GCN INTRODUCTION AND ITS APPLICATION IN 3D POINT CLOUD SEMANTIC SEGMENTATION

Yisong Li (NVIDIA), Guohao Li (KAUST)

OUTLINE

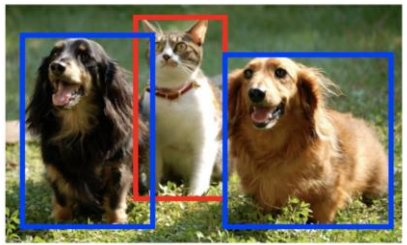
- Grid Data vs General Graphs
- CNN vs GCN
- ResGCN
- Experiments on 3D Cloud Point Segmentation
- Sequential Greedy Architecture Search
- Training efficiency

Papers in this talk

DeepGCNs: Can GCNs go as deep as CNNS. (ICCV 2019 Oral, Guohao Li et.al)

SGAS: Sequential Greedy Architecture Search (arXiv 2019, Guohao Li et.al)

Grid data vs. General graphs

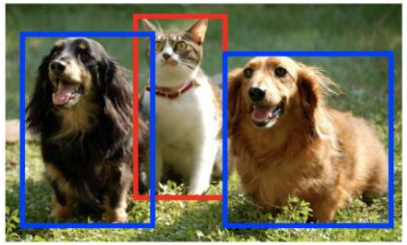


CAT, DOG

Grid Data:

- Image

Grid data vs. General graphs

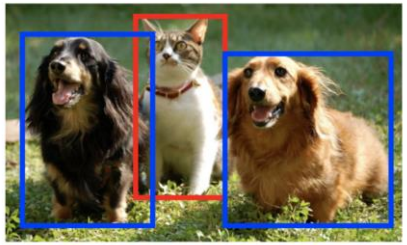


CAT, DOG

Grid Data:

- Image
- Video

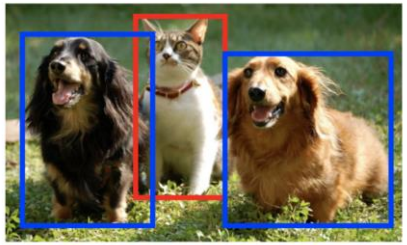
Grid data vs. General graphs



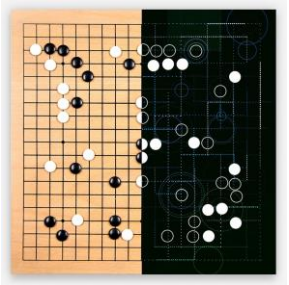
Grid Data:

- Image
- Video
- Audio
- Text

Grid data vs. General graphs



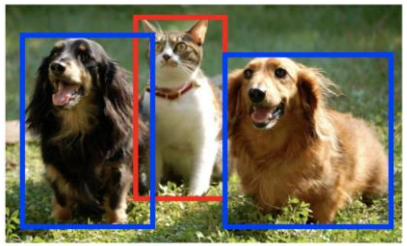
CAT, DOG



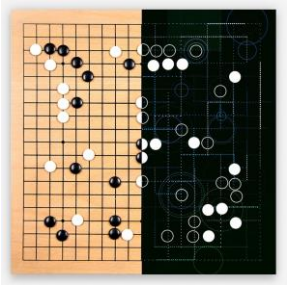
Grid Data:

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

Grid data vs. General graphs



CAT, DOG

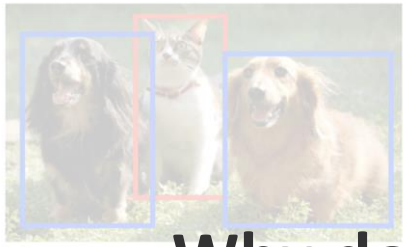


Grid Data:

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

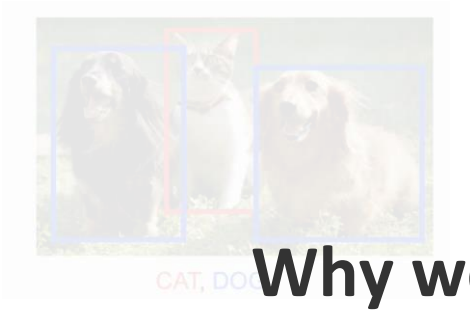
CNN works well

Grid data vs. General graphs



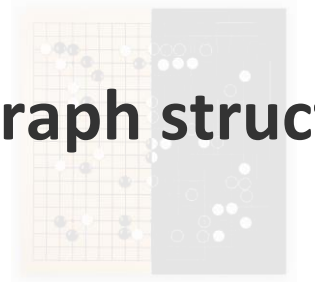
Why do we need graph convolutional networks?

Grid data vs. General graphs



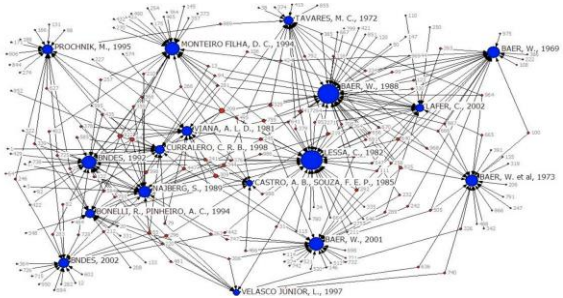
Why we need graph convolutional networks?

Tremendous non-grid graph structured data



Grid data vs. General graphs

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



General Graphs:

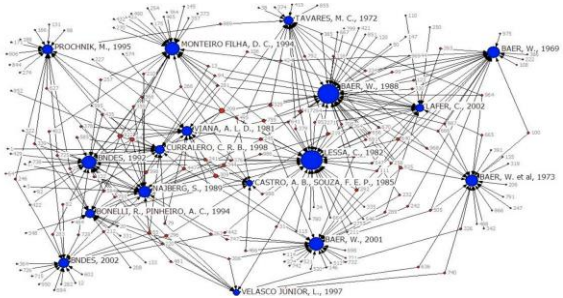
- Social Networks
- Citation Networks

Grid data vs. General graphs

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

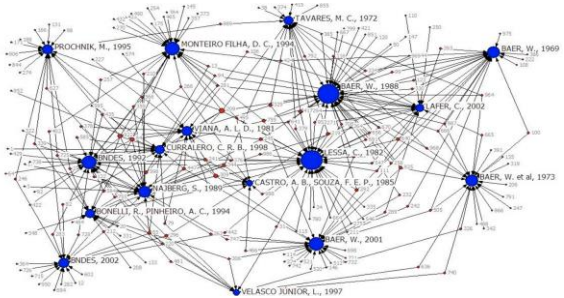
General Graphs:

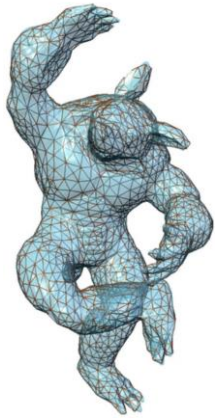
- Social Networks
- Citation Networks
- Molecules



Grid data vs. General graphs

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



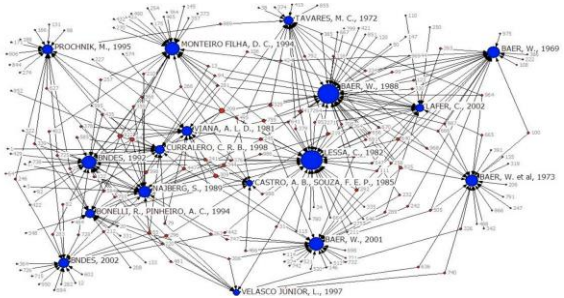


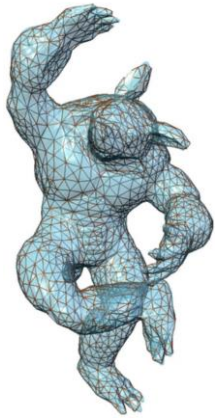
General Graphs:

- 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





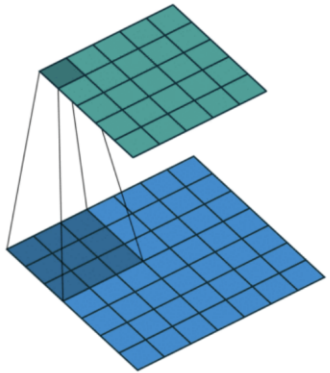
General Graphs:

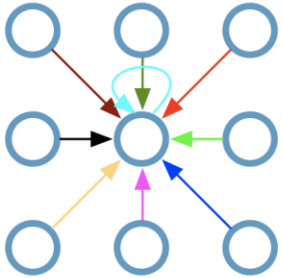
- Social Networks
- Citation Networks
- Molecules
- Point Clouds
- 3D Meshes
- ...

CNN doesn't work
GCN to rescue

CNN vs. GCN - Recap: CNN

Single CNN layer with 3x3 filter:

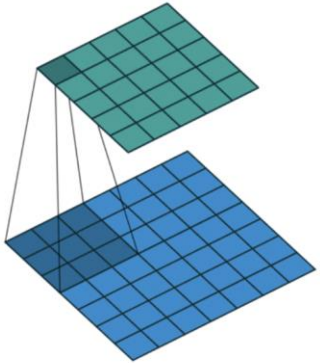


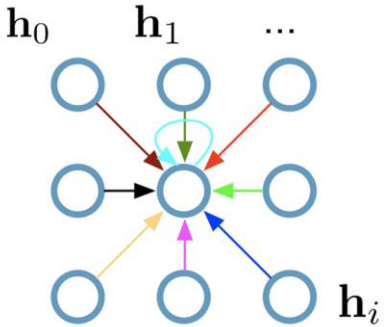


By Thomas Kipf.

CNN vs. GCN - Recap: CNN

Single CNN layer with 3x3 filter:

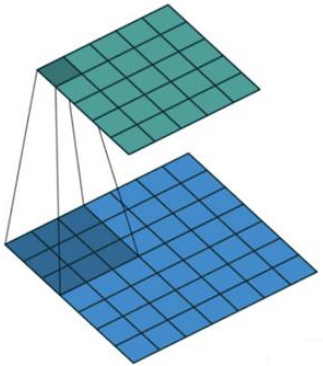


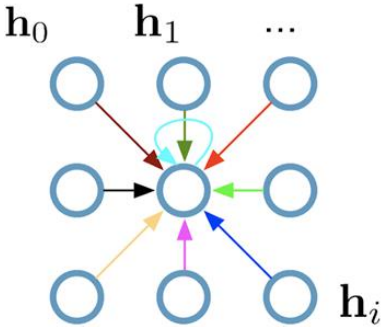


By Thomas Kipf.

CNN vs. GCN - Recap: CNN

Single CNN layer with 3x3 filter:



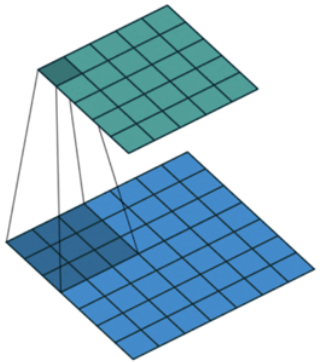


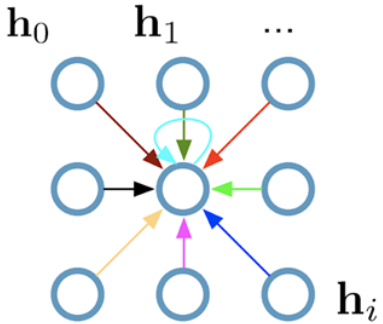
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

By Thomas Kipf.

CNN vs. GCN - Recap: CNN

Single CNN layer with 3x3 filter:





Update for a single pixel:

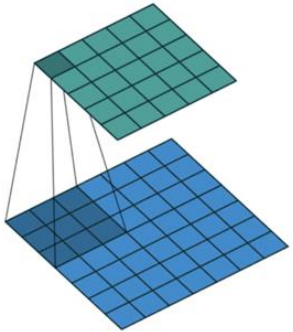
- Transform messages individually $\mathbf{W}_i\mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i\mathbf{h}_i$

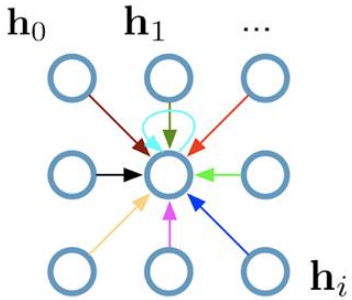
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

By Thomas Kipf.

CNN vs. GCN - Recap: CNN

Single CNN layer with 3x3 filter:





Update for a single pixel:

- Transform messages individually $\mathbf{W}_i\mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i\mathbf{h}_i$

$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

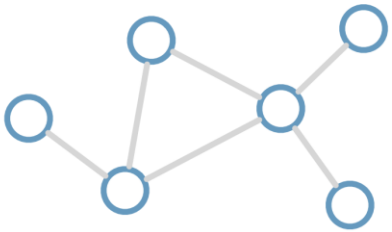
Full update:

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

By Thomas Kipf.

CNN vs. GCN - Introduction: GCN

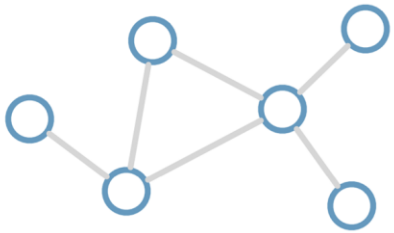
Consider this undirected graph:



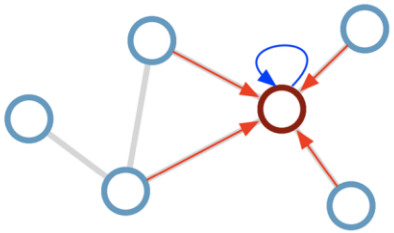
By Thomas Kipf.

CNN vs. GCN - Introduction: GCN

Consider this undirected graph:



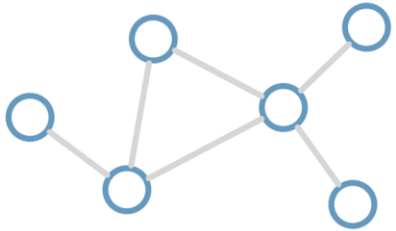
Calculate update for node in red:



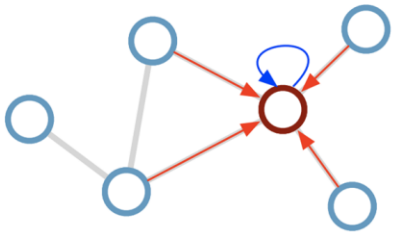
By Thomas Kipf.

CNN vs. GCN - Introduction: GCN

Consider this undirected graph:



Calculate update for node in red:



$\mathcal{N}_i$: neighbor indices

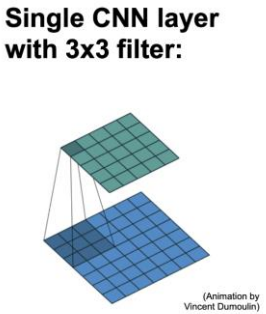
c_{ij} : norm. constant (fixed/trainable)

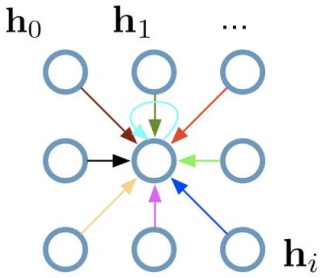
Update rule:

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

By Thomas Kipf.

CNN vs. GCN - Comparison



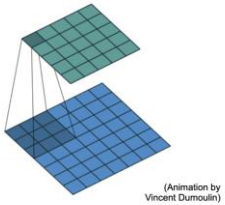


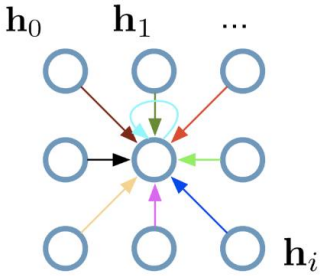
Convolutional Neural Network (CNN)

By Thomas Kipf.

CNN vs. GCN - Comparison

Single CNN layer with 3x3 filter:





Full update:

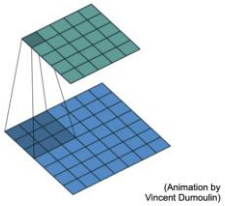
$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

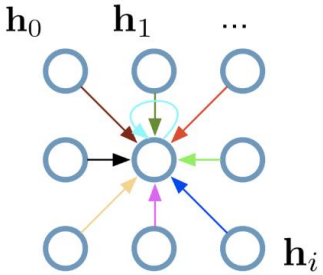
Convolutional Neural Network (CNN)

By Thomas Kipf.

CNN vs. GCN - Comparison

Single CNN layer with 3x3 filter:



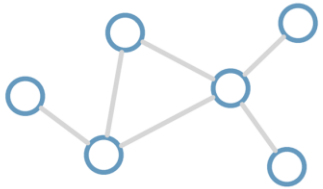


Full update:

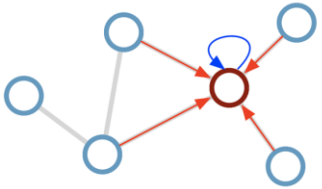
$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

Convolutional Neural Network (CNN)

Consider this undirected graph:



Calculate update for node in red:



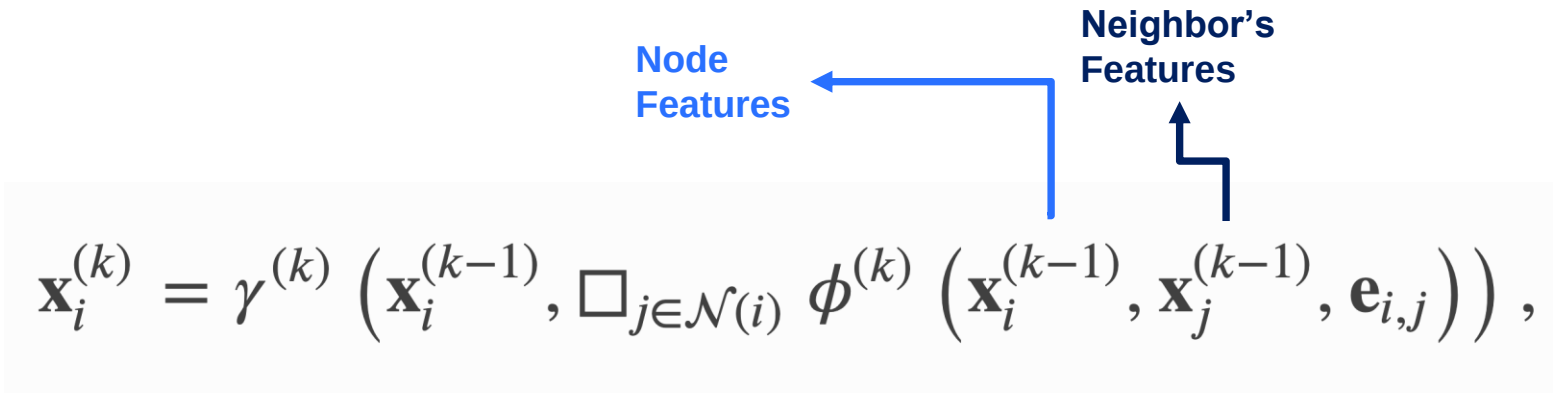
Update rule:

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

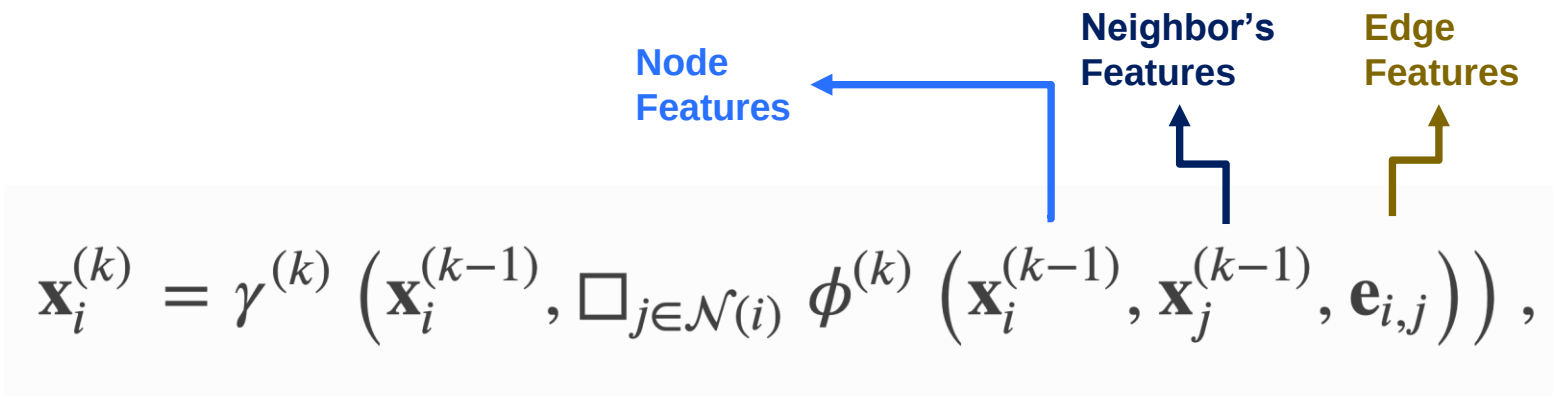
Graph Convolutional Network (GCN)

By Thomas Kipf.

