Guest Lecture

Graph Neural Networks for Representation Learning on Graphs

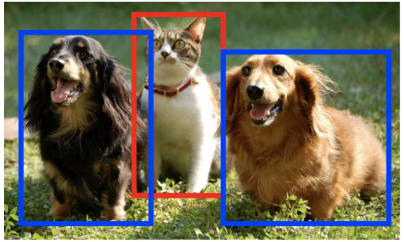
Guohao Li
CS PhD Student @ KAUST
guohao.li@kaust.edu.sa

KAUST IVUL

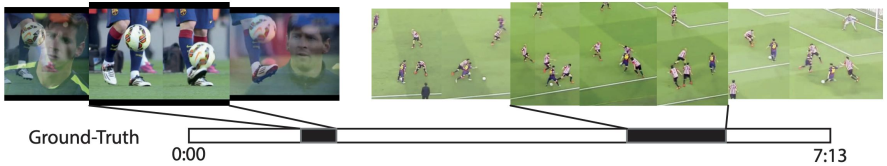
Why do we need graph neural networks?

Grid Data vs. General Graphs

DeepGCNs.org



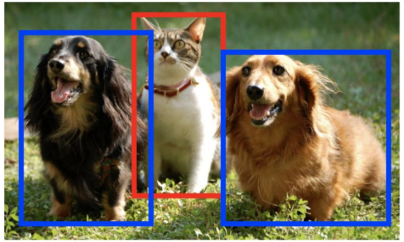
CAT, DOG



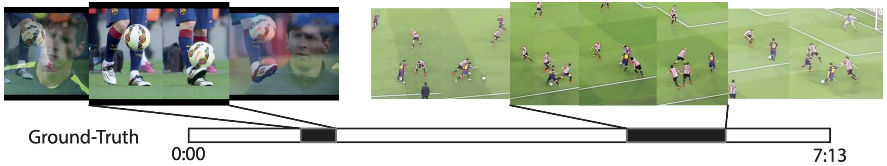
Grid Data:

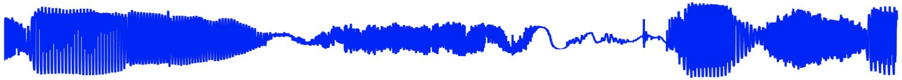
- Image
- Video

Grid Data vs. General Graphs



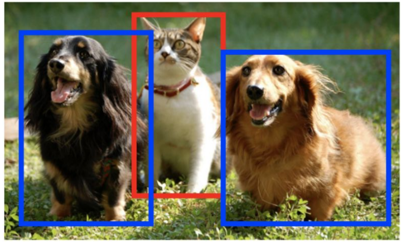
CAT, DOG



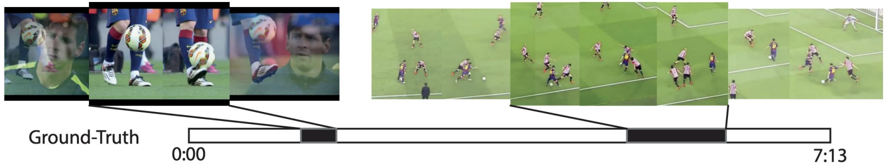


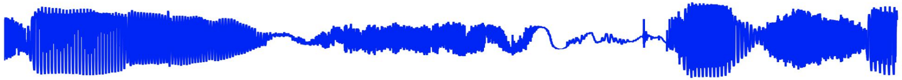
Grid Data:
- Image
- Video
- Audio
- Text

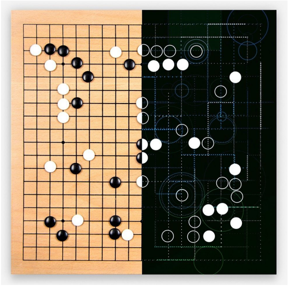
Grid Data vs. General Graphs



CAT, DOG





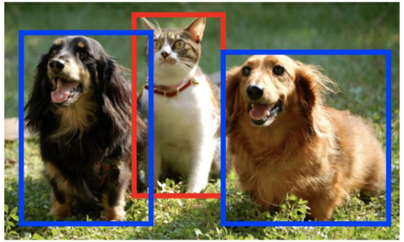


Grid Data :

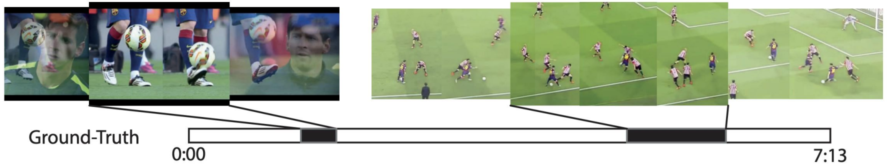
- Image
- Video
- Audio
- Text
- Grid Game (Go)
- ...

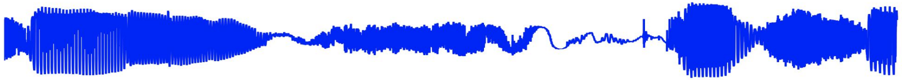
Grid Data vs. General Graphs

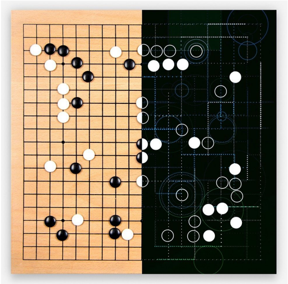
DeepGCNs.org



CAT, DOG







Grid Data :

- Image
- Video
- Audio
- Text
- Grid Game (Go)
- ...

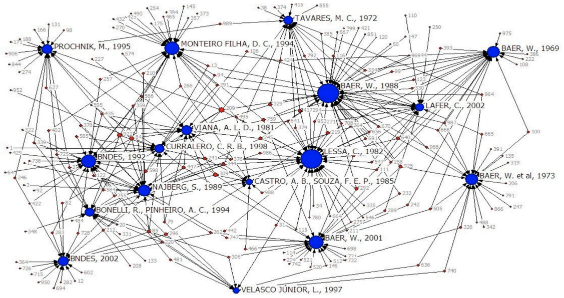
CNN works well

Grid Data vs. General Graphs

How about non-grid graph structured data?

Grid Data vs. General Graphs

Lots of real-world applications need to deal with Non-Grid data



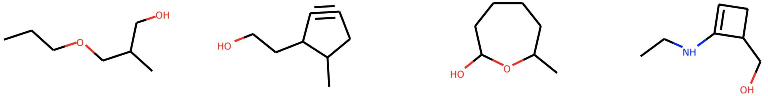
General Graphs:

- Social Networks
- Citation Networks

Grid Data vs. General Graphs

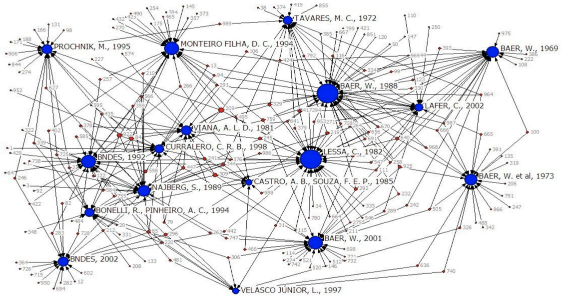
DeepGCNs.org

Lots of real-world applications need to deal with Non-Grid data



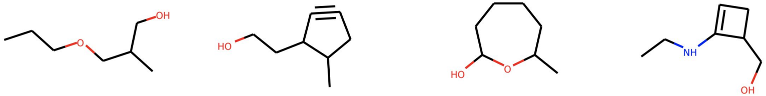
General Graphs :

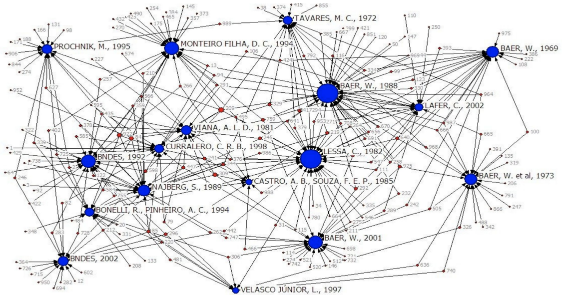
- Social Networks
- Citation Networks
- Molecules

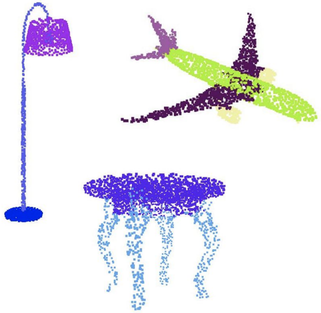


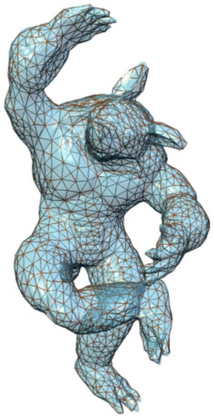
Grid Data vs. General Graphs

Lots of real-world applications need to deal with Non-Grid data







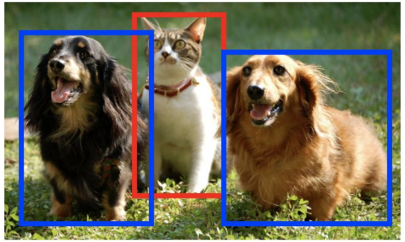


General Graphs:

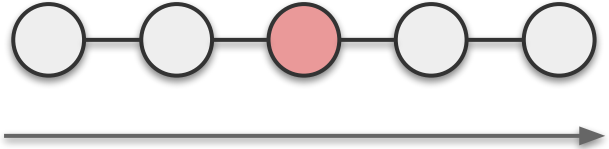
- Social Networks
- Citation Networks
- Molecules
- Point Clouds
- 3D Meshes
- ...

Grid Data vs. General Graphs

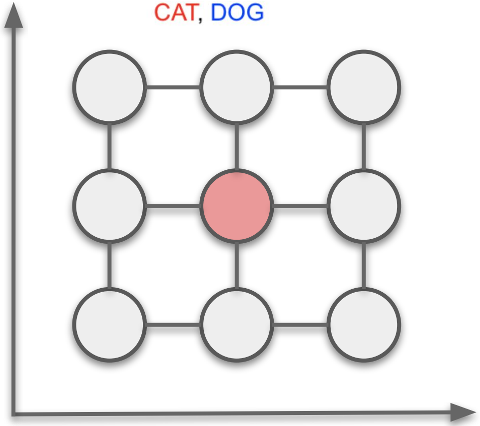
DeepGCNs.org



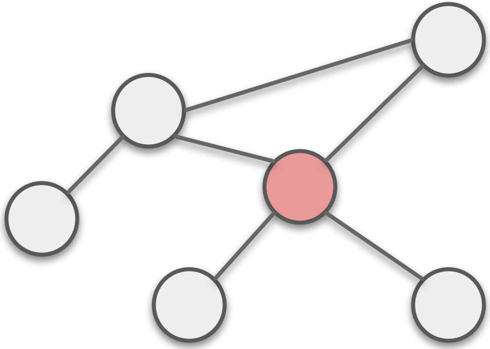
CAT, DOG



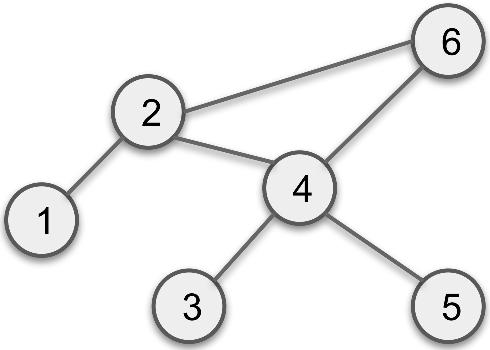
1D Grid



2D Grid



Graph



Undirected Graph

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

$\mathcal{V}$: Vertices

$\mathcal{E}$: Edges

Adjacency Matrix

$$\begin{array}{c c} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Adjacency List

