Foundations of Edge AI

Lecture11

Neural Architecture Search

Lanyu (Lori) Xu

Email: lxu@oakland.edu
Homepage: https://lori930.github.io/
Office: EC 524

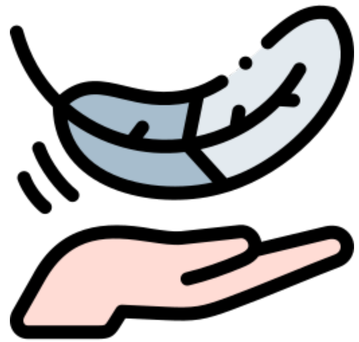
Lecture Plan

Today we will

- Review primitive operations of deep neural networks
- Introduce classic manual-designed building blocks
- Introduce **Neural Architecture Search (NAS)**, an automatic technique for designing NN architectures
  - How to design the search space
  - What are NAS search strategies

Efficiency of Neural Networks

Faster, smaller, greener

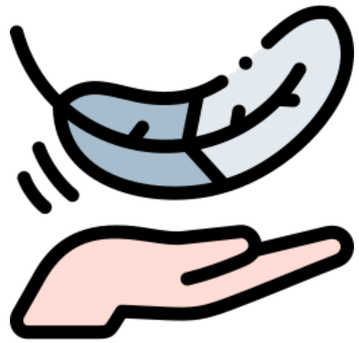


<u>Memory related</u>
 # parameters
Model size
Total/peak #activations

<u>Computation related</u>
MAC
FLOP, FLOPS
OP, OPS

The Goal of NAS

Find the best design point to balance all the design goals



The trade-off between efficiency and accuracy

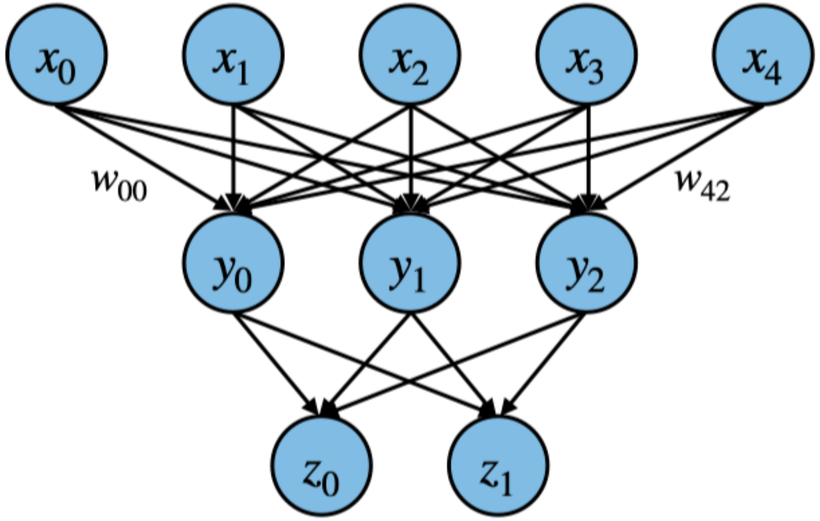
Review Primitive Operations

Fully-Connected Layer (Linear Layer)

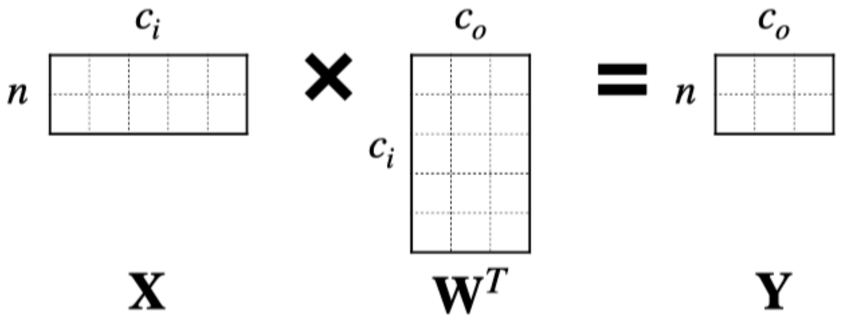
The output neuron is connected to all input neurons

- Shape of tensors
  - Input features $X : (n, c_i)$
  - Output features $Y : (n, c_o)$
  - Weights $W : (c_o, c_i)$
  - Bias $b : (c_o,)$

Notations	
c_i	Input channels
c_o	Output channels
n	Batch size



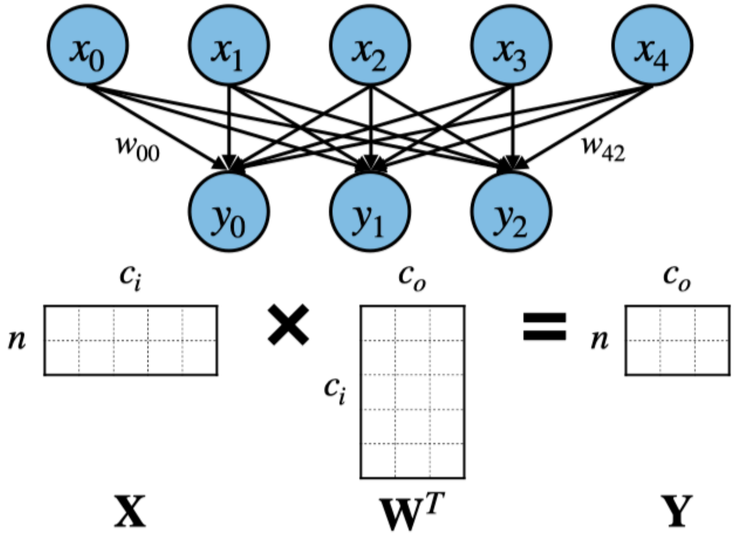
Multilayer Perceptron (MLP)



Parameters

Parameter (synapse/weight) count of the given neural network, i.e., # elements in the weight tensors

Layer	#Parameters (Bias is ignored)
Linear Layer	$c_o \cdot c_i$

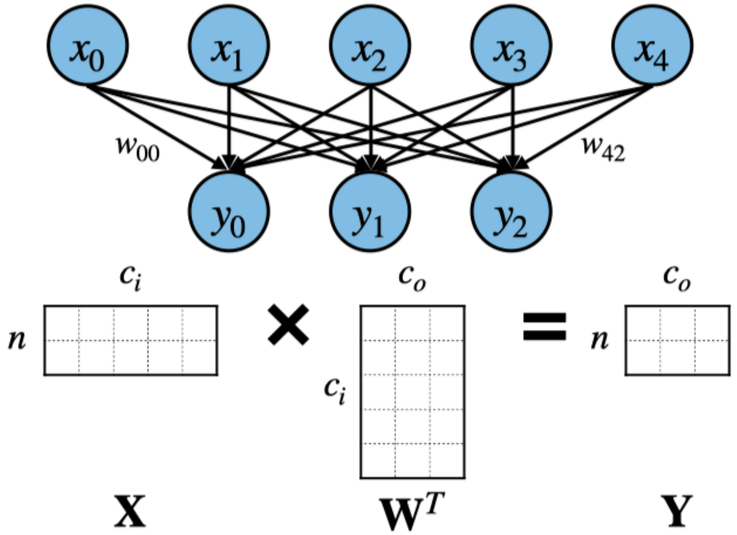


Notations	
c_i, c_o	Input/output
g	Groups
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width

Multiply-Accumulate Operations

MAC

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

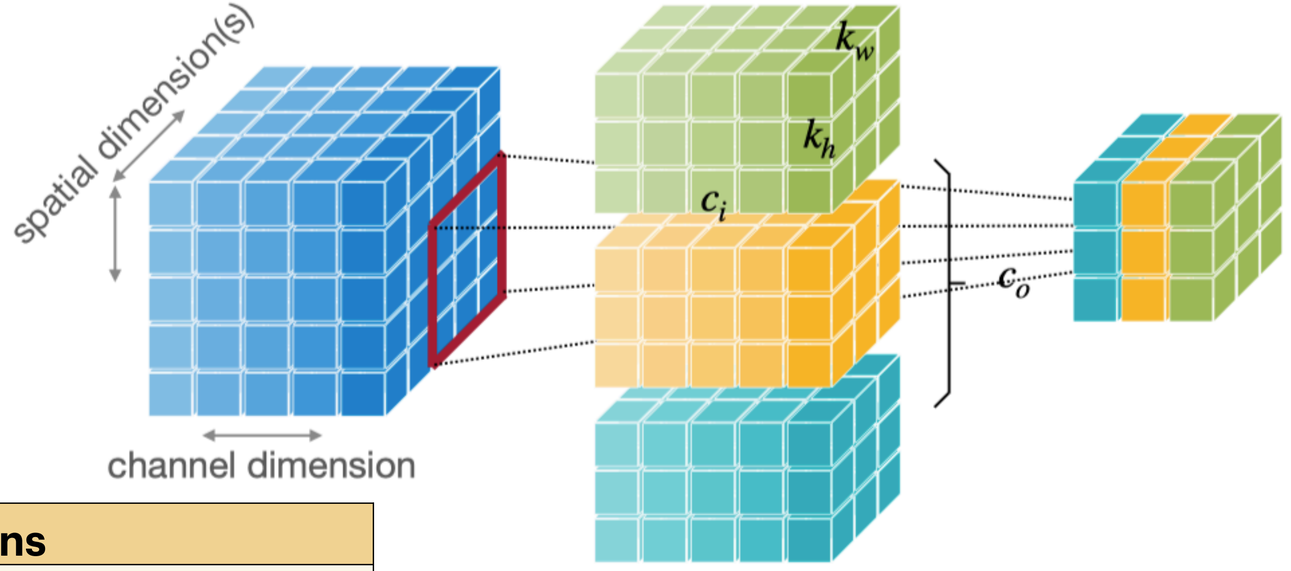


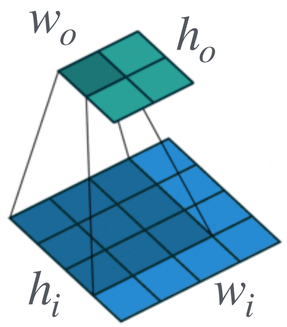
Notations	
c_i, c_o	Input/output
g	Groups
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width

Convolution Layer (2D Conv)

The output neuron is connected to input neurons in the receptive field

- Shape of tensors
  - Input features $X : (n, c_i, h_i, w_i)$
  - Output features $Y : (n, c_o, h_o, w_o)$
  - Weights $W : (c_o, c_i, k_h, k_w)$
  - Bias $b : (c_o,)$



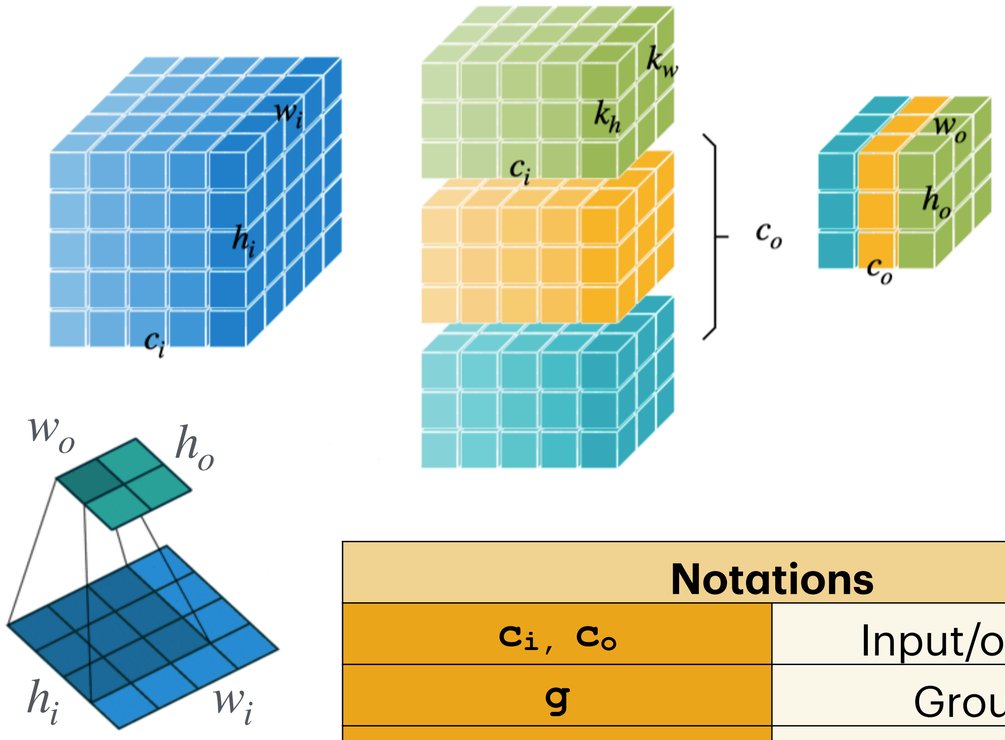


Notations	
c_i	Input channels
c_o	Output channels
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width

