From Architecting Networks to Architecting Network Generators

Ross Girshick

Saining Xie

Alexander Kirillov

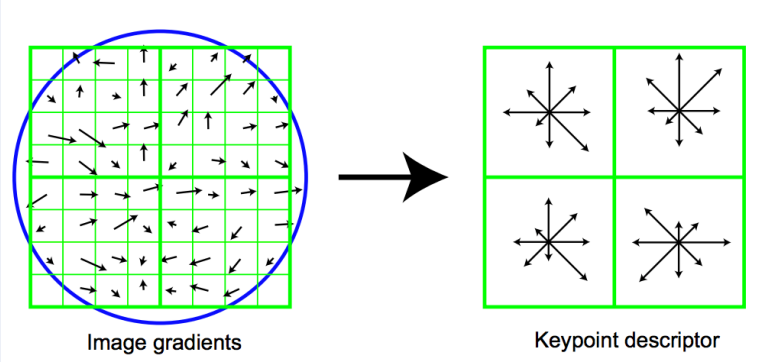
Kaiming He

facebook AI Research

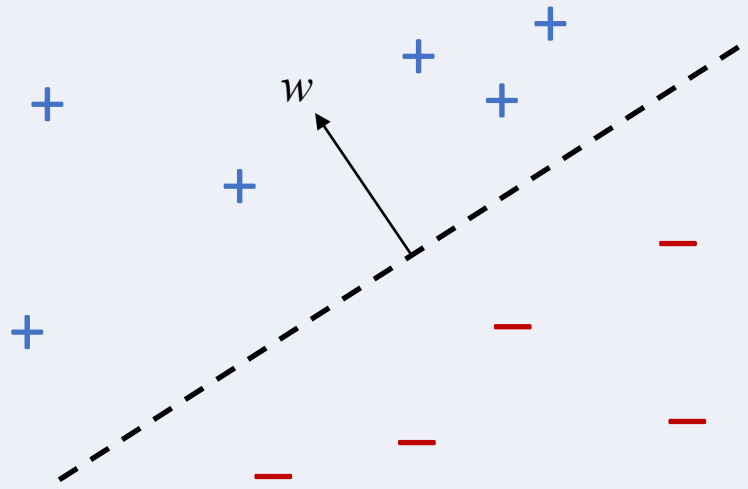
Many slides adapted from Saining Xie

Before 2012 (… but also after 1990)

- Engineering features (SIFT, …), learning weights (SVM, …)



Engineered

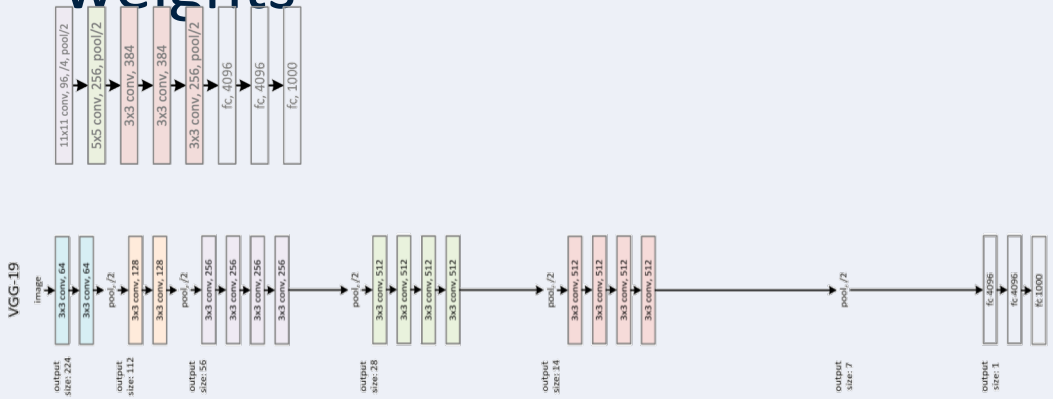


Learned

2012 – now-ish

- Engineering features (SIFT, …), learning weights (SVM, …)

- Engineering networks (AlexNet, VGG, …), learning features and weights



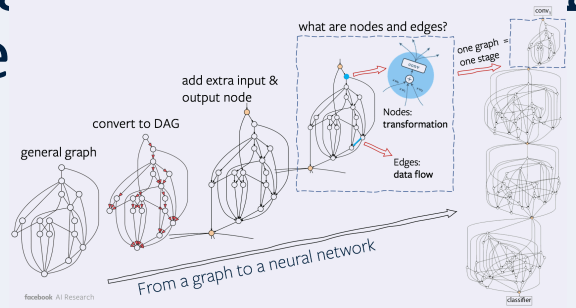
Millions of parameters
- Internal filters
- Classifier weights

Engineered

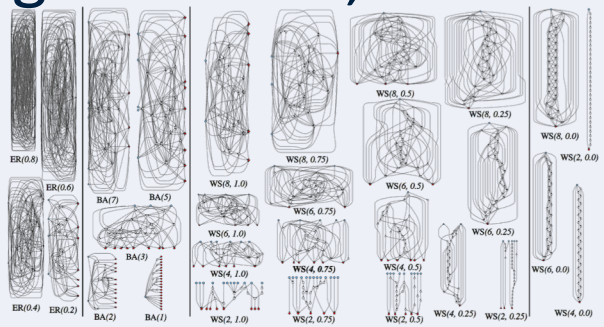
Learned

Now-ish - ???

- Engineering features (SIFT, …), learning weights (SVM, …)
- Engineering networks (AlexNet, VGG, …), learning features and weights
- Engineering network generators, learning networks, features & we



Engineered



Learned

Network Generator (Informally, Simple Case)

- Think of a Python function:

```
def vgg_net(input_res, depth, conv1_out_channels):
    … # code to make the model
    return model
```

Network Generator (Informally, Simple Case)

- Think of a Python function:

```
def vgg_net(input_res, depth, conv1_out_channels):
    … # code to make the model
    return model
```

- Input: some arguments (resolution, widths, depths, strides, …)

Network Generator (Informally, Simple Case)

- Think of a Python function:

```
def vgg_net(input_res, depth, conv1_out_channels):
    … # code to make the model
    return model
```

- Input: some arguments (resolution, widths, depths, strides, …)

- Output: instantiation of a network

Network Generator (Informally, Simple Case)

- Think of a Python function:

```
def vgg_net(input_res, depth, conv1_out_channels):
    … # code to make the model
    return model
```

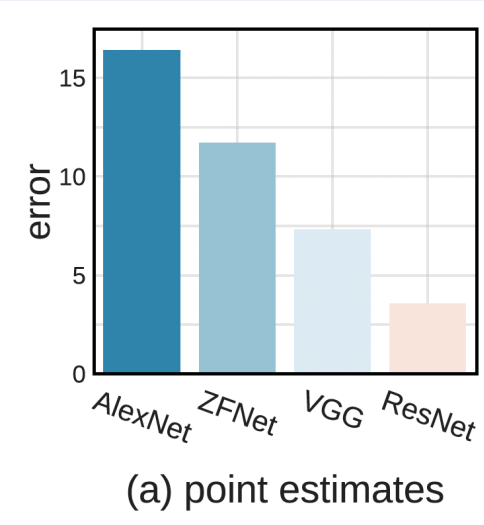
- Input: some arguments (resolution, widths, depths, strides, …)

- Output: instantiation of a network

- All valid input args → **generates a population** of related networks

From Single Networks to Populations of Networks

- Comparing individual models (“point estimates”)
  - E.g., AlexNet vs. VGG-16



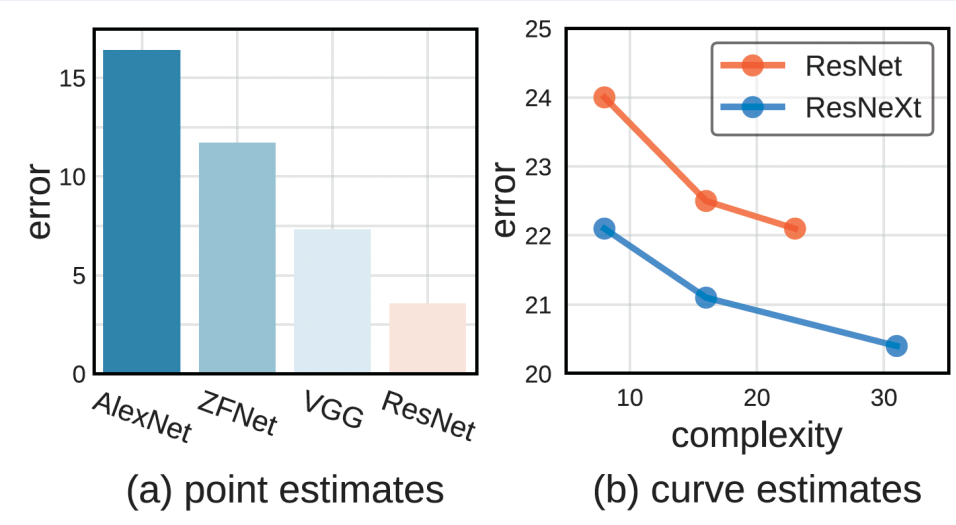
- No controls
- Why is one better than another?

Poster #61 in session 1.2 tomorrow (Tuesday)

Figure from: **On Network Design Spaces for Visual Recognition.** Radosavovic et al. ICCV 2019.

From Single Networks to Populations of Networks

- Comparing a few hand-designed variants ("curve estimates")
  - E.g., ResNet-50, ResNet-101, … vs. ResNeXt-50-32x4d, ResNeXt-101-32x4d, …

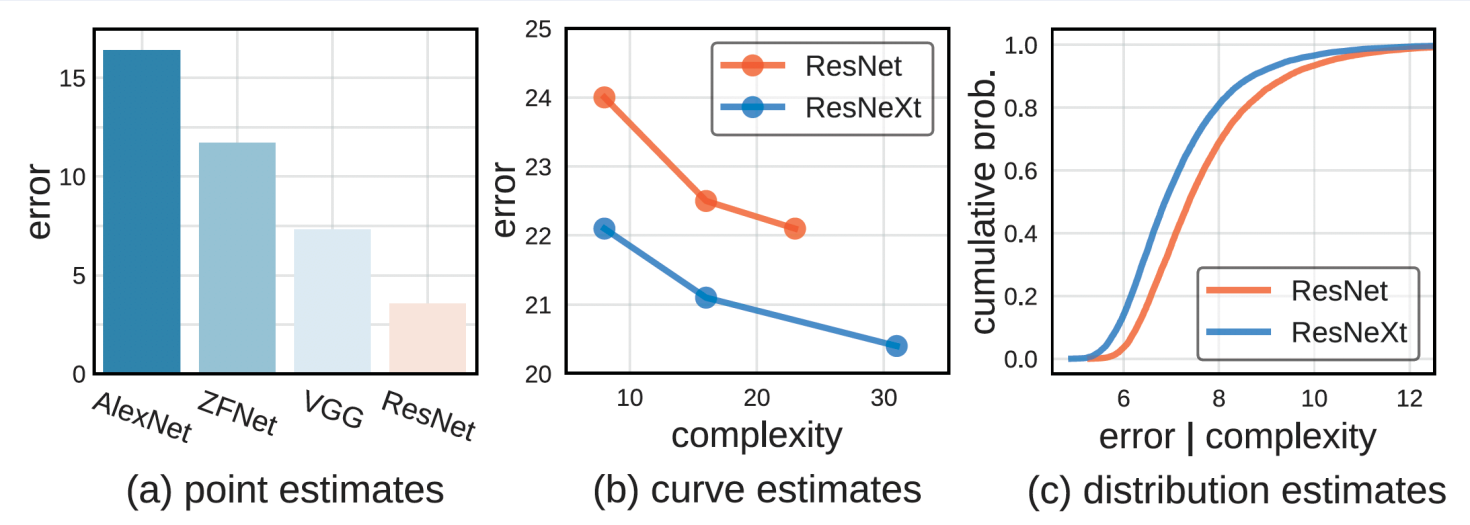


- Realization of FLOPs <-> accuracy
- Vary only one dimension (*e.g.*, depth)

Figure from: **On Network Design Spaces for Visual Recognition.** Radosavovic et al. ICCV 2019.