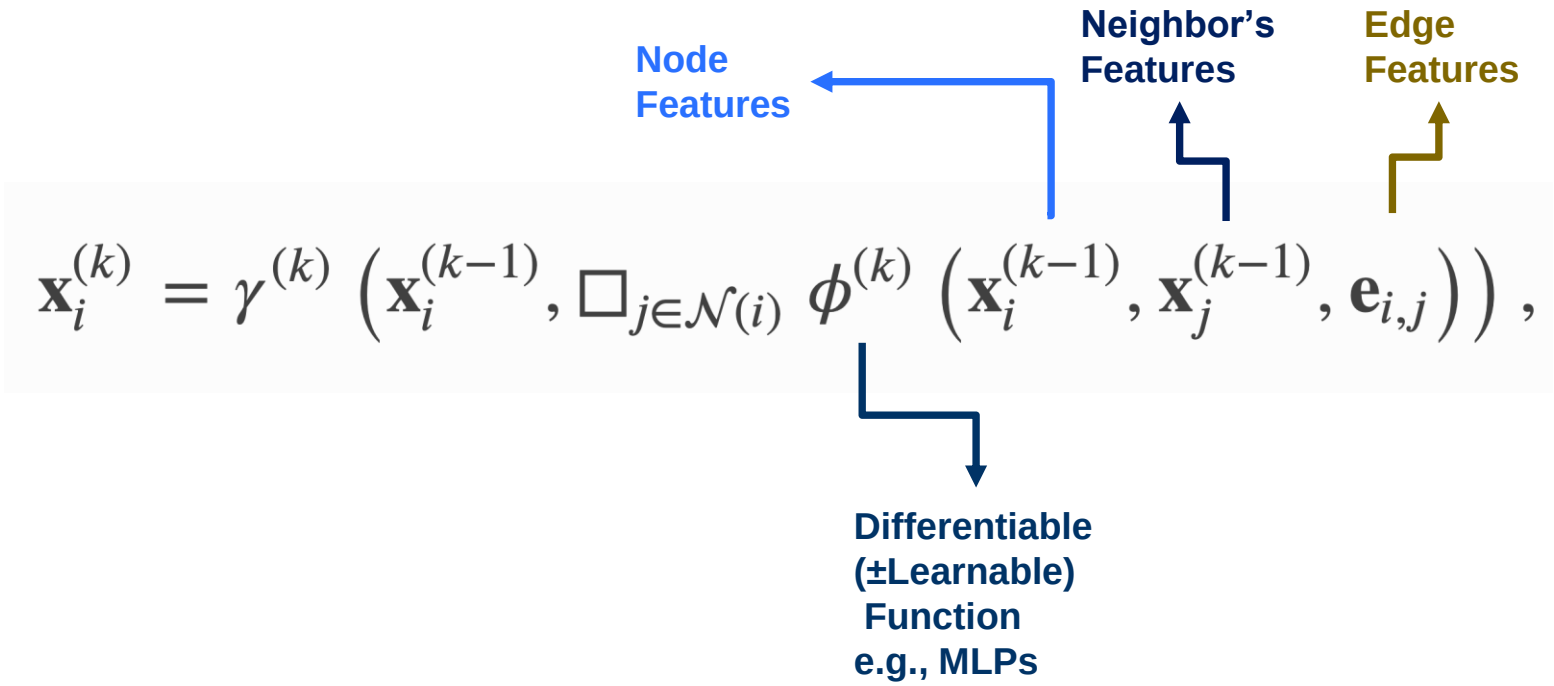
CNN vs. GCN - Message Passing



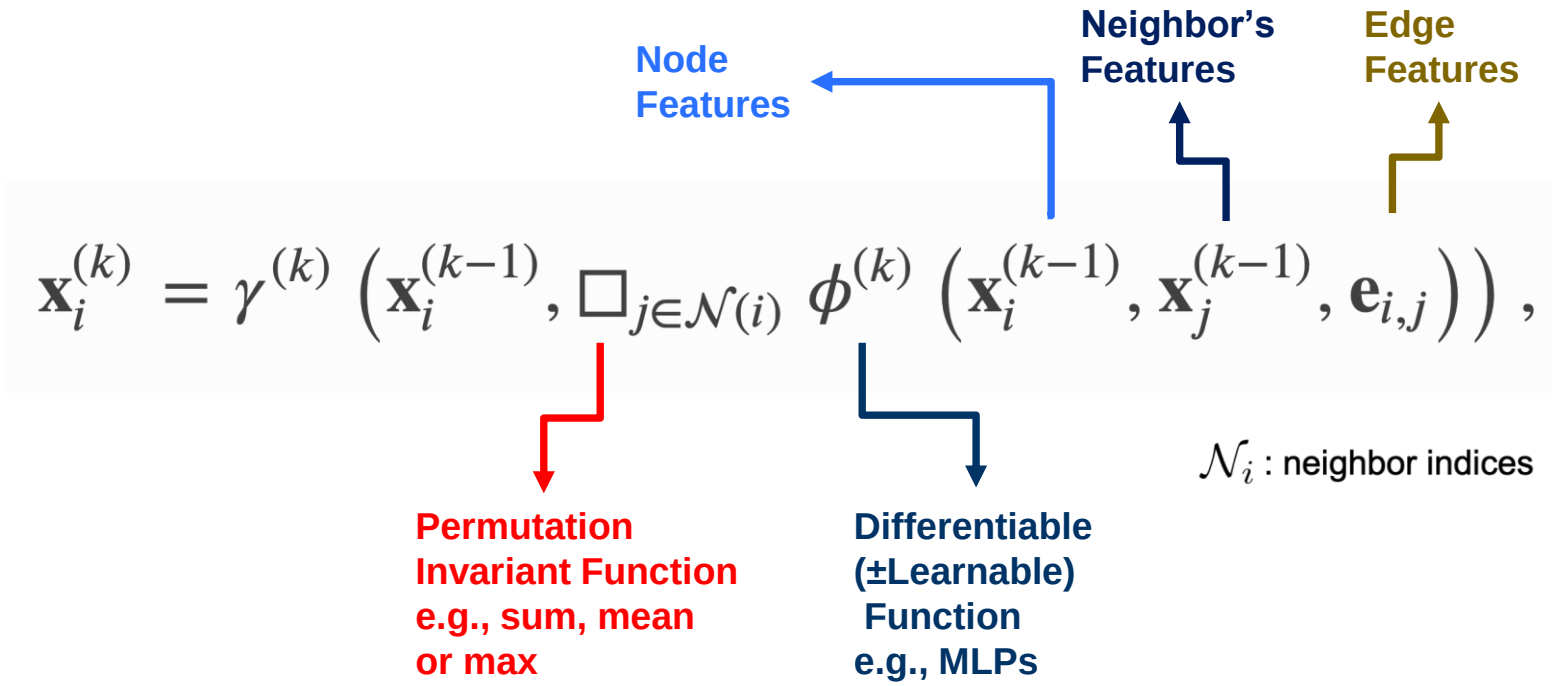
CNN vs. GCN - Message Passing



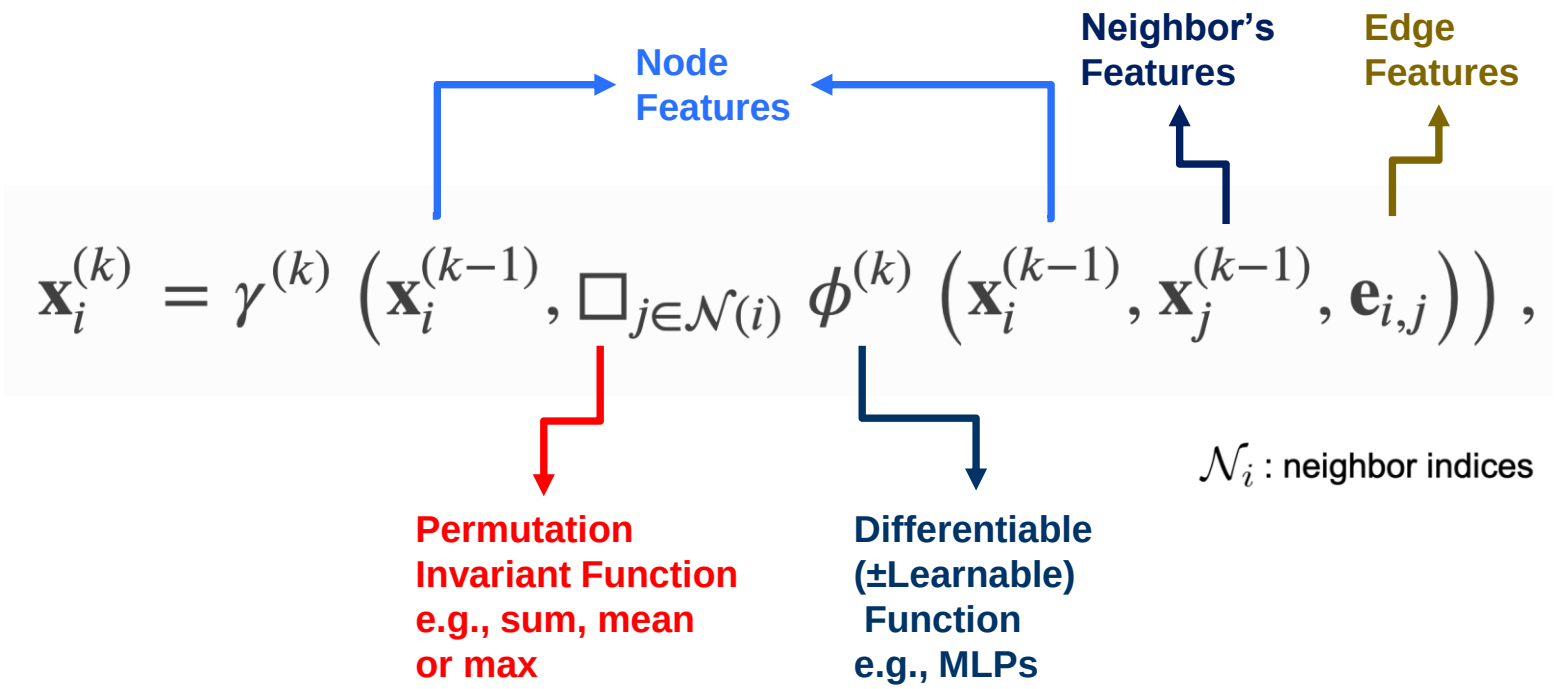
CNN vs. GCN - Message Passing



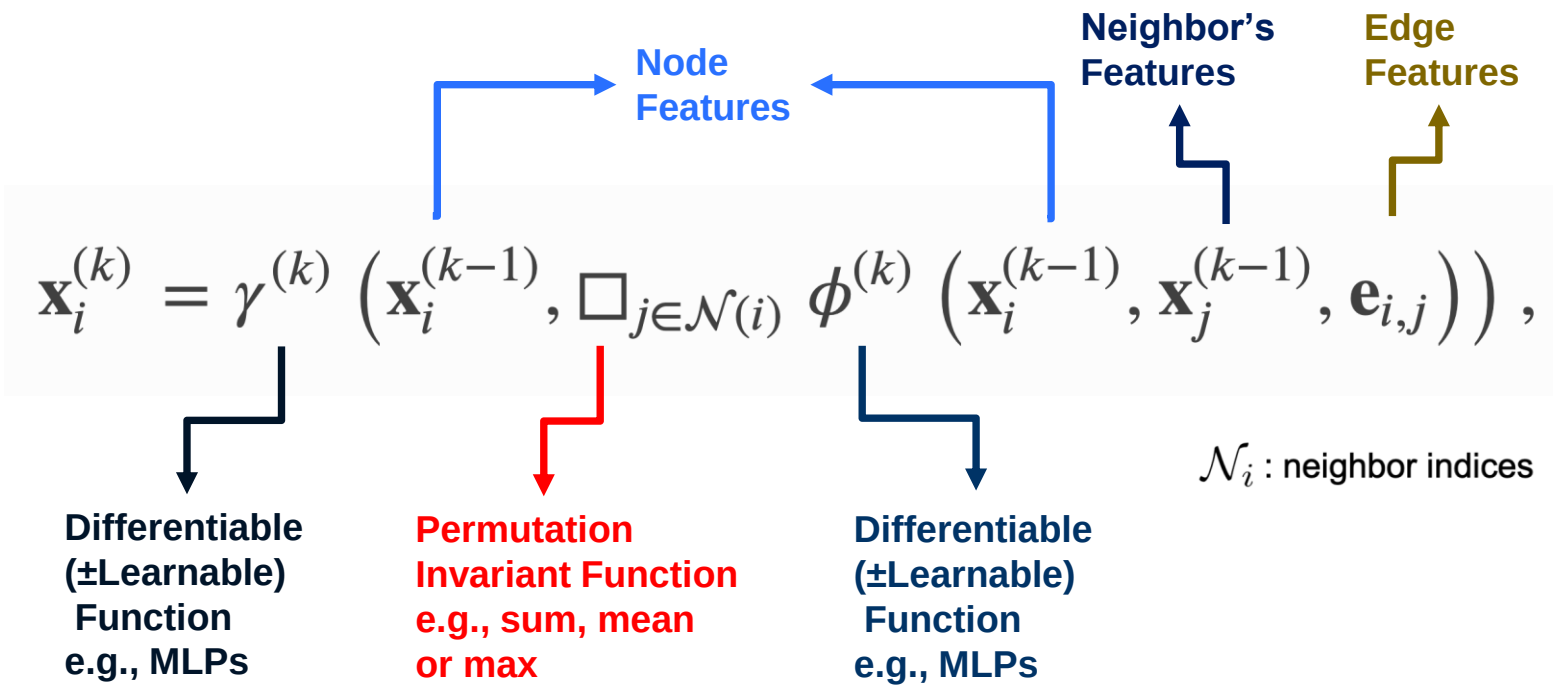
CNN vs. GCN - Message Passing

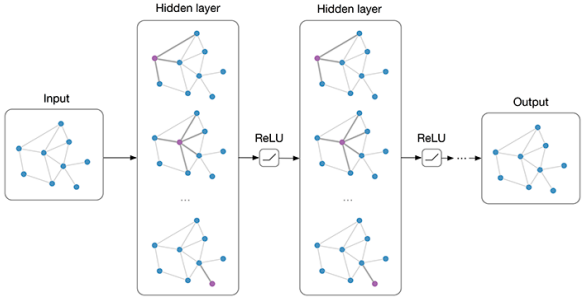


CNN vs. GCN - Message Passing

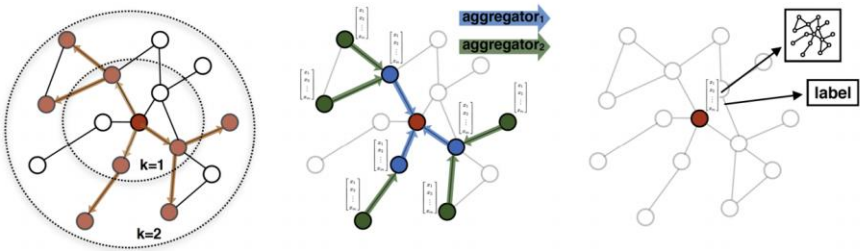


CNN vs. GCN - Message Passing



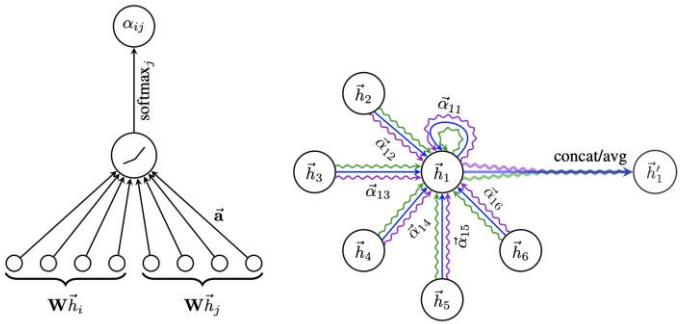


Kipf, T.N. and Welling, M., 2016. Semi-Supervised Classification with Graph Convolutional Networks.

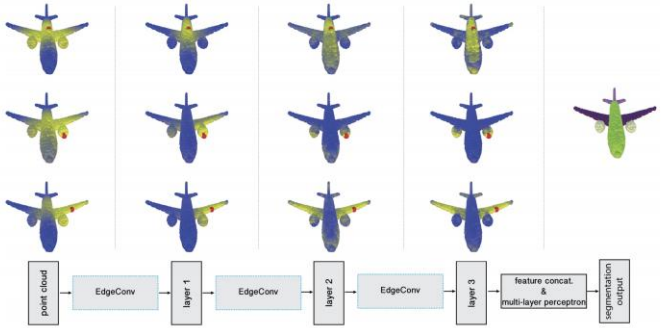


Hamilton, W.L., Ying, R. and Leskovec, J., 2017. Inductive Representation Learning on Large Graphs.

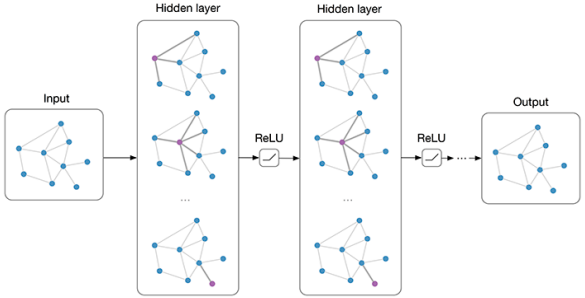
Most SOTA GCN models are no deeper than 3 or 4 layers.



