PhD Defense

Towards Structured Intelligence with Deep Graph Neural Networks

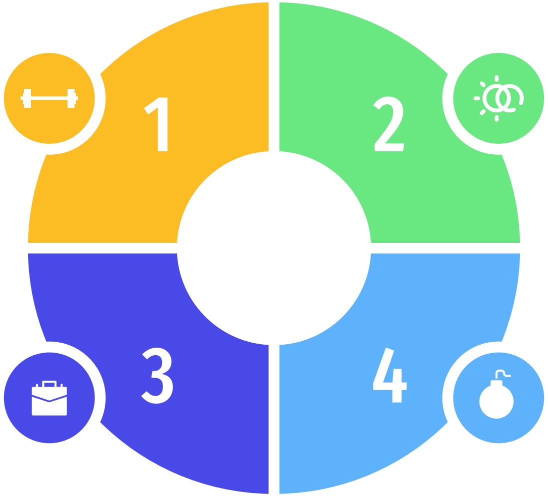
Guohao Li
CS PhD Student @ KAUST
guohao.li@kaust.edu.sa

KAUST IVUL

Towards Structured Intelligence with Deep Graph Neural Networks

Making GCNs Go as Deep as CNNs:
Skip Connections and Dilated Convolutions on Graphs

Making GCNs Go as Deep as CNNs:
Message Aggregation Functions;
Memory Efficiency

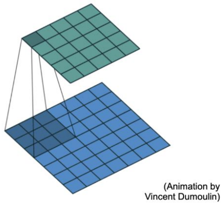


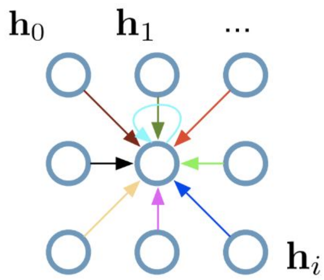
Automate GNN Architecture Design:
Sequential Greedy Architecture Search;
Latency Constraint

Ongoing Work and Research Plan:
Structured Navigation;
Research Plan

CNN vs. GNN - Comparison

Single CNN layer with 3x3 filter:



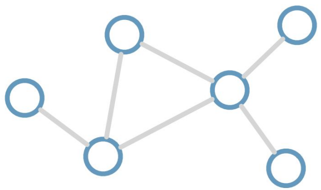


Full update:

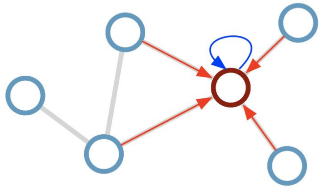
$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

Convolutional Neural Network (CNN)

Consider this undirected graph:



Calculate update for node in red:



Update rule:

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

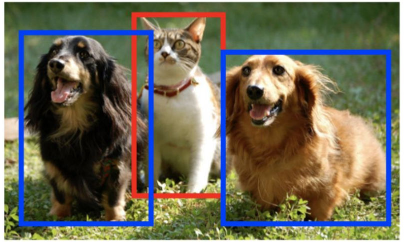
Graph Convolutional Network (GCN)

Slides by Thomas Kipf

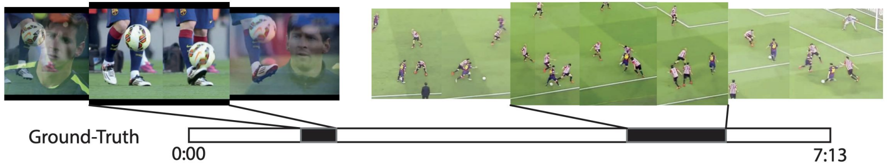
Grid Data vs. General Graphs

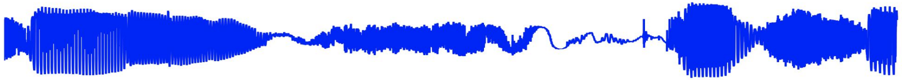
Why do we need graph neural networks?

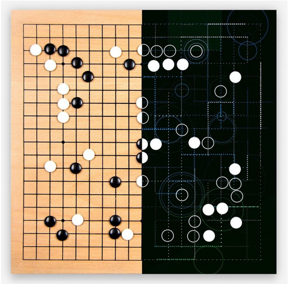
Grid Data vs. General Graphs



CAT, DOG



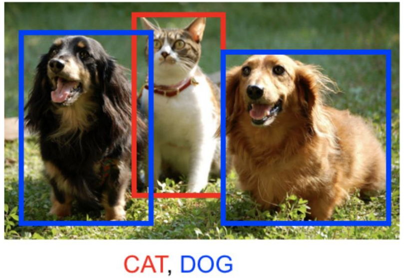


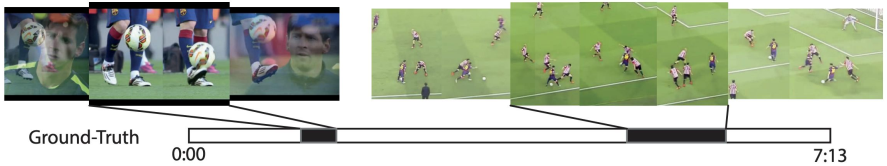


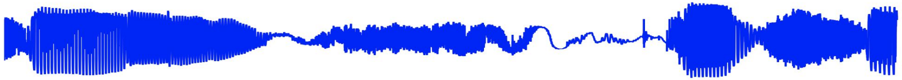
Grid Data:

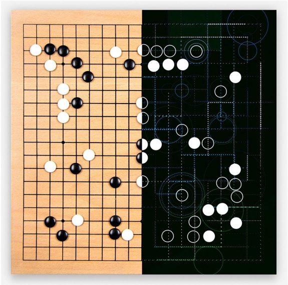
- Image
- Video
- Audio
- Text
- Grid game (Go)
- ...

Grid Data vs. General Graphs









Grid Data:

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

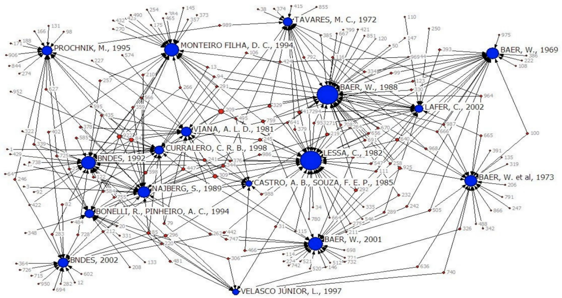
CNN works well

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

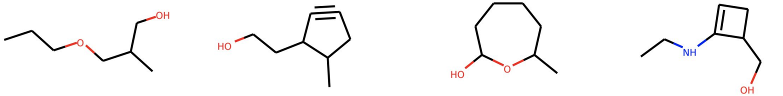


General Graphs:

- Social Networks
- Citation Networks

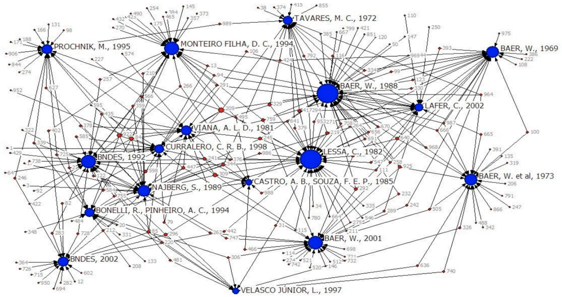
Grid Data vs. General Graphs

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



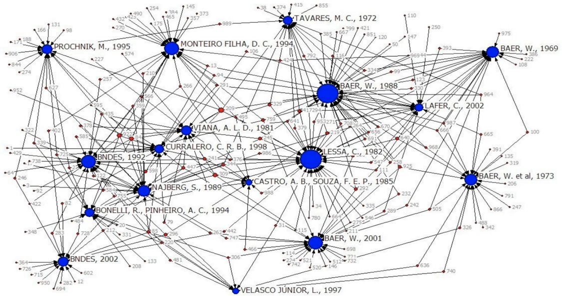
General Graphs:

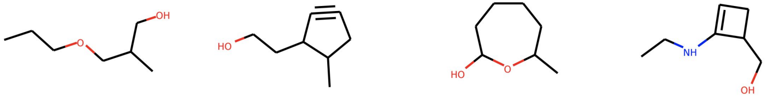
- Social Networks
- Citation Networks
- Molecules

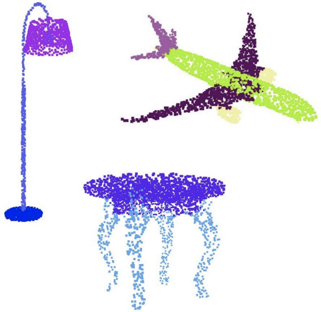


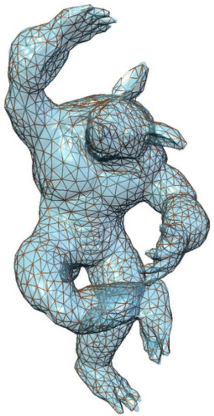
Grid Data vs. General Graphs

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







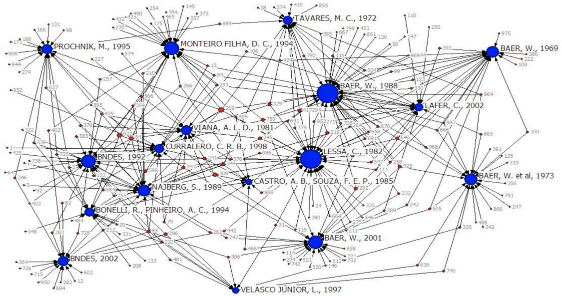


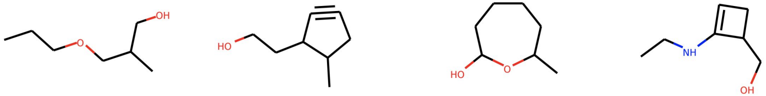
General Graphs:

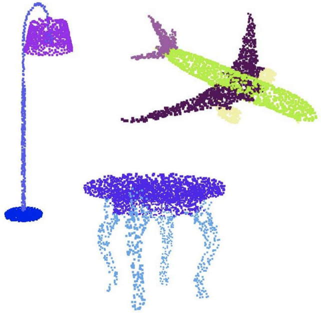
- Social Networks
- Citation Networks
- Molecules
- Point Clouds
- 3D Meshes
- ...

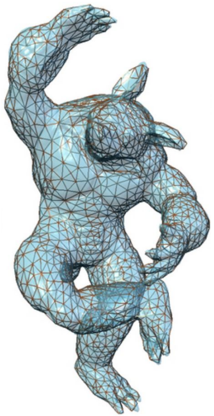
Grid Data vs. General Graphs

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







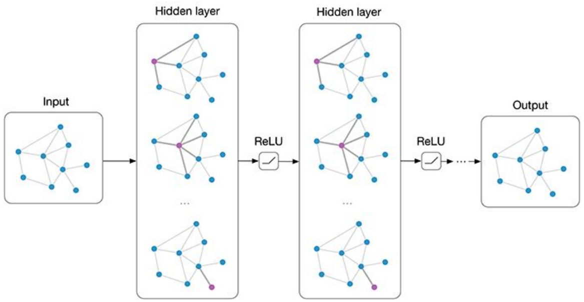


General Graphs :

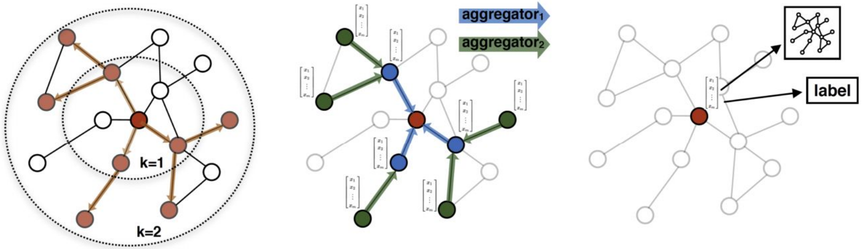
- Social Networks
- Citation Networks
- Molecules
- Point Clouds
- 3D Meshes
- ...

CNN doesn't work

GNN to rescue

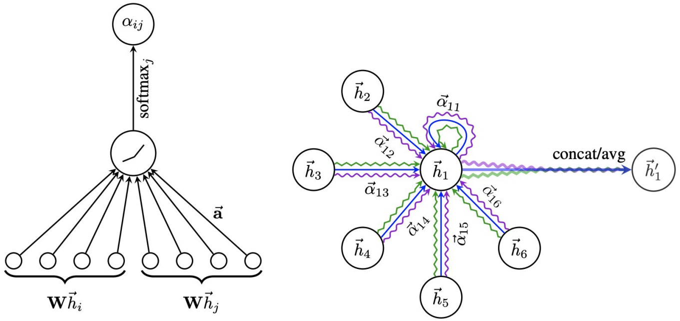


Kipf, T.N. and Welling, M., 2016. Semi-Supervised Classification with Graph Convolutional Networks.

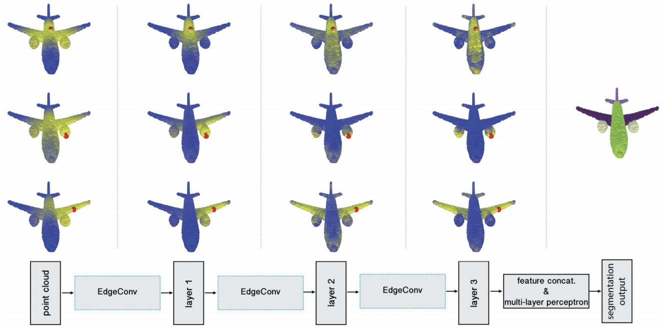


Hamilton, W.L., Ying, R. and Leskovec, J., 2017. Inductive Representation Learning on Large Graphs.

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

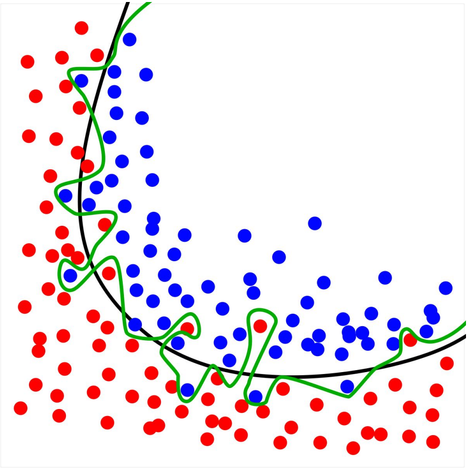


Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y., 2018. Graph Attention Networks.

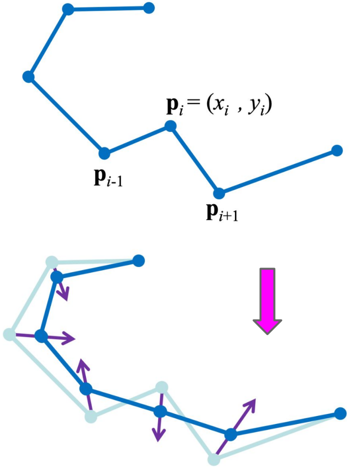


Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M. and Solomon, J.M., 2018. Dynamic Graph CNN for Learning on Point Clouds.

Why GNNs are limited to shallow architectures?



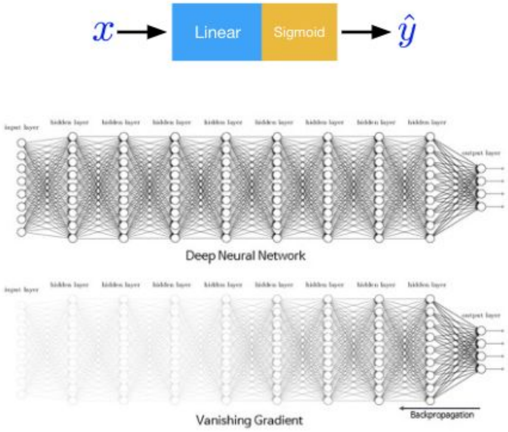
Overfitting



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

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

Making GCNs Go as Deep as CNNs:
Skip Connections and Dilated Convolutions on Graphs

Automate GNN Architecture Design:
Sequential Greedy Architecture Search;
Latency Constraint

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

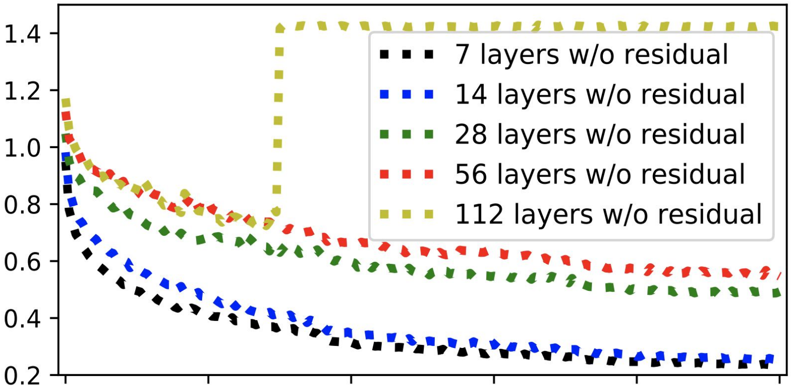
DeepGCNs: Making GCNs Go as Deep as CNNs

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

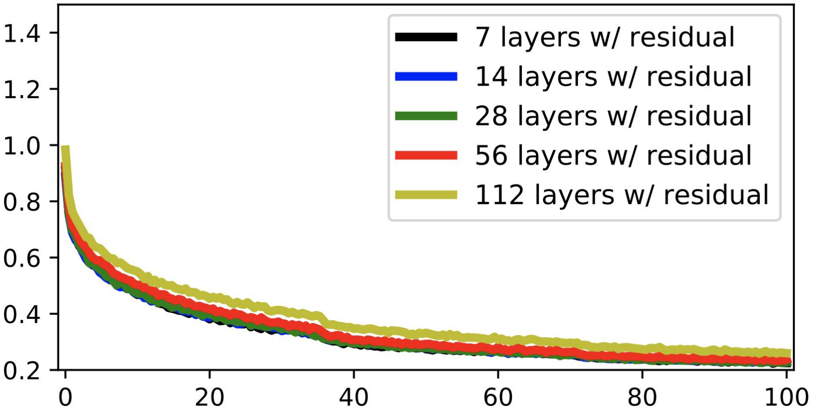
Training Loss of GCNs with varying depth

Deeper GCNs don’t converge well.