Parameters

Parameter (synapse/weight) count of the given neural network, i.e., # elements in the weight tensors

Layer	#Parameters (Bias is ignored)
Linear Layer	$c_o \cdot c_i$
Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w$

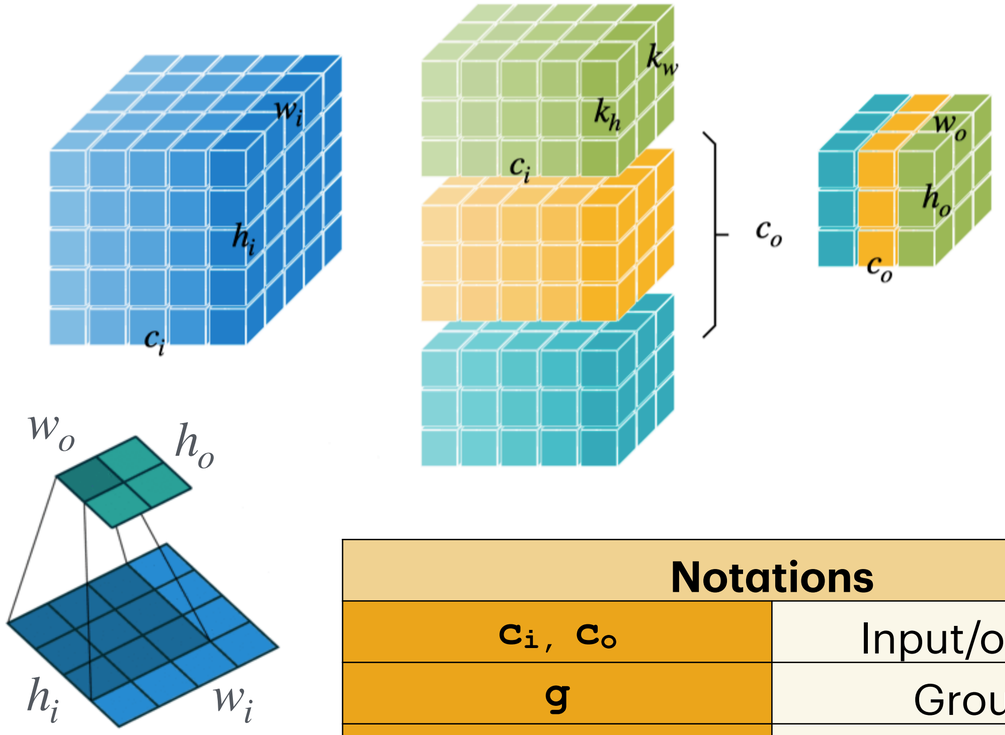


Notations	
c_i, c_o	Input/output
g	Groups
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width

Multiply-Accumulate Operations

MAC

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

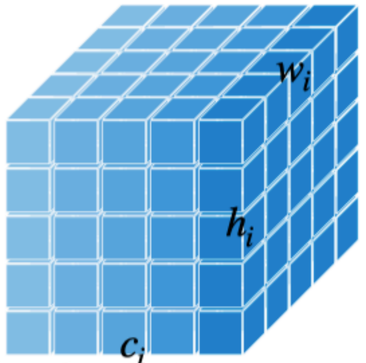


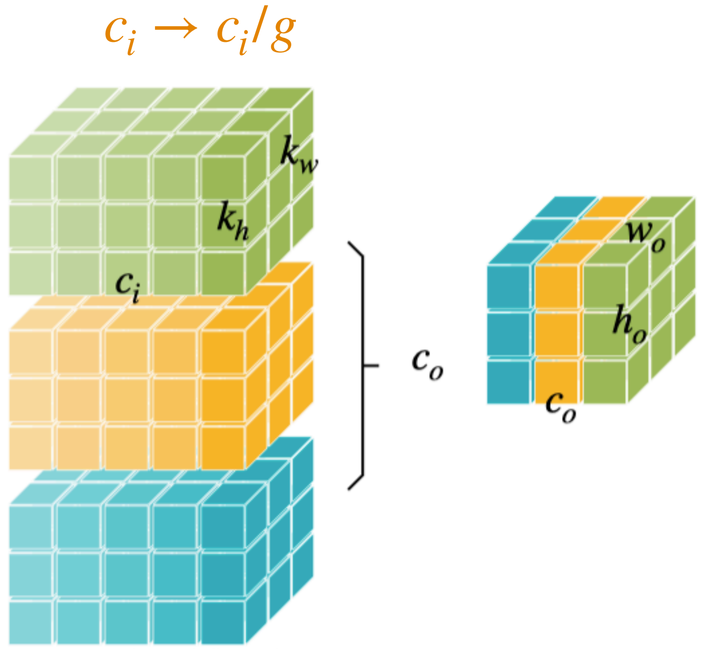
Notations	
c_i, c_o	Input/output
g	Groups
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width

Grouped Convolution Layer

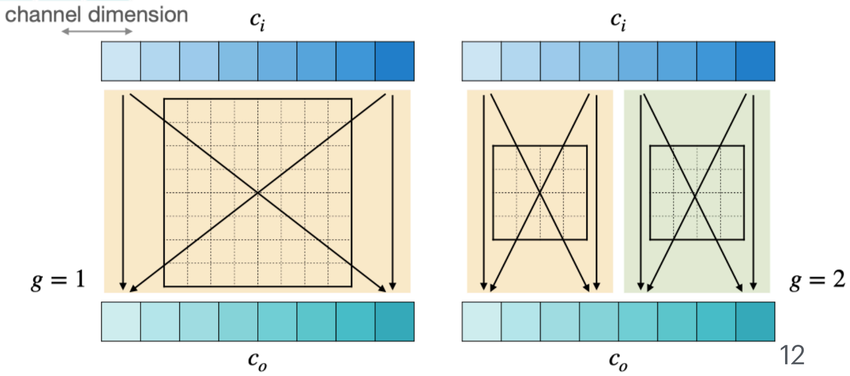
A group of narrower convolutions

- Shape of tensors
  - Input features $X : (n, c_i, h_i, w_i)$
  - Output features $Y : (n, c_o, h_o, w_o)$
  - Weights $W : \sout{(c_o, c_i, k_h, k_w)}\ (c_o/g \cdot g, c_i/g, k_h, k_w)$
  - Bias $b : (c_o,)$





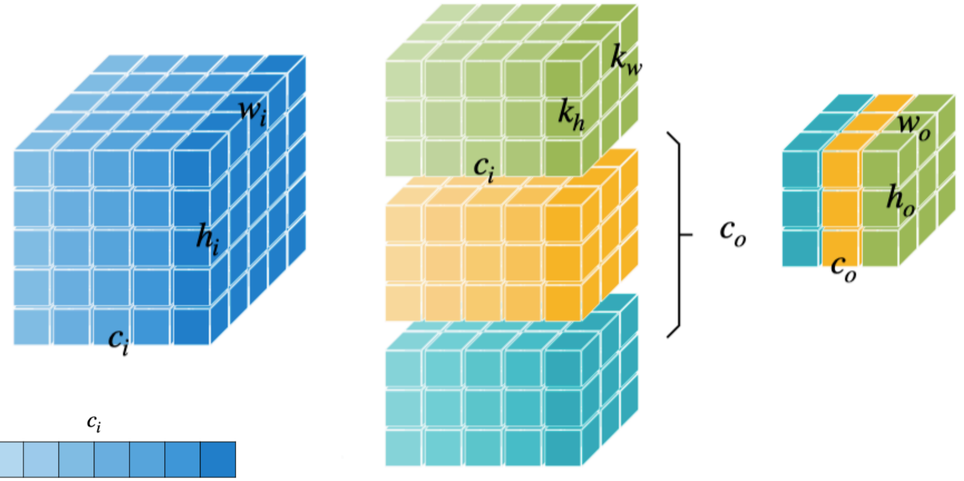
Notations	
c_i, c_o	Input/output
g	Groups
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width

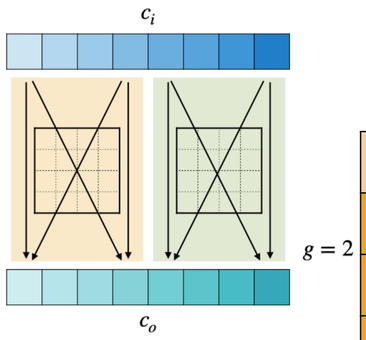


Parameters

Parameter (synapse/weight) count of the given neural network, i.e., # elements in the weight tensors

Layer	#Parameters (Bias is ignored)
Linear Layer	$c_o \cdot c_i$
Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w$
Grouped Convolution	$c_o \cdot c_i/g \cdot k_h \cdot k_w$ $= c_o \cdot c_i \cdot k_h \cdot k_w/g$



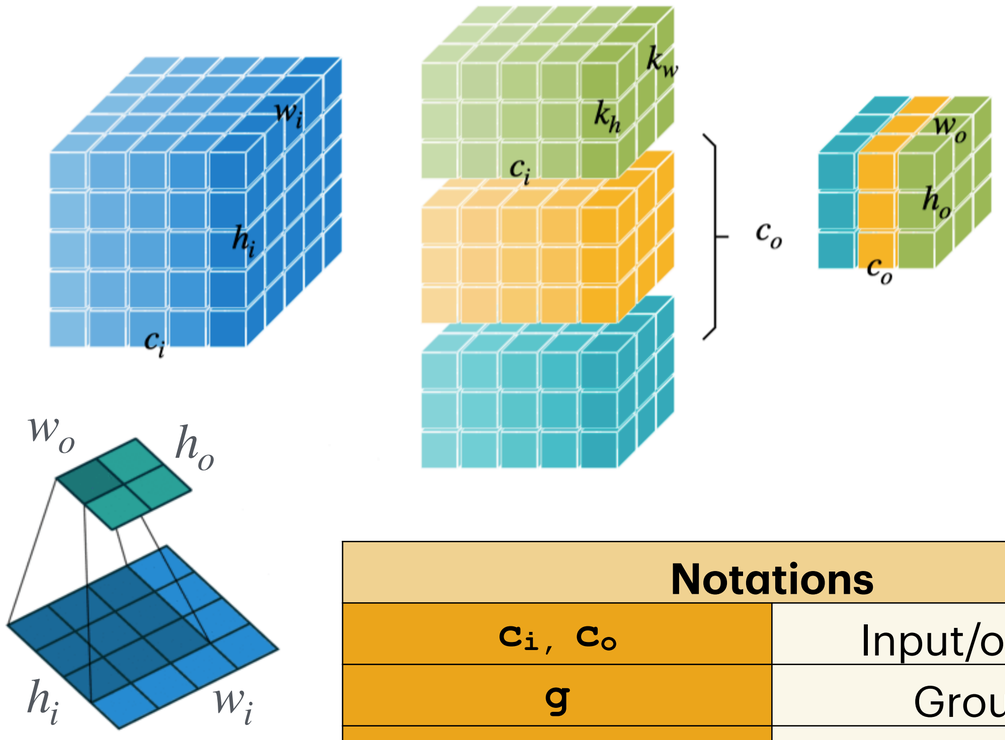


Notations	
c_i, c_o	Input/output
g	Groups
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width

Multiply-Accumulate Operations

MAC

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



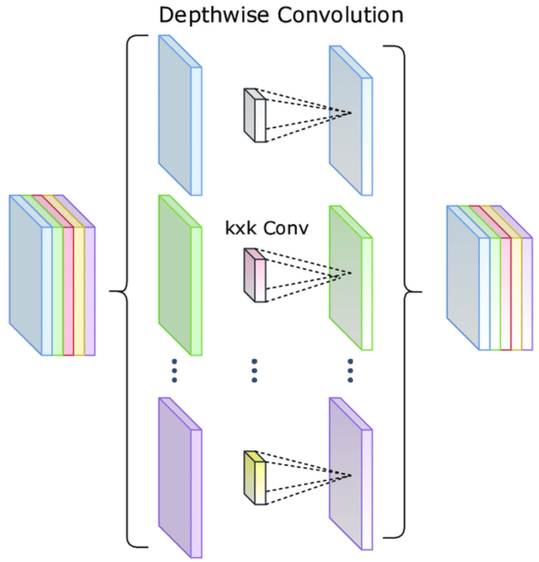
Notations	
c_i, c_o	Input/output
g	Groups
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width