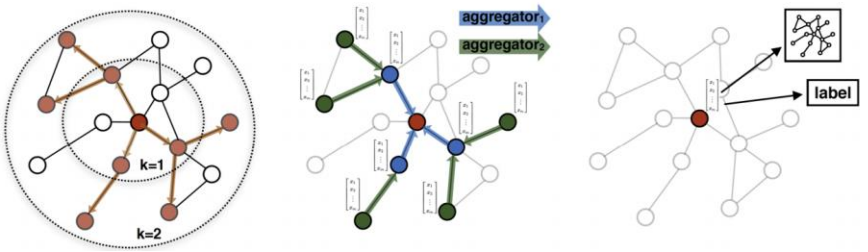
Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. and Bengio, Y., 2018. Graph Attention Networks.



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.



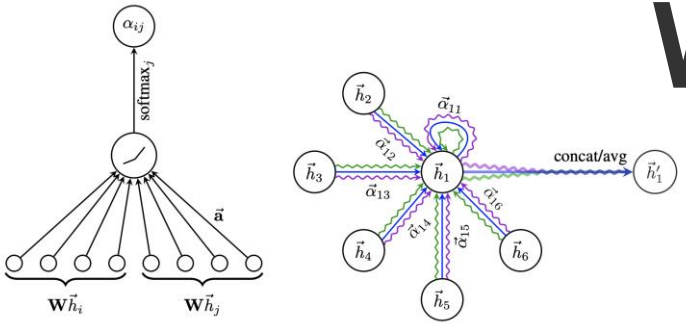
Kipf, T.N. and Welling, M., 2016. Semi-Supervised Classification with Graph Convolutional Networks.



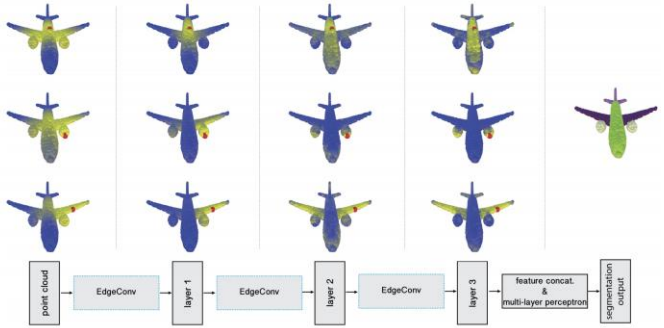
Hamilton, W.L., Ying, R. and Leskovec, J., 2017. Inductive Representation Learning on Large Graphs.

Most SOTA GCN models are no deeper than 3 or 4 layers.

Why?



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.

Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning

Qimai Li[1], Zhichao Han[12], Xiao-Ming Wu[1]*
[1]The Hong Kong Polytechnic University
[2]ETH Zurich
csqmli@comp.polyu.edu.hk, zhhan@student.ethz.ch, xiao-ming.wu@polyu.edu.hk

Abstract

Many interesting problems in machine learning are being revisited with new deep learning tools. For graph-based semi-supervised learning, a recent important development is graph convolutional networks (GCNs), which nicely integrate local vertex features and graph topology in the convolutional layers. Although the GCN model compares favorably with other state-of-the-art methods, its mechanisms are not clear and it still requires considerable amount of labeled data for validation and model selection.

In this paper, we develop deeper insights into the GCN model and address its fundamental limits. First, we show that the graph convolution of the GCN model is actually a special form of Laplacian smoothing, which is the key reason why GCNs work, but it also brings potential concerns of over-smoothing with many convolutional layers. Second, to overcome the limits of the GCN model with shallow architectures, we propose both co-training and self-training approaches to train GCNs. Our approaches significantly improve GCNs in learning with very few labels, and exempt them from requiring additional labels for validation. Extensive experiments on benchmarks have verified our theory and proposals.

amount of *unlabeled* data can be utilized to train with typically a small amount of labeled data.

Many researches have shown that leveraging unlabeled data in training can improve learning accuracy significantly if used properly (Zhu and Goldberg 2009). The key issue is to maximize the effective utilization of structural and feature information of unlabeled data. Due to the powerful feature extraction capability and recent success of deep neural networks, there have been some successful attempts to revisit semi-supervised learning with neural-network-based models, including ladder network (Rasmus et al. 2015), semi-supervised embedding (Weston et al. 2008), planetoid (Yang, Cohen, and Salakhutdinov 2016), and graph convolutional networks (Kipf and Welling 2017).

The recently developed graph convolutional neural networks (GCNNs) (Defferrard, Bresson, and Vandergheynst 2016) is a successful attempt of generalizing the powerful convolutional neural networks (CNNs) in dealing with Euclidean data to modeling graph-structured data. In their pilot work (Kipf and Welling 2017), Kipf and Welling proposed a simplified type of GCNNs, called graph convolu-

Influence of the Parameters. A common parameter of our methods is the number of newly added labels. Adding too many labels will introduce noise, but with too few labels we cannot train a good GCN classifier. As described in the end of Section 4, we can estimate the lower bound of the total number of labels η needed to train a GCN by solving $(\hat{d})^\tau * \eta \approx n$. We use 3η in our experiments. Actually, we found that 2η, 3η and 4η perform similarly in the experiments. We follow Kipf and Welling to set the number of convolutional layers as 2. We also observed in the experiments that 2-layer GCNs performed the best. When the number of convolutional layers grows, the classification accuracy decreases drastically, which is probably due to overfitting.

Computational Cost. For Co-Training, the overhead is the computational cost of the random walk model, which requires solving a sparse linear system. In our experiments, the time is negligible on Cora and CiteSeer as there are only a few thousand vertices. On PubMed, it takes less than 0.38

Over smoothing: the features of vertices within each connected component of the graph will converge to the same values

Graph Neural Networks: A Review of Methods and Applications

Jie Zhou*, Ganqu Cui*, Zhengyan Zhang*, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun

Abstract—Lots of learning tasks require dealing with graph data which contains rich relation information among elements. Modeling physics system, learning molecular fingerprints, predicting protein interface, and classifying diseases require that a model learns from graph inputs. In other domains such as learning from non-structural data like texts and images, reasoning on extracted structures, like the dependency tree of sentences and the scene graph of images, is an important research topic which also needs graph reasoning models. Graph neural networks (GNNs) are connectionist models that capture the dependence of graphs via message passing between the nodes of graphs. Unlike standard neural networks, graph neural networks retain a state that can represent information from its neighborhood with arbitrary depth. Although the primitive GNNs have been found difficult to train for a fixed point, recent advances in network architectures, optimization techniques, and parallel computation have enabled successful learning with them. In recent years, systems based on graph convolutional network (GCN) and gated graph neural network (GGNN) have demonstrated ground-breaking performance on many tasks mentioned above. In this survey, we provide a detailed review over existing graph neural network models, systematically categorize the applications, and propose four open problems for future research.

Index Terms—Deep Learning, Graph Neural Network

Shallow Structure limits the potentials of GCNs

4 Open problems

Although GNNs have achieved great success in different fields, it is remarkable that GNN models are not good enough to offer satisfying solutions for any graph in any condition. In this section, we will state some open problems for further researches.

Shallow Structure Traditional deep neural networks can stack hundreds of layers to get better performance, because deeper structure has more parameters, which improve the expressive power significantly. However, graph neural networks are always shallow, most of which are no more than three layers. As experiments in [62] show, stacking multiple GCN layers will result in over-smoothing, that is to say, all vertices will converge to the same value. Although some researchers have managed to tackle this problem [45], [62], it remains to be the biggest limitation of GNN. Designing real deep GNN is an exciting challenge for future research, and will be a considerable contribution to the understanding of GNN.

A Comprehensive Survey on Graph Neural Networks

Zonghan Wu, Shirui Pan, *Member, IEEE*, Fengwen Chen, Guodong Long,
Chengqi Zhang, *Senior Member, IEEE*, Philip S. Yu, *Fellow, IEEE*

Abstract—Deep learning has revolutionized many machine learning tasks in recent years, ranging from image classification and video processing to speech recognition and natural language understanding. The data in these tasks are typically represented in the Euclidean space. However, there is an increasing number of applications where data are generated from non-Euclidean domains and are represented as graphs with complex relationships and interdependency between objects. The complexity of graph data has imposed significant challenges on existing machine learning algorithms. Recently, many studies on extending deep learning approaches for graph data have emerged. In this survey, we provide a comprehensive overview of graph neural networks (GNNs) in data mining and machine learning fields. We propose a new taxonomy to divide the state-of-the-art graph neural networks into different categories. With a focus on graph convolutional networks, we review alternative architectures that have recently been developed; these learning paradigms include graph attention networks, graph autoencoders, graph generative networks, and graph spatial-temporal networks. We further discuss the applications of graph neural networks across various domains and summarize the open source codes and benchmarks of the existing algorithms on different learning tasks. Finally, we propose potential research directions in this fast-growing field.

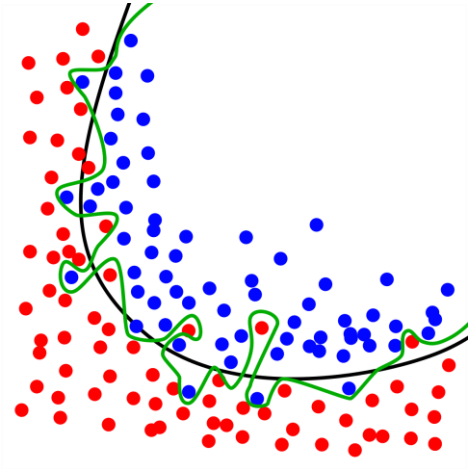
Index Terms—Deep Learning, graph neural networks, graph convolutional networks, graph representation learning, graph autoencoder, network embedding

Receptive Field The receptive field of a node refers to a set of nodes including the central node and its neighbors. The number of neighbors of a node follows a power law distribution. Some nodes may only have one neighbor, while other nodes may neighbors as many as thousands. Though sampling strategies have been adopted [25], [27], [28], how to select a representative receptive field of a node remains to be explored.

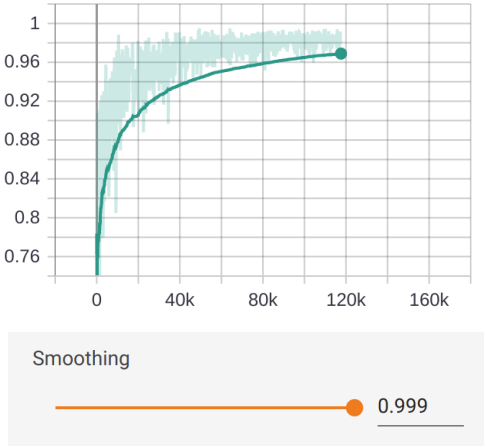
Scalability Most graph neural networks do not scale well for large graphs. The main reason for this is when stacking multiple layers of a graph convolution, a node's final state involves a large number of its neighbors' hidden states, leading to the high complexity of backpropagation. While several approaches try to improve their model efficiency by fast sampling [48], [49] and sub-graph training [25], [28], they are still not scalable enough to handle deep architectures with large graphs.

Receptive Field; the high complexity of backpropagation

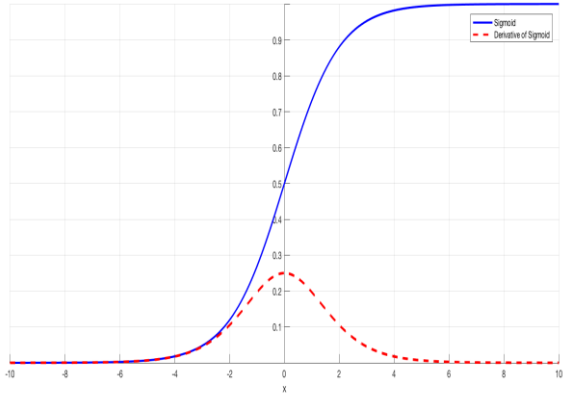
Why GCNs are limited to shallow structures?



Over-fitting



Over-smoothing

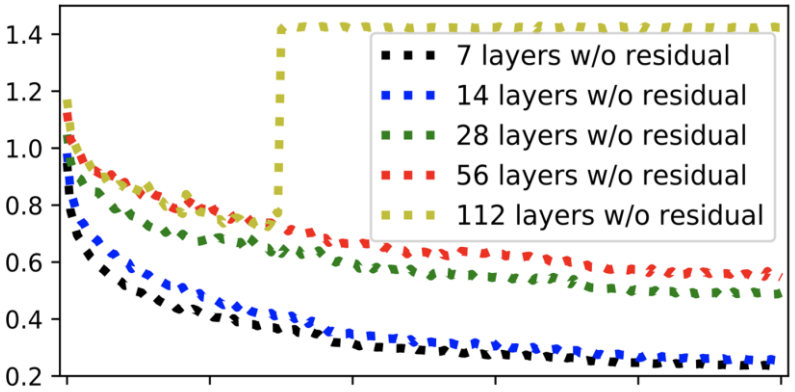


Vanishing Gradient

Figures from https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

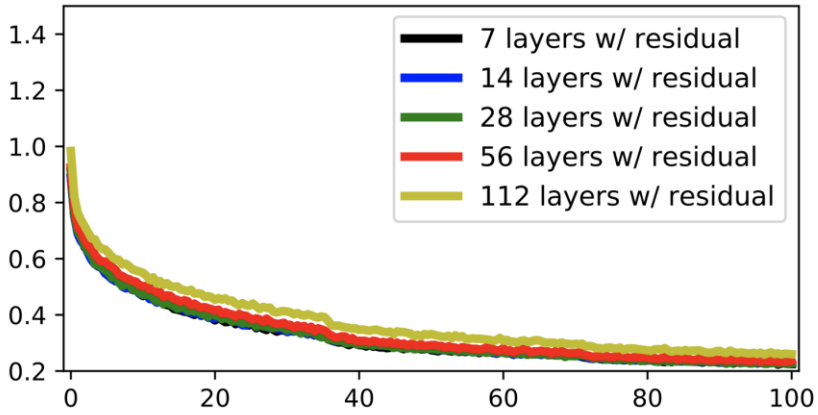
Training Loss of GCNs with varying depth

Deeper GCNs don’t converge well.



PlainGCNs

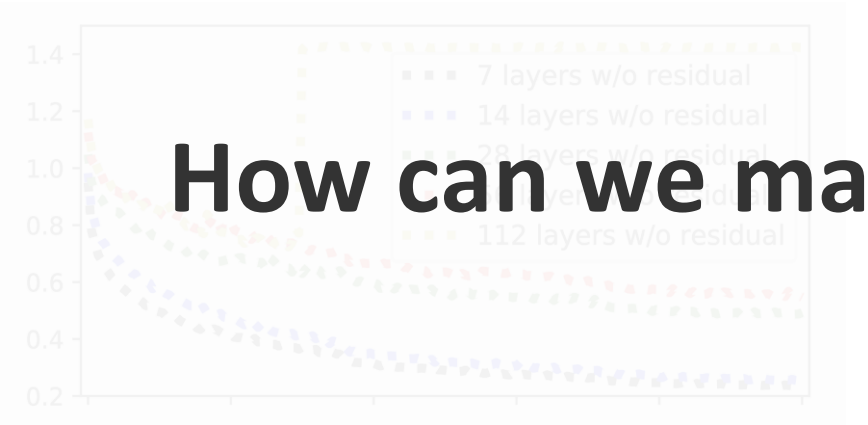
Even a 112-layer deep GCN converges well!!!



ResGCNs

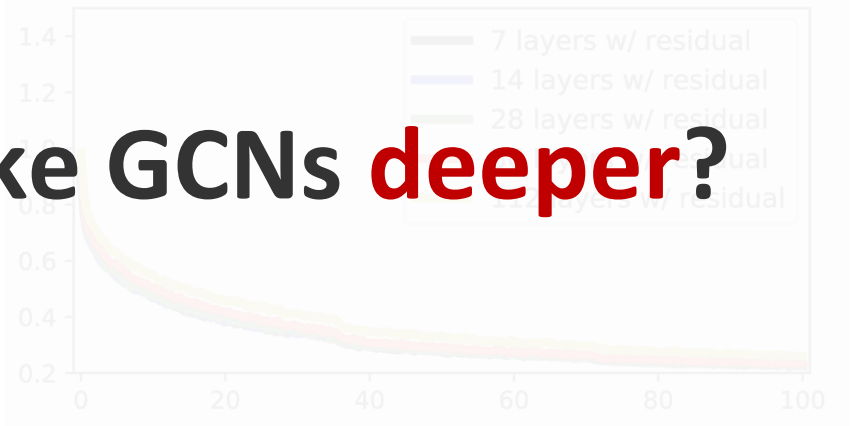
Training Loss of GCNs with varying depth

Deeper GCNs don't converge well.



PlainGCNs

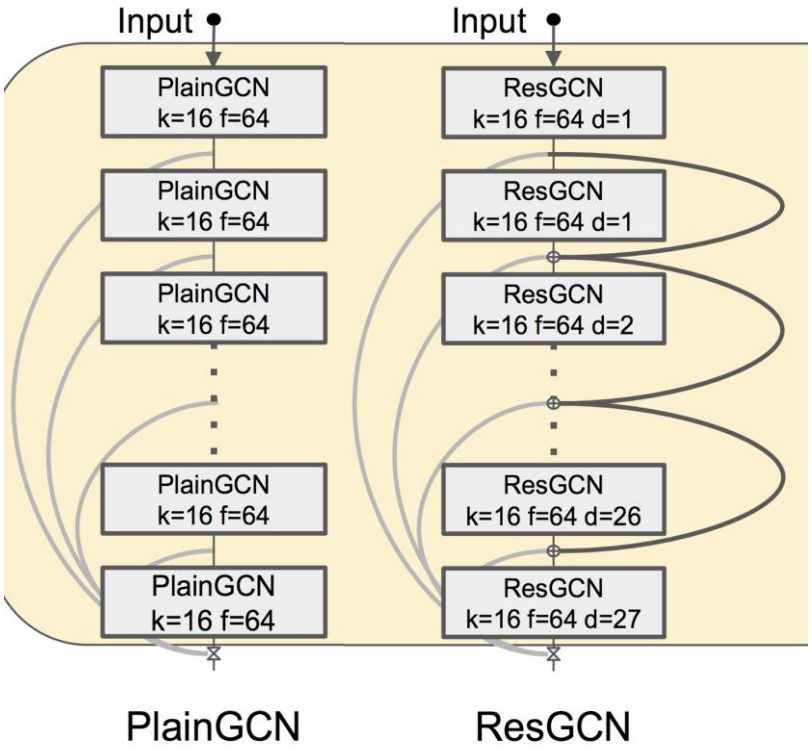
Even a 112-layer deep GCN converges well!!!



