Exploring Randomly Wired Neural Networks for Image Recognition

Saining Xie Alexander Kirillov Ross Girshick Kaiming He

Facebook AI Research (FAIR)

Computer Vision Lab, Hanyang University

Paper review

22 Apr 2019

Jihun Kim

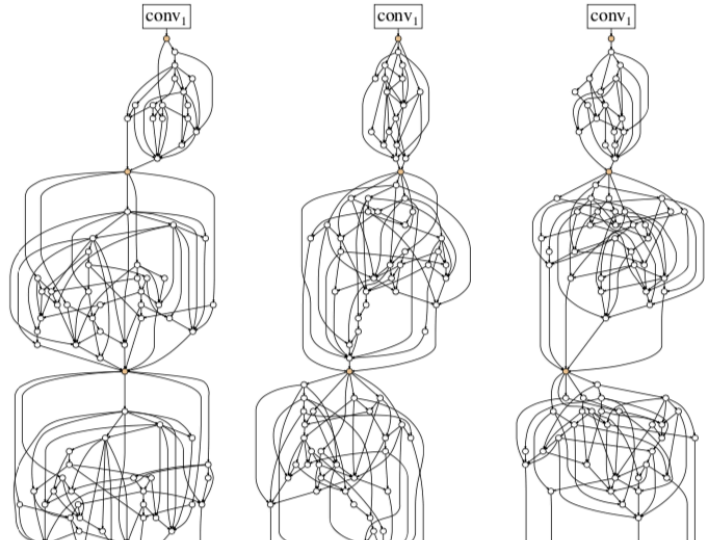
Exploring Randomly Wired Neural Networks for Image Recognition

Saining Xie Alexander Kirillov Ross Girshick Kaiming He

Facebook AI Research (FAIR)

Abstract

Neural networks for image recognition have evolved through extensive manual design from simple chain-like models to structures with multiple wiring paths. The success of ResNets [11] and DenseNets [16] is due in large part to their innovative wiring plans. Now, neural architecture search (NAS) studies are exploring the joint optimization of wiring and operation types, however, the space of possible wirings is constrained and still driven by manual design despite being searched. In this paper, we explore a more diverse set of connectivity patterns through the lens of randomly wired neural networks. *To do this, we first define*

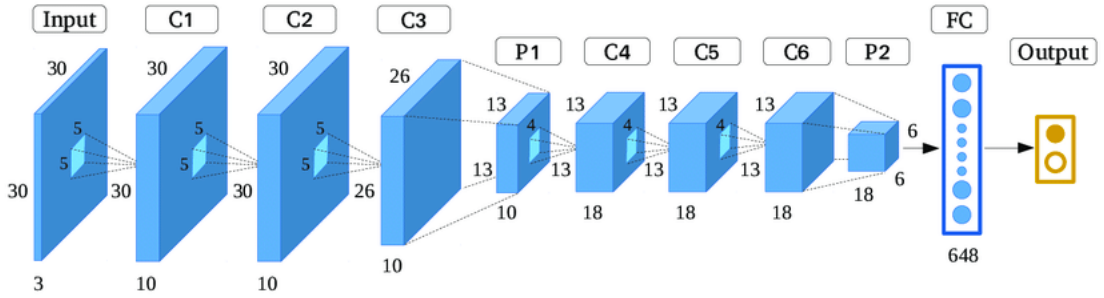


Introduction

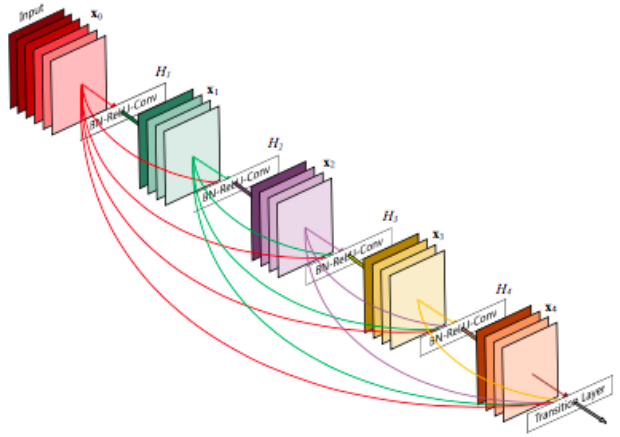
Exploring Randomly Wired Neural Networks
for Image Recognition.

Introduction

- Connectionist approach
  - How computational networks are wired is crucial for building intelligent machines.
  - Recent advances in computer vision have been driven by moving from models with chain-like wiring to more elaborate connectivity patterns.



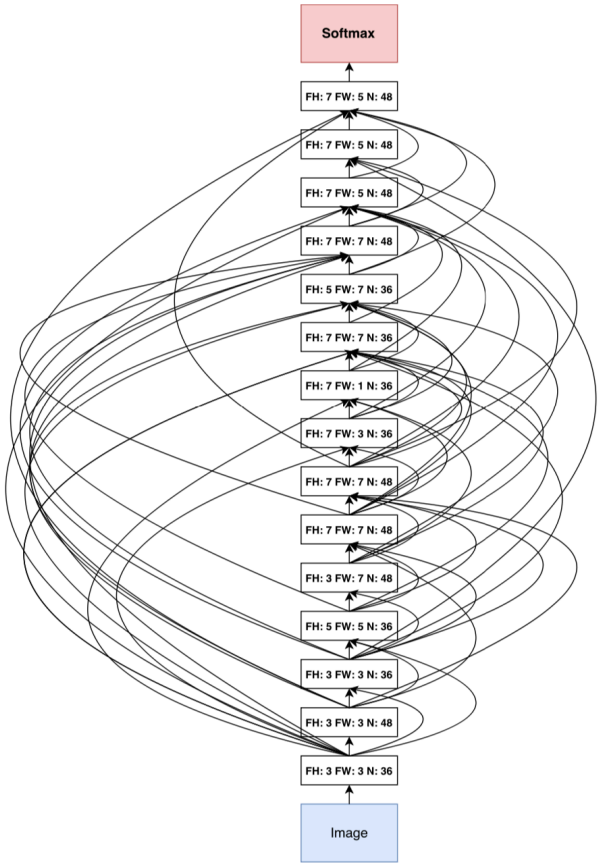
Graphic representation of VGGNet.



