MIT HAN LAB

Lecture 07
Neural Architecture Search

Part I

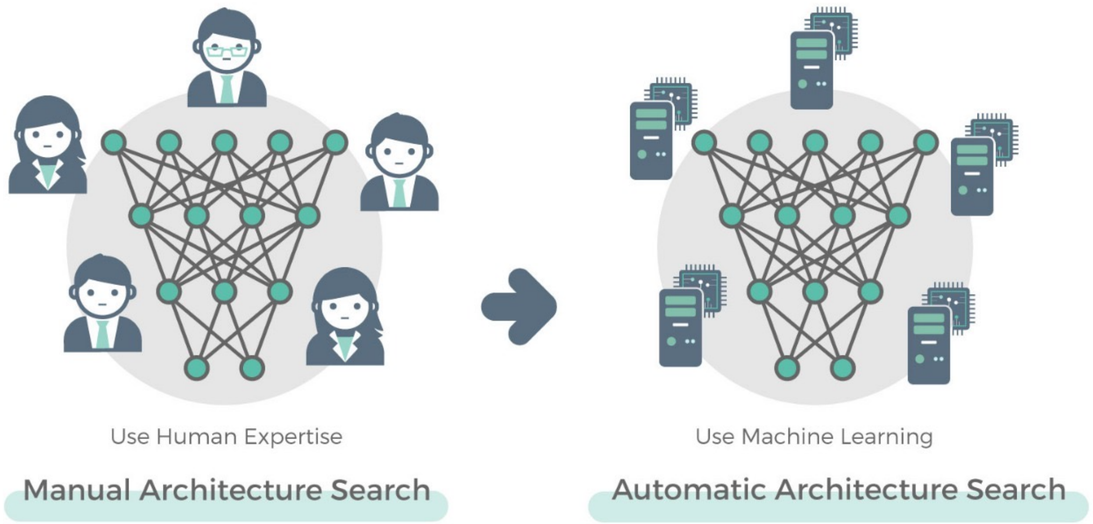
Song Han
songhan@mit.edu

Lecture Plan

Today we will:

1. Introduce basic concepts of neural network architectures and discuss the design principles of several typical manually-design neural network architectures;

2. Introduce neural architecture search, an automatic technique for designing neural network architectures.



Neural Network Architecture

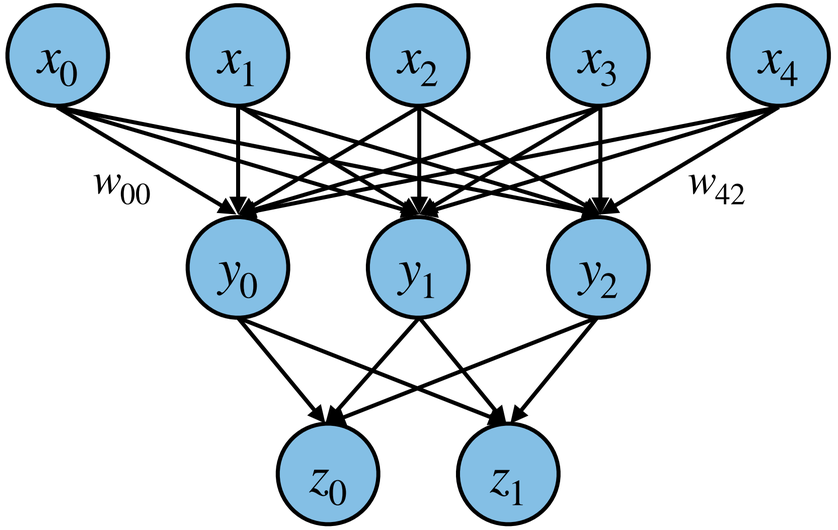
Basic concepts and design principles

Recap: Primitive Operations

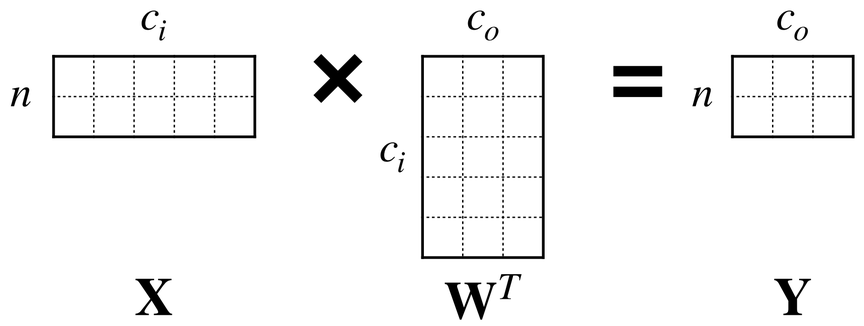
Fully-connected layer / linear layer

- **Shape of Tensors:**
  - Input Features $\mathbf{X}$: (n, c_i)
  - Output Features $\mathbf{Y}$: (n, c_o)
  - Weights $\mathbf{W}$: (c_o, c_i)
  - Bias $\mathbf{b}$: $(c_o,)$

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels



Multilayer Perceptron (MLP)

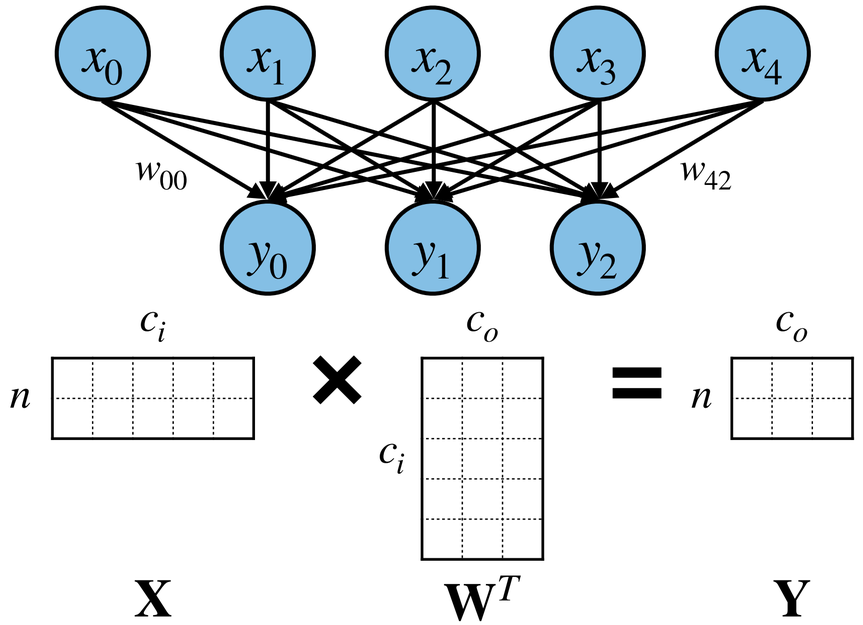


Recap: Primitive Operations

Fully-connected layer / linear layer

Layer	MACs (batch size n=1)
Linear Layer	$c_o \cdot c_i$

* bias is ignored



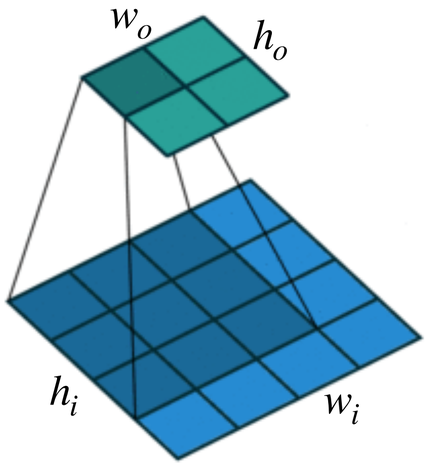
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
h_i, h_o	Input/Output Height
w_i, w_o	Input/Output Width
k_h, k_w	Kernel Height/Width
g	Groups

Recap: Primitive Operations

Convolution layer

- **Shape of Tensors:**

	1D Conv	2D Conv
• Input Features $\mathbf{X}$: ~~(n, c_i)~~	(n, c_i, w_i)	(n, c_i, h_i, w_i)
• Output Features $\mathbf{Y}$: ~~(n, c_o)~~	(n, c_o, w_o)	(n, c_o, h_o, w_o)
• Weights $\mathbf{W}$: ~~(c_o, c_i)~~	(c_o, c_i, k_w)	(c_o, c_i, k_h, k_w)
• Bias $\mathbf{b}$: $(c_o,)$		



Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
h_i, h_o	Input/Output Height
w_i, w_o	Input/Output Width
k_h	Kernel Height
k_w	Kernel Width

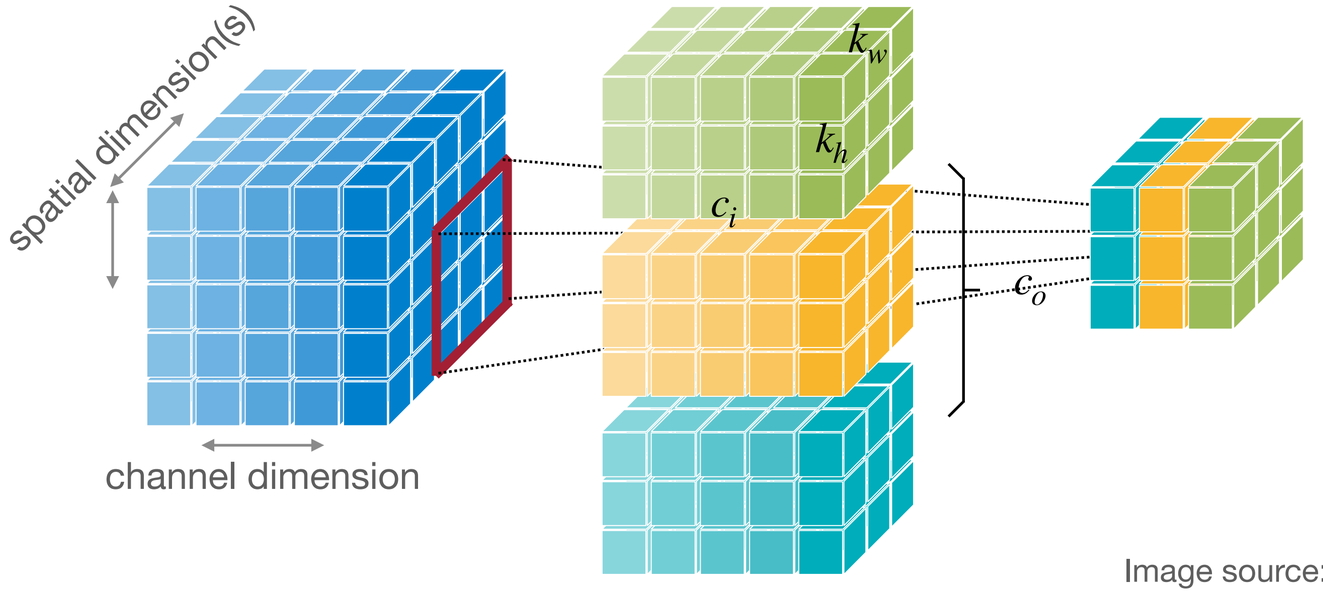


Image source: 1

Recap: Primitive Operations

Convolution layer

Layer	MACs (batch size n=1)
Linear Layer	$c_o \cdot c_i$
Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w \cdot h_o \cdot w_o$

* bias is ignored

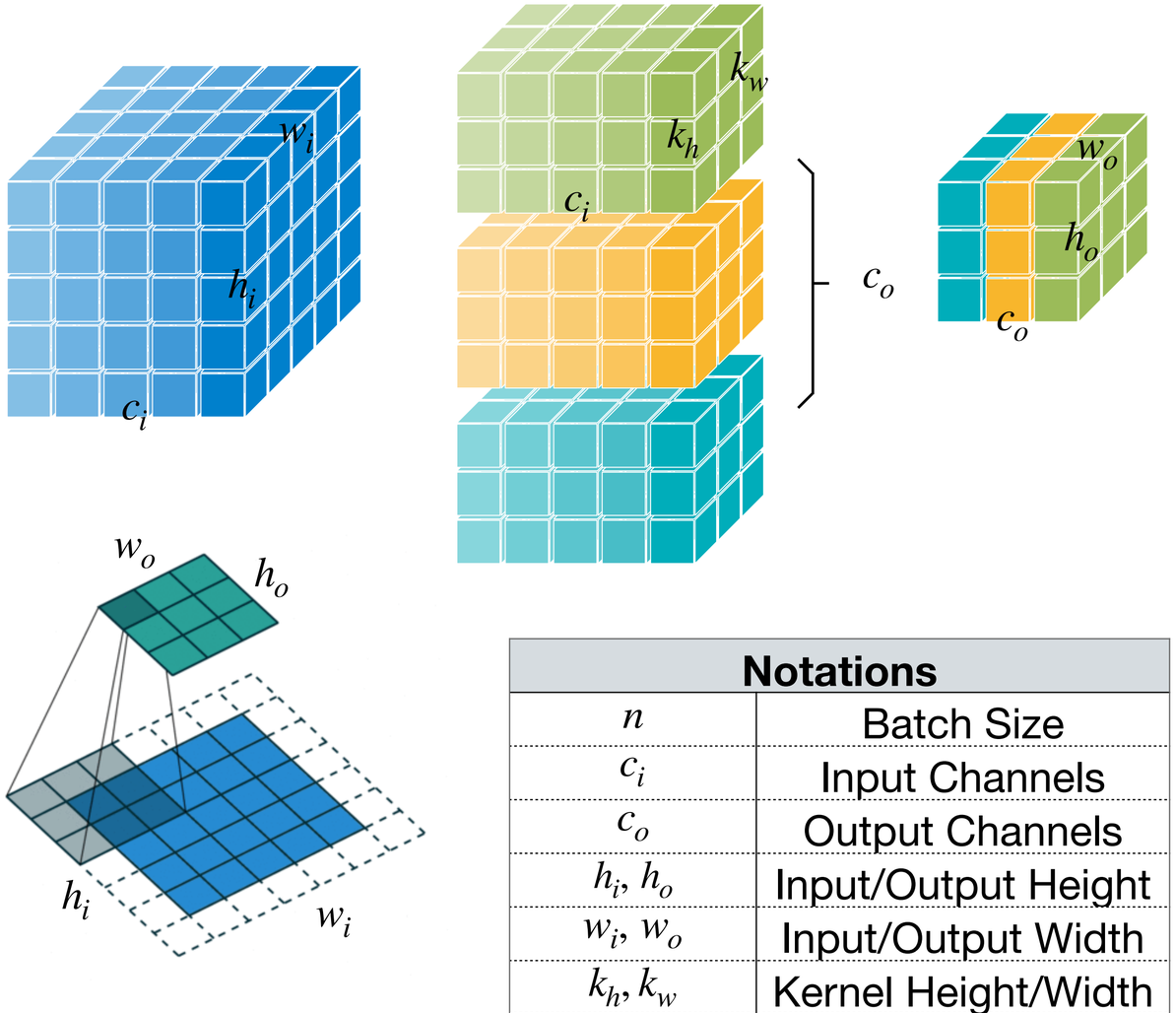


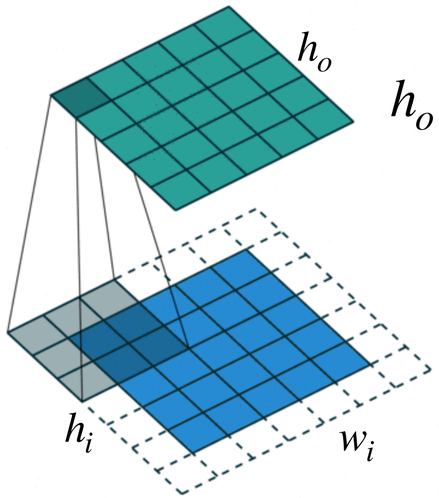
Image source: 1

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
h_i, h_o	Input/Output Height
w_i, w_o	Input/Output Width
k_h, k_w	Kernel Height/Width
g	Groups

Recap: Primitive Operations

Grouped convolution layer

- **Shape of Tensors:**
  - Input Features $\mathbf{X}$: (n, c_i, h_i, w_i)
  - Output Features $\mathbf{Y}$: (n, c_o, h_o, w_o)
  - Weights $\mathbf{W}$: ~~(c_o, c_i, k_h, k_w)~~ $(g \cdot c_o/g, c_i/g, k_h, k_w)$
  - Bias $\mathbf{b}$: $(c_o,)$



$$h_o = \frac{h_i + 2p - k_h}{s} + 1$$

p is padding

s is stride

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
h_i, h_o	Input/Output Height
w_i, w_o	Input/Output Width
k_h	Kernel Height
k_w	Kernel Width
g	Groups

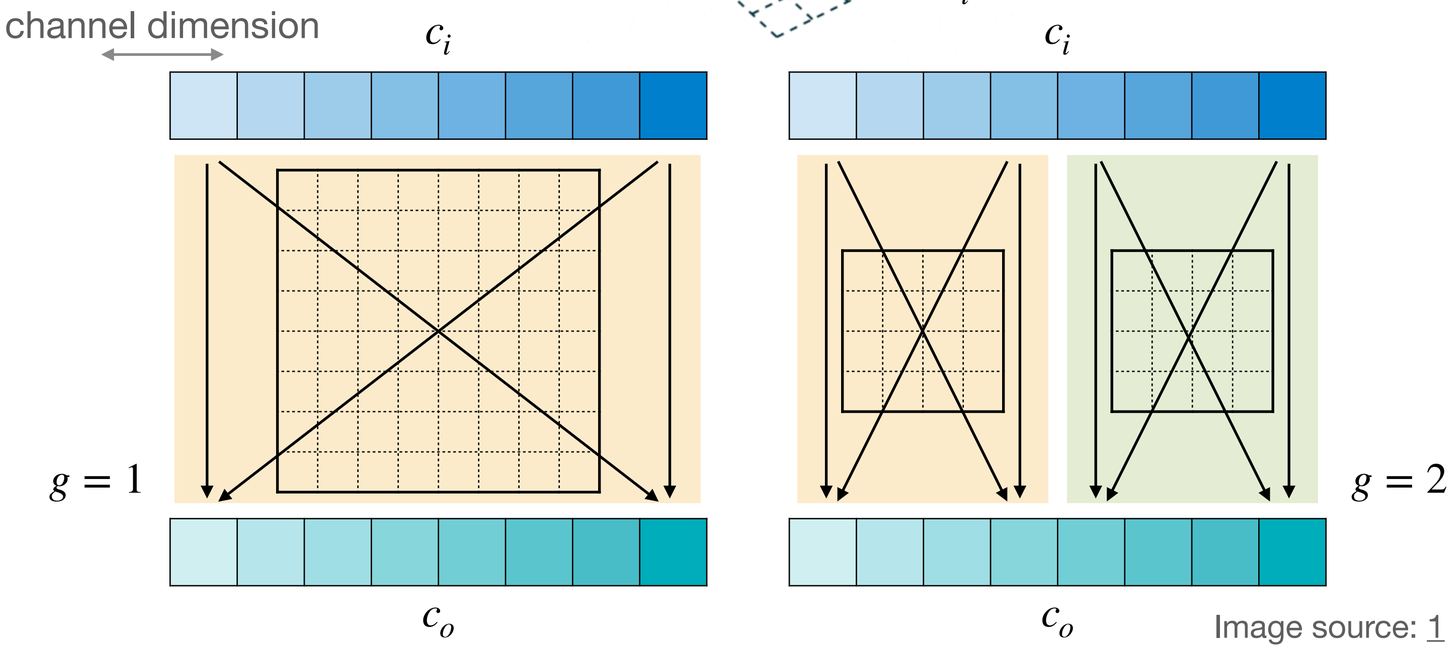


Image source: 1

Recap: Primitive Operations

Grouped convolution layer

Layer	MACs (batch size n=1)
Linear Layer	$c_o \cdot c_i$
Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w \cdot h_o \cdot w_o$
Grouped Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w \cdot h_o \cdot w_o / g$

* bias is ignored

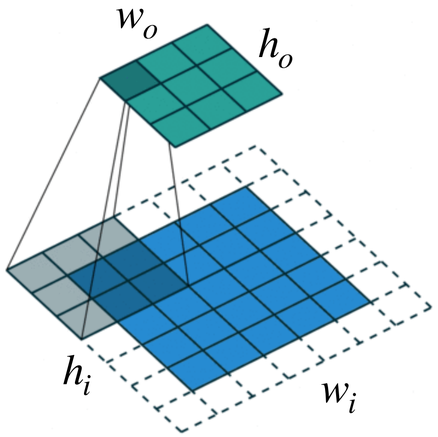
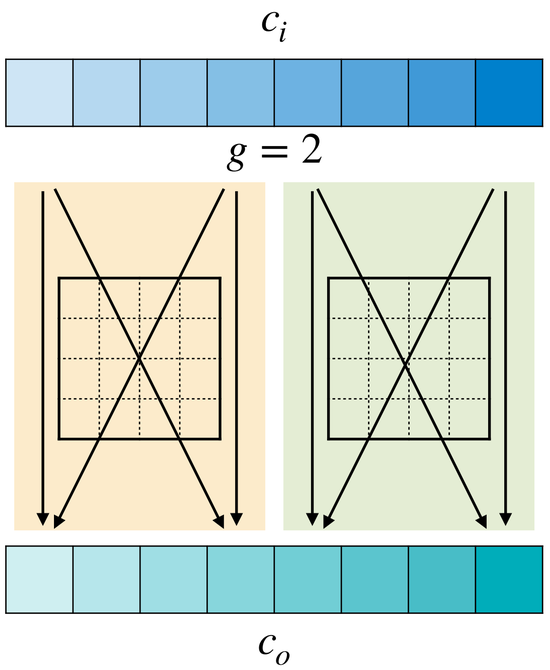


Image source: 1

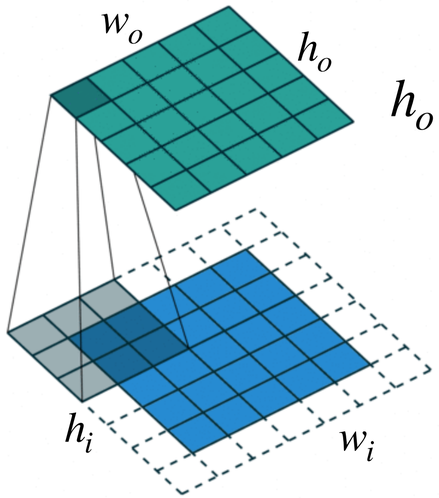


Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
h_i, h_o	Input/Output Height
w_i, w_o	Input/Output Width
k_h, k_w	Kernel Height/Width
g	Groups

Recap: Primitive Operations

Depthwise convolution layer

- **Shape of Tensors:**
  - Input Features $\mathbf{X}$: (n, c_i, h_i, w_i)
  - Output Features $\mathbf{Y}$: (n, c_o, h_o, w_o)
  - Weights $\mathbf{W}$: ~~(c_o, c_i, k_h, k_w)~~ (c, k_h, k_w)
  - Bias $\mathbf{b}$: $(c_o,)$

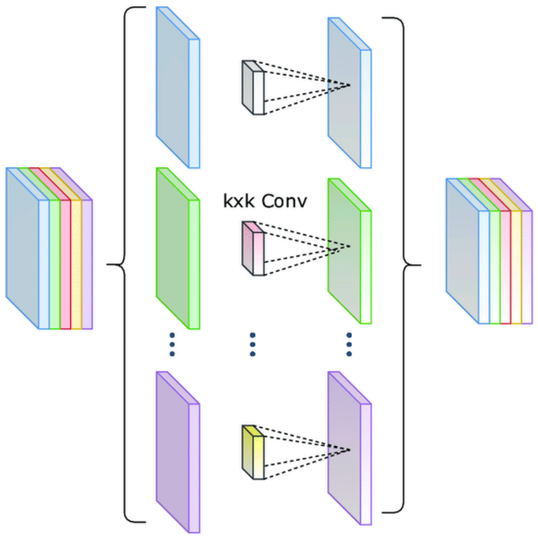


$$h_o = \frac{h_i + 2p - k_h}{s} + 1$$

p is padding

s is stride

Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
h_i, h_o	Input/Output Height
w_i, w_o	Input/Output Width
k_h	Kernel Height
k_w	Kernel Width
g	Groups



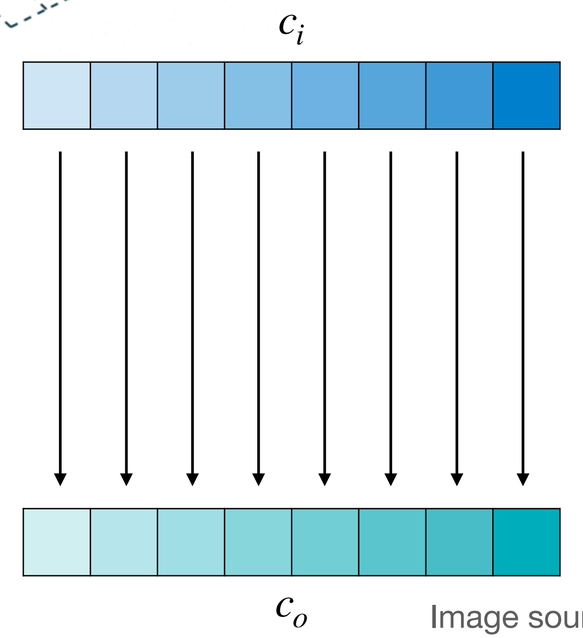


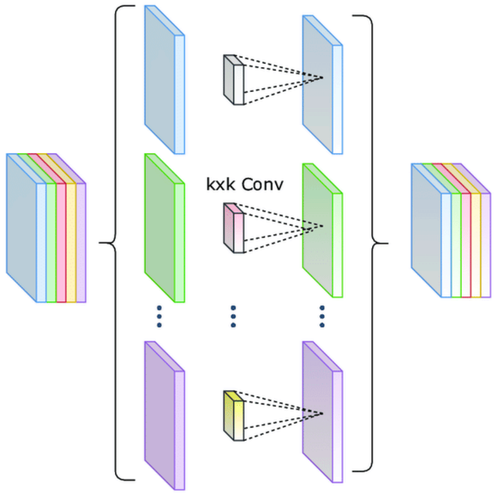
Image source: 1, 2

Recap: Primitive Operations

Depthwise convolution layer

Layer	MACs (batch size n=1)
Linear Layer	$c_o \cdot c_i$
Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w \cdot h_o \cdot w_o$
Grouped Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w \cdot h_o \cdot w_o / g$
Depthwise Convolution	$c_o \cdot k_h \cdot k_w \cdot h_o \cdot w_o$