From Single Networks to Populations of Networks

- Comparing generated populations of models (“distribution estimates”)
  - Define network generators, evaluate population



- Vary all dimensions
- Is A always > B?

Figure from: **On Network Design Spaces for Visual Recognition.** Radosavovic et al. ICCV 2019.

Why This Shift in Perspective is Important

- Discover general principles
  - *E.g.*, grouped conv (extreme: depthwise conv) is good

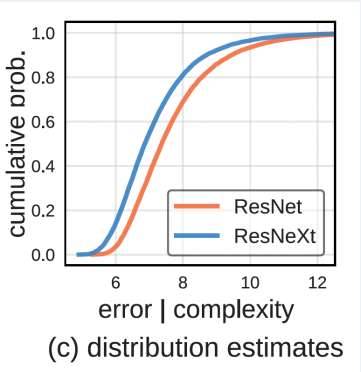


Figure from: **On Network Design Spaces for Visual Recognition.** Radosavovic et al. ICCV 2019.

Why This Shift in Perspective is Important

- Discover general principles
  - *E.g.*, grouped conv (extreme: depthwise conv) is good

- Understand the landscape
  - Are all models basically good? (*e.g.*, insight into NAS)

	#ops	#nodes	output	#cells (B)
NASNet [41]	13	5	L	71,465,842
Amoeba [30]	8	5	L	556,628
PNAS [20]	8	5	A	556,628
ENAS [29]	5	5	L	5,063
DARTS [21]	8	4	A	242

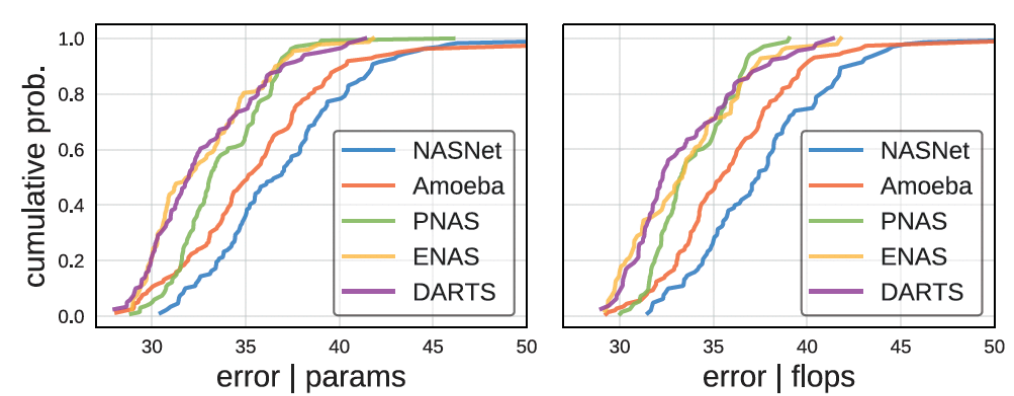
Changes to NAS design spaces in a sequence of NAS papers

Figure from: **On Network Design Spaces for Visual Recognition.** Radosavovic et al. ICCV 2019.

Why This Shift in Perspective is Important

- Discover general principles
  - *E.g.*, grouped conv (extreme: depthwise conv) is good

- Understand the landscape
  - Are all models basically good? (*e.g.*, insight into NAS)



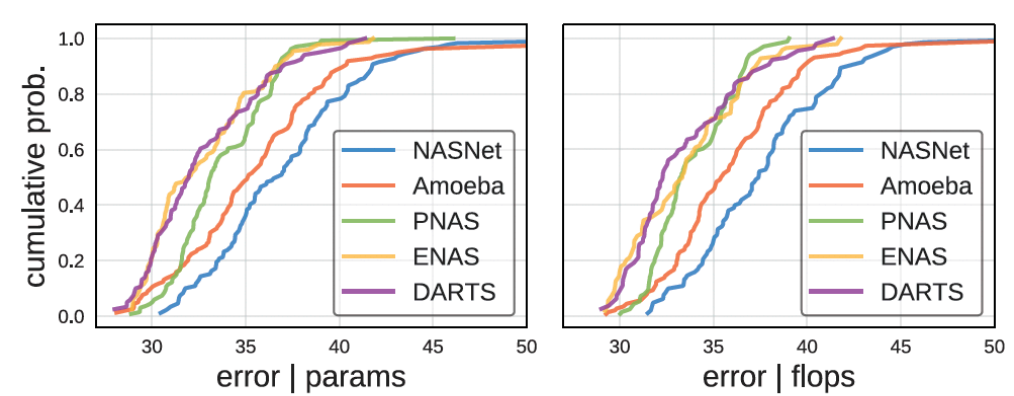
(a) **NAS distribution comparisons** (compare to Fig. 9).

Figure from: **On Network Design Spaces for Visual Recognition.** Radosavovic et al. ICCV 2019.

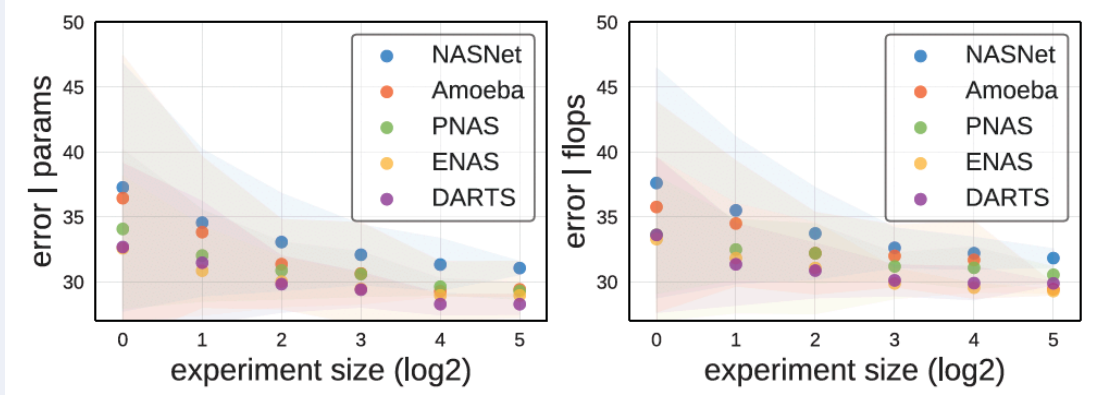
Why This Shift in Perspective is Important

- Discover general principles
  - *E.g.*, grouped conv (extreme: depthwise conv) is good

- Understand the landscape
  - Are all models basically good? (*e.g.*, insight into NAS)



(a) **NAS distribution comparisons** (compare to Fig. 9).

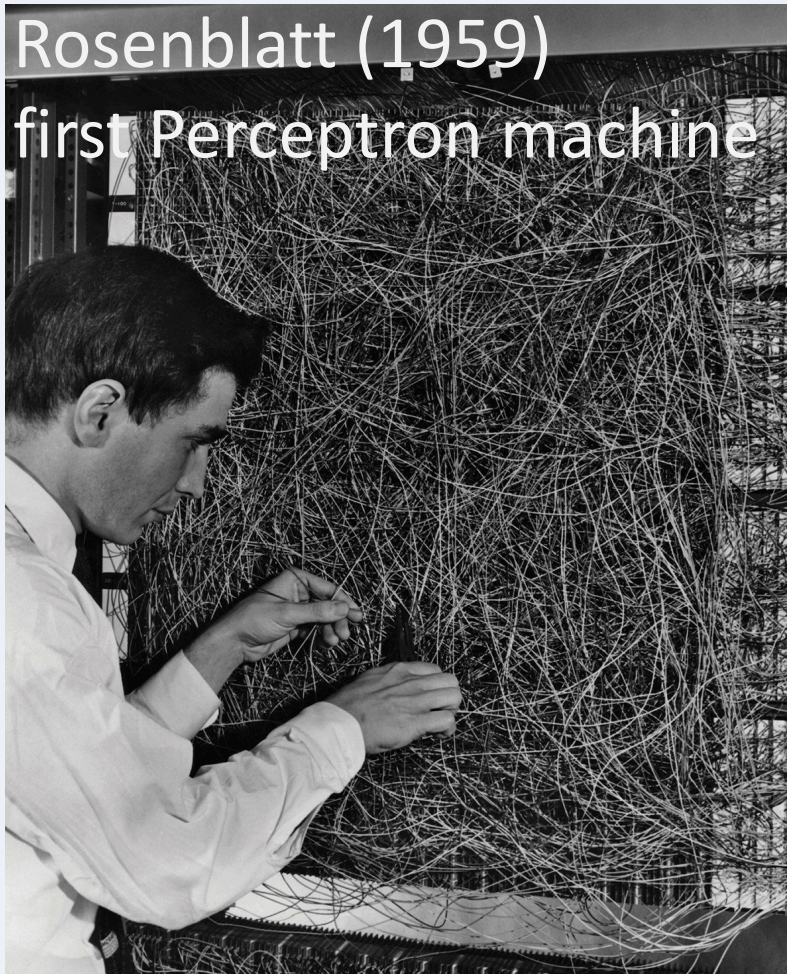


(b) **NAS random search efficiency** (compare to Fig. 10).

Figure from: **On Network Design Spaces for Visual Recognition.** Radosavovic et al. ICCV 2019.

Why This Shift in Perspective is Important

- Discover general principles
  - *E.g.*, grouped conv (extreme: depthwise conv) is good
- Understand the landscape
  - Are all models basically good? (Insight into NAS)
- Explore scientific questions

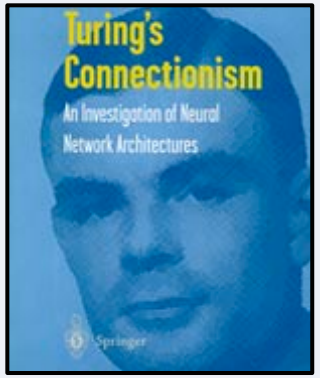


Explore Scientific Questions

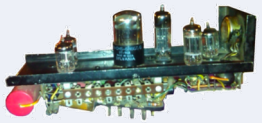
How does *random wiring* compare to *engineered wiring*?