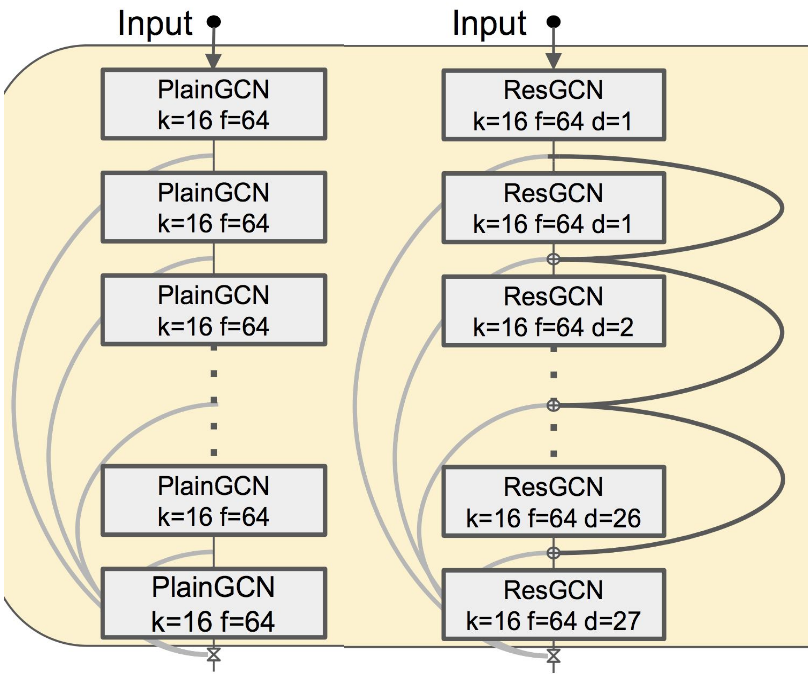
PlainGCNs

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



ResGCNs

Residual Graph Connections



PlainGCN ResGCN

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

An example: ResMRGCN

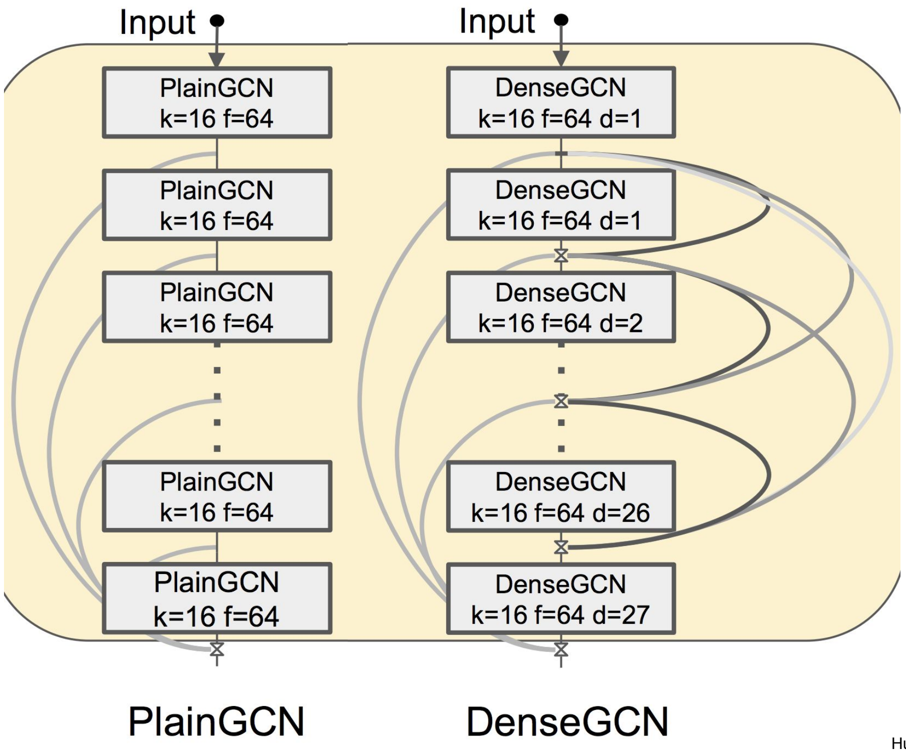
$$h^{res}_{\mathcal{N}^{(d)}(v_l)} = max\left(\{h_{u_l} - h_{v_l} | u_l \in \mathcal{N}^{(d)}(v_l)\}\right),$$ Aggregate

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

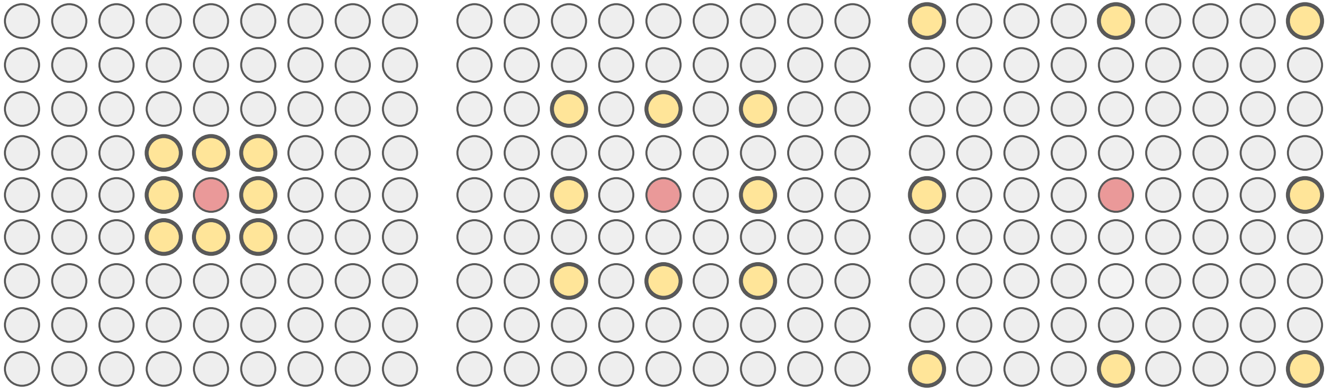
Dense Graph Connections



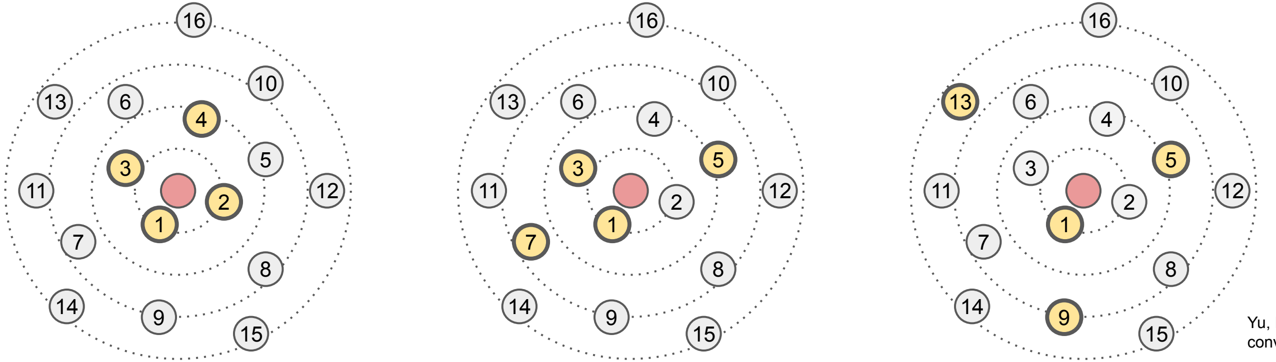
$$
\begin{aligned}
\mathcal{G}_{l+1} &= \mathcal{H}(\mathcal{G}_l, \mathcal{W}_l) \\
&= \mathcal{T}(\mathcal{F}(\mathcal{G}_l, \mathcal{W}_l), \mathcal{G}_l) \\
&= \mathcal{T}(\mathcal{F}(\mathcal{G}_l, \mathcal{W}_l), ..., \mathcal{F}(\mathcal{G}_0, \mathcal{W}_0), \mathcal{G}_0).
\end{aligned}
$$

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

Dilated Graph Convolutions



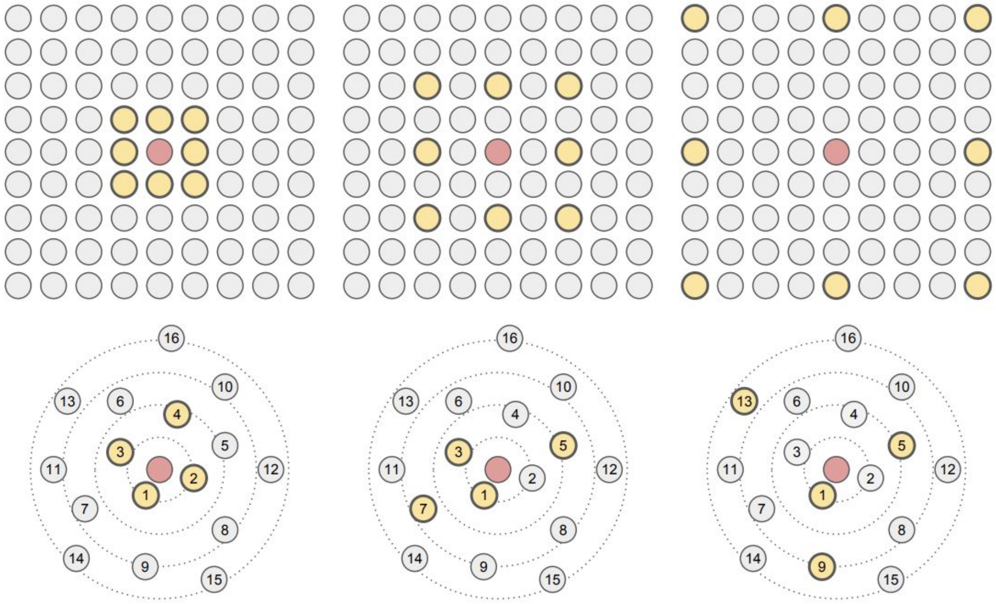
Dilated Convolution on a regular graph, e.g. 2D image



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.

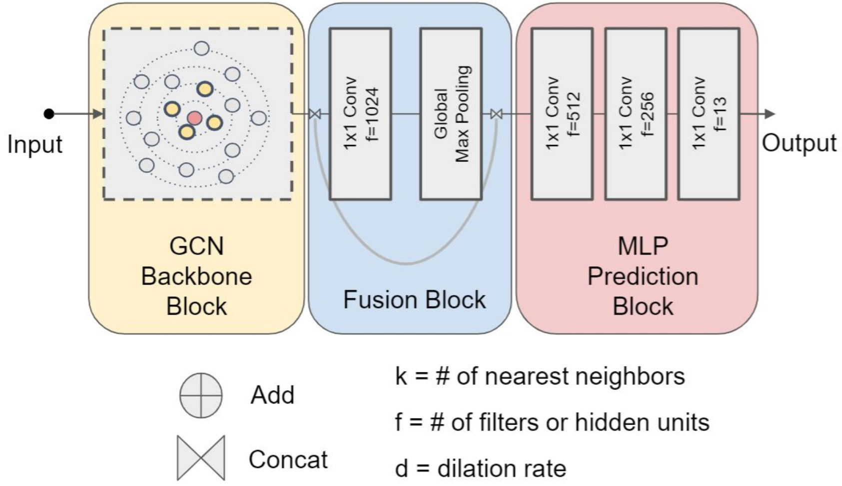
Dilated Graph Convolutions

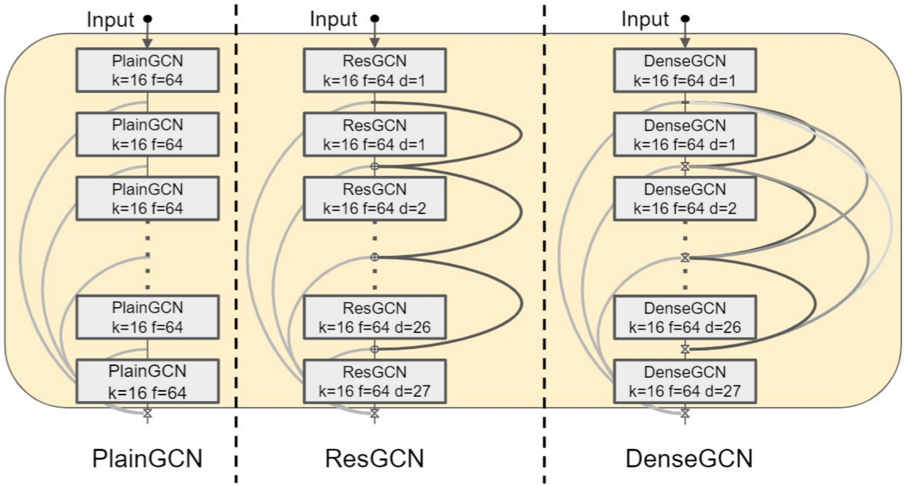


$$\mathcal{N}^{(d)}(v) = \{u_1, u_{1+d}, u_{1+2d}, \ldots, u_{1+(k-1)d}\}.$$

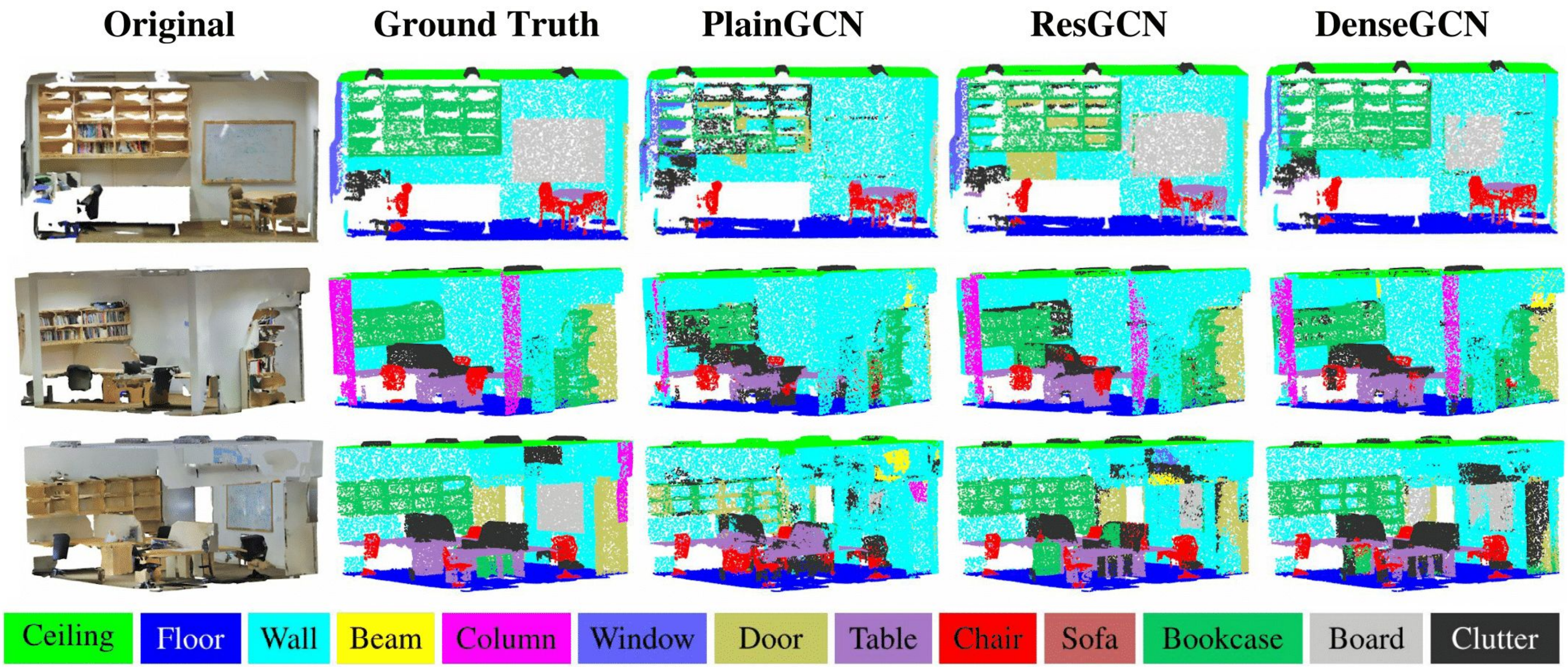
d = dilation rate

Deep Graph Convolutional Networks (DeepGCNs)





Graph Learning on 3D Point Clouds



We outperform other SOTA 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**

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

Consistent improvements across all the classes.

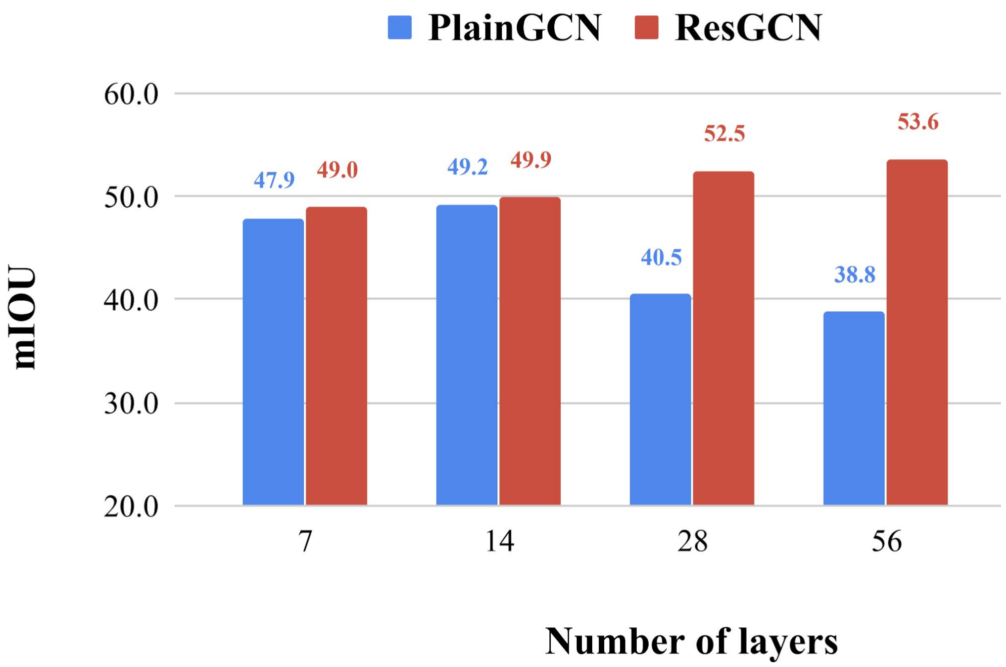
Class	DGCNN [6]	ResGCN-28 (*Ours*)
ceiling	92.7	**93.1**
floor	93.6	**95.3**
wall	77.5	**78.2**
beam	32.0	**33.9**
column	36.3	**37.4**
window	52.5	**56.1**
door	63.7	**68.2**
table	61.1	**64.9**
chair	60.2	**61.0**
sofa	20.5	**34.6**
bookcase	47.7	**51.5**
board	42.7	**51.1**
clutter	51.5	**54.4**
mIOU	56.3	**60.0**

~ 4% boost in mIOU.

Comparison of ResGCN-28 with DGCNN* (Our shallow baseline model).

* We reproduced the results of DGCNN on all classes since the results across all classes were not provided in the DGCNN paper.

PlainGCN VS. ResGCN



Oversmoothing Analysis

Model	Group Dis. Ratio	Instance Info. Gain	mIoU
ResGCN-28	1.73	0.46	52.49
w/o dilation	1.67	0.43	49.64
w/o connection	1.12	0.01	40.47

Analysis of over-smoothing using the Group Distance Ratio (intra group dist. / inter group dist.) and the Instance Information Gain (mutual information between input and final output).

Zhou, K., Huang, X., Li, Y., Zha, D., Chen, R. and Hu, X.. Towards deeper graph neural networks with differentiable group normalization. NeurIPS 2020.

Application in Biology

Wider

Deeper

Number of filters	32	64	128	256
PlainMRGCN-3	95.84	97.60	98.58	99.13
PlainMRGCN-7	97.35	98.69	99.22	**99.38**
PlainMRGCN-14	97.55	99.02	99.31	99.34
PlainMRGCN-28	98.09	99.00	99.02	99.31
PlainMRGCN-56	92.70	97.43	97.31	97.61
PlainMRGCN-112	60.75	71.97	89.69	91.50
ResMRGCN-3	96.04	97.60	98.53	99.09
ResMRGCN-7	97.00	98.43	99.19	99.30
ResMRGCN-14	97.75	98.88	99.26	**99.38**
ResMRGCN-28	98.50	99.16	99.29	**99.41**
ResMRGCN-56	98.62	99.27	**99.36**	**99.40**
ResMRGCN-112	98.41	99.34	**99.38**	**99.39**
DenseMRGCN-3	95.96	97.85	98.66	99.11
DenseMRGCN-7	97.87	98.47	99.31	**99.36**
DenseMRGCN-14	98.93	99.00	99.01	**99.43**
DenseMRGCN-28	99.16	99.29	**99.42**	-
DenseMRGCN-56	99.22	-	-	-

Node classification of biological networks.

Application in Biology

Model	m-F1 score (%)
GraphSAGE [42]	61.20
GATConv [43]	97.30
VR-GCN [57]	97.80
GaAN [58]	98.71
GeniePath [59]	98.50
Cluster-GCN [56]	99.36
ResMRGCN-28 (Ours)	**99.41**
DenseMRGCN-14 (Ours)	**99.43**

Comparison of DeepGCNs with state-of-the-art on PPI node classification.