* bias is ignored



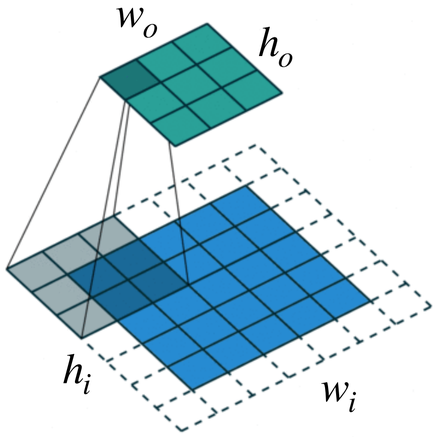
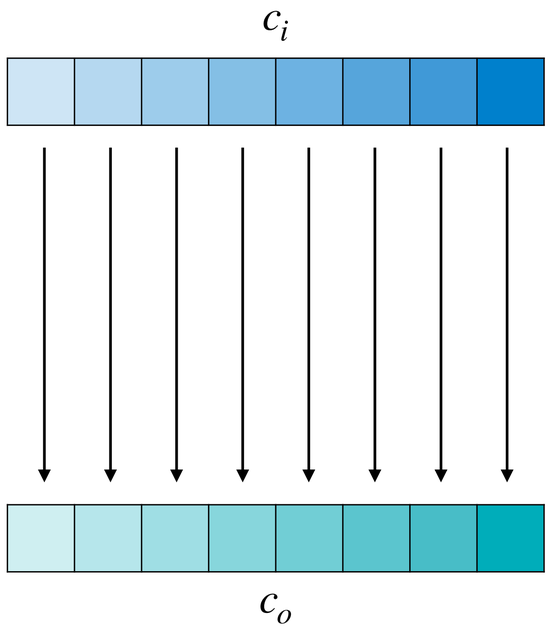


Image source: 1



Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
h_i, h_o	Input/Output Height
w_i, w_o	Input/Output Width
k_h, k_w	Kernel Height/Width
g	Groups

Recap: Primitive Operations

1x1 convolution layer

Layer	MACs (batch size n=1)
Linear Layer	$c_o \cdot c_i$
Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w \cdot h_o \cdot w_o$
Grouped Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w \cdot h_o \cdot w_o / g$
Depthwise Convolution	$c_o \cdot k_h \cdot k_w \cdot h_o \cdot w_o$
1x1 Convolution	$c_o \cdot c_i \cdot h_o \cdot w_o$

* bias is ignored

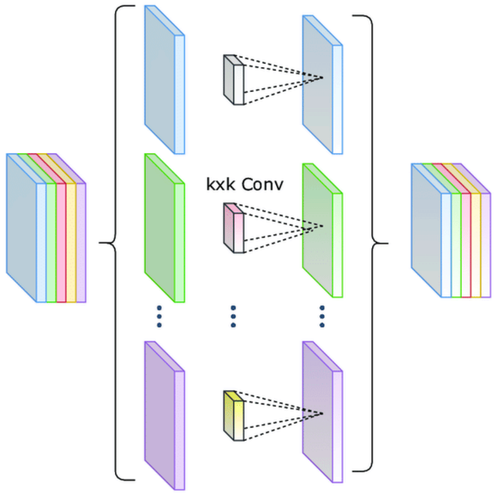
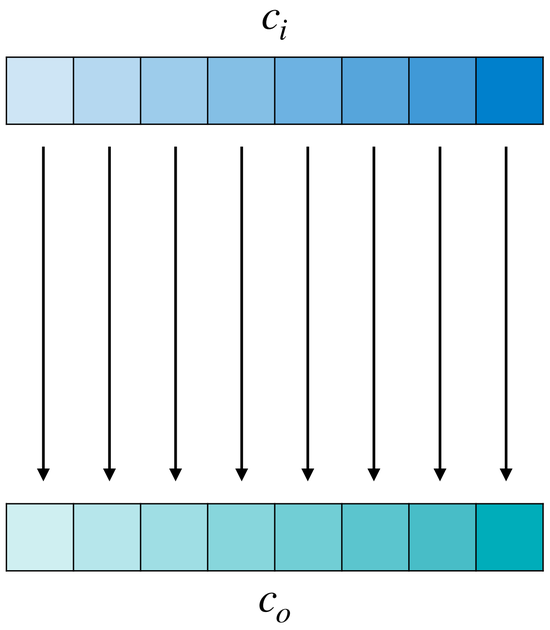


Image source: 1



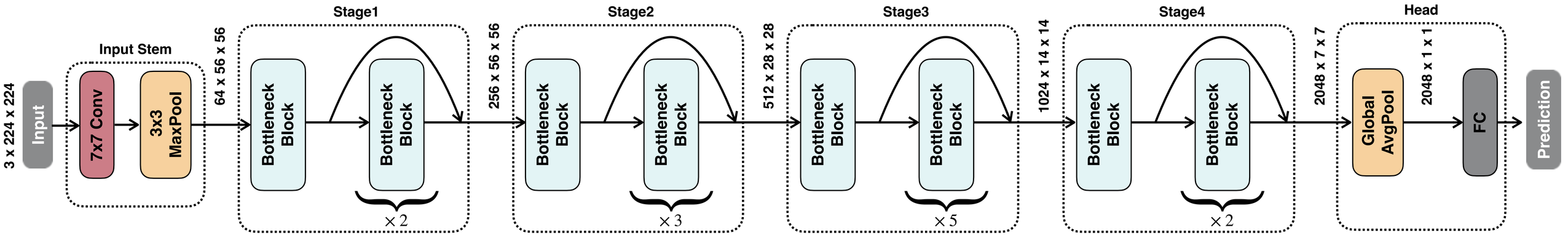
Notations	
n	Batch Size
c_i	Input Channels
c_o	Output Channels
h_i, h_o	Input/Output Height
w_i, w_o	Input/Output Width
k_h, k_w	Kernel Height/Width
g	Groups

Basic Concepts

Stage

- A neural network architecture typically consists of the input stem, the head, and several stages.

- Early stages have larger feature map sizes, so we need to keep the width small to reduce the cost. In contrast, late stages have smaller feature map sizes so that we can increase the width.

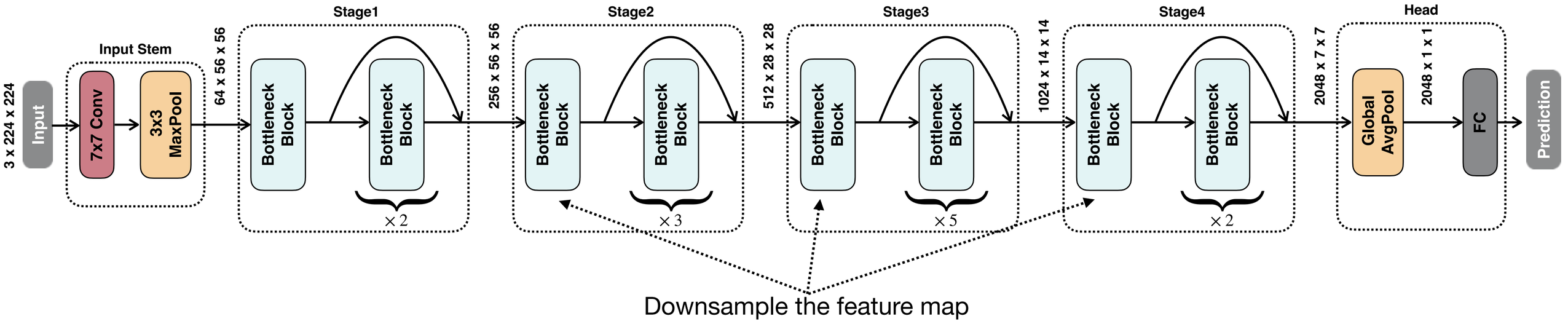


Deep Residual Learning for Image Recognition [He *et al.*, CVPR 2016]

Basic Concepts

Downsample

- Feature map downsampling is usually done at the first block in each stage via stride convolution or pooling.

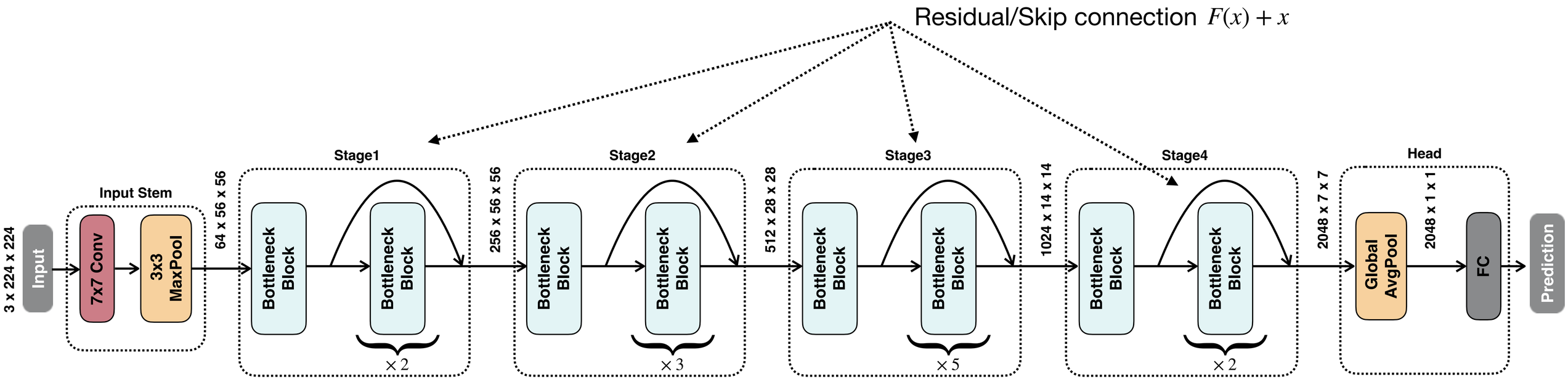


Deep Residual Learning for Image Recognition [He *et al.*, CVPR 2016]

Basic Concepts

Residual/Skip connection

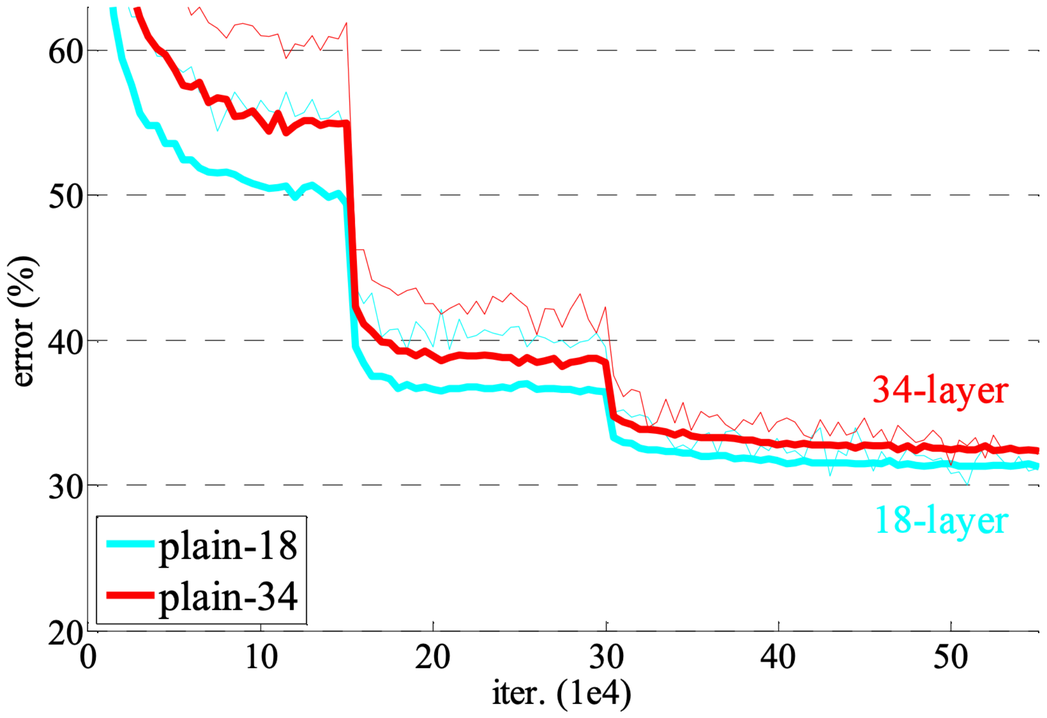
- We can add residual/skip connections for the remaining blocks as their input and output dimensions are the same.

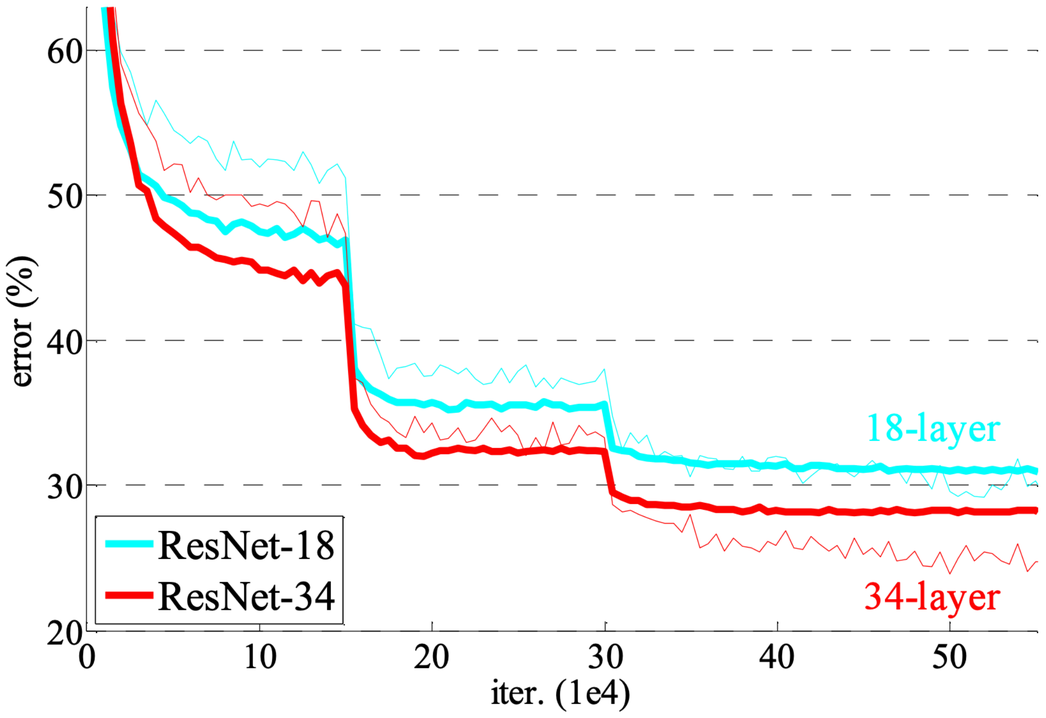


Deep Residual Learning for Image Recognition [He *et al.*, CVPR 2016]

Basic Concepts

Residual/Skip connection





Deep Residual Learning for Image Recognition [He *et al.*, CVPR 2016]

Manually-designed Neural Network

AlexNet: remarkable improvements over previous non-DL methods

Image (3×224×224)

11×11 Conv, channel 96, stride 4, pad 2

3×3 MaxPool, stride 2

5×5 Conv, channel 256, pad 2, groups 2

3×3 MaxPool, stride 2

3×3 Conv, channel 384, pad 1

3×3 Conv, channel 384, pad 1, groups 2

3×3 Conv, channel 256, pad 1, groups 2

3×3 MaxPool, stride 2

Linear, channel 4096

Linear, channel 4096

Linear, channel 1000

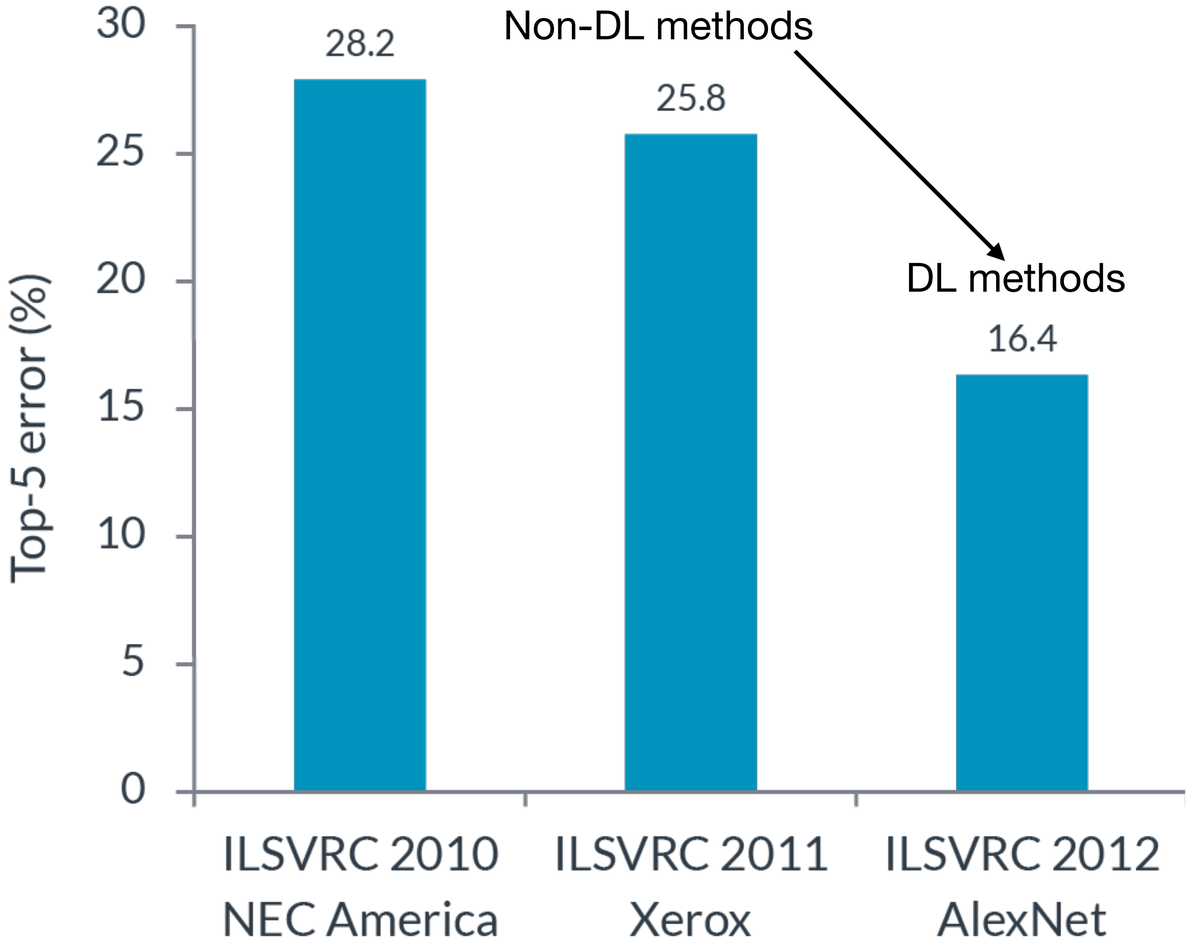
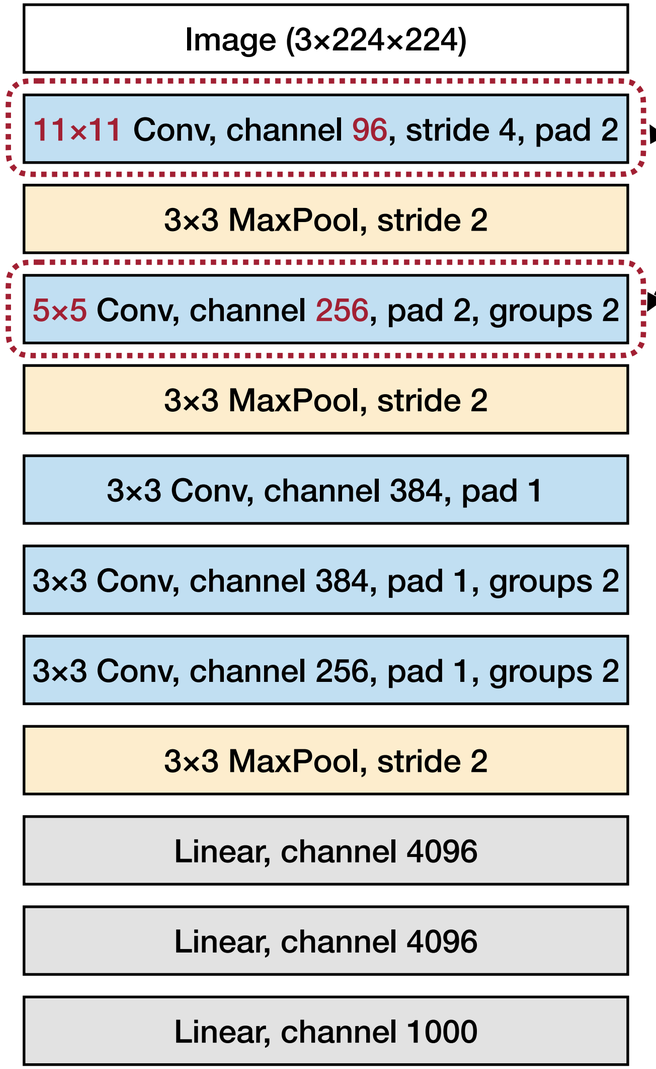


image source: 1

ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky *et al.*, NeurIPS 2012]

Manually-designed Neural Network

AlexNet: large-kernel conv in early stages



Large kernel convolution in early stages

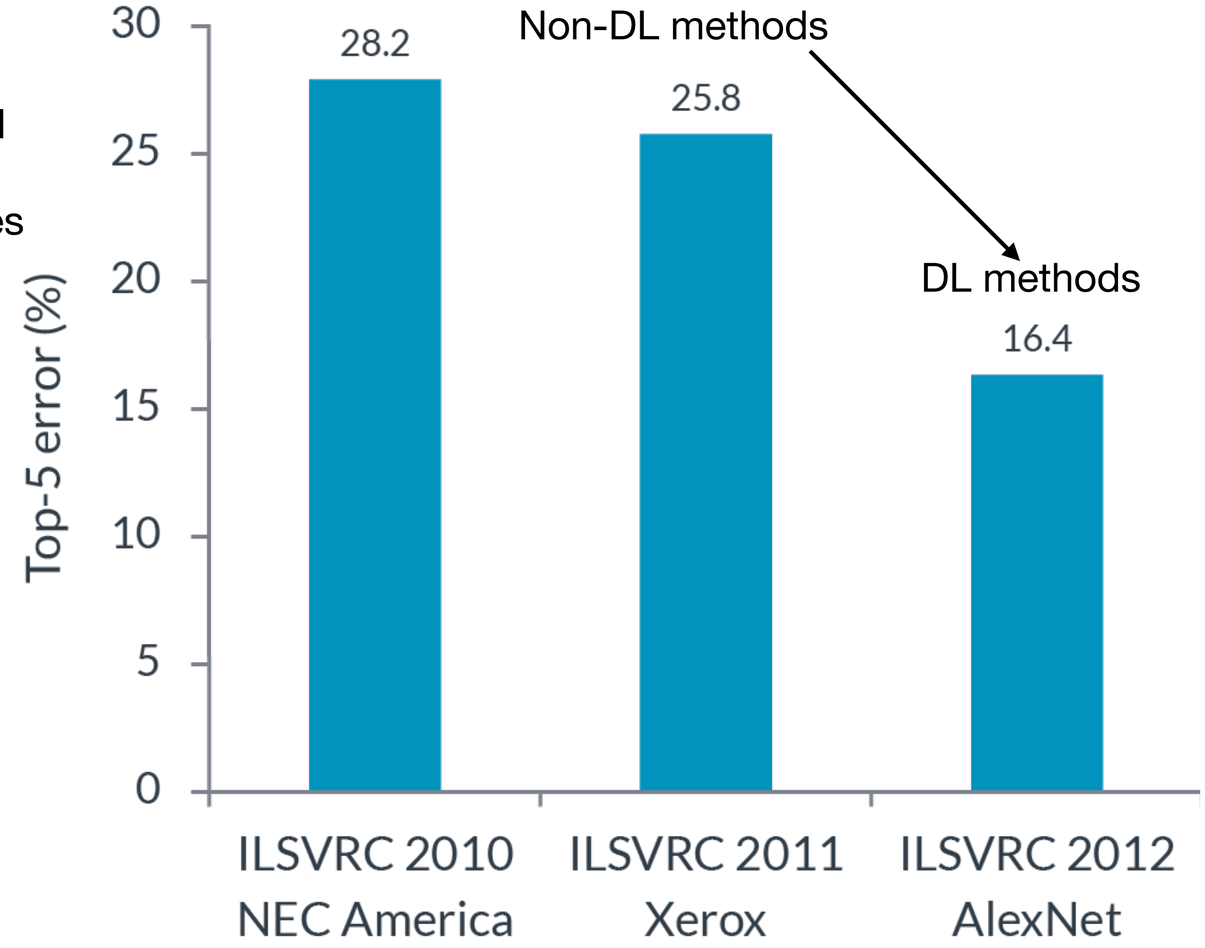


image source: 1

ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky *et al.*, NeurIPS 2012]

Manually-designed Neural Network

VGG: stacking multiple 3x3 convolution layers

Image (3×224×224)
3×3 Conv, channel 64, pad 1
3×3 Conv, channel 64, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 128, pad 1
3×3 Conv, channel 128, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 256, pad 1
3×3 Conv, channel 256, pad 1
3×3 Conv, channel 256, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
2×2 MaxPool, stride 2
Linear, channel 4096
Linear, channel 4096
Linear, channel 1000

- Different from AlexNet, VGG only uses 3x3 convolution. Meanwhile, VGG stacks more layers to maintain a large receptive field.

Very Deep Convolutional Networks for Large-scale Image Recognition [Simonyan *et al.*, ICLR 2015]

Manually-designed Neural Network

VGG: stacking multiple 3x3 convolution layers

Image (3×224×224)
3×3 Conv, channel 64, pad 1
3×3 Conv, channel 64, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 128, pad 1
3×3 Conv, channel 128, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 256, pad 1
3×3 Conv, channel 256, pad 1
3×3 Conv, channel 256, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
2×2 MaxPool, stride 2
Linear, channel 4096
Linear, channel 4096
Linear, channel 1000

- Different from AlexNet, VGG only uses 3x3 convolution. Meanwhile, VGG stacks more layers to maintain a large receptive field.