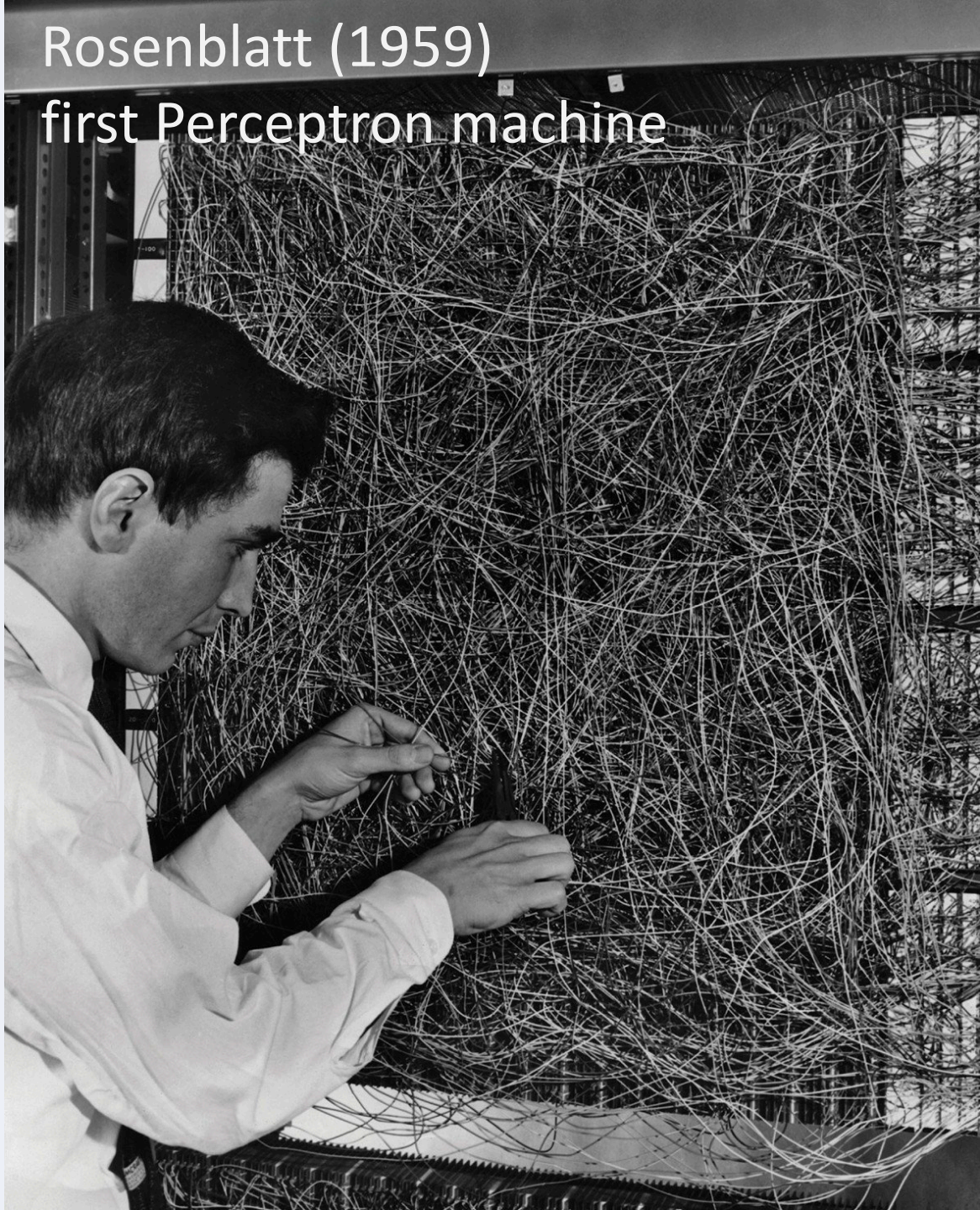
Historical Motivation

Turing (1948)
"Unorganized machines"

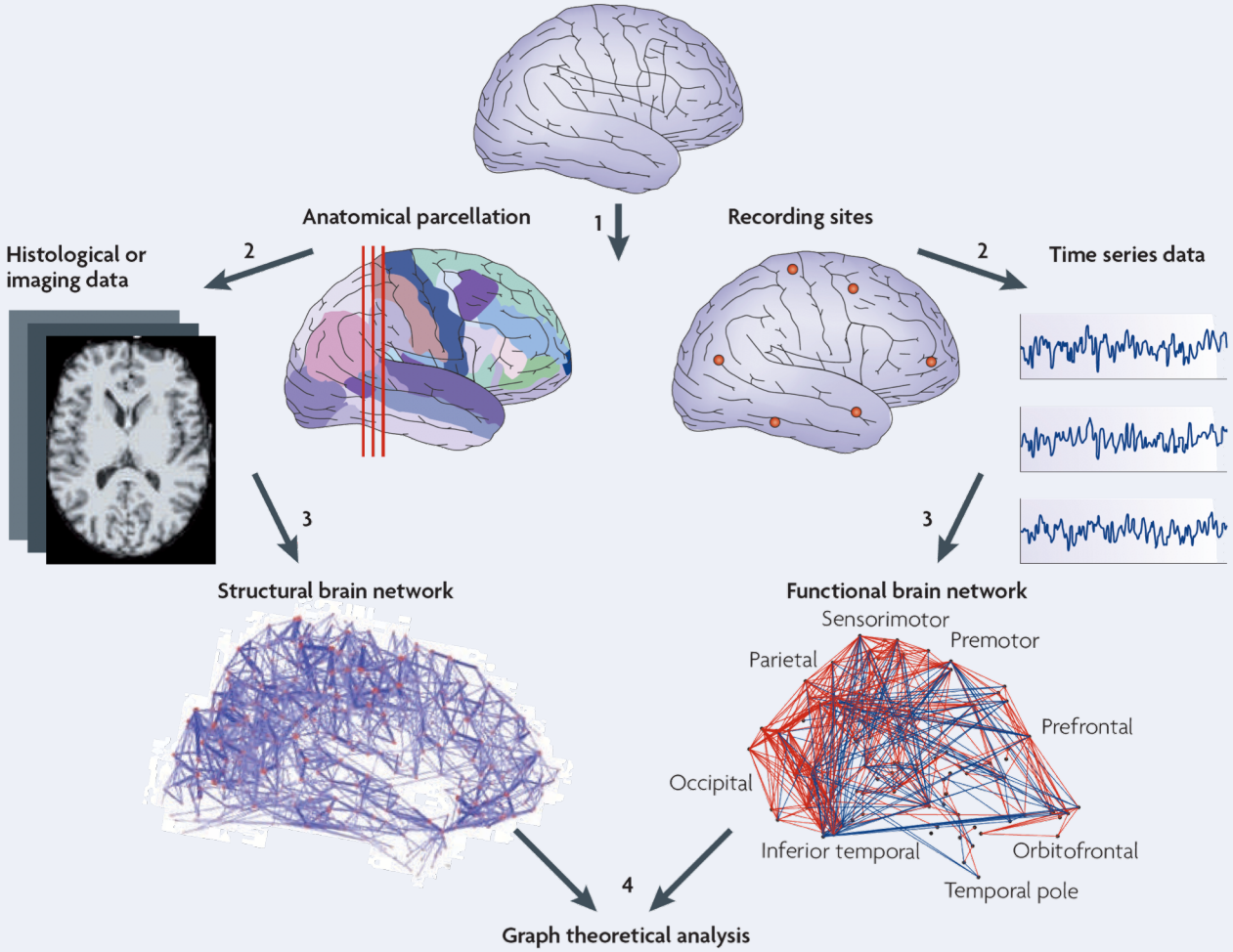


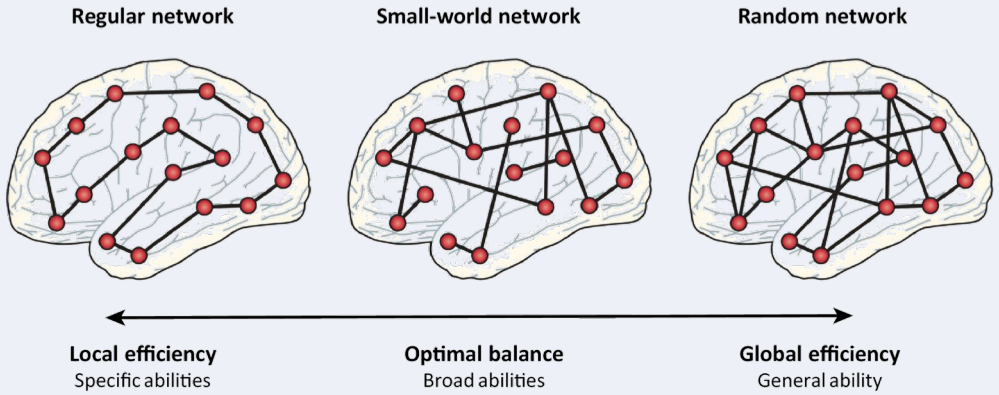
Minsky (1951)
randomly wired "SNARC"





Neuroscience Motivation



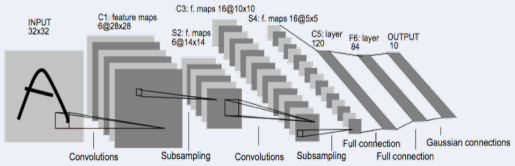


Bullmore and Sporns. "The economy of brain network organization." Nature Reviews Neuroscience, 2012
Bullmore and Sporns. "Complex brain networks: graph theoretical analysis of structural and functional systems." Nature reviews neuroscienc

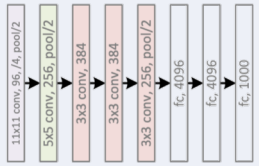
Practical Motivation

chain-like wiring patterns

LeNet



AlexNet

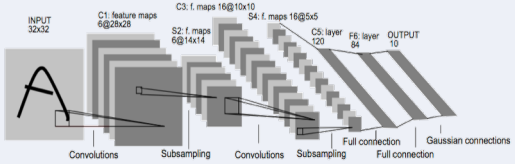


VGGNet

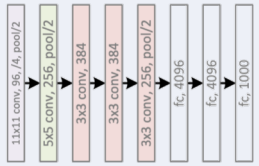
Practical Motivation

chain-like wiring patterns

LeNet



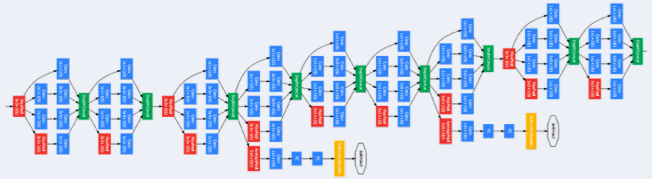
AlexNet



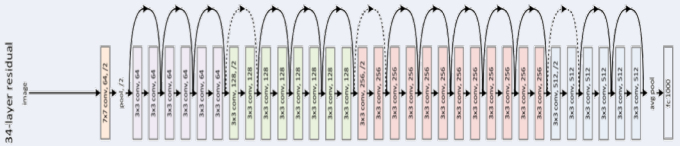
VGGNet

multiple wiring paths

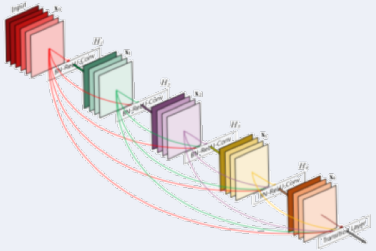
Inception



ResNet

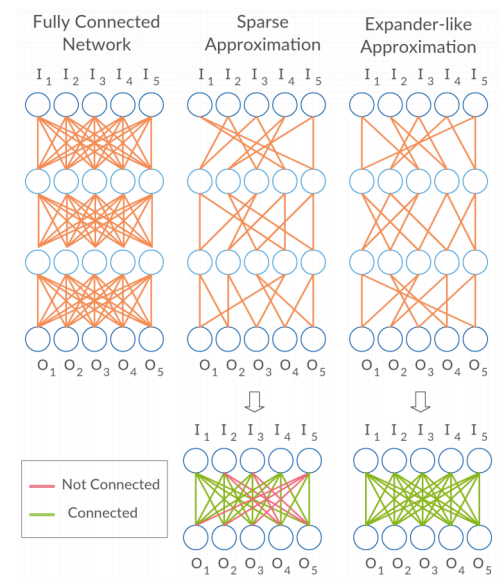


DenseNet



Prior Work: Random Micro Wiring

- Macro vs. micro wiring
  - Macro: wires between “op level” units (*e.g.*, conv) [our focus]
  - Micro: wires inside “op level” units (*e.g.*, sparse conv filters)