Towards Structured Intelligence with Deep Graph Neural Networks

Making GCNs Go as

Deepergcn: All you need to train deeper gcns

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)

Latency Constraint

ICML
International Conference
On Machine Learning

2

Making GCNs Go as Deep as CNNs:

Message Aggregation Functions;
Memory Efficiency

4

Ongoing Work and Research Plan:

Structured Navigation;
Research Plan

Message Passing

Node Features

Neighbor's Features

Edge Features

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

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

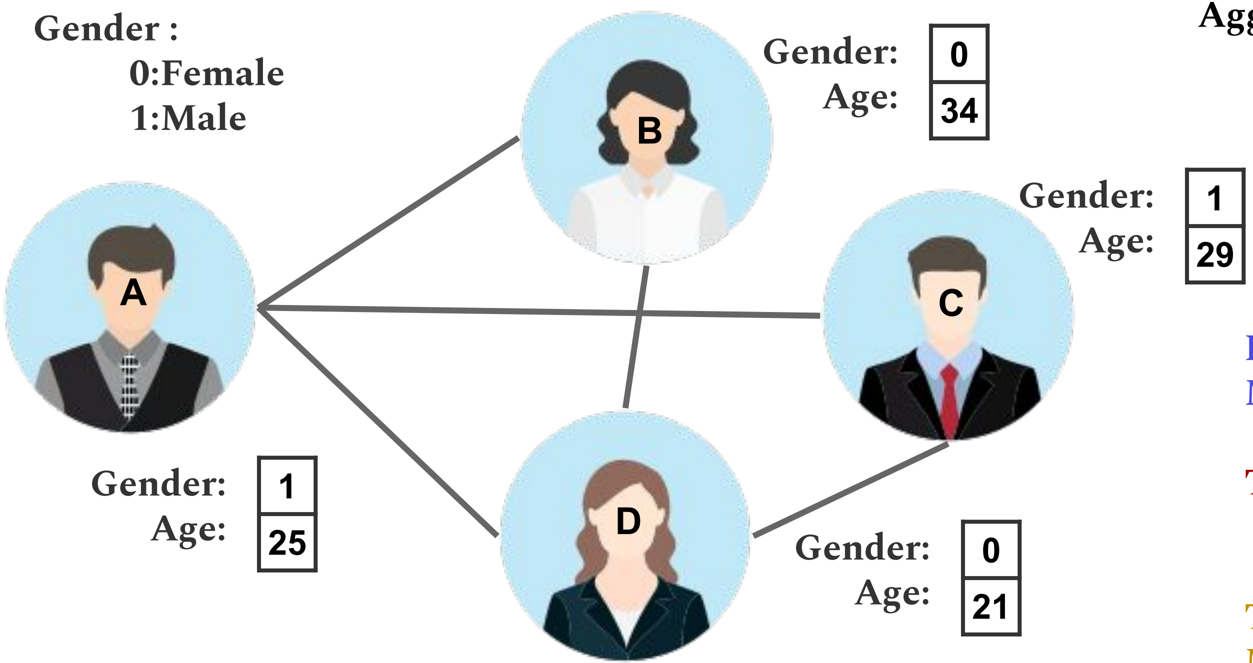
Permutation Invariant Function e.g., sum, mean or max

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

$\mathcal{N}_i$: neighbor indices

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

Aggregation Functions



Aggregation of neighbors of node A

Mean	Sum	Max
1/3	1	1
28	86	34

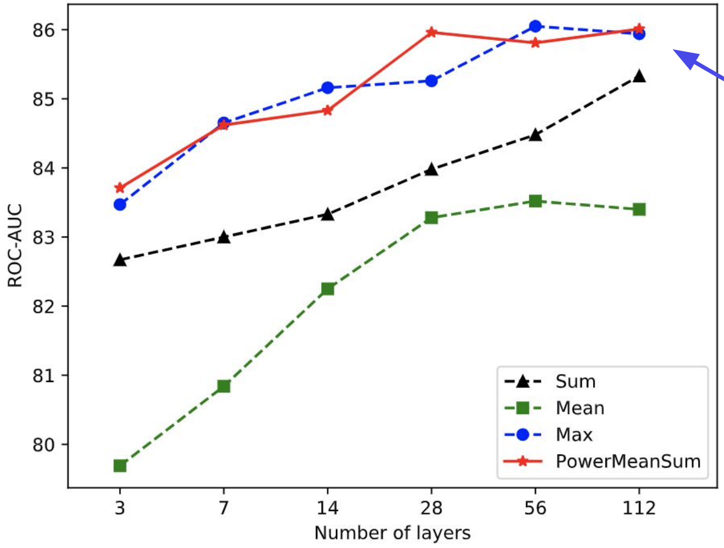
Percentage of male friends of A?
Mean: 1/3

The sum of ages of friends of A?
Sum: 86

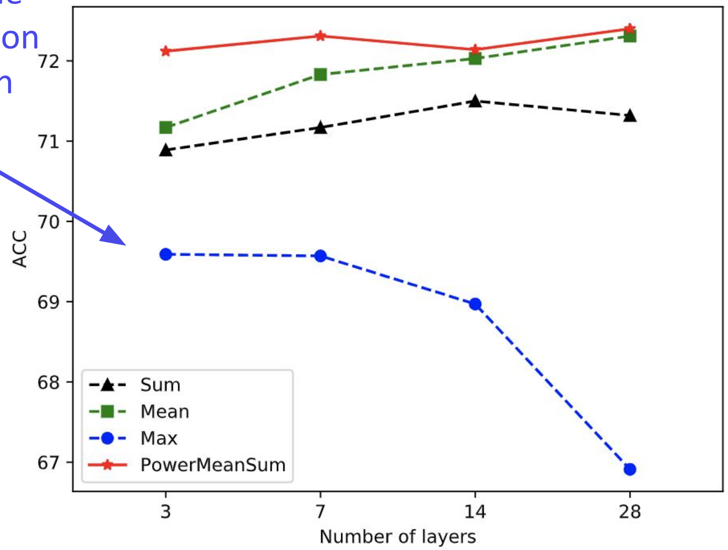
The eldest friend of A?
Max: 34

Different aggregations are good at capturing different properties of graphs.

Aggregation Functions



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



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

The same aggregation function

Aggregation functions perform very differently on different datasets.

Aggregation Functions

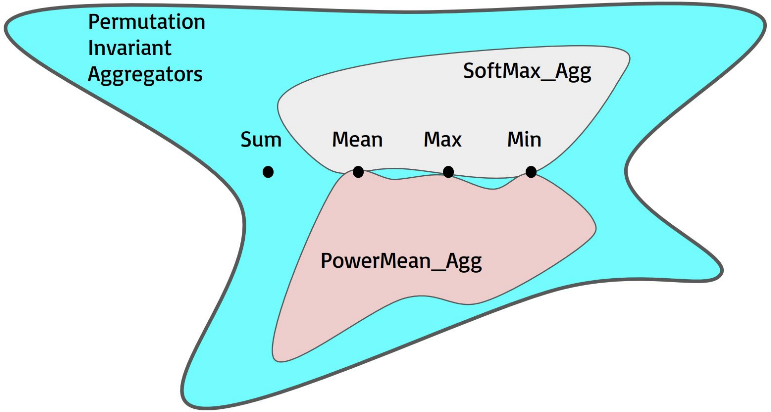


Illustration of Generalized Message Aggregation Functions.

Generalized mean-max aggregation function:

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

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

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

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

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

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

Aggregation Functions

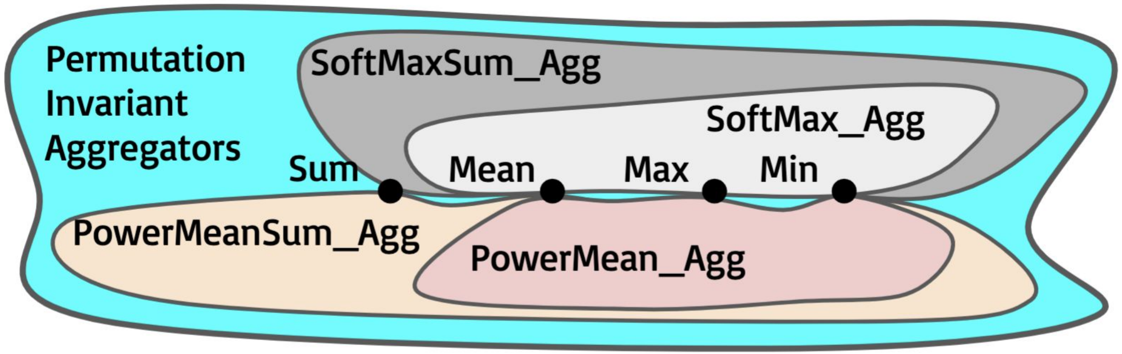


Illustration of Generalized Message Aggregation Functions.

Generalized mean-max aggregation function:

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

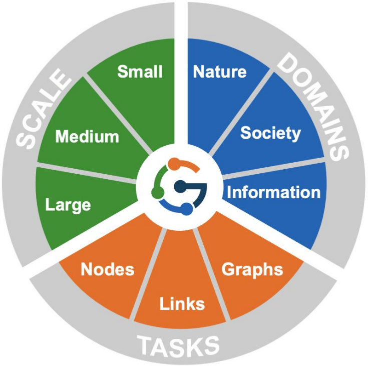
$$PowerMean\_Agg_{p}(\cdot) = \Big(\frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{m}_{vu}^{p}\Big)^{1/p}.$$

Generalized mean-max-sum aggregation function:

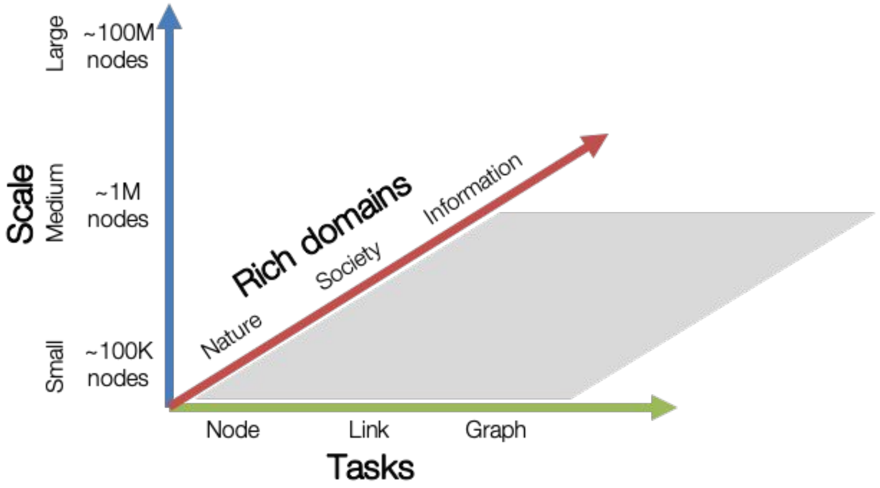
$$|\mathcal{N}(v)|^{y} \cdot \zeta_{x}(\cdot)$$

Differentiable aggregation functions

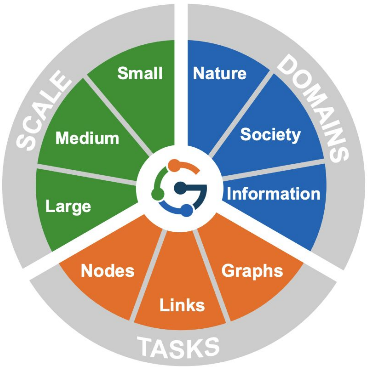
Datasets: Open Graph Benchmark (OGB)



OGB datasets

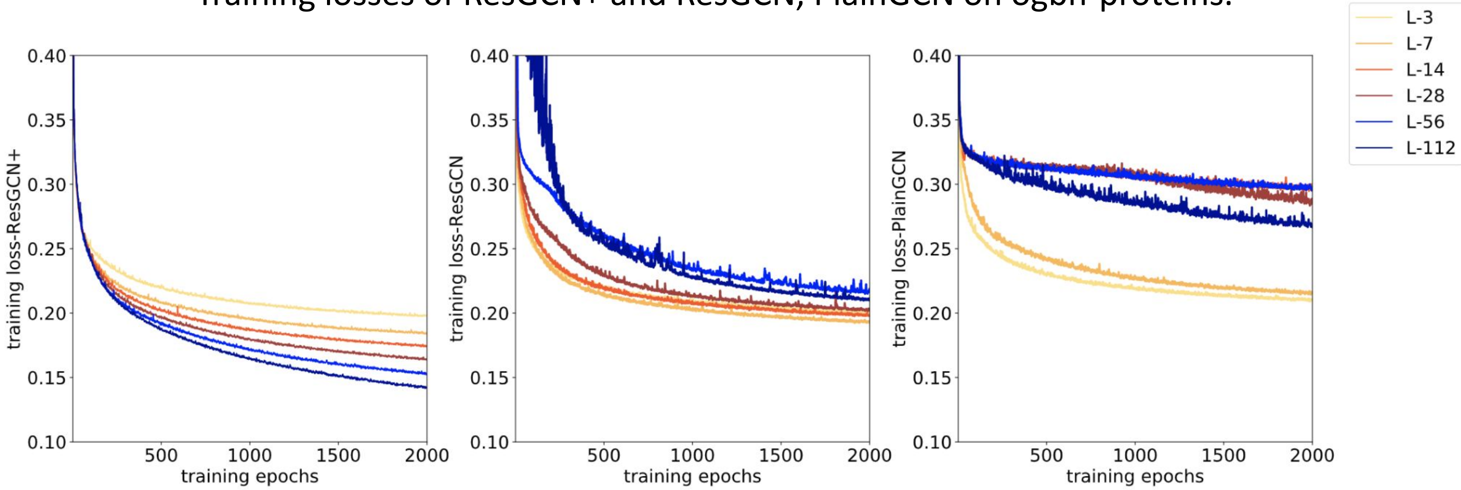


DeeperGCN - Residual Connections



OGB datasets

Training losses of ResGCN+ and ResGCN, PlainGCN on ogbn-proteins.



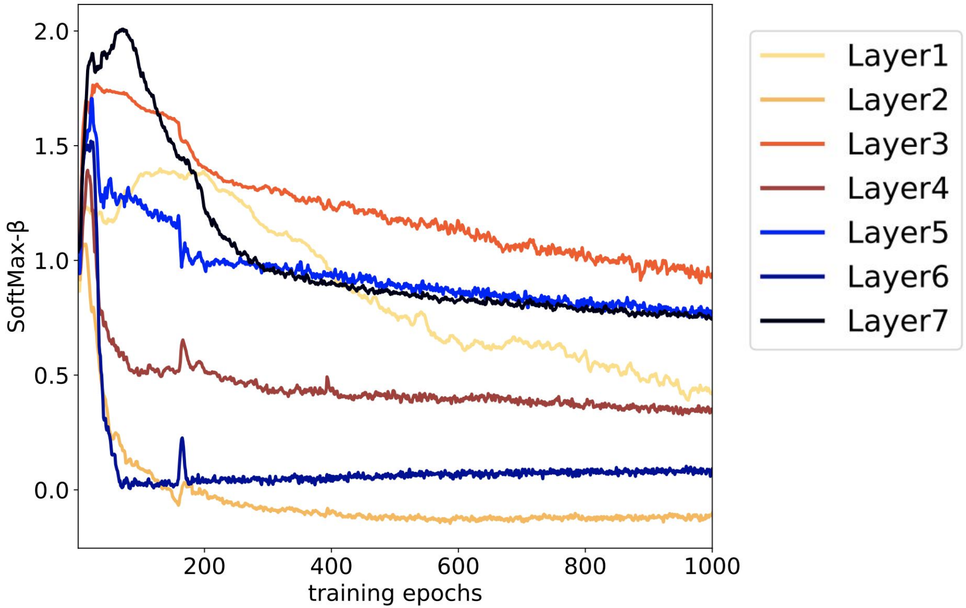
Preactivated residual connections work better.

Results

	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**

DeeperGCN achieves SOTA results on 6 OGB datasets.

Results



Learning curves of 7-layer DyResGEN with SoftMax_Agg(·).

Results

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:

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.

Related Work

The Reversible Residual Network: Backpropagation Without Storing Activations

Aidan N. Gomez*[1], Mengye Ren*[1,2,3], Raquel Urtasun[1,2,3], Roger B. Grosse[1,2]
University of Toronto[1]
Vector Institute for Artificial Intelligence[2]
Uber Advanced Technologies Group[3]
{aidan, mren, urtasun, rgrosse}@cs.toronto.edu

Deep Equilibrium Models

Shaojie Bai
Carnegie Mellon University

J. Zico Kolter
Carnegie Mellon University
Bosch Center for AI

Vladlen Koltun
Intel Labs

DNN: **O(L)**

Reversible CNN / DEQ: **O(1)**

*only consider the L dimension

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)

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:

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

$$X'_0 = \sum_{i=2}^{C} X_i$$

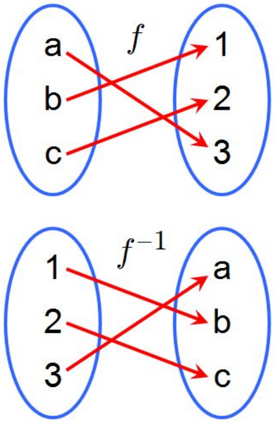
$$X_1 = X'_1 - f_{w_1}(X'_0, A, U).$$

When #group =2:

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

Forward:

$$X'_0 = X_2$$
$$X'_1 = f_{w_1}(X'_0, A, U) + X_1$$
$$X'_2 = f_{w_2}(X'_1, A, U) + X_2$$



Inverse:

$$X_2 = X'_2 - f_{w_2}(X'_1, A, U)$$
$$X'_0 = X_2$$
$$X_1 = X'_1 - f_{w_1}(X'_0, A, U).$$

Memory Efficient GNNs

DEQ-GNN:

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

Forward:

Fixed Point Iteration

Backward:

Implicit Differentiation

Memory Efficient GNNs

DEQ-GNN:

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

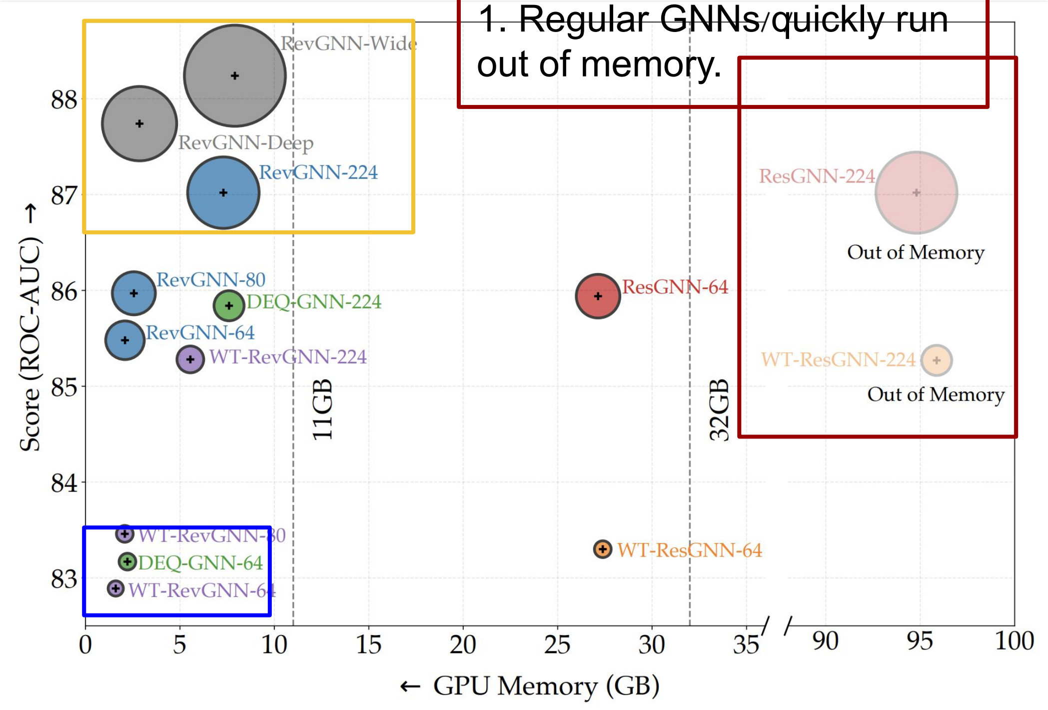
$$\frac{\partial \ell}{\partial (\cdot)} = -\frac{\partial \ell}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial (\cdot)} = -\frac{\partial \ell}{\partial h} \frac{\partial h}{\partial \mathbf{z}^\star_{1:T}} \left(J^{-1}_{g_\theta} \big|_{\mathbf{z}^\star_{1:T}} \right) \frac{\partial f_\theta(\mathbf{z}^\star_{1:T}; \mathbf{x}_{1:T})}{\partial (\cdot)},$$

Results - Summary

2. We can train huge overparameterized RevGNNs on a single GPU and achieve the best performance.

3. We can train smaller GNNs with weight-tying or DEQ and still reach promising results

1. Regular GNNs quickly run out of memory.

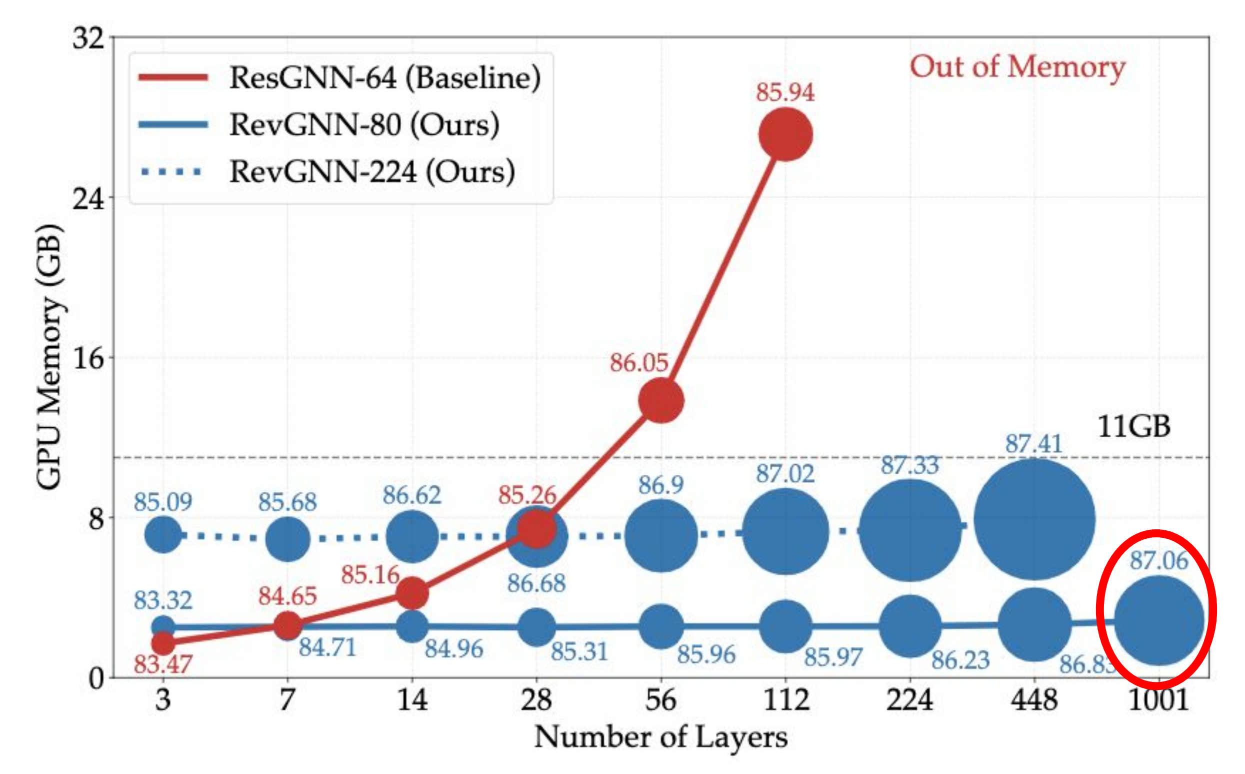


Performance v.s. GPU memory consumption on the ogbn-proteins dataset for 112 layer deep networks.