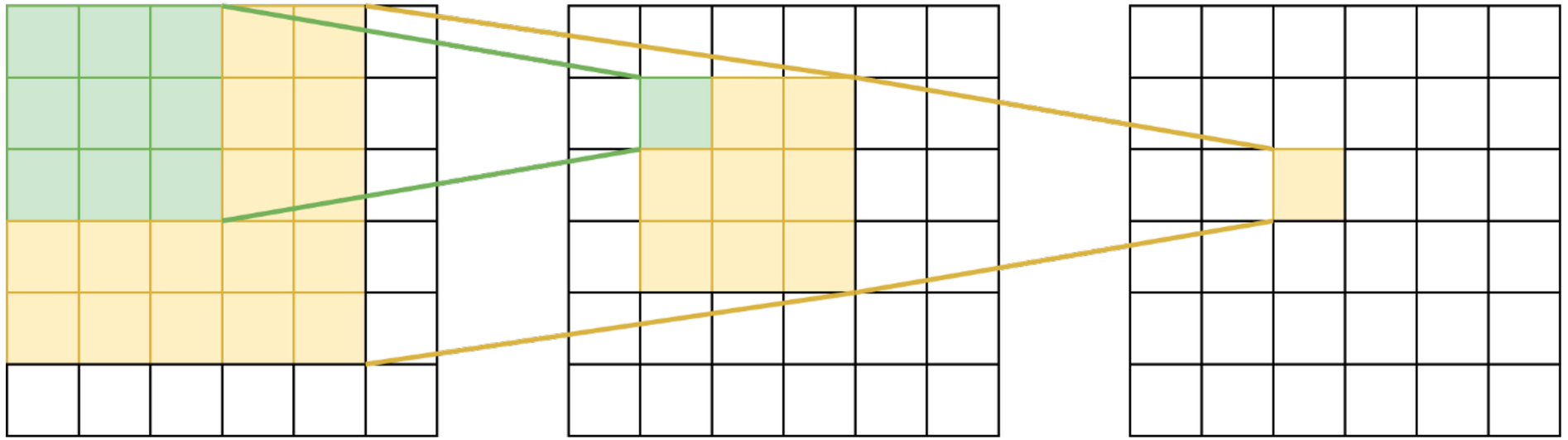


image source: 1

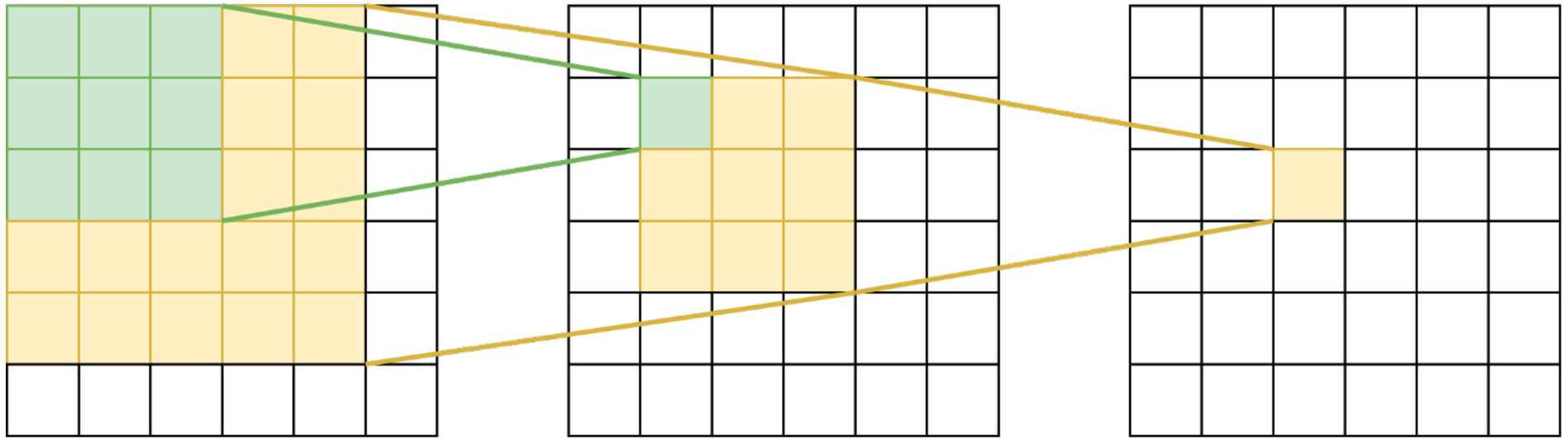
Very Deep Convolutional Networks for Large-scale Image Recognition [Simonyan *et al.*, ICLR 2015]

Manually-designed Neural Network

VGG: stacking multiple 3x3 convolution layers

Image (3×224×224)
3×3 Conv, channel 64, pad 1
3×3 Conv, channel 64, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 128, pad 1
3×3 Conv, channel 128, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 256, pad 1
3×3 Conv, channel 256, pad 1
3×3 Conv, channel 256, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
2×2 MaxPool, stride 2
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
3×3 Conv, channel 512, pad 1
2×2 MaxPool, stride 2
Linear, channel 4096
Linear, channel 4096
Linear, channel 1000

- Different from AlexNet, VGG only uses 3x3 convolution. Meanwhile, VGG stacks more layers to maintain a large receptive field.



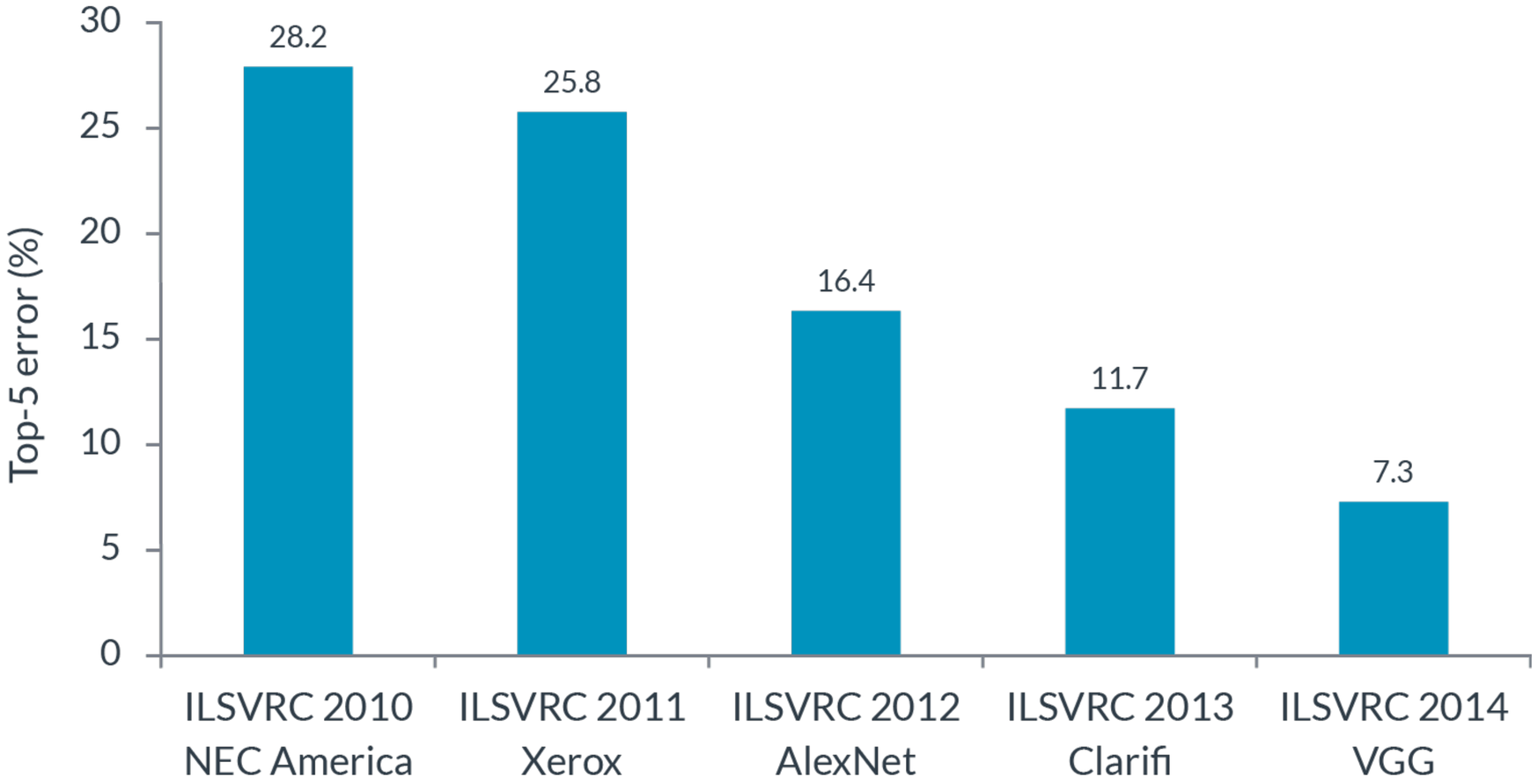
- The computational cost of two 3x3 convolutions is smaller than one 5x5 convolution: 3x3 + 3x3 = 18 < 5x5

image source: 1

Very Deep Convolutional Networks for Large-scale Image Recognition [Simonyan *et al.*, ICLR 2015]

Manually-designed Neural Network

VGG: performance and cost breakdown



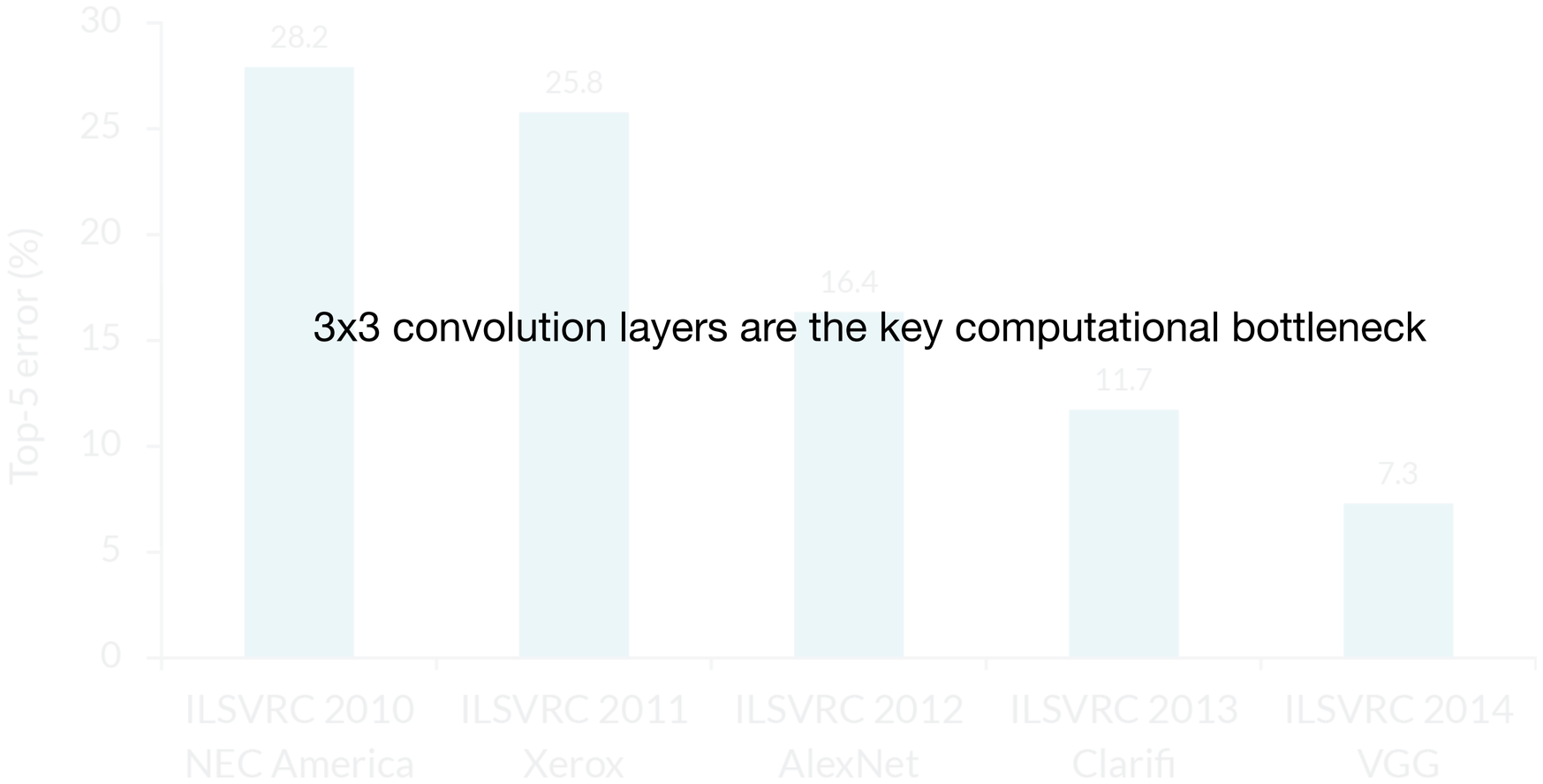
Layer	Weights	FLOP
conv1_1	2K	0.2B
conv1_2	37K	3.7B
conv2_1	74K	1.8B
conv2_2	148K	3.7B
conv3_1	295K	1.8B
conv3_2	590K	3.7B
conv3_3	590K	3.7B
conv4_1	1M	1.8B
conv4_2	2M	3.7B
conv4_3	2M	3.7B
conv5_1	2M	925M
conv5_2	2M	925M
conv5_3	2M	925M
fc6	103M	206M
fc7	17M	34M
fc8	4M	8M
total	138M	30.9B

image source: link

Very Deep Convolutional Networks for Large-scale Image Recognition [Simonyan *et al.*, ICLR 2015]

Manually-designed Neural Network

VGG: performance and cost breakdown



3x3 convolution layers are the key computational bottleneck

Layer	Weights	FLOP
conv1_1	2K	0.2B
conv1_2	37K	3.7B
conv2_1	74K	1.8B
conv2_2	148K	3.7B
conv3_1	295K	1.8B
conv3_2	590K	3.7B
conv3_3	590K	3.7B
conv4_1	1M	1.8B
conv4_2	2M	3.7B
conv4_3	2M	3.7B
conv5_1	2M	925M
conv5_2	2M	925M
conv5_3	2M	925M
fc6	103M	206M
fc7	17M	34M
fc8	4M	8M
total	138M	30.9B

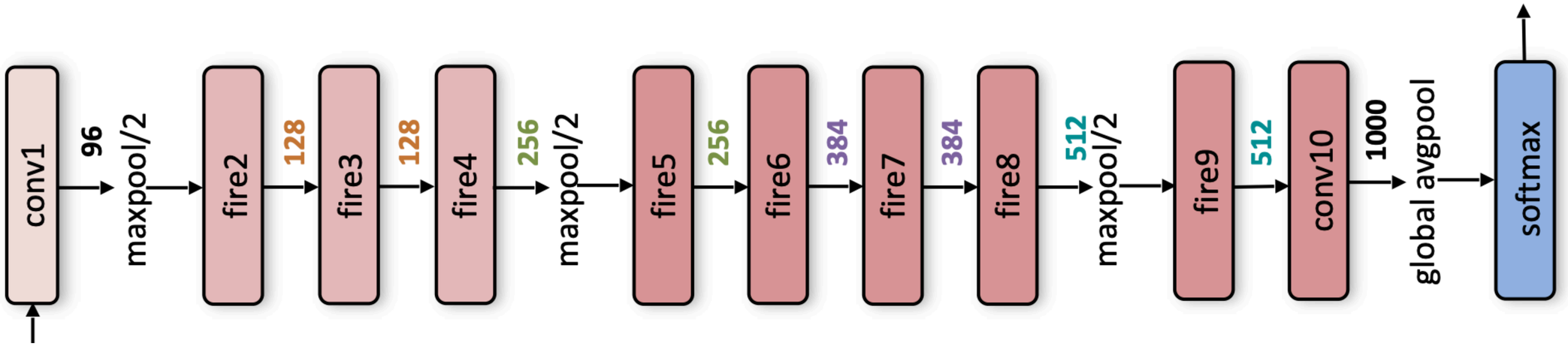
image source: link

Very Deep Convolutional Networks for Large-scale Image Recognition [Simonyan *et al.*, ICLR 2015]

Manually-designed Neural Network

SqueezeNet

- Replace 3x3 convolution with fire modules.
- Use global average pooling in the head to reduce the cost of the head.

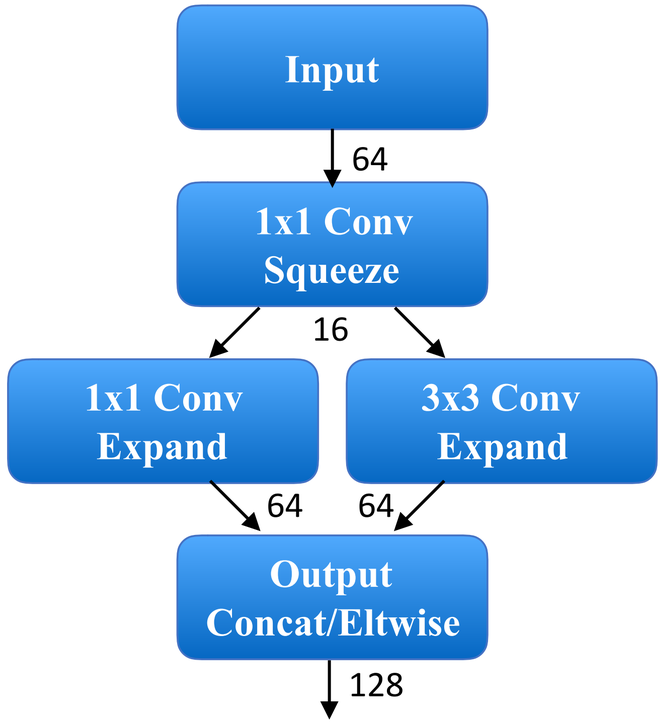


SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size [Iandola *et al.*, arXiv 2016]

Manually-designed Neural Network

Fire module: reduce the cost by using 1x1 convolution

- Reduce the cost of 3x3 convolution layers
  - Replace some 3x3 filters with 1x1 filters.
  - Decrease the number of input channels to 3x3 filters.



SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size [Iandola *et al.*, arXiv 2016]

Manually-designed Neural Network

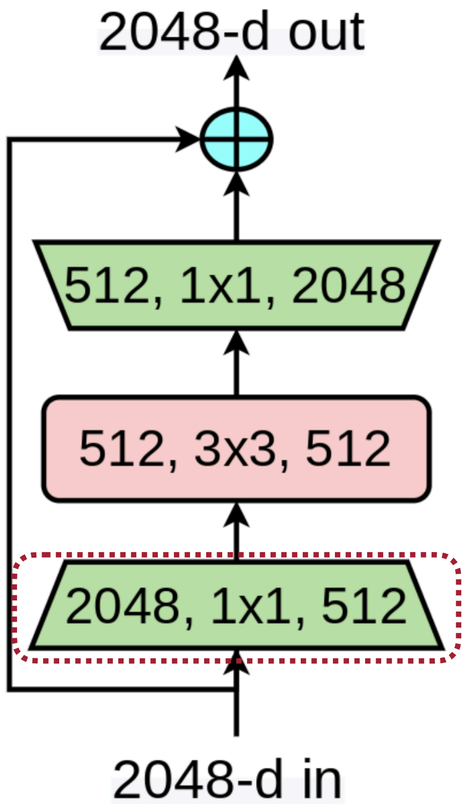
SqueezeNet: AlexNet-level accuracy with 50x-510x smaller model size

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size [Iandola *et al.*, arXiv 2016]

Manually-designed Neural Network

ResNet50: bottleneck block



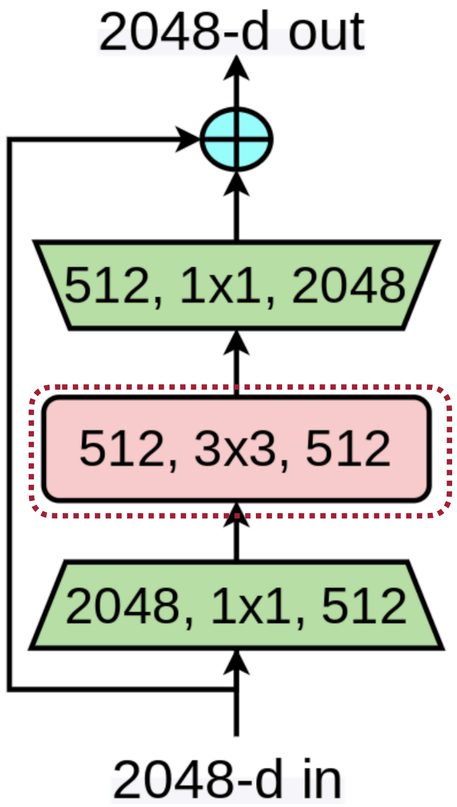
Reduce the number of channels by 4x
via 1x1 convolution

image source: link

Deep Residual Learning for Image Recognition [He *et al.*, CVPR 2016]

Manually-designed Neural Network

ResNet50: bottleneck block



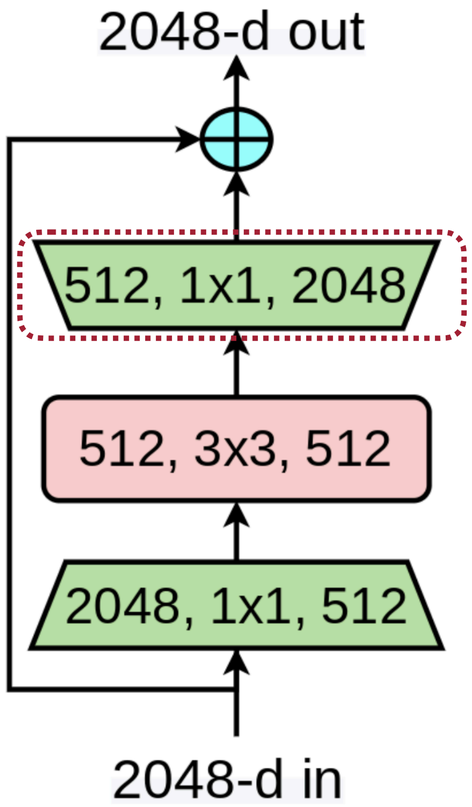
Feed reduced feature map to 3x3 convolution

image source: link

Deep Residual Learning for Image Recognition [He *et al.*, CVPR 2016]

Manually-designed Neural Network

ResNet50: bottleneck block



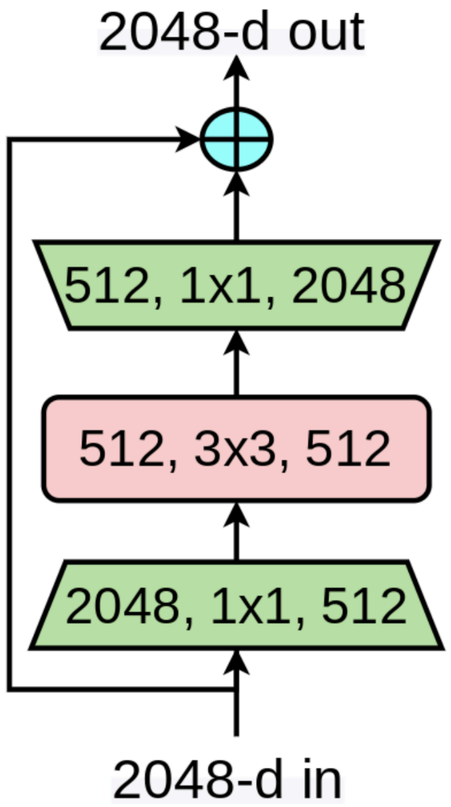
Expand the number of channels via 1x1 convolution

image source: link

Deep Residual Learning for Image Recognition [He *et al.*, CVPR 2016]

Manually-designed Neural Network

ResNet50: bottleneck block



#MACs

2048 x 512 x H x W x 1

512 x 512 x H x W x 9

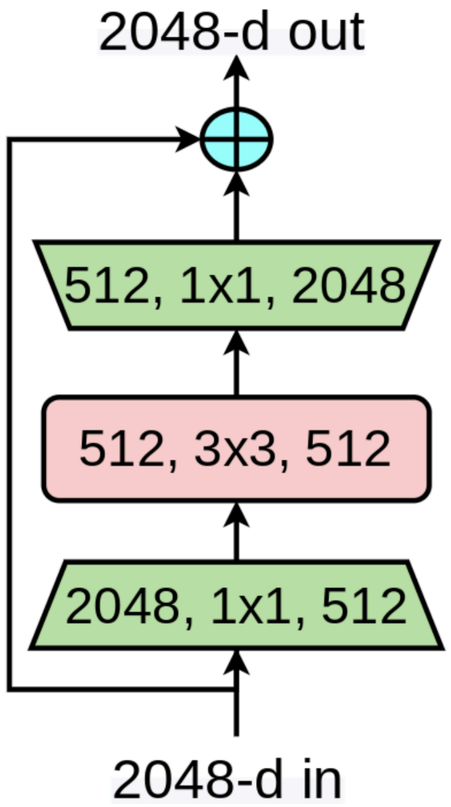
2048 x 512 x H x W x 1

image source: link

Deep Residual Learning for Image Recognition [He *et al.*, CVPR 2016]

Manually-designed Neural Network

ResNet50: bottleneck block



#MACs

2048 x 512 x H x W x 1

512 x 512 x H x W x 9

2048 x 512 x H x W x 1

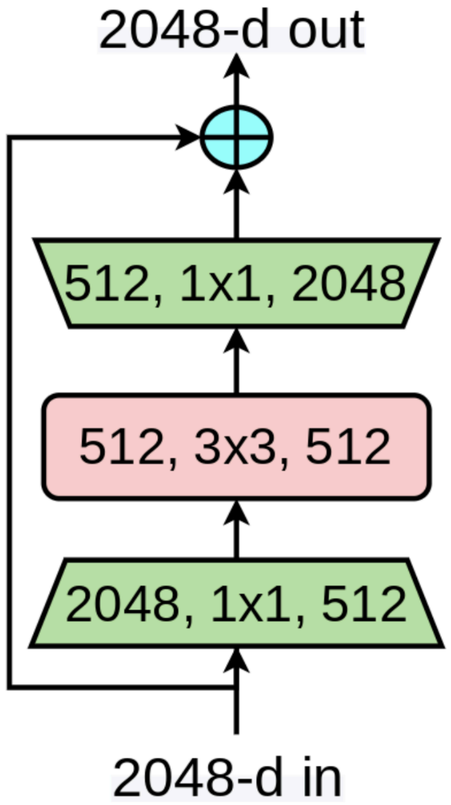
512 x 512 x H x W x 17

image source: link

Deep Residual Learning for Image Recognition [He *et al.*, CVPR 2016]

Manually-designed Neural Network

ResNet50: bottleneck block



#MACs

2048 x 512 x H x W x 1

512 x 512 x H x W x 9

2048 x 512 x H x W x 1

2048 x 2048 x H x W x 9

= 512 x 512 x H x W x 144

8.5x reduction

512 x 512 x H x W x 17

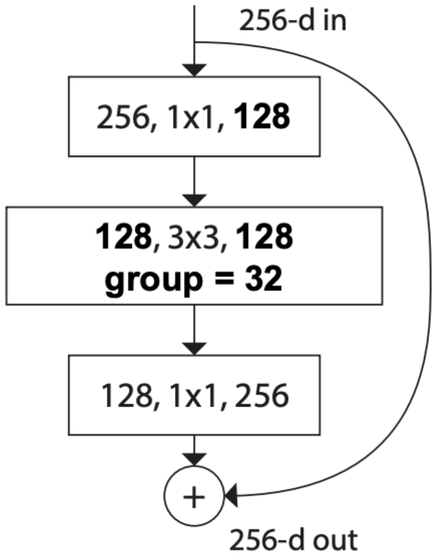
image source: link

Deep Residual Learning for Image Recognition [He *et al.*, CVPR 2016]

Manually-designed Neural Network

ResNeXt: grouped convolution

- Replace 3x3 convolution with 3x3 grouped convolution.

