol{H}_{i,:} = f_\theta(\boldsymbol{X}_{i,:}),$$



Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Predict then propagate: Graph neural networks meet personalized pagerank. In ICLR'19.

APPNP (ICLR'19)

- PPNP needs $O(n^2)$ to calculate the full PPR matrix:

$$\boldsymbol{\Pi}_{\text{ppr}} = \alpha(\boldsymbol{I}_n - (1-\alpha)\hat{\tilde{\boldsymbol{A}}})^{-1}$$

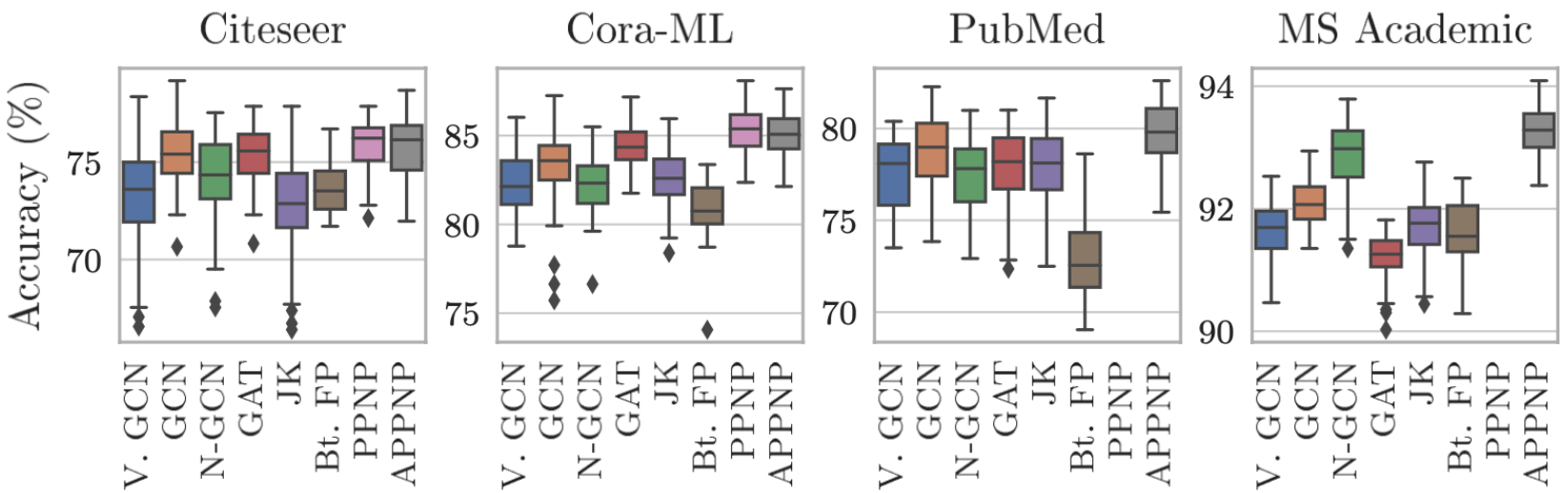
- Approximate PPNP (APPNP):
  - via power iteration step $\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x) = (1-\alpha)\hat{\tilde{\boldsymbol{A}}}\boldsymbol{\pi}_{\text{ppr}}(\boldsymbol{i}_x) + \alpha\boldsymbol{i}_x$

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

(A)PPNP Results

Table 1: Dataset statistics. Shortest path length is denoted by SP.