Results - Complexity Analysis

Method	Memory	Params	Time
Full-batch GNN	$\mathcal{O}(LND)$	$\mathcal{O}(LD^2)$	$\mathcal{O}(L\,\Vert A\Vert_0\,D + LND^2)$
GraphSAGE	$\mathcal{O}(R^LBD)$	$\mathcal{O}(LD^2)$	$\mathcal{O}(R^LND^2)$
VR-GCN	$\mathcal{O}(LND)$	$\mathcal{O}(LD^2)$	$\mathcal{O}(L\,\Vert A\Vert_0\,D + LND^2 + R^LND^2)$
FastGCN	$\mathcal{O}(LRBD)$	$\mathcal{O}(LD^2)$	$\mathcal{O}(RLND^2)$
Cluster-GCN	$\mathcal{O}(LBD)$	$\mathcal{O}(LD^2)$	$\mathcal{O}(L\,\Vert A\Vert_0\,D + LND^2)$
GraphSAINT	$\mathcal{O}(LBD)$	$\mathcal{O}(LD^2)$	$\mathcal{O}(L\,\Vert A\Vert_0\,D + LND^2)$
Weight-tied GNN	$\mathcal{O}(LND)$	$\mathcal{O}(D^2)$	$\mathcal{O}(L\,\Vert A\Vert_0\,D + LND^2)$
RevGNN	$\mathcal{O}(ND)$	$\mathcal{O}(LD^2)$	$\mathcal{O}(L\,\Vert A\Vert_0\,D + LND^2)$
WT-RevGNN	$\mathcal{O}(ND)$	$\mathcal{O}(D^2)$	$\mathcal{O}(L\,\Vert A\Vert_0\,D + LND^2)$
DEQ-GNN	$\mathcal{O}(ND)$	$\mathcal{O}(D^2)$	$\mathcal{O}(K\,\Vert A\Vert_0\,D + KND^2)$
RevGNN + Subgraph Sampling	$\mathcal{O}(BD)$	$\mathcal{O}(LD^2)$	$\mathcal{O}(L\,\Vert A\Vert_0\,D + LND^2)$
WT-RevGNN + Subgraph Sampling	$\mathcal{O}(BD)$	$\mathcal{O}(D^2)$	$\mathcal{O}(L\,\Vert A\Vert_0\,D + LND^2)$
DEQ-GNN + Subgraph Sampling	$\mathcal{O}(BD)$	$\mathcal{O}(D^2)$	$\mathcal{O}(K\,\Vert A\Vert_0\,D + KND^2)$

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)

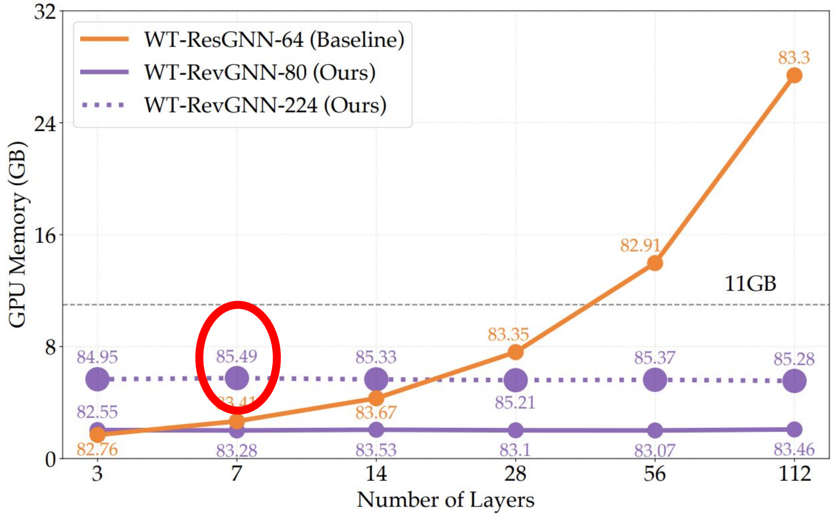
Rank	Method	Test ROC-AUC	Validation ROC-AUC	Contact	References	#Params	Hardware	Date
1	**RevGNN-Wide**	0.8824 ± 0.0015	0.9450 ± 0.0008	Guohao Li - DeepGCNs.org	Paper, Code	68,471,608	NVIDIA RTX 6000 (48G)	Jun 16, 2021
2	**RevGNN-Deep**	0.8774 ± 0.0013	0.9326 ± 0.0006	Guohao Li - DeepGCNs.org	Paper, Code	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)

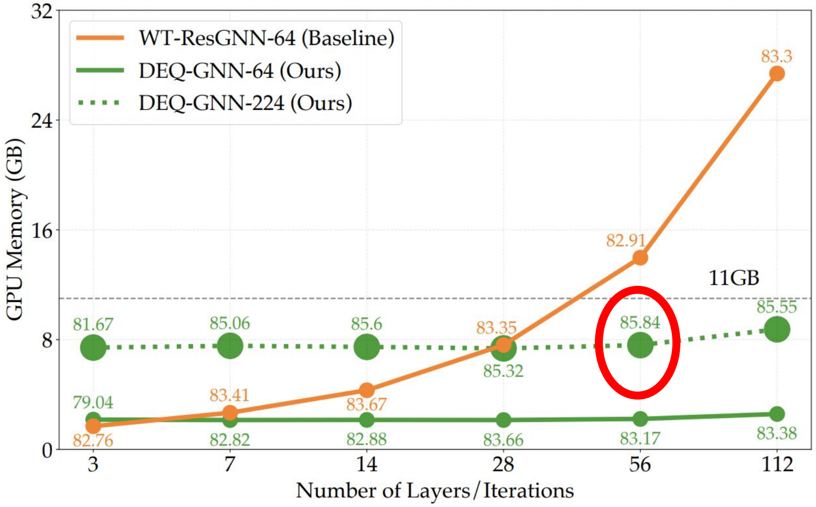
Results - SOTA with RevGNN (ogbn-arxiv)

Rank	Method	Test Accuracy	Validation Accuracy	Contact	References	#Params	Hardware	Date
1	**RevGAT+N.Adj+LabelReuse+SelfKD**	0.7426 ± 0.0017	0.7497 ± 0.0008	Guohao Li - DeepGCNs.org	Paper, Code	2,098,256	NVIDIA Tesla V100 (32GB GPU)	Jun 21, 2021
2	GAT+label reuse+self KD	0.7416 ± 0.0008	0.7514 ± 0.0004	Shunli Ren(CMIC@SJTU)	Paper, Code	1,441,580	GeForce RTX 1080Ti (11GB GPU)	Dec 15, 2020
3	**RevGAT+NormAdj+LabelReuse**	0.7402 ± 0.0018	0.7501 ± 0.0010	Guohao Li - DeepGCNs.org	Paper, Code	2,098,256	NVIDIA Tesla V100 (32GB GPU)	Jun 21, 2021
4	GAT+label+reuse+topo loss	0.7399 ± 0.0012	0.7513 ± 0.0009	Mengyang Niu (DAMO DI)	Paper, Code	1,441,580	Tesla V100 (16GB)	Dec 10, 2020
5	**AGDN (GAT-HA+3_heads+labels)**	0.7398 ± 0.0009	0.7519 ± 0.0009	Chuxiong Sun	Paper, Code	1,508,555	Tesla V100 (32GB GPU)	Jan 3, 2021
6	**UniMP_v2**	0.7397 ± 0.0015	0.7506 ± 0.0009	Weiyue Su (PGL Team)	Paper, Code	687,377	Tesla V100 (32GB)	Nov 24, 2020
7	GAT(norm.adj.)+label reuse+C&S	0.7395 ± 0.0012	0.7519 ± 0.0008	Yangkun Wang (DGL Team)	Paper, Code	1,441,580	p3.8xlarge (15GB GPU)	Nov 24, 2020
8	GAT+norm. adj.+label reuse	0.7391 ± 0.0012	0.7516 ± 0.0008	Yangkun Wang (DGL Team)	Paper, Code	1,441,580	p3.8xlarge (15GB GPU)	Nov 11, 2020
9	**GAT + C&S**	0.7386 ± 0.0014	0.7484 ± 0.0007	Horace He (Cornell)	Paper, Code	1,567,000	GeForce RTX 2080 (11GB GPU)	Oct 27, 2020

Results - Constant Memory and Parameter Complexities



WT-RevGNN.



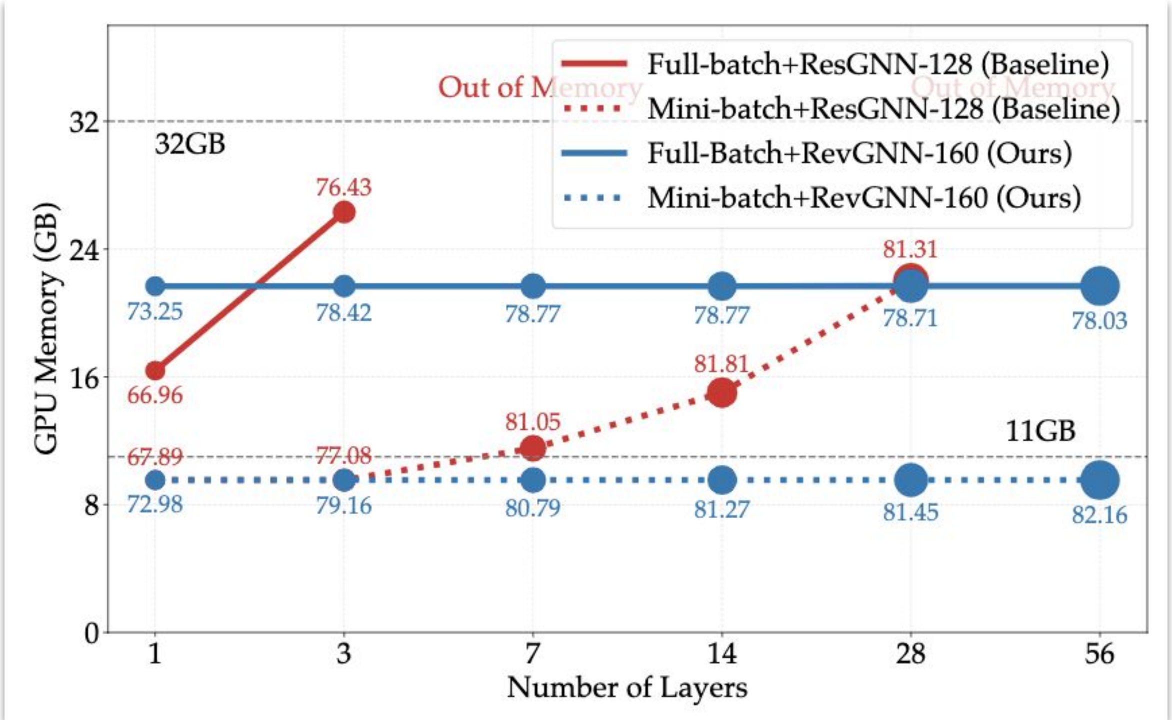
DEQ-RevGNN.

Ablation - Different GNN operators (ogbn-arxiv)

Model	#L	#Ch	ACC ↑	Mem ↓	Params
ResGCN	28	128	72.46 ± 0.29	11.15	491k
RevGCN	28	128	73.01 ± 0.31	**1.84**	262k
RevGCN	28	180	**73.22** ± 0.19	2.73	500k
ResSAGE	28	128	72.46 ± 0.29	8.93	950k
RevSAGE	28	128	72.69 ± 0.23	**1.17**	491k
RevSAGE	28	180	**72.73** ± 0.10	1.57	953k
ResGEN	28	128	72.32 ± 0.27	21.63	491k
RevGEN	28	128	72.34 ± 0.18	**4.08**	262k
RevGEN	28	180	**72.93** ± 0.10	5.67	500k
ResGAT	5	768	73.76 ± 0.13	9.96	3.87M
RevGAT	5	768	74.02 ± 0.18	**6.30**	2.10M
RevGAT	5	1068	**74.05** ± 0.11	8.49	3.88M

RevGNNs are generic and can be applied to different operators.

Ablation - Mini-batch Training (ogbn-products)



Mini-batch training further reduces the memory consumption of RevGNN and improves its accuracy.

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.

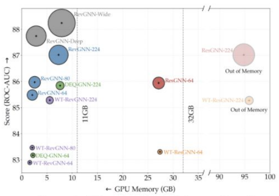


Figure: RevGNNs outperform existing models with significantly less memory consumption.

intel labs KAUST

67 Google Slides

Towards Structured Intelligence with Deep Graph Neural Networks

Making GCNs Go as Deep as CNNs:
Skip Connections and Dilated Convolutions on Graphs

1

2

Func

Mem

SGAS: Sequential Greedy Architecture Search

Authors	Guohao Li, Guocheng Qian, Itzel C Delgadillo, Matthias Müller, Ali Thabet, Bernard Ghanem
Publication date	2020
Journal	Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

CVPR SEATTLE WASHINGTON JUNE 16-18 2020

Automate GNN Architecture Design:
Sequential Greedy Architecture Search;
Latency Constraint

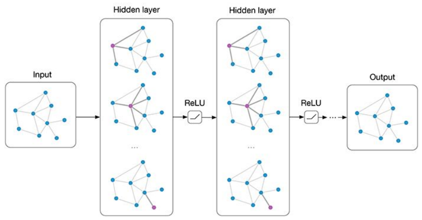
3

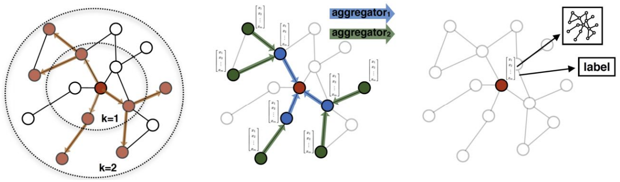
LC-NAS: Latency constrained neural architecture search for point cloud networks

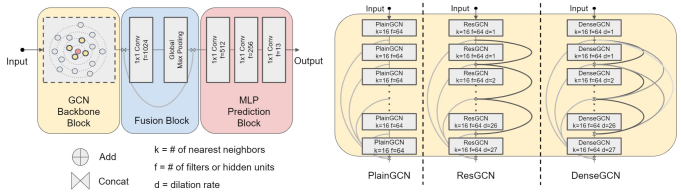
Authors	Guohao Li, Mengmeng Xu, Silvio Giancola, Ali Thabet, Bernard Ghanem
Publication date	2022/9/12
Journal	International Conference on 3D Vision & IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops - Third workshop on Neural Architecture Search Second lightweight NAS challenge

3DV 2022

Automate GNN Architecture Design



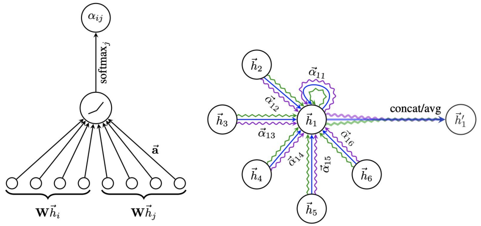


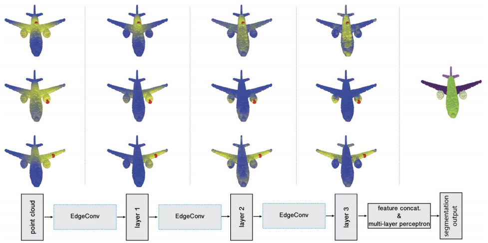


Li, G., Müller, M., Thabet, A. and Ghanem, B., 2019. DeepGCNs: Can GCNs Go as Deep as CNNs?

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.





Designing GNNs is Painful!

A Smarter Way?

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.

SGAS: Sequential Greedy Architecture Search

Search Ranking	Evaluation Ranking: DARTS (1st Order)	DARTS (2nd Order)	SGAS (Criterion 1)	SGAS (Criterion 2)
1	4	8	2	4
2	2	5	3	6
3	7	9	1	1
4	8	6	6	2
5	1	4	8	3
6	5	3	7	10
7	10	7	4	5
8	9	1	5	8
9	6	10	10	9
10	3	2	9	7
Kendall τ	0.16	-0.29	0.56	0.42
Avg. Acc.	97.15	97.18	97.34	97.33

Comparison of search-evaluation Kendall coefficients.

Architectures with a higher validation accuracy during the search phase may perform worse in the evaluation (see Figure 1).

SGAS: Sequential Greedy Architecture Search

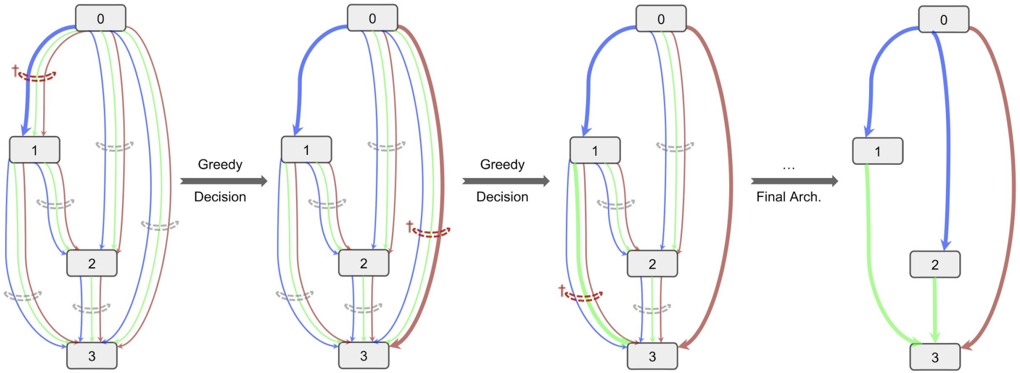


Illustration of Sequential Greedy Architecture Search.

Aiming to alleviate this common issue, we introduce sequential greedy architecture search (SGAS), an efficient method for neural architecture search.

By dividing the search procedure into sub-problems, SGAS chooses and prunes candidate operations in a greedy fashion.