Depthwise Convolution Layer

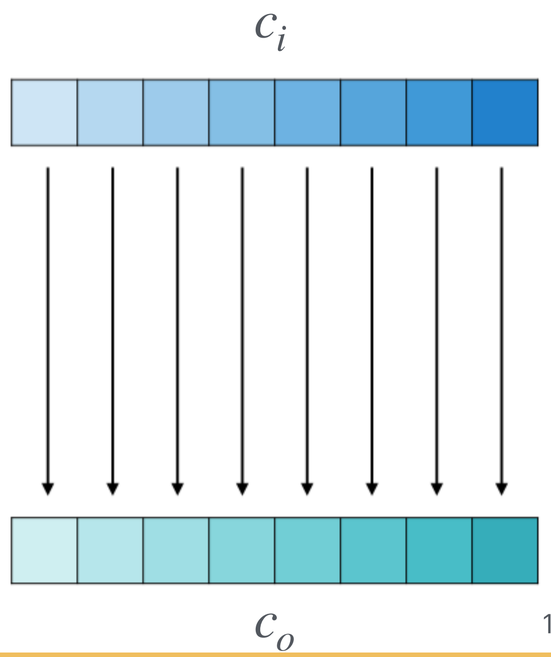
Independent filter for each channel: $g = c_i = c_o$ in grouped convolution

- Shape of tensors
  - Input features $X : (n, c_i, h_i, w_i)$
  - Output features $Y : (n, c_o, h_o, w_o)$
  - Weights $W : ~~(c_o/g \cdot g, c_i/g, k_h, k_w)~~ \; (c_o, 1, k_h, k_w)$
  - Bias $b : (c_o,)$

Notations	
c_i, c_o	Input/output
g	Groups
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width



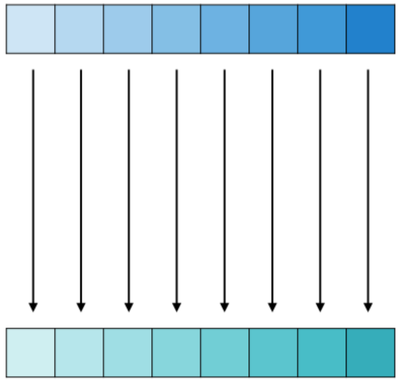
Source

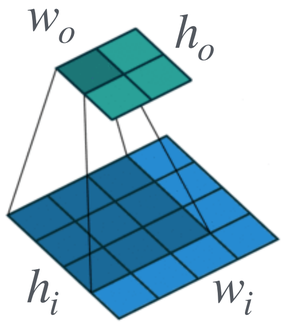


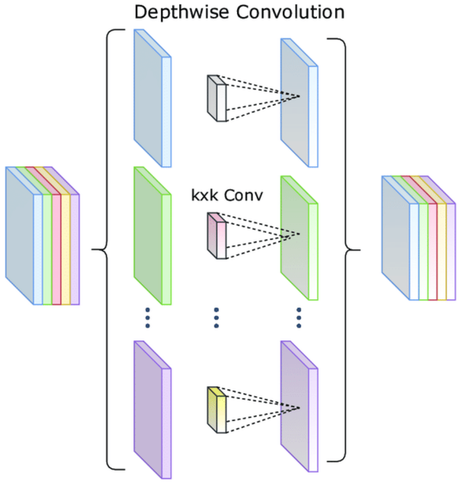
Parameters

Parameter (synapse/weight) count of the given neural network, i.e., # elements in the weight tensors

Layer	#Parameters (Bias is ignored)
Linear Layer	$c_o \cdot c_i$
Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w$
Grouped Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w / g$
Depthwise Convolution	$c_o \cdot k_h \cdot k_w$







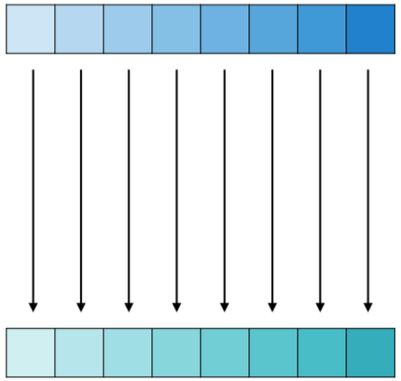
Source

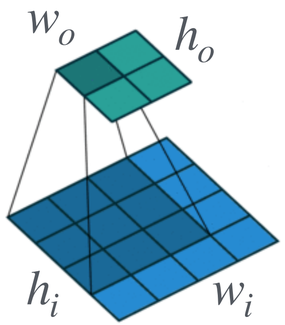
Notations	
c_i, c_o	Input/output
g	Groups
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width

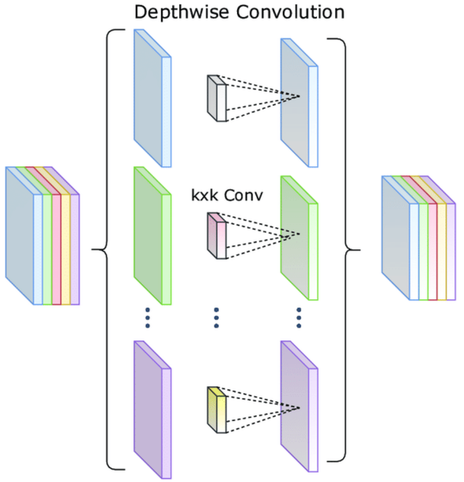
Multiply-Accumulate Operations

MAC

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



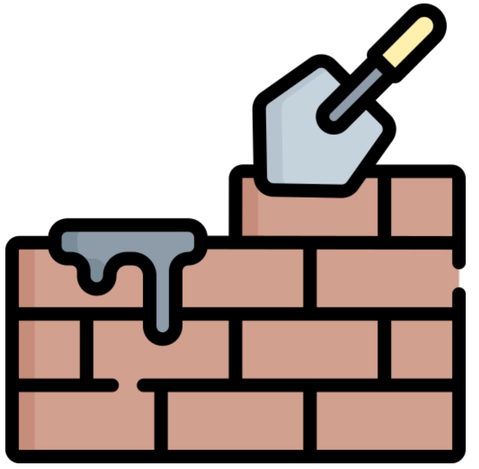




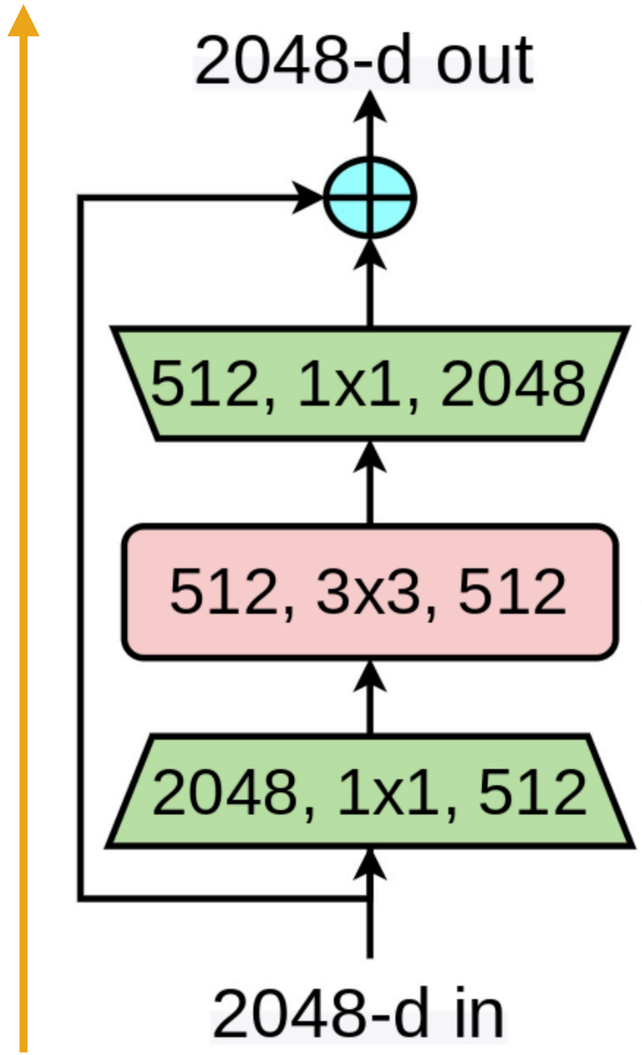
Source

Notations	
c_i, c_o	Input/output
g	Groups
n	Batch size
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_h, k_w	Kernel height/width

Classic Manual-Designed Building Blocks



ResNet50: BottleNeck Block



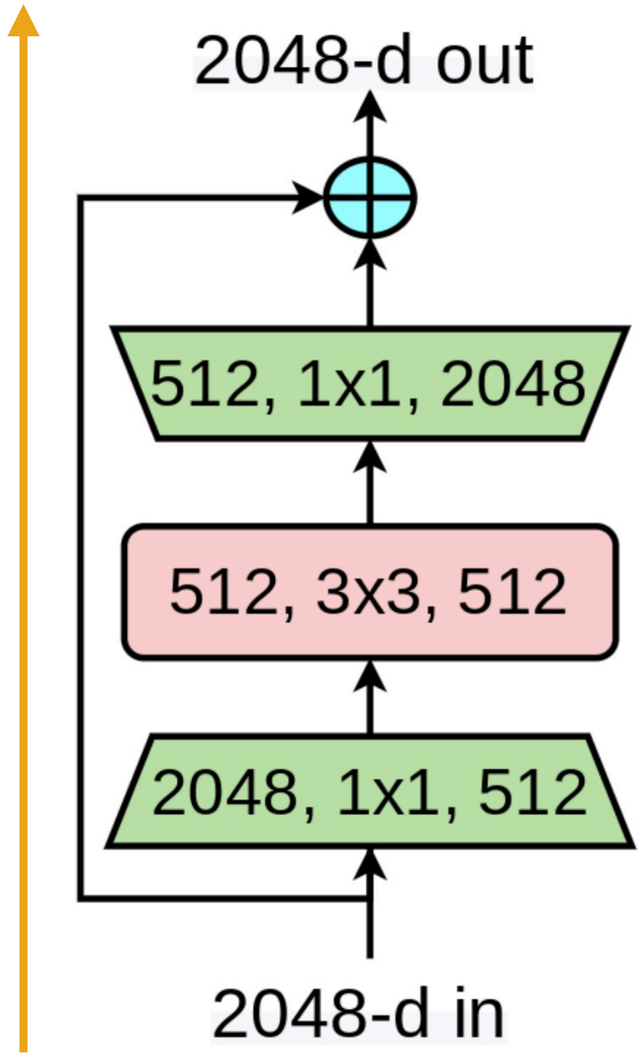
1x1 convolution: expand the #channels

Feed reduced feature map to 3x3 convolution

1x1 convolution: reduce the #channels by 4X

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
Srinivas, A., Lin, T. Y., Parmar, N., Shlens, J., Abbeel, P., & Vaswani, A. (2021). Bottleneck transformers for visual recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 16519-16529).