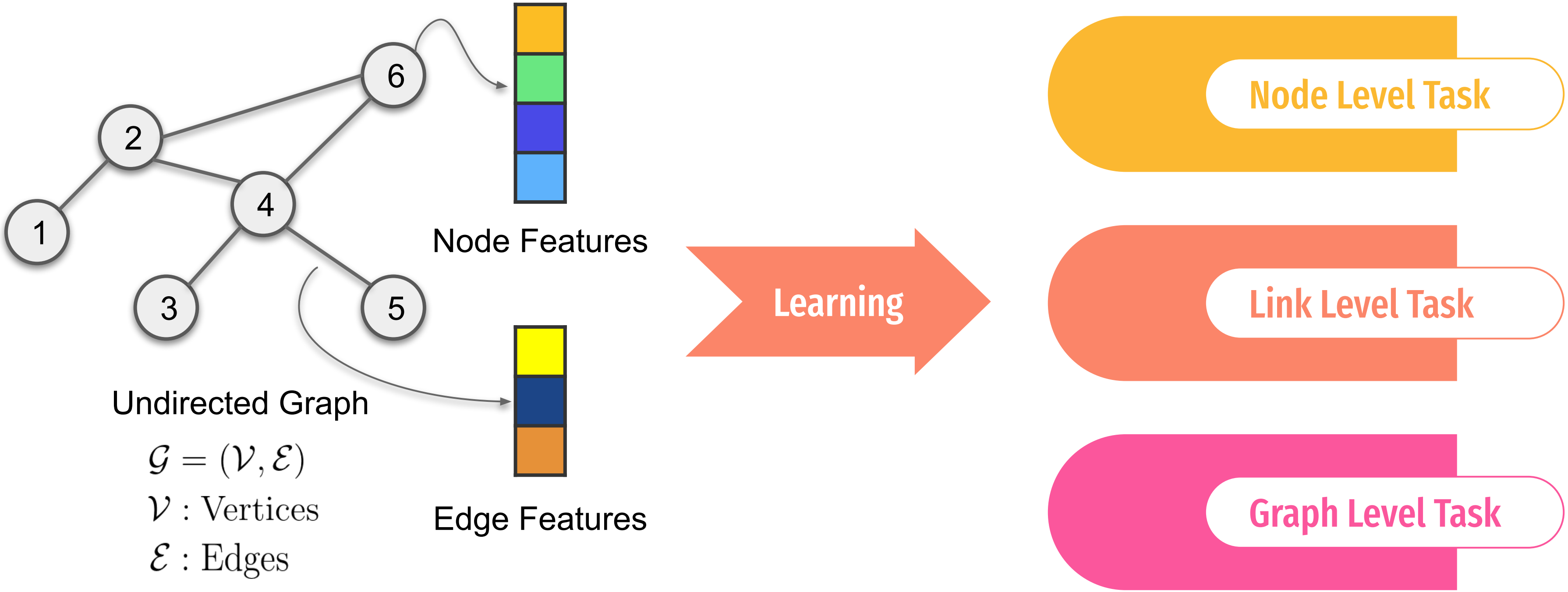
1	→	2			
2	→	1	4	6	
3	→	4			
4	→	2	3	5	6
5	→	4			
6	→	2	4		

Edge List

1	→	2
2	→	1
2	→	4
2	→	6
3	→	4
4	→	2
4	→	3
4	→	5
4	→	6
5	→	4
6	→	2
6	→	4

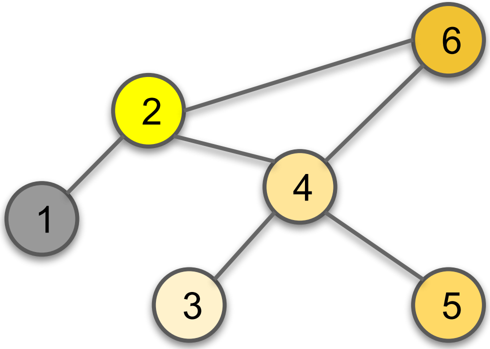
Other types of graphs: directed graphs, weighted graphs, heterogeneous

Graph Representation Learning

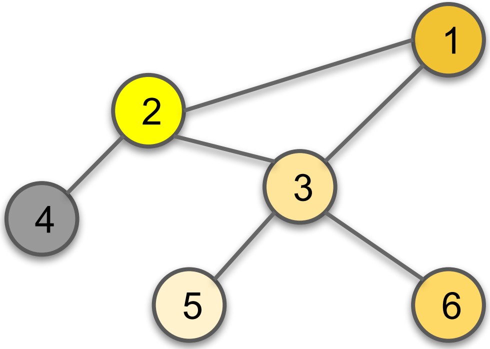


Why we can not use CNN for Graphs

- There is not a canonical order for nodes
- Nodes do not have the same number of neighbors



$$\begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$



$$\begin{array}{cc} & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

They are representing the same graph with different orders (permutations)

Graph Representation Learning

DeepGCNs.org

Why we can not use CNN for Graphs

- There is not a canonical order for nodes
- Nodes do not have the same number of neighbors

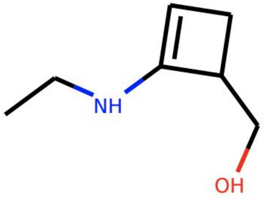
$$
\begin{array}{cc} & \begin{matrix}1&2&3&4&5&6\end{matrix} \\ \begin{matrix}1\\2\\3\\4\\5\\6\end{matrix} & \begin{bmatrix}0&0&0&0&0&1\\0&1&0&0&0&0\\0&0&0&1&0&0\\1&0&0&0&0&0\\0&0&1&0&0&0\\0&0&0&0&1&0\end{bmatrix}\end{array}
\cdot
\begin{array}{cc} & \begin{matrix}1&2&3&4&5&6\end{matrix} \\ \begin{matrix}1\\2\\3\\4\\5\\6\end{matrix} & \begin{bmatrix}0&1&0&0&0&0\\1&0&0&1&0&1\\0&0&0&1&0&0\\0&1&1&0&1&1\\0&0&0&1&0&0\\0&1&0&1&0&0\end{bmatrix}\end{array}
\cdot
\begin{array}{cc} & \begin{matrix}1&2&3&4&5&6\end{matrix} \\ \begin{matrix}1\\2\\3\\4\\5\\6\end{matrix} & \begin{bmatrix}0&0&0&1&0&0\\0&1&0&0&0&0\\0&0&0&0&1&0\\0&0&1&0&0&0\\0&0&0&0&0&1\\1&0&0&0&0&0\end{bmatrix}\end{array}
=
\begin{array}{cc} & \begin{matrix}1&2&3&4&5&6\end{matrix} \\ \begin{matrix}1\\2\\3\\4\\5\\6\end{matrix} & \begin{bmatrix}0&1&1&0&0&0\\1&0&1&1&0&0\\1&1&0&0&1&1\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&1&0&0&0\end{bmatrix}\end{array}
$$

$P \qquad A \qquad P^T \qquad perm(A)$

They are representing the same graph with different orders (permutations)

Graph Representation Learning

DeepGCNs.org

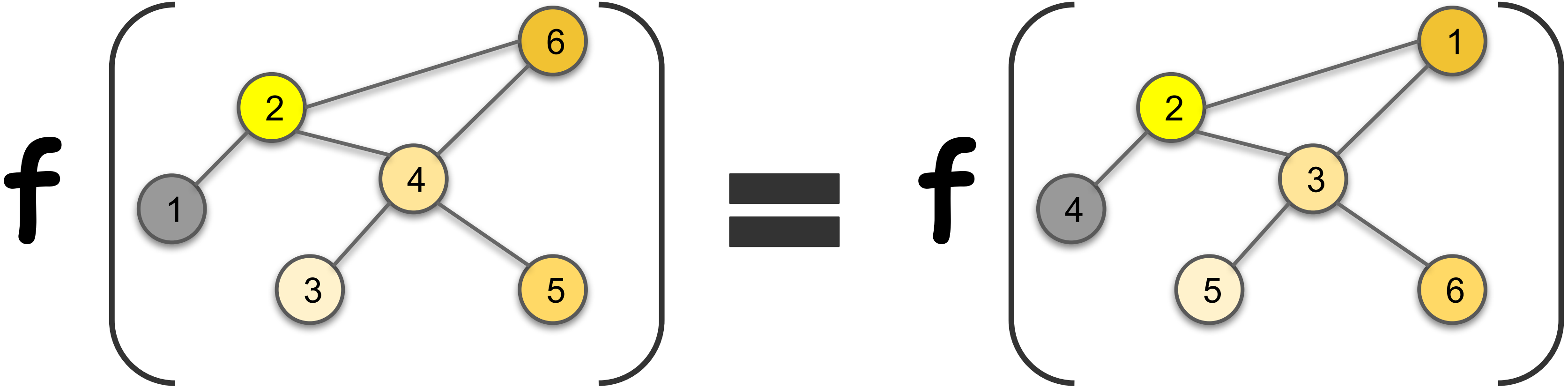


Toxic or Not Toxic

Need to guarantee permutation invariance

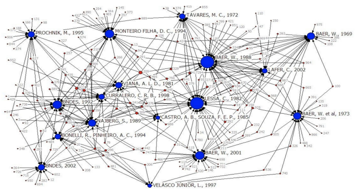
Graph Level Prediction

$$f(X, A) = f(PX, PAP^T)$$



Graph Representation Learning

DeepGCNs.org

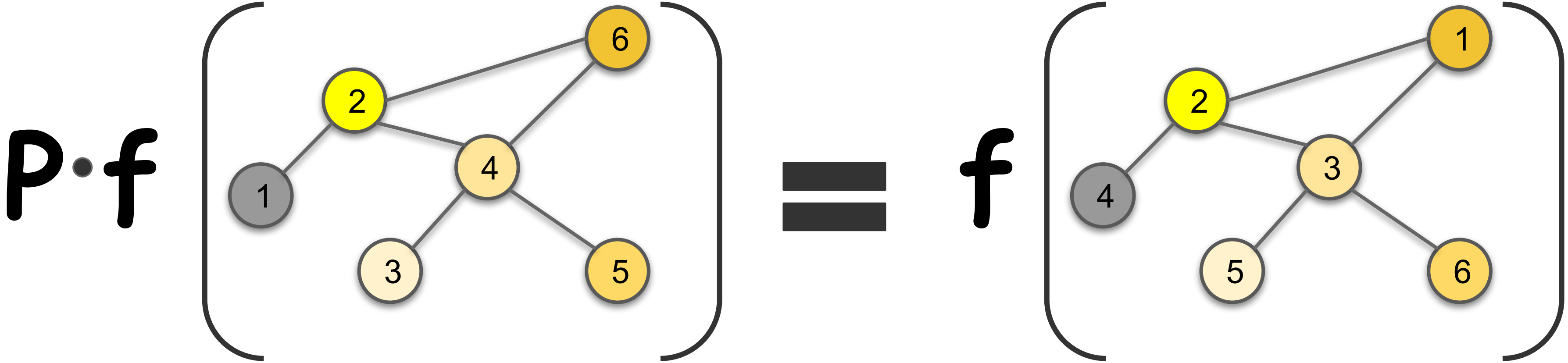


Subject of Papers

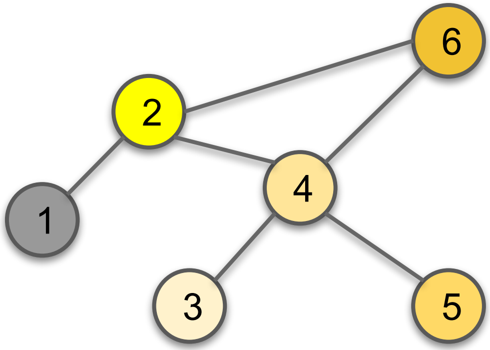
Need to guarantee permutation equivariance