SGAS: Sequential Greedy Architecture Search

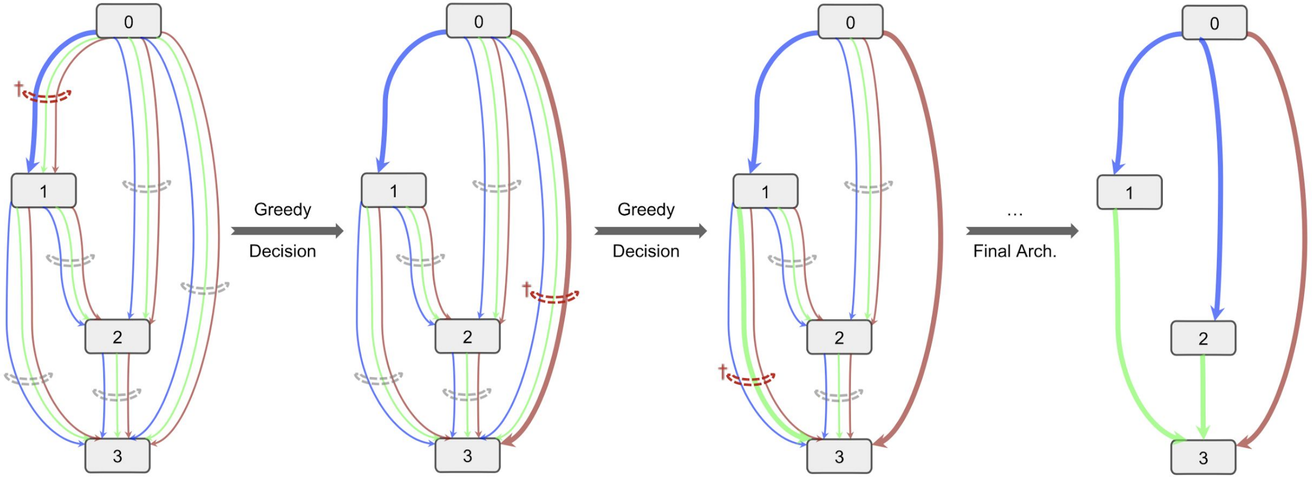


Illustration of Sequential Greedy Architecture Search.

SGAS: Sequential Greedy Architecture Search

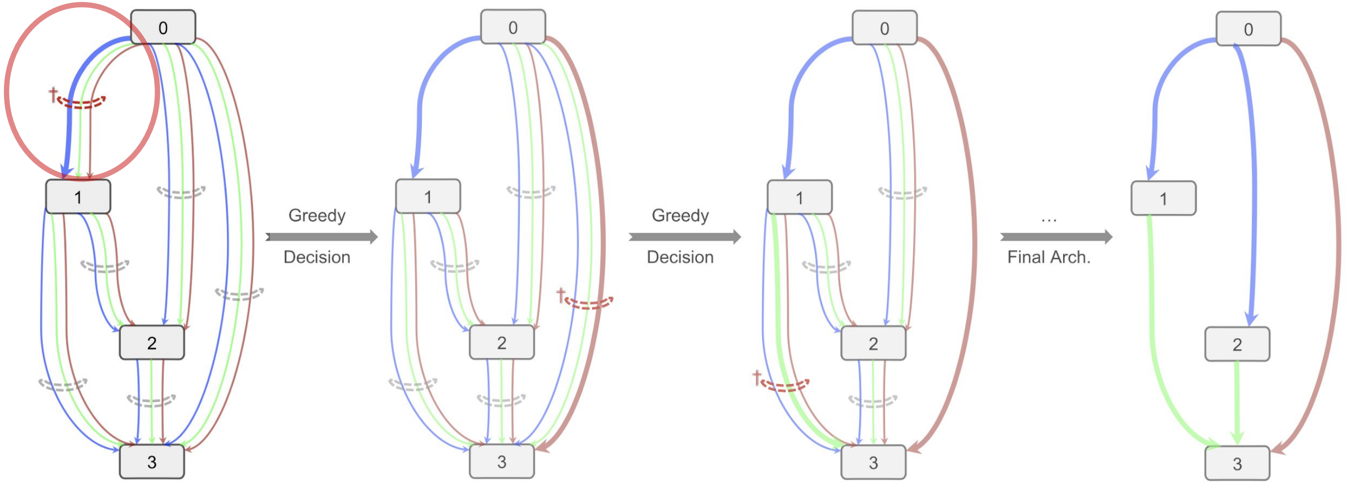


Illustration of Sequential Greedy Architecture Search.

1 If a decision epoch, select an edge $(i^\dagger, j^\dagger)$ based on the greedy *Selection Criterion*

SGAS: Sequential Greedy Architecture Search

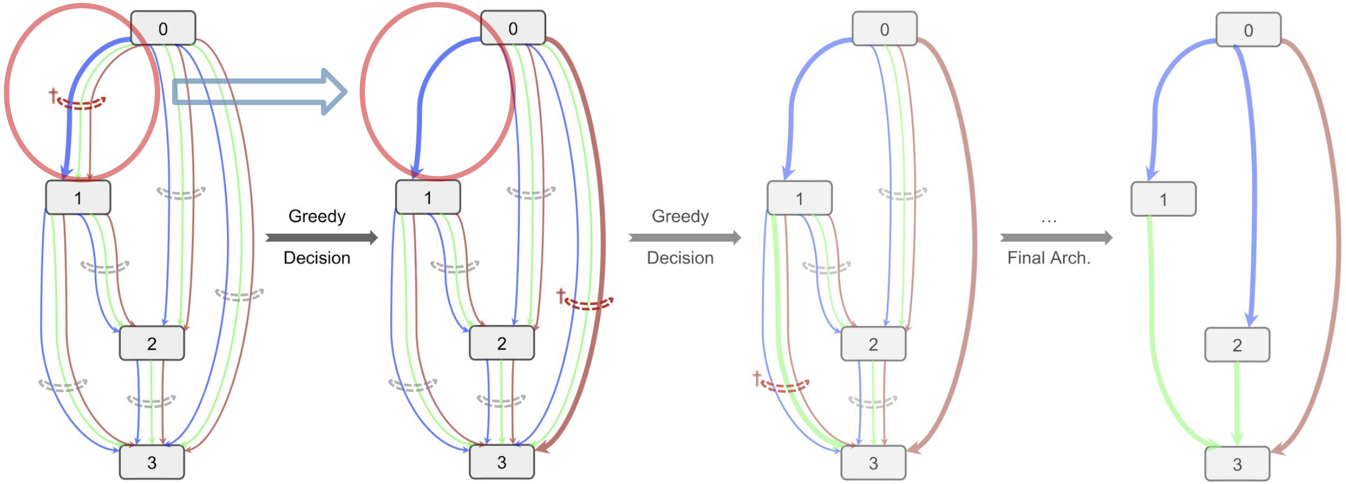


Illustration of Sequential Greedy Architecture Search.

1. If a decision epoch, select an edge $(i^\dagger, j^\dagger)$ based on the greedy *Selection Criterion*
2. Determine the operation by replacing $\bar{o}^{(i^\dagger, j^\dagger)}$ with $o^{(i^\dagger, j^\dagger)} = \text{argmax}_{o \in \mathcal{O}} \ \alpha_o^{(i^\dagger, j^\dagger)}$

SGAS: Sequential Greedy Architecture Search

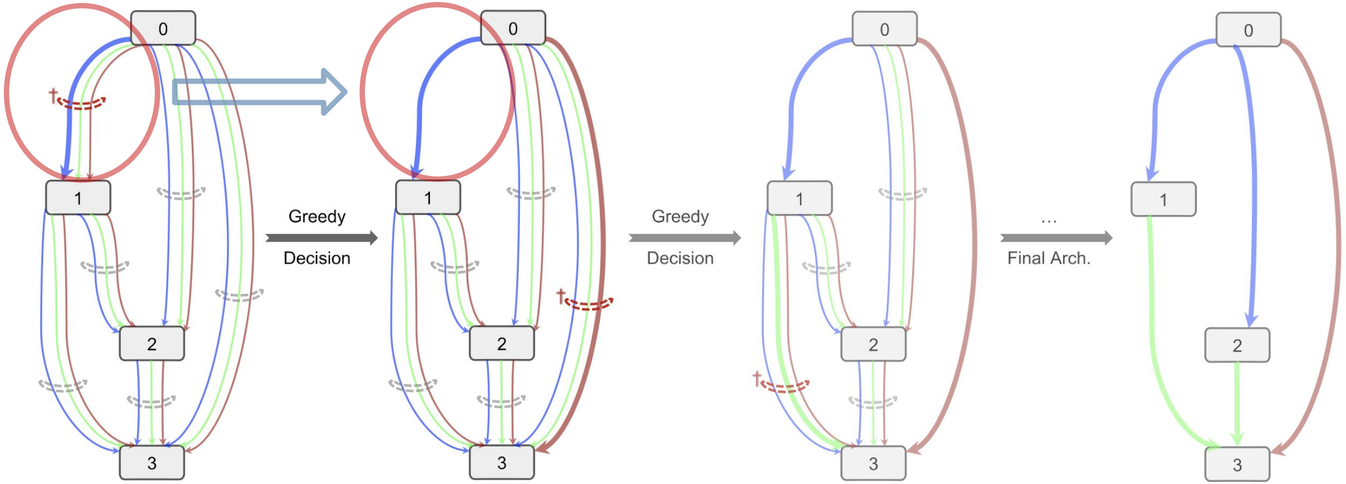


Illustration of Sequential Greedy Architecture Search.

1. If a decision epoch, select an edge $(i^\dagger, j^\dagger)$ based on the greedy *Selection Criterion*
2. Determine the operation by replacing $\bar{o}^{(i^\dagger, j^\dagger)}$ with $o^{(i^\dagger, j^\dagger)} = \operatorname{argmax}_{o \in \mathcal{O}} \ \alpha_o^{(i^\dagger, j^\dagger)}$
3. Prune unchosen weights from $\mathcal{W}$, Remove $\alpha^{(i^\dagger, j^\dagger)}$ from $\mathcal{A}$

SGAS: Sequential Greedy Architecture Search

Repeat…

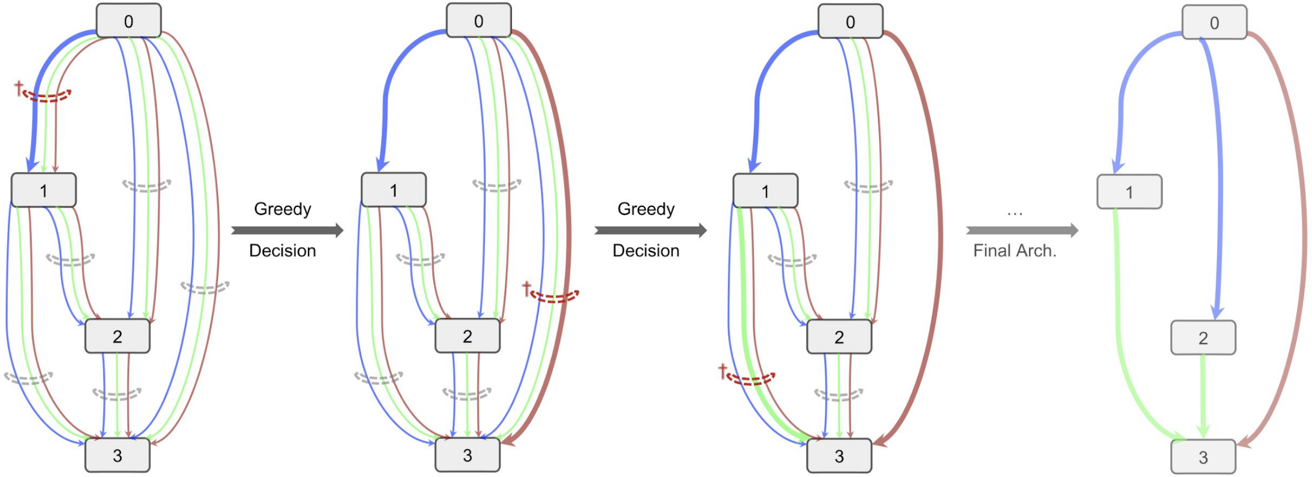
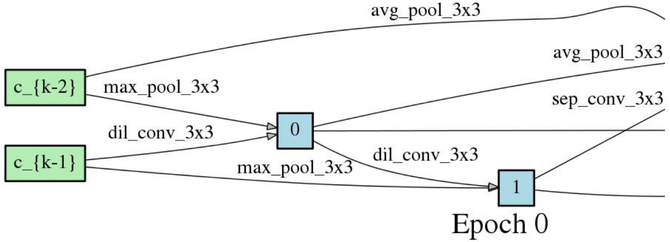


Illustration of Sequential Greedy Architecture Search.

1. If a decision epoch, select an edge $(i^\dagger, j^\dagger)$ based on the greedy *Selection Criterion*
2. Determine the operation by replacing $\bar{o}^{(i^\dagger, j^\dagger)}$ with $o^{(i^\dagger, j^\dagger)} = \mathrm{argmax}_{o \in \mathcal{O}} \ \alpha_o^{(i^\dagger, j^\dagger)}$
3. Prune unchosen weights from $\mathcal{W}$, Remove $\alpha^{(i^\dagger, j^\dagger)}$ from $\mathcal{A}$

SGAS - Selection Criteria

To maintain the optimality, the design of the selection criterion is crucial.



Edge Importance:

$$S_{EI}^{(i,j)} = \sum_{o \in \mathcal{O}, o \neq zero} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}$$

Selection Certainty:

$$p_o^{(i,j)} = \frac{\exp(\alpha_o^{(i,j)})}{S_{EI}^{(i,j)} \sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}, o \in \mathcal{O}, o \neq zero$$

$$S_{SC}^{(i,j)} = 1 - \frac{-\sum_{o \in \mathcal{O}, o \neq zero} p_o^{(i,j)} \log(p_o^{(i,j)})}{\log(|\mathcal{O}| - 1)}$$

Selection Stability:

$$S_{SS}^{(i,j)} = \frac{1}{K} \sum_{t=T-K}^{T-1} \sum_{o_t \in \mathcal{O} o_t, \neq zero} \min(p_{o_t}^{(i,j)}, p_{o_T}^{(i,j)})$$

SGAS - Selection Criteria

Criterion 1:

$$S_1^{(i,j)} = \text{normalize}(S_{EI}^{(i,j)}) * \text{normalize}(S_{SC}^{(i,j)})$$

Criterion 2:

$$S_2^{(i,j)} = S_1^{(i,j)} * \text{normalize}(S_{SS}^{(i,j)})$$

$normalize(\cdot)$: a standard Min-Max scaling normalization

Edge Importance:

$$S_{EI}^{(i,j)} = \sum_{o \in \mathcal{O}, o \neq zero} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}$$

Selection Certainty:

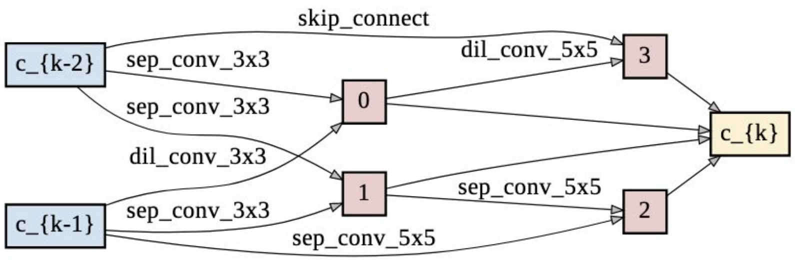
$$p_o^{(i,j)} = \frac{\exp(\alpha_o^{(i,j)})}{S_{EI}^{(i,j)} \sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})}, o \in \mathcal{O}, o \neq zero$$

$$S_{SC}^{(i,j)} = 1 - \frac{-\sum_{o \in \mathcal{O}, o \neq zero} p_o^{(i,j)} \log(p_o^{(i,j)})}{\log(|\mathcal{O}| - 1)}$$

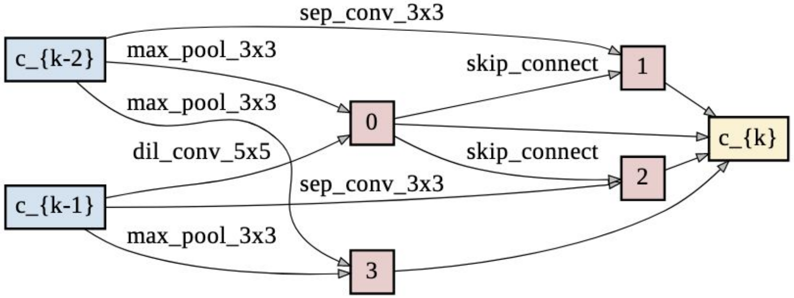
Selection Stability:

$$S_{SS}^{(i,j)} = \frac{1}{K} \sum_{t=T-K}^{T-1} \sum_{o_t \in \mathcal{O} o_t, \neq zero} \min(p_{o_t}^{(i,j)}, p_{o_T}^{(i,j)})$$

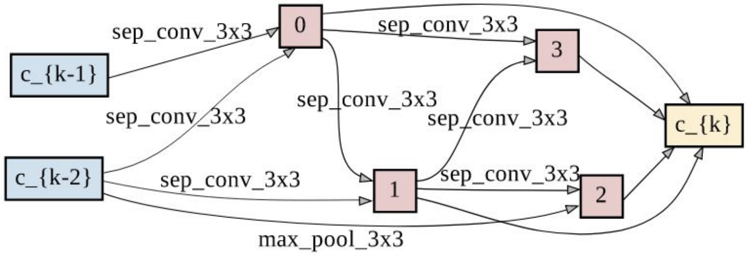
Results – SGAS for CNN on CIFAR-10



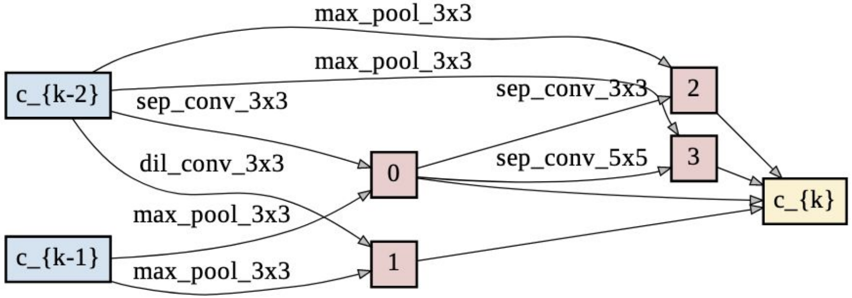
(a) Normal cell of the best model with SGAS (Cri. 1) on CIFAR-10



(b) Reduction cell of the best model with SGAS (Cri. 1) on CIFAR-10



(c) Normal cell of the best model with SGAS (Cri. 2) on CIFAR-10



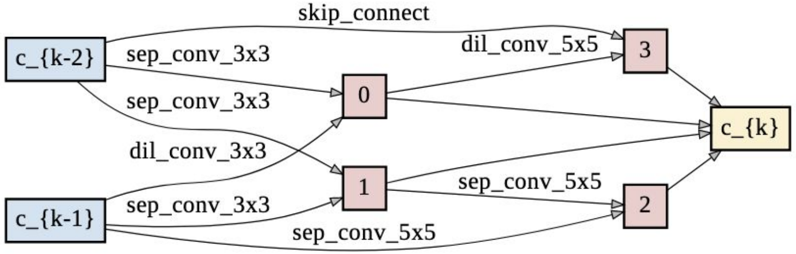
(d) Reduction cell of the best model with SGAS (Cri. 2) on CIFAR-10

Results – SGAS for CNN on CIFAR-10

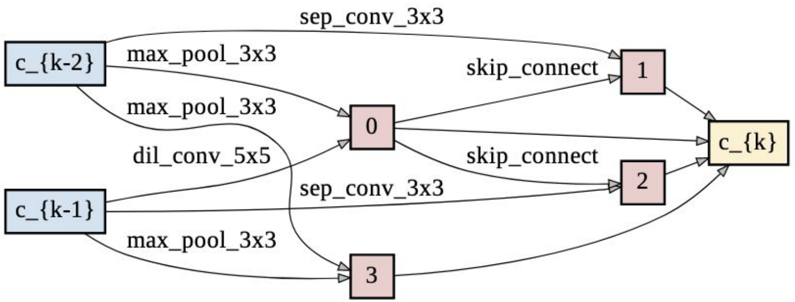
Architecture	Test Err. (%)	Params (M)	Search Cost (GPU-days)	Search Method
DenseNet-BC [18]	3.46	25.6	-	manual
NASNet-A [55]	2.65	3.3	1800	RL
AmoebaNet-A [36]	3.34±0.06	3.2	3150	evolution
AmoebaNet-B [36]	2.55±0.05	2.8	3150	evolution
Hier-Evolution [28]	3.75±0.12	15.7	300	evolution
PNAS [27]	3.41±0.09	3.2	225	SMBO
ENAS [34]	2.89	4.6	0.5	RL
NAONet-WS [31]	3.53	3.1	0.4	NAO
DARTS (1^{st} order) [29]	3.00±0.14	3.3	0.4	gradient
DARTS (2^{nd} order) [29]	2.76±0.09	3.3	1	gradient
SNAS (mild) [49]	2.98	2.9	1.5	gradient
ProxylessNAS [7]	2.08	-	4	gradient
P-DARTS [8]	2.5	3.4	0.3	gradient
BayesNAS [52]	2.81±0.04	3.4	0.2	gradient
PC-DARTS [50]	2.57±0.07	3.6	0.1	gradient
SGAS (Cri.1 avg.)	2.66±0.24*	3.7	0.25	gradient
SGAS (Cri.1 best)	2.39	3.8	0.25	gradient
SGAS (Cri.2 avg.)	2.67±0.21*	3.9	0.25	gradient
SGAS (Cri.2 best)	2.44	4.1	0.25	gradient

Performance comparison with state-of-the-art image classifiers on CIFAR-10.