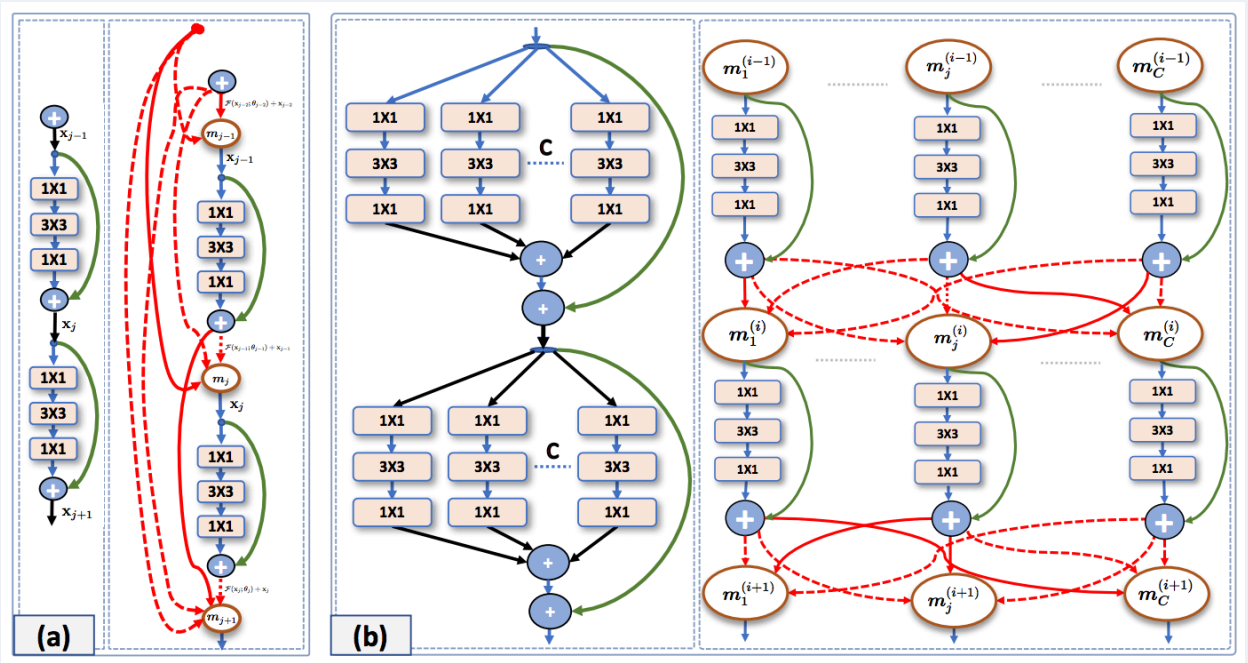


E.g., micro wiring: random expander graphs

Deep Expander Networks: Efficient Deep Networks from Graph Theory.
Ameya Prabhu, Girish Varma, Anoop Namboodiri. ECCV 2018.

Prior Work: Connectivity within ResNe(X)t

- Optimize for weights and connectivity masks with gradient descent
- Focus: learned wiring



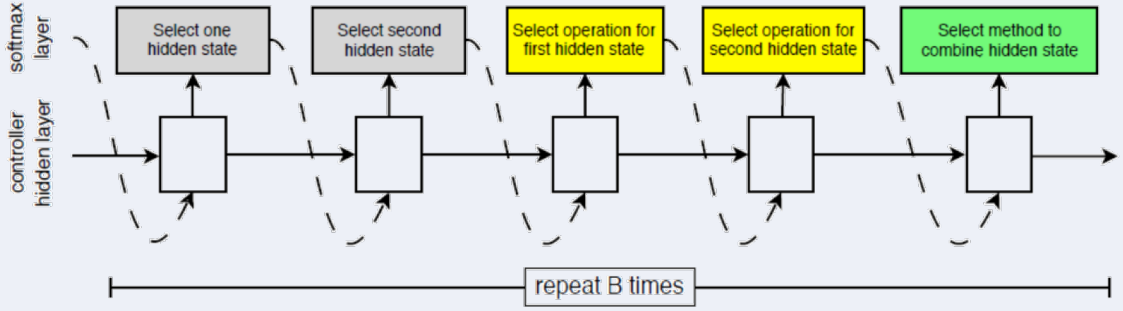
Karim Ahmed and Lorenzo Torresani, "**MaskConnect: Connectivity Learning by Gradient Descent**", ECCV 2018

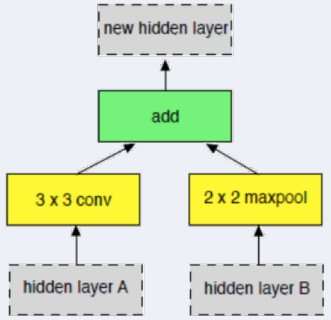
Prior Work: Learned Wiring with NAS

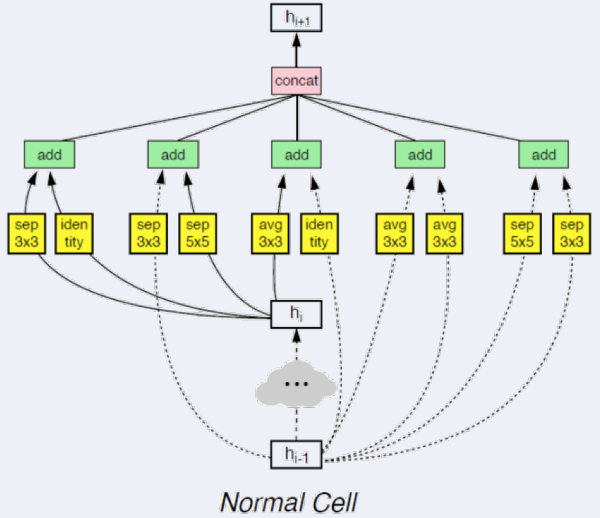
Searching for

(1) operations

(2) wiring patterns







Normal Cell

Barret Zoph, Quoc V. Le, ":Neural Architecture Search with Reinforcement Learning", ICLR 2017

Network Generators (More Formally)

$$g: \Theta \mapsto \mathcal{N}$$

g is a mapping from a parameter space Θ, to a network architecture space $\mathcal{N}$

Network Generators (More Formally)

$$g\colon \Theta \mapsto \mathcal{N}$$

g is a mapping from a parameter space Θ, to a network architecture space $\mathcal{N}$

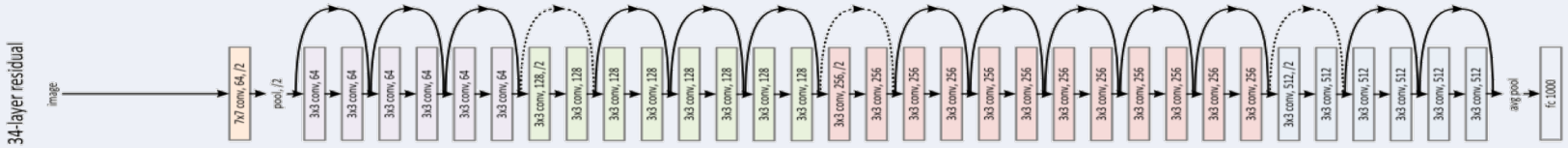
- The parameters $\theta \in \Theta$ specify the instantiated network

Network Generators (More Formally)

$$g: \Theta \mapsto \mathcal{N}$$

g is a mapping from a parameter space Θ, to a network architecture space $\mathcal{N}$

- The parameters $\theta \in \Theta$ specify the instantiated network
- E.g., ResNet generator: θ = {#stages, #blocks per stage, depth, width, filter sizes, …}

$g(\theta) =$ 

Stochastic Network Generators

- The network generator $g(\theta)$ performs a deterministic mapping

(DARTS: Differentiable Architecture Search, Liu et al., 2018)

Stochastic Network Generators

- The network generator $g(\theta)$ performs a deterministic mapping

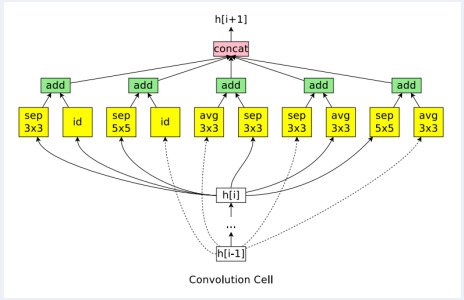
- s: seed of a pseudo-random number generator used by g

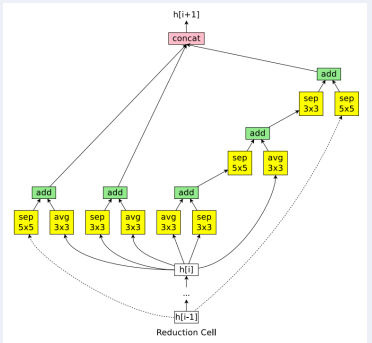
(DARTS: Differentiable Architecture Search, Liu et al., 2018)

Stochastic Network Generators

- The network generator $g(\theta)$ performs a deterministic mapping
- s: seed of a pseudo-random number generator used by g
- $g(\theta, s)$ is a stochastic network generator

$$g(\theta, 1) = \quad , \quad g(\theta, 2) = \quad , \quad \ldots$$





Network Generator Perspective

- Encapsulates the *entire* network generation process
  - Empirical priors (what we already know works well)
  - Learnable parameters θ
  - Randomization s
  - Misc. heuristic rules are revealed

E.g., NAS from the Network Generator Perspective

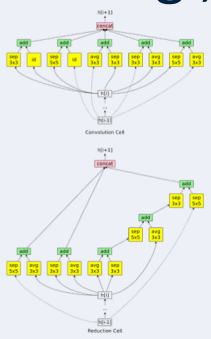
- LSTM controller weights: θ

E.g., NAS from the Network Generator Perspective

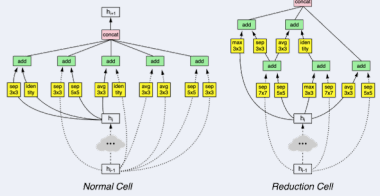
- LSTM controller weights: θ
- Sample actions (insert an conv, connect two nodes, …) determined by θ, s

E.g., NAS from the Network Generator Perspective

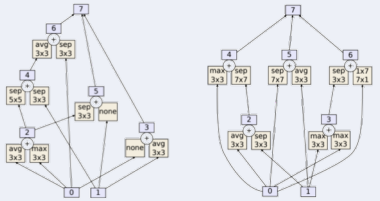
- LSTM controller weights: θ
- Sample actions (insert an conv, connect two nodes, …) determined by θ, s
- Allowed wiring patterns is a small subset of all possible graphs
  - E.g., 5 nodes in a cell always have input degree 2 and output degree 1



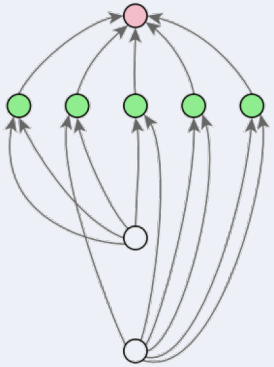
ENAS, ICML 2018



NASNet-A, CVPR 2018



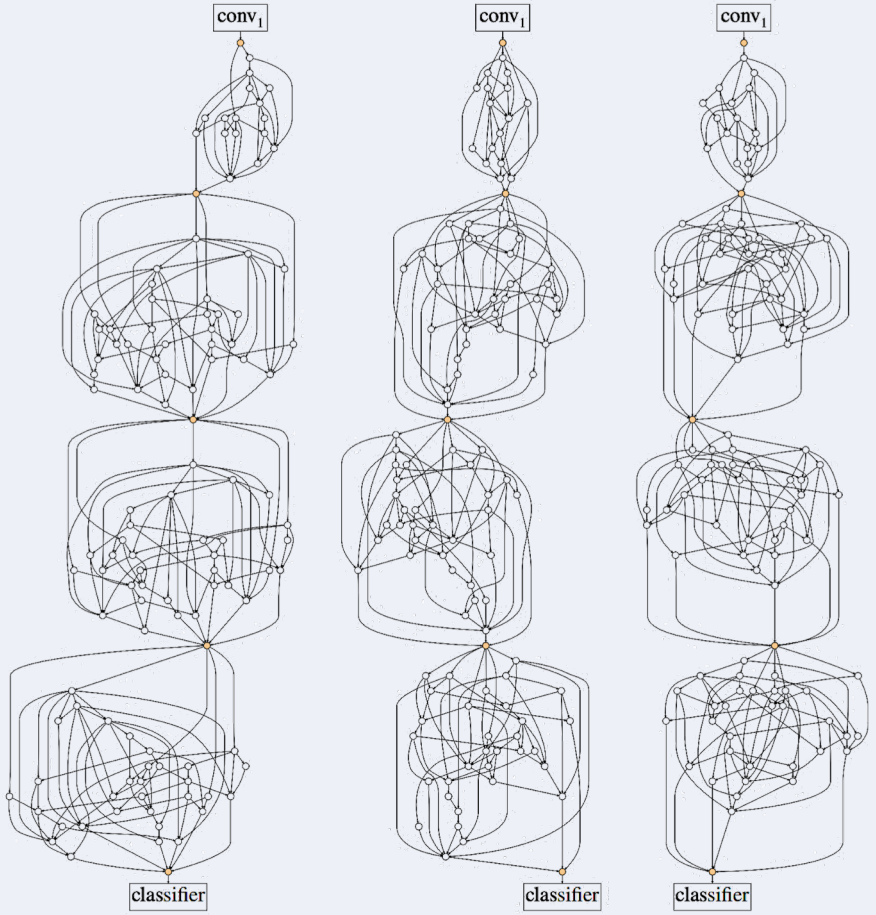
Amoeba Net, AAAI 2019



The generator design involves a strong prior

"Misc. heuristic rules"