Dataset	Type	Classes	Features	Nodes	Edges	Label rate	Avg. SP
CITESEER	Citation	6	3703	2110	3668	0.036	9.31
CORA-ML	Citation	7	2879	2810	7981	0.047	5.27
PUBMED	Citation	3	500	19 717	44 324	0.003	6.34
MS ACADEMIC	Co-author	15	6805	18 333	81 894	0.016	5.43



DropEdge (ICLR'20)

- Issues that prevents GNNs from being deeper:
  - over-fitting
  - over-smoothing

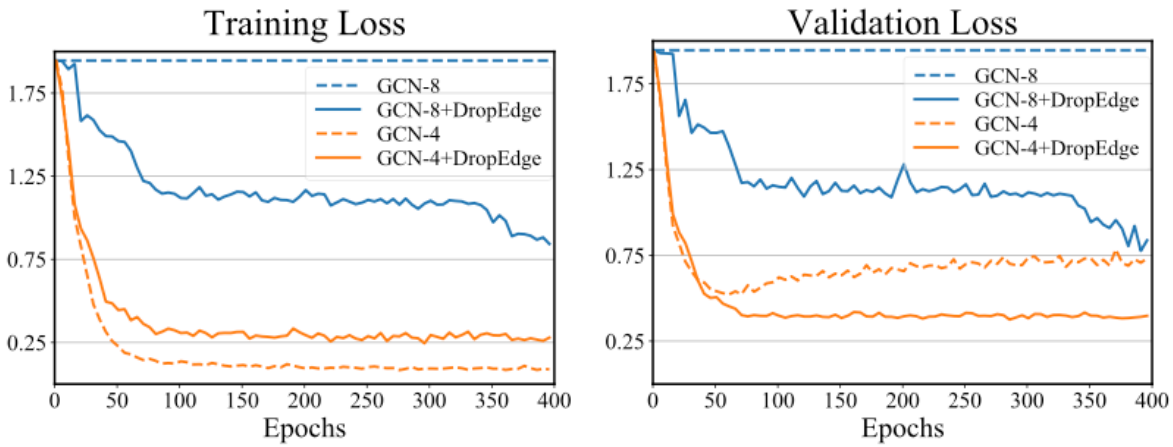


Figure 1: Performance of Multi-layer GCNs on Cora. We implement 4-layer GCN w and w/o DropEdge (in orange), 8-layer GCN w and w/o DropEdge (in blue)[2]. GCN-4 gets stuck in the over-fitting issue attaining low training error but high validation error; the training of GCN-8 fails to converge satisfactorily due to over-smoothing. By applying DropEdge, both GCN-4 and GCN-8 work well for both training and validation.

Rong, Yu, Wenbing Huang, Tingyang Xu, and Junzhou Huang. "Dropedge: Towards deep graph convolutional networks on node classification." arXiv preprint arXiv:1907.10903 (2019).

DropEdge (ICLR'20)

- DropEdge: $\boldsymbol{A}_{\text{drop}} = \boldsymbol{A} - \boldsymbol{A}'$

- Prevent over-fitting:
  - unbiased data augmentation
- Alleviate over-smoothing:
  - Slow down the convergence of over-smoothing
  - Reduce information loss

Rong, Yu, Wenbing Huang, Tingyang Xu, and Junzhou Huang. "Dropedge: Towards deep graph convolutional networks on node classification." arXiv preprint arXiv:1907.10903 (2019).

Discussions of DropEdge

- DropEdge vs Dropout
  - Dropout: no help to prevent over-smoothing
- DropEdge v.s. Node sampling
  - GraphSAGE, FastGCN, ASGCN
- DropEdge v.s. Graph-Sparsification
  - Random vs Fixed

DropEdge Performance

Table 1: Testing accuracy (%) comparisons on different backbones w and w/o DropEdge.