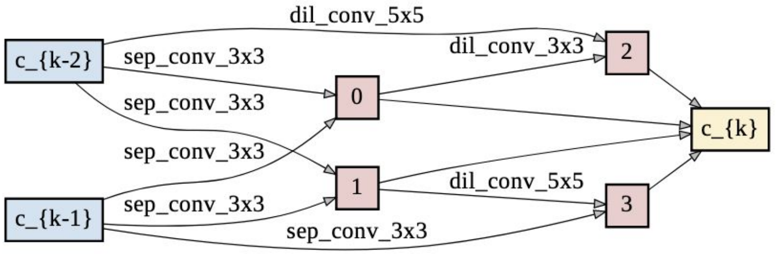
Results – SGAS for CNN on ImageNet



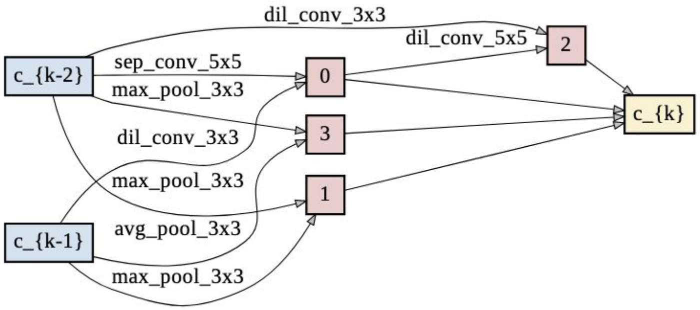
(a) Normal cell of the best model with SGAS (Cri. 1) on ImageNet



(b) Reduction cell of the best model with SGAS (Cri. 1) on ImageNet



(c) Normal cell of the best model with SGAS (Cri. 2) on ImageNet



(d) Reduction cell of the best model with SGAS (Cri. 2) on ImageNet

Results – SGAS for CNN on ImageNet

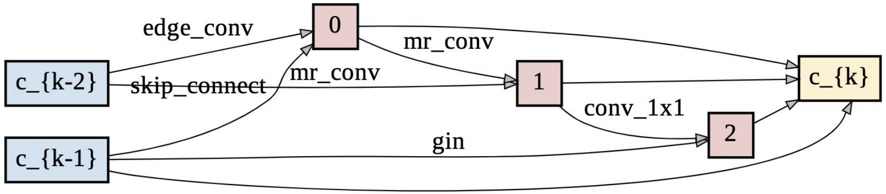
Architecture	Test Err. (%) top-1	Test Err. (%) top-5	Params (M)	$\times+$ (M)	Search Cost (GPU-days)	Search Method
Inception-v1 [41]	30.2	10.1	6.6	1448	-	manual
MobileNet [16]	29.4	10.5	4.2	569	-	manual
ShuffleNet 2x (v1) [51]	26.4	10.2	$\sim$5	524	-	manual
ShuffleNet 2x (v2) [32]	25.1	-	$\sim$5	591	-	manual
NASNet-A [55]	26	8.4	5.3	564	1800	RL
NASNet-B [55]	27.2	8.7	5.3	488	1800	RL
NASNet-C [55]	27.5	9	4.9	558	1800	RL
AmoebaNet-A [36]	25.5	8	5.1	555	3150	evolution
AmoebaNet-B [36]	26	8.5	5.3	555	3150	evolution
AmoebaNet-C [36]	24.3	7.6	6.4	570	3150	evolution
PNAS [27]	25.8	8.1	5.1	588	225	SMBO
MnasNet-92 [42]	25.2	8	4.4	388	-	RL
DARTS (2^{nd} order) [29]	26.7	8.7	4.7	574	4.0	gradient
SNAS (mild) [49]	27.3	9.2	4.3	522	1.5	gradient
ProxylessNAS [7]	24.9	7.5	7.1	465	8.3	gradient
P-DARTS [8]	24.4	7.4	4.9	557	0.3	gradient
BayesNAS [52]	26.5	8.9	3.9	-	0.2	gradient
PC-DARTS [50]	25.1	7.8	5.3	586	0.1	gradient
SGAS (Cri.1 avg.)	24.4 ± 0.2	7.3 ± 0.1	5.3	579	0.25	gradient
SGAS (Cri.1 best)	24.2	7.2	5.3	585	0.25	gradient
SGAS (Cri.2 avg.)	24.4 ± 0.2	7.4 ± 0.1	5.4	597	0.25	gradient
SGAS (Cri.2 best)	**24.1**	7.3	5.4	598	0.25	gradient

Performance comparison with state-of-the-art image classifiers on ImageNet.

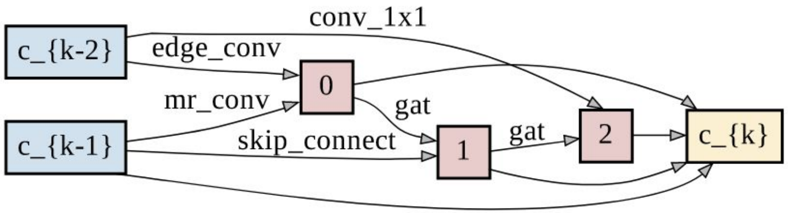
Results – SGAS for GCN on ModelNet

Architecture	OA (%)	Params (M)	Search Cost (GPU-days)
3DmFV-Net [3]	91.6	45.77	manual
SpecGCN [46]	91.5	2.05	manual
PointNet++ [37]	90.7	1.48	manual
PCNN [2]	92.3	8.2	manual
PointCNN [25]	92.2	0.6	manual
DGCNN [47]	92.2	1.84	manual
KPConv [44]	92.9	14.3	manual
Random Search	92.65±0.33	8.77	random
SGAS (Cri.1 avg.)	92.69±0.20	8.78	0.19
SGAS (Cri.1 best)	92.87	8.63	0.19
SGAS (Cri.2 avg.)	92.92±0.19	8.87	0.19
SGAS (Cri.2 best)	**93.23**	8.49	0.19
SGAS (Cri.2 small best)	93.07	3.86	0.19

Comparison with state-of-the-art architectures for 3D object classification on ModelNet40.



(a) Normal cell of the best model with SGAS (Cri. 1) on ModelNet

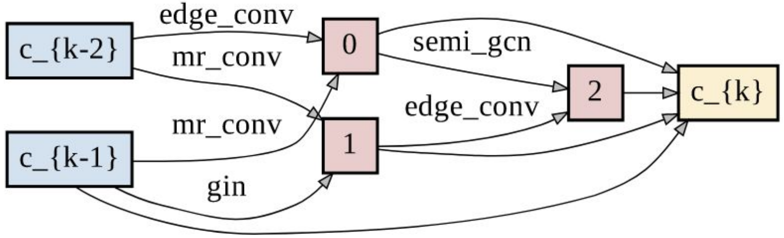


(b) Normal cell of the best model with SGAS (Cri. 2) on ModelNet

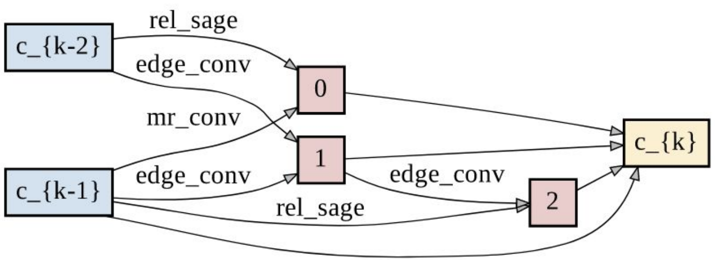
Results – SGAS for GCN on PPI

Architecture	micro-F1 (%)	Params (M)	Search Cost (GPU-days)
GraphSAGE (LSTM) [14]	61.2	0.26	manual
GeniePath [30]	97.9	1.81	manual
GAT [44]	97.3±0.2	3.64	manual
DenseMRGCN-14 [23]	99.43	53.42	manual
ResMRGCN-28 [23]	99.41	14.76	manual
Random Search	99.36±0.04	23.70	random
SGAS (Cri.1 avg.)	99.38±0.17	25.01	0.003
SGAS (Cri.1 best)	**99.46**	23.18	0.003
SGAS (Cri.2 avg.)	99.40±0.09	25.93	0.003
SGAS (Cri.2 best)	**99.46**	29.73	0.003
SGAS (small)	98.89	0.40	0.003

Comparison with state-of-the-art architectures for node classification on PPI.

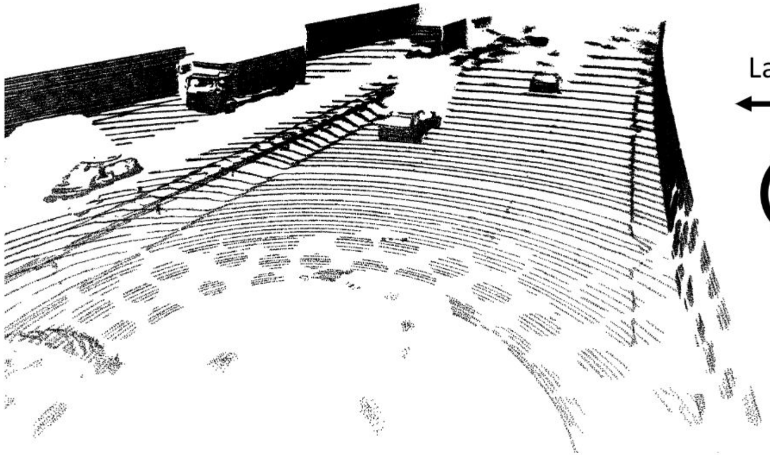


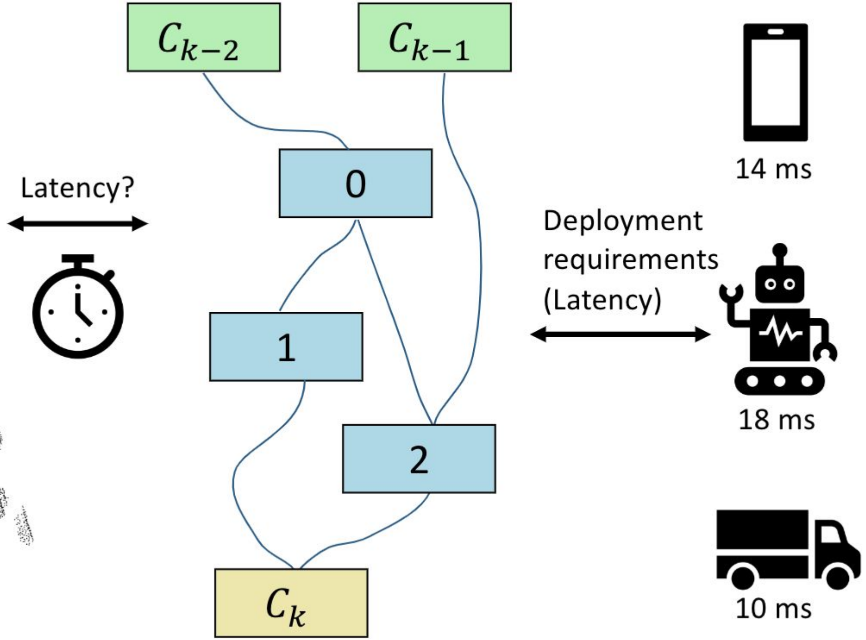
(a) Normal cell of the best model with SGAS (Cri. 1) on PPI



(b) Normal cell of the best model with SGAS (Cri. 2) on PPI

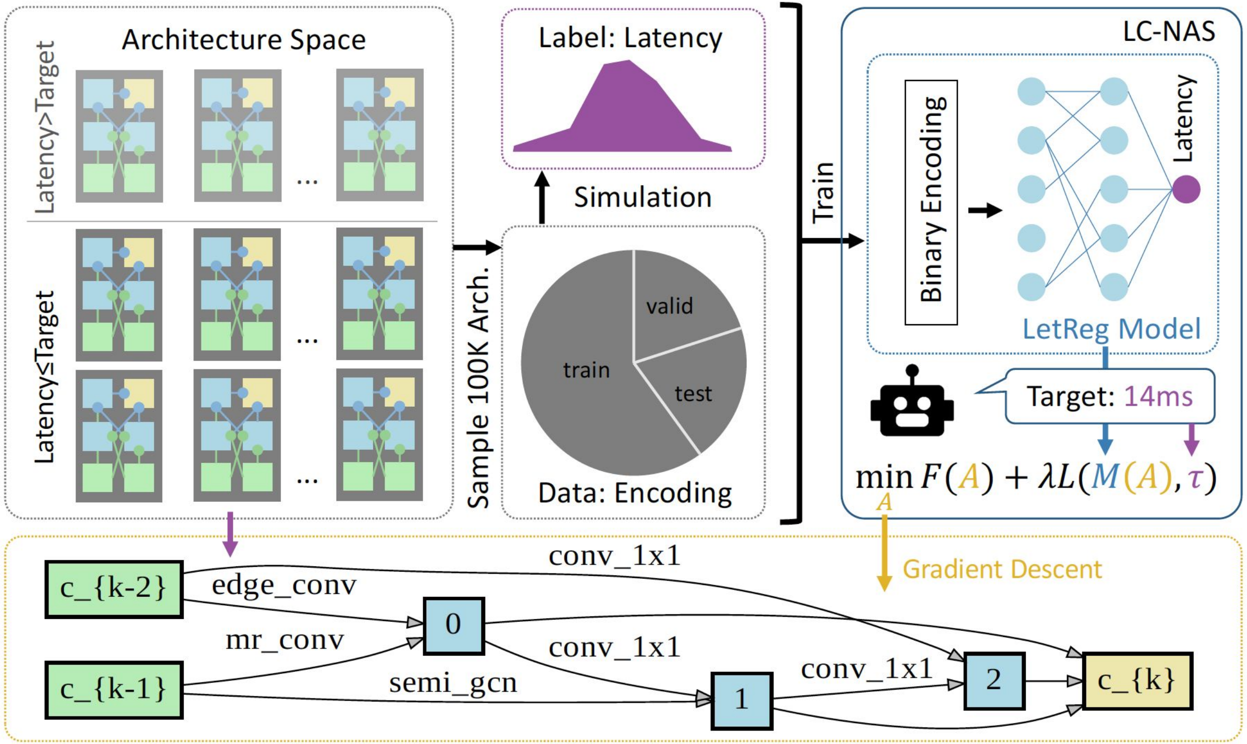
LC-NAS: Latency Constrained Neural Architecture Search



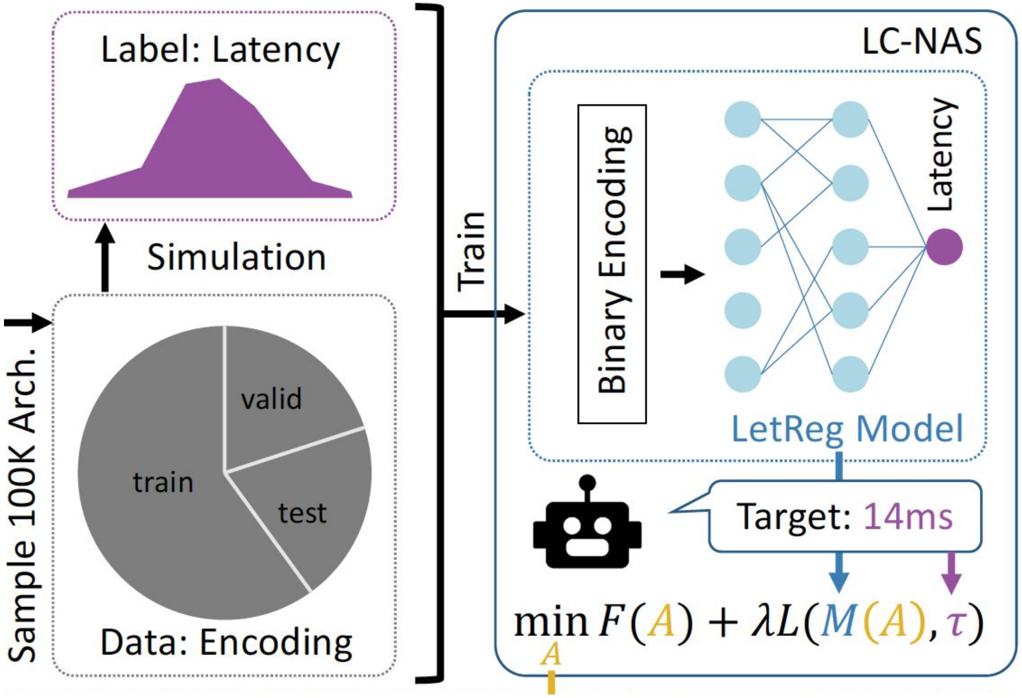


How to find the best performing model given an inference latency budget?

LC-NAS - Pipeline



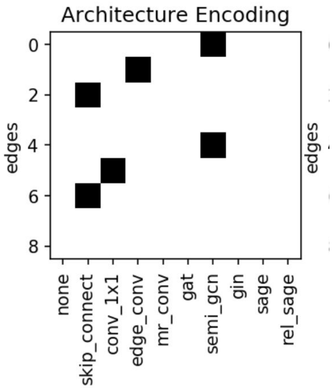
LC-NAS – Latency Regressor



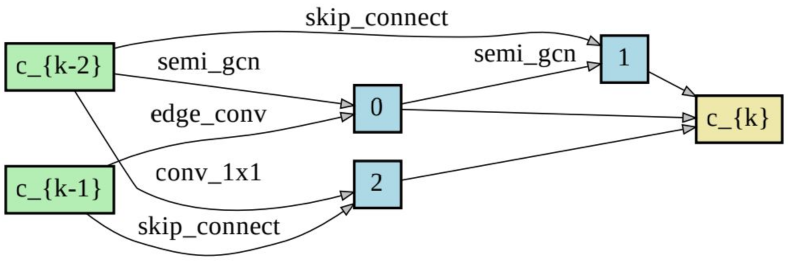
Our search space is a DAG with 9 edges and 10 candidate operations for each edge.

9×10 binary encoding matrix $\mathbf{E} \in \{0, 1\}^{9 \times 10}$

LC-NAS – Latency Regressor

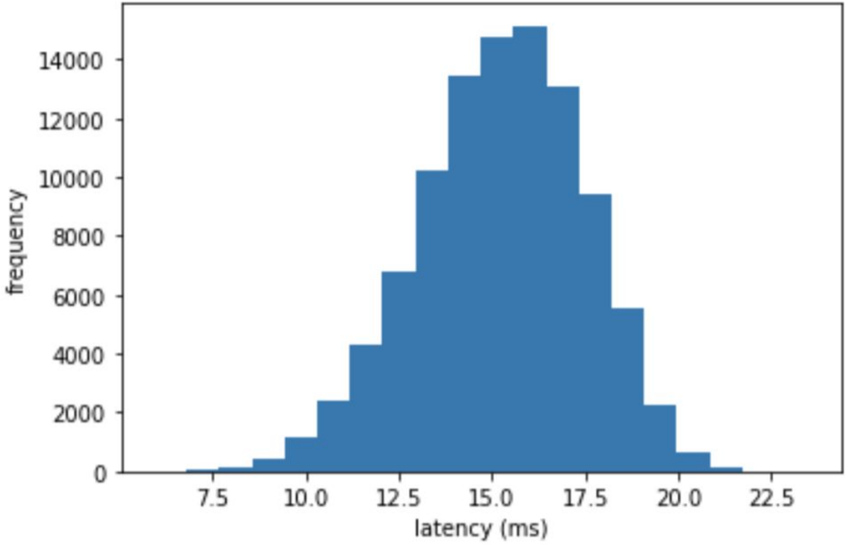


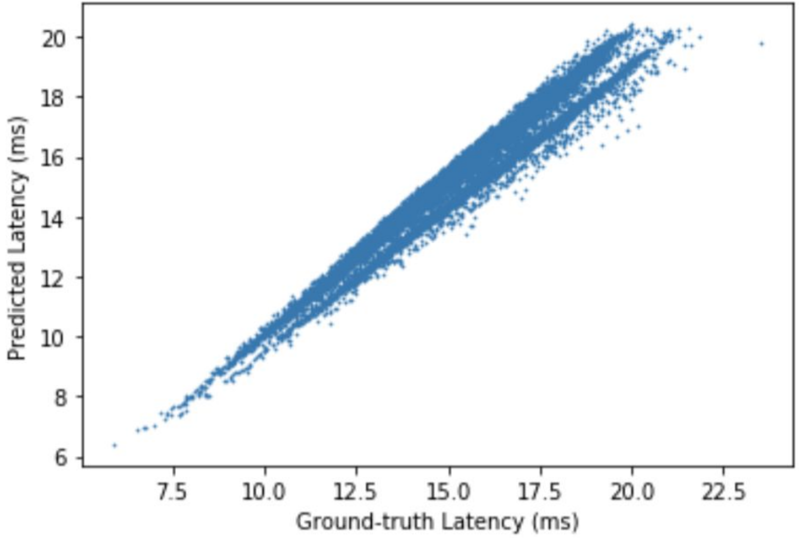
Our search space is a DAG with 9 edges and 10 candidate operations for each edge.