Network Generators for Randomly Wired Networks



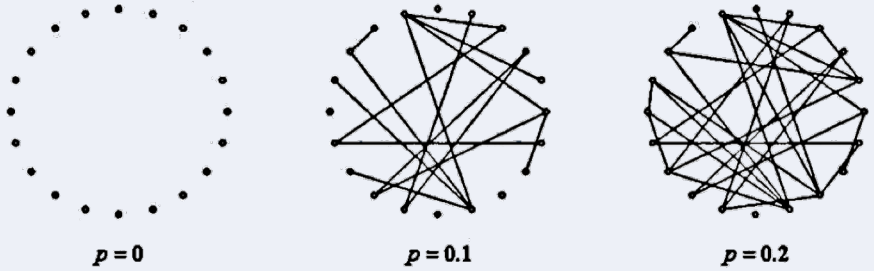
How do we design neural networks like these?

Step 1. Generate a general graph G

Step 2. Map graph G to a valid neural network

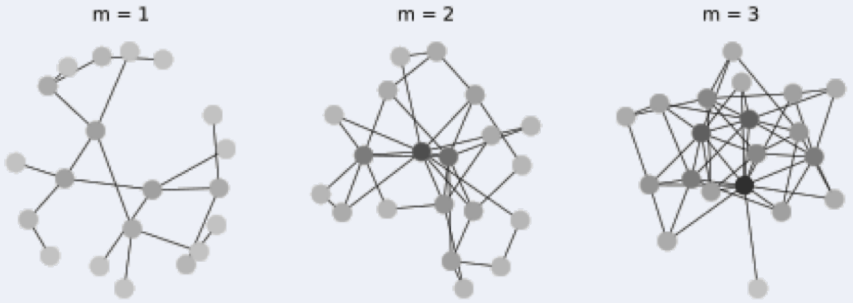
Random Graph Models

Erdös-Rényi (ER), 1959



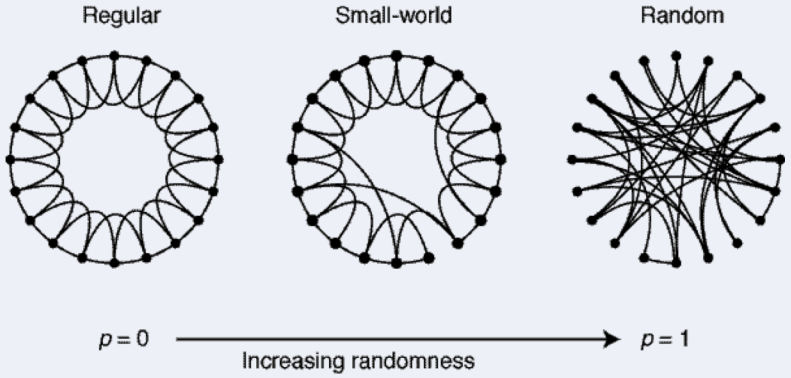
Barabási–Albert (BA), 1998

“Scale-free” graphs



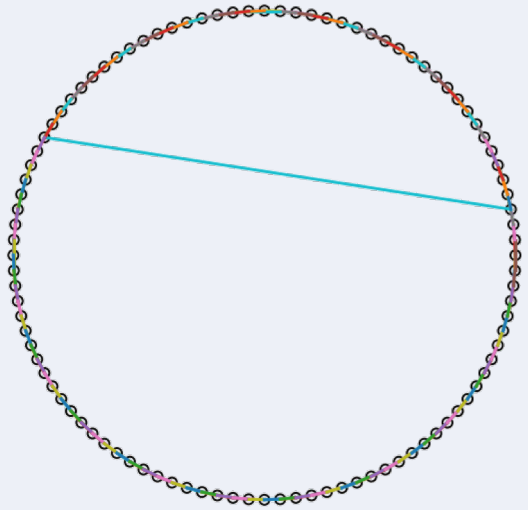
Watts–Strogatz (WS), 1998

“Small-world” graphs

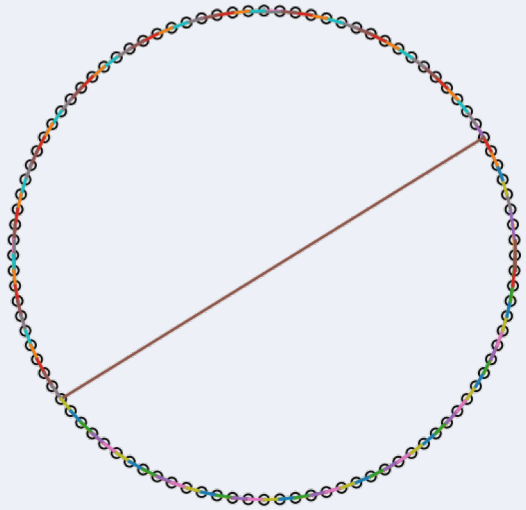


Example: Small-World Ggraphs

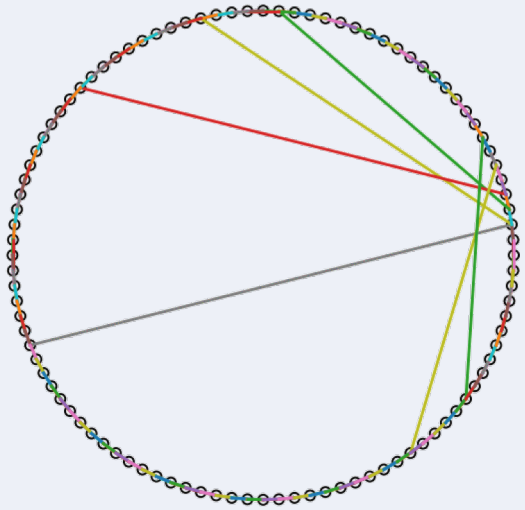
100 nodes, average degree = 4



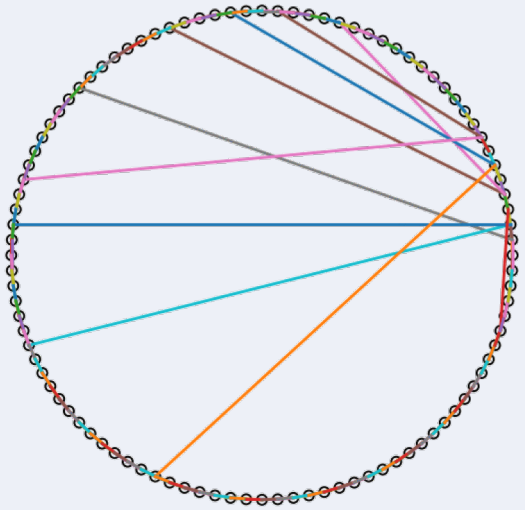
P = 0.1



P = 0.25



P = 0.5



P = 0.9

More Regular

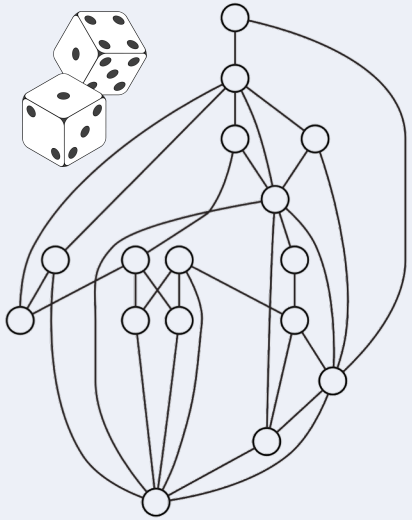
“Small world”

More “Random”