ResNet50: Bottleneck Block



#MACs

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

🤔 **What if there is only one 3x3 convolution layer?**

$$2048 \times 512 \times H \times W \times 1$$

$$512 \times 3 \times 3 \times 512 \times H \times W$$

$$2048 \times 512 \times H \times W \times 1$$

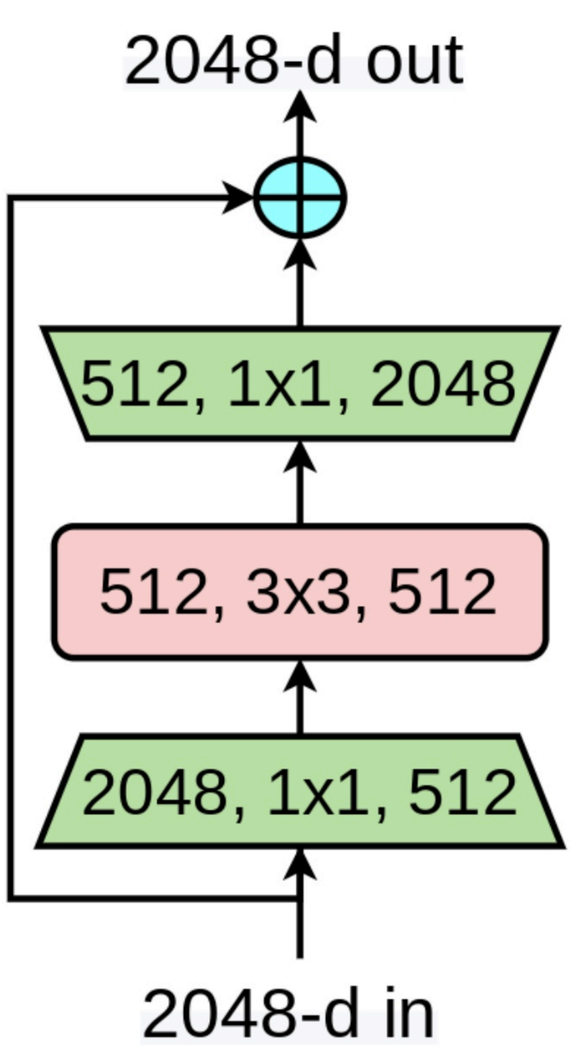
$$2048 \times 3 \times 3 \times 2048 \times H \times W$$
$$= 512 \times 512 \times 144 \times H \times W$$

$\approx 8.5X$ reduction

$$512 \times 512 \times 17 \times H \times W$$

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
Srinivas, A., Lin, T. Y., Parmar, N., Shlens, J., Abbeel, P., & Vaswani, A. (2021). Bottleneck transformers for visual recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 16519-16529).

ResNet50: Bottleneck Block



#Parameters

Layer	#Parameters
Linear Layer	$c_o \cdot c_i$
Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w$
Grouped Convolution	$c_o \cdot c_i \cdot k_h \cdot k_w / g$
Depthwise Convolution	$c_o \cdot k_h \cdot k_w$

🤔 **What if there is only one 3x3 convolution layer?**

$$2048 \times 512 \times 1$$

$$512 \times 3 \times 3 \times 512$$

$$2048 \times 512 \times 1$$

$$2048 \times 3 \times 3 \times 2048$$
$$= 512 \times 512 \times 144$$

$\approx 8.5X$ reduction

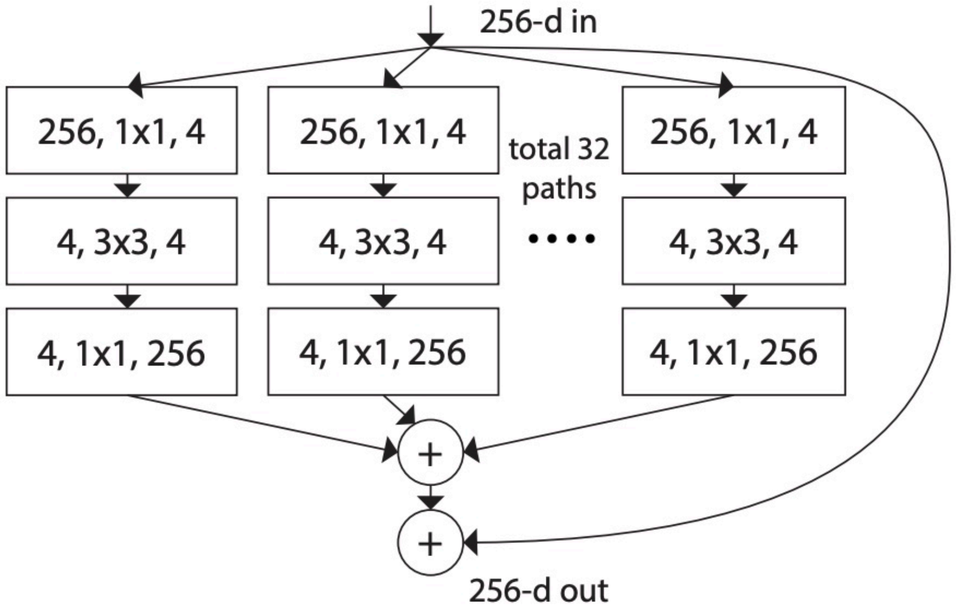
$$512 \times 512 \times 17$$

#MACs, #Params are the first order of approximation of the model efficiency.

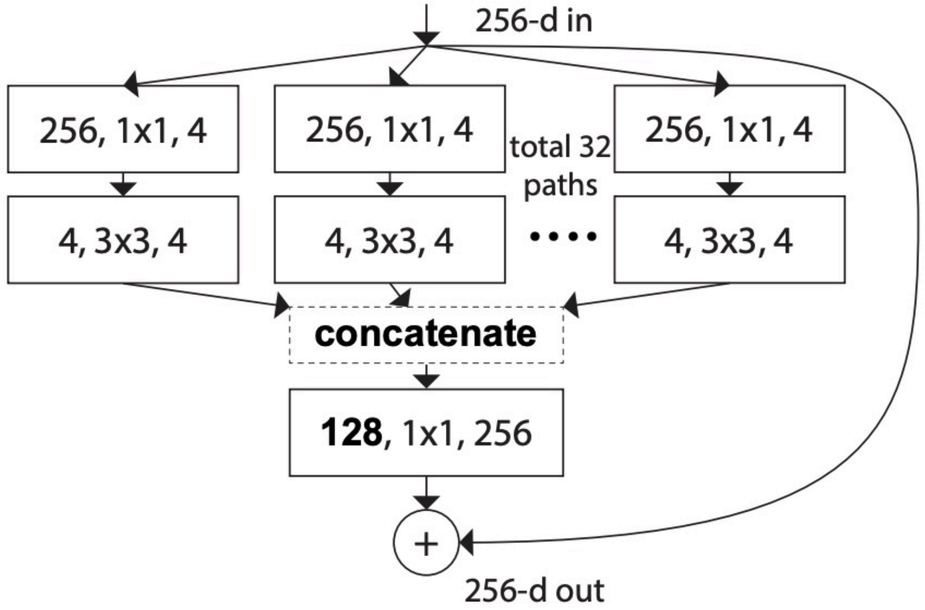
He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
Srinivas, A., Lin, T. Y., Parmar, N., Shlens, J., Abbeel, P., & Vaswani, A. (2021). Bottleneck transformers for visual recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 16519-16529).

ResNeXt: Grouped Convolution

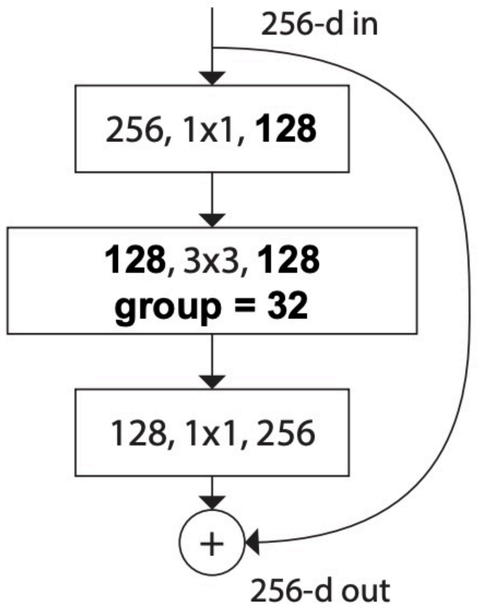
Replace 3x3 convolution with 3x3 grouped convolution



(a) Aggregated residual transformations



(b) Equivalent to (a), implemented as early concatenation

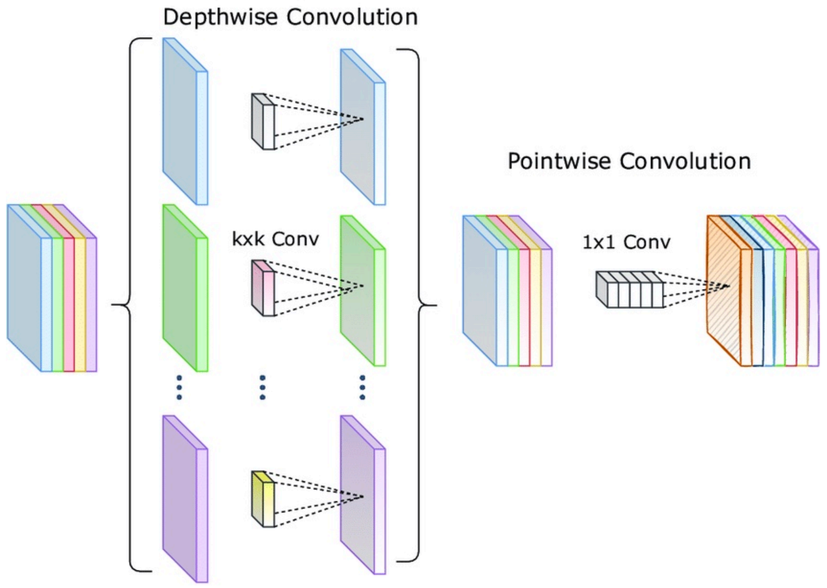


(c) Equivalent to (a,b), implemented as group convolutions

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500).

MobileNet: Depthwise-separable Block

- Depthwise convolution is an extreme case of group convolution ($c_o = c_i = g$)
- Use depthwise convolution to **capture spatial information**
- Use pointwise convolution (1x1 convolution) to **fuse/exchange information** across different channels



Source

Depthwise conv (Filter size=$k_h \times k_w \times 1 \times C_o$)

🤩 Low computational cost and fewer parameters

😣 Low capacity because it does not combine information across channels

Pointwise conv (Filter size=$1 \times 1 \times C_i \times C_o$)

🤩 Very efficient, 1x1 convolution, combine cross-channel information

😣 No spatial interaction

Normal conv (Filter size=$k_h \times k_w \times C_i \times C_o$)

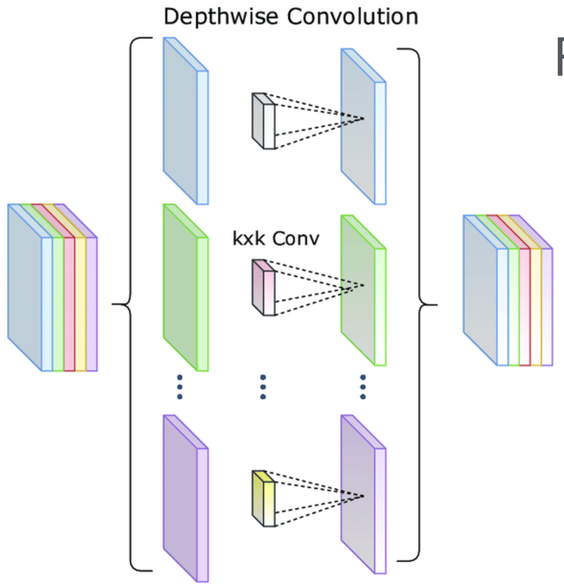
🤩 High capacity, capture both spatial and cross-channel information

😣 More parameters, high computational cost

Howard, A. G. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

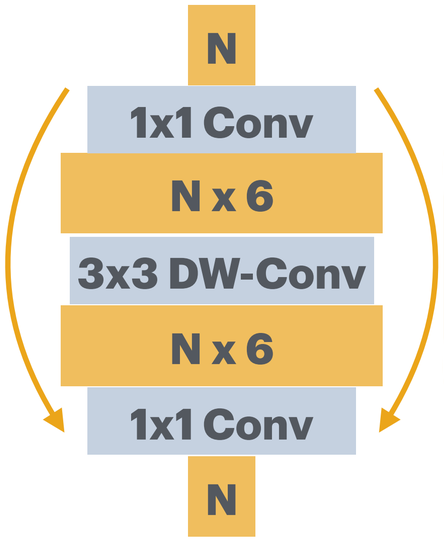
MobileNetV2: Inverted Bottleneck Block

- Problem: compared to normal convolution, depthwise convolution has **a much lower capacity**
- Solution: **Increase the depthwise convolution's input and output channels** to capture more features independently
  - Depthwise convolution's cost only grows linearly, so the cost is still affordable.
  - # MACs = $k_h \cdot k_w \cdot c_o \cdot h_o \cdot w_o$, #Params = $c_o \cdot k_h \cdot k_w$



Source

Feature Channels



#MACs (N=160)

$$160 \times (160 \times 6) \times H \times W \times 1$$

$$(160 \times 6) \times 3 \times 3 \times H \times W$$

$$160 \times (160 \times 6) \times H \times W \times 1$$

🤔 **What if there is only one 3x3 convolution layer?**

$$160 \times 160 \times H \times W \times 3 \times 3$$
$$= 960 \times H \times W \times 240$$