ResGCNs

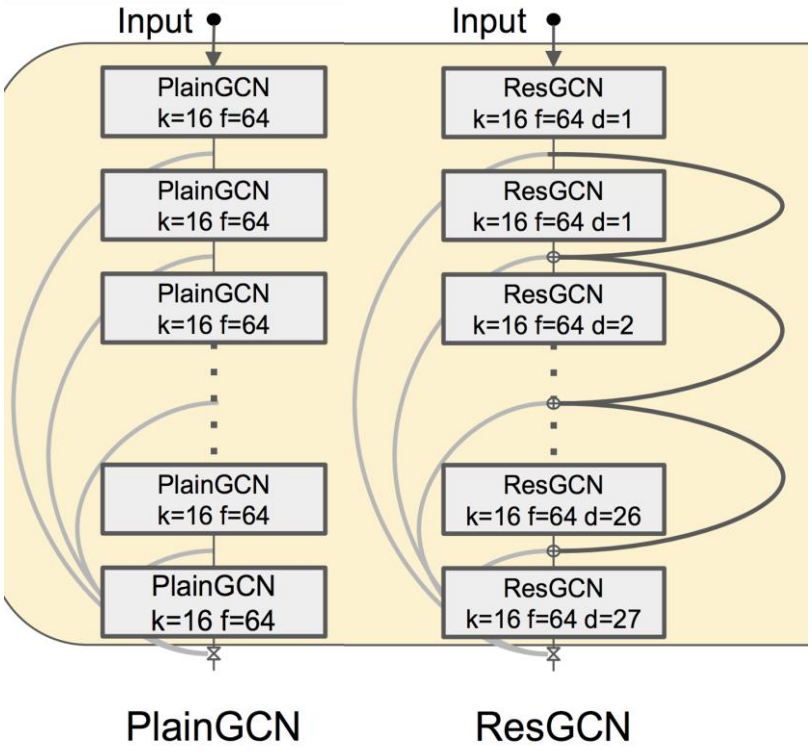
How can we make GCNs deeper?

Residual Graph Connections



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

Residual Graph Connections



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

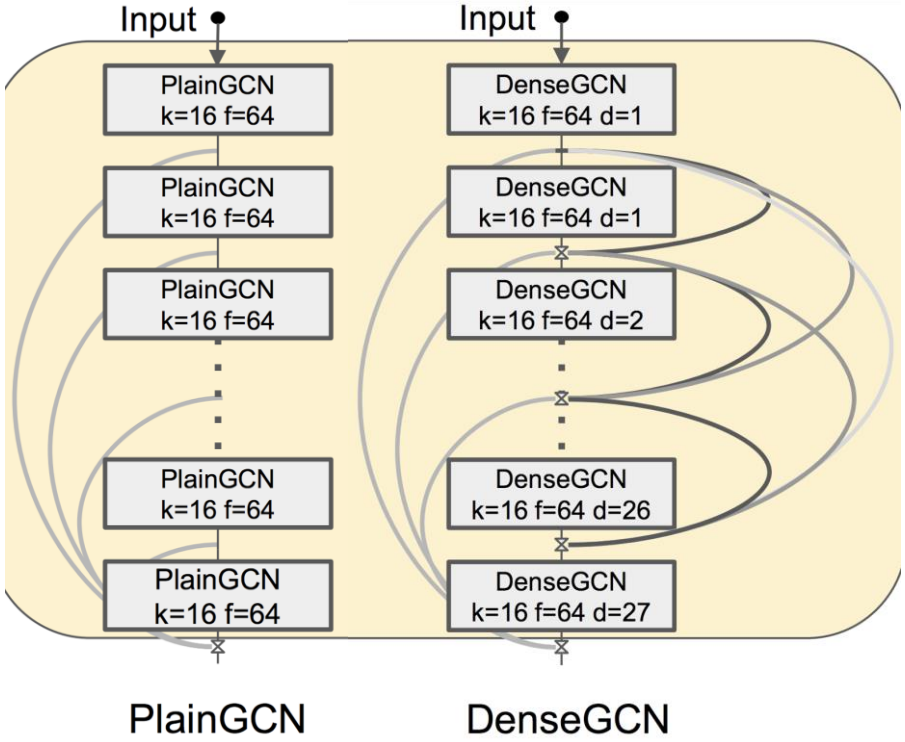
An example: ResMRGCN

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

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

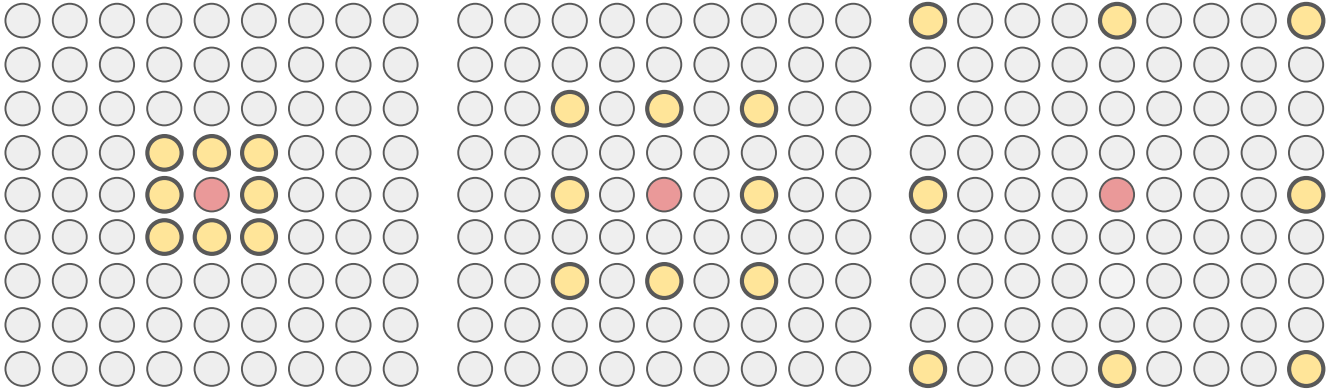
$$h_{v_{l+1}} = h^{res}_{v_{l+1}} + h_{v_l}. \quad \text{Skip connection}$$

Dense Graph Connections

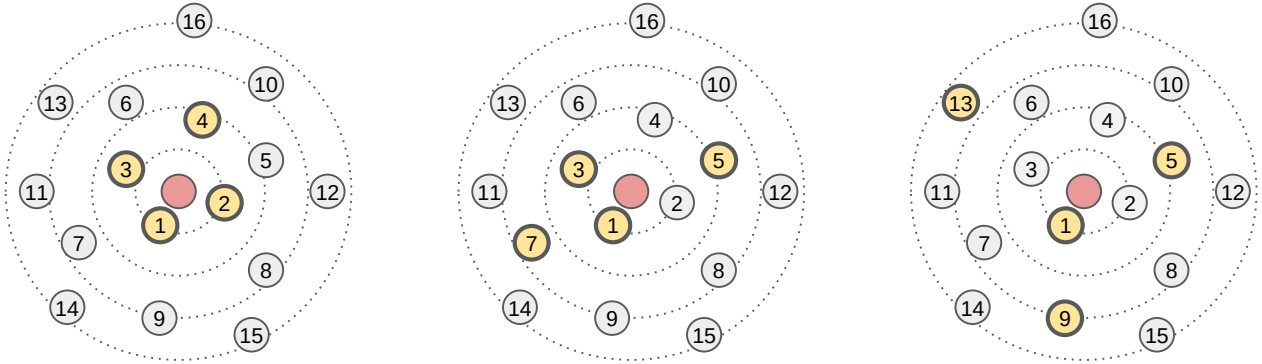


$$\begin{aligned}\mathcal{G}_{l+1} &= \mathcal{H}(\mathcal{G}_l, \mathcal{W}_l) \\ &= \mathcal{T}(\mathcal{F}(\mathcal{G}_l, \mathcal{W}_l), \mathcal{G}_l) \\ &= \mathcal{T}(\mathcal{F}(\mathcal{G}_l, \mathcal{W}_l), ..., \mathcal{F}(\mathcal{G}_0, \mathcal{W}_0), \mathcal{G}_0).\end{aligned}$$

Dilated Graph Convolutions

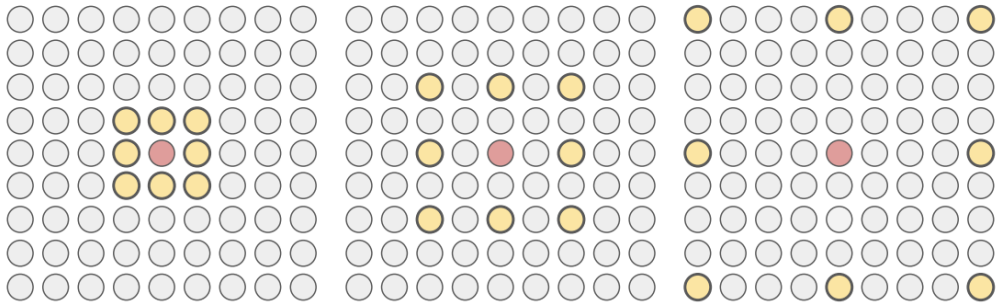


Dilated Convolution on a regular graph, e.g. 2D image



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

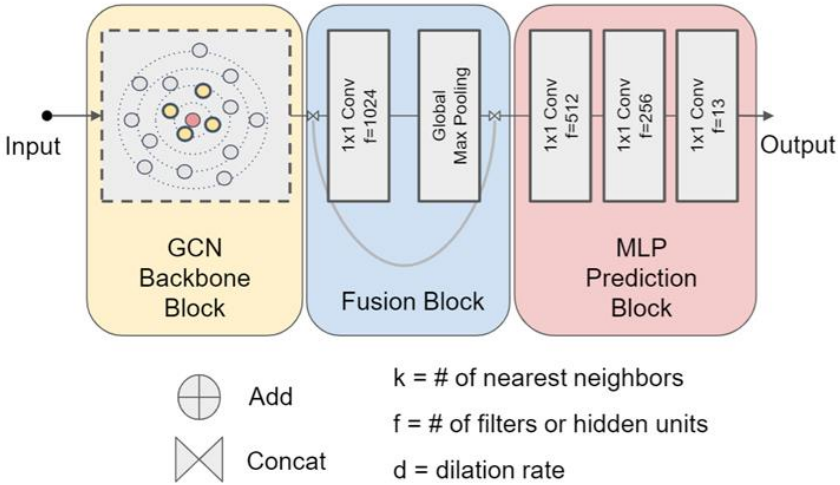
Dilated Graph Convolutions

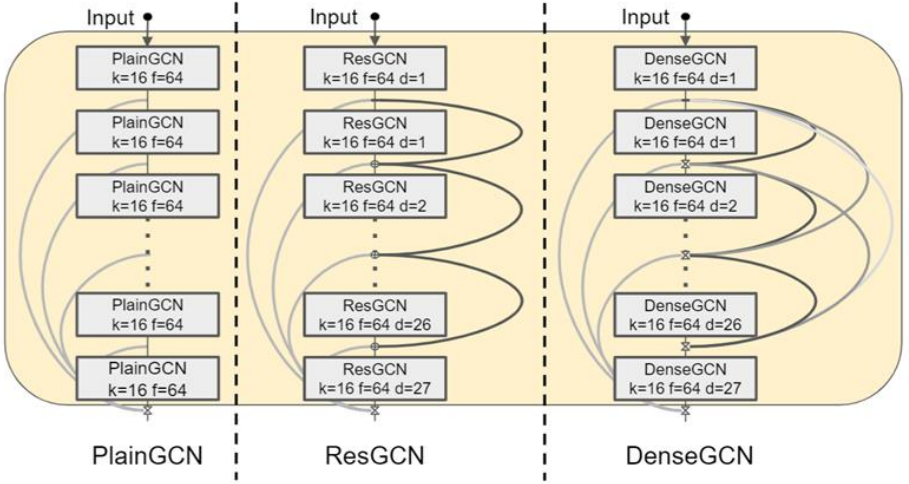


$$\mathcal{N}^{(d)}(v) = \{u_1, u_{1+d}, u_{1+2d}, ..., u_{1+(k-1)d}\}.$$

d = dilation rate

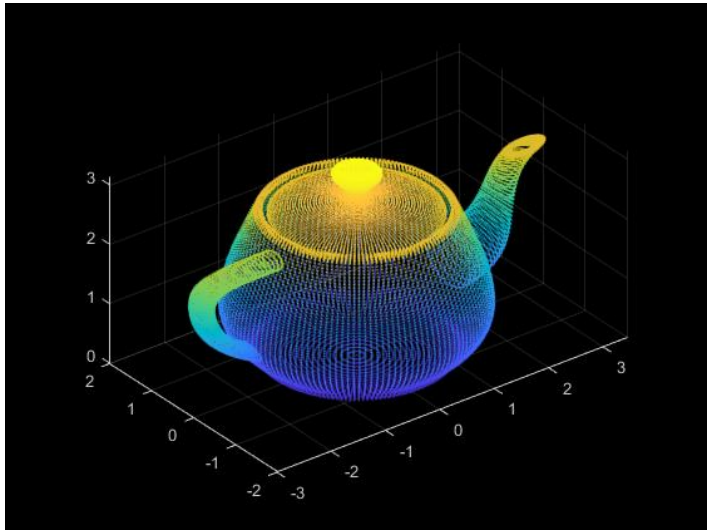
Deep Graph Convolutional Networks (GCNs)





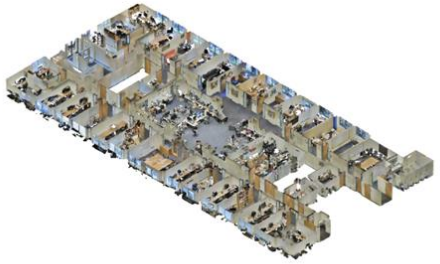
Experiments

Graph Learning on 3D Point Clouds

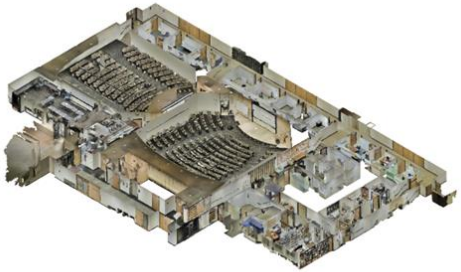


- Point clouds are unordered and irregular
- Represented by 3D coordinates and extra features such as color, surface normal, etc.
- We use k-NN to construct the directed dynamic edges between points at every GCN layer in the feature space.

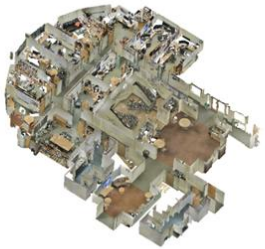
Stanford 3D Large-Scale Indoor Spaces Dataset



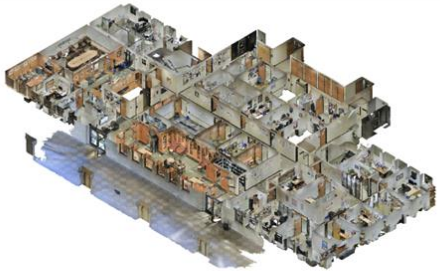
Area 1



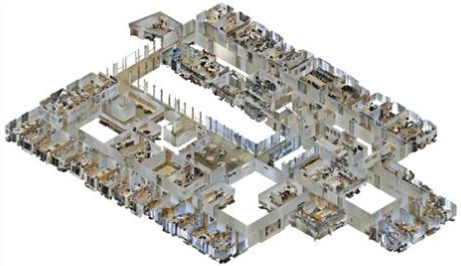
Area 2



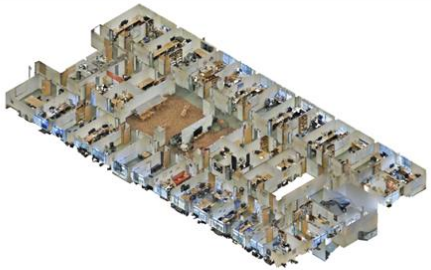
Area 3



Area 4



Area 5



Area 6

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

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**

Table 1. 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**

Table 2. 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.

Consistent improvements across all the classes.

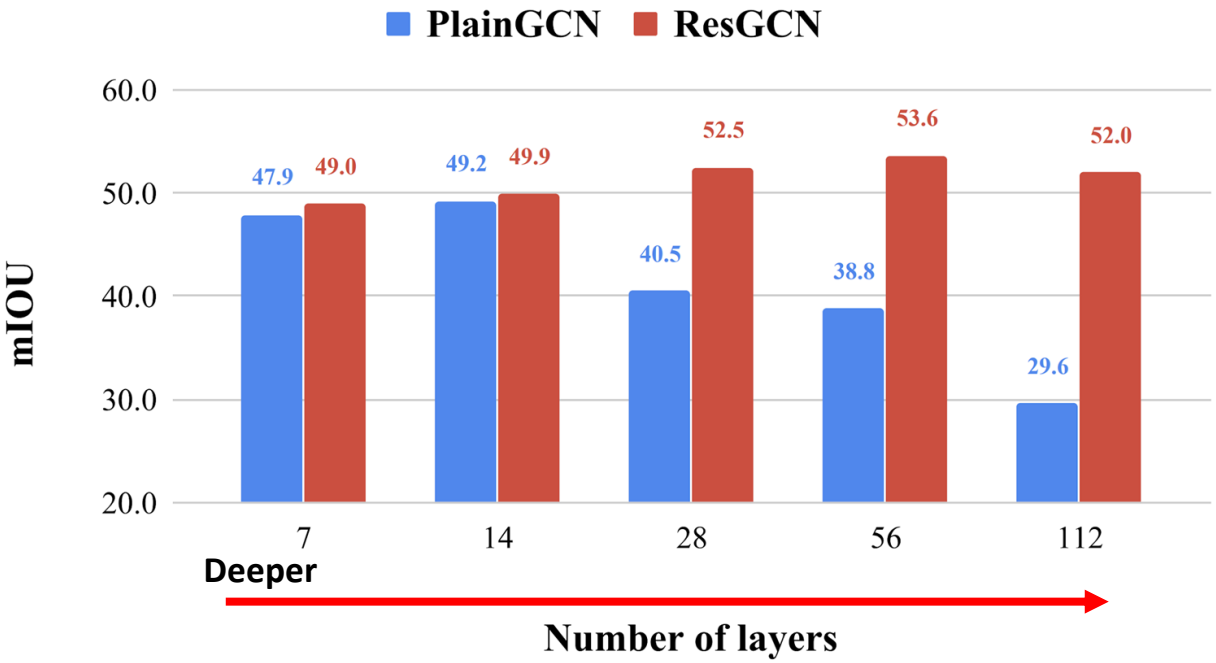
Class	DGCNN [6]	ResGCN-28 (*Ours*)
ceiling	92.7	**93.1**
floor	93.6	**95.3**
wall	77.5	**78.2**
beam	32.0	**33.9**
column	36.3	**37.4**
window	52.5	**56.1**
door	63.7	**68.2**
table	61.1	**64.9**
chair	60.2	**61.0**
sofa	20.5	**34.6**
bookcase	47.7	**51.5**
board	42.7	**51.1**
clutter	51.5	**54.4**
mIOU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

~ 4% boost in mIOU.