9×10 binary encoding matrix $\mathbf{E} \in \{0,1\}^{9\times 10}$

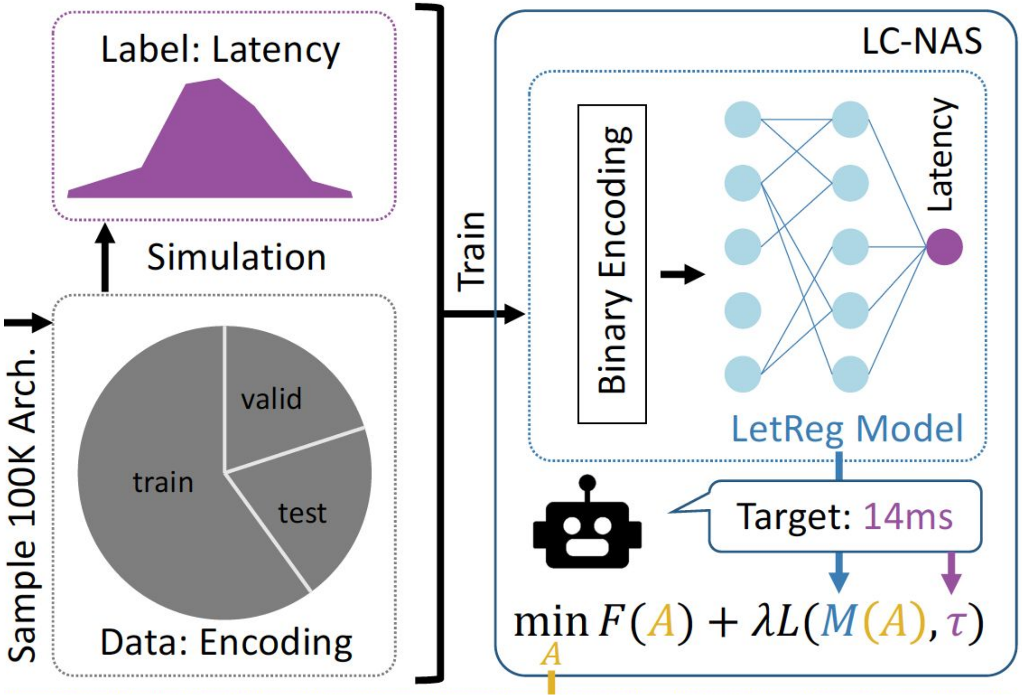
LC-NAS – Latency Regressor





LatReg Model (a 3-layer MLP): data distribution and performance.

LC-NAS - Target Latency as Constraint



$$\min_{\mathcal{A}} \quad \mathcal{L}_{val}(\mathcal{W}^*(\mathcal{A}), \mathcal{A}) \; + \lambda \; max(LatReg(\mathcal{E}(\mathcal{A})) - target, \; 0)$$
$$\text{s.t.} \quad \mathcal{W}^*(\mathcal{A}) = \text{argmin}_{\mathcal{W}} \; \mathcal{L}_{train}(\mathcal{W}, \mathcal{A})$$

LC-NAS - Target Latency as Constraint

$$\beta_{m,n} = softmax(\alpha_{m,n}|\boldsymbol{\alpha}_m) = \frac{\exp(\alpha_{m,n})}{\sum_k \exp(\alpha_{m,k})}$$

$$\zeta_{m,n} = \frac{1}{\beta_{m,n}} \quad \text{if} \quad n = n^*$$

$$\zeta_{m,n} = 0 \quad \text{if} \quad n \neq n^*$$

$$\mathcal{E}(\alpha_{m,n}) = \tilde{e}_{m,n} \approx \beta_{m,n} \cdot \zeta_{m,n}$$

Approximate non-differentiable heuristics to make it differentiable

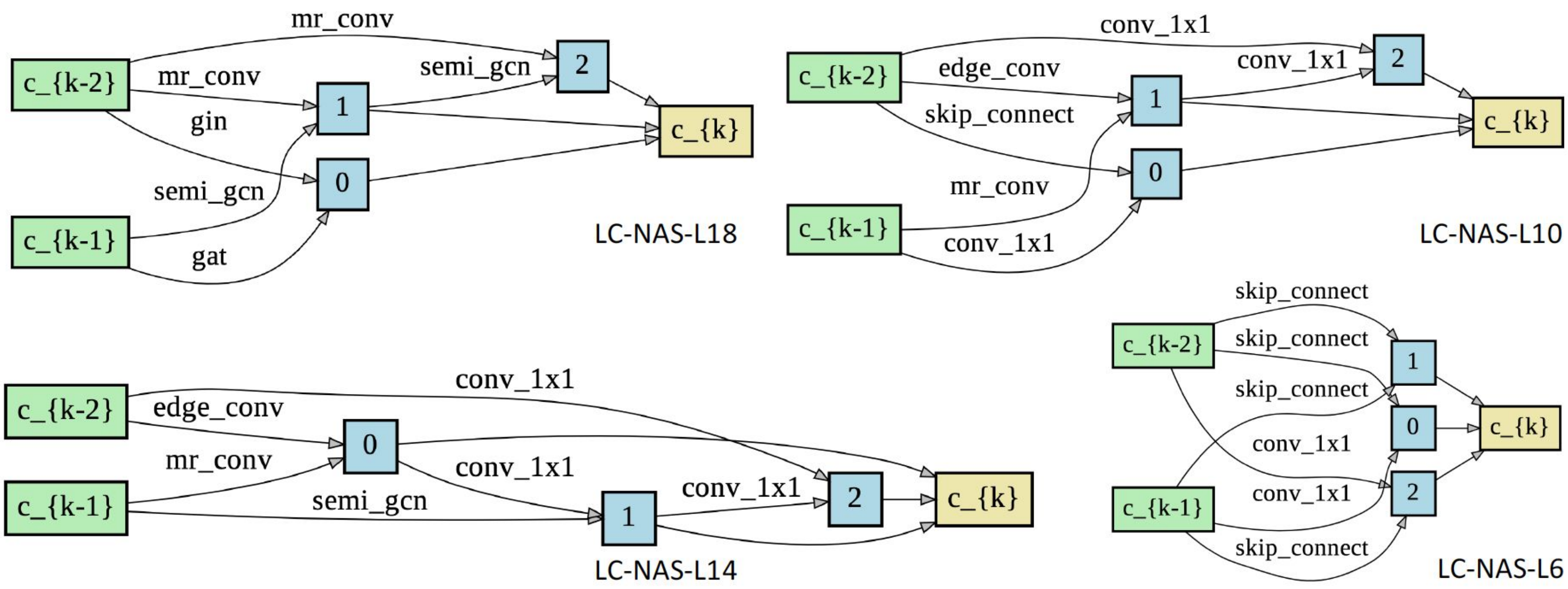
LC-NAS - Target Latency as Constraint

$$\frac{\partial \mathcal{L}_{lat}}{\partial \alpha_{m,n}} = \sum_k \frac{\partial \mathcal{L}_{lat}}{\partial \beta_{m,k}} \cdot \frac{\partial \beta_{m,k}}{\partial \alpha_{m,n}} = \sum_k \frac{\partial \mathcal{L}_{lat}}{\partial \tilde{e}_{m,k}} \cdot \frac{\partial \tilde{e}_{m,k}}{\partial \beta_{m,k}} \cdot \frac{\partial \beta_{m,k}}{\partial \alpha_{m,n}} \tag{3}$$

where $\frac{\partial \beta_{m,k}}{\partial \alpha_{m,n}} = \beta_{m,n} - \beta_{m,n}^2$ if $n = k$ and $\frac{\partial \beta_{m,k}}{\partial \alpha_{m,n}} = -\beta_{m,n} \cdot \beta_{m,k}$ if $n \neq k$. Since $\frac{\partial \tilde{e}_{m,k}}{\partial \beta m,k} = \zeta_{m,k}$. We obtain the gradient as follows:

$$\frac{\partial \mathcal{L}_{lat}}{\partial \alpha_{m,n}} = \frac{\partial \mathcal{L}_{lat}}{\partial \tilde{e}_{m,n^*}} \cdot \frac{1}{\beta_{m,n^*}} \cdot \frac{\partial \beta_{m,n^*}}{\partial \alpha_{m,n}} = \begin{cases} \frac{\partial \mathcal{L}_{lat}}{\partial \tilde{e}_{m,n^*}} \cdot (1 - \beta_{m,n^*}) & \text{for } n = n^* \\ \frac{\partial \mathcal{L}_{lat}}{\partial \tilde{e}_{m,n^*}} \cdot -\beta_{m,n} & \text{for } n \neq n^* \end{cases}$$

LC-NAS for GCN on ModelNet



Discovered Cells

LC-NAS for GCN on ModelNet

Method	Latency (ms) Target	Predicted	Measured	# Param. (M)	Accuracy (%) O.A.	C.A.
LC-NAS-18	18	17.06	16.66	3.91	**92.79**	89.66
LC-NAS-16	16	13.71	13.57	3.91	92.62	90.13
LC-NAS-14	14	12.64	12.41	3.91	92.42	89.16
LC-NAS-12	12	10.07	9.96	3.85	92.34	89.57
LC-NAS-10	10	11.02	11.09	3.86	92.75	**90.76**
LC-NAS-8	8	7.84	7.51	3.71	90.40	85.36
LC-NAS-6	6	6.12	5.47	3.61	90.51	84.71
Average	-	11.21	10.95	3.82	91.98	88.48

Evaluation on ModelNet40.

LC-NAS for GCN on ModelNet

Method	Lat. (ms)	O.A. (%)	Method	Lat. (ms)	O.A. (%)
PointNet [37]	4.21	89.2	LC-NAS-18	16.66	**92.79**
PointNet++ [38]	23.51	90.7	LC-NAS-16	13.57	92.62
DGCNN [53]	9.42	92.2	LC-NAS-14	12.41	92.42
PointCNN [27]	26.79	92.2	LC-NAS-12	9.96	92.34
PosPool (S) [31]	15.93	92.6	LC-NAS-10	11.09	92.75
SGAS [25]	16.62	92.9	LC-NAS-8	7.51	90.40
KPConv [50]	26.81	92.9	LC-NAS-6	5.47	90.51
RS-CNN [30]	58.4	93.6	-	-	-
DeepGCN [23]	56.7	93.6	-	-	-
PointMLP [32]	44.5	**94.1**	-	-	-

Comparison with SOTA on ModelNel40.

LC-NAS - Transfer on PartNet

Method	Lat. (ms)	Avg.	Bed	Bott	Chair	Clock	Dish	Disp	Door	Ear	Fauc	Knife	Lamp	Micro	Frid	Stora	Table	Trash	Vase
PointCNN	1402	46.49	41.9	41.8	**43.9**	36.3	58.7	82.5	37.8	48.9	**60.5**	34.1	20.1	**58.2**	42.9	49.4	21.3	53.1	**58.9**
SGAS	185	48.28	**43.4**	50.8	41.2	38.8	61.4	82.6	37.1	48.8	56.1	49.4	21.2	56.5	44.5	**49.4**	29.3	54.4	56.0
deep LPN	191	38.60	29.5	42.1	41.8	34.7	33.2	81.6	34.8	**49.6**	53.0	44.8	**28.4**	33.5	32.3	41.1	**36.3**	43.1	57.8
LC-NAS-10	**143**	48.10	41.4	50.5	39.6	37.8	61.1	**82.9**	37.4	48.4	53.6	48.5	22.3	57.8	**46.6**	47.9	31.1	**54.8**	56.0
LC-NAS-14	152	**48.55**	41.9	**51.7**	39.7	**39.6**	**61.5**	82.5	**39.3**	49.0	54.7	**55.3**	22.2	55.1	45.2	48.0	30.3	54.6	54.9

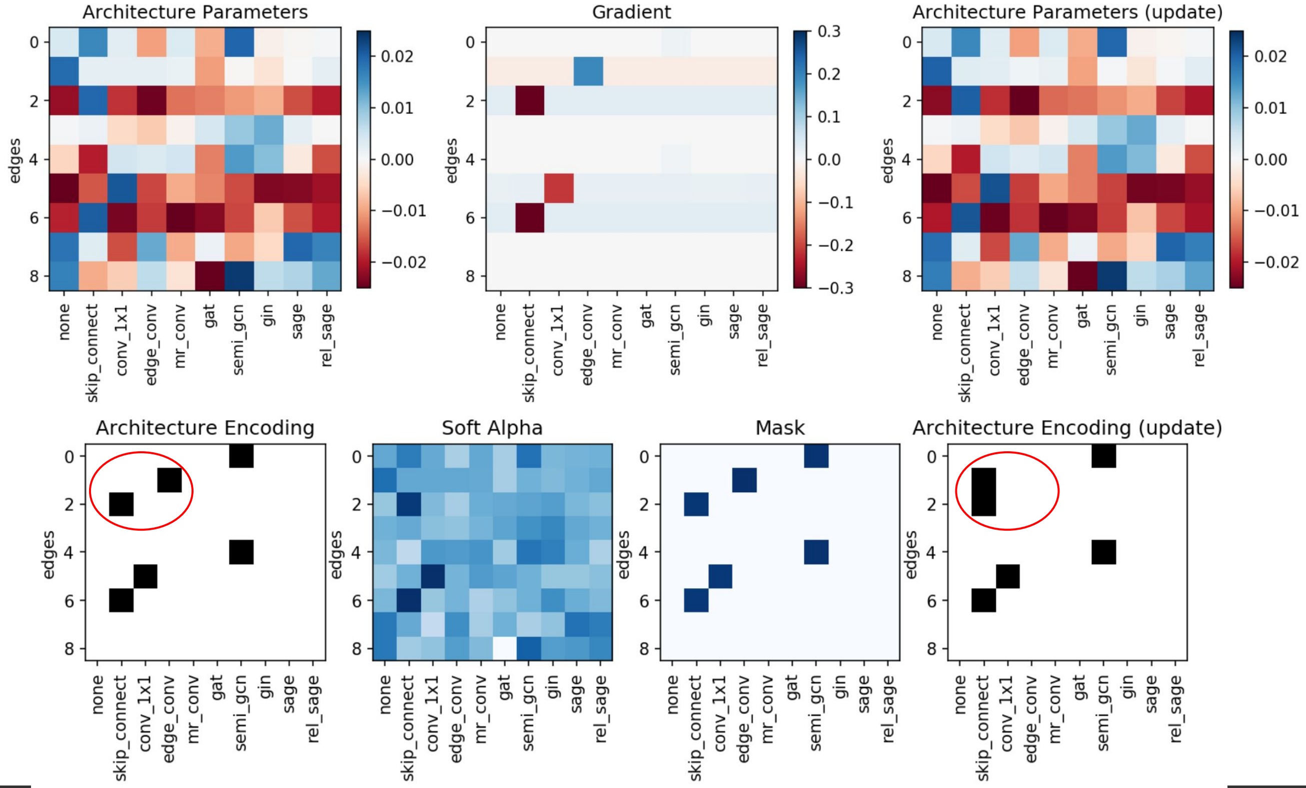
Part Segmentation on PartNet (part mIoU % on level 3).

LC-NAS - Ablation

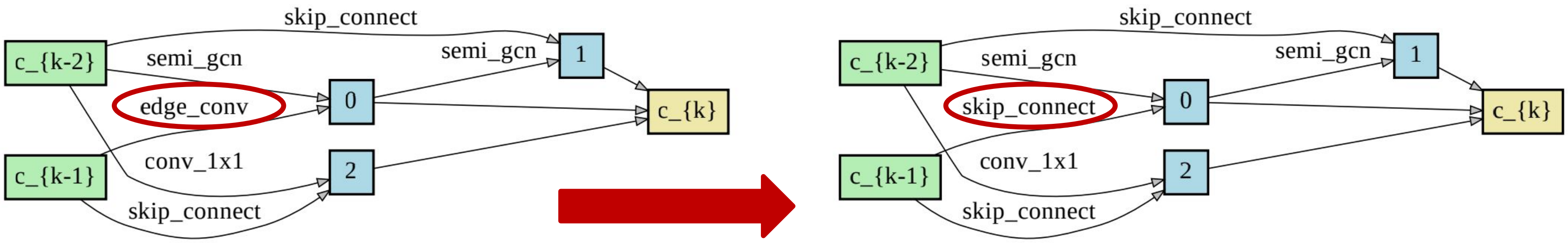
λ	Latency (ms)	O.A. (%)	C.A. (%)
0.5	6.60	90.24	84.70
0.1	6.19	90.76	85.37
0.05	6.35	90.28	84.90
0.01	9.64	92.26	89.00
0.005	8.82	86.83	80.65
0.001	14.63	92.63	89.98
0.0005	12.08	92.50	89.75
0.0001	18.71	92.50	89.68

Ablation on Non-Targeted Latency Loss.

LC-NAS – Gradient Visualization



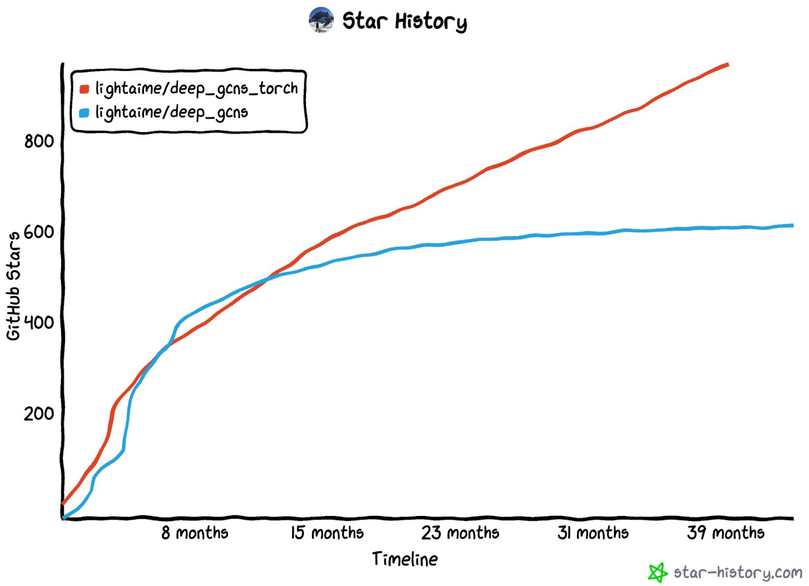
LC-NAS – Gradient Visualization



Search dynamic with a targeted latency constraint

Open Source

DeepGCNs.org



> 1500 Stars (Pytorch + Tensorflow), 1200 citations

PyG

DGL
DEEP GRAPH LIBRARY

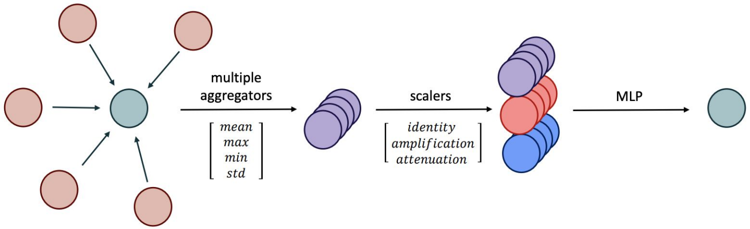
Available on PyG and DGL

Open Source

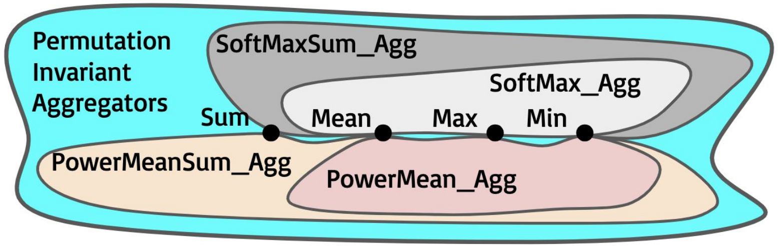
- ✓ *Principled* **Aggregations**
- ✓ **Scalable Link Prediction**
- ✓ **Temporal Samplers**
- ✓ **...**

PyG 2.1 Release

Join PyG.org Team as a core member



Corso, G., Cavalleri, L., Beaini, D., Liò, P. and Veličković, P., 2020. Principal neighbourhood aggregation for graph nets.