Dataset	Backbone	2 layers		8 layers		32 layers	
		Orignal	DropEdge	Orignal	DropEdge	Orignal	DropEdge
Cora	GCN	86.10	**86.50**	78.70	**85.80**	71.60	**74.60**
	ResGCN	-	-	85.40	**86.90**	85.10	**86.80**
	JKNet	-	-	86.70	**87.80**	87.10	**87.60**
	IncepGCN	-	-	86.70	**88.20**	87.40	**87.70**
	GraphSAGE	87.80	**88.10**	84.30	**87.10**	31.90	**32.20**
Citeseer	GCN	75.90	**78.70**	74.60	**77.20**	59.20	**61.40**
	ResGCN	-	-	77.80	**78.80**	74.40	**77.90**
	JKNet	-	-	79.20	**80.20**	71.70	**80.00**
	IncepGCN	-	-	79.60	**80.50**	72.60	**80.30**
	GraphSAGE	78.40	**80.00**	74.10	**77.10**	37.00	**53.60**
Pubmed	GCN	90.20	**91.20**	90.10	**90.90**	84.60	**86.20**
	ResGCN	-	-	89.60	**90.50**	90.20	**91.10**
	JKNet	-	-	90.60	**91.20**	89.20	**91.30**
	IncepGCN	-	-	90.20	**91.50**	OOM	**90.50**
	GraphSAGE	90.10	**90.70**	90.20	**91.70**	41.30	**47.90**
Reddit	GCN	96.11	**96.13**	96.17	**96.48**	45.55	**50.51**
	ResGCN	-	-	96.37	**96.46**	93.93	**94.27**
	JKNet	-	-	96.82	**97.02**	OOM	OOM
	IncepGCN	-	-	96.43	**96.87**	OOM	OOM
	GraphSAGE	96.22	**96.28**	96.38	**96.42**	96.43	**96.47**

GCNII (ICML'20)

- Previous works:
  - Tackle over-smoothing: JKNet, DropEdge
  - Decoupled models: SGC, (A)PPNP
  - Successful case in CNNs: ResNet

- Question: How to design deep GNNs?

Chen, Ming, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. "Simple and deep graph convolutional networks." In ICML'20.

GCNII (ICML'20)

- Initial residual connection:
  - Similar approach to APPNP (but APPNP remains shallow)
  - Combine the smoothed representations with initial features

$$(1-\alpha)\tilde{\boldsymbol{P}}\boldsymbol{H}^{(\ell)}+\alpha\boldsymbol{H}^{(0)}$$

- Identity mapping:
  - Similar to the motivation of ResNet
  - Add an identity matrix to the weight matrix

$$(1-\beta_\ell)\mathbf{I}_n+\beta_\ell\mathbf{W}^{(\ell)}$$

- GCNII (Combine the two techniques)

$$\mathbf{H}^{(\ell+1)}=\sigma\left(\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}+\alpha_\ell\mathbf{H}^{(0)}\right)\left((1-\beta_\ell)\mathbf{I}_n+\beta_\ell\mathbf{W}^{(\ell)}\right)\right)$$

Chen, Ming, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. "Simple and deep graph convolutional networks." In ICML'20.

GCNII Results

- GCNII*: employ different weights for PH and $H^{(0)}$:

$$\mathbf{H}^{(\ell+1)} = \sigma\left((1-\alpha_\ell)\tilde{\mathbf{P}}\mathbf{H}^{(\ell)}\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}_1^{(\ell)}\right) + \right.$$
$$\left. +\alpha_\ell \mathbf{H}^{(0)}\left((1-\beta_\ell)\mathbf{I}_n + \beta_\ell \mathbf{W}_2^{(\ell)}\right)\right).$$

Method	Cora	Cite.	Pumb.	Cham.	Corn.	Texa.	Wisc.
GCN	85.77	73.68	88.13	28.18	52.70	52.16	45.88
GAT	86.37	74.32	87.62	42.93	54.32	58.38	49.41
Geom-GCN-I	85.19	**77.99**	90.05	60.31	56.76	57.58	58.24
Geom-GCN-P	84.93	75.14	88.09	60.90	60.81	67.57	64.12
Geom-GCN-S	85.27	74.71	84.75	59.96	55.68	59.73	56.67
APPNP	87.87	76.53	89.40	54.3	73.51	65.41	69.02
JKNet	85.25 (16)	75.85 (8)	88.94 (64)	60.07 (32)	57.30 (4)	56.49 (32)	48.82 (8)
JKNet(Drop)	87.46 (16)	75.96 (8)	89.45 (64)	62.08 (32)	61.08 (4)	57.30 (32)	50.59 (8)
Incep(Drop)	86.86 (8)	76.83 (8)	89.18 (4)	61.71 (8)	61.62 (16)	57.84 (8)	50.20 (8)
GCNII	**88.49** (64)	77.08 (64)	89.57 (64)	60.61 (8)	74.86 (16)	69.46 (32)	74.12 (16)
GCNII*	88.01 (64)	77.13 (64)	**90.30** (64)	**62.48** (8)	**76.49** (16)	**77.84** (32)	**81.57** (16)