Node Level Prediction

$$P \cdot f(X, A) = f(PX, PAP^T)$$



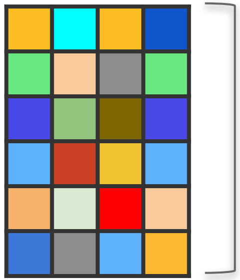
Graph Representation Learning

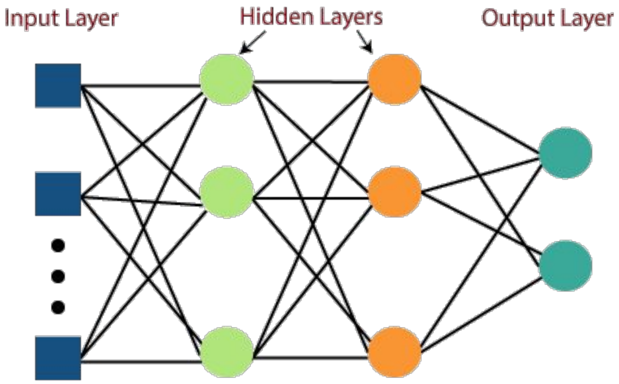


Adjacency Matrix

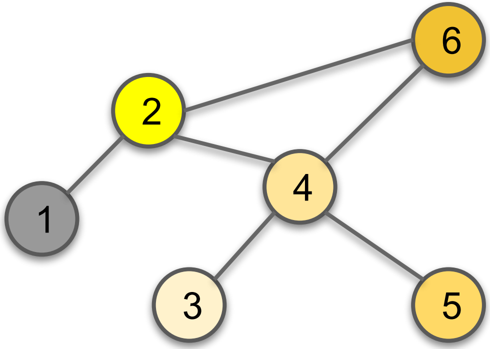
$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0 & 1 & 0 & 0 & 0 & 0 \\
2 & 1 & 0 & 0 & 1 & 0 & 1 \\
3 & 0 & 0 & 0 & 1 & 0 & 0 \\
4 & 0 & 1 & 1 & 0 & 1 & 1 \\
5 & 0 & 0 & 0 & 1 & 0 & 0 \\
6 & 0 & 1 & 0 & 1 & 0 & 0
\end{array}
$$

Node Features

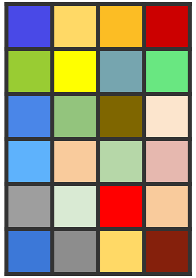




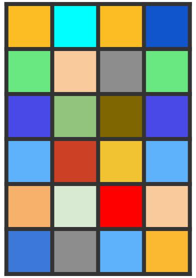
Multilayer Perceptron is not permutation equivariance



Updated Node Features | Adjacency Matrix | Node Features | Weights



$$= \begin{array}{c|cccccc} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 1 & 0 & 1 \\ 3 & 0 & 0 & 0 & 1 & 0 & 0 \\ 4 & 0 & 1 & 1 & 0 & 1 & 1 \\ 5 & 0 & 0 & 0 & 1 & 0 & 0 \\ 6 & 0 & 1 & 0 & 1 & 0 & 0 \end{array} \cdot$$



$$\cdot \ \mathbf{W}$$

N*D' | N*N | N*D | D*D'

How to guarantee permutation equivariance?

$$P \cdot f(X, A) = f(PX, PAP^T)$$

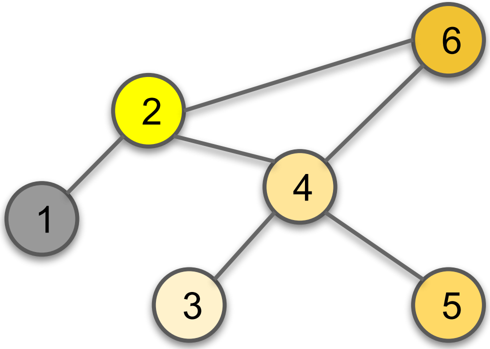
$$f(X, A) = AXW$$ (Activation function is omitted)

$$f(PX, PAP^T) = PAP^T PXW$$
$$= PAXW = P \cdot f(X, A)$$

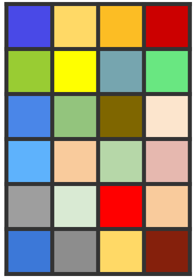
Permutation matrices are orthogonal matrices, i.e. $P^T P = I$

Graph Convolutional Networks

DeepGCNs.org



Updated Node Features



N*D'

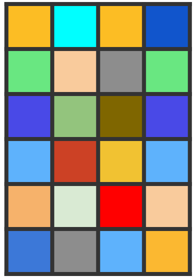
$$=$$

Normalized Adjacency Matrix

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

N*N

Node Features

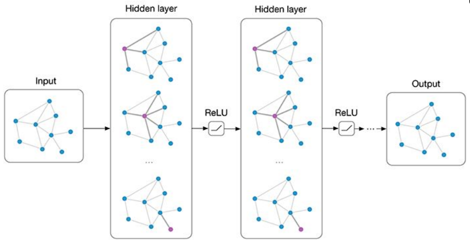


N*D

Weights

$$\cdot \; W$$

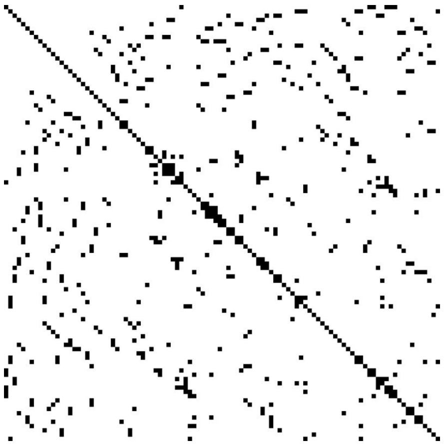
D*D'



Kipf, T.N. and Welling, M., 2016. Semi-Supervised Classification with Graph Convolutional Networks.

- Activation function is used
- Self loop is added to the to the adjacency matrix
- Degree normalization is applied

Neural Message Passing



Adjacency matrix is very sparse which is not memory efficient

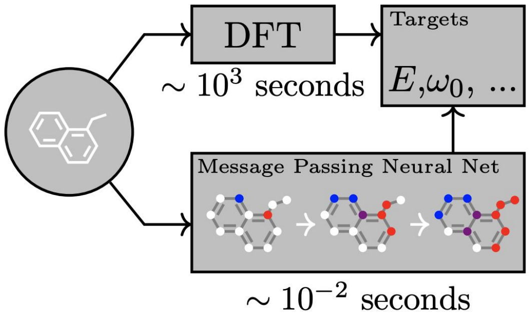
PyG

DGL DEEP GRAPH LIBRARY

Most of the current GNN libraries implement GNNs using message passing

Neural Message Passing for Quantum Chemistry

Justin Gilmer [1] Samuel S. Schoenholz [1] Patrick F. Riley [2] Oriol Vinyals [3] George E. Dahl [1]



Neural Message Passing

Node Features

Neighbor's Features

Edge Features

$$\mathbf{x}_i^{(k)} = \gamma^{(k)}\left(\mathbf{x}_i^{(k-1)}, \square_{j\in\mathcal{N}(i)}\,\phi^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{i,j}\right)\right),$$

$\mathcal{N}_i$: neighbor indices

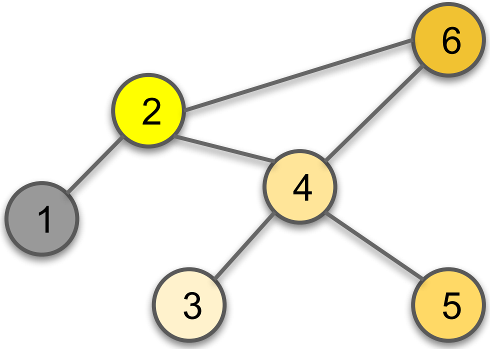
Differentiable (±Learnable) Function
e.g., MLPs
Vertex Update

Permutation Invariant Function
e.g., sum, mean or max
Message Aggregation

Differentiable (±Learnable) Function
e.g., MLPs
Message Construction

https://pytorch-geometric.readthedocs.io

Neural Message Passing

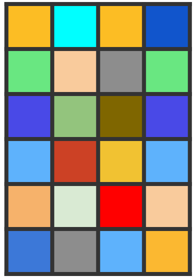


Adjacency Matrix

$$
\begin{array}{c|c}
 & \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} &
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0
\end{bmatrix}
\end{array}
$$

N*N

Node Features



N*D

$$f(X, A) = AXW$$

$$\mathbf{x}_i^{(k)} = \gamma^{(k)}\left(\mathbf{x}_i^{(k-1)}, \square_{j\in\mathcal{N}(i)}\, \phi^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{i,j}\right)\right),$$

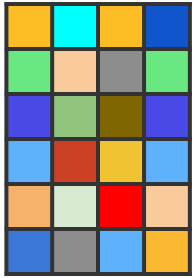
$\mathcal{N}_i$: neighbor indices

Edge List

1	→	2
2	→	1
2	→	4
2	→	6
3	→	4
4	→	2
4	→	3
4	→	5
4	→	6
5	→	4
6	→	2
6	→	4

2*|E|

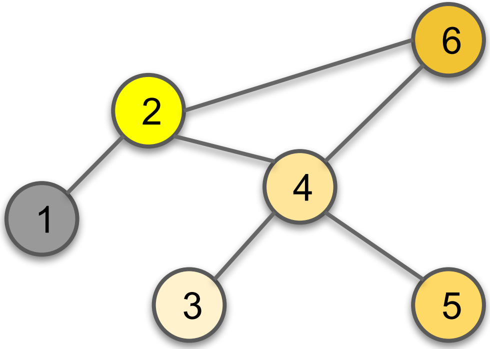
Node Features



N*D

Neural Message Passing

DeepGCNs.org



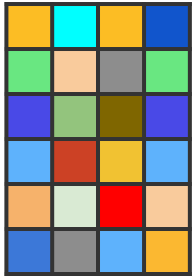
- Easy to handle edge features
- Easy to implement complex message construction functions such as attention
- Friendly to complex aggregation function such as max, softmax, powermean

Edge List

1	→	2
2	→	1
2	→	4
2	→	6
3	→	4
4	→	2
4	→	3
4	→	5
4	→	6
5	→	4
6	→	2
6	→	4

2*|E|

Node Features



N*D

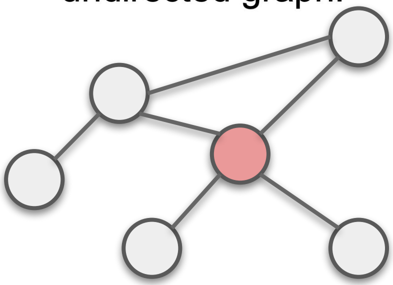
$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)} \, \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{i,j} \right) \right),$$

$\mathcal{N}_i$: neighbor indices

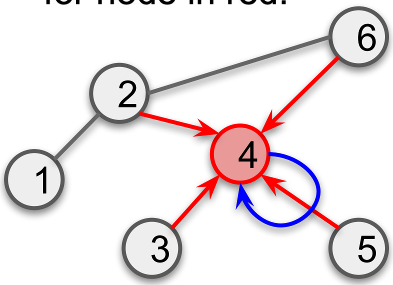
GNN vs. CNN

DeepGCNs.org

Consider this undirected graph:



Calculate update for node in red:

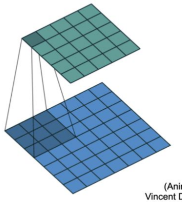


Update rule: $$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

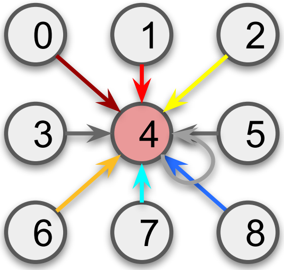
i=4, j∈{2,3,5,6}

Graph Neural Network (GNN)

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



Full update:

$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

Convolutional Neural Network (CNN)

Building Very Deep Graph Neural Networks for Representation Learning on Graphs

DeepGCNs: Can GCNs Go as Deep as CNNs?