Graphic representation of DenseNet block.

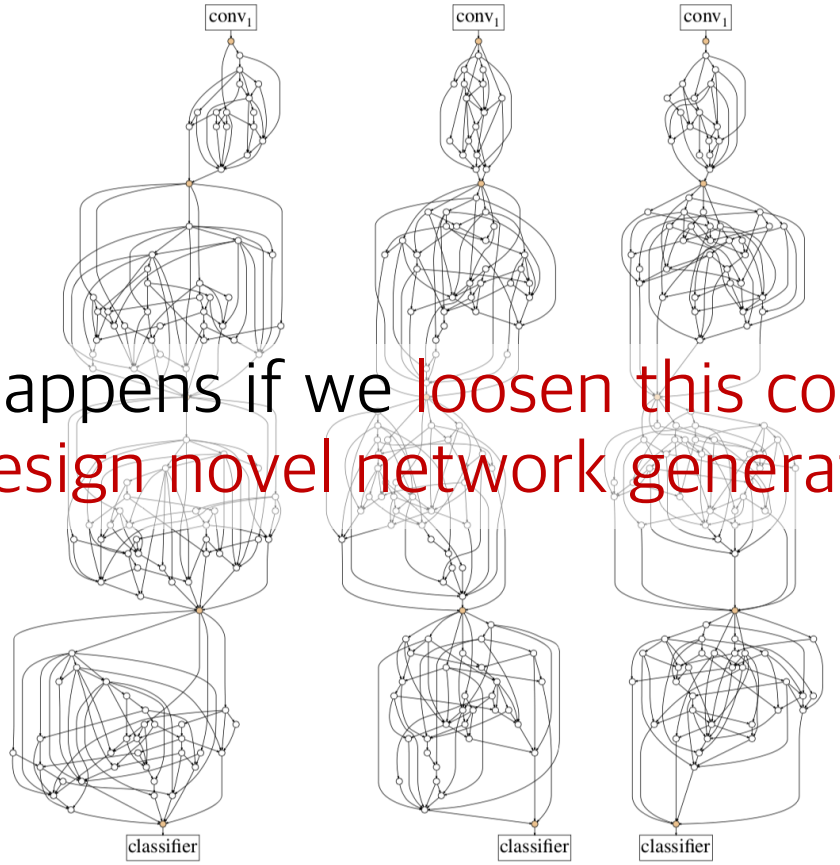
Introduction

- NAS (Neural Architecture Search)
  - Advancing this trend, NAS has emerged as a promising direction for jointly searching wiring patterns and which operations to perform.
  - However, like the wiring patterns in ResNet and DenseNet, the NAS network generator is hand designed and the space of allowed wiring patterns is constrained in a small subset of all possible graphs.



Convolutional architecture generated by NAS

Introduction



"What happens if we loosen this constraint and design novel network generators?"

Randomly wired neural networks generated by WS model

Related Work

Exploring Randomly Wired Neural Networks
for Image Recognition.

Related Work

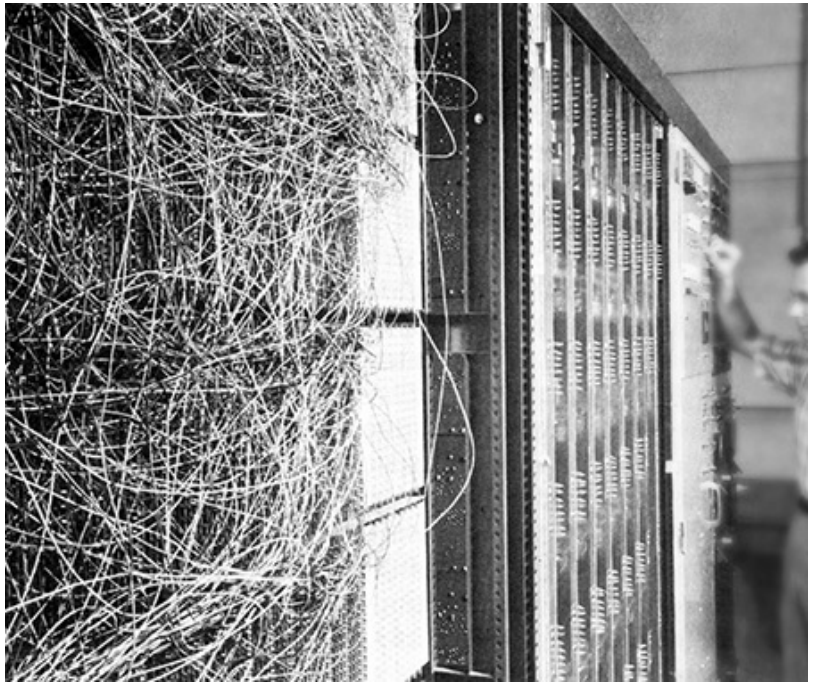
- Network Wiring
  - Early recurrent and convolutional neural networks (RNNs and CNNs) use chain-like wiring patterns.
  - The LSTM, Inception, ResNet, and DenseNet wiring patterns are effective in general.