$1 : 1.37$

$$960 \times H \times W \times 329$$

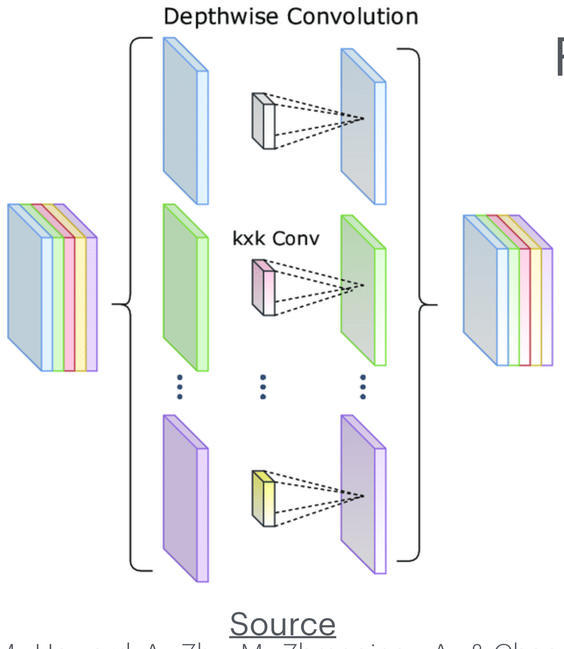
Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

MobileNetV2: Inverted Bottleneck Block

- Problem: compared to normal convolution, depthwise convolution has **a much lower capacity**
- Solution: **Increase the depthwise convolution's input and output channels** to improve its capacity

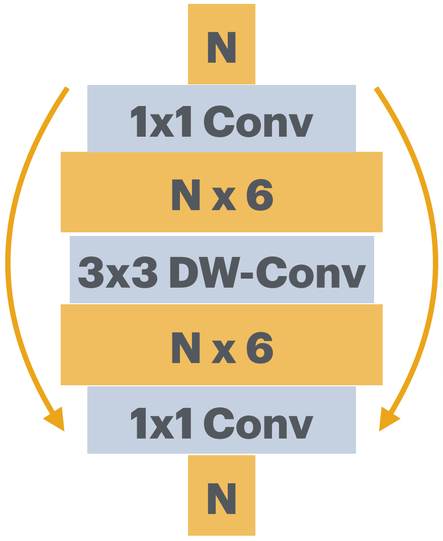
Widely used back to 2016-ish.

  - Depthwise convolution's cost only grows linearly, so the cost is still affordable.
  - # MACs = $k_h \cdot k_w \cdot c_o \cdot h_o \cdot w_o$, #Params = $c_o \cdot k_h \cdot k_w$



Source

Feature Channels



#Params (N=160)

$160 \times (160 \times 6) \times 1$

$(160 \times 6) \times 3 \times 3$

$160 \times (160 \times 6) \times 1$

🤔 **What if there is only one 3x3 convolution layer?**

$160 \times 160 \times 3 \times 3$

$= 960 \times 240$

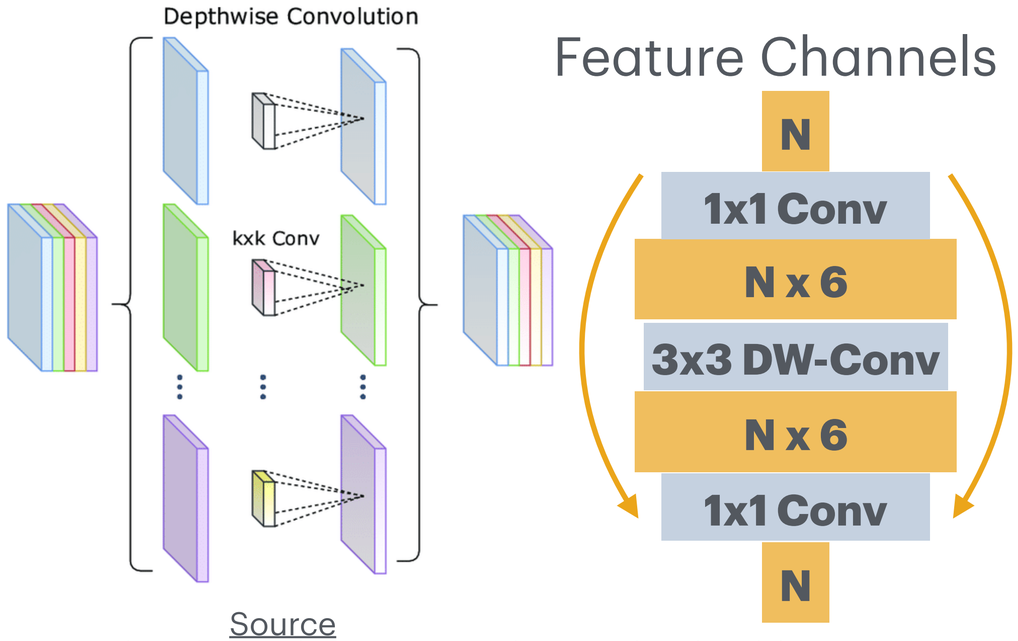
$1 : 1.37$

960×329

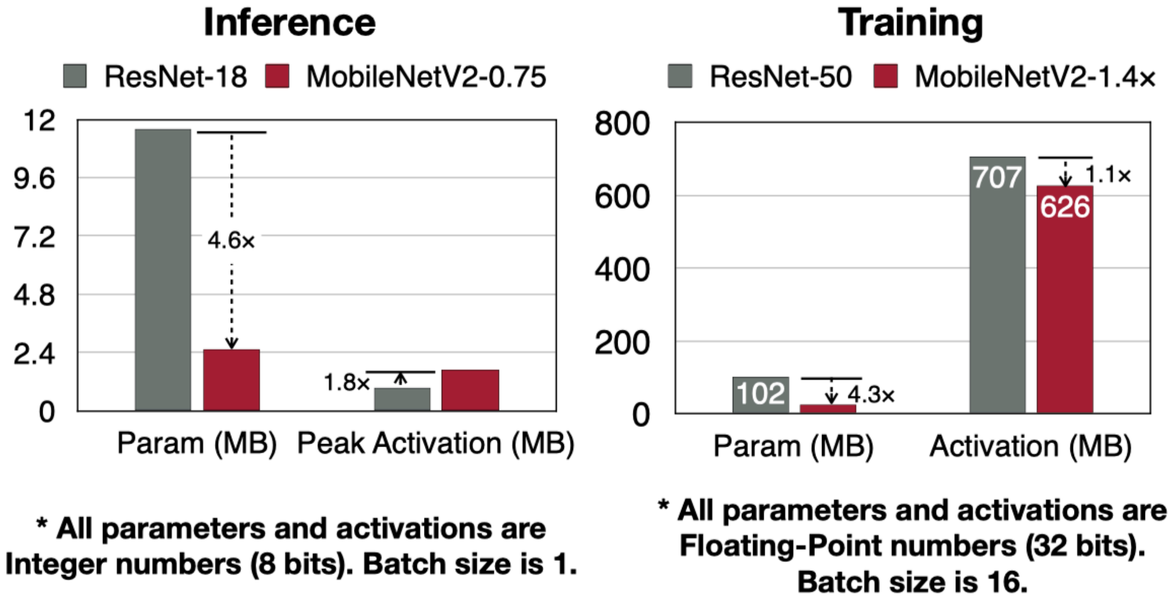
Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

MobileNetV2: Inverted Bottleneck Block

- Problem: compared to normal convolution, depthwise convolution has a much lower capacity
- Solution: Increase the depthwise convolution's input and output channels to improve its capacity
  - Depthwise convolution's cost only grows linearly, so the cost is still affordable.
  - However, this design is not memory-efficient for both inference and training



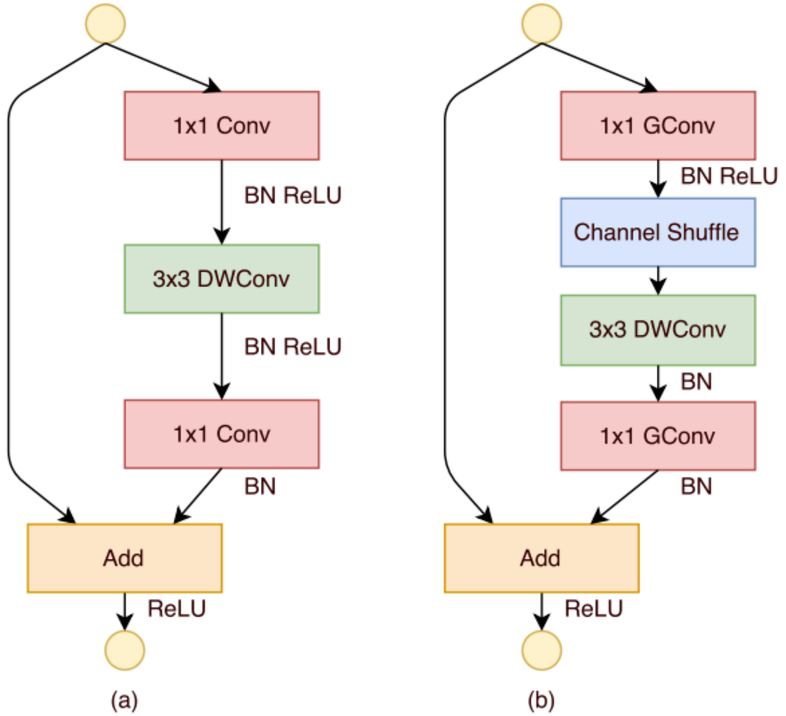
Source



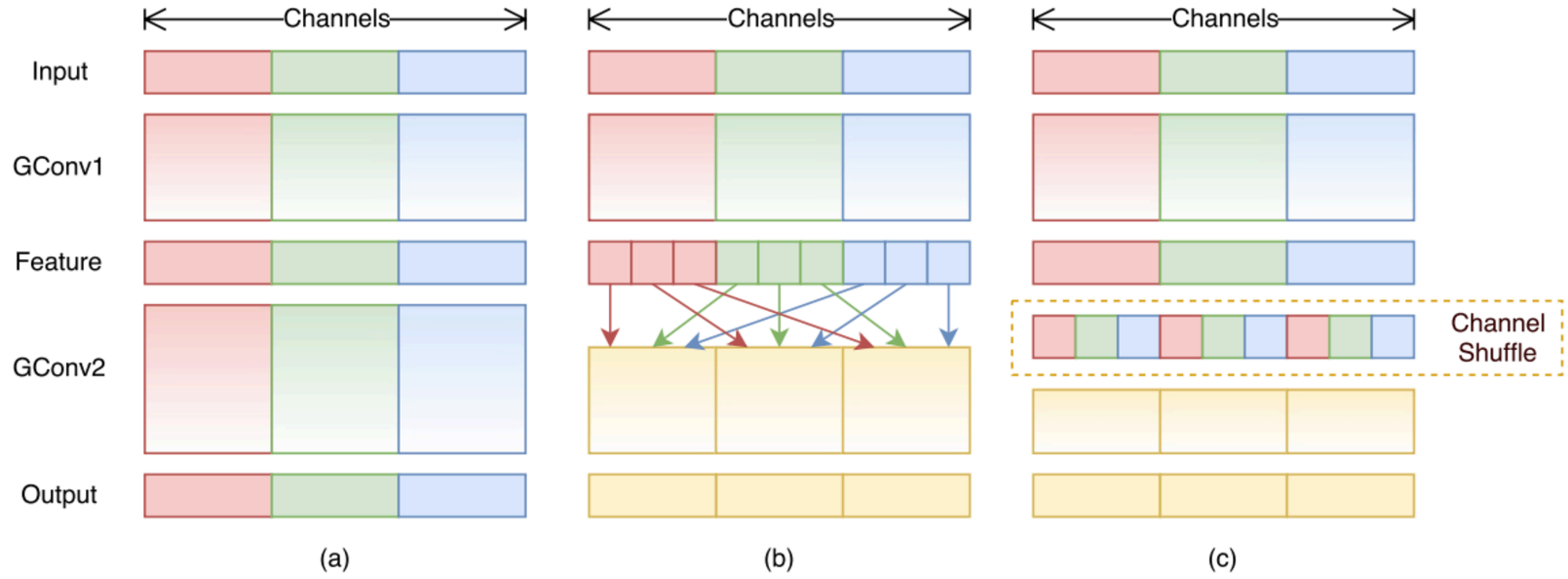
Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

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



Bottleneck unit w/ DWConv vs. ShuffleNet Units w/ pointwise GConv & channel shuffle



Channel shuffle with two stacked group convolutions

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6848-6856).