Table 2. 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



Ablation Study

skip connections, dilation, depth, width, # of NNs

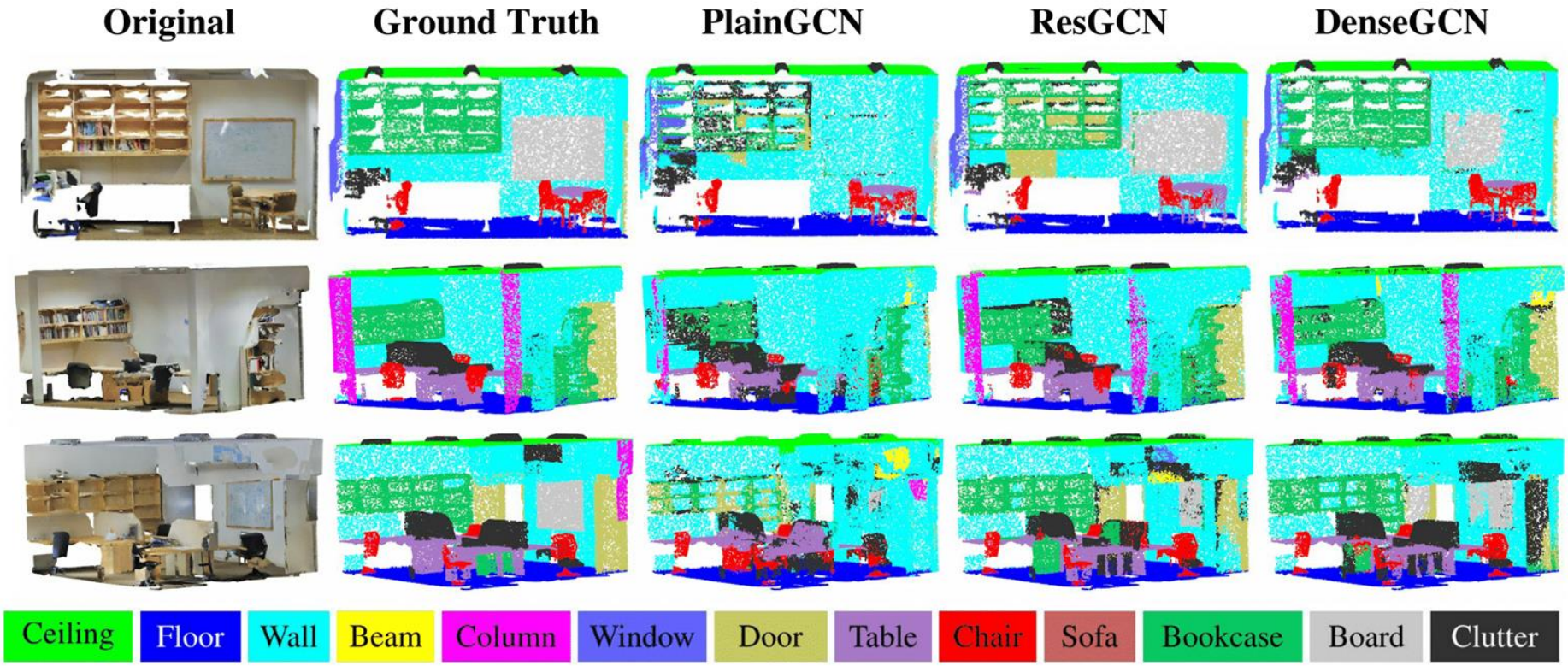
Ablation Study

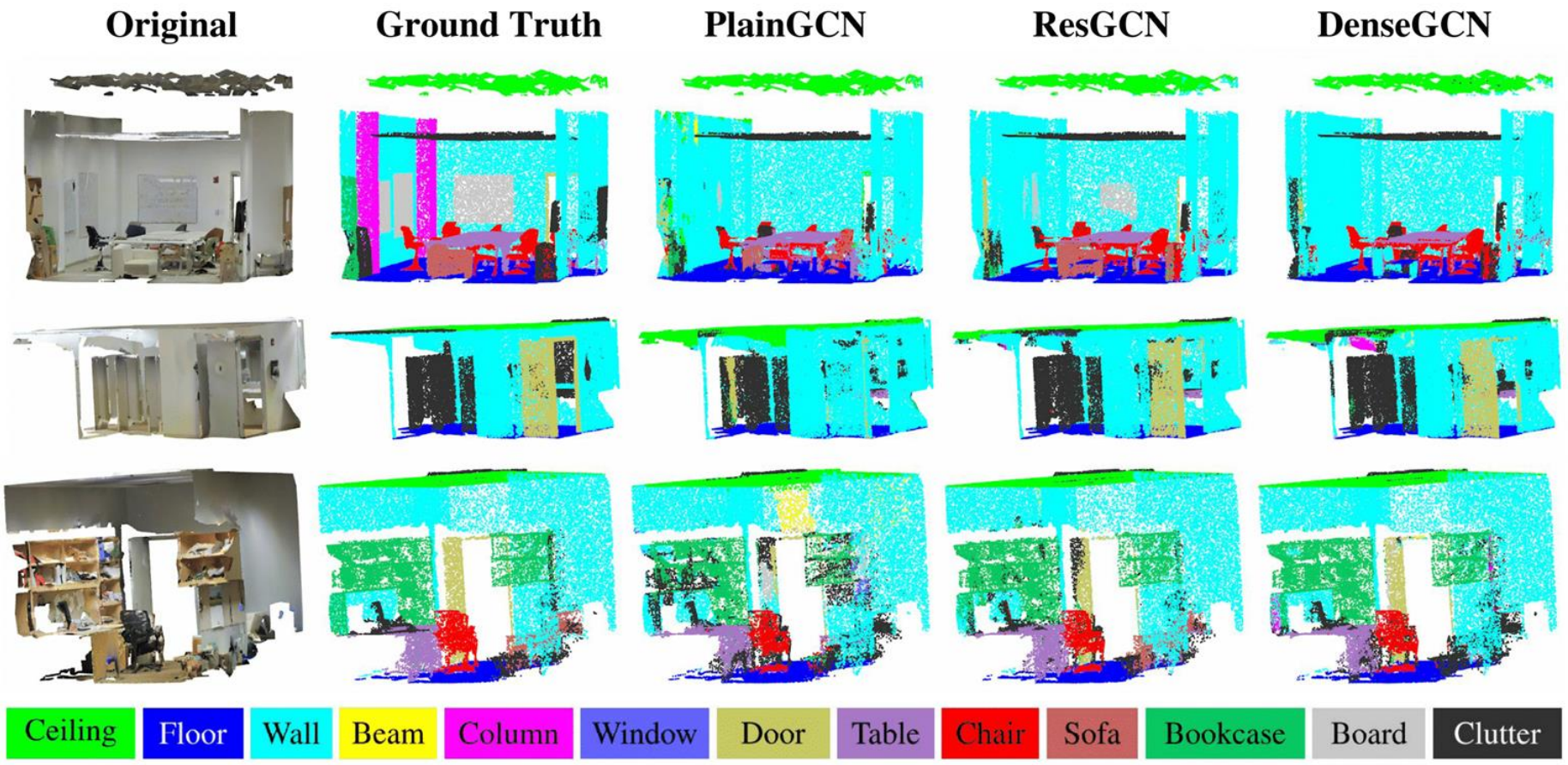
Ablation	Model	mIoU	ΔmIoU	dynamic	connection	dilation	stochastic	# NNs	# filters	# layers
Reference	*ResGCN-28*	**52.49**	0.00	✓	⊕	✓	✓	16	64	28
Dilation		51.98	-0.51	✓	⊕	✓		16	64	28
		49.64	-2.85	✓	⊕			16	64	28
	PlainGCN-28	**40.31**	-12.18	✓				16	64	28
Fixed k-NN		48.38	-4.11		⊕			16	64	28
		43.43	-9.06					16	64	28
	DenseGCN-28	**51.27**	-1.22	✓	⋈	✓	✓	8	32	28
Connections		40.47	-12.02	✓		✓	✓	16	64	28
		38.79	-13.70	✓		✓	✓	8	64	56
		49.23	-3.26	✓		✓	✓	16	64	14
		47.92	-4.57	✓		✓	✓	16	64	7
Neighbors		49.98	-2.51	✓	⊕	✓	✓	8	64	28
		49.22	-3.27	✓	⊕	✓	✓	4	64	28
Depth	*ResGCN-56*	**53.64**	1.15	✓	⊕	✓	✓	8	64	56
	ResGCN-14	49.90	-2.59	✓	⊕	✓	✓	16	64	14
	ResGCN-7	48.95	-3.53	✓	⊕	✓	✓	16	64	7
Width	*ResGCN-28W*	**53.78**	1.29	✓	⊕	✓	✓	8	128	28
		49.18	-3.31	✓	⊕	✓	✓	32	32	28
		48.80	-3.69	✓	⊕	✓	✓	16	32	28
		45.62	-6.87	✓	⊕	✓	✓	16	16	28

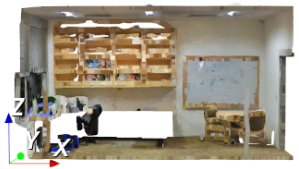
Table 3. Ablation study on area 5 of S3DIS.

Qualitative Results

Visualizations on S3DIS


Original
Ground Truth
PlainGCN
ResGCN
DenseGCN
Ceiling
Floor
Wall
Beam
Column
Window
Door
Table
Chair
Sofa
Bookcase
Board
Clutter


Original
Ground Truth
PlainGCN
ResGCN
DenseGCN
Ceiling
Floor
Wall
Beam
Column
Window
Door
Table
Chair
Sofa
Bookcase
Board
Clutter



Original

Ground Truth

ResGCN-28

1/2x filters

1/4x filters

Reduce Network Width

Original

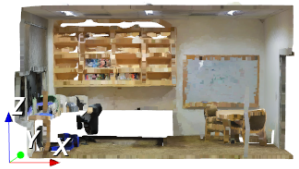
Ground Truth

ResGCN-28

1/2x NNs

1/4x NNs

Reduce Kernel Size

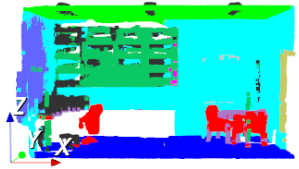


Original

Ground Truth

ResGCN-28

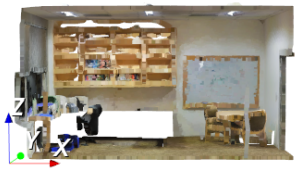
1/2x layers



1/4x layers

Reduce Network Depth

Ceiling Floor Wall Beam Column Window Door Table Chair Sofa Bookcase Board Clutter



Original

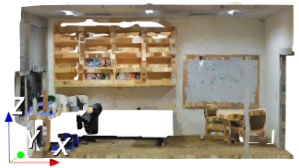
Ground Truth

ResGCN-28

w/o stochastic

w/o dilation

No Dilation



Original

Ground Truth

ResGCN-28

ResGCN-28W

ResGCN-56

Wider

Deeper

Ceiling | Floor | Wall | Beam | Column | Window | Door | Table | Chair | Sofa | Bookcase | Board | Clutter

More Results

GCN variants

- ResEdgeConv
- ResGraphSAGE
- ResGIN
- ResMRGCN

Model	mIoU	ΔmIoU	dynamic	connection	dilation	stochastic	# NNs	# filters	# layers
ResEdgeConv-28	**52.49**	0.00	✓	⊕	✓	✓	16	64	28
PlainGCN-28	**40.31**	-12.18	✓				16	64	28
ResGraphSAGE-28	**49.20**	-3.29	✓	⊕	✓	✓	16	64	28
ResGraphSAGE-N-28	**49.02**	-3.47	✓	⊕	✓	✓	16	64	28
ResGIN-ϵ-28	**42.81**	-9.68	✓	⊕	✓	✓	16	64	28
ResMRGCN-28	**51.17**	-1.32	✓	⊕	✓	✓	16	64	28

Table 4. Comparisons of Deep GCNs variants on area 5 of S3DIS.

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

ResEdgeConv

$$h^{res}_{v_{l+1}} = mlp\left((1+\epsilon)\cdot h_{v_l} + sum(\{h_{u_l} | u_l \in \mathcal{N}^{(d)}(v_l)\})\right),$$
$$h_{v_{l+1}} = h^{res}_{v_{l+1}} + h_{v_l}.$$

ResGIN

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

ResGraphSAGE

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

ResMRGCN

More Results

Wider

Deeper

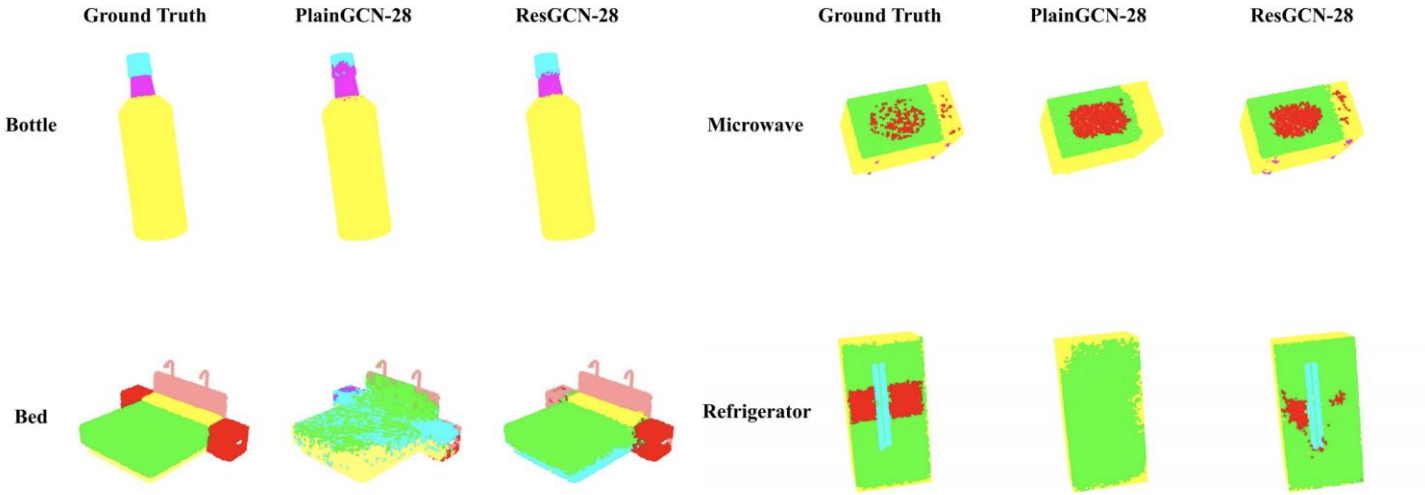
Number of filters	32	64	128	256
PlainMRGCN-3	95.84	97.60	98.58	99.13
PlainMRGCN-7	97.35	98.69	99.22	**99.38**
PlainMRGCN-14	97.55	99.02	99.31	99.34
PlainMRGCN-28	98.09	99.00	99.02	99.31
PlainMRGCN-56	92.70	97.43	97.31	97.61
PlainMRGCN-112	60.75	71.97	89.69	91.50
ResMRGCN-3	96.04	97.60	98.53	99.09
ResMRGCN-7	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	-	-	-

Table 5. Node classification of biological networks.

More Results

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**

Table 6. Comparison of DeepGCNs with state-of-the-art on PPI node classification.



Method	bed	bottle	chair	clock	dishw.	disp.	door	earph.	fauc.	knife	lamp	micro.	fridge	st. furn.	table	tr. can	vase
PointNet [33]	13.4	29.5	27.8	28.4	48.9	76.5	30.4	33.4	47.6	32.9	18.9	37.2	33.5	38.0	29.0	34.8	44.4
PointNet++ [34]	30.3	**41.4**	**39.2**	**41.6**	50.1	80.7	32.6	38.4	**52.4**	34.1	**25.3**	48.5	36.4	40.5	**33.9**	46.7	49.8
SpiderCNN [53]	**36.2**	32.2	30.0	24.8	50.0	80.1	30.5	37.2	44.1	22.2	19.6	43.9	39.1	**44.6**	20.1	42.4	32.4
ResGCN-28 (*Ours*)	35.2	36.8	33.8	32.6	**52.7**	**84.4***	**42.5***	**41.9**	49.7	**35.4***	20.0	**54.3**	**46.1***	42.5	14.8	**49.7**	**50.8**
PointCNN [54]	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

Table 7. Comparison of ResGCN-28 with other methods on PartNet Part Segmentation.

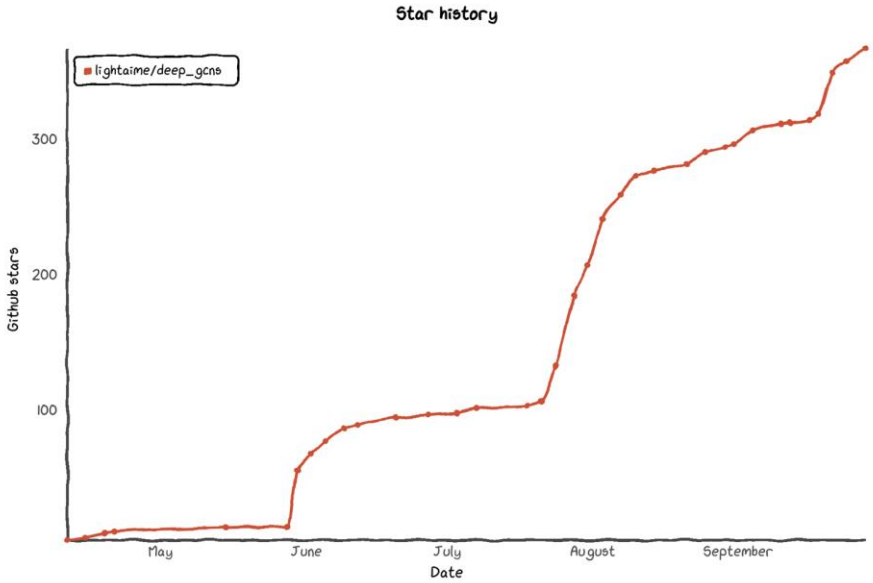
Conclusion

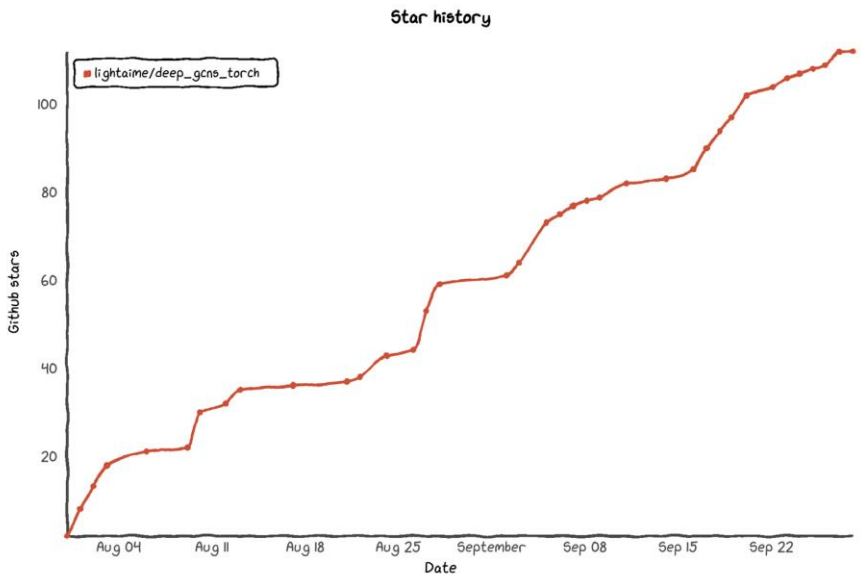
- Extensive experiments show that by adding skip connections to GCNs, we can alleviate the difficulty of training, which is the primary problem impeding GCNs to go deeper
- Dilated graph convolutions help to gain a larger receptive field without loss of resolution

Future Work

- Transfer other operators, e.g. deformable convolutions, pooling, normalization
- Transfer other architectures, e.g. feature pyramid architectures
- Different distance measures to compute dilated k-nn
- Construct graphs using different k at each layer
- Better dilation rate schedules