- Neural Architecture Search (NAS)
  - Recent research on NAS mainly focuses on optimization methods.
  - Network generator is largely unchanged in these works.

Related Work

- Randomly Wired Machines
  - In 1940s, Turing suggested a concept of *unorganized machines*, which is a form of the earliest randomly connected neural networks.
  - Minsky and Rosenblatt built earliest randomly connected neural networks in 1950s.

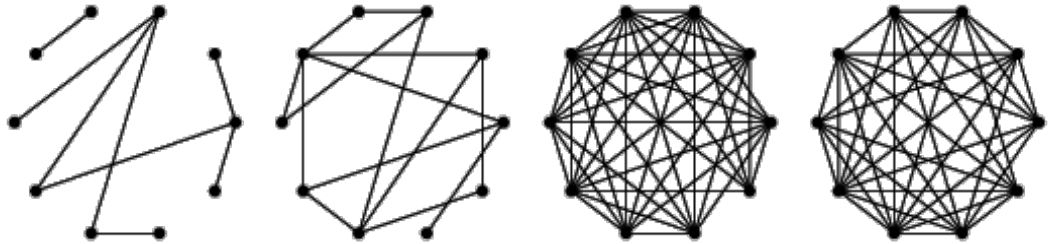
- Relation to Neuroscience
  - Random graph modeling has been used as a tool to study the neural networks of human brains.



The Mark 1 Perceptron
(Source: Arvin Calspan Advanced Technology Center; Hecht-Nielsen, R. Neurocomputing (Reading, Mass.: Addison-Wesley, 1990).)

Related Work

- Random graphs in graph theory
  - Random graphs exhibit different probabilistic behaviors depending on the random process defined by the model.
  - The definition of the random graph model determines the prior knowledge encoded in the resulting graphs and may connect them to naturally occurring phenomena.



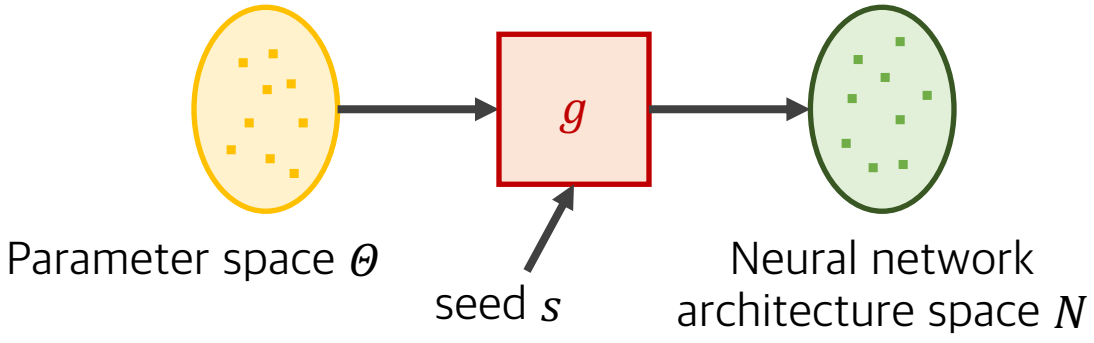
Methodology

Exploring Randomly Wired Neural Networks
for Image Recognition.

Methodology > Network Generators

- We define a network generator as a mapping g from a parameter space Θ to a space of neural network architectures N, $g{:}\Theta \mapsto N$.
  - The generator g determines how the computational graph is wired.

- Stochastic Network Generators
  - The above network network generator g(θ) performs a deterministic mapping.
  - We can extend $g(\theta)$ to $g(\theta, s)$, in which an additional argument s is the seed of a pseudo-random number generator that is used internally by g.



Methodology > Network Generators

- NAS from the Network Generator Perspective
  - The NAS methods utilize an LSTM “controller” in the process of generating network architectures.
  - The weight matrices of the LSTM are the parameters θ of the generator, which are optimized by RL.
  - The output of each LSTM time-step is a probability distribution conditioned on θ.
  - Given this distribution and the seed s, each step samples a construction action.
  - However, There are also hand-designed rules defined to map the sampled actions to a computational DAG, and these rules are also part of g.
  - Network space N has been carefully restricted by hand-designed rules.

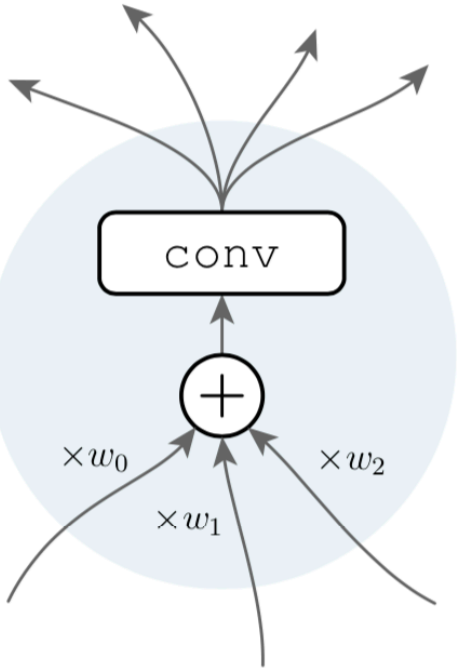
Methodology > Randomly Wired Neural Networks

- Generating General Graphs
  - Network generator starts by generating a general graph.
    - It generates a set of nodes and edges that connect nodes, without restricting how the graphs correspond to neural networks.
  - Once a graph is obtained, it is mapped to a computable neural network.
  - The authors intentionally use a simple mapping.