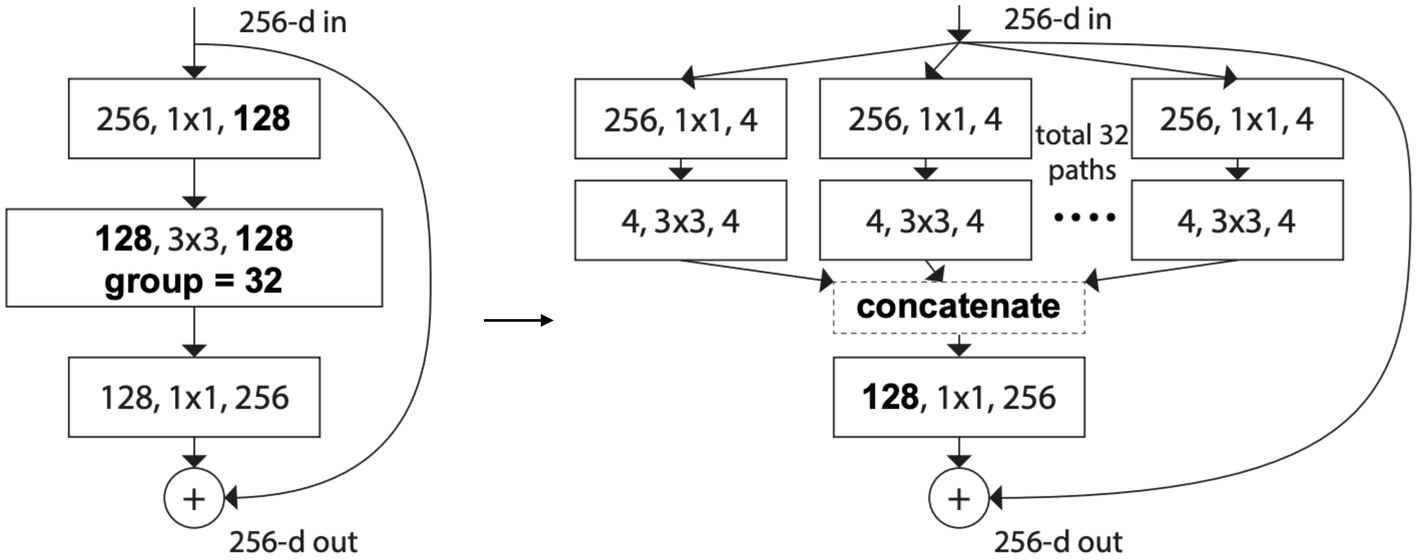


Aggregated Residual Transformations for Deep Neural Networks [Xie *et al.*, CVPR 2017]

Manually-designed Neural Network

ResNeXt: grouped convolution

- Replace 3x3 convolution with 3x3 grouped convolution.
- Equivalent to a multi-path block.

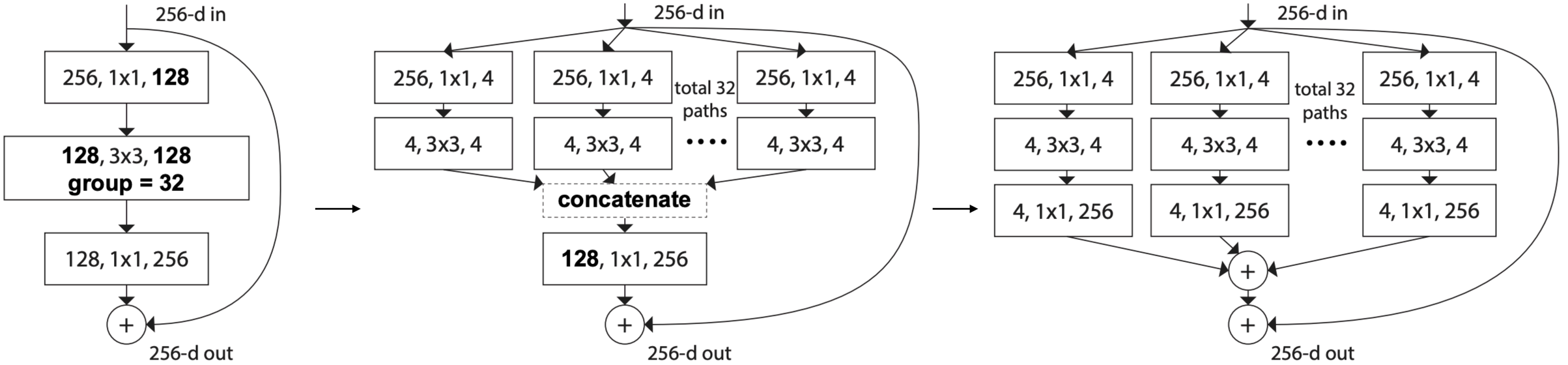


Aggregated Residual Transformations for Deep Neural Networks [Xie *et al.*, CVPR 2017]

Manually-designed Neural Network

ResNeXt: grouped convolution

- Replace 3x3 convolution with 3x3 grouped convolution.
- Equivalent to a multi-path block.



Aggregated Residual Transformations for Deep Neural Networks [Xie *et al.*, CVPR 2017]

Manually-designed Neural Network

MobileNet: depthwise-separable block

- Depthwise convolution is an extreme case of group convolution where the group number equals the number of input channels.

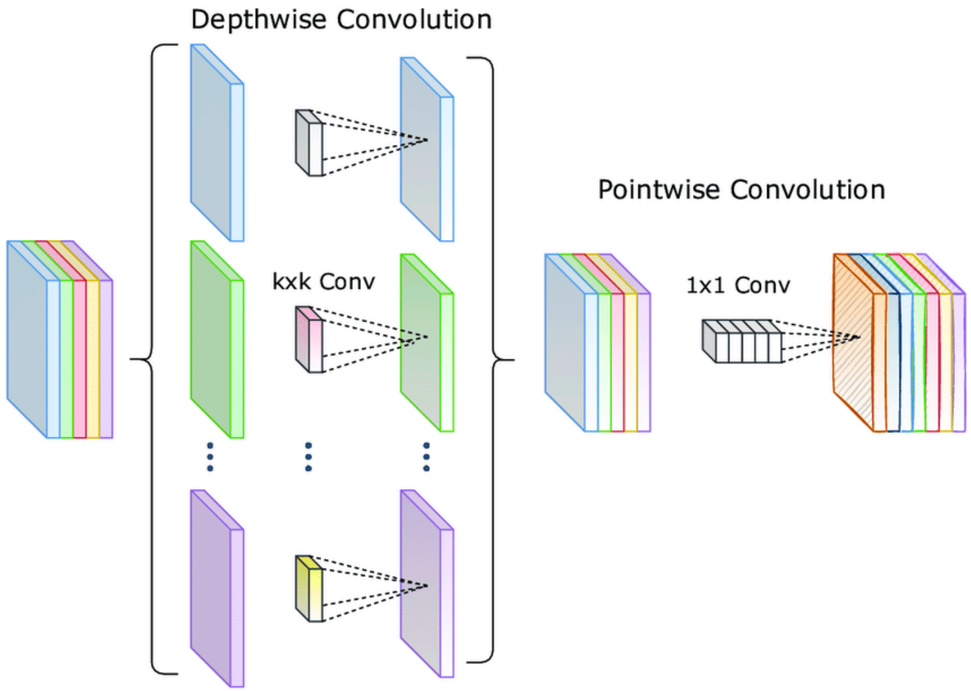


image source: link

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Howard *et al.*, arXiv 2017]

Manually-designed Neural Network

MobileNet: depthwise-separable block

- Depthwise convolution is an extreme case of group convolution where the group number equals the number of input channels.
- Use depthwise convolution to capture spatial information.

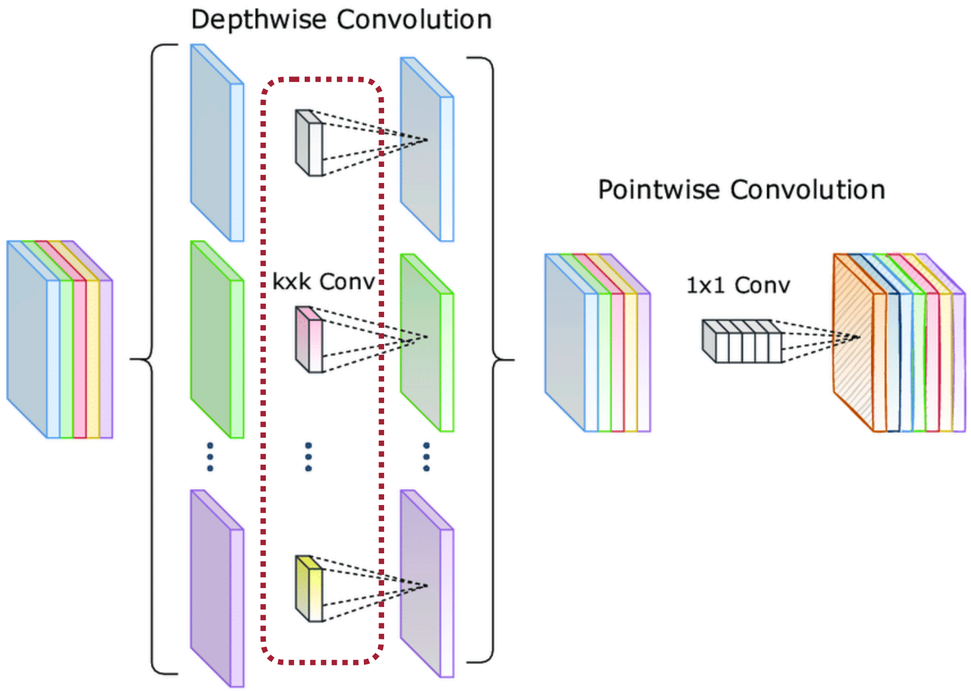


image source: link

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Howard *et al.*, arXiv 2017]

Manually-designed Neural Network

MobileNet: depthwise-separable block

- Depthwise convolution is an extreme case of group convolution where the group number equals the number of input channels.
- Use depthwise convolution to capture spatial information.
- Use 1x1 convolution to fuse/exchange information across different channels.

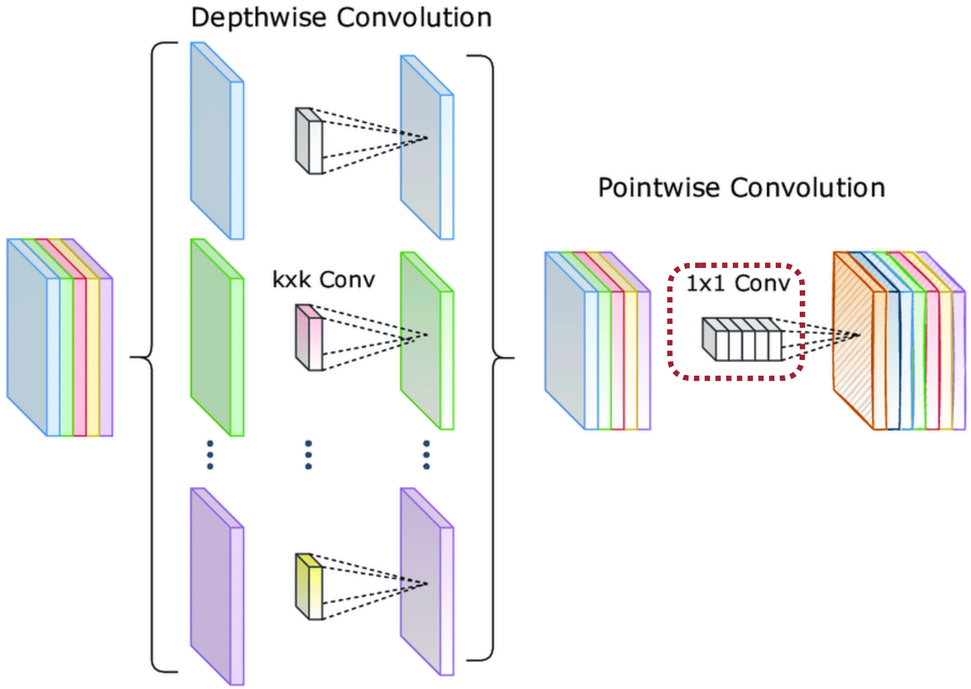


image source: link

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Howard *et al.*, arXiv 2017]

Manually-designed Neural Network

MobileNetV2: inverted bottleneck block

- Depthwise convolution has a much lower capacity compared to normal convolution.
  - Increase the depthwise convolution's input and output channels to improve its capacity.
  - Depthwise convolution’s cost only grows linearly. Therefore, the cost is still affordable.

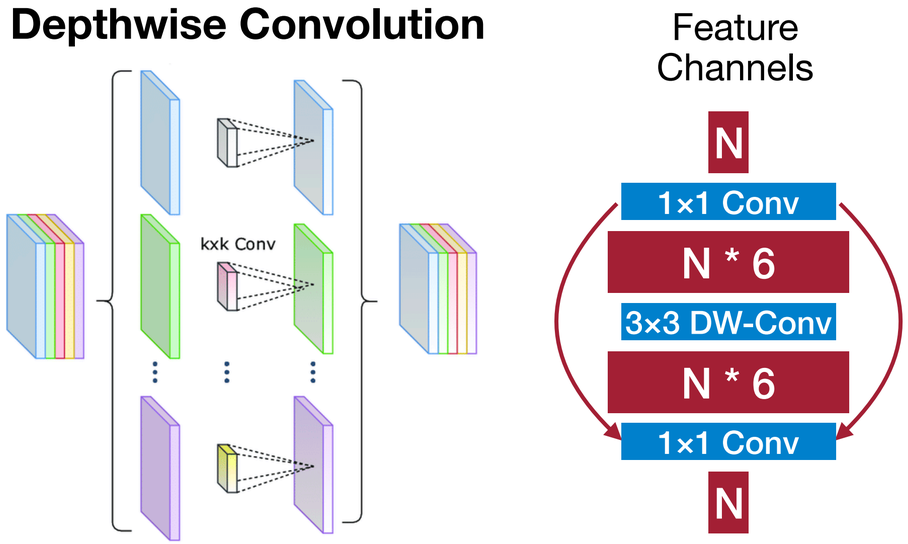


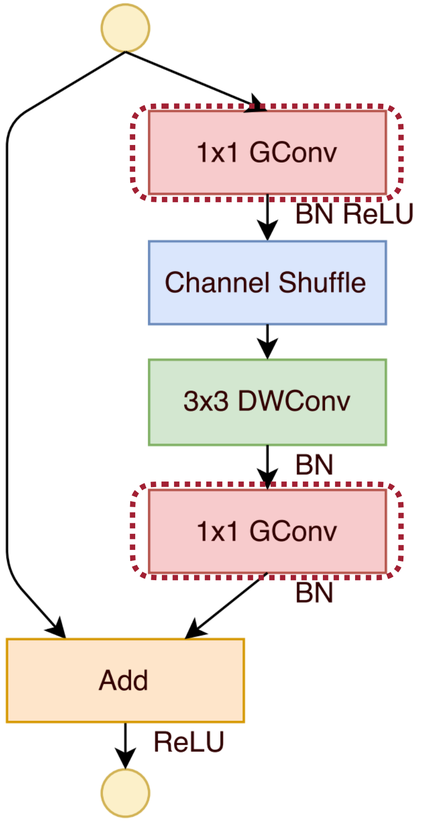
Image source: 1

MobileNetV2: Inverted Residuals and Linear Bottlenecks [Sandler *et al.*, CVPR 2018]

Manually-designed Neural Network

ShuffleNet: 1x1 group convolution

- Further reduce the cost by replacing 1x1 convolution with 1x1 group convolution.

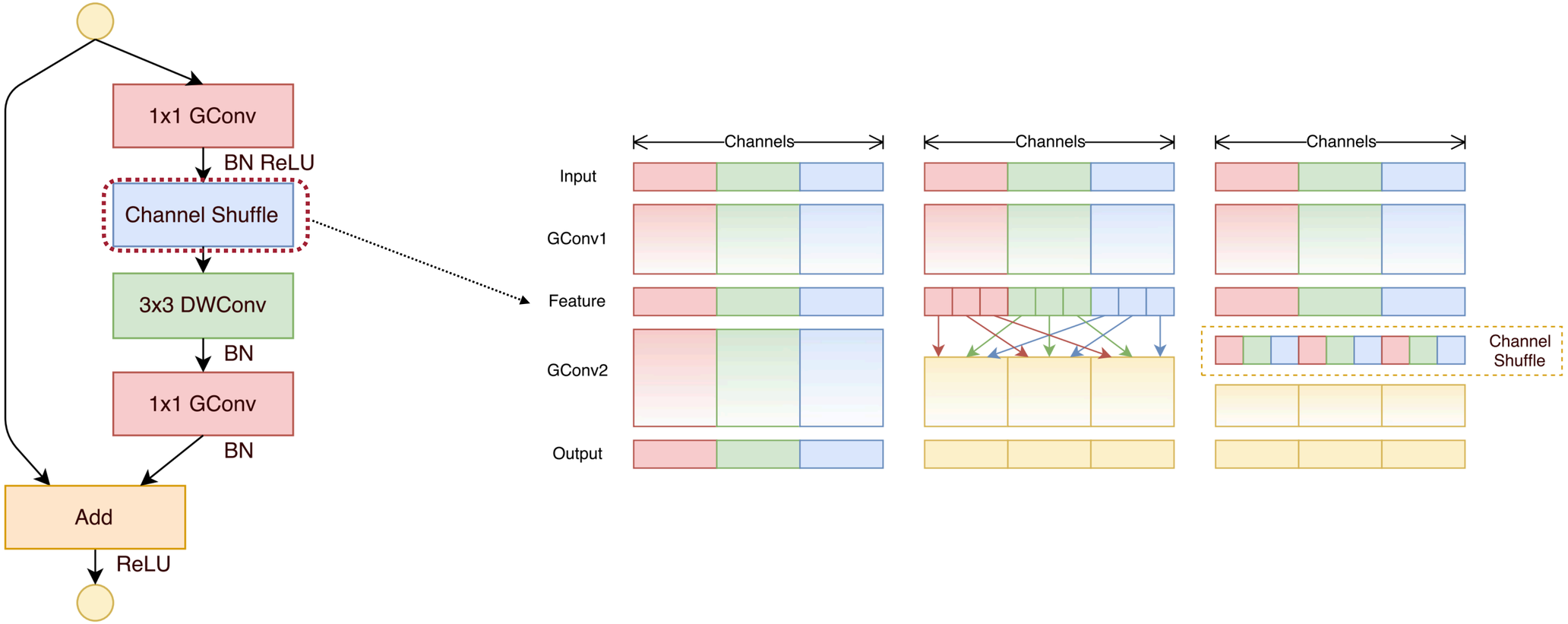


ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices [Zhang *et al.*, CVPR 2018]

Manually-designed Neural Network

ShuffleNet: 1x1 group convolution & channel shuffle

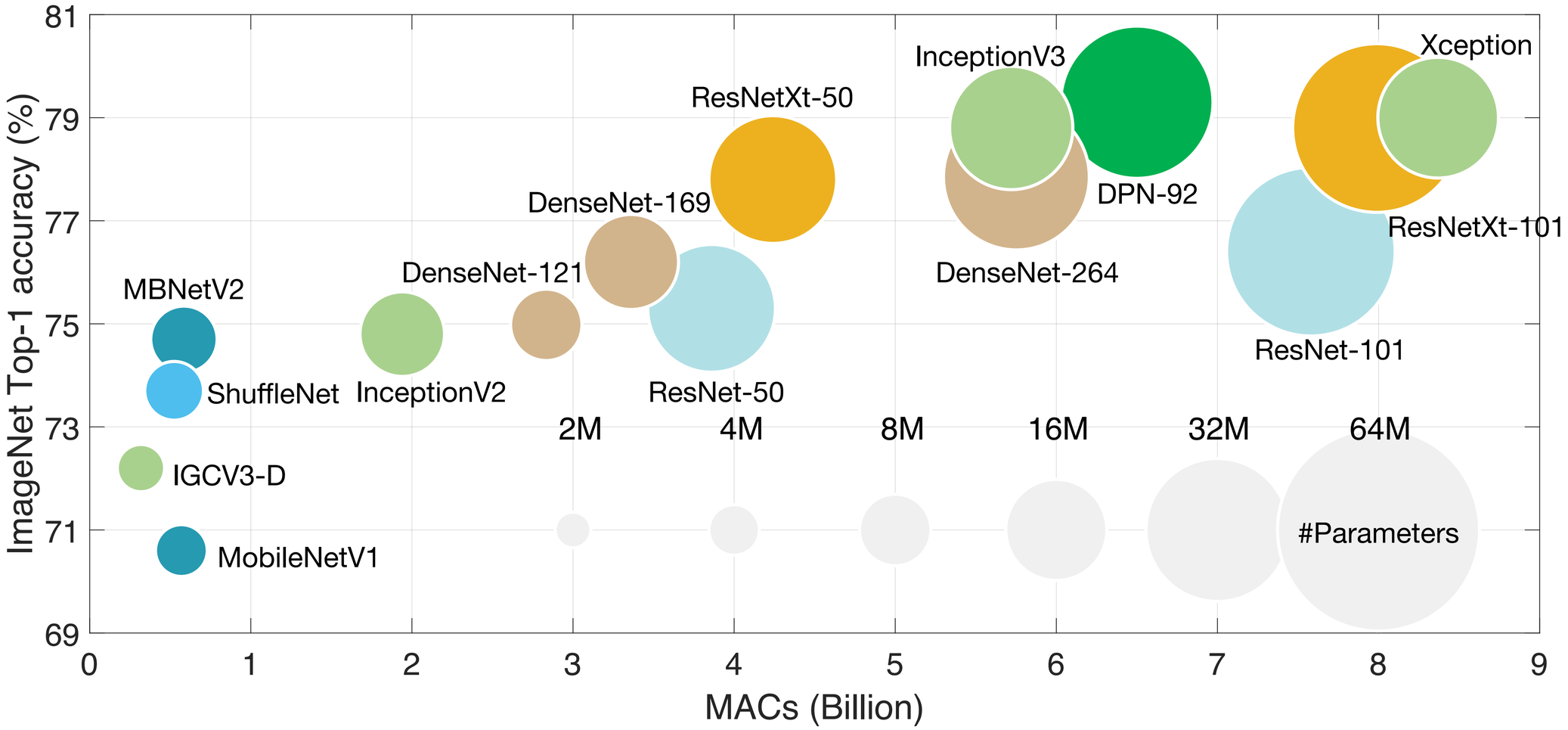
- Further reduce the cost by replacing 1x1 convolution with 1x1 group convolution.
- Exchange information across different groups via channel shuffle.



ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices [Zhang *et al.*, CVPR 2018]

Manually-designed Neural Network

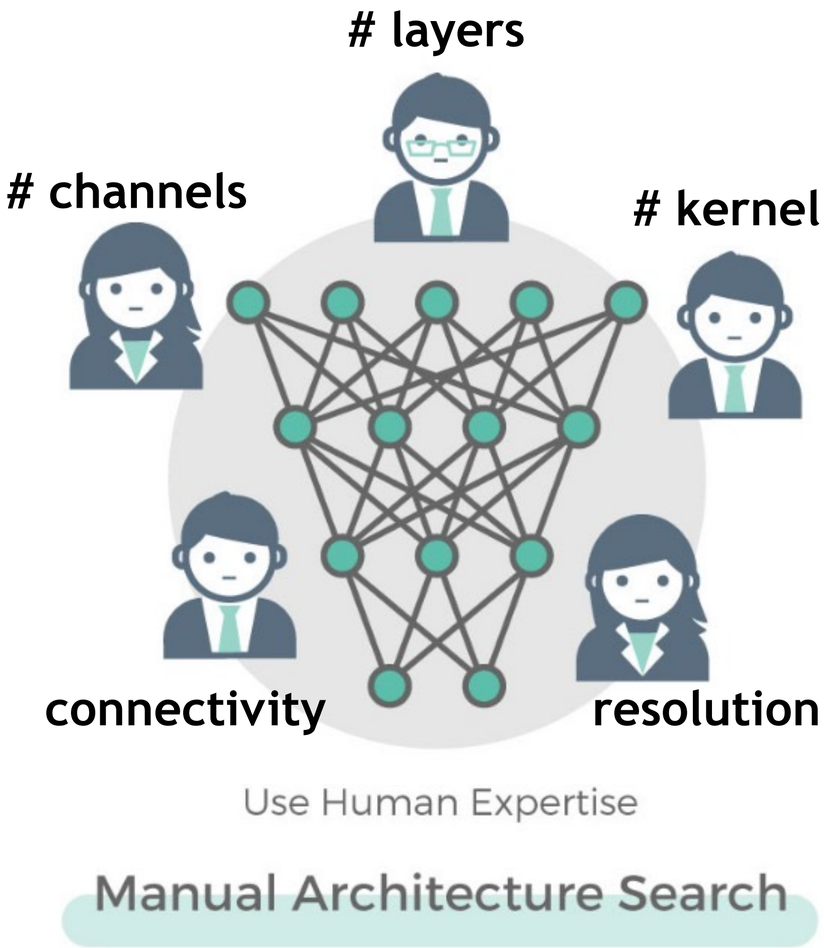
Accuracy-efficiency trade-off on ImageNet



Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey [Deng *et al.*, IEEE 2020]