Authors	Guohao Li, Matthias Muller, Ali Thabet, Bernard Ghanem
Publication date	2019
Conference	Proceedings of the IEEE International Conference on Computer Vision (ICCV)
Pages	9267-9276

ICCV 2019 Seoul, Korea

DeepGCNs: Making GCNs Go as Deep as CNNs

Authors	Guohao Li, Matthias Müller, Guocheng Qian, Itzel C Delgadillo, Abdulellah Abualshour, Ali Thabet, Bernard Ghanem
Publication date	2021
Journal	IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE

DeeperGCN: Training Deeper GCNs with Generalized Aggregation Functions

Authors	Guohao Li, Chenxin Xiong, Ali Thabet, Bernard Ghanem
Publication date	2020/6/13
Journal	arXiv preprint arXiv:2006.07739

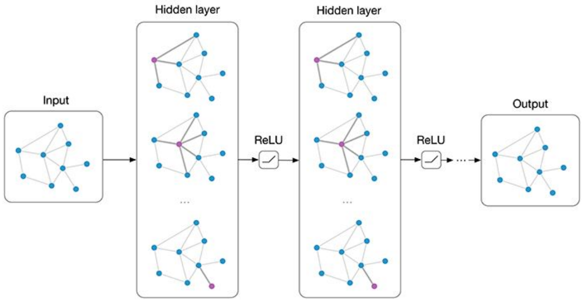
Training Graph Neural Networks with 1000 Layers

Authors	Guohao Li, Matthias Müller, Bernard Ghanem, Vladlen Koltun
Publication date	2021/6/14
Journal	International Conference on Machine Learning (ICML)

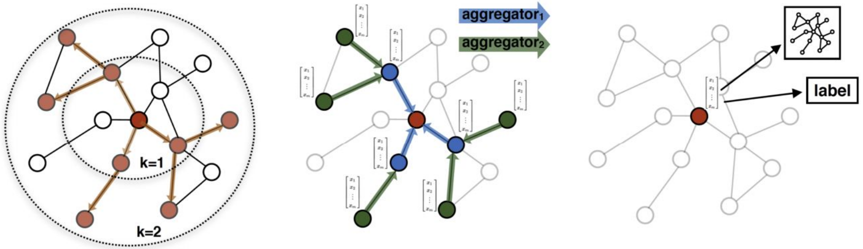
ICML International Conference On Machine Learning

Making GCNs Go as Deep as CNNs:

Skip Connections and Dilated Convolutions on Graphs;
Message Aggregation Functions;
Memory Efficiency

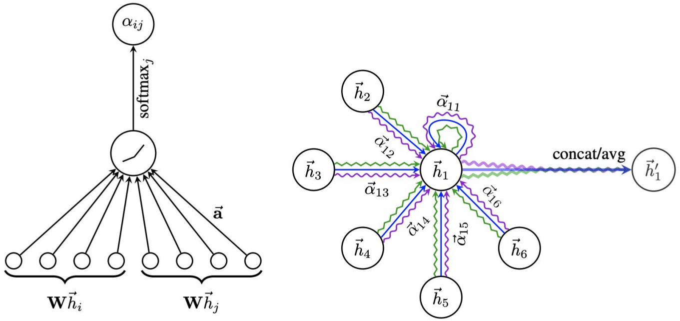


Kipf, T.N. and Welling, M., 2016. Semi-Supervised Classification with Graph Convolutional Networks.

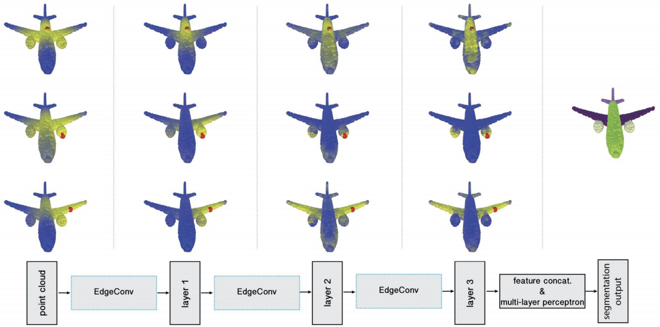


Hamilton, W.L., Ying, R. and Leskovec, J., 2017. Inductive Representation Learning on Large Graphs.

Most of SOTA GNNs are not deeper than 3 or 4 layers.

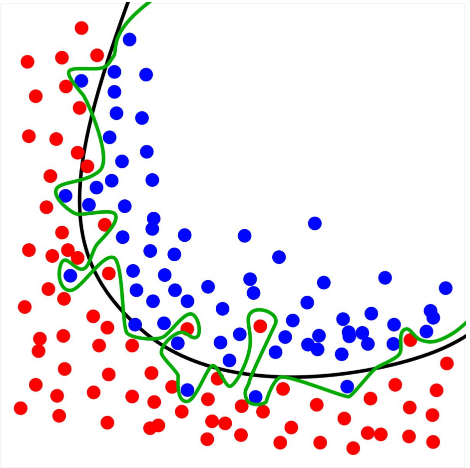


Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y., 2018. Graph Attention Networks.



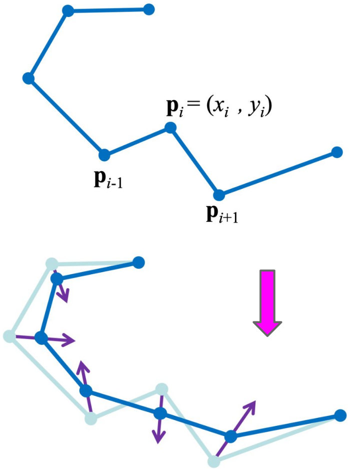
Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M. and Solomon, J.M., 2018. Dynamic Graph CNN for Learning on Point Clouds.

Why GNNs are limited to shallow structures?



Overfitting

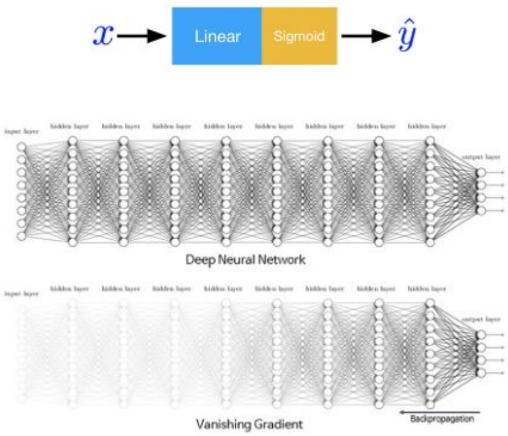
Figures from https://en.wikipedia.org/wiki/Overfitting



$$\mathbf{p}_i \leftarrow \mathbf{p}_i + \frac{1}{2}L(\mathbf{p}_i) \quad L(\mathbf{p}_i) = \frac{1}{2}(\mathbf{p}_{i+1} - \mathbf{p}_i) + \frac{1}{2}(\mathbf{p}_{i-1} - \mathbf{p}_i)$$

Oversmoothing

Figures from https://graphics.stanford.edu/courses/cs468-12-spring/LectureSlides/06_smoothing.pdf

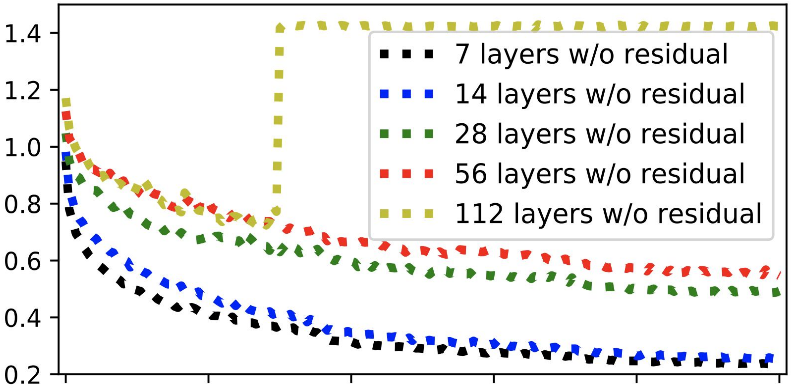


Vanishing Gradient

Figures from https://www.kaggle.com/getting-started/118228

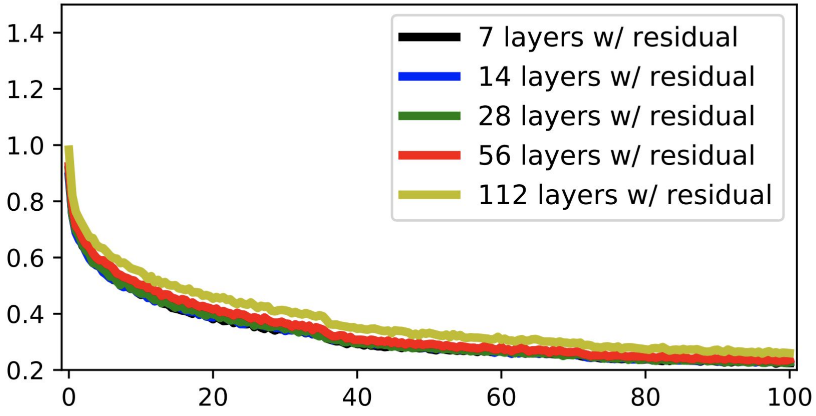
Training Loss of GCNs with varying depth

Deeper GCNs don’t converge well.



PlainGCNs

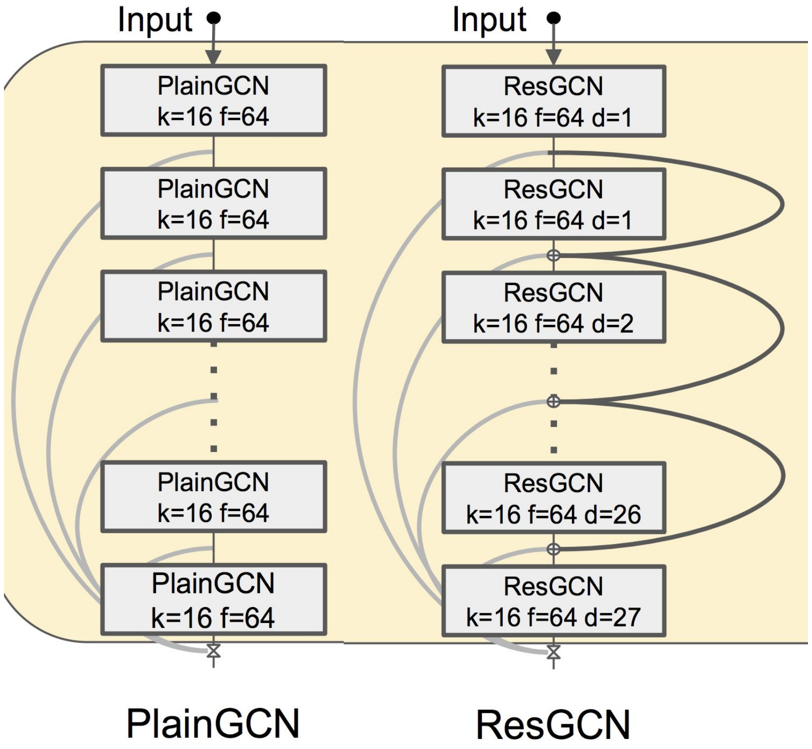
Even a 112-layer deep GCN converges well!!!