Methodology > Randomly Wired Neural Networks

- Edge Operations
  - The edges are data flow.
  - A directed edge sends data from one node to another node.

- Node Operations
  - Aggregation
    - The input data to a node are combined via a weighted sum; the weights are learnable and positive.
  - Transformation
    - ReLU-convolution-BN triplet
    - The same type of convolution is used for all nodes.
  - Distribution
    - The same copy of the transformed data is sent out by the output edges of the node.



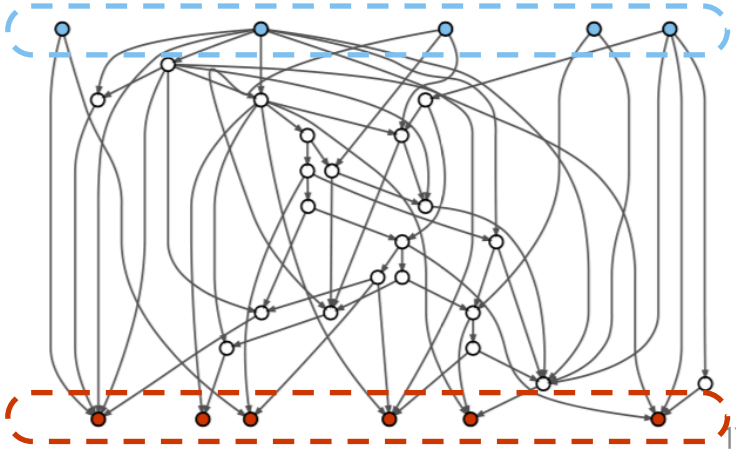
Node operations.
Illustrating a node (blue circle) with 3 input edges and 4 output edges.

Methodology > Randomly Wired Neural Networks

- Nice Properties of Node Operation
  - Additive aggregation (unlike concatenation) maintains the same number of output channels as input channels.
    - Prevents the convolution that follows from growing large in computation.
  - The transformation should have the same number of output and input channels.
    - Makes sure the transformed data can be combined with the data from any other nodes.
    - Keeps the FLOPs and parameter count unchanged for each node.
  - Aggregation and distribution are almost parameter-free.
- These properties nearly decouple FLOPs and parameter count from network wiring.

Methodology > Randomly Wired Neural Networks

- Input and Output Nodes
  - A general graph is not yet a valid neural network because it may have multiple input nodes and multiple output nodes.
  - We create a single extra node that is connected to all original input nodes, and another single extra node that is connected to all original output nodes.
    - These two nodes perform no convolution.
    - The unique input node sends out the same copy of input data to all original input nodes.
    - The unique output node compute (unweighted) average from all original output nodes.

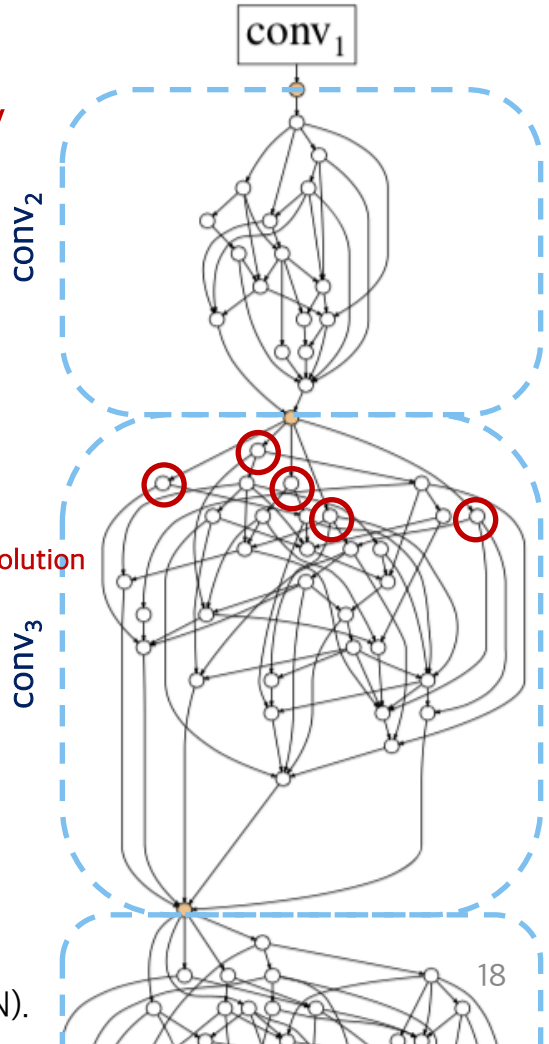


Methodology > Randomly Wired Neural Networks

- Stages
  - It is common to divide a network into stages that progressively down-sample feature maps.
  - Simple strategy
    - One random graph represents one stage, and it is connected to its preceding/succeeding stage by its unique input/output node.
    - For all nodes that are directly connected to the input node, their transformations are modified to have a stride of 2.

stage	output	*small regime*	*regular regime*
conv$_1$	112×112	3×3 conv, $C/2$	
conv$_2$	56×56	3×3 conv, C	***random wiring*** $N/2$, C
conv$_3$	28×28	***random wiring*** N, C	***random wiring*** N, $2C$
conv$_4$	14×14	***random wiring*** N, $2C$	***random wiring*** N, $4C$
conv$_5$	7×7	***random wiring*** N, $4C$	***random wiring*** N, $8C$
classifier	1×1	1×1 conv, 1280-d global average pool, 1000-d *fc*, softmax	

⬅ **RandWire architectures** for small & regular networks.
A random graph is denoted by the node count (N)
and channel count for each node (C).
conv denotes a ReLU-Conv-BN triplet (except conv1 is Conv-BN).