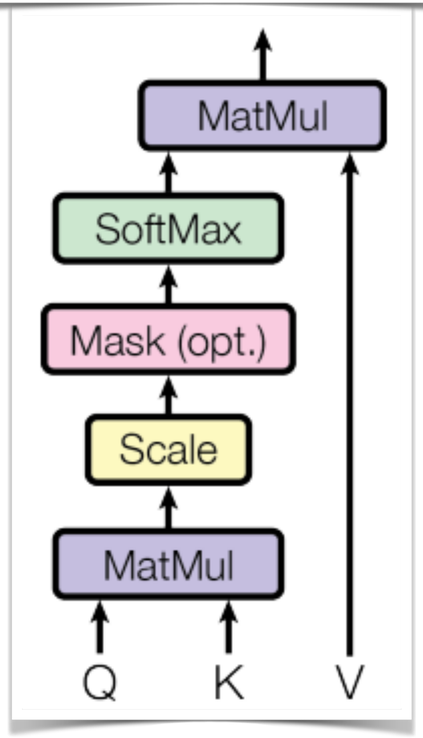
Transformer: Multi-Head Self-Attention

MHSA

- **Project** Q, K, and V with h **different**, learned linear projections
- Perform the scaled dot-product attention function on each of these projected versions of Q, K, and V in parallel
- Concatenate the output values
- Project the output values again, resulting in the final values

$$\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



Scaled dot-product attention

Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

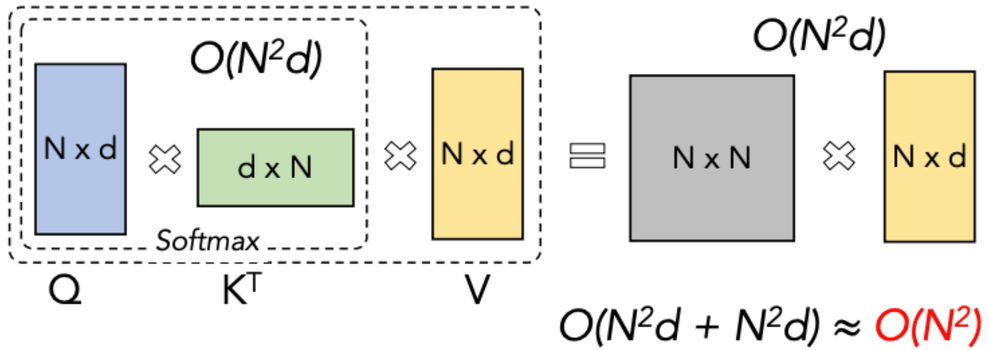
Transformer: Multi-Head Self-Attention

MHSA

- Project Q, K, and V with h different, learned linear projections
- Perform the **scaled dot-product attention** function on each of these projected versions of Q, K, and V **in parallel**
- Concatenate the output values
- Project the output values again, resulting in the final values

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Computations for vanilla self attention

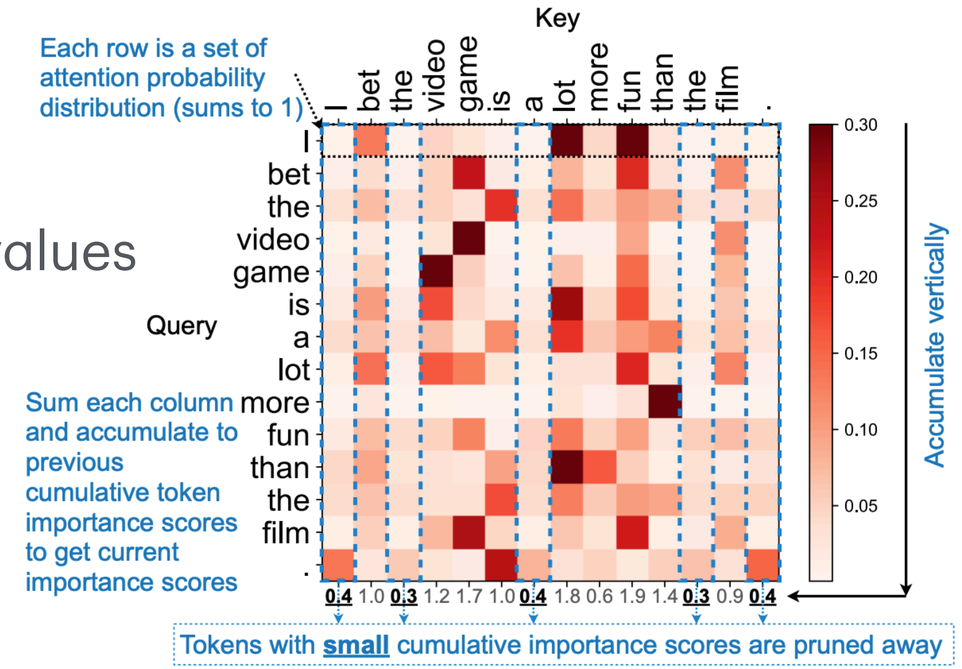
Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

Transformer: Multi-Head Self-Attention

MHSA

- Project Q, K, and V with h different, learned linear projections
- Perform the scaled dot-product attention function on each of these projected versions of Q, K, and V in parallel
- Concatenate the output values
- Project the output values again, resulting in the final values

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



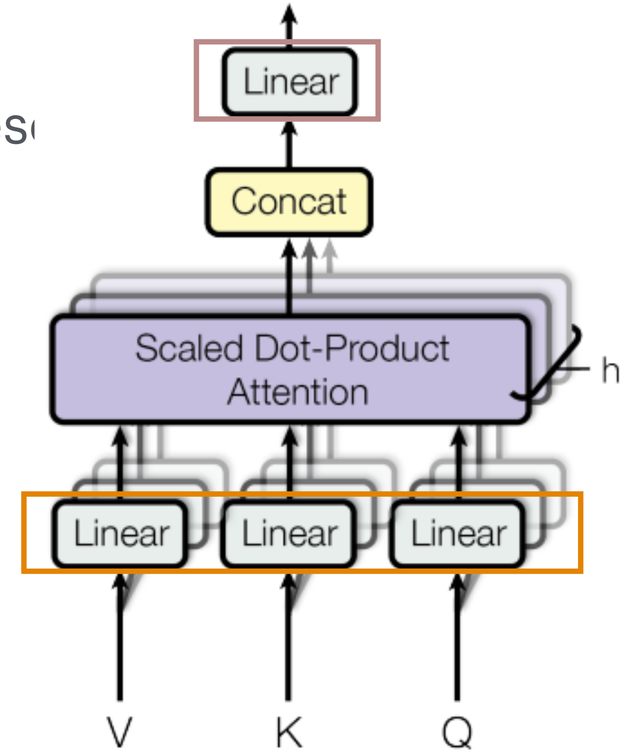
Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

Transformer: Multi-Head Self-Attention

MHSA

- Project Q, K, and V with h different, learned linear projections
- Perform the scaled dot-product attention function on each of thes projected versions of Q, K, and V in parallel
- **Concatenate** the output values
- **Project** the output values again, resulting in the final values

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$
$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$



Attention mechanism

Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.

What is Neural Architecture Search?

From Manual Design to Automatic Design

Huge design space, manual design is unscalable

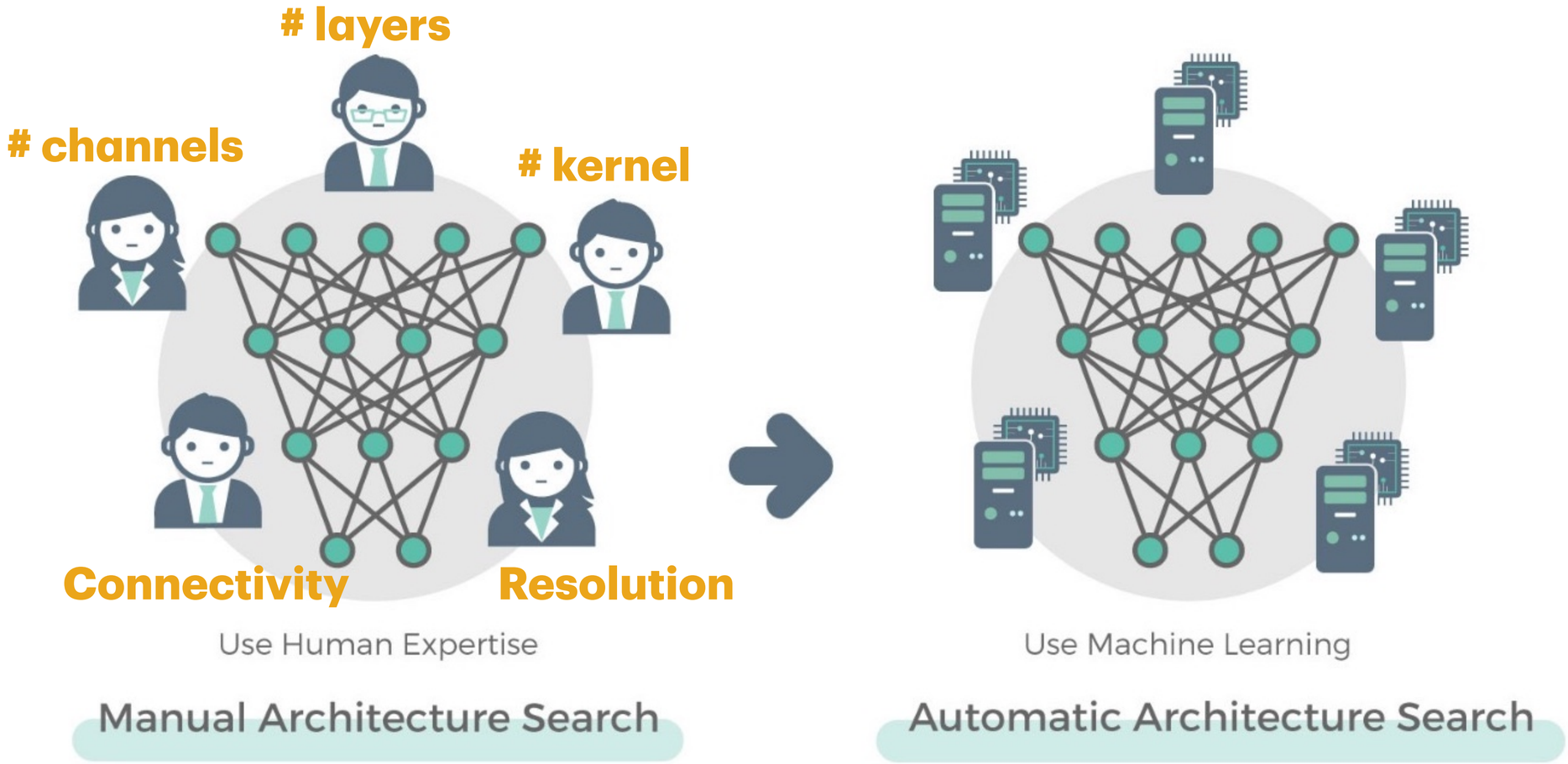
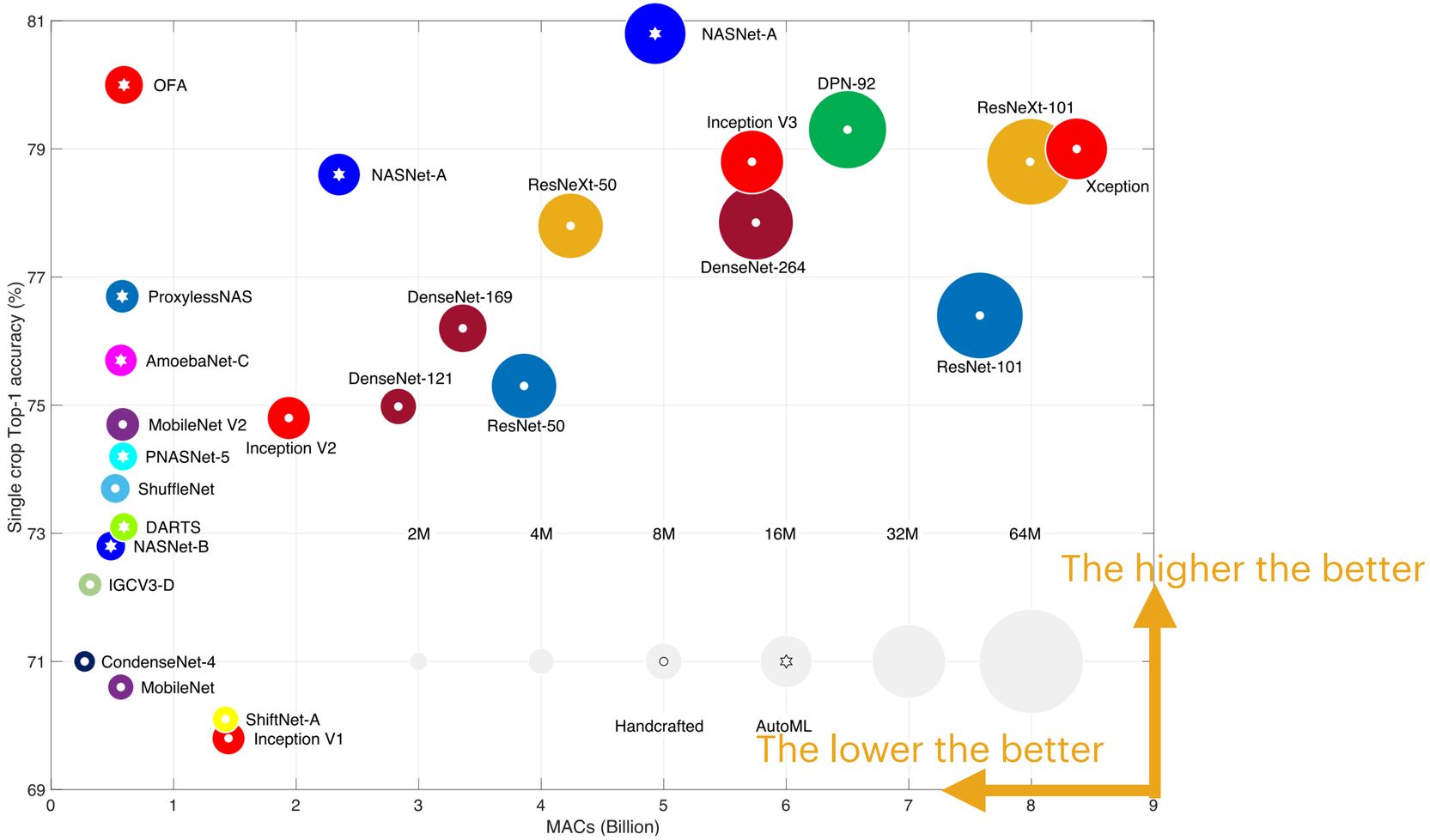