DeeperGCN

- Generalized Aggregation Function (Mean-Max)
  - Find a better aggregator than *mean* and *max*
- SoftMax_Agg: $\sum_{u \in \mathcal{N}(v)} \frac{exp(\beta \mathbf{m}_{vu})}{\sum_{i \in \mathcal{N}(v)} exp(\beta \mathbf{m}_{vi})}$
  - $\lim_{\beta \to 0} \text{SoftMax\_Agg}_\beta = \text{Mean}$
  - $\lim_{\beta \to \infty} \text{SoftMax\_Agg}_\beta = \text{Max}$
- PowerMean: $\left(\frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{m}_{vu}^p\right)^{1/p}$
  - $\text{PowerMean\_Agg}_{p=1} = \text{Mean}$
  - $\lim_{p \to \infty} \text{PowerMean\_Agg}_p = \text{Max}$

Different GNN architectures

- ResGCN:
  - GraphGonv → Normalization → ReLU → Addition
- ResGCN+:
  - Normalization → ReLU → GraphGonv → Addition
- ResGEN:
  - ResGCN + generalized message aggregators
- DyResGEN:
  - Dynamicly learn the parameters of generalized message aggregators at every gradient descent step

DeeperGCN Results

Table 1: Ablation study on residual connections, model depth and aggregators on ogbn-proteins.

	PlainGCN			ResGCN			ResGCN+		
#layers	Sum	Mean	Max	Sum	Mean	Max	Sum	Mean	Max
3	0.824	0.793	**0.834**	0.824	0.786	0.824	0.830	0.792	0.829
7	0.811	0.796	0.823	0.831	0.803	0.843	0.841	0.813	0.845
14	0.821	0.802	0.824	0.843	0.808	0.850	0.840	0.813	0.848
28	0.819	0.794	0.825	0.837	0.807	0.847	0.845	0.819	0.855
56	0.824	0.808	0.825	0.841	0.813	**0.851**	0.843	0.810	0.853
112	0.823	0.810	0.824	0.840	0.805	**0.851**	0.853	0.820	**0.858**
avg.	0.820	0.801	0.826	0.836	0.804	0.844	0.842	0.811	**0.848**

Table 4: Ablation study on DyResGEN on ogbn-proteins. β, p and s denote the learned parameters.

	SoftMax_Agg			PowerMean_Agg		
#layers	Fixed	β	$\beta\&s$	Fixed	p	$p\&s$
3	0.821	0.832	0.837	0.802	0.818	0.838
7	0.835	0.846	0.848	0.797	0.841	0.851
14	0.833	0.849	0.851	0.814	0.840	0.849
28	0.845	0.852	0.853	0.816	0.847	**0.854**
56	0.849	**0.860**	0.854	0.818	0.846	-
112	0.844	0.858	0.858	0.824	-	-
avg.	0.838	**0.850**	**0.850**	0.812	0.838	**0.848**

Deep GNNs (From 112 to 1000 layers)

- DeeperGCN: All You Need to Train Deeper GCNs
  - Li et al., June 2020
  - Generalized Message Aggregation
  - Pre-activation residual connections
  - Up to 112 layers (GPU memory bounded)

- Training Graph Neural Networks with 1000 Layers
  - Li et al., ICML 2021
  - Reversible connections
  - Up to 1000 layers

Li, Guohao, et al. "Deepergcn: All you need to train deeper gcns." *arXiv preprint arXiv:2006.07739* (2020).
Li, Guohao, et al. "Training Graph Neural Networks with 1000 Layers." arXiv preprint arXiv:2106.07476 (2021).