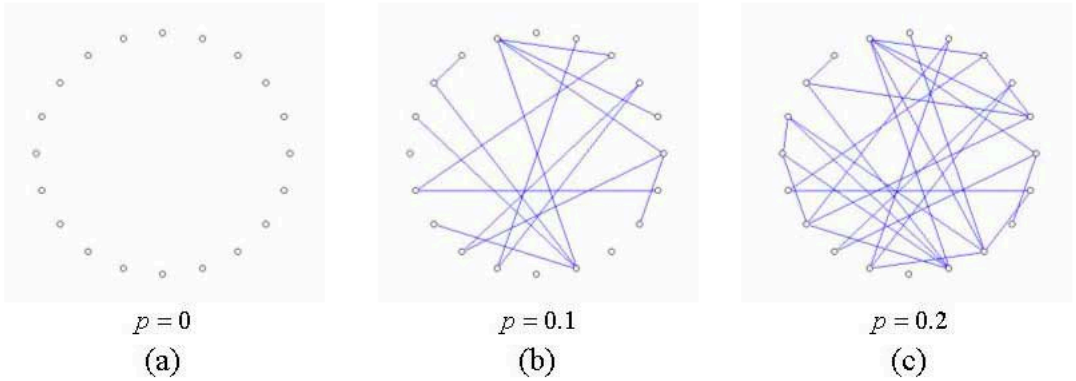
Methodology > Random Graph Models

- Three classical random graph models are used in this study.
  - Erdős–Rényi (ER), 1959.
  - Barabási–Albert (BA), 1999.
  - Watts–Strogatz (WS), 1998.

- These models all generate undirected graphs.
  - We use a simple heuristic to turn them into DAGs.

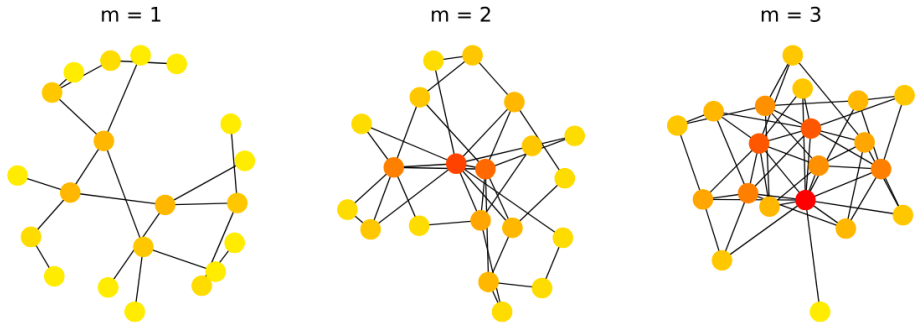
Methodology > Random Graph Models

- Erdős–Rényi (ER)
  - An edge between two nodes is connected with probability P.
    - This process is iterated for all pairs of nodes.
  - The ER generation model has only a single parameter P, and is denoted as ER(P).
  - Any graph with N nodes has non-zero probability of being generated by the ER model.



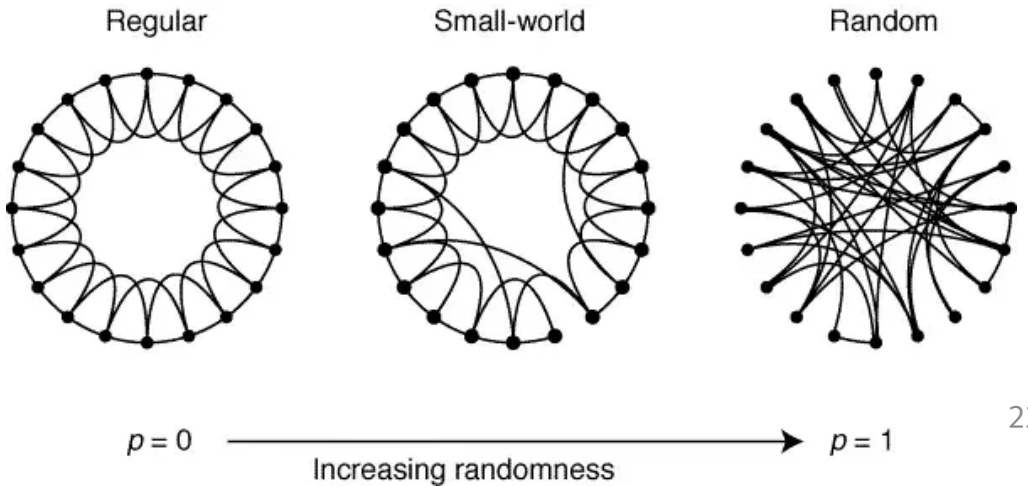
Methodology > Random Graph Models

- Barabási-Albert (BA)
  - The initial state is M nodes without any edges $(1 \leq M < N)$.
  - The method sequentially adds a new node with M new edges.
    - For a node to be added, it will be connected to an existing node v with probability proportional to v's degree.
    - The new node repeatedly adds non-duplicate edges in this way until it has M edges.
    - Then this is iterated until the graph has N nodes.
  - This model has only a single parameter M, and is denoted as BA(M).
  - BA(M) only covers a small subset of all possible N-node graphs.