Figure source

From Manual Design to Automatic Design

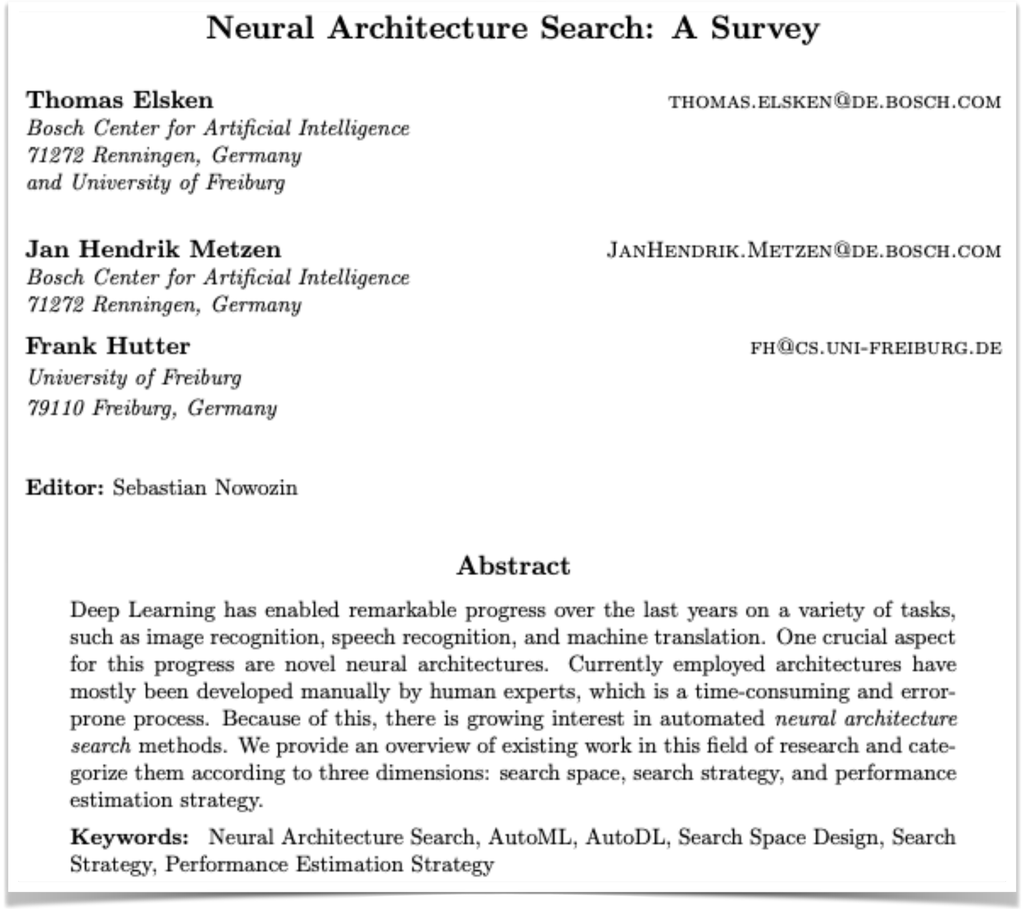
Neural architecture search reduces the computational cost



Deng, L., Li, G., Han, S., Shi, L., & Xie, Y. (2020). Model compression and hardware acceleration for neural networks: A comprehensive survey. Proceedings of the IEEE, 108(4), 485-532.

Framework of Neural Architecture Search

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

Thomas Elsken THOMAS.ELSKEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany
and University of Freiburg

Jan Hendrik Metzen JANHENDRIK.METZEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany

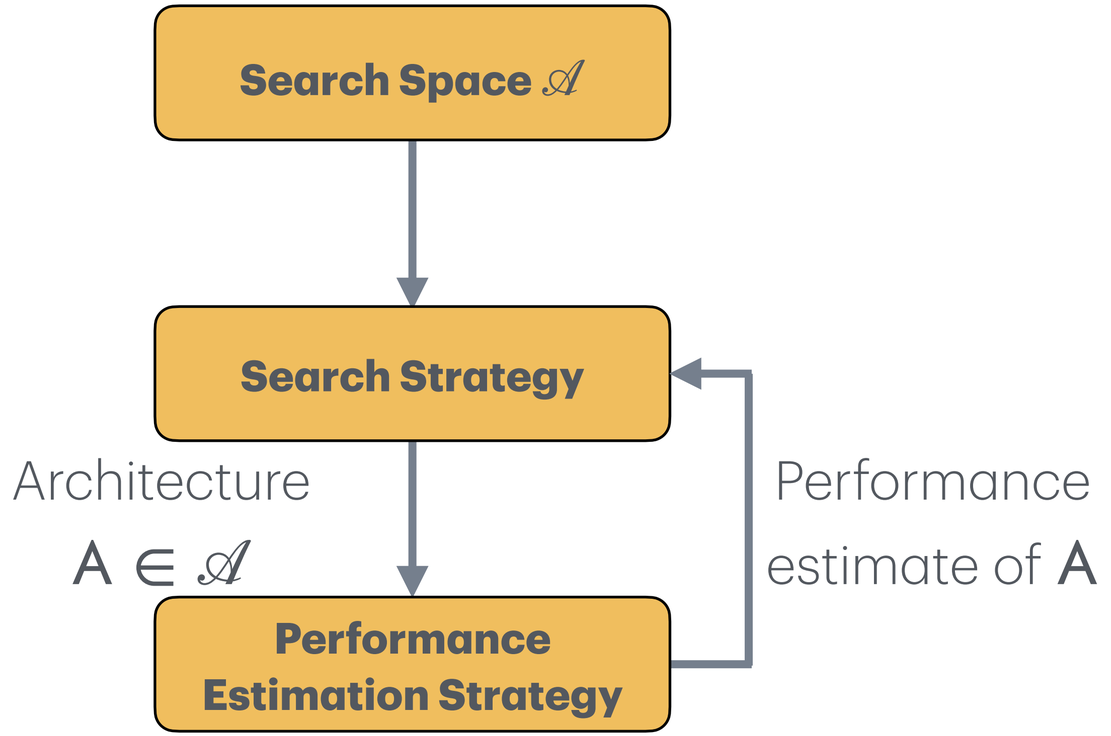
Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg
79110 Freiburg, Germany

Editor: Sebastian Nowozin

Abstract

Deep Learning has enabled remarkable progress over the last years on a variety of tasks, such as image recognition, speech recognition, and machine translation. One crucial aspect for this progress are novel neural architectures. Currently employed architectures have mostly been developed manually by human experts, which is a time-consuming and error-prone process. Because of this, there is growing interest in automated *neural architecture search* methods. We provide an overview of existing work in this field of research and categorize them according to three dimensions: search space, search strategy, and performance estimation strategy.

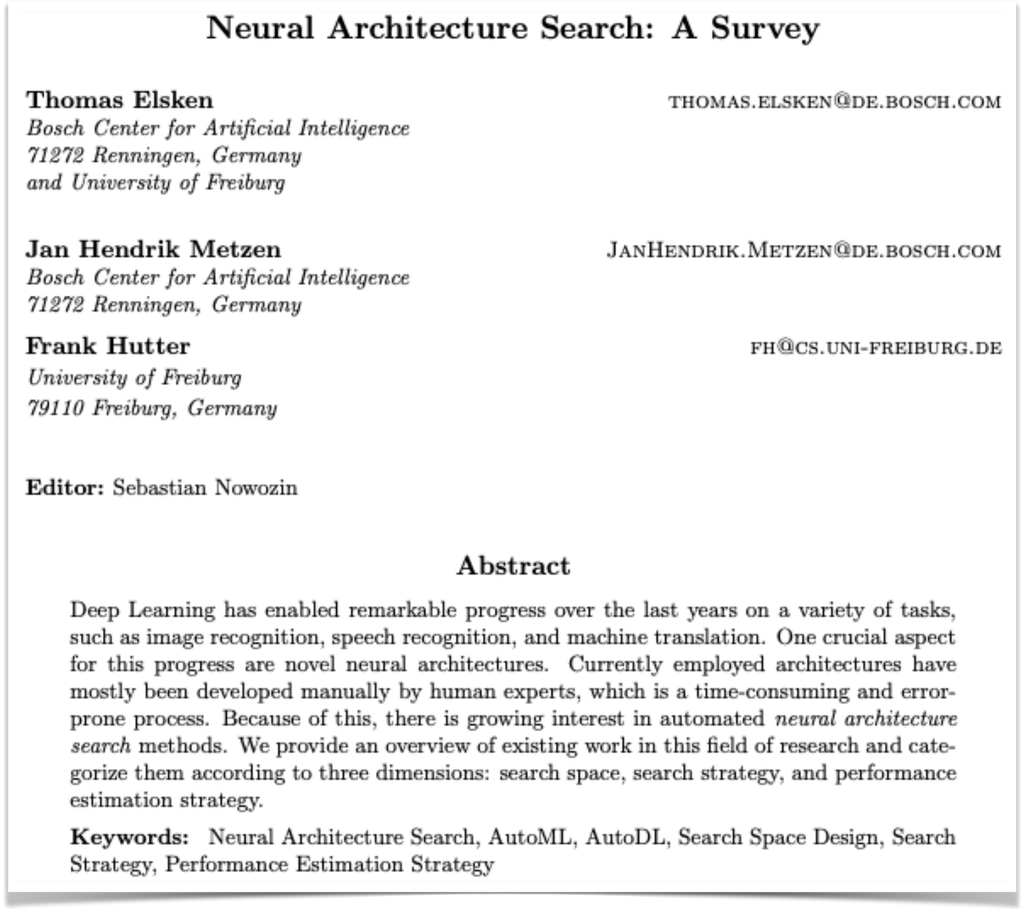
Keywords: Neural Architecture Search, AutoML, AutoDL, Search Space Design, Search Strategy, Performance Estimation Strategy



Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.