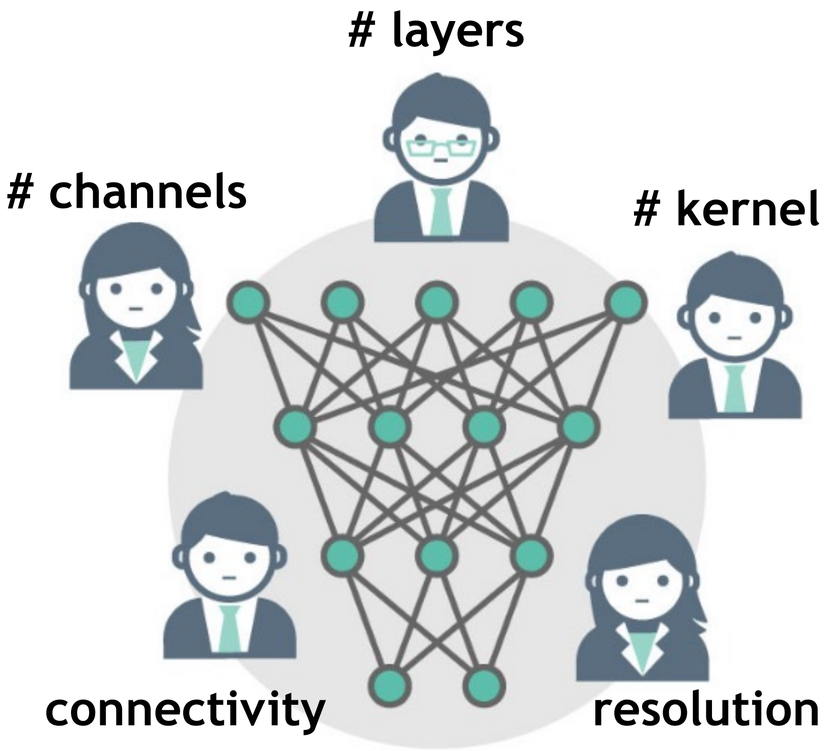
From Manual Design to Automatic Design

Huge design space, manual design is unscalable



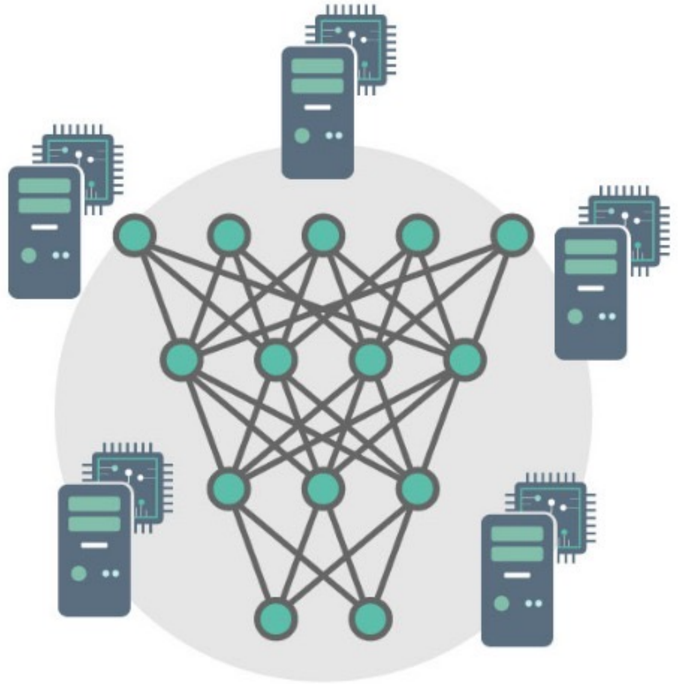
From Manual Design to Automatic Design

Huge design space, manual design is unscalable



Use Human Expertise

Manual Architecture Search

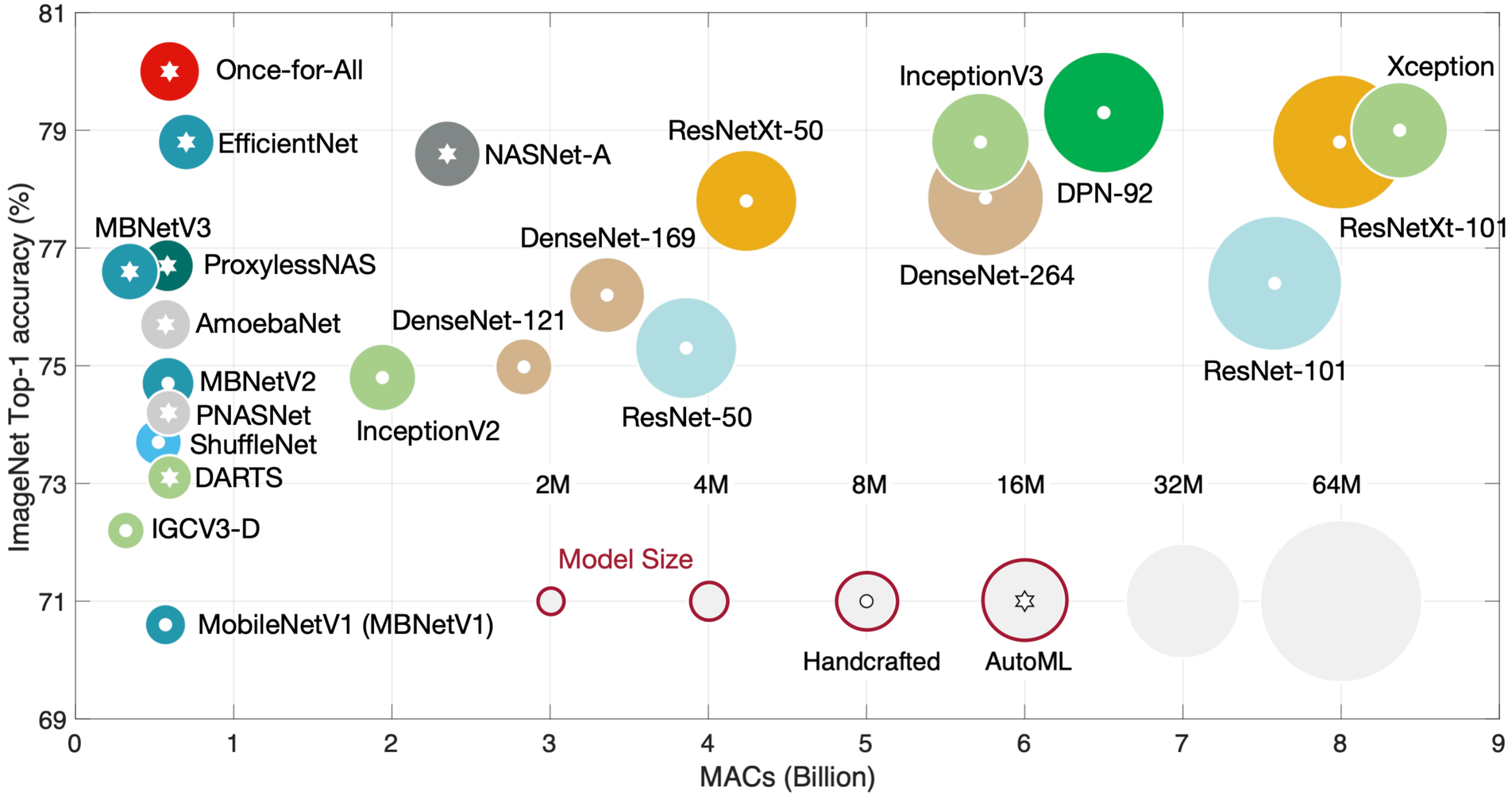


Use Machine Learning

Automatic Architecture Search

From Manual Design to Automatic Design

Huge design space, manual design is unscalable



Once-for-All: Train One Network and Specialize it for Efficient Deployment [Cai *et al.*, ICLR 2020]

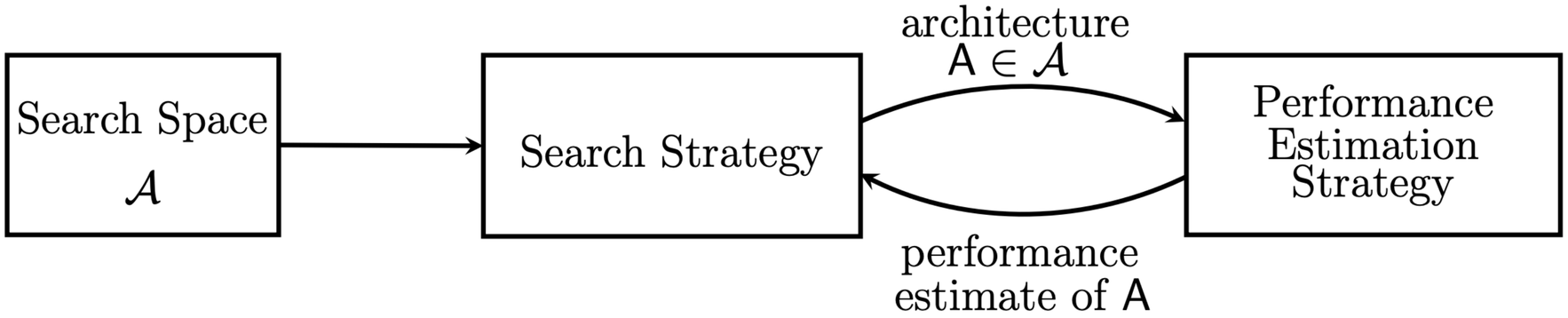
What is NAS?

Framework of Neural Architecture Search (NAS)

Illustration of NAS

Components and goal

- The goal of NAS is to find the best neural network architecture in the search space, maximizing the objective of interest (e.g., accuracy, efficiency, etc).



Neural Architecture Search: A Survey [Elskan *et al.*, JMLR 2019]

Illustration of NAS

Search space

- Search space is a set of candidate neural network architectures.

Neural Architecture Search: A Survey [Elskan *et al.*, JMLR 2019]

Illustration of NAS

Search space: example

- Search space is a set of candidate neural network architectures.

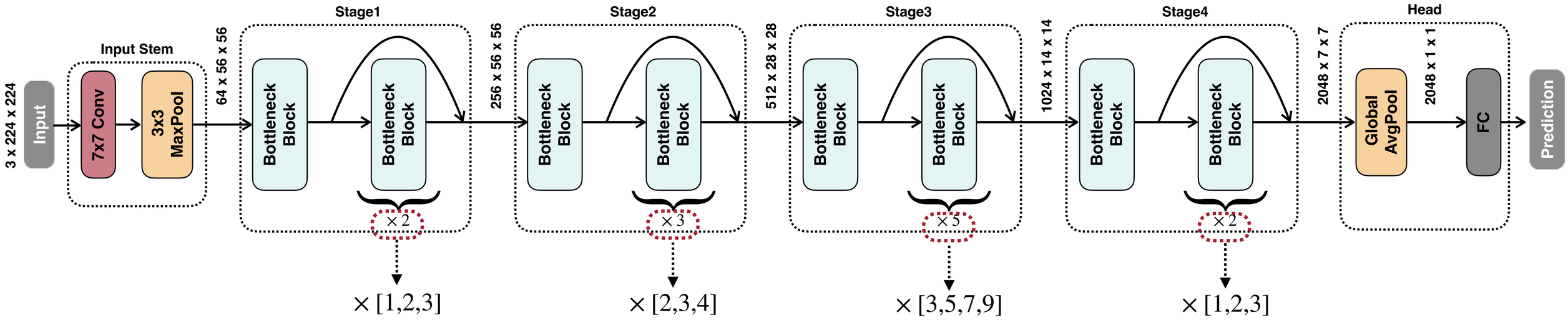
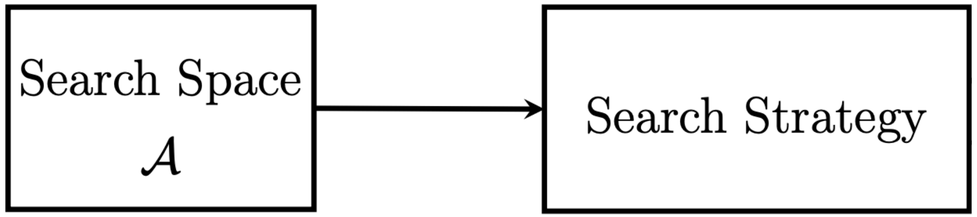


Illustration of NAS

Search strategy

- Search space is a set of candidate neural network architectures.
- Search strategy defines how to explore the search space.



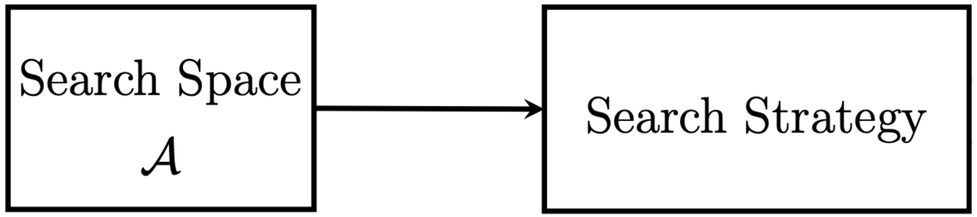
Neural Architecture Search: A Survey [Elskan *et al.*, JMLR 2019]

Illustration of NAS

Search strategy: example

- Search space is a set of candidate neural network architectures.
- Search strategy defines how to explore the search space.

Random search: randomly sample an architecture from the search space

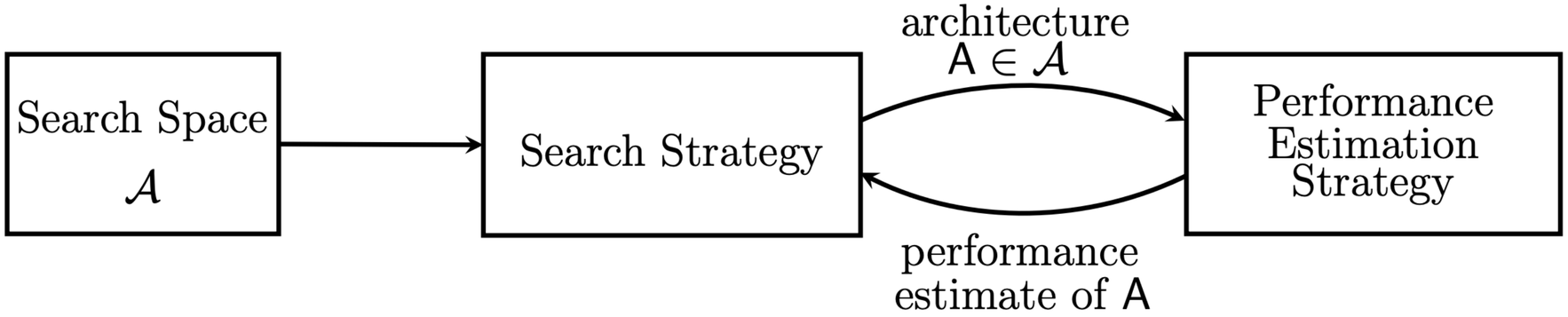


Neural Architecture Search: A Survey [Elskan *et al.*, JMLR 2019]

Illustration of NAS

Performance estimation

- Search space is a set of candidate neural network architectures.
- Search strategy defines how to explore the search space.
- Performance estimation strategy defines how to estimate/predict the performance of a given neural network architecture in the design space.

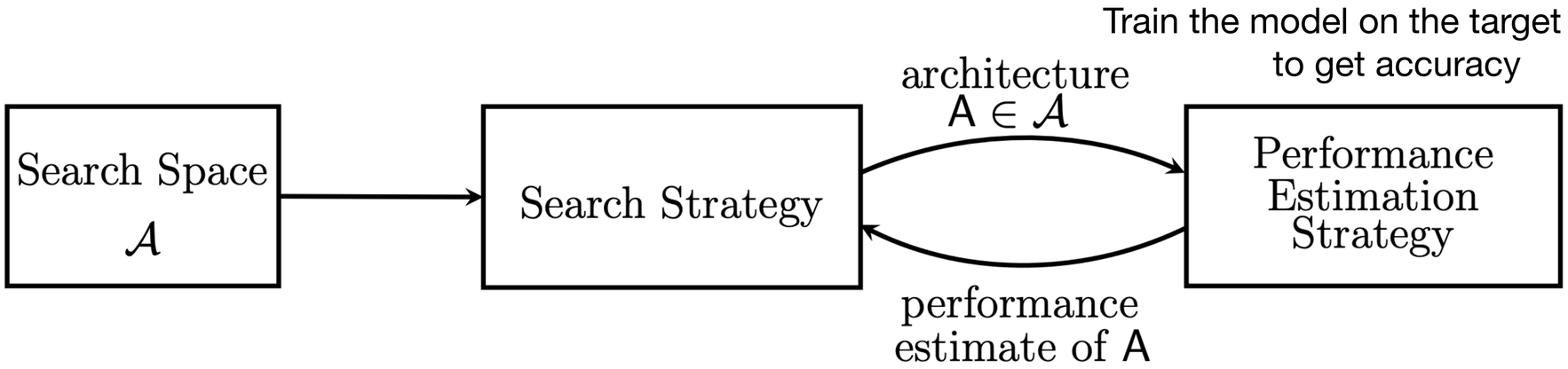


Neural Architecture Search: A Survey [Elskan *et al.*, JMLR 2019]

Illustration of NAS

Performance estimation: example

- Search space is a set of candidate neural network architectures.
- Search strategy defines how to explore the search space.
- Performance estimation strategy defines how to estimate/predict the performance of a given neural network architecture in the design space.



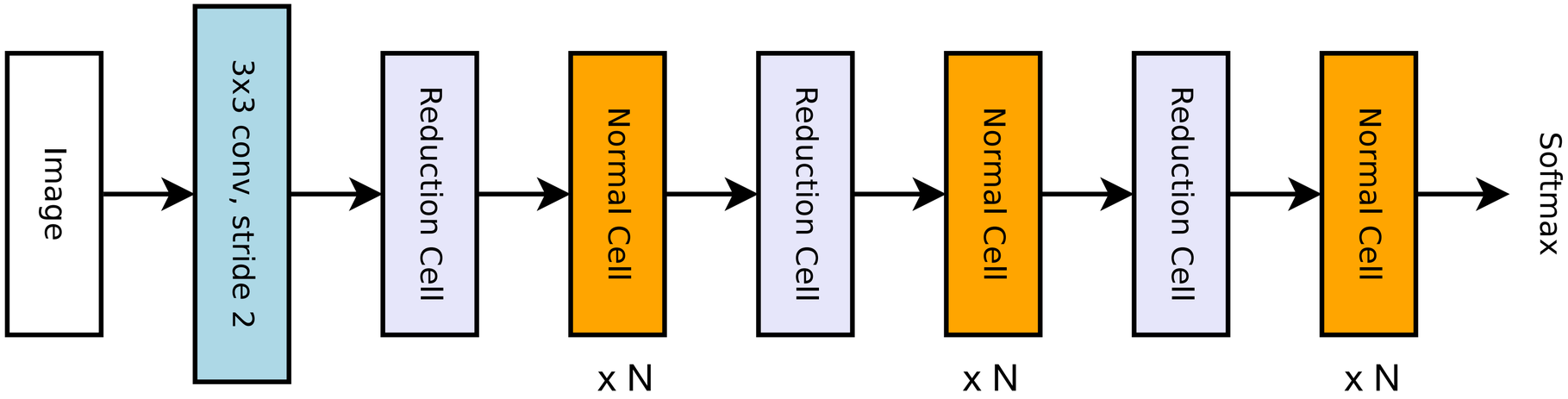
Neural Architecture Search: A Survey [Elskan *et al.*, JMLR 2019]

NAS Search Space

1. **Cell-level Search Space**
2. **Network-level Search Space**

Cell-level Search Space

Classifiers have normal (stride = 1) and reduction (stride > 1) cells



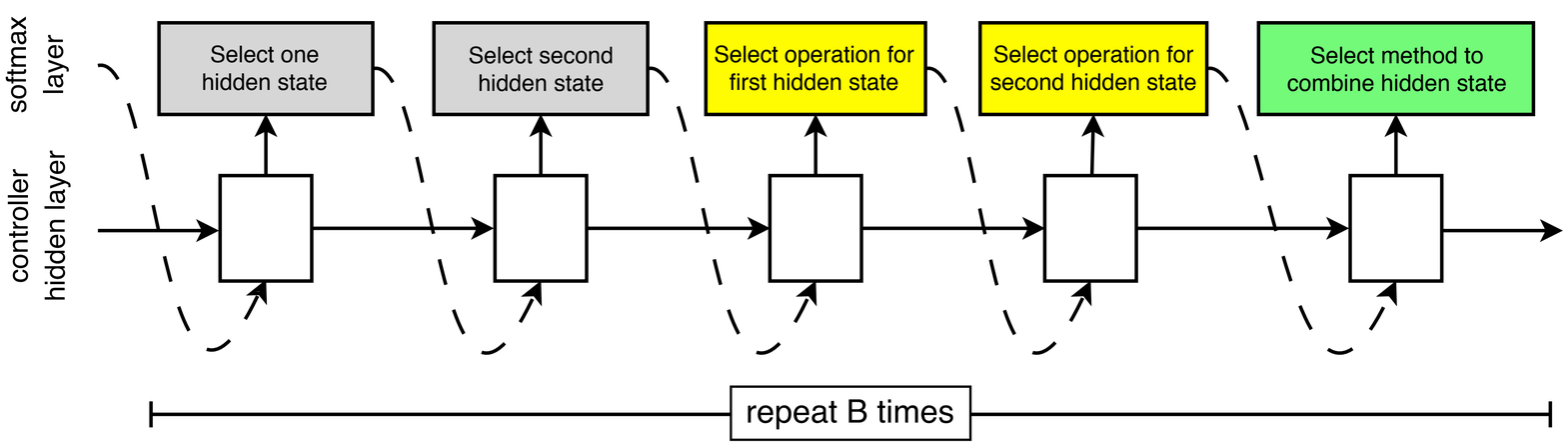
Classifier architecture on the ImageNet dataset

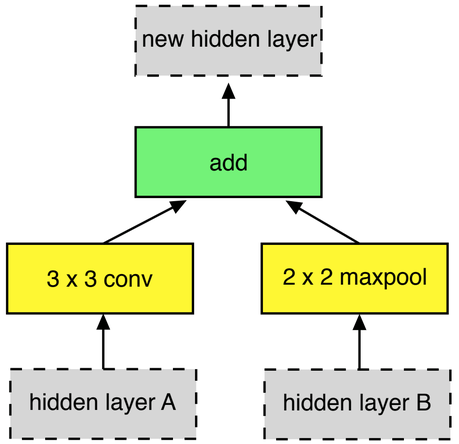
Learning Transferable Architectures for Scalable Image Recognition [Zoph *et al.*, CVPR 2018]

Cell-level Search Space

NASNet cell-level search space

- Left: **An RNN controller** generates the candidate cells five steps: finding two inputs, selecting two input transformation operations (e.g. convolution / pooling / identity), and finally selecting the method to combine the results. These five steps will be repeated for B times.
- Right: A cell generated after one step.



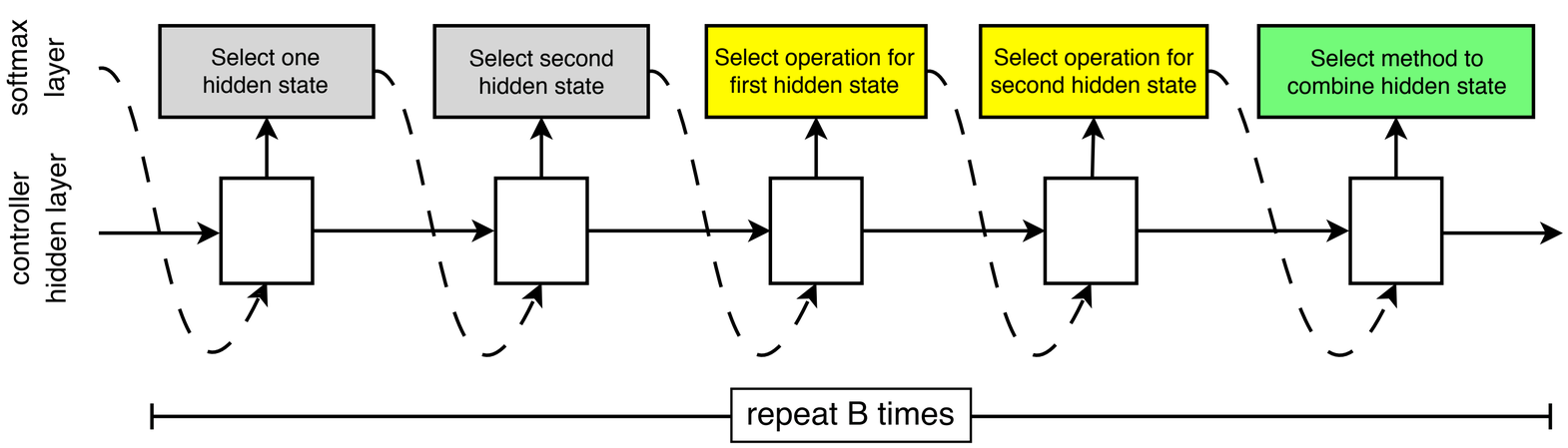


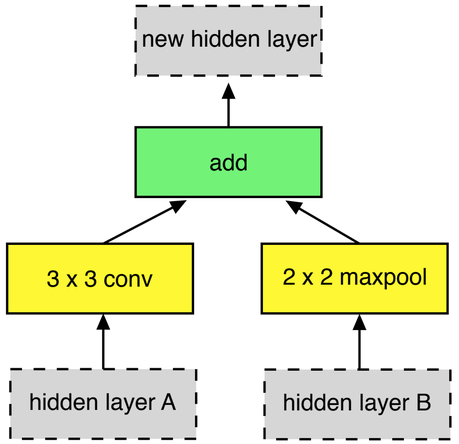
Learning Transferable Architectures for Scalable Image Recognition [Zoph *et al.*, CVPR 2018]

Cell-level Search Space

NASNet cell-level search space

- Question: Assuming that we have two candidate inputs, M candidate operations to transform the inputs and N potential operations to combine hidden states, what is the size of the search space in NASNet if we have B layers?
- Hint: Consider it step by step!



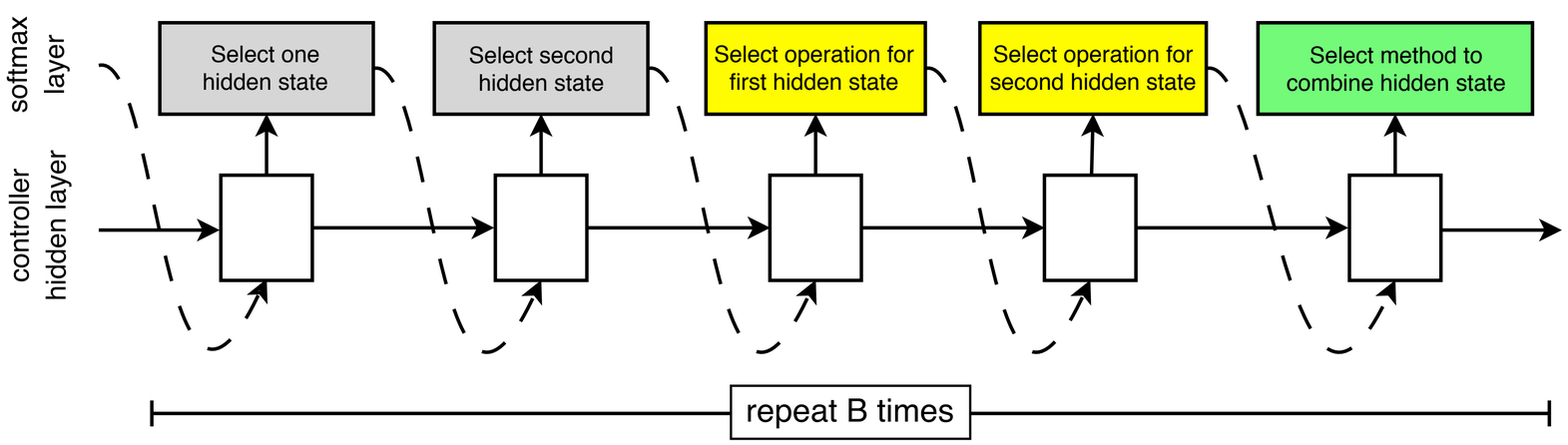


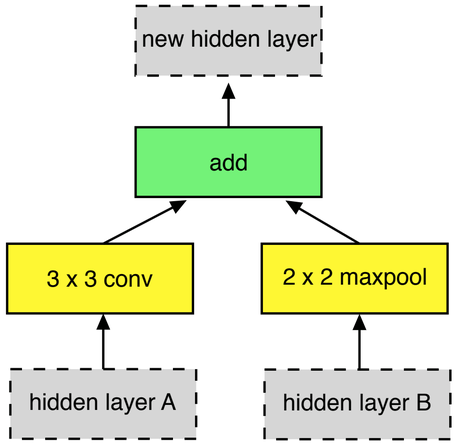
Learning Transferable Architectures for Scalable Image Recognition [Zoph *et al.*, CVPR 2018]

Cell-level Search Space

NASNet cell-level search space

- Question: Assuming that we have two candidate inputs, M candidate operations to transform the inputs and N potential operations to combine hidden states, what is the size of the search space in NASNet if we have B layers?
- Hint: Consider it step by step!
- Answer: $(2 \times 2 \times M \times M \times N)^B = 4^B M^{2B} N^B$.
- Assume M=5, N=2, B=5, we have 3.2×10^{11} candidates in the design space.