Rewiring probability P

Network Generator

(random)
general graph

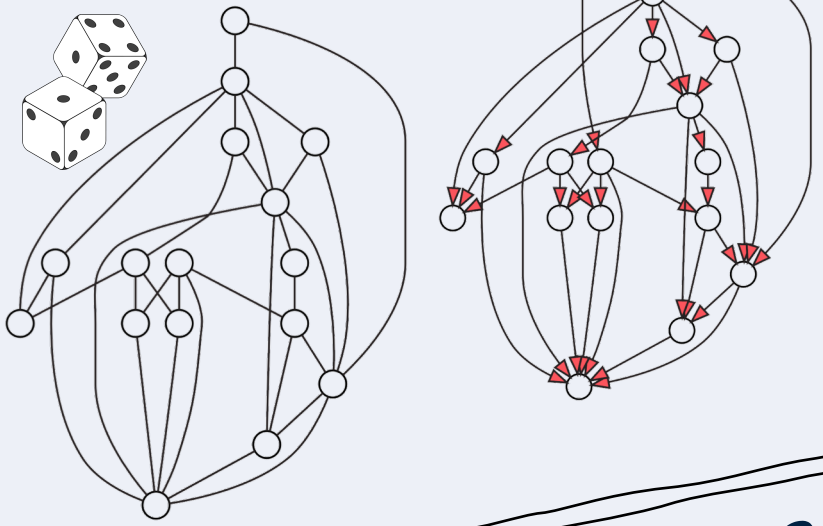


From a graph to a neural network

Network Generator

convert to DAG

(random)
general graph



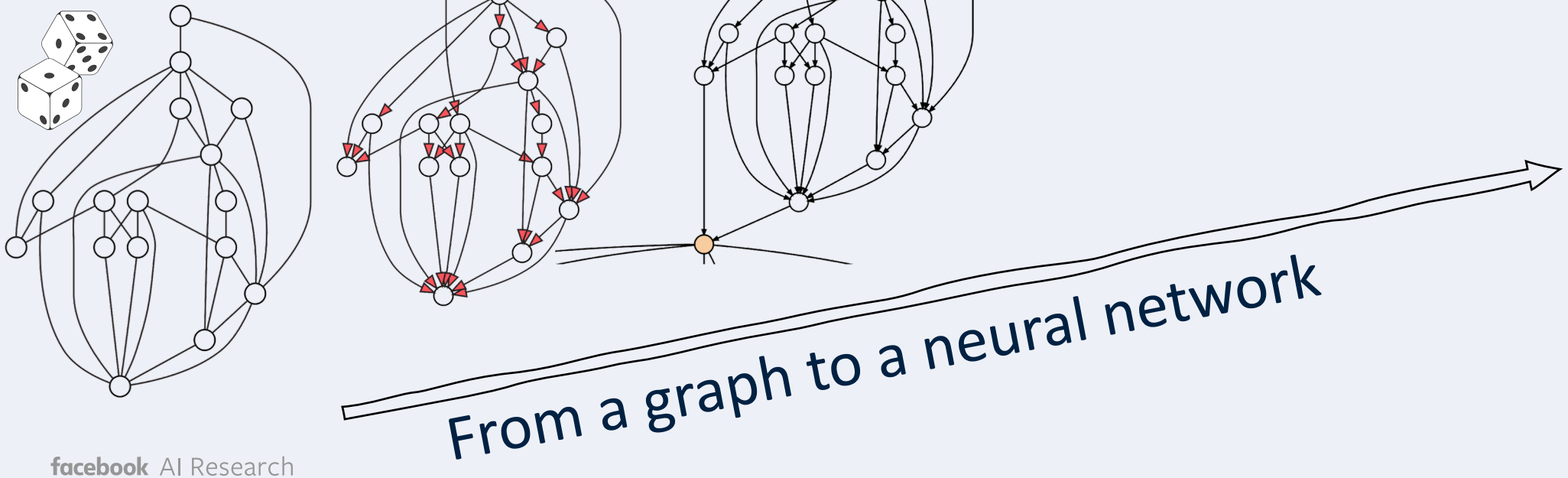
From a graph to a neural network

Network Generator

add extra input &
output node

convert to DAG

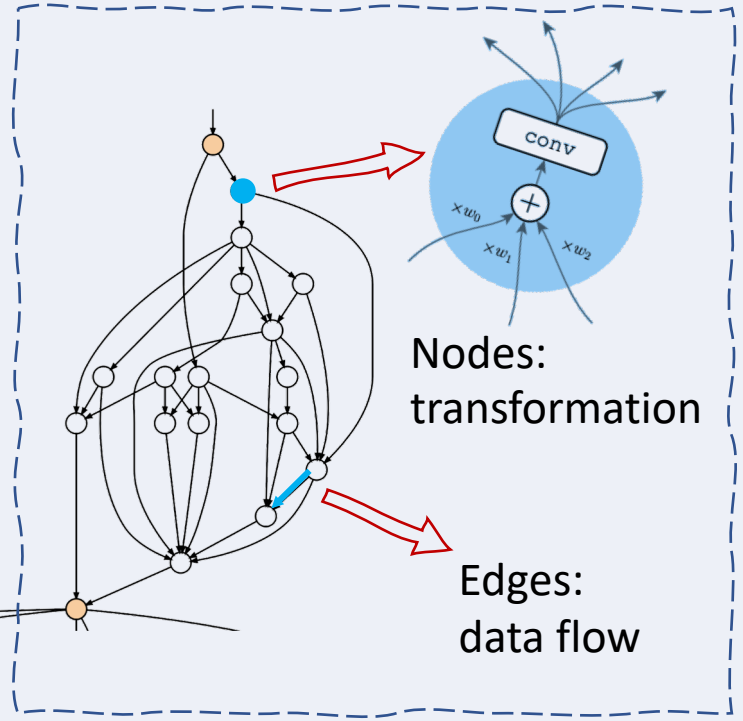
(random)
general graph



From a graph to a neural network

Network Generator

what are nodes and edges?



add extra input &
output node

convert to DAG

(random)
general graph

From a graph to a neural network

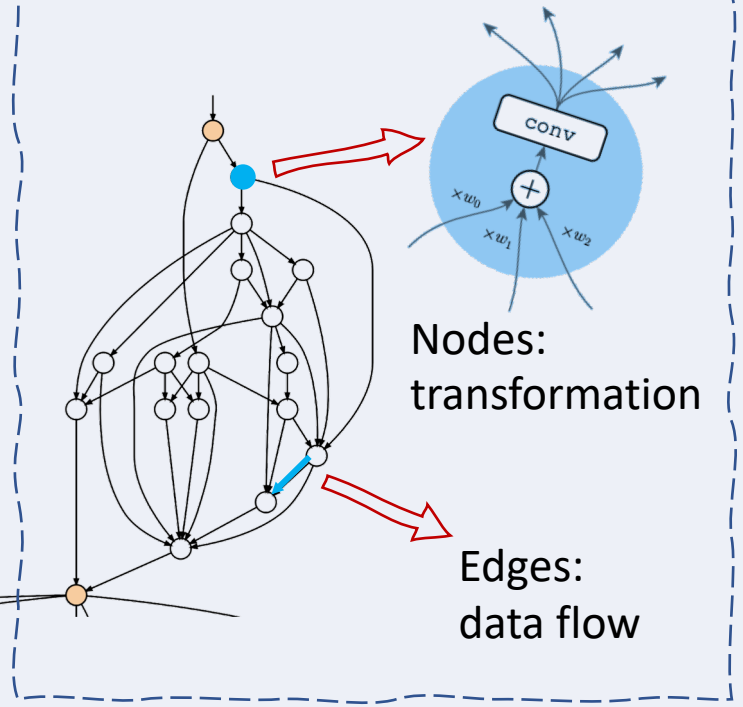
Network Generator

(random)
general graph

convert to DAG

add extra input &
output node

what are nodes and edges?



one graph =
one stage

From a graph to a neural network

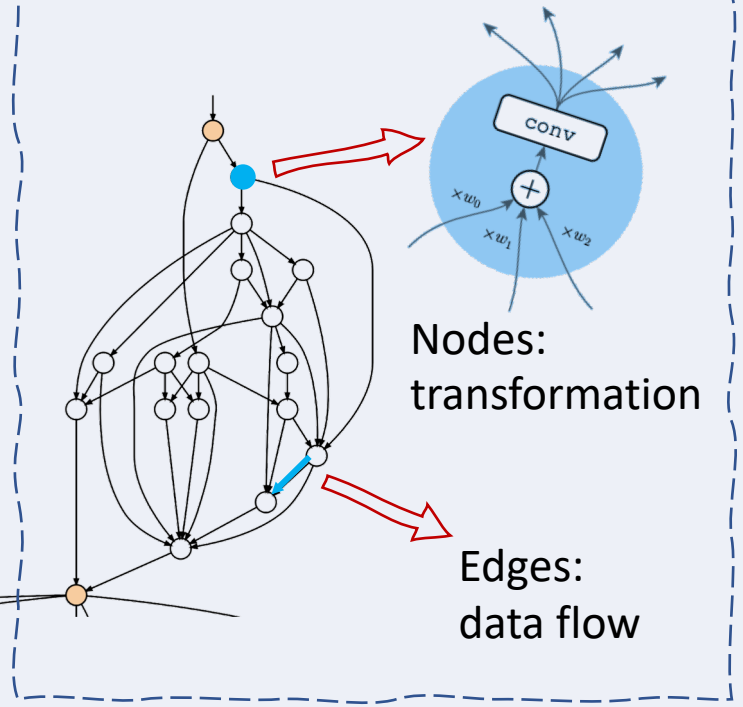
Network Generator

(random)
general graph

convert to DAG

add extra input &
output node

what are nodes and edges?



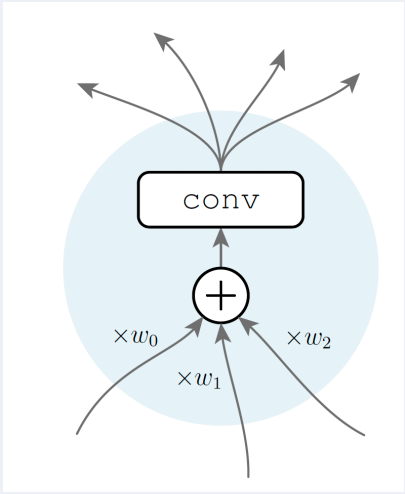
one graph
one stage

From a graph to a neural network

"Misc. heuristic rules"

Notes on Nodes and Edges

- Overall computation $\propto$ # nodes
- # nodes is fixed
- Tiny extra computation on each edge ($\times w_i$)