ResGCNs

Residual Graph Connections

DeepGCNs.org



$$\mathcal{G}_{l+1} = \mathcal{H}(\mathcal{G}_l, \mathcal{W}_l) = \mathcal{F}(\mathcal{G}_l, \mathcal{W}_l) + \mathcal{G}_l.$$

An example:

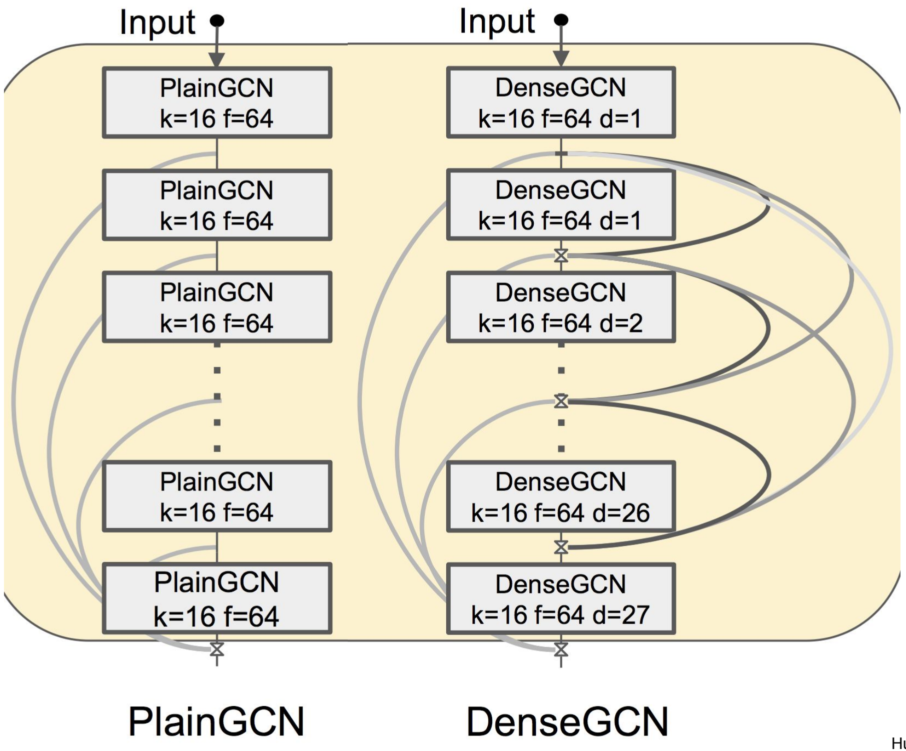
$$h^{res}_{\mathcal{N}^{(d)}(v_l)} = max\left(\{h_{u_l} - h_{v_l} | u_l \in \mathcal{N}^{(d)}(v_l)\}\right),$$ Aggregate

$$h^{res}_{v_{l+1}} = mlp\left(concat\left(h_{v_l}, h^{res}_{\mathcal{N}^{(d)}(v_l)}\right)\right),$$ Update

$$h_{v_{l+1}} = h^{res}_{v_{l+1}} + h_{v_l}.$$ Skip connection

He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

Dense Graph Connections

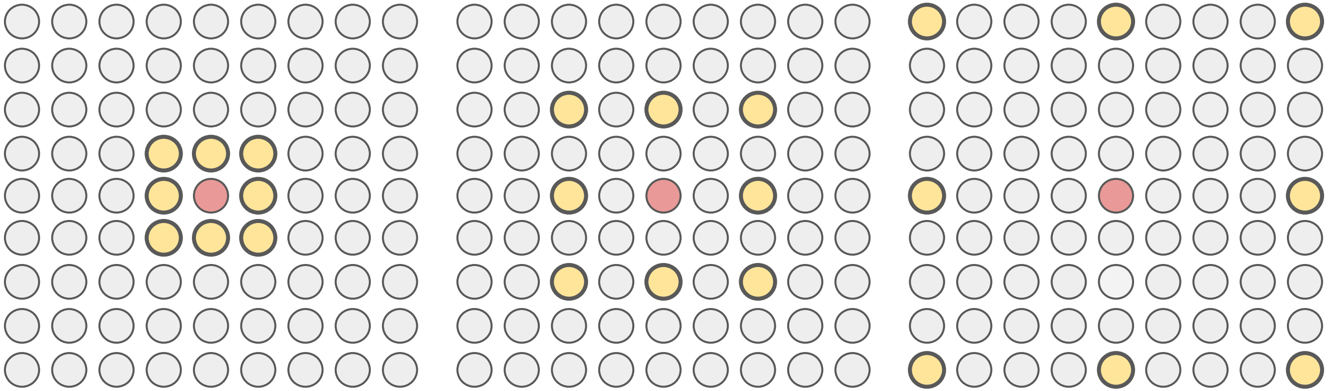


$$\begin{aligned}\mathcal{G}_{l+1} &= \mathcal{H}(\mathcal{G}_l, \mathcal{W}_l) \\ &= \mathcal{T}(\mathcal{F}(\mathcal{G}_l, \mathcal{W}_l), \mathcal{G}_l) \\ &= \mathcal{T}(\mathcal{F}(\mathcal{G}_l, \mathcal{W}_l), ..., \mathcal{F}(\mathcal{G}_0, \mathcal{W}_0), \mathcal{G}_0).\end{aligned}$$

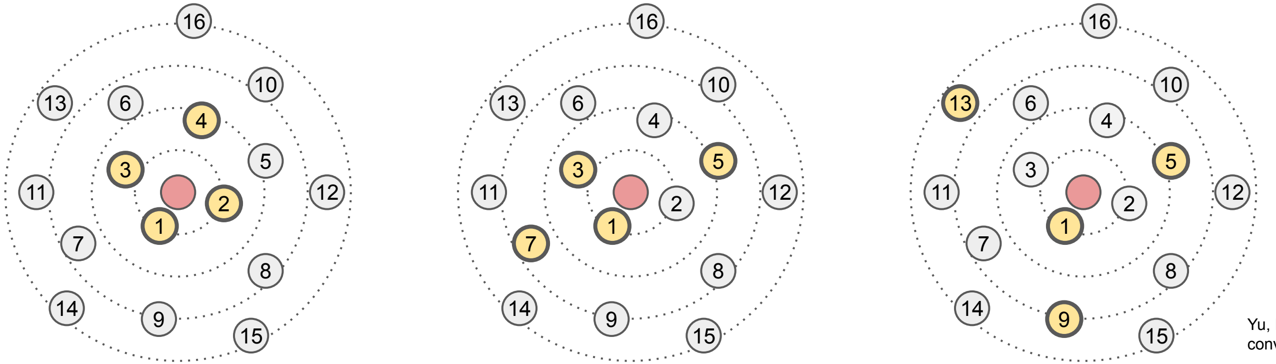
Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Dilated Graph Convolutions

DeepGCNs.org



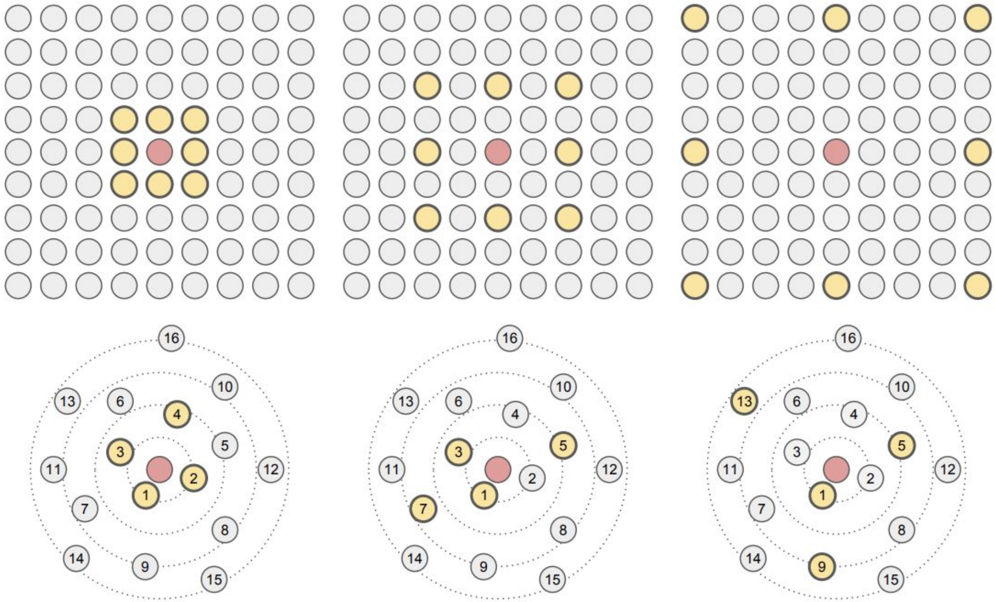
Dilated Convolution on a regular graph, e.g. 2D image



Dilated graph Convolution on an irregular graph, e.g. 3D point cloud

Yu, Fisher, and Vladlen Koltun. "Multi-scale context aggregation by dilated convolutions." International Conference on Learning Representations. 2016.

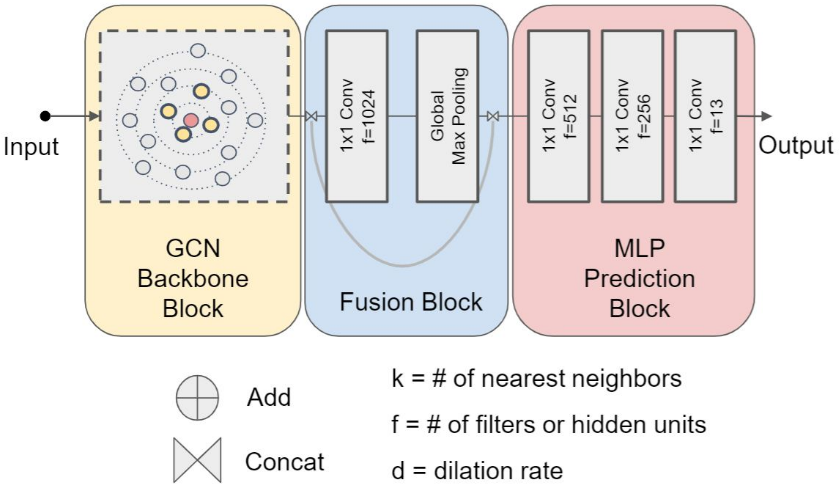
Dilated Graph Convolutions

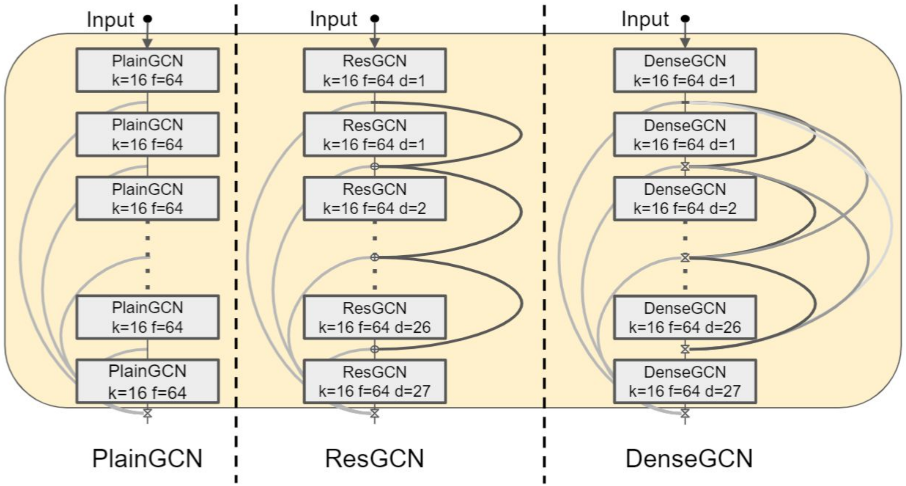


$$\mathcal{N}^{(d)}(v) = \{u_1, u_{1+d}, u_{1+2d}, \ldots, u_{1+(k-1)d}\}.$$

d = dilation rate

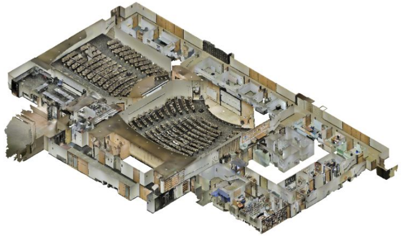
Deep Graph Convolutional Networks (DeepGCNs)