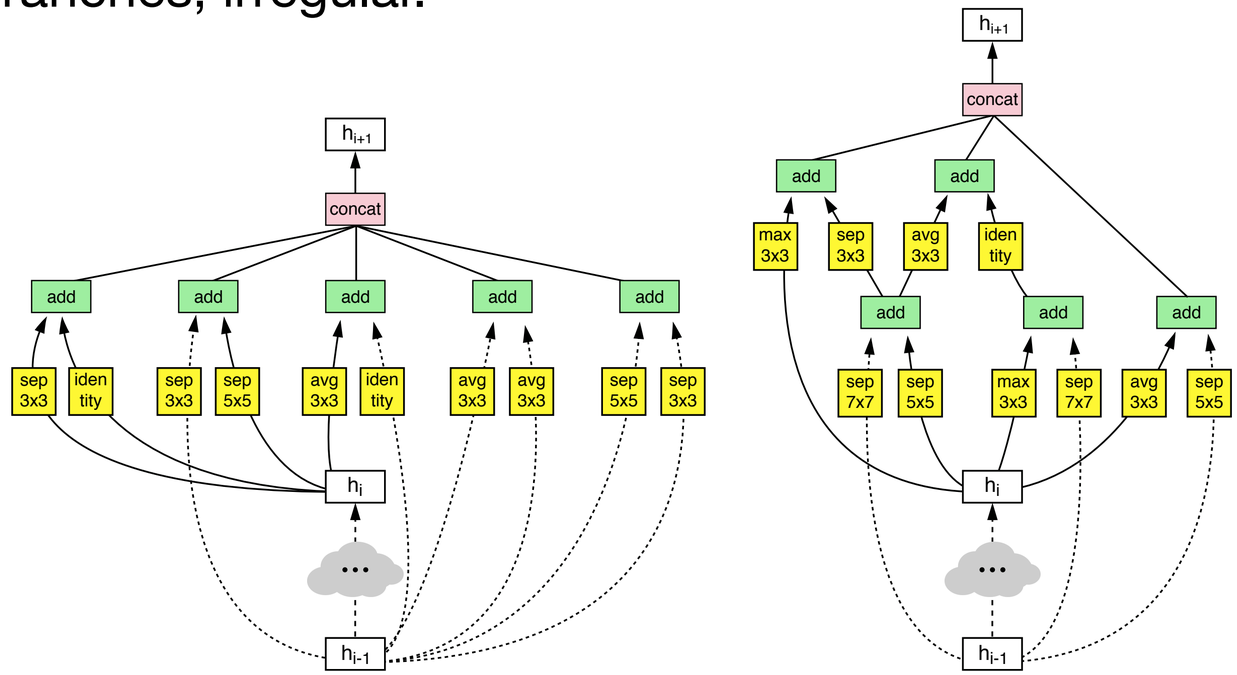


Learning Transferable Architectures for Scalable Image Recognition [Zoph *et al.*, CVPR 2018]

Cell-level Search Space

NASNet cell-level search space

- **Discussion**: What are the challenges for NASNet?
  Think about it from two perspectives: the **search cost** and the **hardware efficiency** of NASNets.
- **Hint**: the RNN controller requires accuracy feedback, which is obtained from training models from scratch; multiple branches, irregular.



Normal Cell *Reduction Cell*

Cell architectures found by NASNet

Learning Transferable Architectures for Scalable Image Recognition [Zoph *et al.*, CVPR 2018]

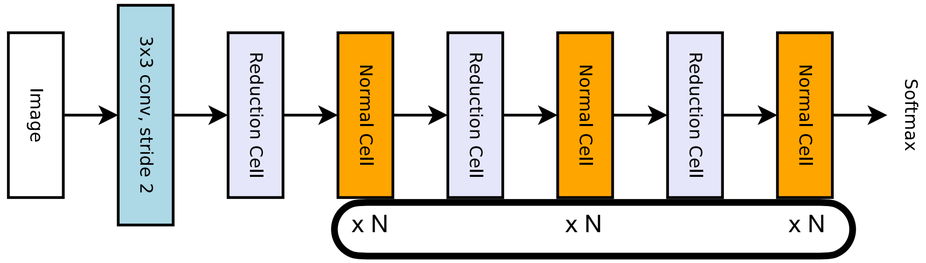
NAS Search Space

1. Cell-level Search Space
2. **Network-level Search Space**

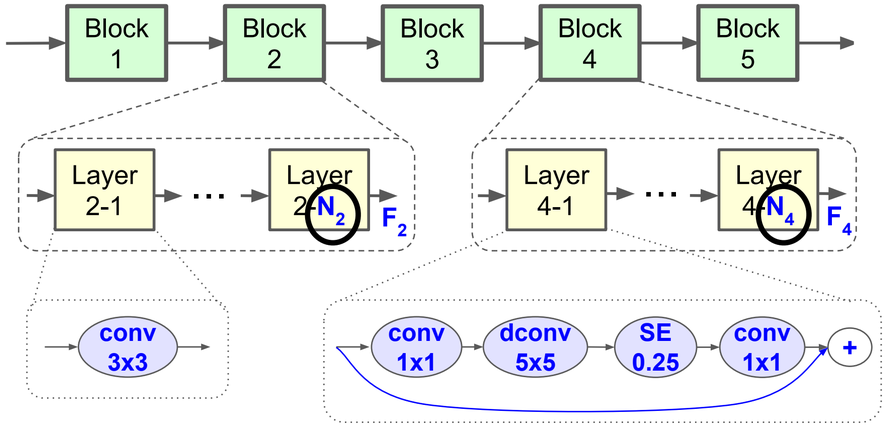
Network-Level Search Space

Network-level search space with fixed topology

- Network-level search space with fixed connection patterns is commonly used. Only the depth (i.e. number of building blocks) for each stage is searched.



NASNet [Zoph *et al.*, CVPR 2018]

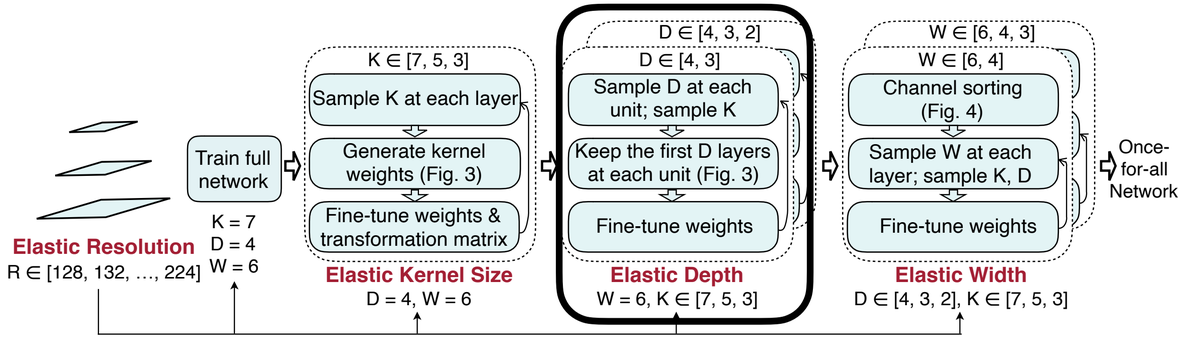


MnasNet [Tan *et al.*, CVPR 2019]

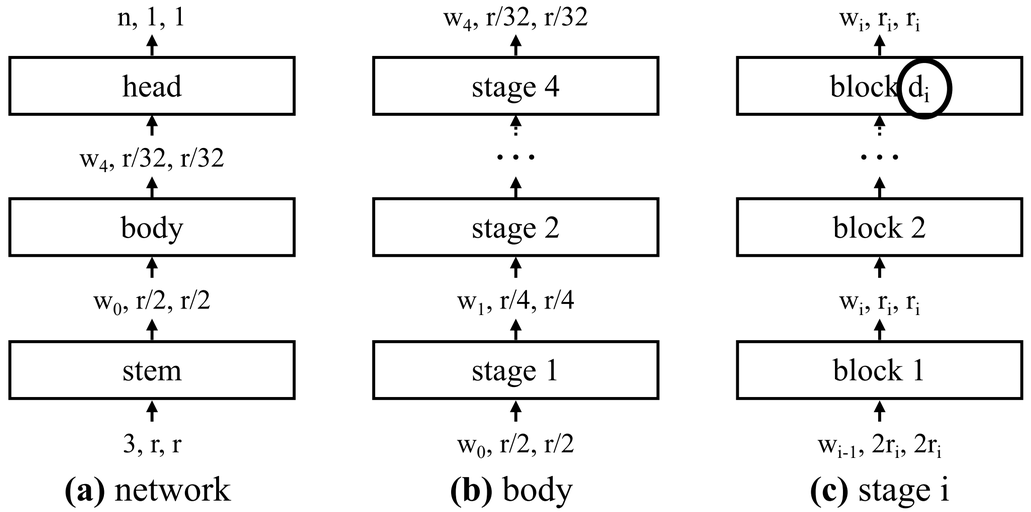
Network-Level Search Space

Network-level search space with fixed topology

- Network-level search space with fixed connection patterns is commonly used. Only the depth (i.e. number of building blocks) for each stage is searched.



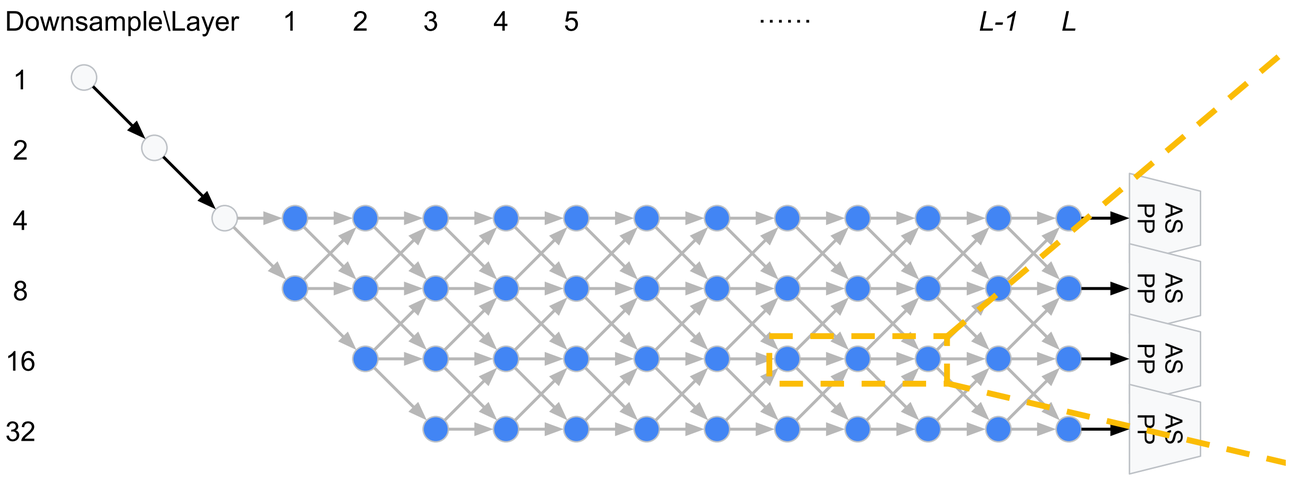
Once-for-All [Cai *et al.*, ICLR 2020]

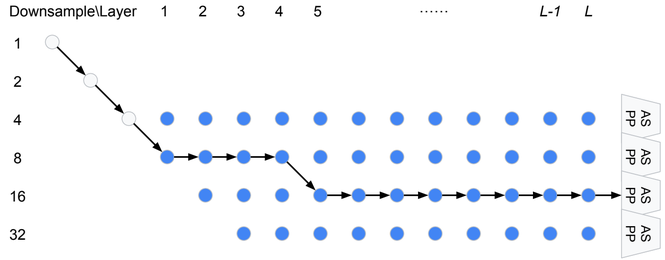


RegNet [Radosavovic *et al.*, CVPR 2020]

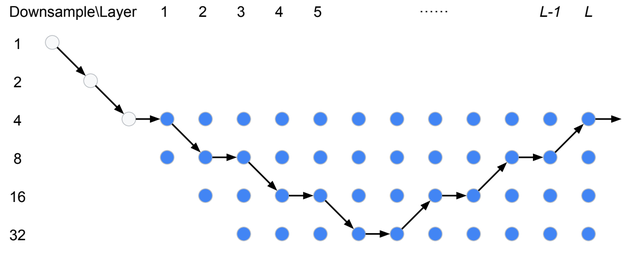
Network-Level Search Space

Network-level search space with searchable topology

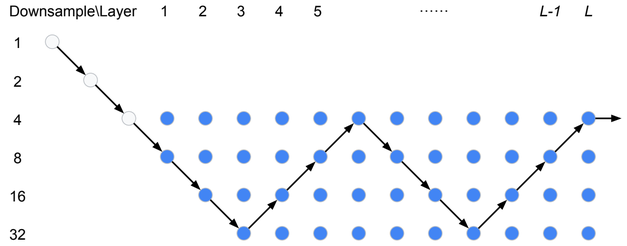




(a) Network level architecture used in DeepLabv3 [9].



(b) Network level architecture used in Conv-Deconv [56].



(c) Network level architecture used in Stacked Hourglass [55].

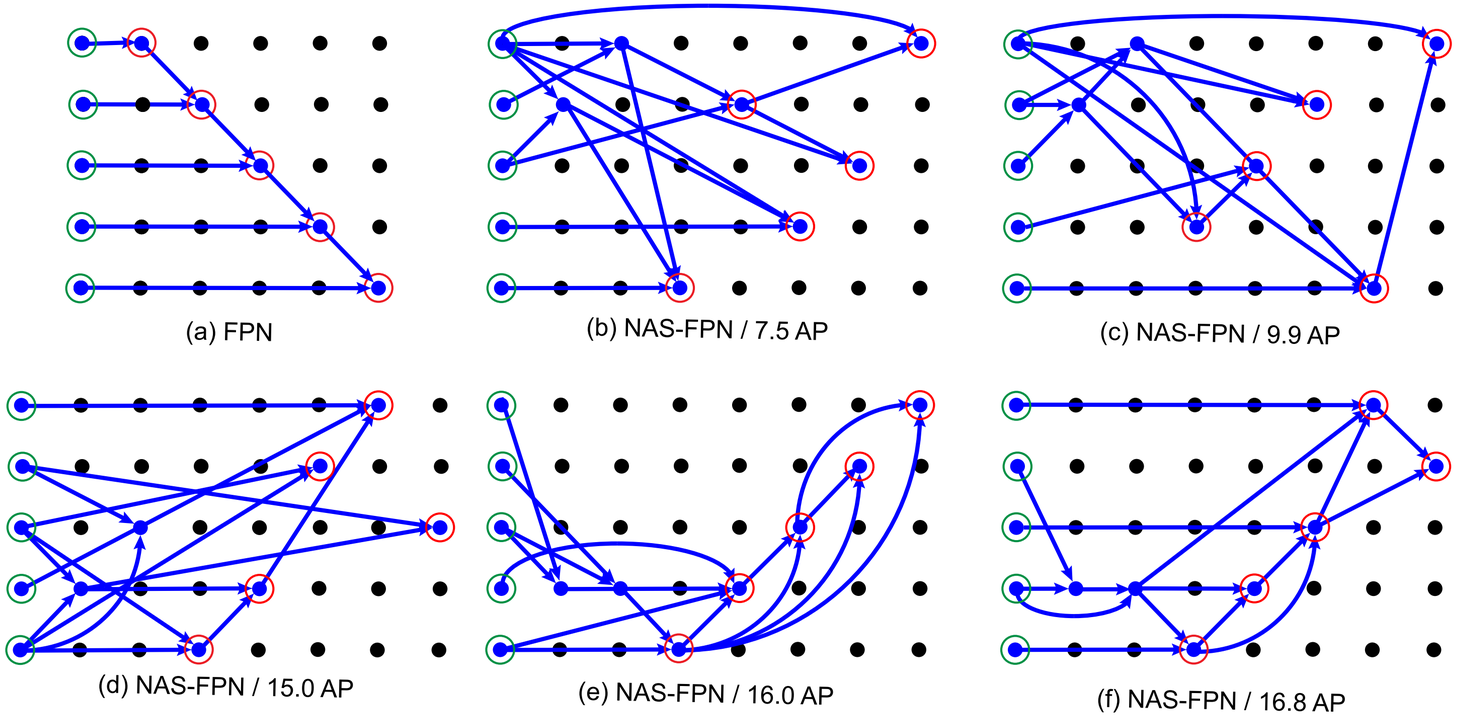
- Left: the network-level search space in AutoDeepLab. **Each path along the blue nodes** corresponds to a network architecture in the search space.
- Right: representative manual designs can be represented using this formulation. E.g.: DeepLabV3, UNet, Stacked Hourglass.

Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation [Liu *et al*., CVPR 2019]

Network-Level Search Space

Network-level search space with searchable topology

- NAS-FPN extends similar idea to object detection. Instead of always upsampling by 2x in the vanilla FPN, the most performant candidate (f) sometimes perform agressive 4x/8x upsampling.

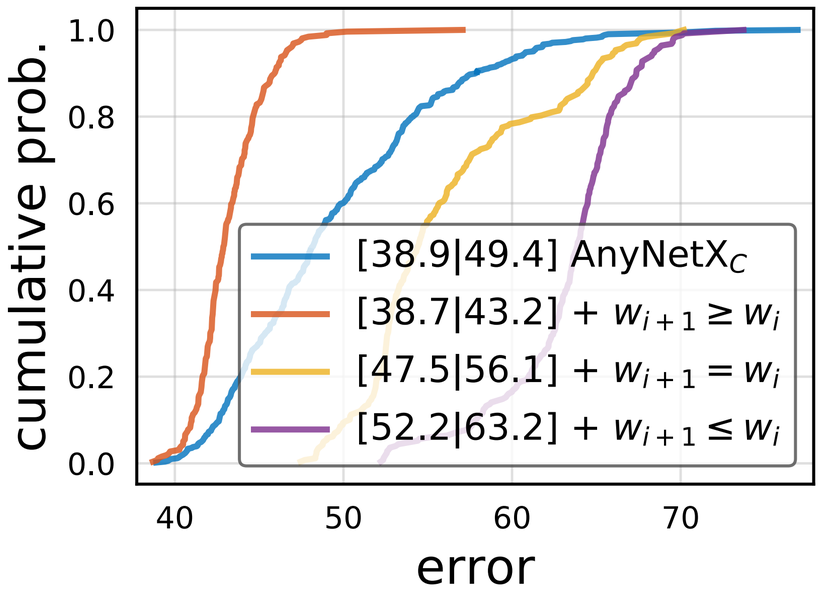


NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection [Ghaisi *et al.*, CVPR 2019]

Design the Search Space

Design the Search Space

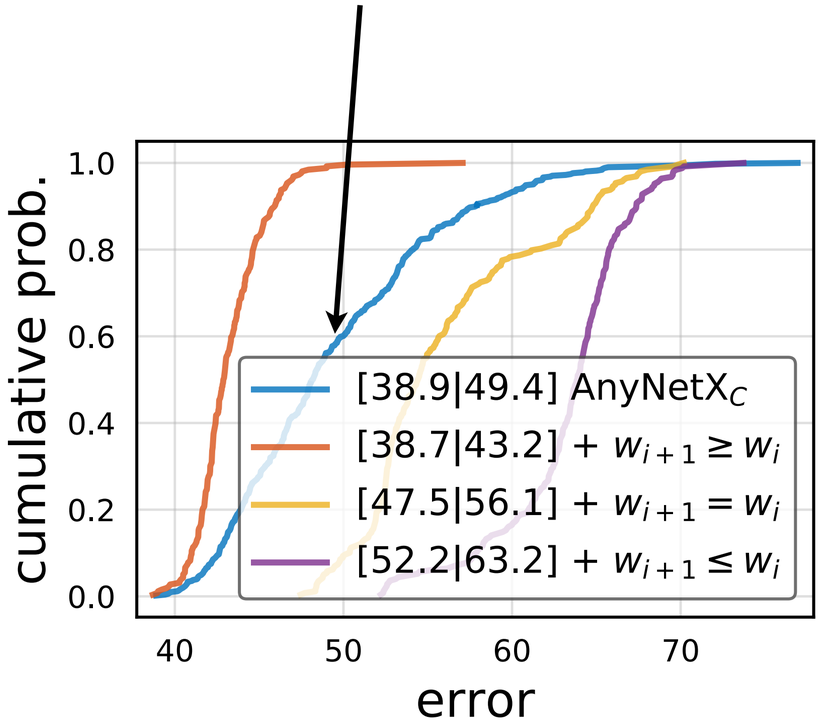
- RegNet: we can use the **cumulative error distribution** as a criterion for the search space quality.
- **Discussion**: for the four search spaces listed in the figure below, which one is the best and why?



On Network Design Spaces for Visual Recognition [Radosavovic *et al.*, ICCV 2019]

Design the Search Space

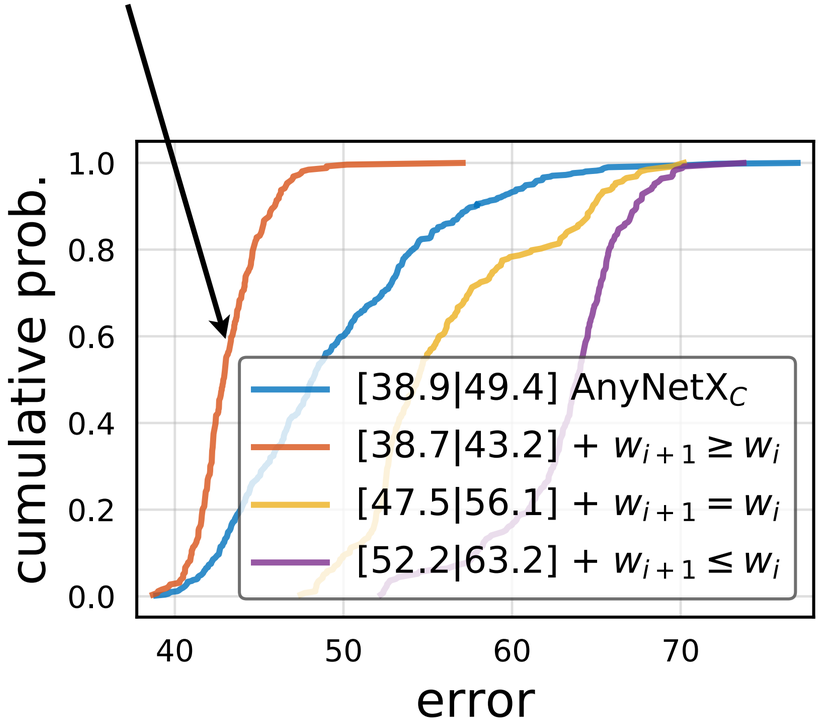
For the search space characterized by the **blue** curve,
38.9% of the models have more than 49.4% error.



On Network Design Spaces for Visual Recognition [Radosavovic *et al.*, ICCV 2019]

Design the Search Space

For the search space characterized by the **orange** curve,
38.7% of the models have more than 43.2% error.