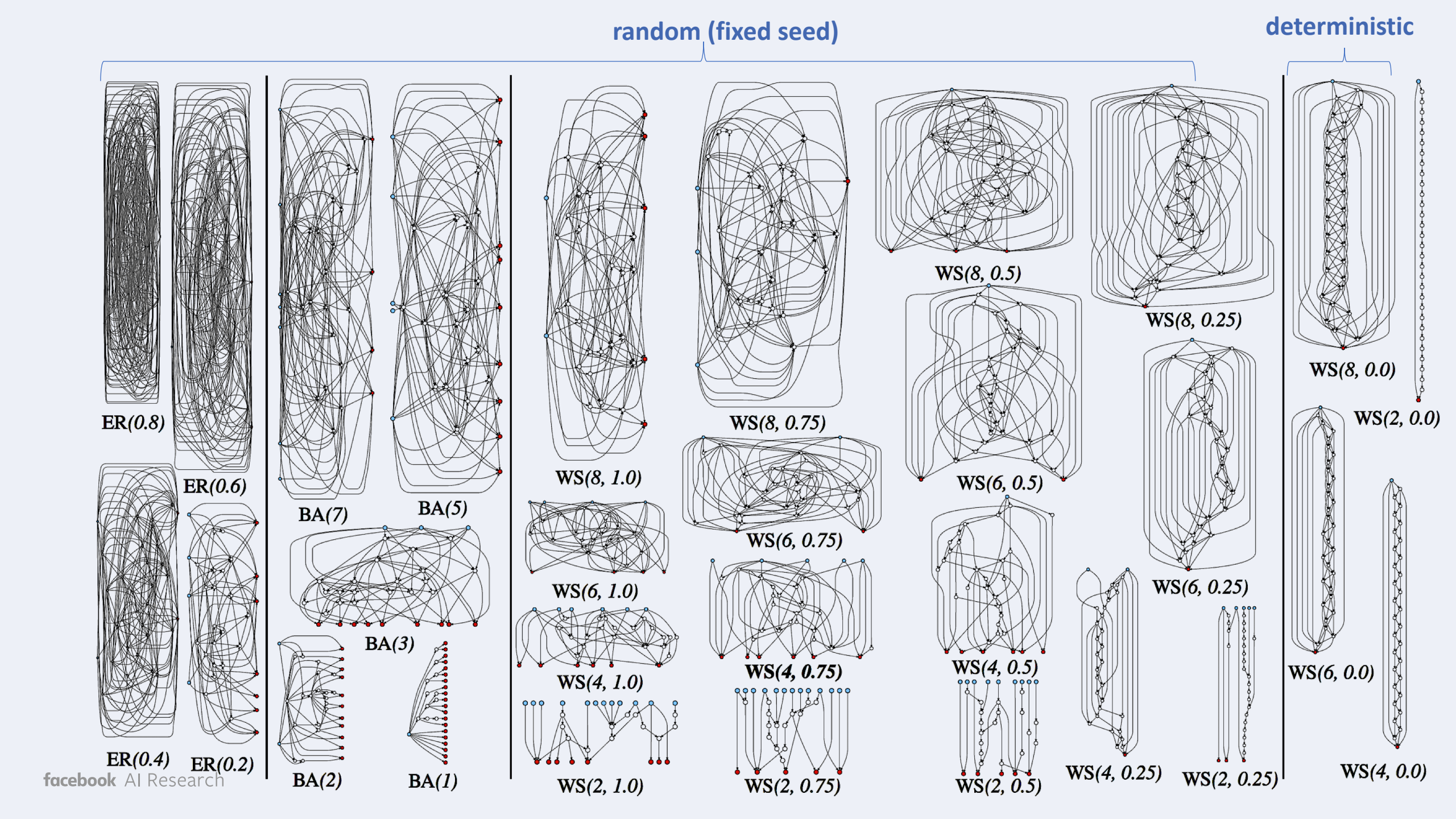
Activation copies

ReLU, conv, BN

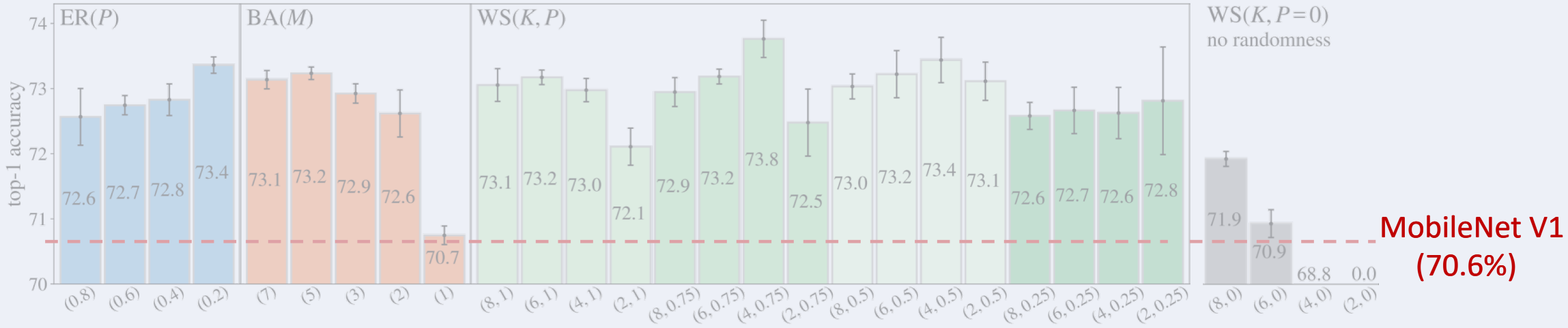
Weighted sum

Input activations

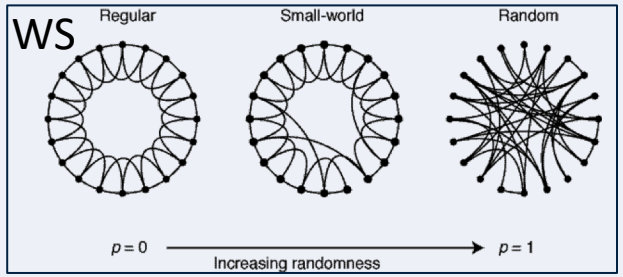
- Differences in accuracy are about wiring, not FLOPs or parameters


random (fixed seed)
deterministic
ER(0.8)
ER(0.6)
ER(0.4)
ER(0.2)
BA(7)
BA(5)
BA(3)
BA(2)
BA(1)
WS(8, 1.0)
WS(6, 1.0)
WS(4, 1.0)
WS(2, 1.0)
WS(8, 0.75)
WS(6, 0.75)
WS(4, 0.75)
WS(2, 0.75)
WS(8, 0.5)
WS(6, 0.5)
WS(4, 0.5)
WS(2, 0.5)
WS(8, 0.25)
WS(6, 0.25)
WS(4, 0.25)
WS(2, 0.25)
WS(8, 0.0)
WS(6, 0.0)
WS(4, 0.0)
WS(2, 0.0)
facebook AI Research

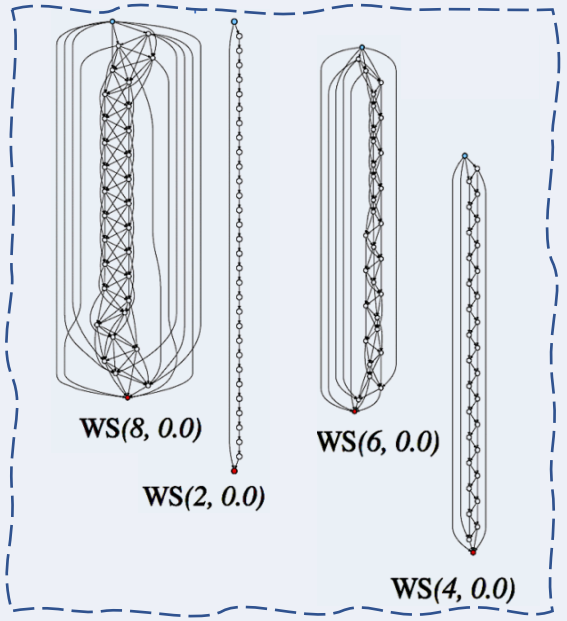
Small Model Regime – Engineered Baseline



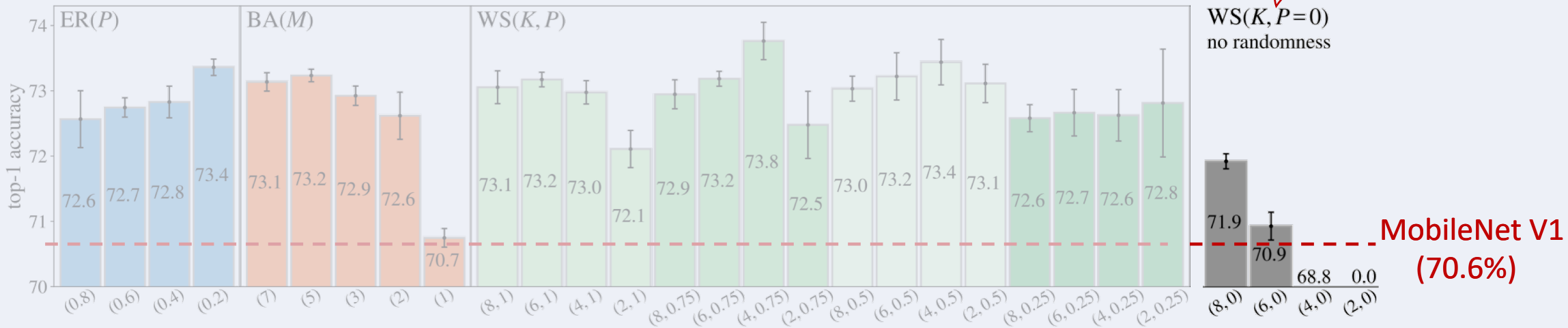
All models: ~ 580M FLOPs



Extreme case: deterministic

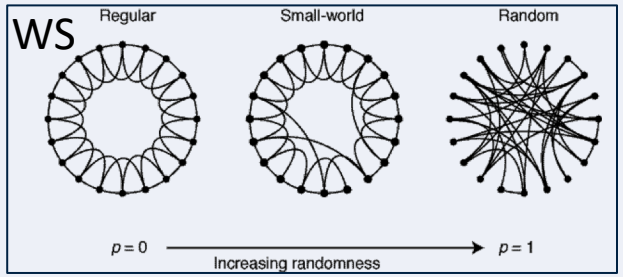


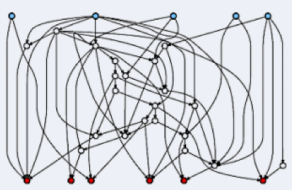
(68.8% - 71.9%)



MobileNet V1 (70.6%)

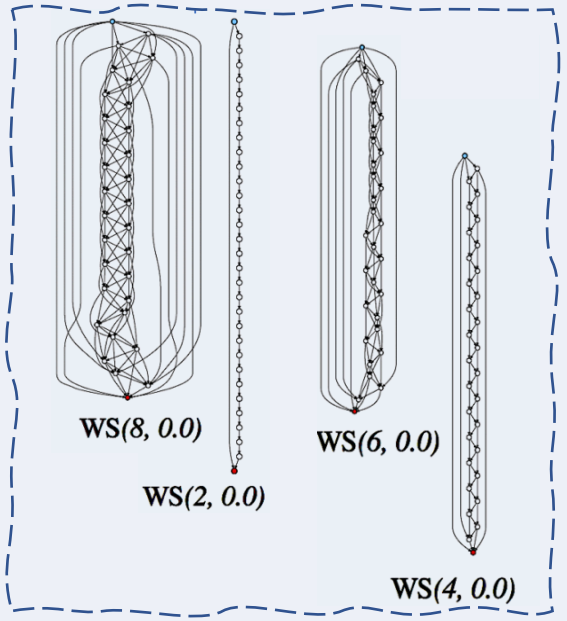
All models: ~ 580M FLOPs



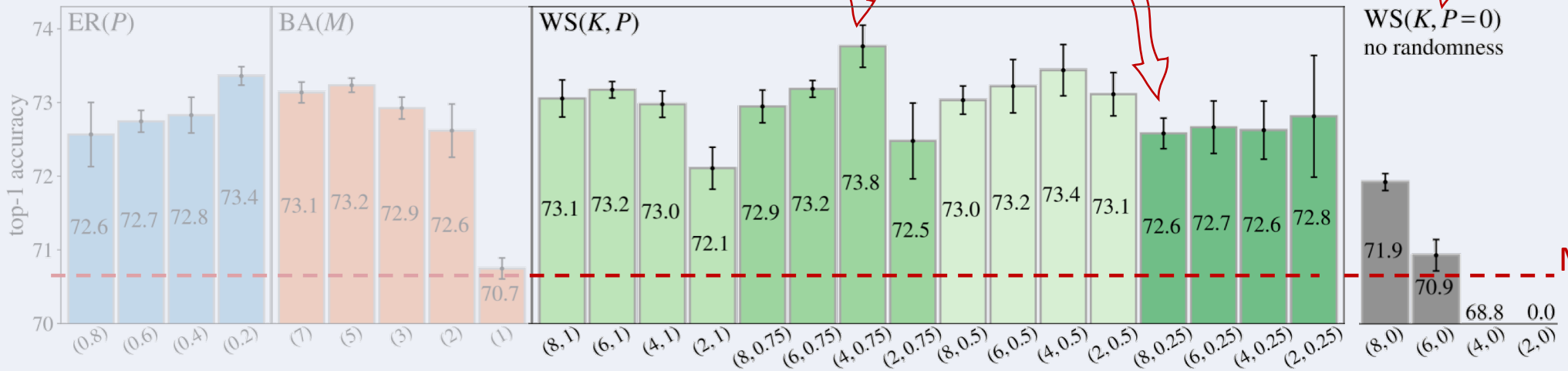


(73.8%)

(72.6%)



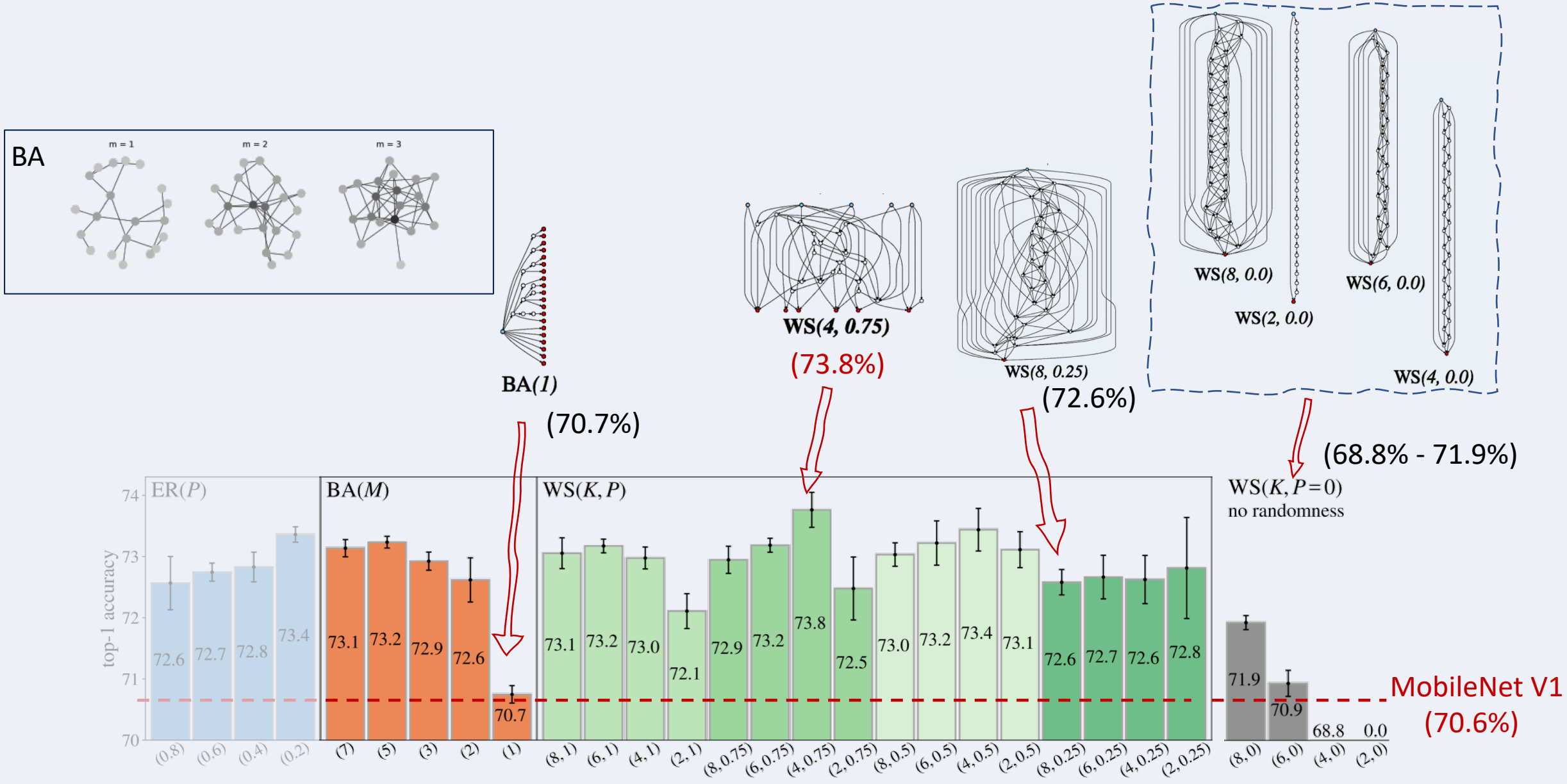
(68.8% - 71.9%)