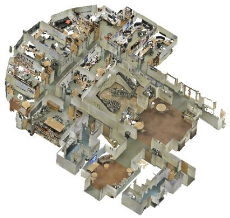


Stanford 3D Large-Scale Indoor Spaces Dataset

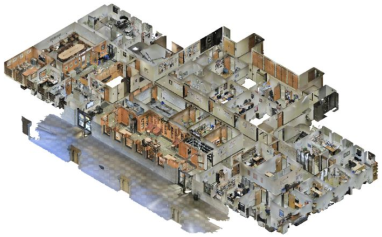
Area 1



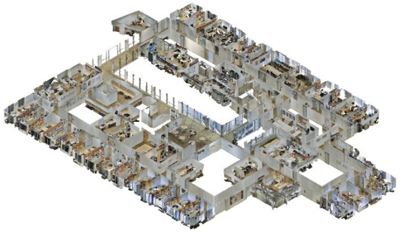
Area 2



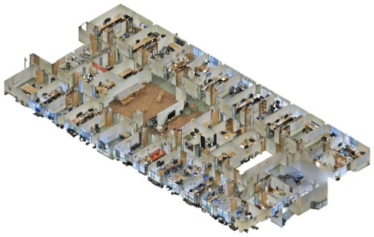
Area 3



Area 4



Area 5



Area 6

~ 700 million points

Node features: coordinates and colors

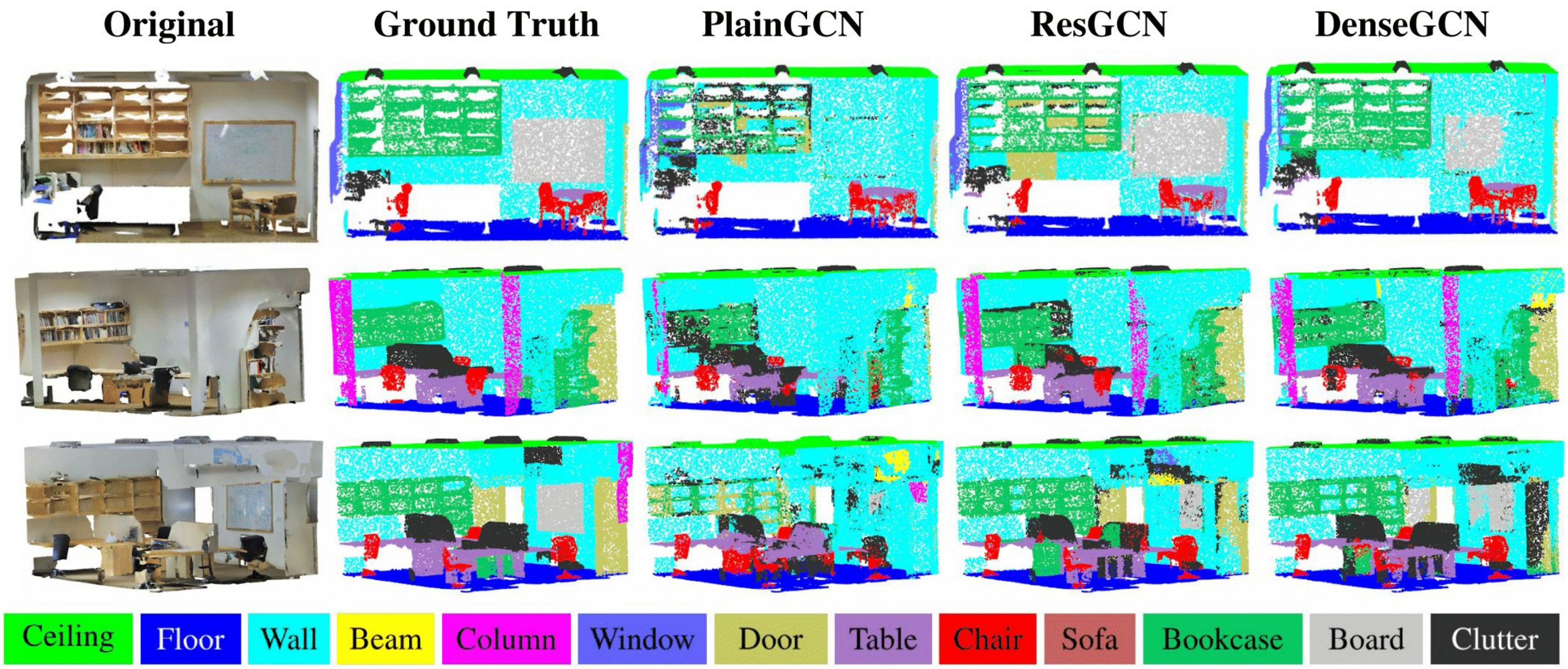
Node classification with 13 classes

Construct edges by kNN

http://buildingparser.stanford.edu/dataset.html

Graph Learning on 3D Point Clouds

DeepGCNs.org



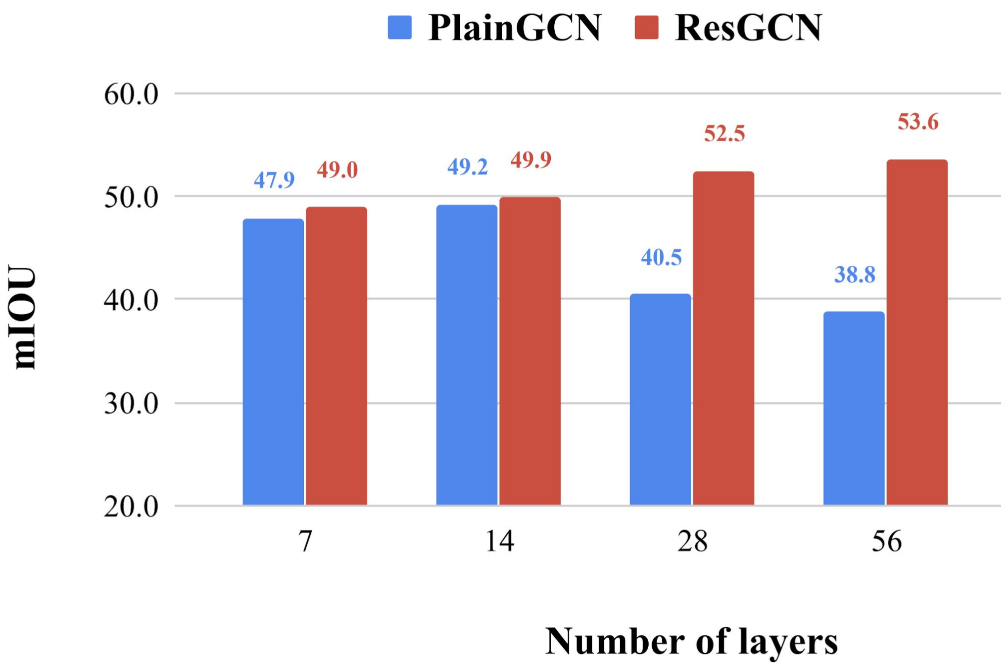
We outperform other SOTAs in 9 out of 13 classes

Method	OA	mIOU	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [29]	78.5	47.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	54.1	42.0	9.6	38.2	29.4	35.2
MS+CU [8]	79.2	47.8	88.6	**95.8**	67.3	36.9	24.9	48.6	52.3	51.9	45.1	10.6	36.8	24.7	37.5
G+RCU [8]	81.1	49.7	90.3	92.1	67.9	**44.7**	24.2	52.3	51.2	58.1	47.4	6.9	39.0	30.0	41.9
PointNet++ [31]	-	53.2	90.2	91.7	73.1	42.7	21.2	49.7	42.3	62.7	59.0	19.6	45.8	48.2	45.6
3DRNN+CF [51]	**86.9**	56.3	92.9	93.8	73.1	42.5	25.9	47.6	59.2	60.4	**66.7**	24.8	**57.0**	36.7	51.6
DGCNN [43]	84.1	56.1	-	-	-	-	-	-	-	-	-	-	-	-	-
ResGCN-28 (*Ours*)	85.9	**60.0**	**93.1**	95.3	**78.2**	33.9	**37.4**	**56.1**	**68.2**	**64.9**	61.0	**34.6**	51.5	**51.1**	**54.4**

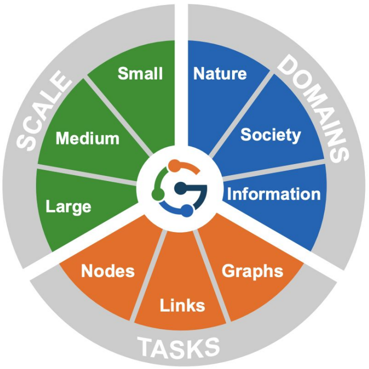
Table 1. Comparison of ResGCN-28 with state-of-the-art.

PlainGCN VS. ResGCN

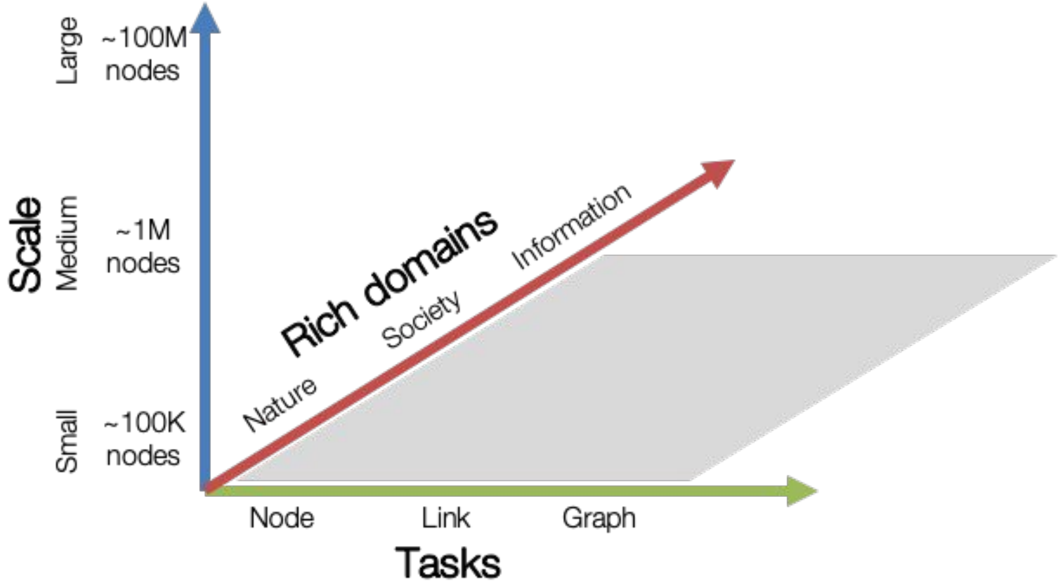
DeepGCNs.org



Datasets: Open Graph Benchmark (OGB)



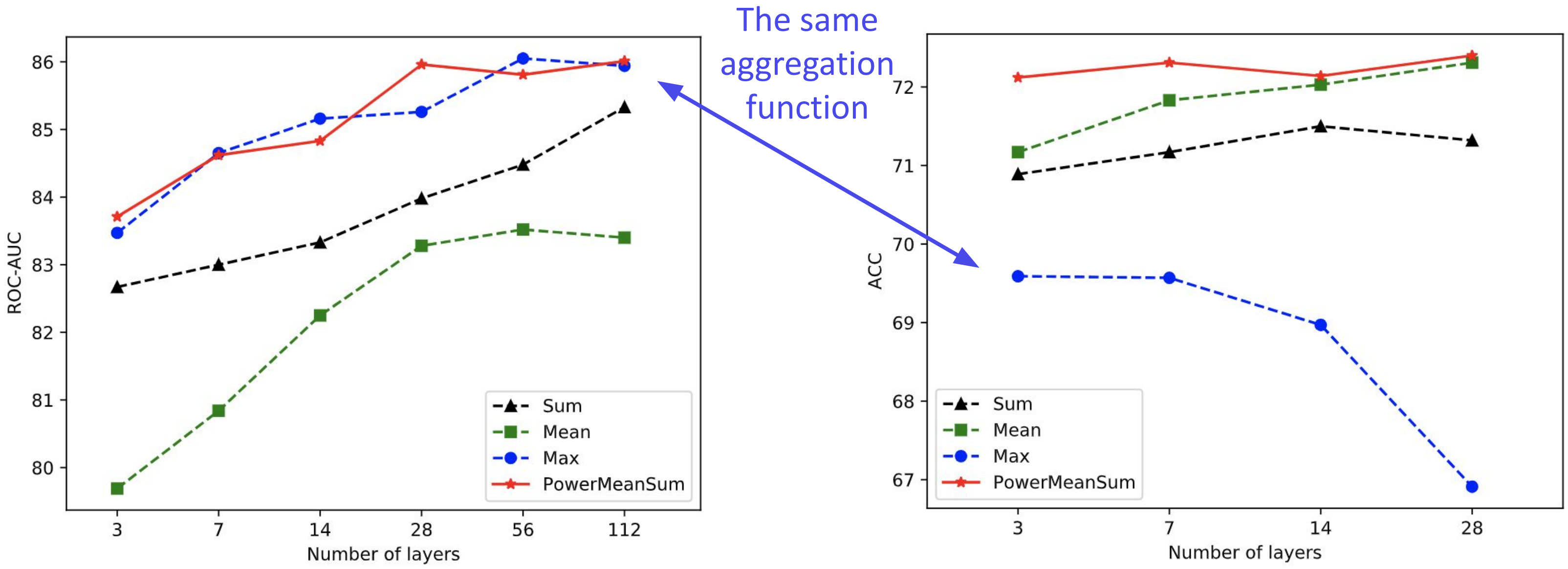
OGB datasets



Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M. and Leskovec, J.. Open graph benchmark: Datasets for machine learning on graphs. NeurIPS 2020.