Methodology > Random Graph Models

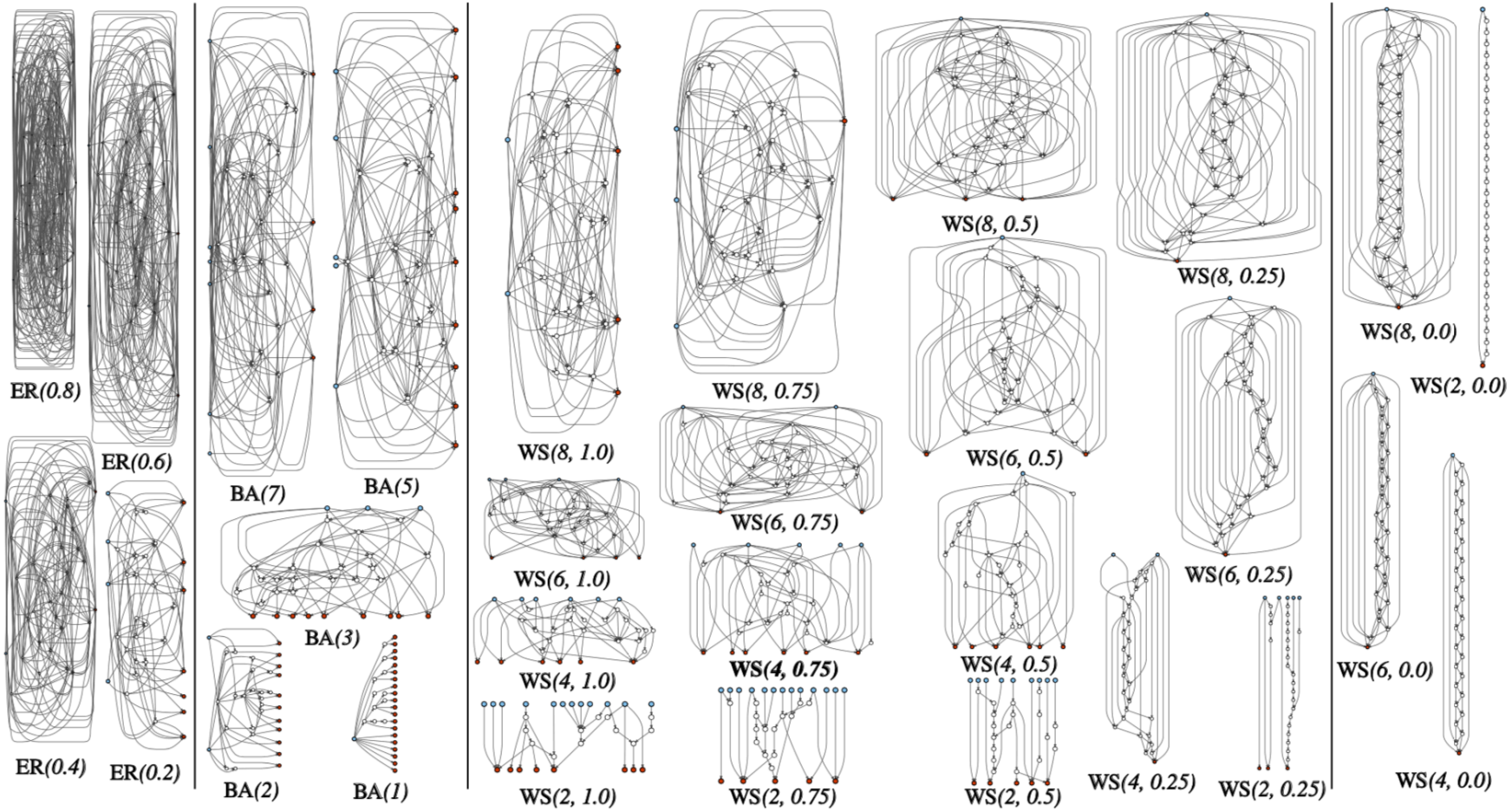
- Watts-Strogatz (WS)
  - The WS model was defined to generate small-world graphs.
  - Initially, the N nodes are regularly placed in a ring and each node is connected to its $K/2$ neighbors on both sides (K is an even number).
  - Then, in a clockwise loop, for every node v, the edge that connects v to its clockwise i-th next node is rewired with probability P.
    - "Rewiring" is defined as uniformly choosing a random node that is not v and that is not a duplicate edge.
    - This loop is repeated $K/2$ times for $1 \leq i \leq K/2$.
  - K and P are the only two parameters of the WS model, denoted as WS(K, P).
  - WS(K, P) only covers a small subset of all possible N-node graphs.



Methodology > Design and Optimization

- Randomly wired neural networks are generated by a stochastic network generator $g(\theta, s)$.
  - The optimization is essentially done by *trial-and-error* by human designers.
  - The accuracy variation of these networks is small for different seeds s.