Recall Backpropagation in GNNs

- Graph convolution：

$$\boldsymbol{H}^{(l+1)} = \boldsymbol{A}\boldsymbol{H}^{(l)}\boldsymbol{W}_l$$

- Backward of graph convolution：

$$\nabla_{\boldsymbol{H}^{(l)}} = \boldsymbol{A}^T \nabla_{\boldsymbol{H}^{(l+1)}} \boldsymbol{W}_l^T$$

$$\nabla_{\boldsymbol{W}_l} = \boldsymbol{H}^{(l)^T} \boldsymbol{A}^T \nabla_{\boldsymbol{H}^{(l+1)}}$$

$$\nabla_{\boldsymbol{A}} = \nabla_{\boldsymbol{H}^{(l+1)}} \boldsymbol{W}_l^T \boldsymbol{H}^{(l)^T}$$

- We need to save $\boldsymbol{H}^{(l)}$ for each layer, which costs O(ND) memory per layer。

GNNs with 1000 Layers (ICML'21)

- Challenges: O(LND) memory, linear to the number layers
- **Reversible connections!**
- (inspired by NeurIPS 2017: The reversible residual network: Backpropagation without storing activations)
- Grouped Reversible GNN block:

$$\langle X_1, X_2, ..., X_C \rangle \mapsto \langle X_1', X_2', ..., X_C' \rangle$$

Forward (from X_i to X_i')

$$X_0' = \sum_{i=2}^{C} X_i$$

$$X_i' = f_{w_i}(X_{i-1}', A, U) + X_i,\ i \in \{1, \cdots, C\}$$

Backward (from X_i' to X_i)