Aggregation Functions

DeepGCNs.org



(a) different aggregators on the obgn-protein dataset. **(b) different aggregations on the obgn-arxiv dataset.**

Aggregation functions perform very differently on different datasets.

Message Passing

DeepGCNs.org

Node Features

Neighbor's Features

Edge Features

$$\mathbf{x}_i^{(k)} = \gamma^{(k)} \left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)} \, \phi^{(k)} \left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{i,j} \right) \right),$$

Differentiable (±Learnable) Function e.g., MLPs

Permutation Invariant Function e.g., sum, mean or max

Differentiable (±Learnable) Function e.g., MLPs

$\mathcal{N}_i$: neighbor indices

By https://pytorch-geometric.readthedocs.io

Aggregation Functions

DeepGCNs.org

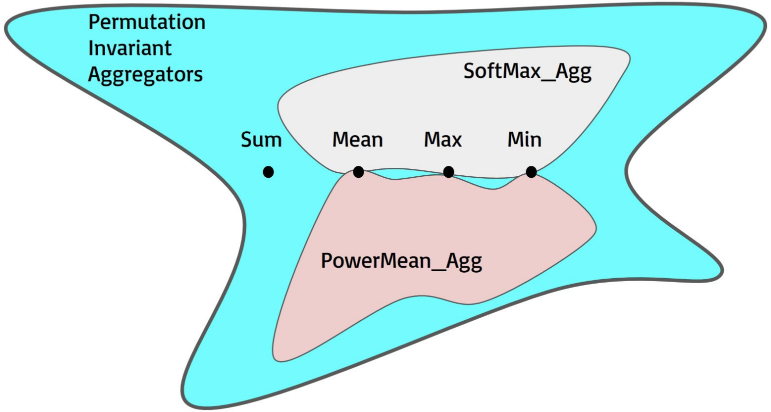


Illustration of Generalized Message Aggregation Functions

Generalized mean-max aggregation function:

$$SoftMax\_Agg_{\beta}(\cdot) = \sum_{u \in \mathcal{N}(v)} \frac{\exp(\beta \mathbf{m}_{vu})}{\sum_{i \in \mathcal{N}(v)} \exp(\beta \mathbf{m}_{vi})} \cdot \mathbf{m}_{vu}.$$

$$\lim_{\beta \to 0} \mathrm{SoftMax\_Agg}_{\beta}(\cdot) = \mathrm{Mean}(\cdot)$$

$$\lim_{\beta \to \infty} \mathrm{SoftMax\_Agg}_{\beta}(\cdot) = \mathrm{Max}(\cdot)$$

$$PowerMean\_Agg_{p}(\cdot) = (\frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{m}_{vu}^{p})^{1/p}.$$

$$\mathrm{PowerMean\_Agg}_{p=1}(\cdot) = \mathrm{Mean}(\cdot)$$

$$\lim_{p \to \infty} \mathrm{PowerMean\_Agg}_{p}(\cdot) = \mathrm{Max}(\cdot)$$

Aggregation Functions

DeepGCNs.org

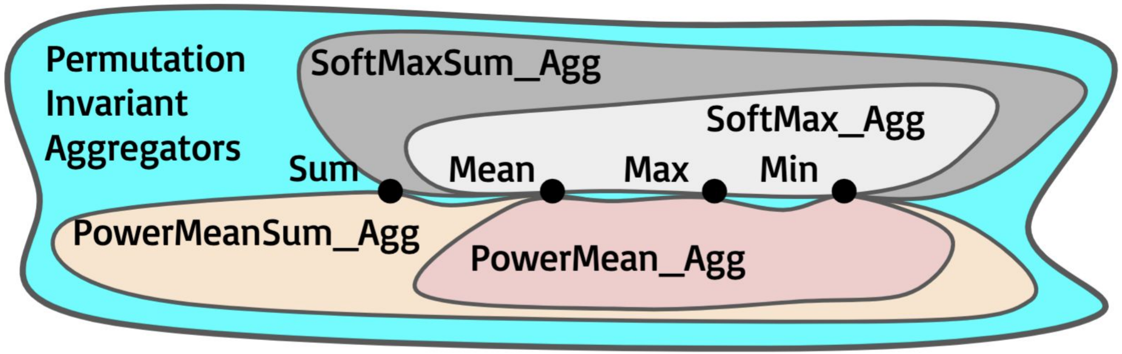


Illustration of Generalized Message Aggregation Functions

Generalized mean-max aggregation function:

$$SoftMax\_Agg_{\beta}(\cdot) = \sum_{u \in \mathcal{N}(v)} \frac{\exp(\beta \mathbf{m}_{vu})}{\sum_{i \in \mathcal{N}(v)} \exp(\beta \mathbf{m}_{vi})} \cdot \mathbf{m}_{vu}.$$

$$PowerMean\_Agg_{p}(\cdot) = \left(\frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{m}_{vu}^{p}\right)^{1/p}.$$

Generalized mean-max-sum aggregation function:

$$\left|\mathcal{N}(v)\right|^{y} \cdot \zeta_{x}(\cdot)$$

Differentiable aggregation functions

Xu, K., Hu, W., Leskovec, J. and Jegelka, S.. How powerful are graph neural networks?. ICLR 2018.

	GraphSAGE	GCN	GaAN				Ours
ogbn-proteins	77.68 ± 0.20	72.51 ± 0.35	78.03 ± 0.73				**86.16 ± 0.16**
	GraphSAGE	GCN	GaAN	GCNII	JKNet	DAGNN	
ogbn-arxiv	71.49 ± 0.27	71.74 ± 0.29	71.97 ± 0.24	**72.74 ± 0.16**	72.19 ± 0.21	72.09 ± 0.25	72.32 ± 0.27
	GraphSAGE	GCN	ClusterGCN	GraphSAINT	GAT		
ogbn-products	78.29 ± 0.16	75.64 ± 0.21	78.97 ± 0.33	80.27 ± 0.26	79.45 ± 0.59		**81.64 ± 0.30**
	GIN	GCN	GIN*	GCN*	HIMP		
ogbg-molhiv	75.58 ± 1.40	76.06 ± 0.97	77.07 ± 1.49	75.99 ± 1.19	78.80 ± 0.82		**78.87 ± 1.24**
ogbg-molpcba	22.66 ± 0.28	20.20 ± 0.24	27.03 ± 0.23	24.24 ± 0.34			**27.81 ± 0.38***
ogbg-ppa	68.92 ± 1.00	68.39 ± 0.84	70.37 ± 1.07	68.57 ± 0.61			**77.12 ± 0.71**
	GraphSAGE	GCN	DeepWalk				
ogbl-collab	48.10 ± 0.81	44.75 ± 1.07	50.37 ± 0.34				**52.73 ± 0.47**

Table 4. DeeperGCN achieves SOTA results on 6 OGB datasets.

Results

DeepGCNs.org

Leaderboard for ogbg-ppa

~7%

The multi-class classification accuracy on the test and validation sets. The higher, the better.

Package: >=1.1.1

Rank	Method	Test Accuracy	Validation Accuracy	Contact	References	#Params	Hardware	Date
1	DeeperGCN	0.7712 ± 0.0071	0.7313 ± 0.0078	Guohao Li - DeepGCNs.org	Paper, Code	2,336,421	NVIDIA Tesla V100 (32GB GPU)	Jun 16, 2020
2	GIN+virtual node	0.7037 ± 0.0107	0.6678 ± 0.0105	Weihua Hu – OGB team	Paper, Code	3,288,042	GeForce RTX 2080 (11GB GPU)	May 1, 2020
3	GIN	0.6892 ± 0.0100	0.6562 ± 0.0107	Weihua Hu – OGB team	Paper, Code	1,836,942	GeForce RTX 2080 (11GB GPU)	May 1, 2020
4	GCN+virtual node	0.6857 ± 0.0061	0.6511 ± 0.0048	Weihua Hu – OGB team	Paper, Code	1,930,537	GeForce RTX 2080 (11GB GPU)	May 1, 2020
5	GCN	0.6839 ± 0.0084	0.6497 ± 0.0034	Weihua Hu – OGB team	Paper, Code	479,437	GeForce RTX 2080 (11GB GPU)	May 1, 2020

Leaderboard for ogbg-molpcba

The Average Precision (AP) score on the test and validation sets. The higher, the better.

Note: The evaluation metric has been changed from PRC-AUC (Aug 11, 2020).

Package: >=1.2.2

Rank	Method	Test AP	Validation AP	Contact	References	#Params	Hardware	Date
1	DeeperGCN+virtual node	0.2781 ± 0.0038	0.2920 ± 0.0025	Guohao Li - DeepGCNs.org	Paper, Code	5,550,208	NVIDIA Tesla V100 (32GB GPU)	Aug 11, 2020
2	GIN+virtual node	0.2703 ± 0.0023	0.2798 ± 0.0025	Weihua Hu – OGB team	Paper, Code	3,374,533	GeForce RTX 2080 (11GB GPU)	Aug 11, 2020
3	GCN+virtual node	0.2424 ± 0.0034	0.2495 ± 0.0042	Weihua Hu – OGB team	Paper, Code	2,017,028	GeForce RTX 2080 (11GB GPU)	Aug 11, 2020
4	GIN	0.2266 ± 0.0028	0.2305 ± 0.0027	Weihua Hu – OGB team	Paper, Code	1,923,433	GeForce RTX 2080 (11GB GPU)	Aug 11, 2020
5	GCN	0.2020 ± 0.0024	0.2059 ± 0.0033	Weihua Hu – OGB team	Paper, Code	565,928	GeForce RTX 2080 (11GB GPU)	Aug 11, 2020

Leaderboard for ogbn-proteins

~7.5%

The ROC-AUC score on the test set. The higher, the better.

Rank	Method	ROC-AUC	Contact	References	Date
1	DeeperGCN	0.8580 ± 0.0017	Guohao Li - DeepGCNs.org	Paper, Code	Jun 16, 2020
2	GeniePath-BS	0.7825 ± 0.0035	Zhengwei WU (AGL Team)	Paper, Code	Jun 10, 2020
3	GaAN	0.7803 ± 0.0073	Wenjin Wang (PGL Team)	Paper, Code	May 26, 2020
4	GraphSAGE	0.7768 ± 0.0020	Matthias Fey – OGB team	Paper, Code	May 1, 2020
5	MLP	0.7204 ± 0.0048	Matthias Fey – OGB team	Paper, Code	May 1, 2020
6	Node2vec	0.6881 ± 0.0065	Matthias Fey – OGB team	Paper, Code	May 1, 2020
7	GCN	0.6511 ± 0.0152	Matthias Fey – OGB team	Paper, Code	May 1, 2020

DeeperGCN ranked top 1 on several datasets at the time of submission.

Memory complexity of training GNNs

Full batch: **O(LND)**

L - number of layers

N - number of nodes

D - number of features

(assume D is the same for all the layers)

Mini-batch:

GraphSage, ClusterGCN, so on.

Cluster-GCN: **O(LND) - > O(LBD)**

B - number of nodes in subgraphs, B<N

This work:

O(LND) - > O(ND)

- How can we reduce memory complexity?

- Can we reduce the memory complexity in the L dimension?

Chiang, Wei-Lin, et al. "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks." SIGKDD. 2019.