Methodology > Design and Optimization



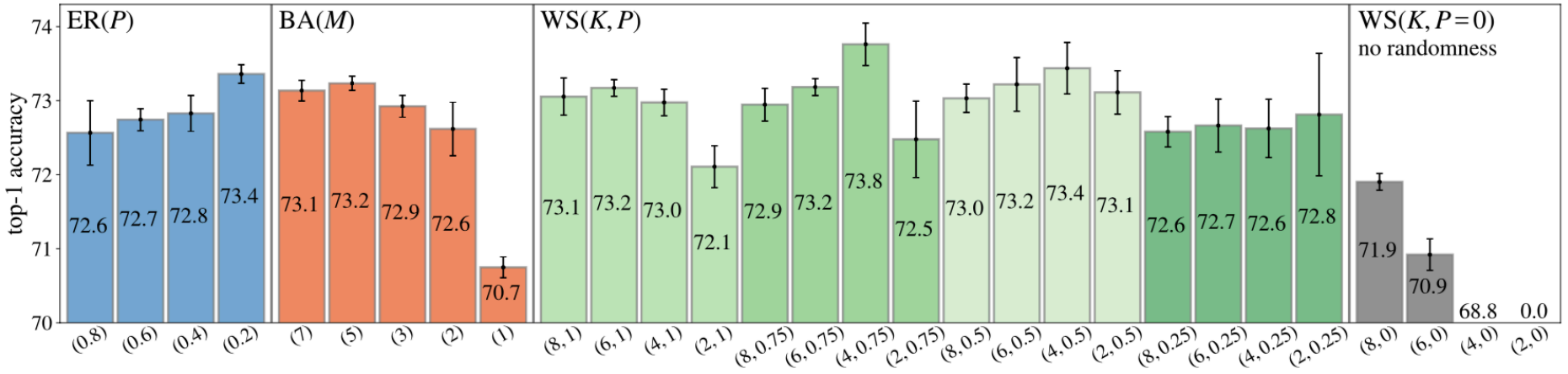
Visualization of the random graphs generated by ER, BA, and WS.
The node count is N=32 for each graph.

Experiments

Exploring Randomly Wired Neural Networks
for Image Recognition.

Experiments > Analysis Experiments

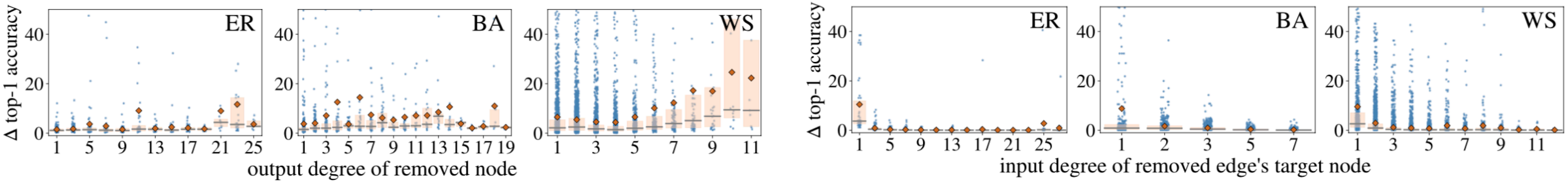
- Random Graph Generator
  - All random generators provide decent accuracy over all 5 network instances.
  - The variation among the random network instances is low.
  - Random generator design plays an important role.



Comparison on random graph generators: ER, BA, and WS in the small computation regime.
Each bar represents the results of a generator under a parameter setting for P, M, or (K, P).
The results are ImageNet top-1 accuracy, shown as mean and standard deviation (std) over 5 random network instances sampled by a generator.

Experiments > Analysis Experiments

- Graph Damage
  - Given a network instance *after training*, the authors randomly removed one node or one edge from the graph and evaluated the validation accuracy.
  - “Hub” nodes in WS that send information to many nodes are influential.

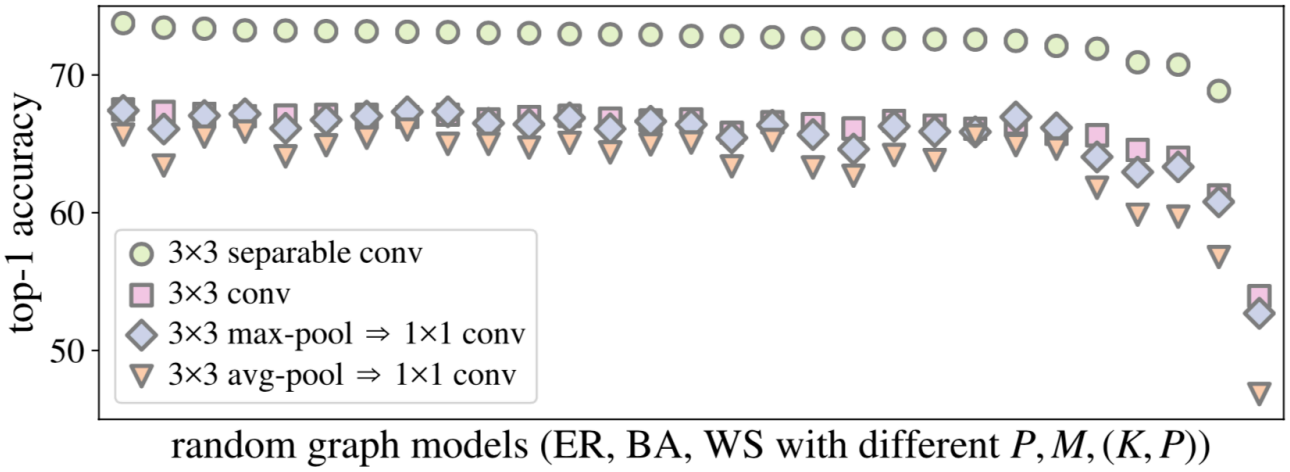


Graph damage ablation

Red circle: mean; gray bar: median; orange box: interquartile range; blue dot: an individual damaged instance.