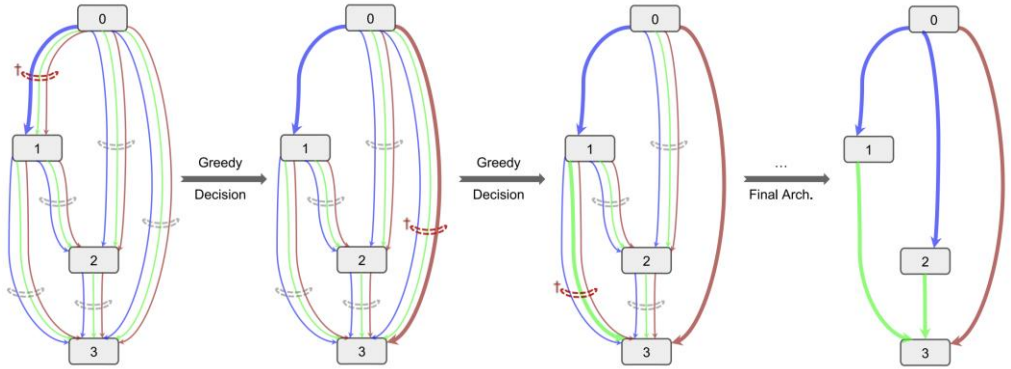
TensorFlow Repo ☆ 500+ Stars Pytorch Repo





https://www.deepgcns.org

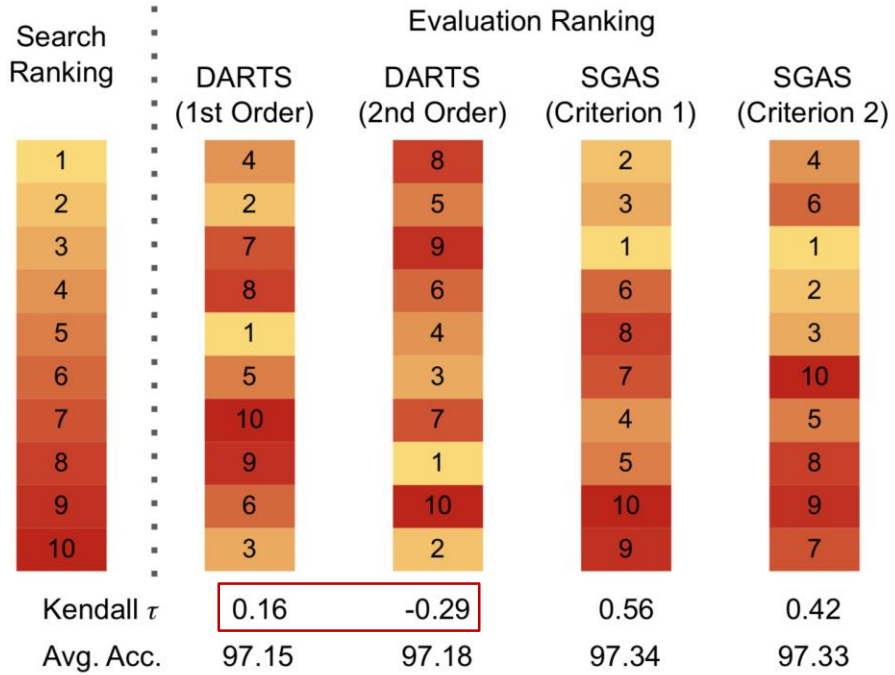
Follow-up works



SGAS: Sequential Greedy Architecture Search (arXiv 2019, Guohao Li et.al)

https://sites.google.com/kaust.edu.sa/sgas

Degenerate search-evaluation correlation problem



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

This harms the search performance!

Figure 1. Comparison of search-evaluation Kendall τ coefficients.

SGAS: 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.

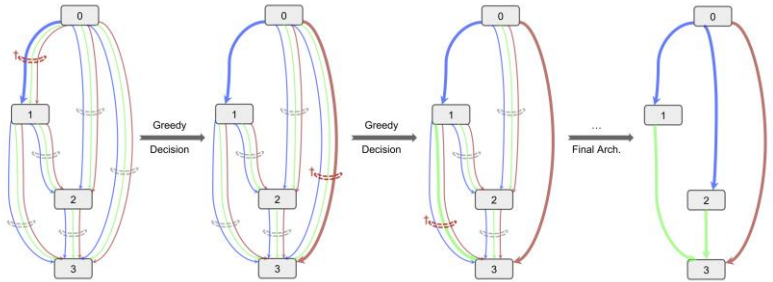


Figure 2. Illustration of Sequential Greedy Architecture Search.

SGAS: Sequential Greedy Architecture Search

We apply SGAS to search architectures for Convolutional Neural Networks (CNN) and Graph Convolutional Networks (GCN).

Extensive experiments show that SGAS is able to find SOTA architectures with minimal computational cost for tasks such as:

- image classification,
- point cloud classification,
- node classification in protein-protein interaction graphs.

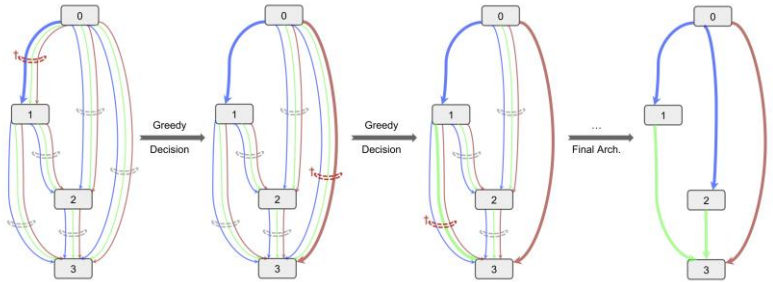


Figure 2. Illustration of Sequential Greedy Architecture Search.

SGAS: Sequential Greedy Architecture Search

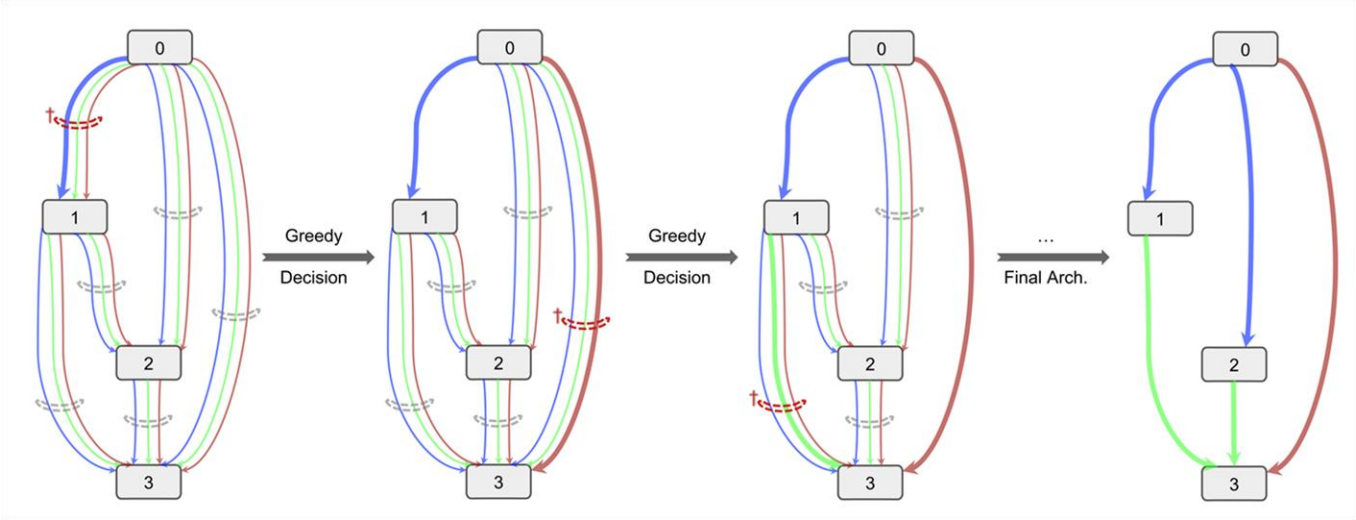


Figure 2. Illustration of Sequential Greedy Architecture Search.

SGAS: Sequential Greedy Architecture Search

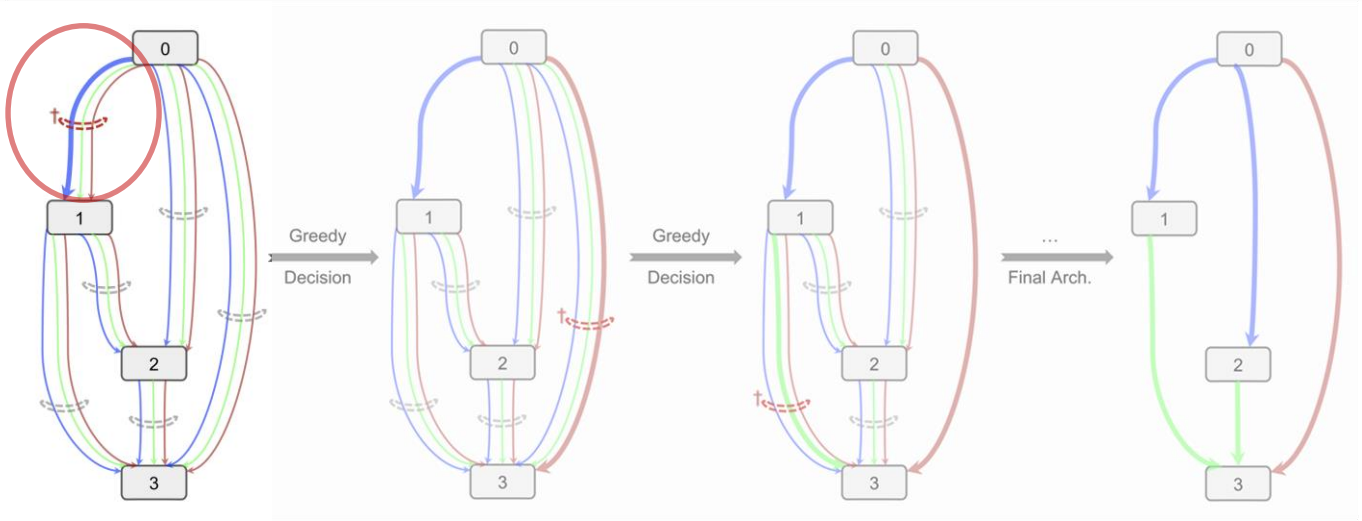


Figure 2. Illustration of Sequential Greedy Architecture Search.

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

SGAS: Sequential Greedy Architecture Search

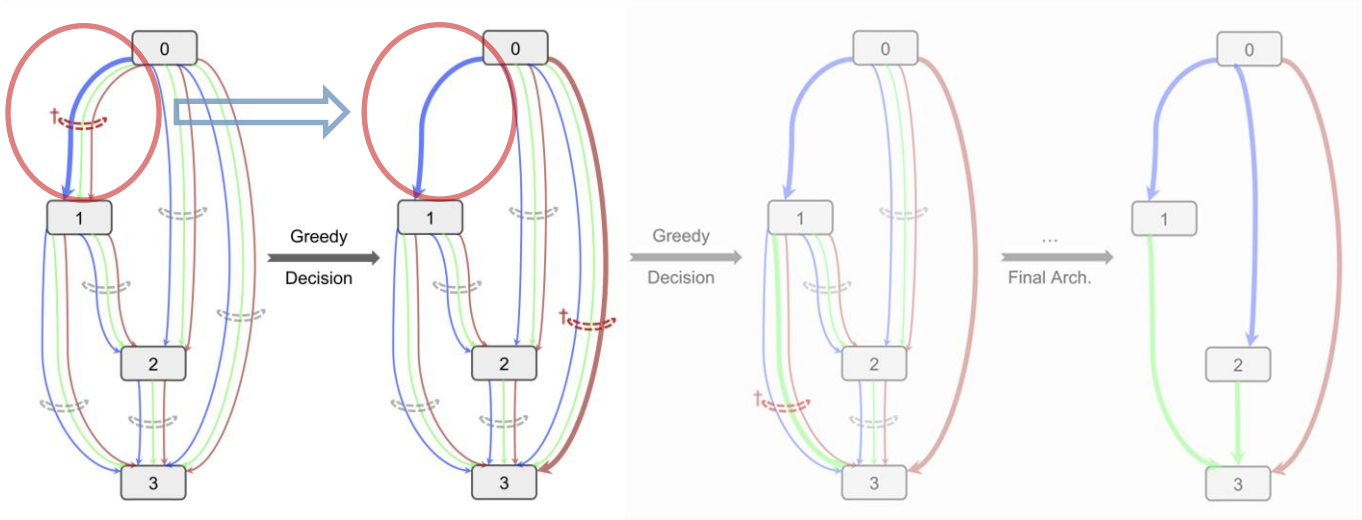


Figure 2. Illustration of Sequential Greedy Architecture Search.

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

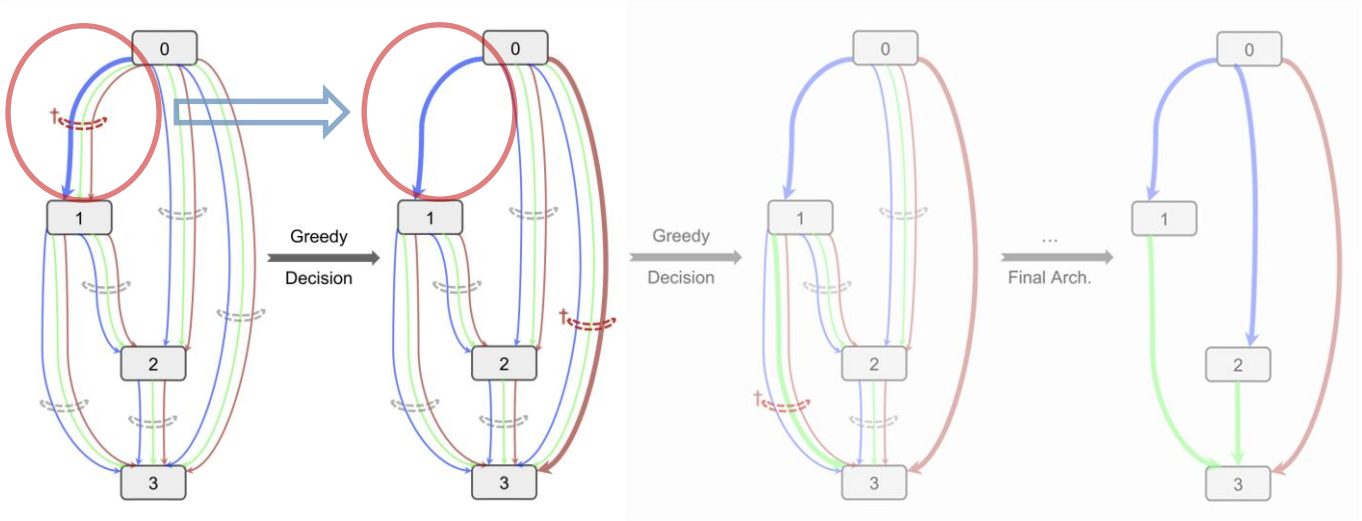


Figure 2. Illustration of Sequential Greedy Architecture Search.

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 $o^{(i^\dagger,j^\dagger)} = \text{argmax}_{o\in\mathcal{O}}\ \alpha_o^{(i^\dagger,j^\dagger)}$
3. Prune unchosen weights from $\mathcal{W}$, Remove $\alpha^{(i^\dagger,j^\dagger)}$ from $\mathcal{A}$

SGAS: Sequential Greedy Architecture Search

Repeat…

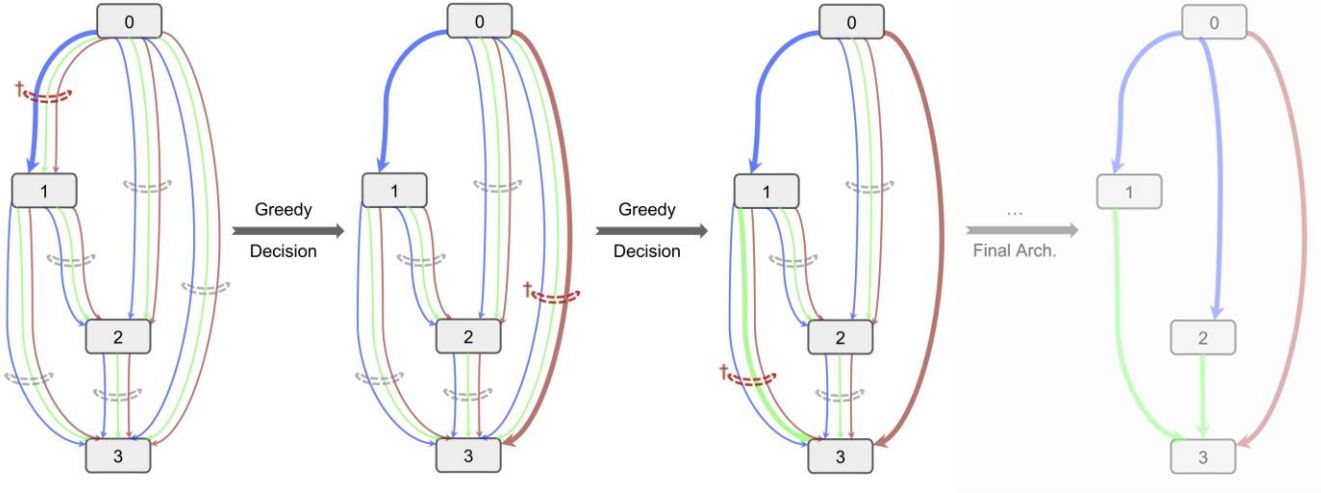


Figure 2. Illustration of Sequential Greedy Architecture Search.

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 $o^{(i^\dagger,j^\dagger)} = \text{argmax}_{o\in\mathcal{O}}\ \alpha_o^{(i^\dagger,j^\dagger)}$
3. Prune unchosen weights from $\mathcal{W}$, Remove $\alpha^{(i^\dagger,j^\dagger)}$ from $\mathcal{A}$

SGAS: Sequential Greedy Architecture Search

Repeat…

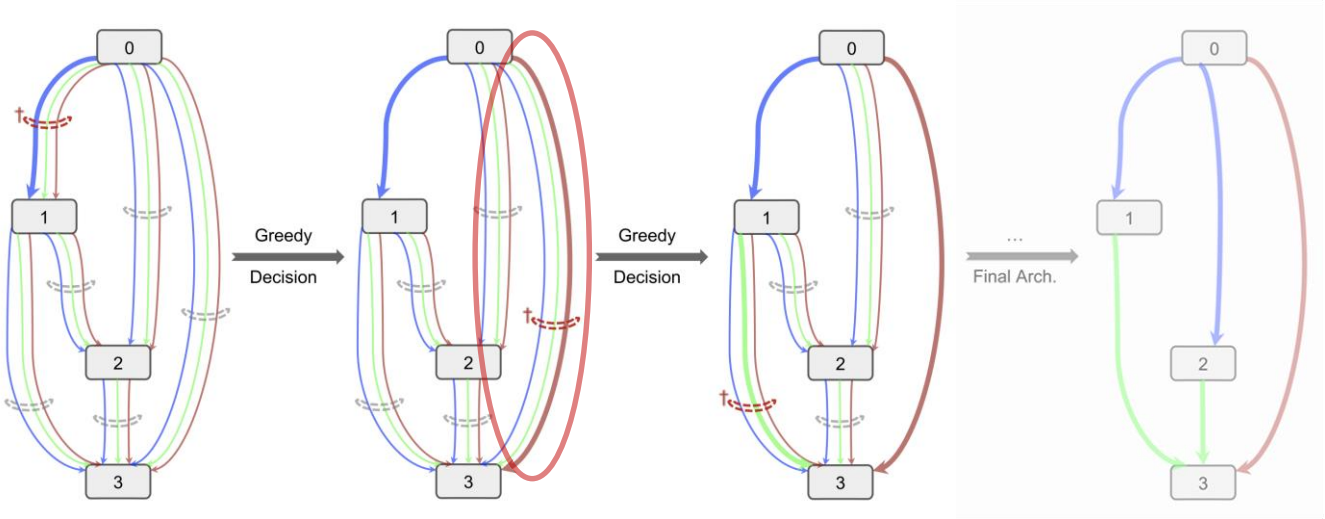


Figure 2. Illustration of Sequential Greedy Architecture Search.