$$X_i = X_i' - f_{w_i}(X_{i-1}', A, U),\ i \in \{2, \cdots, C\}$$

$$X_0' = \sum_{i=2}^{C} X_i$$

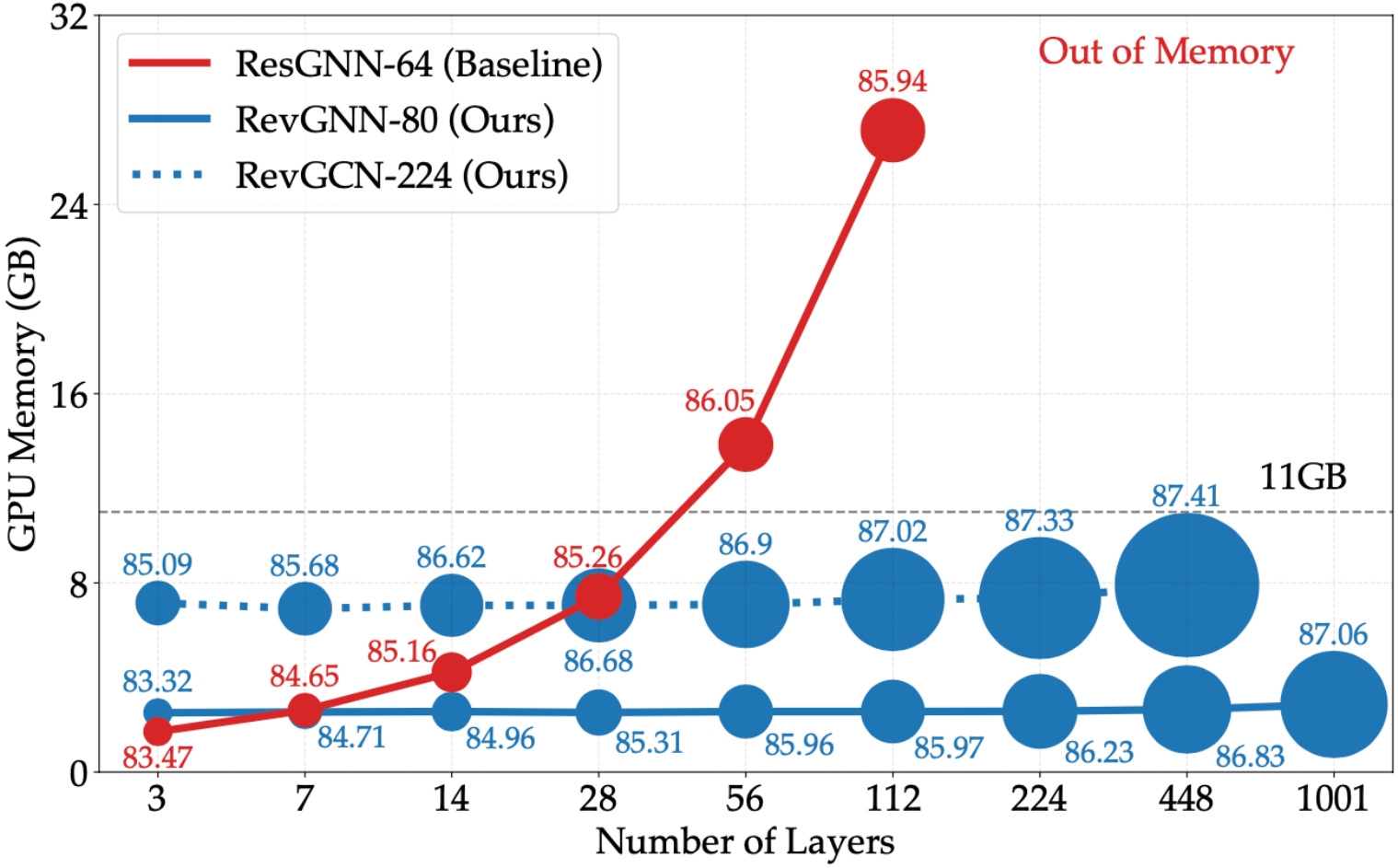
$$X_1 = X_1' - f_{w_1}(X_0', A, U).$$

RevGNN on ogbn-proteins

- ogbn-proteins dataset：
  - Node: proteins
  - Edge: biologically meaningful associations (e.g., homology)