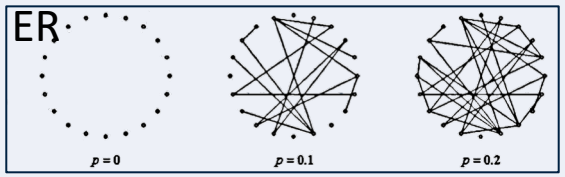
MobileNet V1 (70.6%)

facebook AI Research

All models: ~ 580M FLOPs

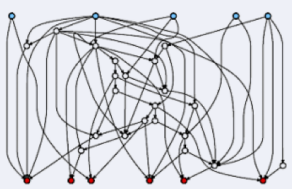


All models: ~ 580M FLOPs

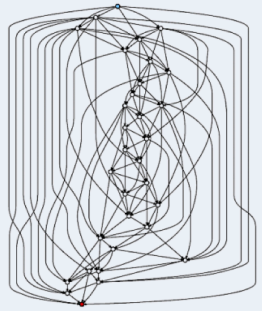


(72.8%)

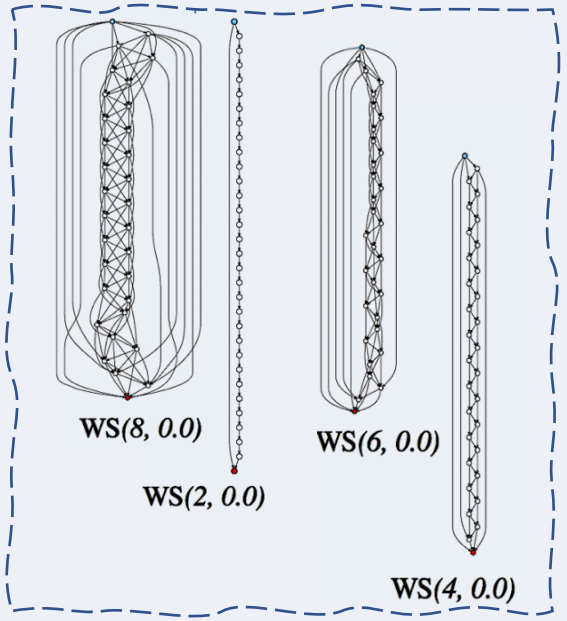
(70.7%)



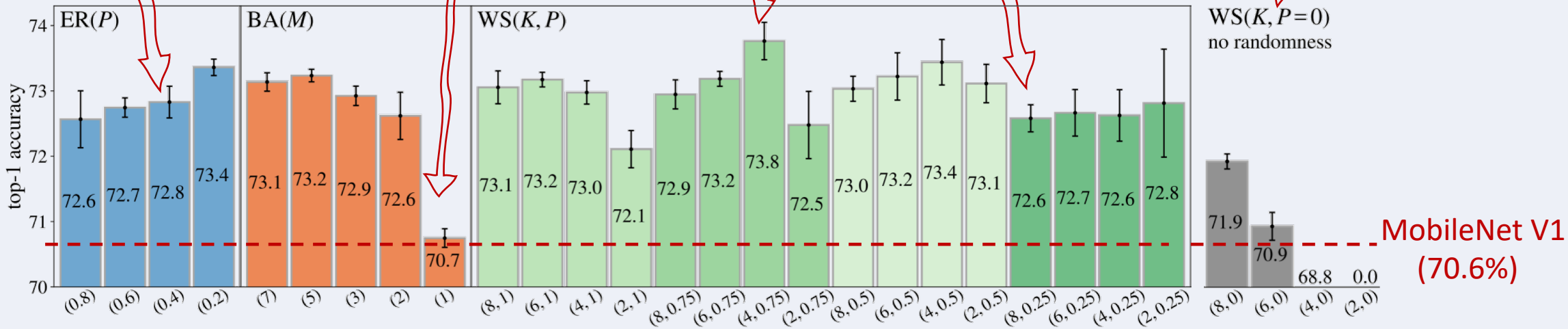
(73.8%)



(72.6%)



(68.8% - 71.9%)



MobileNet V1 (70.6%)

All models: ~ 580M FLOPs

Results – Small Computation Regime (< 600M FLOPs)

	Network	Top-1 Acc	FLOPs
Hand designed	MobileNet	70.6	569
	MobileNet v2	74.7	585
	ShuffleNet	73.7	524
	ShuffleNet v2	74.9	591

Results – Small Computation Regime (< 600M FLOPs)

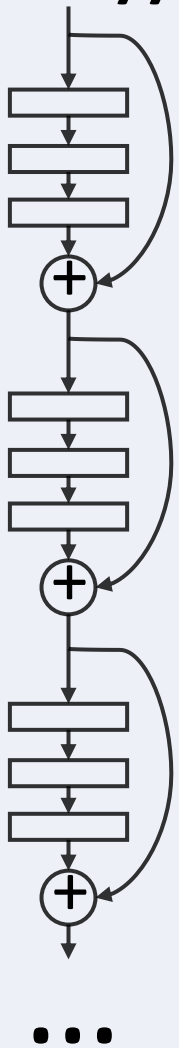
	Network	Top-1 Acc	FLOPs
Hand designed	MobileNet	70.6	569
	MobileNet v2	74.7	585
	ShuffleNet	73.7	524
	ShuffleNet v2	74.9	591
NAS	NASNet-A	74.0	564
	NASNet-C	72.5	558
	AmoebaNet-A	74.5	555
	AmoebaNet-C	75.7	570
	PNAS	74.2	588
	DARTS	73.1	595

Results – Small Computation Regime (< 600M FLOPs)

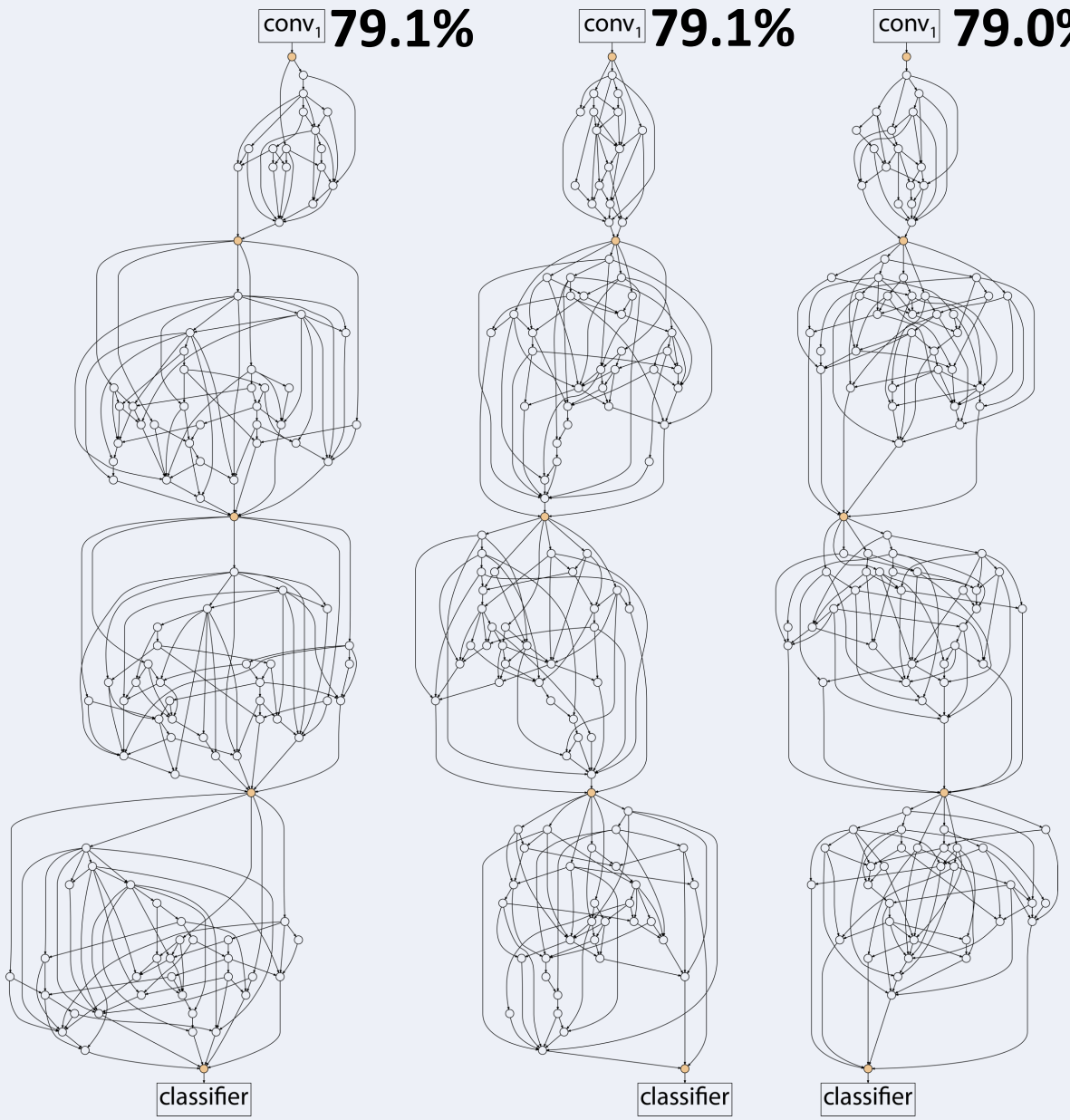
	Network	Top-1 Acc	FLOPs
Hand designed	MobileNet	70.6	569
	MobileNet v2	74.7	585
	ShuffleNet	73.7	524
	ShuffleNet v2	74.9	591
NAS	NASNet-A	74.0	564
	NASNet-C	72.5	558
	AmoebaNet-A	74.5	555
	AmoebaNet-C	75.7	570
	PNAS	74.2	588
	DARTS	73.1	595
Randomly Wired	**RandWire-WS**	**74.7** $\pm$0.25	**583** $\pm$6.2

ResNet-50

ImageNet Top1 Acc: **77.1%**



RandWire (same FLOPs)



Conclusions

- The next frontier: Architecting network generators?
- Study *populations* of networks
- We explored randomly wired nets – *they can work surprisingly well!*
- Heuristic rules are pervasive even in “random” and “automatic” generators