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 $o^{(i^\dagger,j^\dagger)} = \text{argmax}_{o\in\mathcal{O}}\ \alpha_o^{(i^\dagger,j^\dagger)}$
3. Prune unchosen weights from $\mathcal{W}$, Remove $\alpha^{(i^\dagger,j^\dagger)}$ from $\mathcal{A}$

SGAS: Sequential Greedy Architecture Search

Repeat…

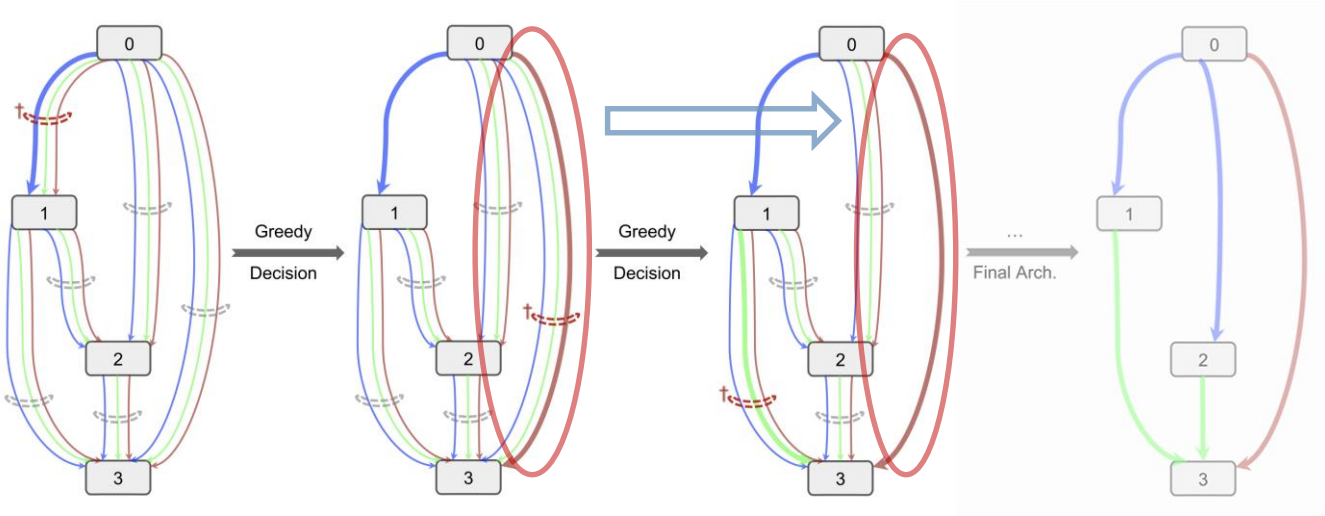


Figure 2. Illustration of Sequential Greedy Architecture Search.

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 $o^{(i^\dagger,j^\dagger)} = \text{argmax}_{o\in\mathcal{O}} \ \alpha_o^{(i^\dagger,j^\dagger)}$
3. Prune unchosen weights from $\mathcal{W}$, Remove $\alpha^{(i^\dagger,j^\dagger)}$ from $\mathcal{A}$

SGAS: Sequential Greedy Architecture Search

Until…

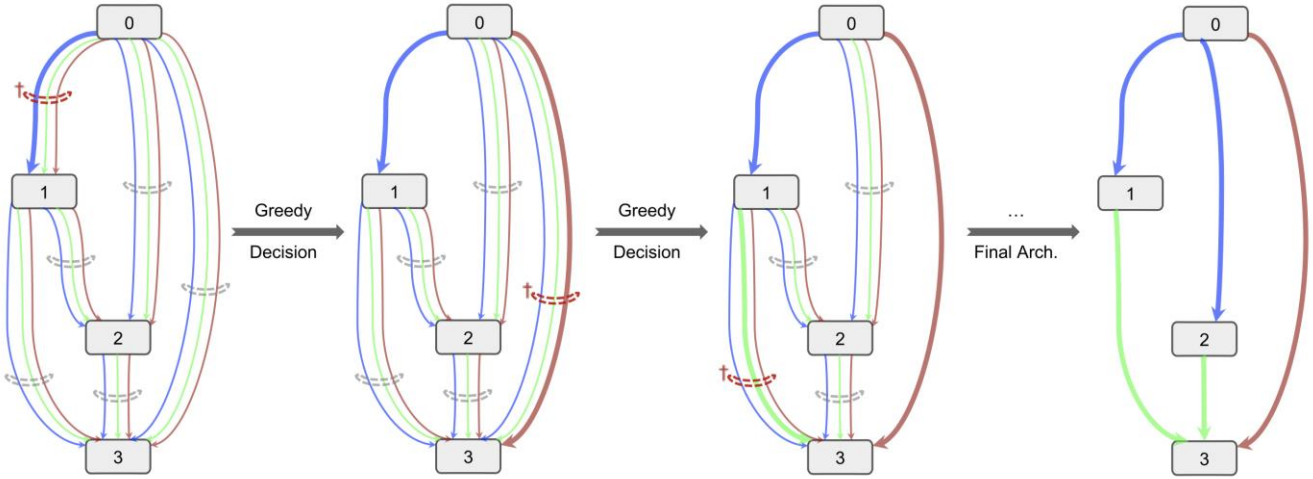


Figure 2. Illustration of Sequential Greedy Architecture Search.

A stand-alone architecture without weight sharing is obtained.

SGAS: Sequential Greedy Architecture Search

Until…

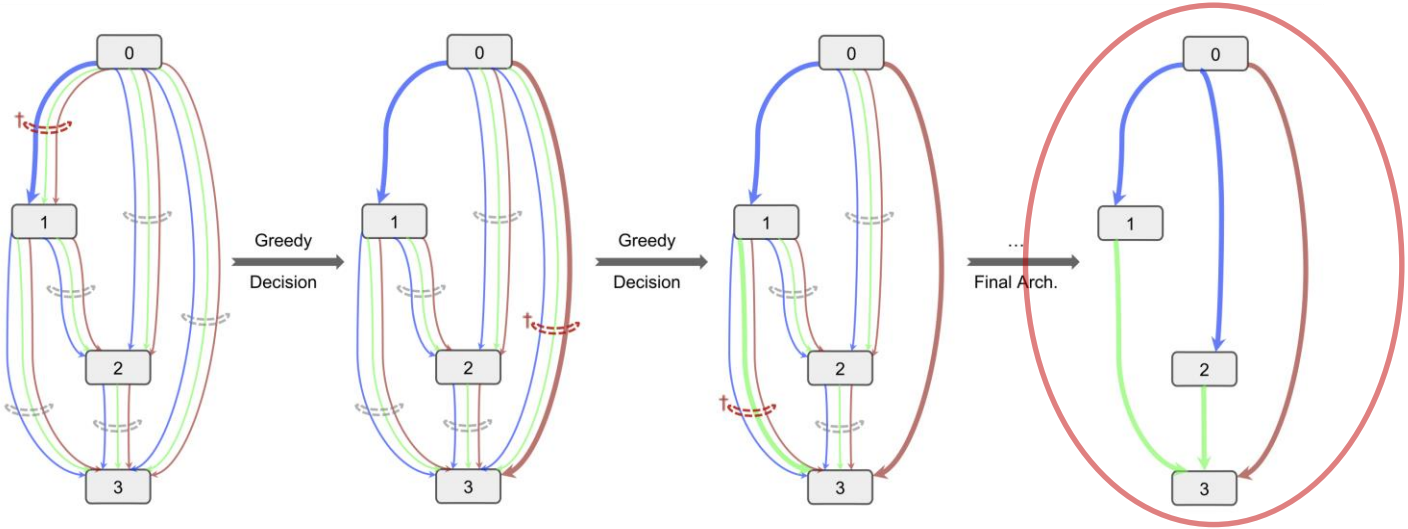


Figure 2. Illustration of Sequential Greedy Architecture Search.

A stand-alone architecture without weight sharing is obtained.

SGAS: Sequential Greedy Architecture Search

For the selection criterion, we consider three aspects of edges:

- Edge Importance
- Selection Certainty
- Selection Stability

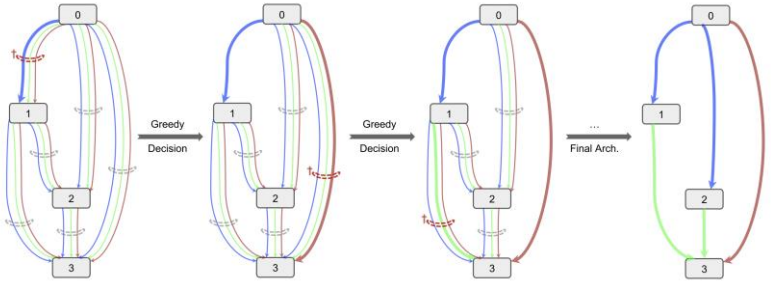


Figure 2. Illustration of Sequential Greedy Architecture Search.

SGAS: Sequential Greedy Architecture Search

For the selection criterion, we consider three aspects of edges:

- Edge Importance
- Selection Certainty
- Selection Stability

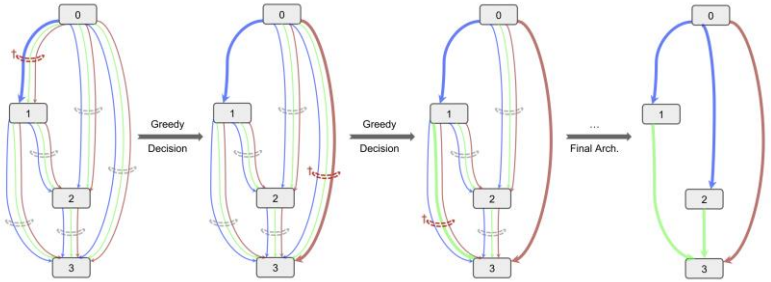


Figure 2. Illustration of Sequential Greedy Architecture Search.

Criterion 1 = (Edge Importance, Selection Certainty)

SGAS: Sequential Greedy Architecture Search

For the selection criterion, we consider three aspects of edges:

- Edge Importance
- Selection Certainty
- Selection Stability

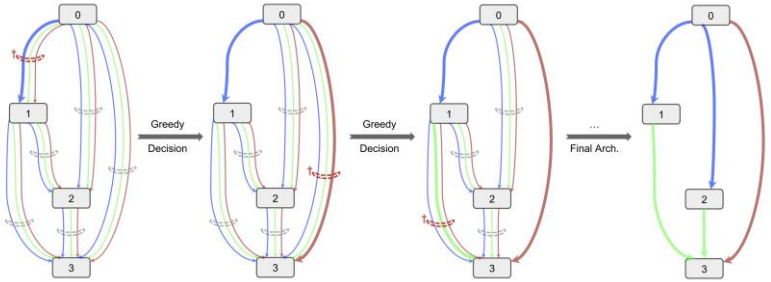
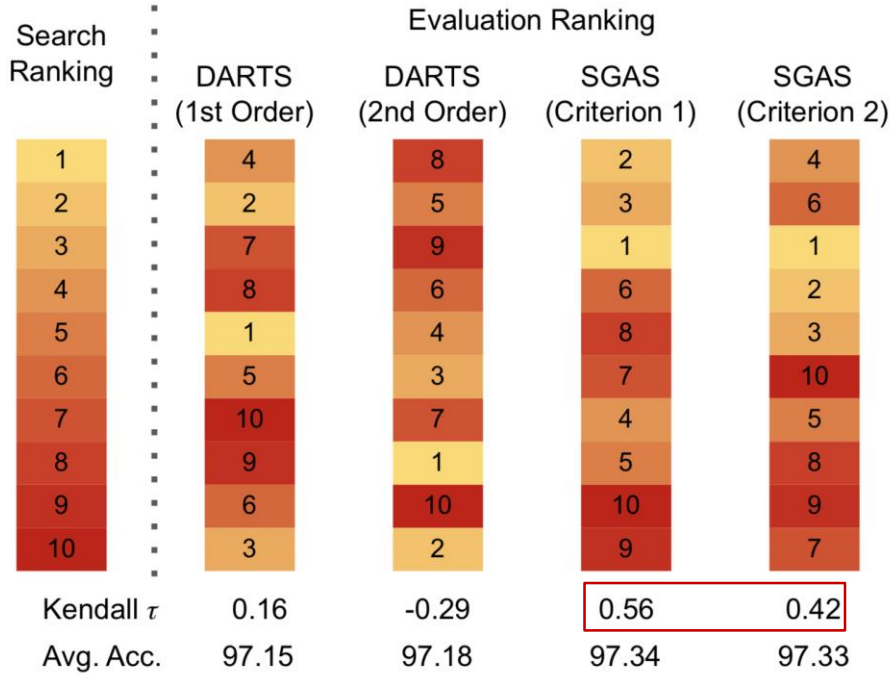


Figure 2. Illustration of Sequential Greedy Architecture Search.

Criterion 1 = (Edge Importance, Selection Certainty)

Criterion 2 = (Edge Importance, Selection Certainty, Selection Stability)

Degenerate search-evaluation correlation problem



SGAS with Criterion 1 and 2 improves the Kendall tau correlation coefficients to 0.56 and 0.42 respectively.

Figure 1. Comparison of search-evaluation Kendall τ coefficients.

Degenerate search-evaluation correlation problem

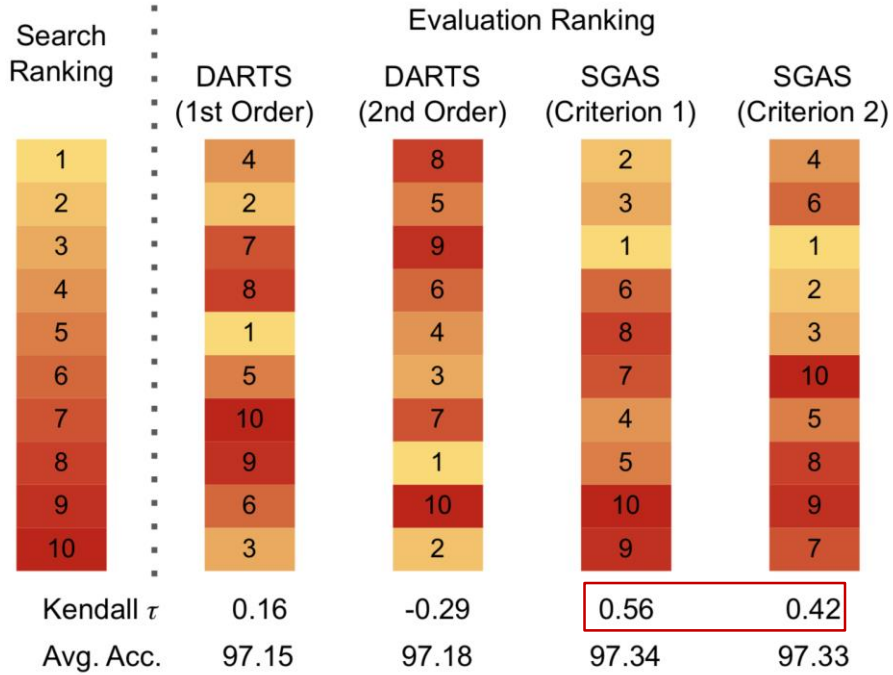


Figure 1. Comparison of search-evaluation Kendall τ coefficients.

SGAS with Criterion 1 and 2 improves the Kendall tau correlation coefficients to 0.56 and 0.42 respectively.

As expected from the much higher search-evaluation correlation SGAS outperform DARTS in terms of average accuracy significantly.

Experiments and Results

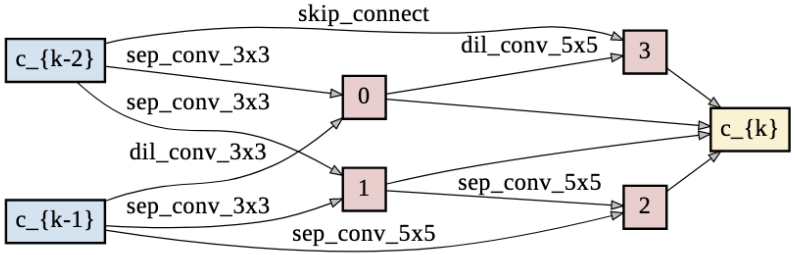
- **Search architectures for both CNNs and GCNs.**
- **The CNN architectures discovered by SGAS outperform the SOTA in image classification on CIFAR-10 and ImageNet.**
- **The discovered GCN architectures outperform the SOTA methods for node classification in biological graphs using the PPI dataset and point cloud classification using the ModelNet dataset**

Results – SGAS for CNN on CIFAR-10

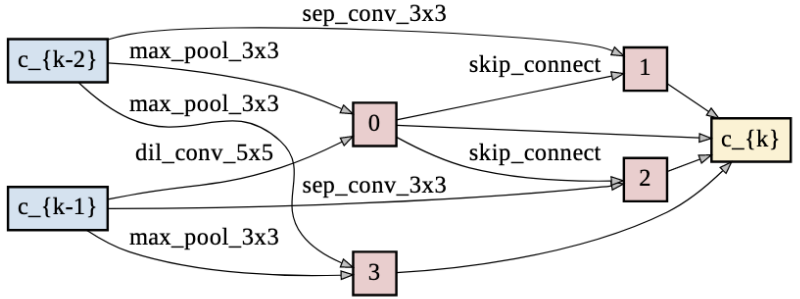
Architecture	Test Err. (%)	Params (M)	Search Cost (GPU-days)	Search Method
DenseNet-BC [18]	3.46	25.6	-	manual
NASNet-A [55]	2.65	3.3	1800	RL
AmoebaNet-A [36]	3.34±0.06	3.2	3150	evolution
AmoebaNet-B [36]	2.55±0.05	2.8	3150	evolution
Hier-Evolution [28]	3.75±0.12	15.7	300	evolution
PNAS [27]	3.41±0.09	3.2	225	SMBO
ENAS [34]	2.89	4.6	0.5	RL
NAONet-WS [31]	3.53	3.1	0.4	NAO
DARTS (1st order) [29]	3.00±0.14	3.3	0.4	gradient
DARTS (2nd 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

Table 1. Performance comparison with state-of-the-art image classifiers on CIFAR-10.