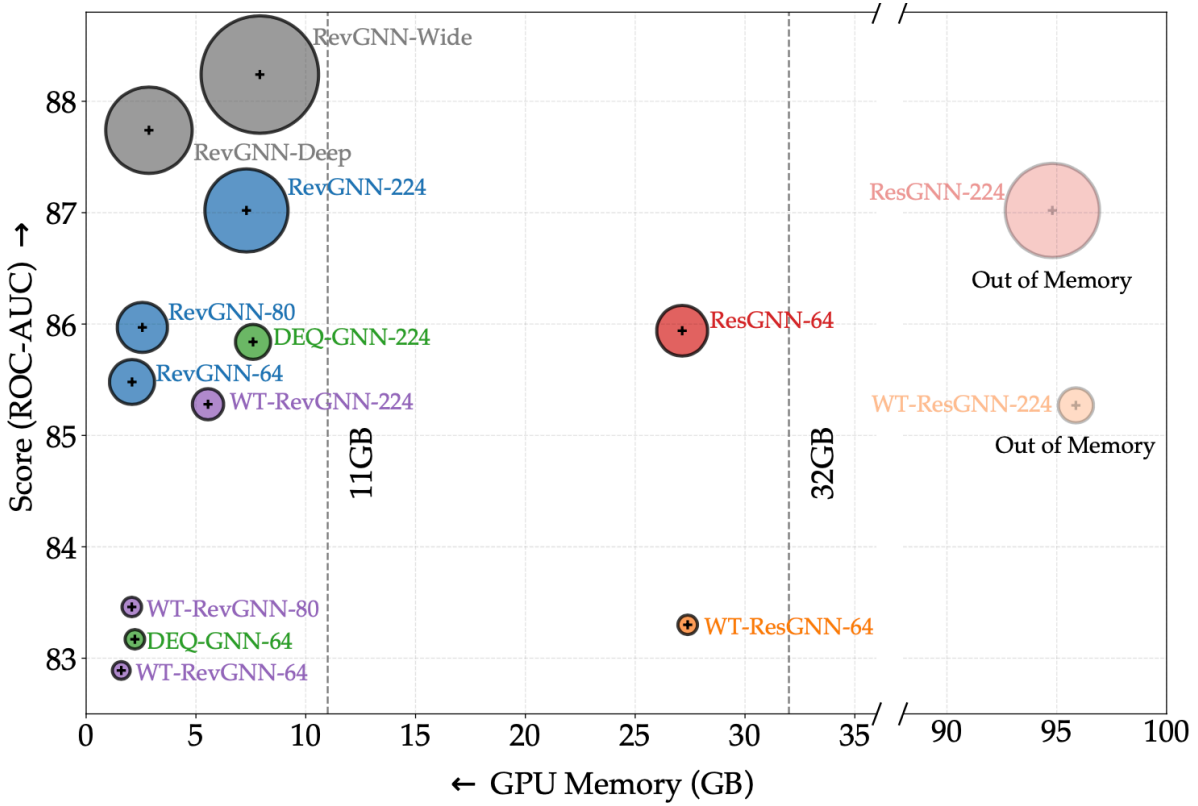
Model	ROC-AUC ↑	Mem ↓	Params
GCN (Kipf & Welling)	72.51 ± 0.35	4.68	96.9k
GraphSAGE (Hamilton et al.)	77.68 ± 0.20	3.12	193k
DeeperGCN (Li et al.)	86.16 ± 0.16	27.1	2.37M
UniMP (Shi et al.)	86.42 ± 0.08	27.2	1.91M
GAT (Veličković et al.)	86.82 ± 0.21	6.74	2.48M
UniMP+CEF (Shi et al.)	86.91 ± 0.18	27.2	1.96M
Ours (RevGNN-Deep)	87.74 ± 0.13	**2.86**	20.03M
Ours (RevGNN-Wide)	**88.24** ± 0.15	7.91	68.47M

RevGNN v.s. ResGNN



RevGNN v.s. all variants

- **RevGNN-Wide**
  - 448 layers+224 hidden
- **RevGNN-Deep**
  - 1001 layers+80 hidden
- Compared with RevGNN/ResGNN/WT/DEQ-x (x: hidden)
- Datapoint size is proportional to \sqrt(#parameters)



Summary

- JKNet (ICML'18): layer aggregation
- PPNP (ICLR'19): propagation of predictions
- DropEdge (ICLR'20): randomly drop edges

- GCNII (ICML'20): 64-layer GNNs
- DeeperGCN (Arxiv 2020): 112-layer GNNs
- RevGNN (ICML'21): 1000-layer GNNs

Homework 8: Deep GNNs

- Comment on deep GNNs:
  - Due by 11th Sept.
  - **No coding**
  - Write your comments (Reading other papers if possible)
  - Post them to https://discuss.cogdl.ai/t/topic/104
  - Discuss with others
  - Send your comments and discussions (via screenshots) to our email

- Final DDL of HWs: 20th Sept.

Course Project

- DDL(strict): 20th Sept.

- Submission details:
  - A technique report (background, method, result, …)
  - A poster for communication
  - Online demo (https://app.cogdl.ai)
  - With the help of TAs

Thank you !

Collaborators:

Zhenyu Hou, Yuxiao Dong, Jie Tang, et al. (**THU**)

Qingfei Zhao, Xinije Zhang, Peng Zhang, et al. (**Zhipu AI**)

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

Yang Yang (**ZJU**)

Yukuo Cen, KEG, Tsinghua U.

https://github.com/THUDM/cogdl

Online Discussion Forum

https://discuss.cogdl.ai/