Experiments > Analysis Experiments

- Node operations
  - Replacing the transformation of all nodes with the specified alternative.
  - Almost all networks still converge to non-trivial results.
  - The generators roughly maintain their orders of accuracy.



Alternative node operation

Each column is the mean accuracy of the same set of 5 random graphs equipped with different node operations.

Experiments > Comparisons

- Small computation regime
  - The result of RandWire with WS(4, 0.75) is better than or comparable to almost all existing hand-designed wiring (MobileNet/ShuffleNet) and NAS-based results.

network	top-1 acc.	top-5 acc.	FLOPs (M)	params (M)
MobileNet [15]	70.6	89.5	569	4.2
MobileNet v2 [40]	74.7	-	585	6.9
ShuffleNet [54]	73.7	91.5	524	5.4
ShuffleNet v2 [30]	**74.9**	92.2	591	7.4
NASNet-A [56]	74.0	91.6	564	5.3
NASNet-B [56]	72.8	91.3	488	5.3
NASNet-C [56]	72.5	91.0	558	4.9
Amoeba-A [34]	74.5	92.0	555	5.1
Amoeba-B [34]	74.0	91.5	555	5.3
Amoeba-C [34]	**75.7**	**92.4**	570	6.4
PNAS [26]	74.2	91.9	588	5.1
DARTS [27]	73.1	91.0	595	4.9
RandWire-WS	**74.7**$_{\pm 0.25}$	**92.2**$_{\pm 0.15}$	583$_{\pm 6.2}$	5.6$_{\pm 0.1}$

ImageNet: small computation regime (*i.e.,* < 600M FLOPs)

Experiments > Comparisons

- Regular computation regime
  - Using regularization method
    - For each training mini-batch, randomly remove one edge whose target node has input degre e > 1 with probability of 0.1.
    - This

network	top-1 acc.	top-5 acc.	FLOPs (B)	params (M)
ResNet-50 [11]	77.1	93.5	4.1	25.6
ResNeXt-50 [52]	78.4	94.0	4.2	25.0
RandWire-WS, C=109	**79.0**$_{\pm 0.17}$	**94.4**$_{\pm 0.11}$	4.0$_{\pm 0.09}$	31.9$_{\pm 0.66}$
ResNet-101 [11]	78.8	94.4	7.8	44.6
ResNeXt-101 [52]	79.5	94.6	8.0	44.2
RandWire-WS, C=154	**80.1**$_{\pm 0.19}$	**94.8**$_{\pm 0.18}$	7.9$_{\pm 0.18}$	61.5$_{\pm 1.32}$

ImageNet: regular computation regime

Experiments > Comparisons

- Larger computation
  - RandWire have mean accuracy 0.7%~1.3% lower than the most accurate NAS results, but use only ~2/3 FLOPs and ~3/4 parameters.
    - RandWire are trained for 100 epochs and not on the target image size, vs. the NAS methods which use >250 epochs and train on the target size.
    - The model has no search on operations, unlike NAS.
    - These gaps will be explored in future work.

network	test size	epochs	top-1 acc.	top-5 acc.	FLOPs (B)	params (M)
NASNet-A [56]	331^2	>250	82.7	96.2	23.8	88.9
Amoeba-B [34]	331^2	>250	82.3	96.1	22.3	84.0
Amoeba-A [34]	331^2	>250	82.8	96.1	23.1	86.7
PNASNet-5 [26]	331^2	>250	82.9	96.2	25.0	86.1
RandWire-WS	320^2	100	$81.6_{\pm 0.13}$	$95.6_{\pm 0.07}$	$16.0_{\pm 0.36}$	$61.5_{\pm 1.32}$

ImageNet: large computation regime

Experiments > Comparisons

- COCO object detection
  - Use Faster R-CNN with FPN as the object detector.

backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet-50 [11]	37.1	58.8	39.7	21.9	40.8	47.6
ResNeXt-50 [52]	38.2	60.5	41.3	23.0	41.5	48.8
RandWire-WS, $C=109$	**39.9**	**61.9**	**43.3**	**23.6**	**43.5**	**52.7**
ResNet-101 [11]	39.8	61.7	43.3	23.7	43.9	51.7
ResNeXt-101 [52]	40.7	62.9	44.5	24.4	44.8	52.7
RandWire-WS, $C=154$	**41.1**	**63.1**	**44.6**	**24.6**	**45.1**	**53.0**

COCO object detection results

Conclusion

Exploring Randomly Wired Neural Networks
for Image Recognition.

Conclusion

- Exploring randomly wired neural networks driven by three classical random graph models from graph theory.
- The mean accuracy of these models is competitive with hand-designed and optimized models from recent work on NAS.
- The authors hope that future work exploring new generator designs may yield new, powerful networks designs.

Exploring Randomly Wired Neural Networks for Image Recognition

Saining Xie Alexander Kirillov Ross Girshick Kaiming He

Facebook AI Research (FAIR)

Computer Vision Lab, Hanyang University

Paper review

22 Apr 2019

Jihun Kim