Memory Efficient GNNs

$$\langle X_1, X_2, ..., X_C \rangle \mapsto \langle X'_1, X'_2, ..., X'_C \rangle$$

Reversible GNN:

Forward:

$$X'_0 = \sum_{i=2}^{C} X_i$$
$$X'_i = f_{w_i}(X'_{i-1}, A, U) + X_i,\ i \in \{1, \cdots, C\},$$

Inverse:

$$X_i = X'_i - f_{w_i}(X'_{i-1}, A, U),\ i \in \{2, \cdots, C\}$$
$$X'_0 = \sum_{i=2}^{C} X_i$$
$$X_1 = X'_1 - f_{w_1}(X'_0, A, U).$$

Weight-tied Reversible GNN:

$$f_{w_i}^{(1)} := f_{w_i}^{(2)} \ldots := f_{w_i}^{(L)},\ i \in \{1, \cdots, C\}$$

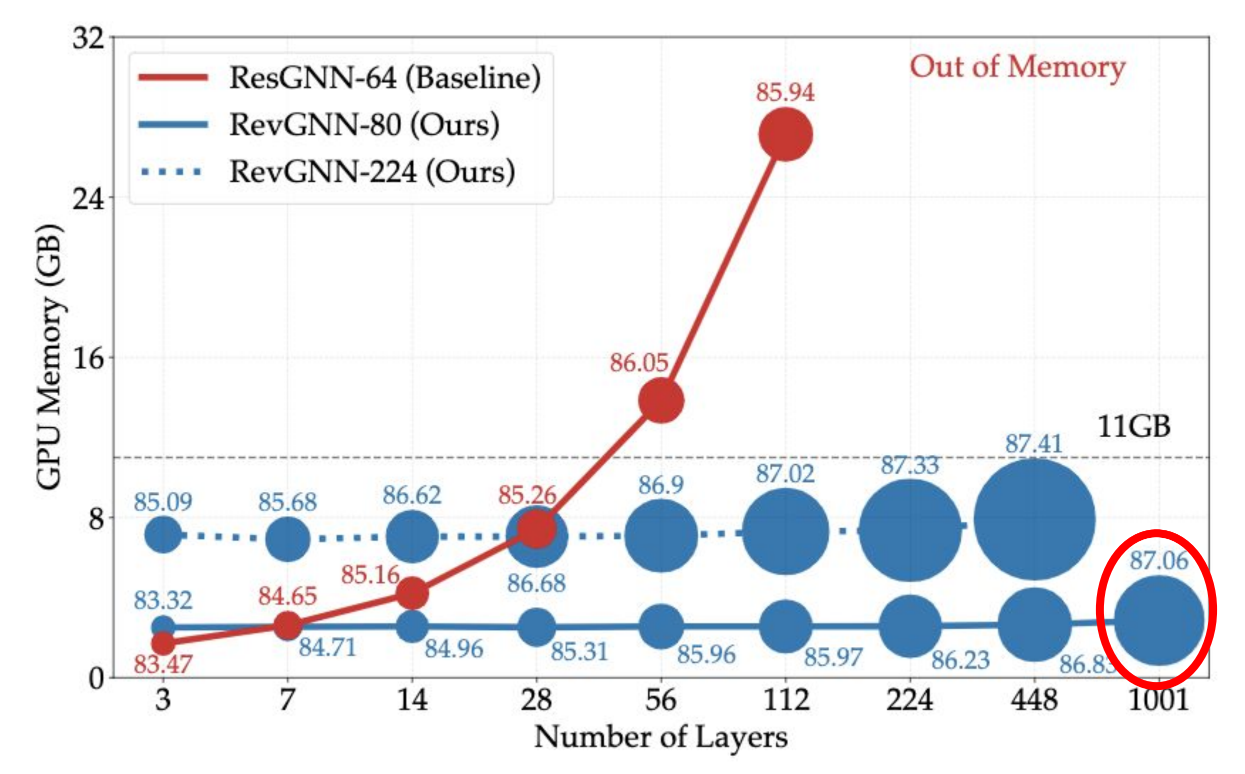
DEQ-GNN:

$$Z^* = f_w^{\mathrm{DEQ}}(Z^*, X, A, U),$$

Do not need to store the intermediate node features.

O(LND) - > O(ND)

Results: Constant Memory with RevGNN



Train 1001-layer GNN with only 2.86G peak GPU memory!

The deepest GNN by one order of magnitude.

Results: SOTA with RevGNN (ogbn-proteins)

Rank	Method	Test ROC-AUC	Validation ROC-AUC	Contact	References	#Params	Hardware	Date
1	**RevGNN-Wide**	0.8824 ± 0.0015	0.9450 ± 0.0008	Guohao Li - DeepGCNs.org	Paper, Code	68,471,608	NVIDIA RTX 6000 (48G)	Jun 16, 2021
2	**RevGNN-Deep**	0.8774 ± 0.0013	0.9326 ± 0.0006	Guohao Li - DeepGCNs.org	Paper, Code	20,031,384	NVIDIA RTX 6000 (48G)	Jun 16, 2021
3	**GAT+BoT**	0.8765 ± 0.0008	0.9280 ± 0.0008	Yangkun Wang (DGL Team)	Paper, Code	2,484,192	Tesla A100 (40GB GPU)	Jun 16, 2021
4	GAT + labels + node2vec	0.8711 ± 0.0007	0.9217 ± 0.0011	Huixuan Chi	Paper, Code	6,360,470	Tesla V100 (32GB)	Jun 7, 2021
5	**GIPA**	0.8700 ± 0.0010	0.9187 ± 0.0003	Qinkai Zheng (GeaLearn Team)	Paper, Code	4,831,056	GeForce Titan RTX (24GB GPU)	May 13, 2021
6	**UniMP+CrossEdgeFeat**	0.8691 ± 0.0018	0.9258 ± 0.0009	Yelrose (PGL Team)	Paper, Code	1,959,984	Tesla V100 (32GB)	Nov 24, 2020
7	GAT+EdgeFeatureAtt	0.8682 ± 0.0021	0.9194 ± 0.0003	Yangkun Wang (DGL Team)	Paper, Code	2,475,232	p3.8xlarge (15GB GPU)	Nov 6, 2020
8	**UniMP**	0.8642 ± 0.0008	0.9175 ± 0.0006	Yunsheng Shi (PGL team)	Paper, Code	1,909,104	Tesla V100 (32GB)	Sep 8, 2020
9	**DeeperGCN+FLAG**	0.8596 ± 0.0027	0.9132 ± 0.0022	Kezhi Kong	Paper, Code	2,374,568	GeForce RTX 2080 Ti (11GB GPU)	Oct 20, 2020

68M parameters
(about a half of GPT)

GNN1000 is on State of AI Report 2021

State of AI Report 2021

The **State of AI Report** analyses the most interesting developments in AI. We aim to trigger an informed conversation about the state of AI and its implication for the future. The Report is produced by AI investors **Nathan Benaich** and **Ian Hogarth**.

Graph Neural Networks: improving the memory and parameter efficiency of large models

While very expressive and powerful, GNN model size doesn't scale well alongside dataset size due to the complexity of modelling millions of nodes and billions of connections. This is problematic for real-world problems when deploying large GNNs for equally large graph datasets without sacrificing model parameters.

- To overcome the memory bottleneck of large GNNs, we either need new hardware or model architectures that consume less memory.
- A method called deep reversible architectures (RevGNN) offers memory consumption that is independent of the number of layers in a model. RevGNN has a very large capacity at low memory cost and only slightly increased training time compared to baseline GNNs (ResGNN). Their deepest model, RevGNN-Wide, is the deepest GNN to date with 1000 layers.
- With only a fraction of the memory footprint, RevGNNs outperform some baselines on a node prediction benchmark task. But depth still doesn't help in most tasks, which is worthy of future investigation.

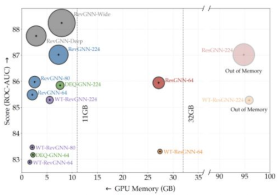


Figure: RevGNNs outperform existing models with significantly less memory consumption.

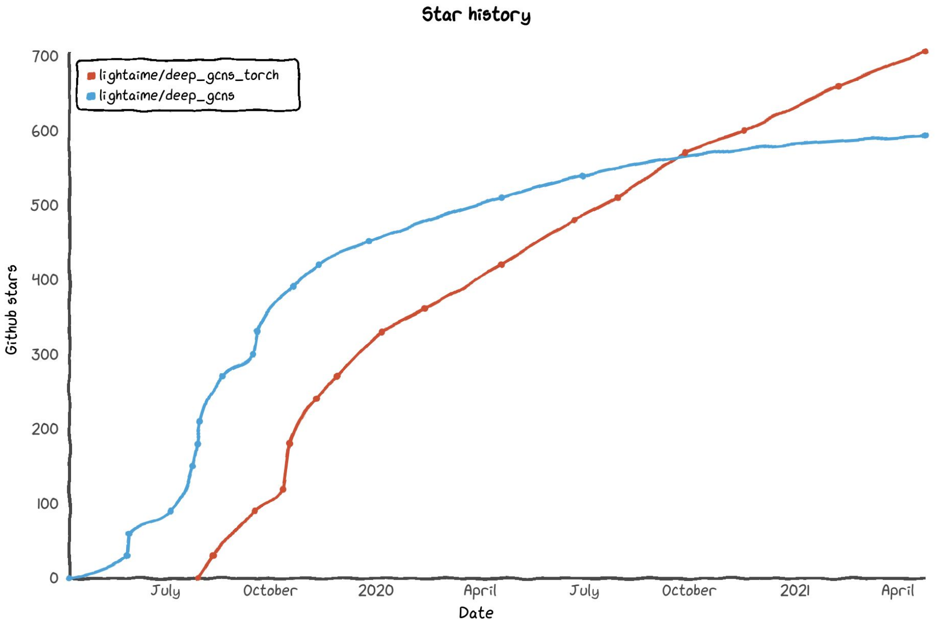
stateof.ai 2021

67

Google Slides

Open Source

DeepGCNs.org



> 1400 Stars (Pytorch + Tensorflow), 800 citations

PyTorch geometric

DEEP GRAPH LIBRARY

Available on PyG and DGL

Acknowledgement


Bernard Ghanem


Matthias Müller


Ali Thabet


Silvio Giancola


Neil Smith


Vladlen koltun


Guocheng Qian


Itzel C. Delgadillo


Abdulellah Abualshour

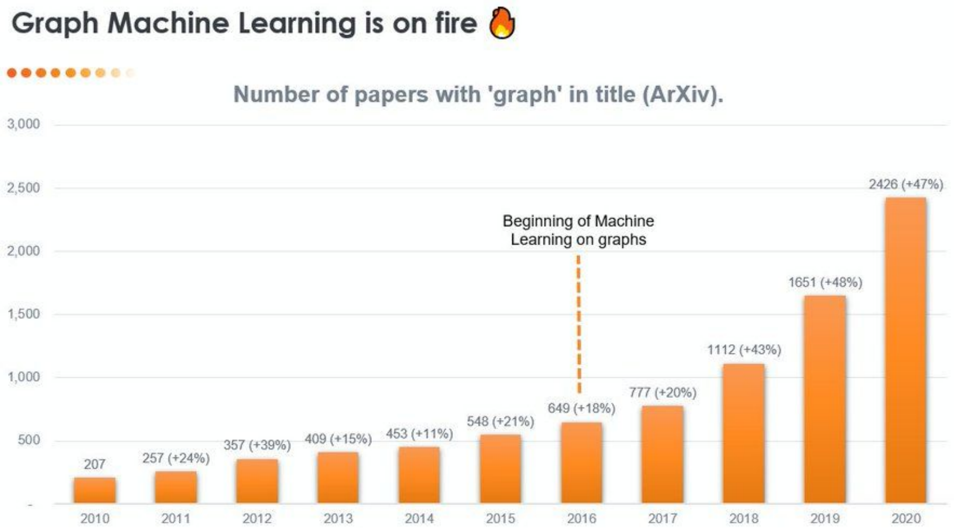

Chenxin Xiong


Jesus Zarzar


Mengmeng Xu


Kezhi Kong

Graph ML is hot



LOG CONFERENCE

Graph Neural Networks for Representation Learning on Graphs

1. **Why GNNs:**
   Grid;
   Graph

2. **Machine Learning on Graphs:**
   Graph Representation;
   Graph Learning Tasks;
   Permutation Invariance

3. **Graph Neural Networks:**
   A Simple GNN and GCN;
   Neural Message Passing;

4. **Making GCNs Go as Deep as CNNs:**
   Skip Connections;
   Generalized Aggregation;
   Reversible Connections

Thanks

Graph Neural Networks for Representation Learning on Graphs

Guohao Li
CS PhD Student @ KAUST
guohao.li@kaust.edu.sa

KAUST IVUL