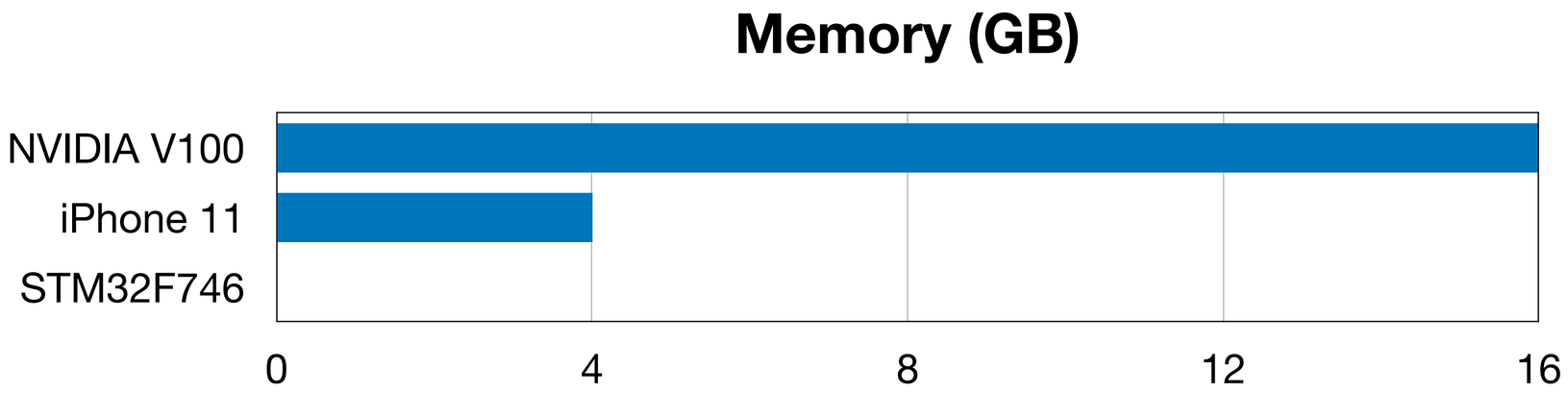


On Network Design Spaces for Visual Recognition [Radosavovic *et al.*, ICCV 2019]

Design the Search Space

Memory is important for TinyML

	Cloud AI (NVIDIA V100)	→	Mobile AI (iPhone 11)	→	Tiny AI (STM32F746)		ResNet-50	MobileNetV2	MobileNetV2 (int8)
Memory	16 GB	4× →	4 GB	3100× →	**320 kB**	← gap →	7.2 MB	6.8 MB	1.7 MB
Storage	TB~PB	1000× →	>64 GB	64000× →	**1 MB**	← gap →	102MB	13.6 MB	3.4 MB



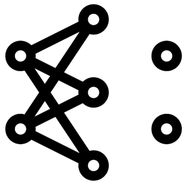
MCUNet: Tiny Deep Learning on IoT Devices [Lin *et al.*, NeurIPS 2020]

Design the Search Space

Memory is important for TinyML

Different from Mobile AI

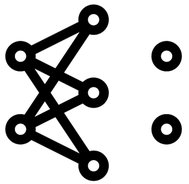
+



Latency Constraint

Energy Constraint

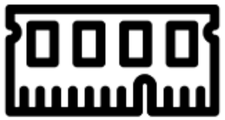
+



Latency Constraint

Energy Constraint

Memory Constraint

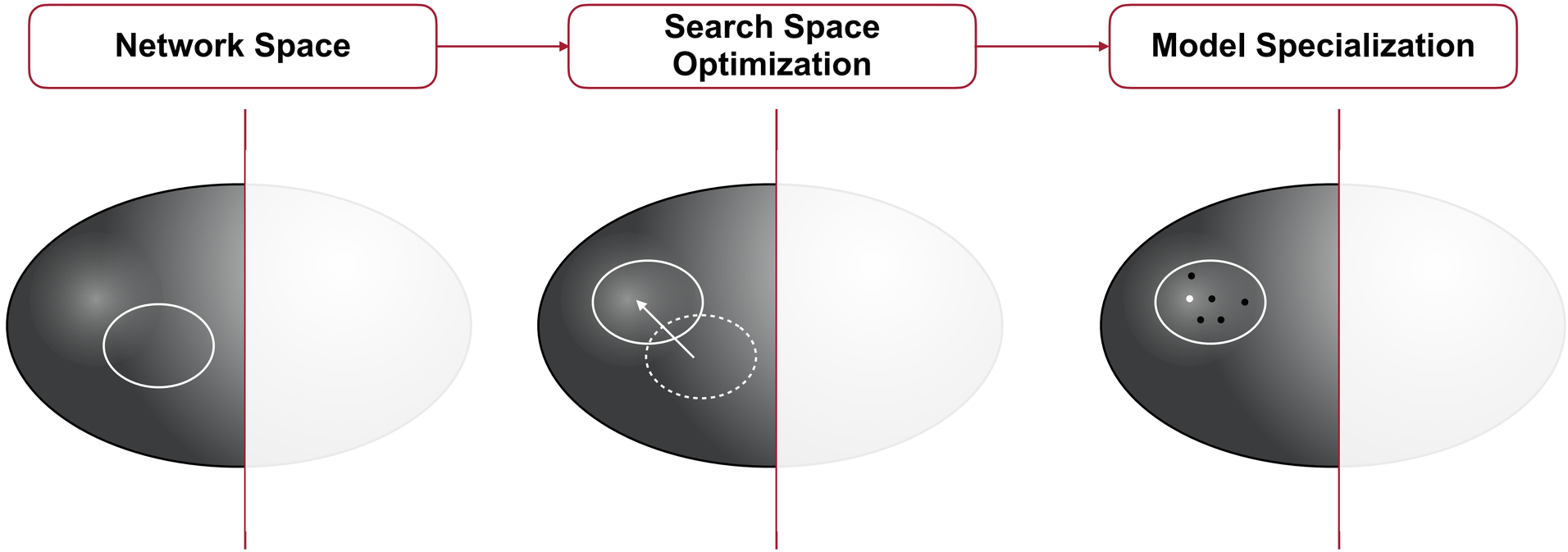


MCUNet: Tiny Deep Learning on IoT Devices [Lin *et al.*, NeurIPS 2020]

Design the Search Space for TinyML

Search space design is crucial for NAS performance
There is no prior expertise on MCU model design

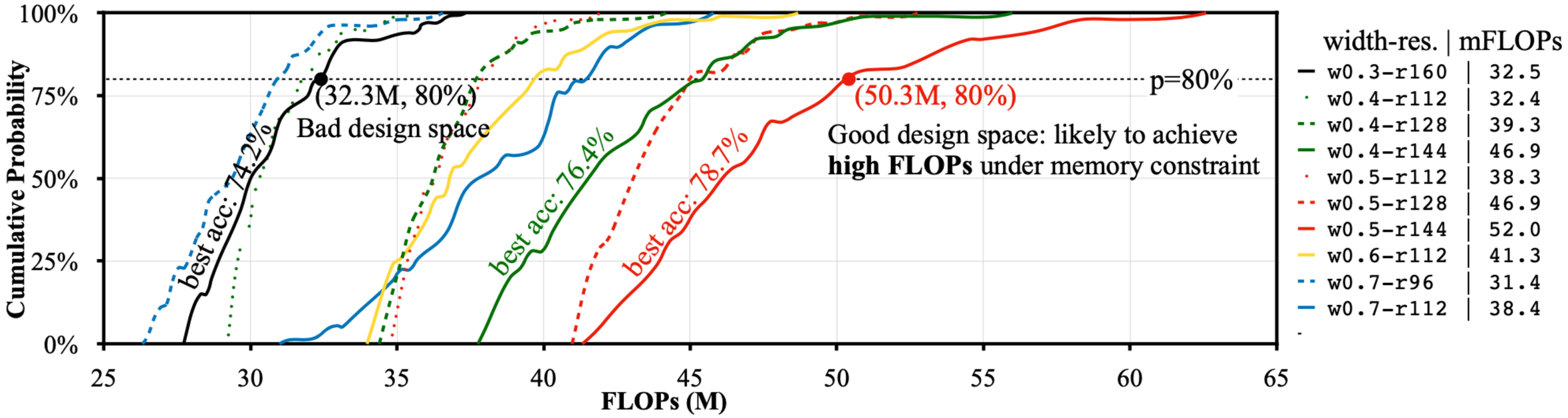
TinyNAS:
(1) Automated search space optimization
(2) Resource-constrained model specialization



MCUNet: Tiny Deep Learning on IoT Devices [Lin *et al.*, NeurIPS 2020]

Design the Search Space for TinyML

Analyzing **FLOPs distribution** of satisfying models:
Larger FLOPs -> Larger model capacity -> More likely to give higher accuracy



MCUNet: Tiny Deep Learning on IoT Devices [Lin *et al.*, NeurIPS 2020]

Design the Search Space for TinyML

- **Discussion**: what are the advantages / disadvantages of these two methods (RegNet, MCUNet) for search space design?

	R-18@224	Rand Space	Huge Space	Our Space
Acc.	80.3%	74.7±1.9%	77.0%	**78.7%**

Table 5. Our search space achieves the best accuracy, closer to ResNet-18@224 resolution (OOM). Randomly sampled and a huge space (contain many configs) leads to worse accuracy.

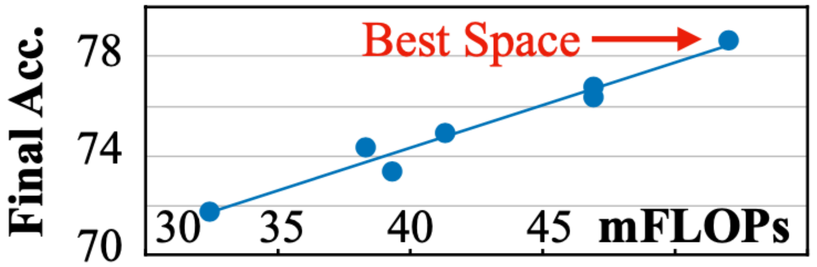


Figure 8. Search space with higher mean FLOPs leads to higher final accuracy.

Better search space, better final accuracy.

MCUNet: Tiny Deep Learning on IoT Devices [Lin *et al.*, NeurIPS 2020]

NAS Search Strategy

Search Strategy

Grid search

- Grid search is the traditional way of hyper parameter optimization. The entire design space is represented as the Cartesian product of single dimension design spaces (e.g. resolution in [1.0x, 1.1x,1.2x], width in [1.0x, 1.1x, 1.2x]).
- To obtain the accuracy of each candidate network, we train them from scratch.

Resolution / Width	**1.0x**	**1.1x**	**1.2x**
1.0x	50.0%	53.0%	**54.9%**
1.1x	51.0%	53.5%	55.4%
1.2x	52.0%	54.1%	56.2%

Satisfies the latency constraint

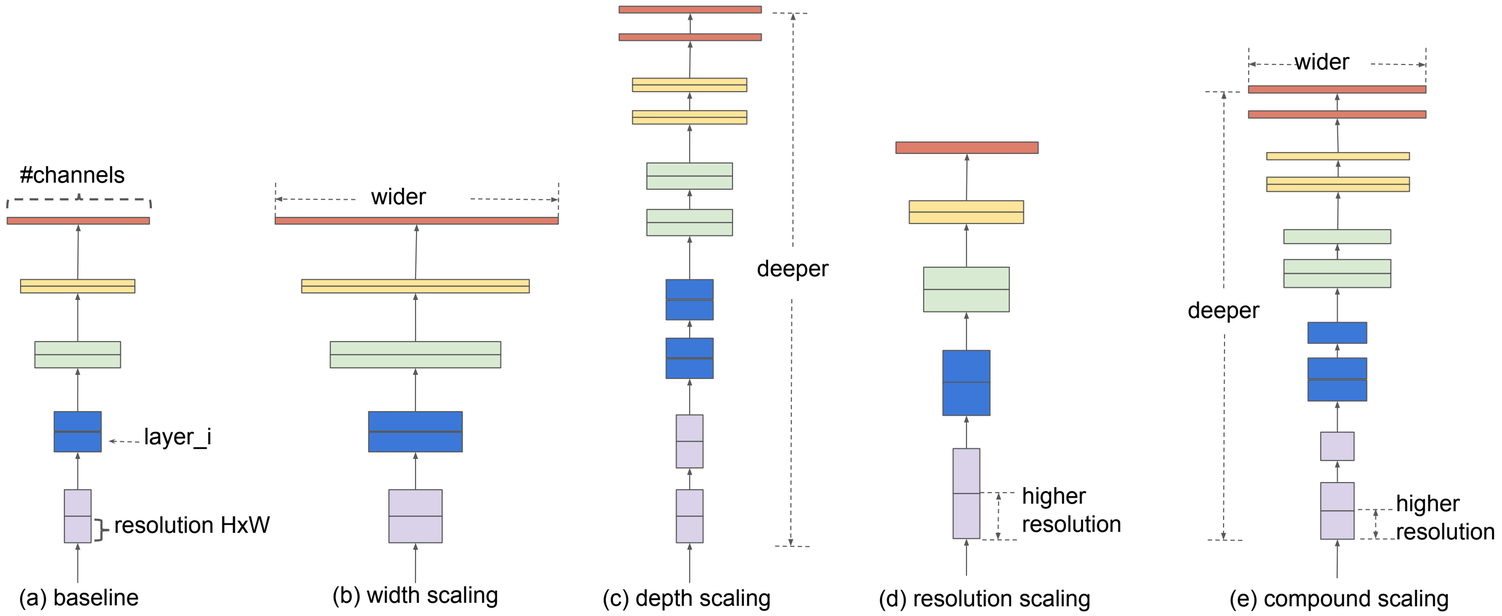
Breaks the latency constraint

Numbers in the grids correspond to the accuracy of candidate networks.

Search Strategy

Grid search

- EfficientNet applies **compound scaling** on **depth, width and resolution** to a starting network. It performs **grid search** on α, β, γ values such that the total FLOPs of the new model will be $2\times$ of the original network.



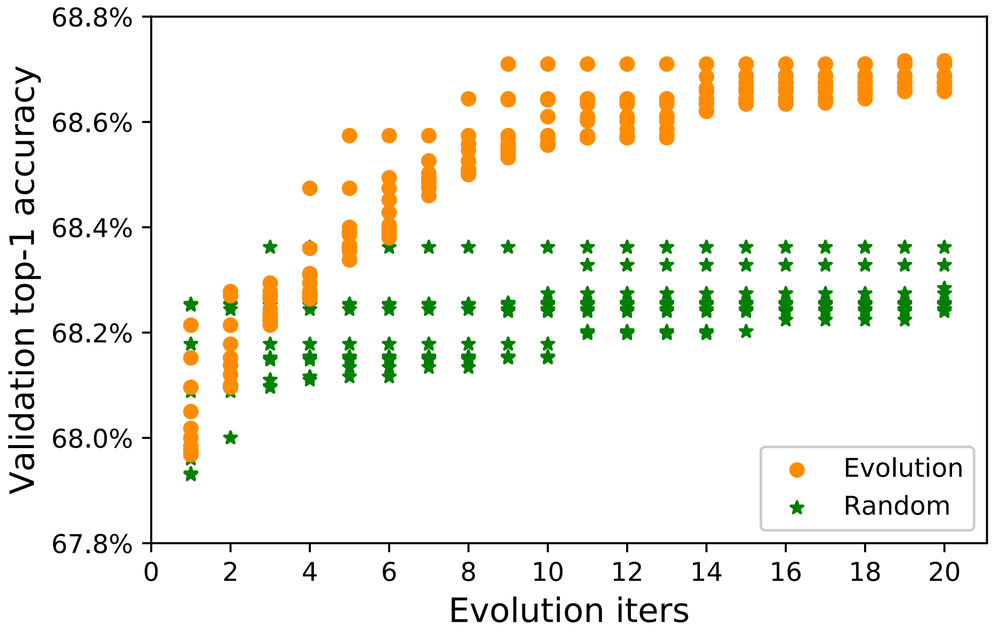
$$\text{depth: } d = \alpha^\phi$$
$$\text{width: } w = \beta^\phi$$
$$\text{resolution: } r = \gamma^\phi$$
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$
$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [Tan and Le, ICML 2019]

Search Strategy

Random search

- In Single-Path-One-Shot, with a good search space, random search is a very competitive baseline compared with advanced methods such as evolutionary architecture search.
- Left: subnet validation accuracy in the search phase; right: ImageNet FLOPs/accuracy tradeoff with subnets trained from scratch.



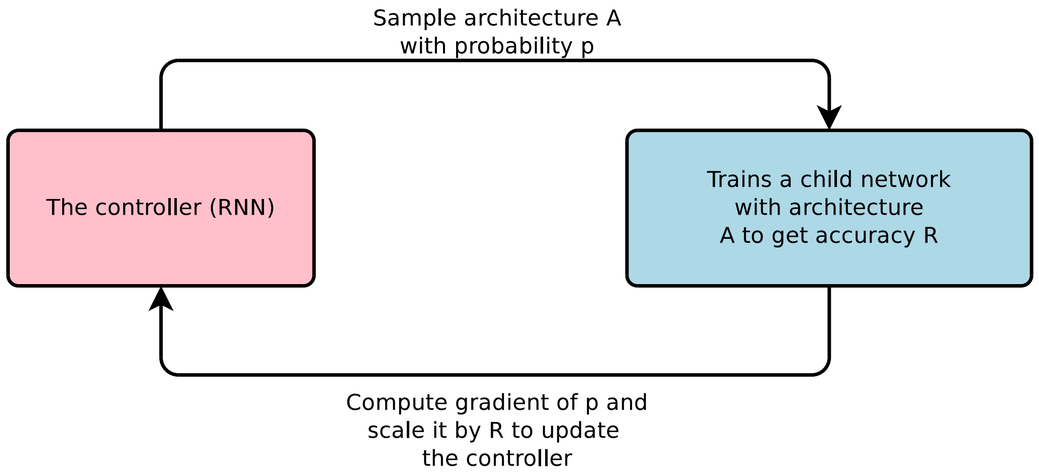
model	FLOPs	top-1 acc(%)
all choice_3	324M	73.4
all choice_5	321M	73.5
all choice_7	327M	73.6
all choice_x	326M	73.5
random select (5 times)	~320M	~73.7
SPS + random search	323M	73.8
ours (fully-equipped)	319M	**74.3**

Single Path One-Shot Neural Architecture Search with Uniform Sampling [Guo *et al.*, ECCV 2020]

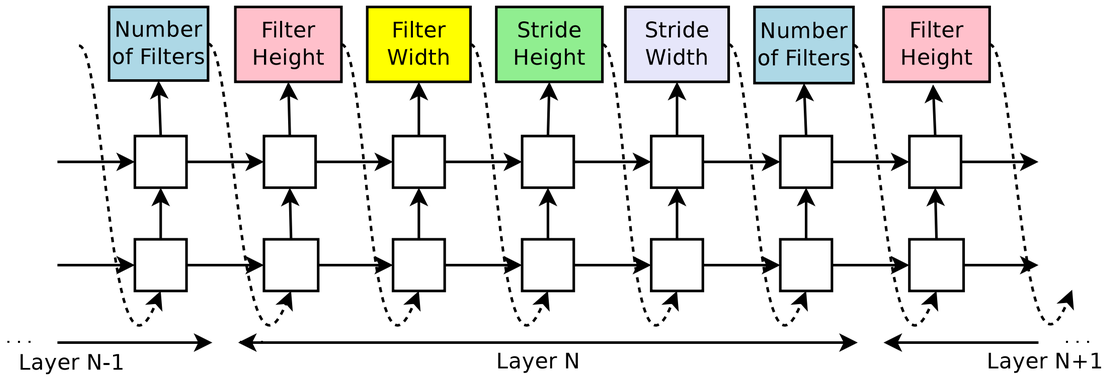
Search Strategy

Reinforcement learning

- Challenge: R (accuracy) is a non-differentiable objective. How to update the RNN parameters?



Overview of RL-based NAS



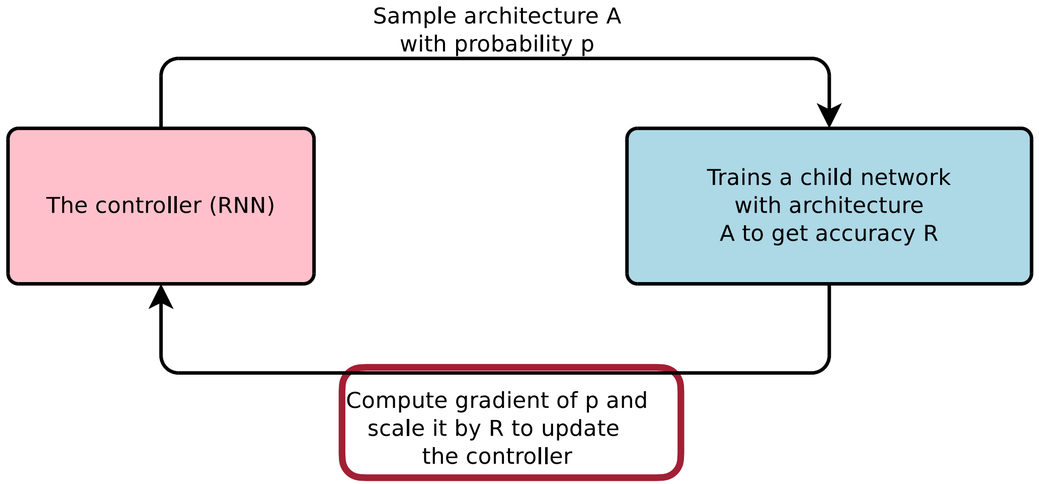
The RNN controller

Neural Architecture Search with Reinforcement Learning [Zoph and Le, ICLR 2017]

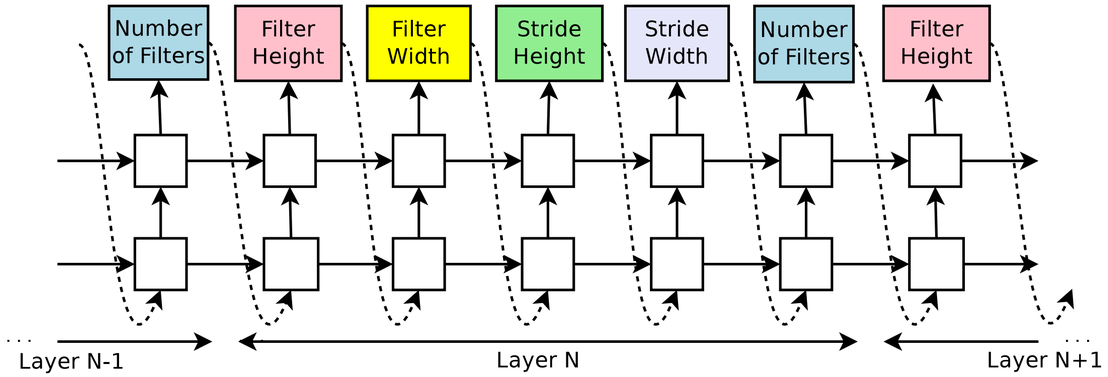
Search Strategy

Reinforcement learning

- Challenge: R (accuracy) is a non-differentiable objective. How to update the RNN parameters? Solution: Using a policy gradient method (REINFORCE) to update the parameters of the RNN controller.



Overview of RL-based NAS



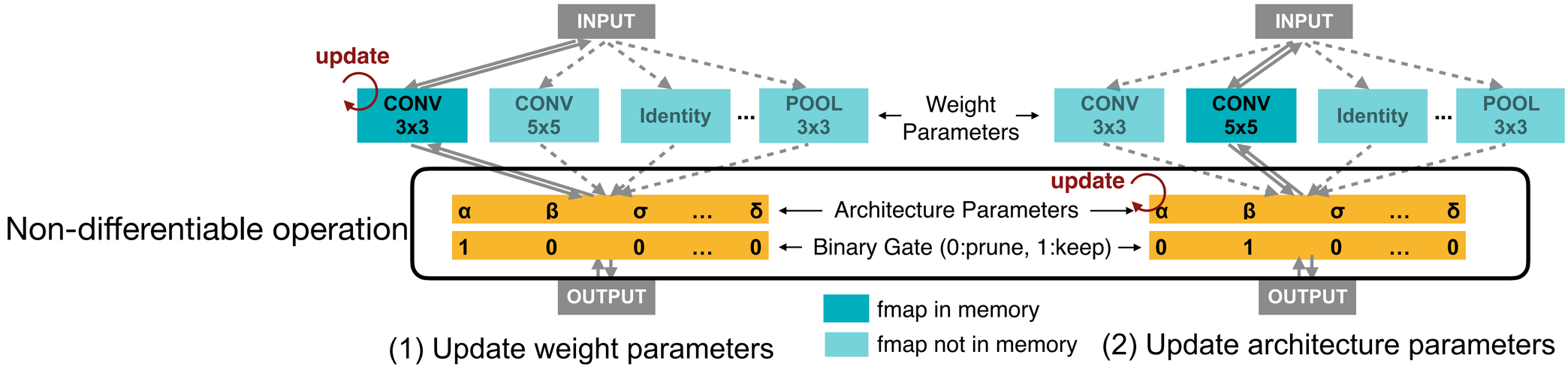
The RNN controller

Neural Architecture Search with Reinforcement Learning [Zoph and Le, ICLR 2017]

Search Strategy

Reinforcement learning

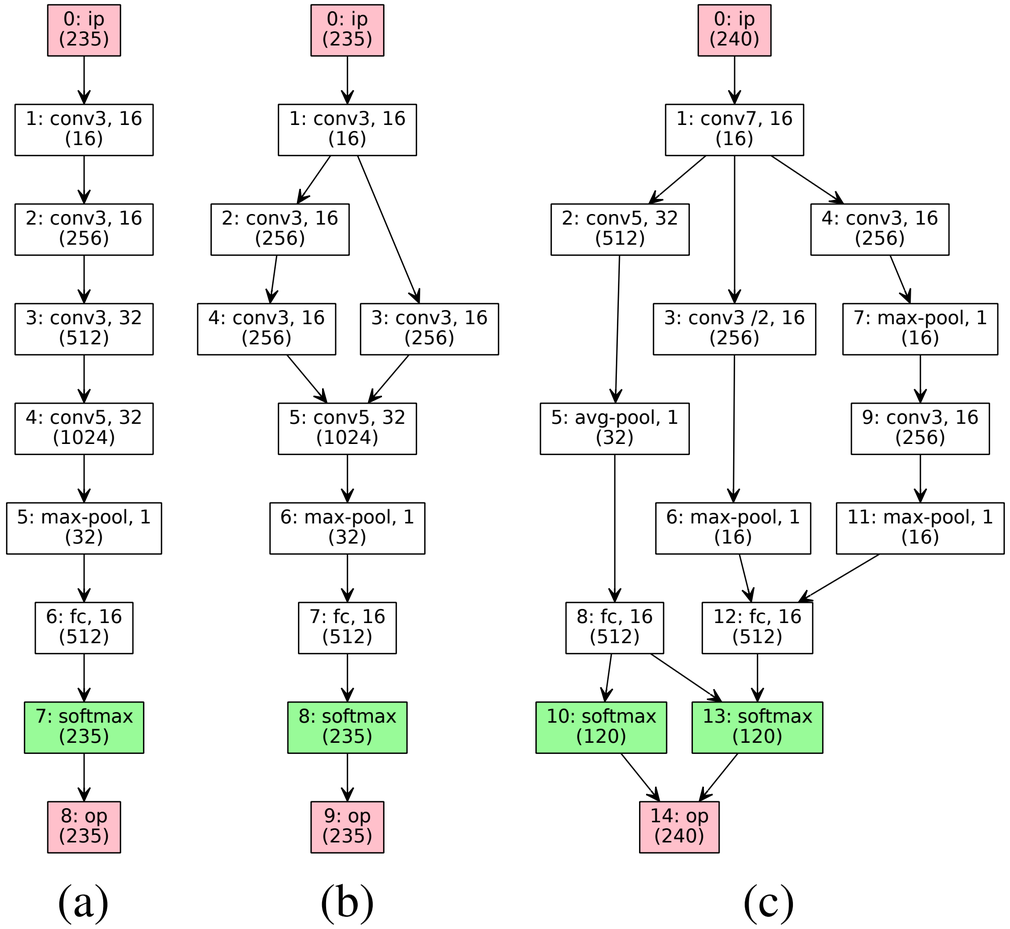
- RL-based search strategy can be generalized to other optimization problems with non-differentiable objectives.
- In ProxylessNAS, the architecture parameters are binarized to 0 and 1 values, indicating whether a path should be activated.
- The binarization process is non-differentiable, and it is possible to use RL to approximate the gradient w.r.t. the architecture parameters.



ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware [Cai *et al.*, ICLR 2019]

Search Strategy

Bayesian optimization



- **Idea**: balancing **exploitation** and **exploration** with the acquisition function (i.e. what is the next architecture to explore) in BO.
- Prioritizing exploration: go from a to c.
- Prioritizing exploitation: go from a to b.