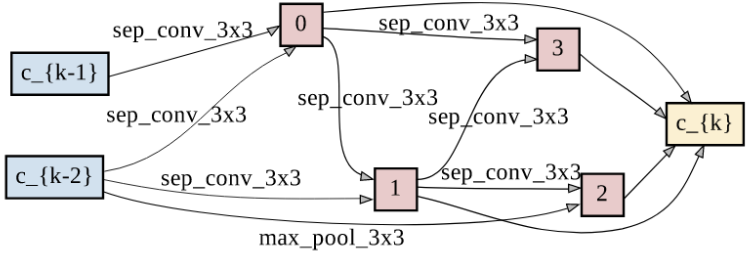
Results – SGAS for CNN on CIFAR-10



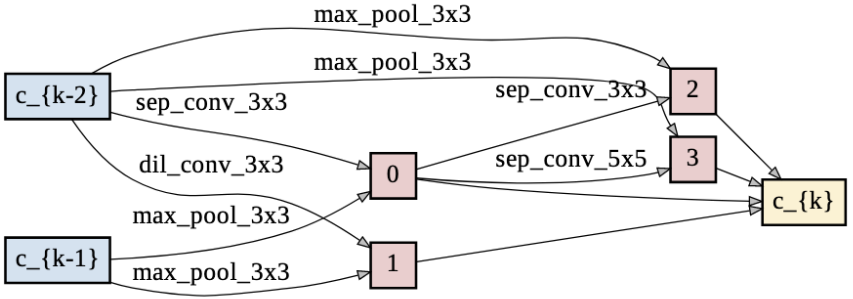
(a) Normal cell of the best model with SGAS (Cri. 1) on CIFAR-10



(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



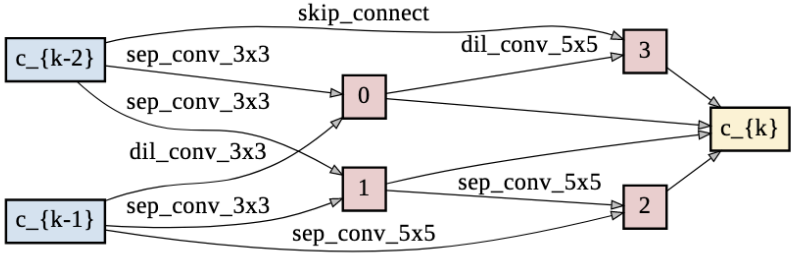
(d) Reduction cell of the best model with SGAS (Cri. 2) on CIFAR-10

Results – SGAS for CNN on ImageNet

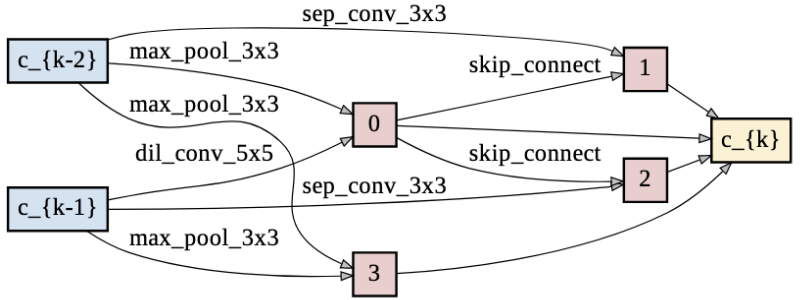
Architecture	Test Err. (%) top-1	Test Err. (%) top-5	Params (M)	×+ (M)	Search Cost (GPU-days)	Search Method
Inception-v1 [41]	30.2	10.1	6.6	1448	-	manual
MobileNet [16]	29.4	10.5	4.2	569	-	manual
ShuffleNet 2x (v1) [51]	26.4	10.2	∼5	524	-	manual
ShuffleNet 2x (v2) [32]	25.1	-	∼5	591	-	manual
NASNet-A [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

Table 2. Performance comparison with state-of-the-art image classifiers on ImageNet.

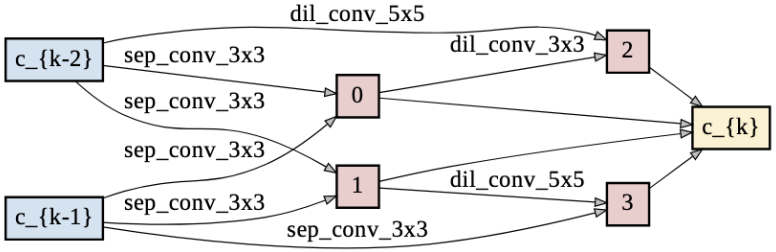
Results – SGAS for CNN on ImageNet



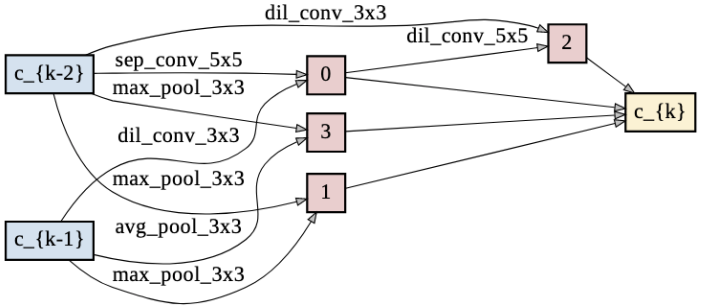
(a) Normal cell of the best model with SGAS (Cri. 1) on ImageNet



(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

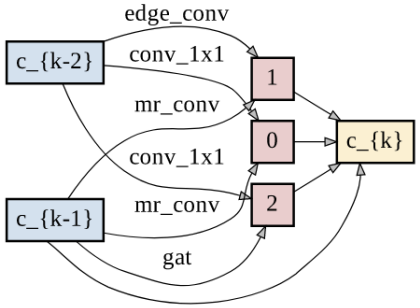


(d) Reduction cell of the best model with SGAS (Cri. 2) on ImageNet

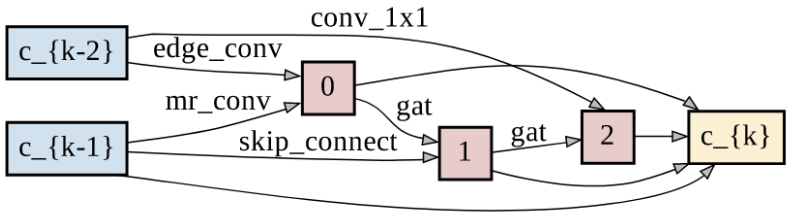
Results – SGAS for GCN on ModelNet

Architecture	OA (%)	Params (M)	Search Cost (GPU-days)
3DmFV-Net [3]	91.6	45.77	manual
SpecGCN [45]	91.5	2.05	manual
PointNet++ [35]	90.7	1.48	manual
PCNN [2]	92.3	8.2	manual
PointCNN [25]	92.2	0.6	manual
DGCNN [46]	92.2	1.84	manual
KPConv [43]	92.9	14.3	manual
Random Search	93.24±0.43	3.94	random
SGAS (Cri.1 avg.)	93.49±0.27	3.95	0.19
SGAS (Cri.1 best)	93.83	3.95	0.19
SGAS (Cri.2 avg.)	93.78±0.32	3.98	0.19
SGAS (Cri.2 best)	**94.38**	3.85	0.19

Table 3. 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

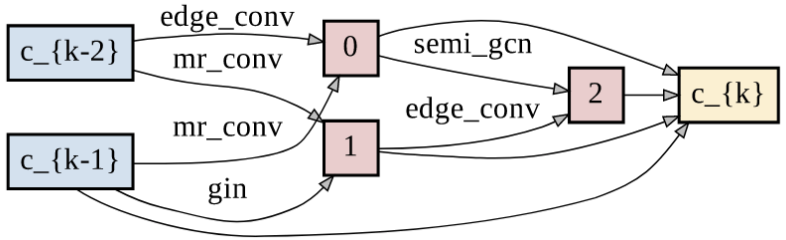


(b) Normal cell of the best model with SGAS (Cri. 2) on ModelNet

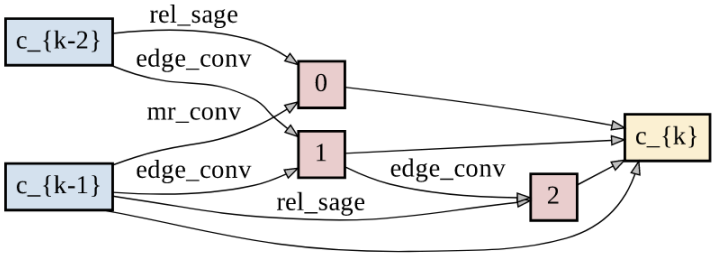
Results – SGAS for GCN on PPI

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

Table 4. Comparison with state-of-the-art architectures for node classification on PPI.

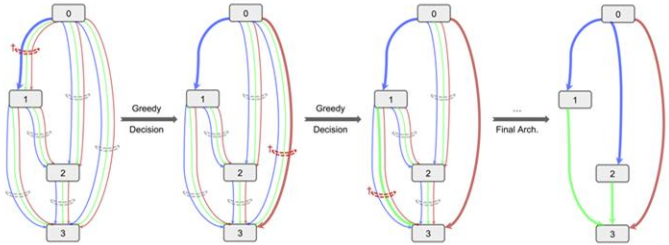


(a) Normal cell of the best model with SGAS (Cri. 1) on PPI

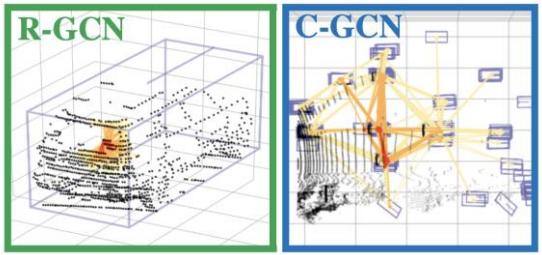


(b) Normal cell of the best model with SGAS (Cri. 2) on PPI

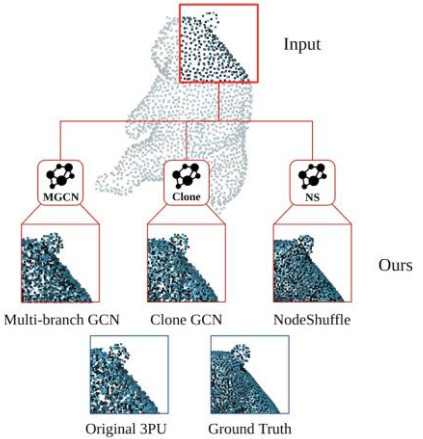
Follow-up works



SGAS: Sequential Greedy Architecture Search. Guohao Li. et al.

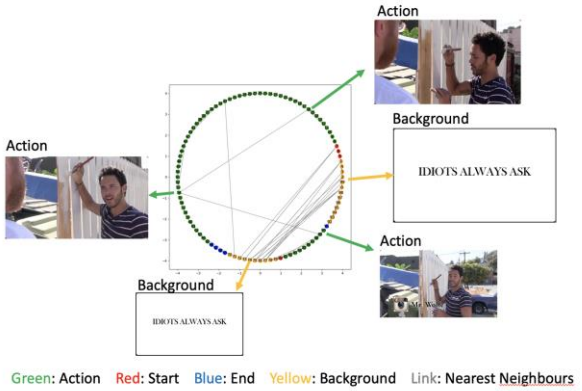


PointRGCN: Graph Convolution Networks for 3D Vehicles Detection Refinement. Jesue Zarzar. et al.

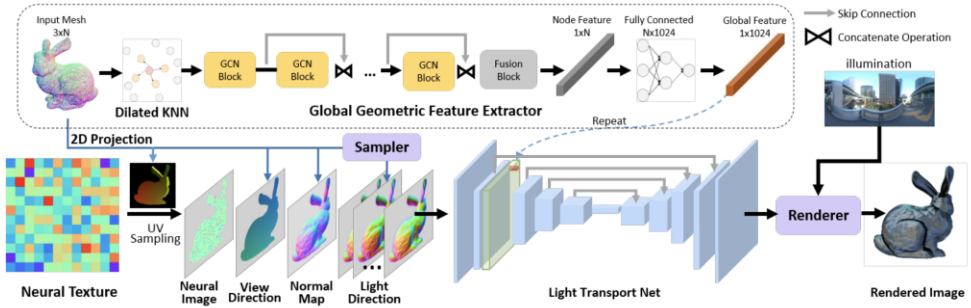


PU-GCN: Point Cloud Upsampling via Graph Convolutional Network. Guocheng Qian. et al.

Follow-up works



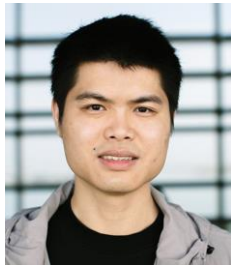
G-TAD: Sub-Graph Localization for Temporal Action Detection. Mengmeng xu. et al.



A Neural Rendering Framework for Free-Viewpoint Relighting. Zhang Chen. et al.

Our team

DeepGCNs.org



Guohao Li

Matthias Müller

Want to know more about IVUL?
Go to ivul.kaust.edu.sa
Or
lightaime@gmail.com

Guocheng Qian

Itzel C. Delgadillo

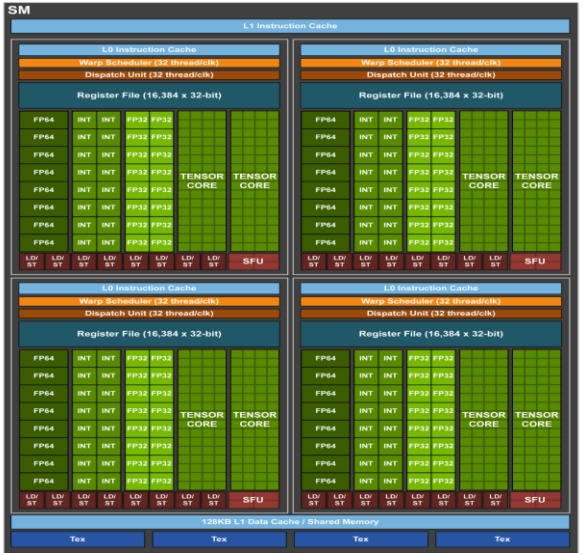
Ali Thabet

Bernard Ghanem

Abdulellah Abualshour

Tensor Core

New hardware unit in Volta GPU aiming at accelerate matrix computation and training speed of DNN



Tensor Core in V100

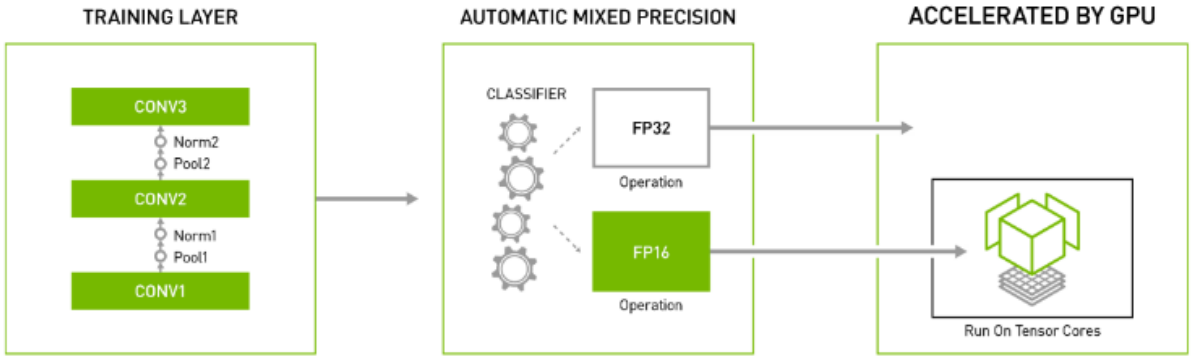
$$D = \begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix} \begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix} + \begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}$$

FP16 or FP32 FP16 FP16 FP16 or FP32

Main function: mix precision FMA (Fused Multiply-Add)

MIX PRECISION TRAINING

Easy to Use, Greater Performance and Boost in Productivity



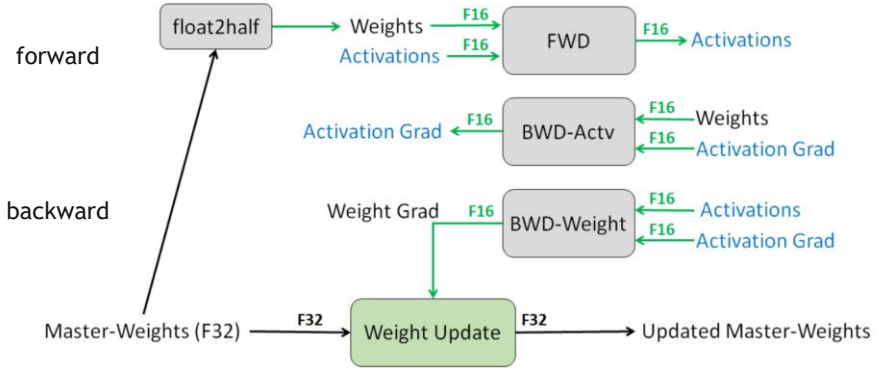
Insert ~ two lines of code to introduce Automatic Mixed-Precision and get upto 3X speedup

AMP uses a graph optimization technique to determine FP16 and FP32 operations

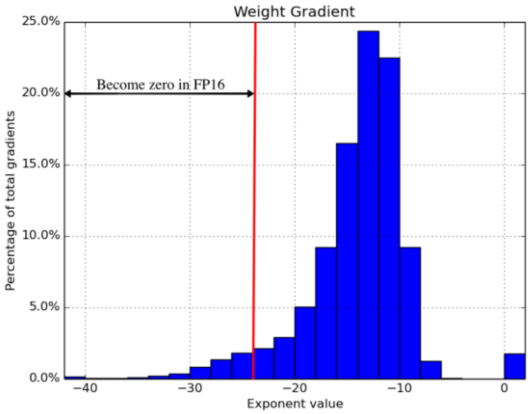
Support for TensorFlow, PyTorch and MXNet

Unleash the next generation AI performance and get faster to the market!

MIX PRECISION TRAINING



- Forward: do computation via FP16
- Backward: SGD via FP32 on master copy



- FP16 representation lead to gradient update=0
- Mechanism of floats adding lead to gradient update bias

MIX PRECISION TRAINING

Add Just A Few Lines of Code, Get Upto 3X Speedup

TensorFlow

```
os.environ['TF_ENABLE_AUTO_MIXED_PRECISION'] = '1'
```

```
export TF_ENABLE_AUTO_MIXED_PRECISION=1
```

PyTorch

```
model, optimizer = amp.initialize(model, optimizer)
```

```
with amp.scale_loss(loss, optimizer) as scaled_loss:
    scaled_loss.backward()
```

MXNet

```
amp.init()
amp.init_trainer(trainer)
with amp.scale_loss(loss, trainer) as scaled_loss:
   autograd.backward(scaled_loss)
```

More details: https://developer.nvidia.com/automatic-mixed-precision

Training Efficiency

<num_gpu, batch size, layers>	Tensorflow on V100 FP32 (s/epoch)	Tensorflow on V100 FP16 with AMP (s/epoch)	Through put in FP32 (image/s)	Through put in FP16 with AMP (image/s)	Speedup
(1, 4, 28)	4044.32	2210.01	4.92	9.01	1.83
(2, 4, 28)	2097.09	1352.96	9.49	14.71	1.55
(4, 4, 28)	1068.43	797.34	18.63	24.95	1.34
(8, 4, 28)	546.74	417.36	36.39	47.67	1.31

28-layer ResGCN, GPU Driver:418.67, CUDA 10.1,CUDNN 7.6.1, V100 16g with Nvlink

Using NVIDIA V100 Tensor Core GPUs and Mix-Precision Training, we've been able to achieve an impressive speedup versus the baseline FP32 implementation.

Useful materials or tools

GCN:

- Pytorch Geometric: https://pytorch-geometric.readthedocs.io
- Deep Graph Library: https://www.dgl.ai/
- TensorFlow Graphics: https://github.com/tensorflow/graphics

AMP:

- AMP for Deep Learning: https://developer.nvidia.com/automatic-mixed-precision
- AMP SDK: https://docs.nvidia.com/deeplearning/sdk/mixed-precision-training/index.html
- Tensor Core: https://developer.nvidia.com/tensor-cores


THANKS!
NVIDIA