Li, G., Xiong, C., Thabet, A. and Ghanem, B., 2020. Deepergcn: All you need to train deeper gcns.

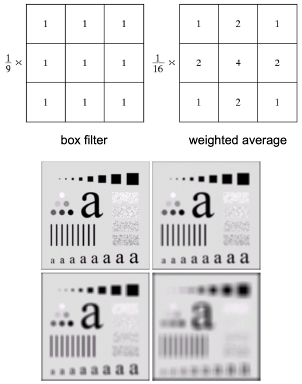
Make the concept of aggregation a first-class principle in PyG

Training “Deep” GNNs

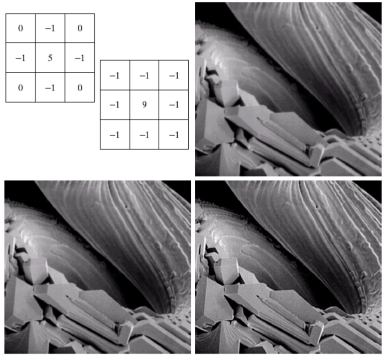
- Training “Deep” GNNs
  - Skip Connection: JKNet (Xu et al., 2018), DeepGCNs (Li et al., 2019; 2020), Aff-GCN (Gong et al., 2020), GCNII (Chen et al., 2020), Implicit Acceleration (Xu et al., 2021), GNN1000 (Li et al., 2021)
  - Normalization & Regularization: DropEdge (Rong et al., 2020), DropConnect (Hasanzadeh et al., 2020), PairNorm (Zhao & Akoglu, 2019), WeightNorm (Oono & Suzuki, 2019), DiffGroupNorm (Zhou et al., 2020), GraphNorm (Cai et al., 2020)
  - Efficient Propagation: SGC (Wu et al., 2019), APPNP (Klicpera et al., 2019), PPRGo (Bojchevski et al., 2020), DAGNN (Liu et al., 2020), SIGN (Frasca et al., 2020)

Discussions

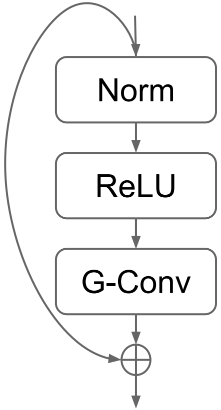
- Over-smoothing assumption is too strong (e.g. ignoring weights and activations)
- Why do not over-smooth? Possibly, Identity mapping, Invertible Graph Conv
- Depth and diameter (1001 layers GNN on ogbn-proteins with a graph diameter as 9)
- Depth and width (compounding scaling rule)
- Depth and datasets (benefit more on geometric graphs, 3D, proteins, molecules but less on citation networks)
- OOD split on OGB is challenging, need other techniques to help (transfer learning, zero-shot learning)



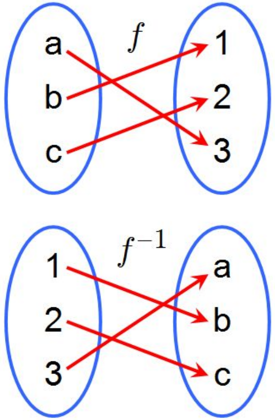
Smoothing Filter



Sharpening Filter



identity mapping if **W**= **0**



Invertible ⇄ Bijective

Towards Structured Intelligence with Deep Graph Neural Networks

Making GCNs Go as Deep as CNNs:
Skip Connections and Dilated Convolutions on Graphs

Making GCNs Go as Deep as CNNs:
Message Aggregation Functions;
Memory Efficiency

Automate GNN Architecture Design:
Sequential Greedy Architecture Search;
Latency Constraint

Ongoing Work and Research Plan:
Structured Navigation;
Research Plan

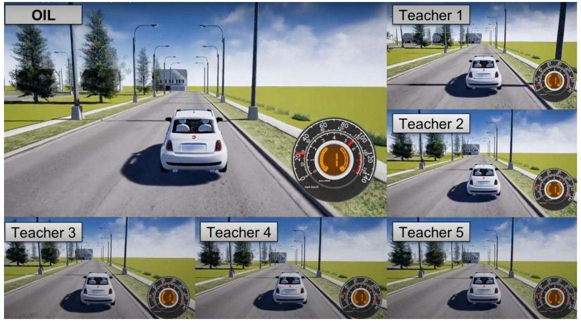
1

2

3

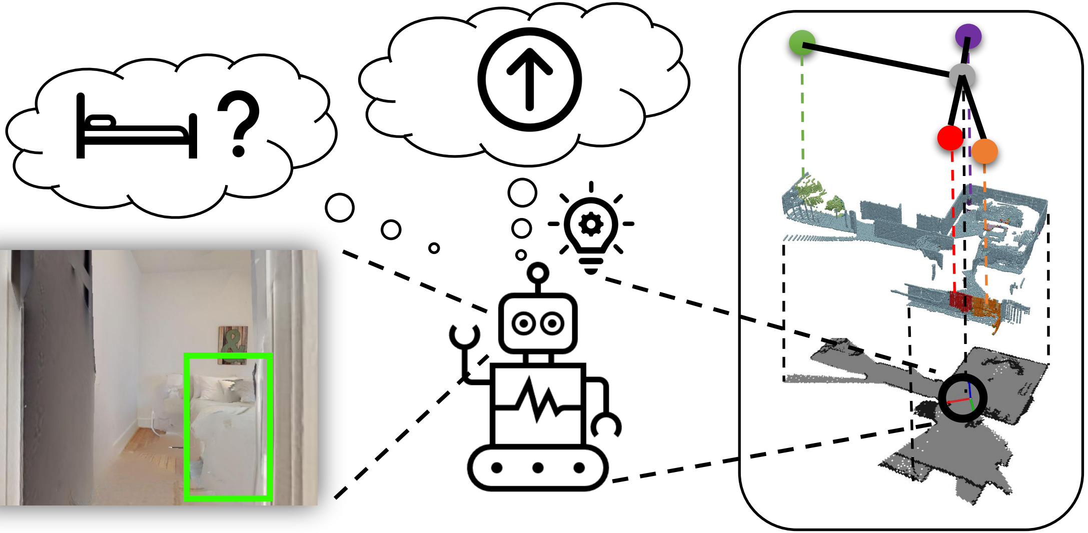
4

Structured Navigation



OIL RSS'19

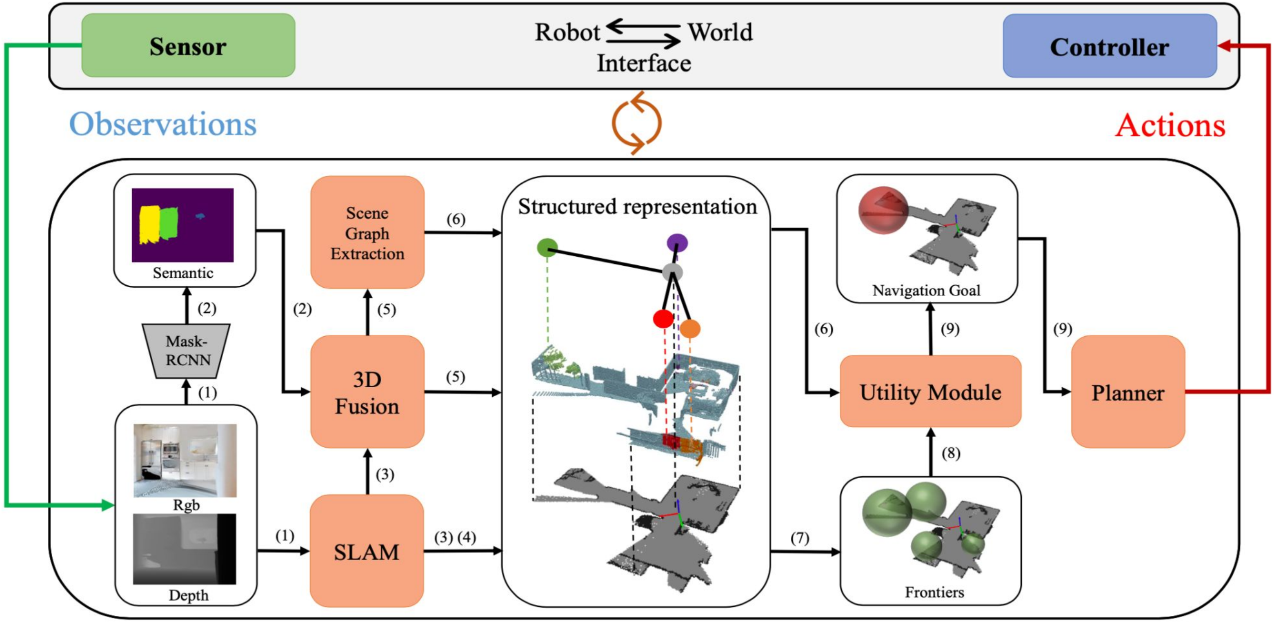
ROBOTICS SCIENCE AND SYSTEMS



Object Goal Navigation

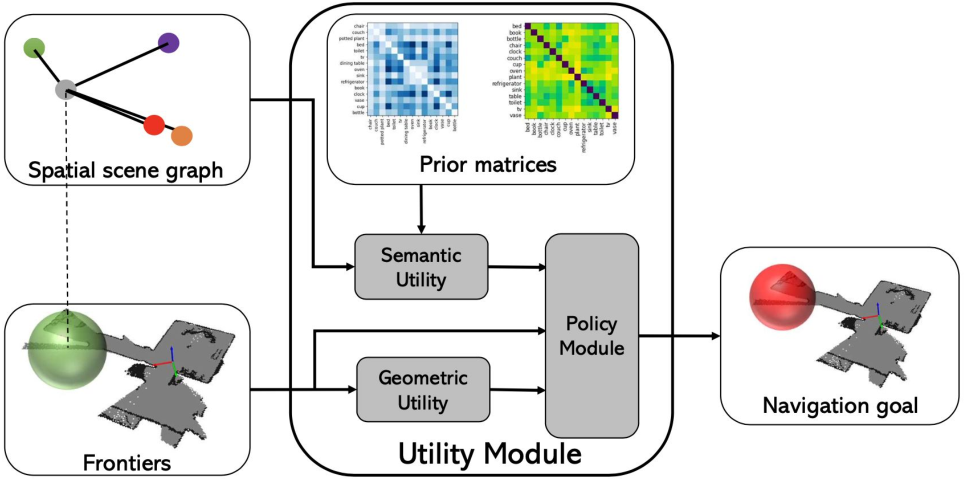
StructNav ICRA'23 Submission

Structured Navigation



StructNav Pipeline

Structured Navigation



Our method:
Inject semantics to Geometric Frontiers with Scene Graph and Large-Scale Language Model

+Training Free
+ 4.3% Success Rate
+ 7.5% Success-weighed Path Length (SPL)

Contributed projects

FLAG: Robust Optimization as Data Augmentation for Large-scale Graphs (CVPR'2022)
Kezhi Kong, **Guohao Li**, Mucong Ding, Zuxuan Wu,
Chen Zhu, Bernard Ghanem, Gavin Taylor, Tom Goldstein

ASSA: Anisotropic Separable Set Abstraction for Efficient Point Cloud Representation Learning (NeurIPS'2021 Spotlight)
Guocheng Qian, Hasan Hammoud, **Guohao Li**, Ali Thabet, Bernard Ghanem

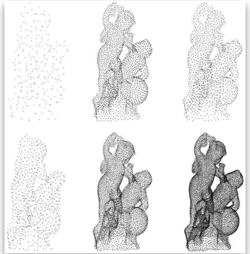
PU-GCN: Point Cloud Upsampling via Graph Convolutional Network (CVPR'2021)
Guocheng Qian, Abdulellah Abualshour, **Guohao Li**,
Ali Thabet, Bernard Ghanem

Learning Scene Flow in 3D Point Clouds with Noisy Pseudo Labels
Anonymous Submission

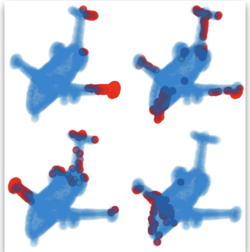
When NAS Meets Trees: A New Paradigm for Neural Architecture Search
Anonymous Submission

Knowledge-aware Global Reasoning for Situation Recognition
Anonymous Submission

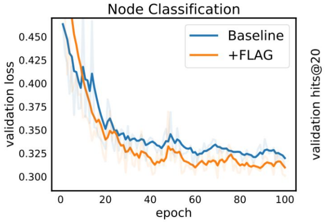
UnrealNAS: Can We Search Neural Architectures with Unreal Data?
Anonymous Submission



PUGCN CVPR'21

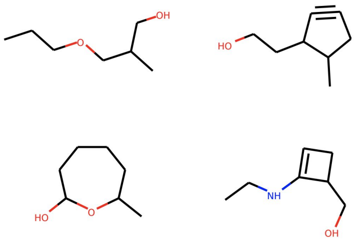


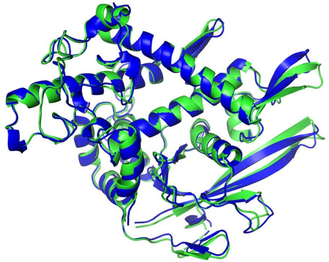
ASSA NeurIPS'21

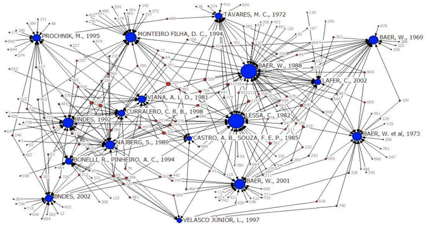


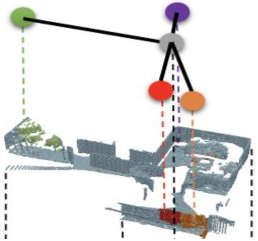
FLAG CVPR'22

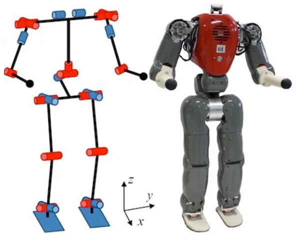
Towards Structured Intelligence with Deep Graph Neural Networks











Architectures.

- How to train large-scale GNNs efficiently?
- What are the proper inductive biases to add to GNNs?
- Is there a universal model for learning on different types of graphs?

Learning Paradigms.

- How to pre-train models like GPT-3 and BERT for graphs?
- How to efficiently transfer pre-trained GNNs?
- How to make learning automatic?

Applications

- Learning on irregular 3D geometric data;
- Building structured knowledge representation of dynamic environments for embodied agents;
- Training transferable representation with GNNs on large-scale biological networks and 3D molecular graphs for scientific applications such as drug discovery, molecular property prediction and molecular design.

PhD Journey

2018
Started my PhD at KAUST in Fall 2018

2019
Published Papers in RSS, CVPRW and ICCV (Oral)

2020
Published 1 Paper in CVPR
Interned at Intel ISL
Research Excellence Awards at KAUST
1 st Place at NEOM AI challenge

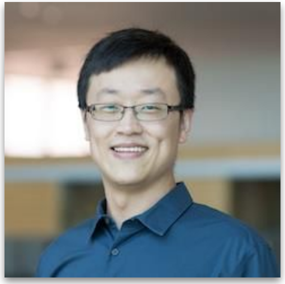
2021
Published Papers in CVPR, TPAMI, ICML, NeurIPS
Visited ETH CVL
Awardee at OGB-LSC @ KDD Cup

2022
Published Papers in CVPR, 3DV
GML4VC Tutorial at CVPR
Intern at Kumo.AI
Join PyG.org
Dean's List Award

Acknowledgement

Bernard Ghanem

Pietro Liò



Xin Gao

Helmut Pottmann

Committee Members

Acknowledgement

Matthias Müller

Vladlen koltun

Suryansh Kumar

Fisher Yu

Matthias Fey

Jure Leskovec

intel ETH zürich kumo

Internship Mentors

Acknowledgement


Matthias Müller


Ali Thabet


Guocheng Qian


Silvio Giancola


Neil Smith


Itzel C. Delgadillo


Abdulellah Abualshour


Chenxin Xiong


Jesus Zarzar

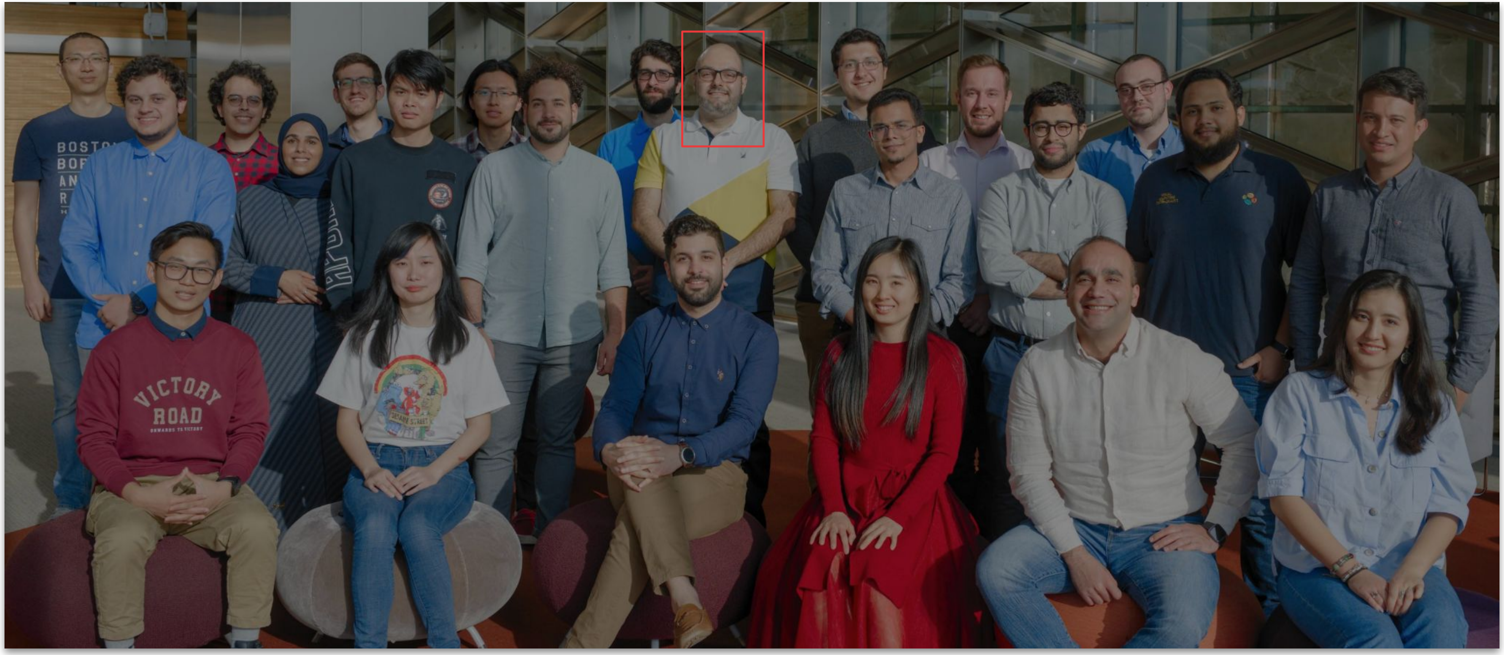

Mengmeng Xu


Kezhi Kong

Collaborators

Tom Goldstein
Zhu Chen
Vincent Casser
Dominik L Michels
Hasan Hammoud
Weijiang Yu
Haofan Wang
Junting Chen
Bing Li
Cheng Zheng
Chen Zhao
Xuanyang Zhang
Kurt Keutzer
Shanghang Zhang
Zhen Dong
Kaicheng Zhou
Qiang Zhou
Mingfei Guo
Kumail Al Hamoud
Yasir Ghunaim
Rana AlShedayed
Hani Itani
Jinjie Mai
…

Acknowledgement



جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

IVUL Image and Video Understanding Lab

Acknowledgement

My Beloved Family

Pair Programming Buddy - Eigen

Acknowledgement

Bernard Ghanem

Took at ICCV Deadline

My PhD Supervisor

Thanks Everyone

Towards Structured Intelligence with Deep Graph Neural Networks

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

KAUST IVUL