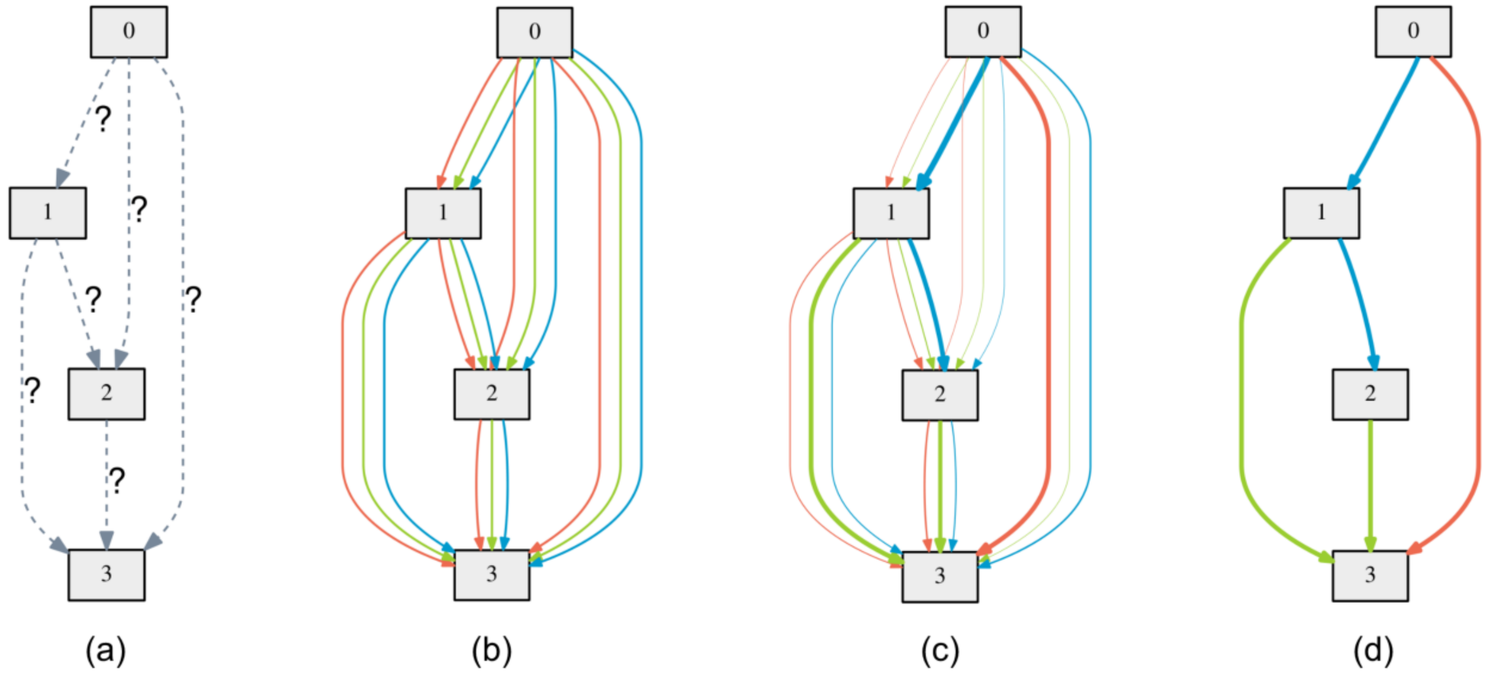
Neural Architecture Search with Bayesian Optimisation and Optimal Transport [Kandasamy *et al.*, NeurIPS 2018]

Search Strategy

Gradient-based search

- In DARTS, we represent the output at each node as a weighted sum of outputs from different edges (i.e. candidate op.s).

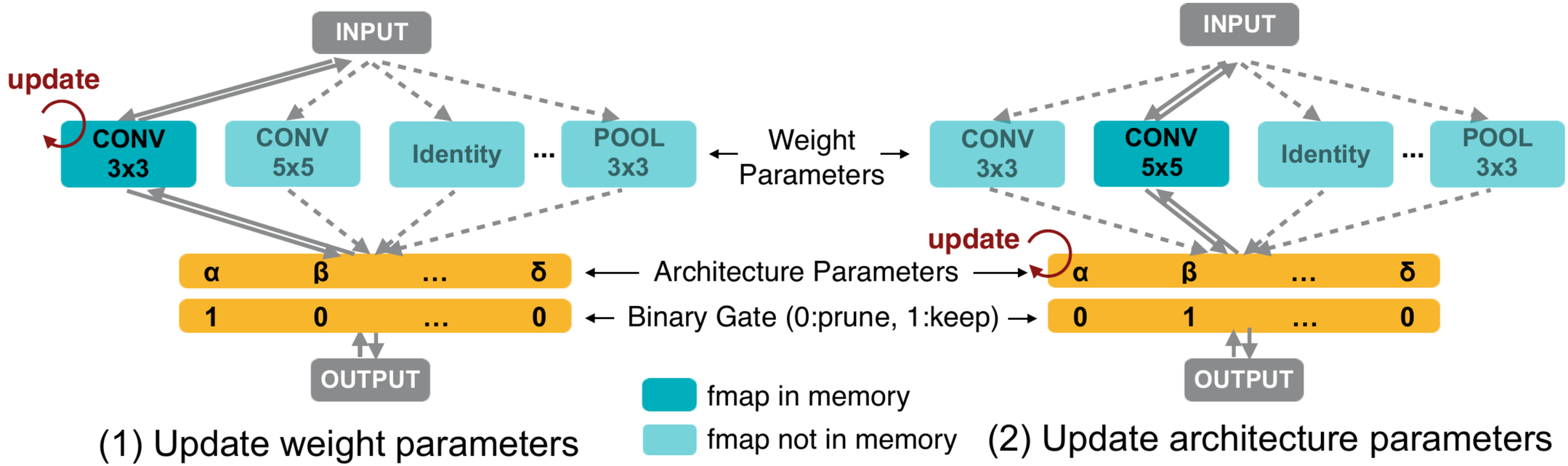
$$\bar{o}^{(i,j)}(x) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in \mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x)$$



DARTS: Differentiable Architecture Search [Liu *et al.*, ICLR 2019]

Search Strategy

Gradient-based search



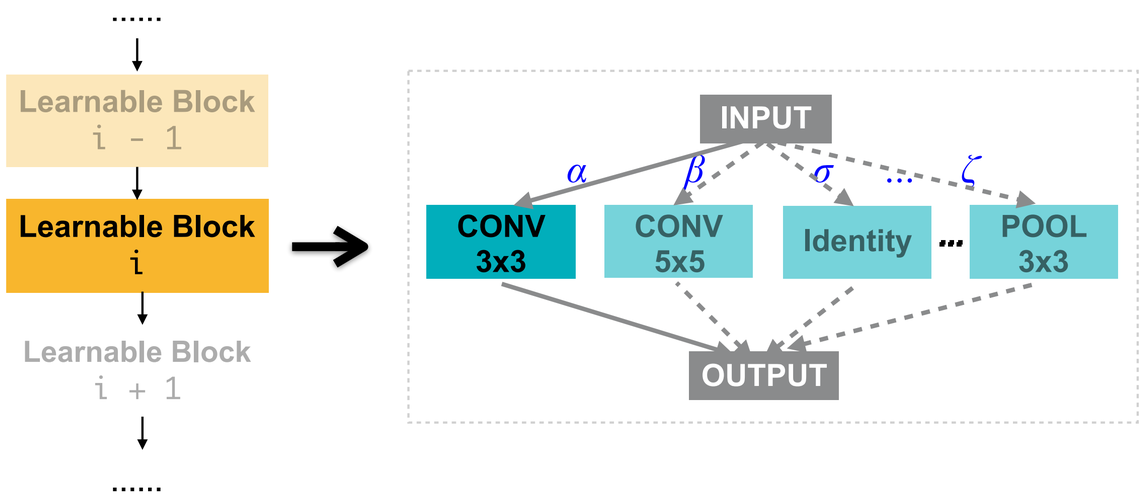
- ProxylessNAS alternatively updates **weight parameters** and **architecture parameters**.
- Continuos relaxation makes it possible for us to calculate the gradients of the architecture parameters.

ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware [Cai *et al.*, ICLR 2019]

Search Strategy

Gradient-based search

- It is also possible to take latency into account for gradient-based search.
- Here, F is a latency prediction model (typically a regressor or a lookup table). With such formulation, we can calculate an additional gradient for the architecture parameters from the latency penalty term.



$$\mathbb{E}[\text{Latency}] = \alpha \times F(\text{conv\_3x3}) + \beta \times F(\text{conv\_5x5}) + \sigma \times F(\text{identity}) + \ldots\ldots + \zeta \times F(\text{pool\_3x3})$$

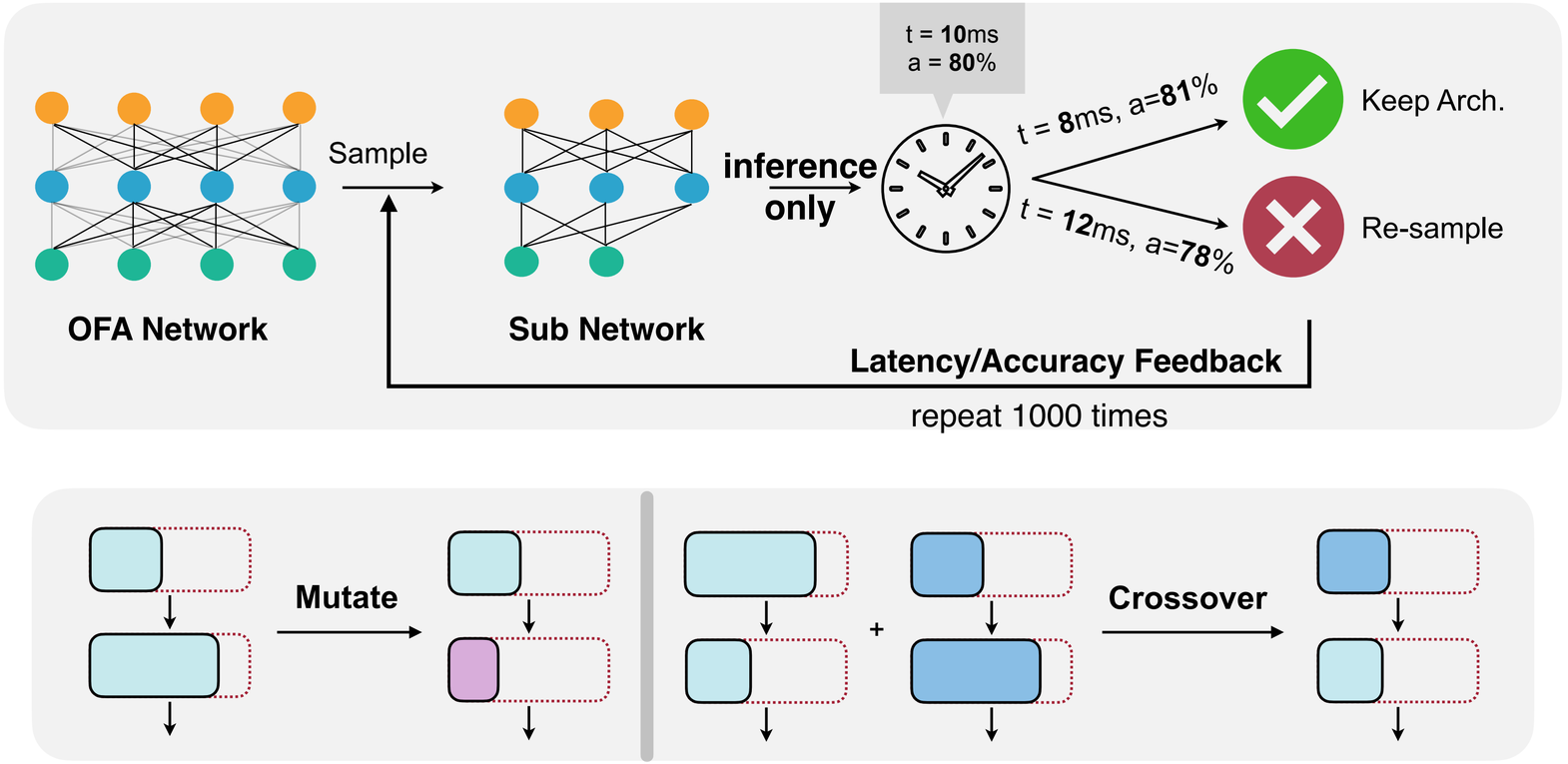
$$\mathbb{E}[\text{latency}] = \sum_i \mathbb{E}[\text{latency}_i]$$

$$Loss = Loss_{CE} + \lambda_1 ||w||_2^2 + \lambda_2 \mathbb{E}[\text{latency}]$$

ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware [Cai *et al.*, ICLR 2019]

Search Strategy

Evolutionary search

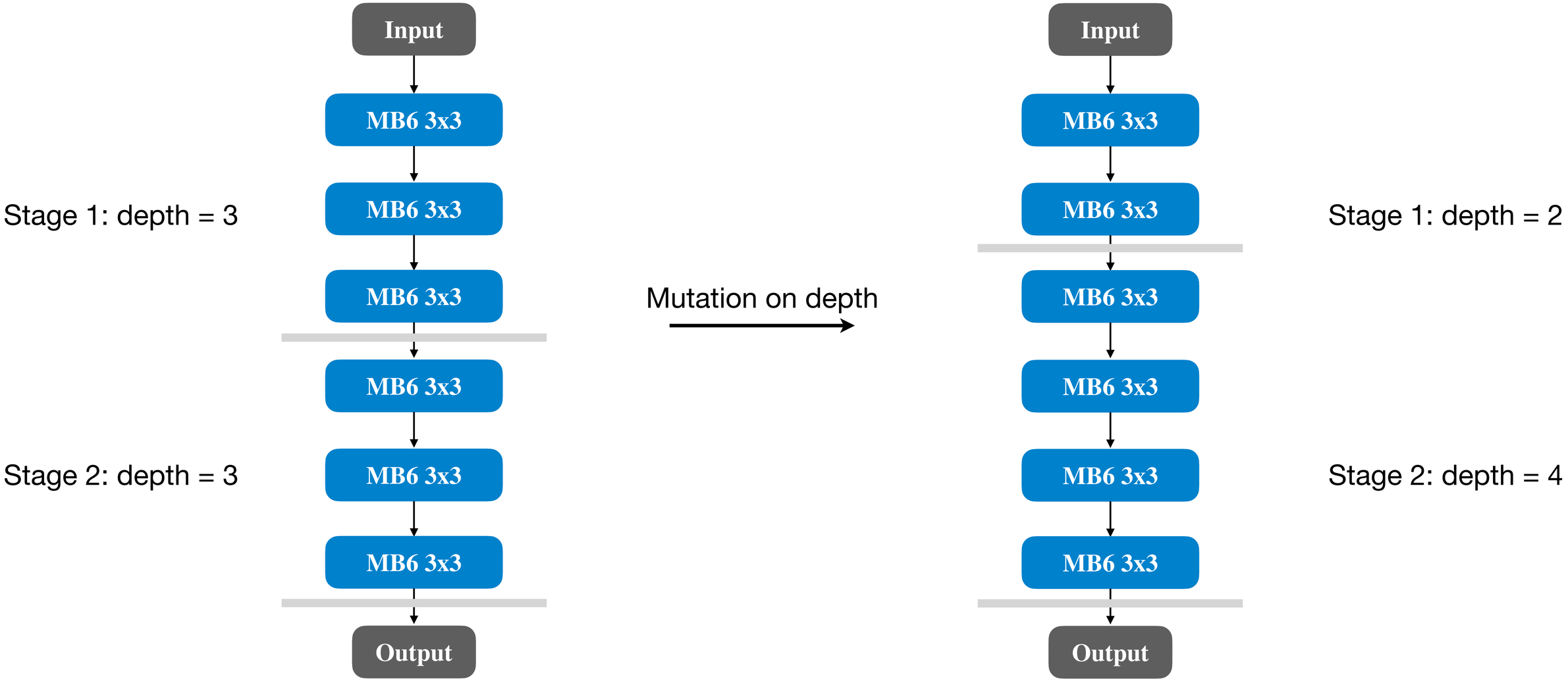


An overview for evolution-based architecture search

PVNAS: 3D Neural Architecture Search with Point-Voxel Convolution [Liu *et al.*, TPAMI 2021]

Search Strategy

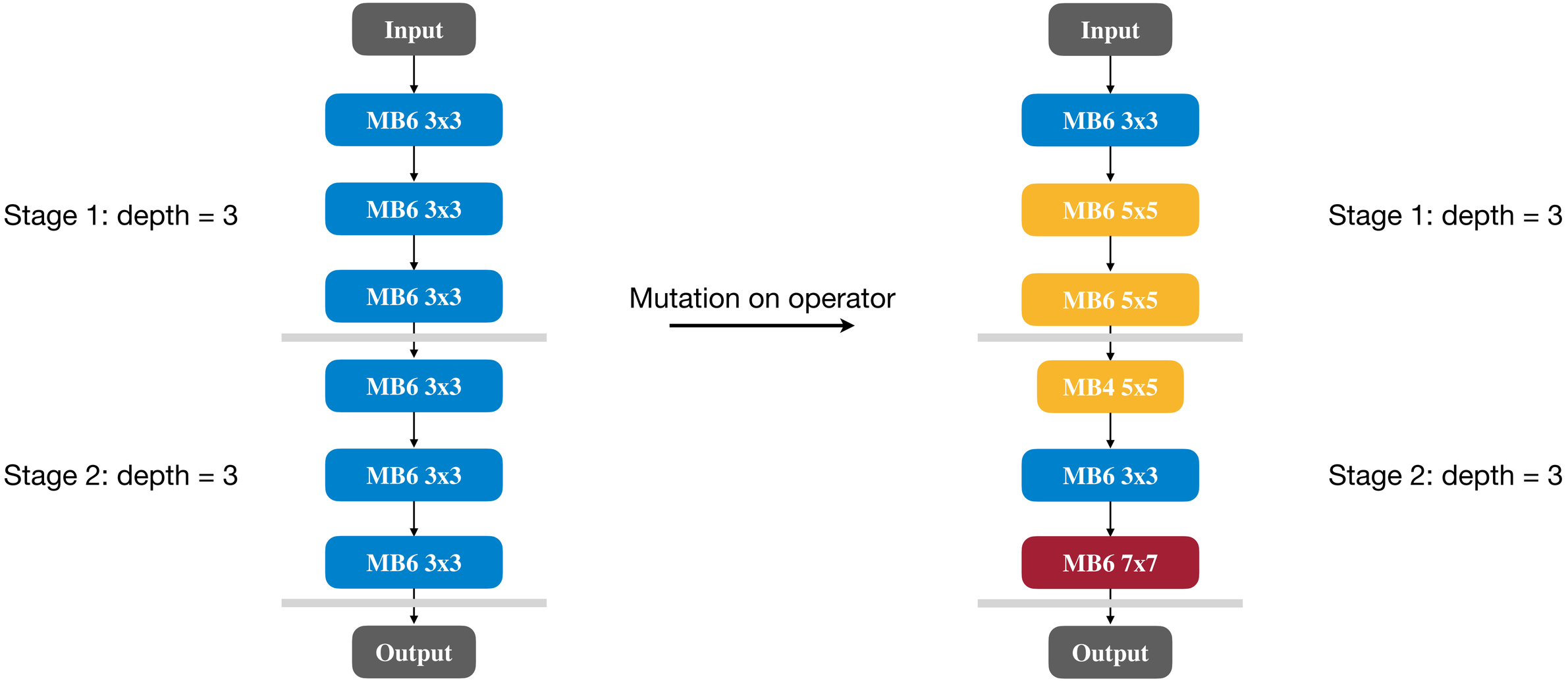
Evolutionary search



Once-for-All: Train One Network and Specialize it for Efficient Deployment [Cai *et al.*, ICLR 2020]

Search Strategy

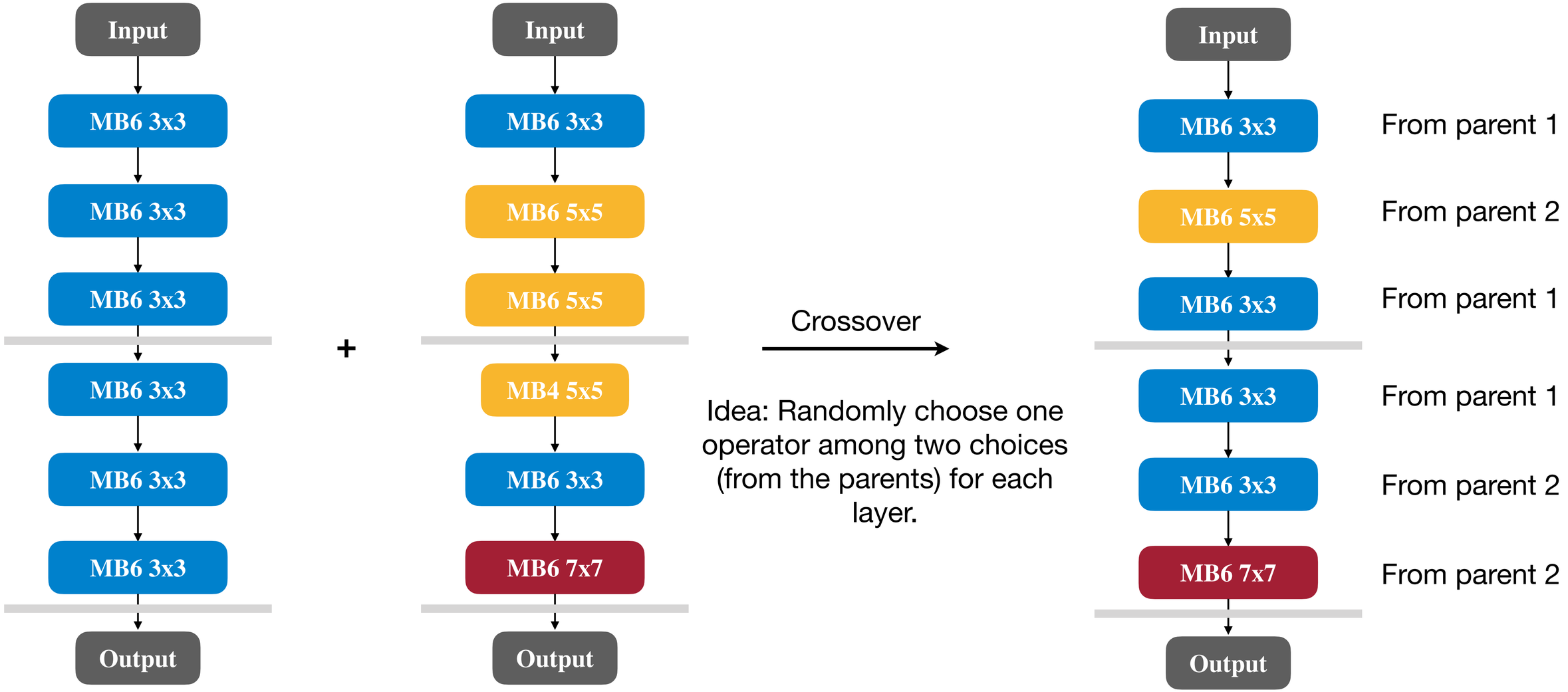
Evolutionary search



Once-for-All: Train One Network and Specialize it for Efficient Deployment [Cai *et al.*, ICLR 2020]

Search Strategy

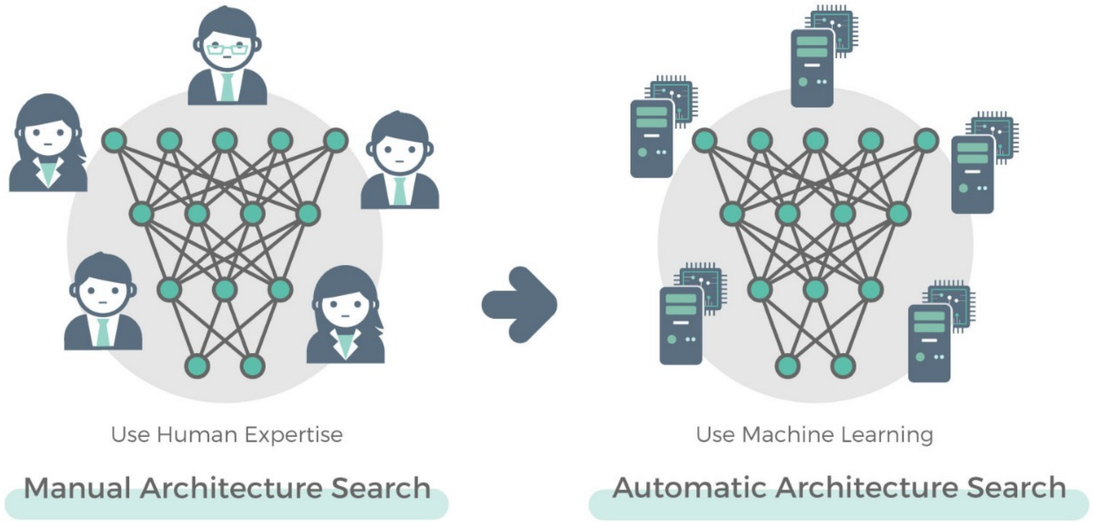
Evolutionary search



Once-for-All: Train One Network and Specialize it for Efficient Deployment [Cai *et al.*, ICLR 2020]

Summary of Today's Lecture

- In this lecture, we introduce:
  - Basic concepts and manually designed neural networks
  - What is neural architecture search (NAS)
  - The search space for NAS
  - The search strategy for NAS
- We will cover in later lectures:
  - Training in NAS
  - Hardware-aware NAS
  - Training-free NAS
  - Neural Hardware Architecture Search
  - NAS Applications



References

1. Deep Residual Learning for Image Recognition [He et al., CVPR 2016]
2. ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky et al., NeurIPS 2012]
3. Very Deep Convolutional Networks for Large-scale Image Recognition [Simonyan et al., ICLR 2015]
4. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size [Iandola et al., arXiv 2016]
5. Aggregated Residual Transformations for Deep Neural Networks [Xie et al., CVPR 2017]
6. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Howard et al., arXiv 2017]
7. MobileNetV2: Inverted Residuals and Linear Bottlenecks [Sandler et al., CVPR 2018]
8. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices [Zhang et al., CVPR 2018]
9. Neural Architecture Search: A Survey [Elskan et al., JMLR 2019]
10. Learning Transferable Architectures for Scalable Image Recognition [Zoph et al., CVPR 2018]
11. DARTS: Differentiable Architecture Search [Liu et al., ICLR 2019]

References

- 12. MnasNet: Platform-Aware Neural Architecture Search for Mobile [Tan *et al.*, CVPR 2019]
- 13. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware [Cai et al., ICLR 2019]
- 14. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search [Wu et al., CVPR 2019]
- 15. Designing Network Design Spaces [Radosavovic *et al.*, CVPR 2020]
- 16. Single Path One-Shot Neural Architecture Search with Uniform Sampling [Guo *et al.*, ECCV 2020]
- 17. Once-for-All: Train One Network and Specialize it for Efficient Deployment [Cai *et al.*, ICLR 2020]
- 18. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation [Liu *et al*., CVPR 2019]
- 19. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection [Ghaisi et al., CVPR 2019]
- 20. Exploring Randomly Wired Neural Networks for Image Recognition [Xie et al., ICCV 2019]
- 21. MCUNet: Tiny Deep Learning on IoT Devices [Lin et al., NeurIPS 2020]

References

- 22. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [Tan and Le, ICML 2019]
- 23. Neural Architecture Search with Reinforcement Learning [Zoph and Le, ICLR 2017]
- 24. PVNAS: 3D Neural Architecture Search with Point-Voxel Convolution [Liu *et al.*, TPAMI 2021]
- 25. Regularized Evolution for Image Classifier Architecture Search [Real *et al.*, AAAI 2019]