Framework of Neural Architecture Search

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

Thomas Elsken THOMAS.ELSKEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany
and University of Freiburg

Jan Hendrik Metzen JANHENDRIK.METZEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany

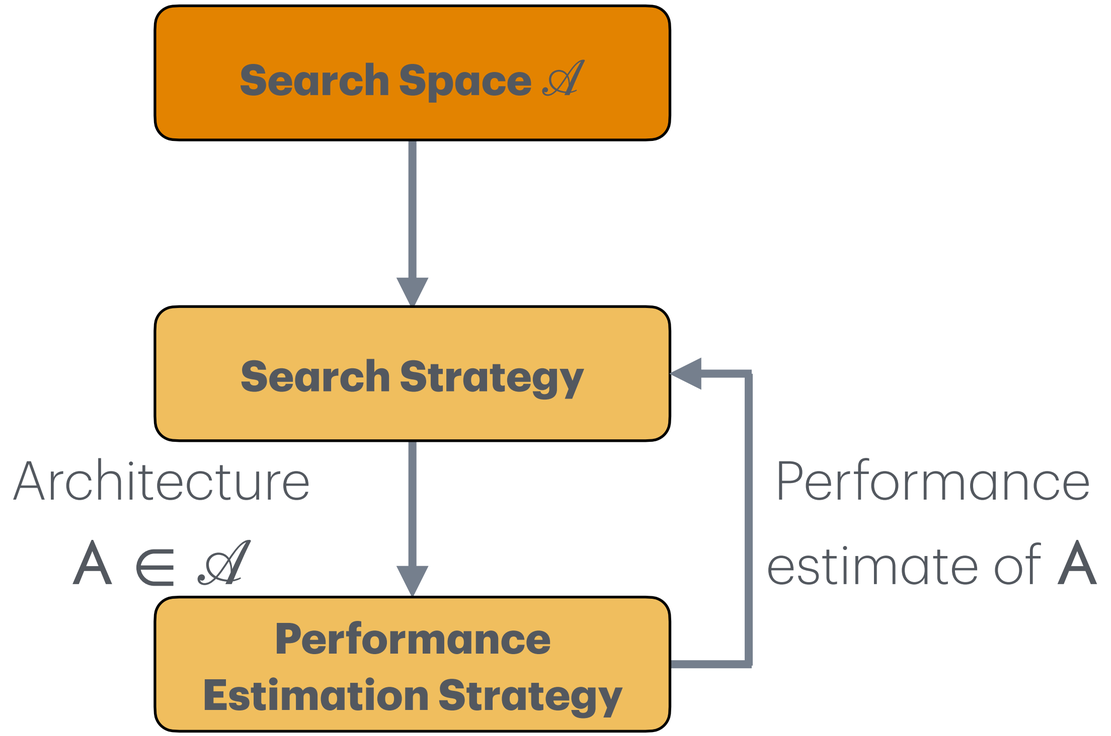
Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg
79110 Freiburg, Germany

Editor: Sebastian Nowozin

Abstract

Deep Learning has enabled remarkable progress over the last years on a variety of tasks, such as image recognition, speech recognition, and machine translation. One crucial aspect for this progress are novel neural architectures. Currently employed architectures have mostly been developed manually by human experts, which is a time-consuming and error-prone process. Because of this, there is growing interest in automated *neural architecture search* methods. We provide an overview of existing work in this field of research and categorize them according to three dimensions: search space, search strategy, and performance estimation strategy.

Keywords: Neural Architecture Search, AutoML, AutoDL, Search Space Design, Search Strategy, Performance Estimation Strategy



Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.

Search Space in NAS

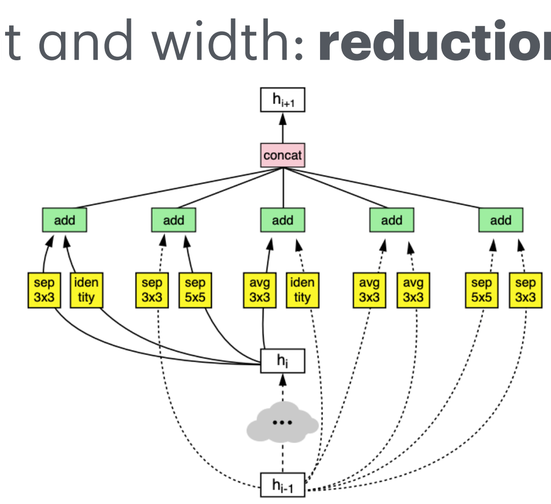
Search space is a set of candidate neural network architectures

- **Cell-level** search space
- **Network-level** search space
  - Depth dimension
  - Resolution dimension
  - Width dimension
  - Kernel size dimension
  - Topology connection
- Deisgn the search space

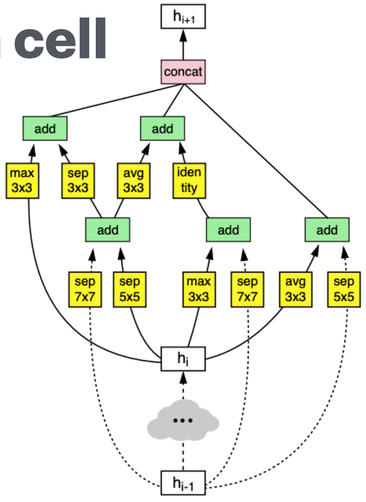
Cell-level Search Space: NASNet

- Search for the best convolutional layer (or "cell") on the CIFAR-10 dataset and then apply this cell to the ImageNet dataset by stacking together more copies of this cell, each with their own parameters to design a convolutional architecture.

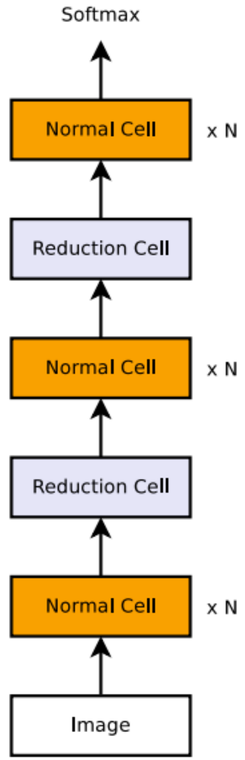
- Two types of convolutional cells that return a feature map
  - of the same dimension: **normal cell**
  - of half its original height and width: **reduction cell**



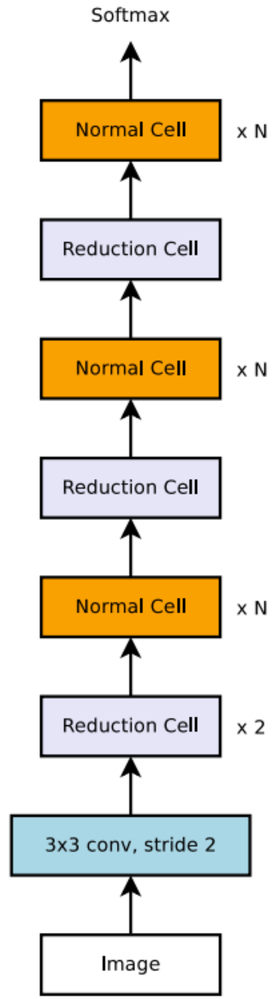
Normal Cell



Reduction Cell



CIFAR10 Architecture

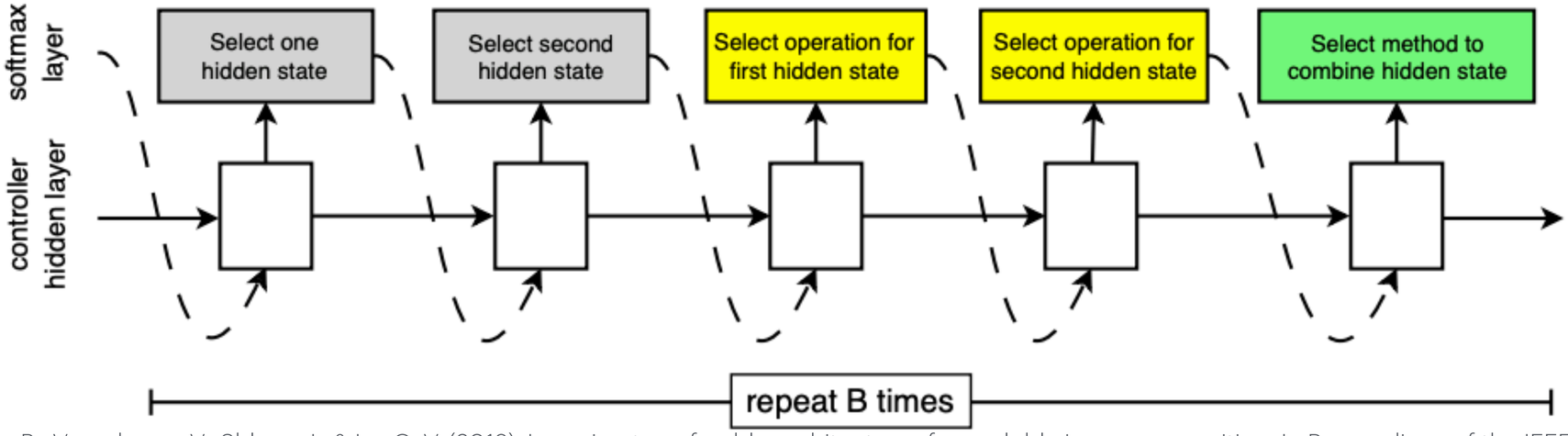


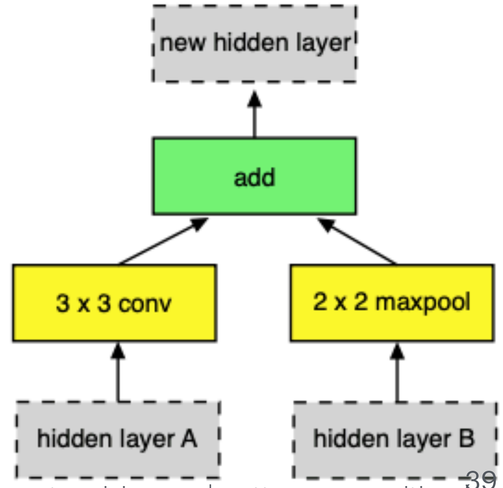
ImageNet Architecture

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8697-8710).

Cell-level Search Space: NASNet

- Controller model architecture for recursively constructing one block of a convolutional cell. Empirically, $B = 5$
- Left: **An RNN controller** generates the candidate cells by
  - Select two inputs from the set of hidden states
  - Select two input transformation operations
  - Select the method to combine the results
- Right: one example constructed block (cell)





Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8697-8710).

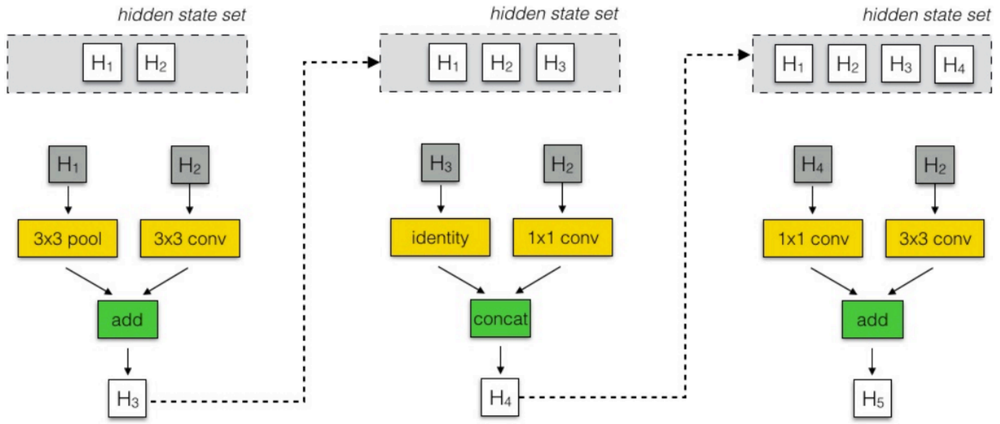
Cell-level Search Space: NASNet

Construction of one block

- Step 1: Select a hidden state from h_i, h_{i-1} or from the set of hidden states created in previous blocks
- Step 2: Select a second hidden state from the same options as in Step 1
- Step 3: Select an operation to apply to the hidden state selected in Step 1
- Step 4: Select an operation to apply to the hidden state selected in Step 2
- Step 5: select a method to combine the outputs of Step 3 and 4 to create a new hidden state
  - Either (1) element-wise addition betwen two hidden states
  - Or (2) concatenation along the filter dimension

In steps 3 and 4, the controller RNN selects an operation to apply to the hidden states. We collected the following set of operations based on their prevalence in the CNN literature:

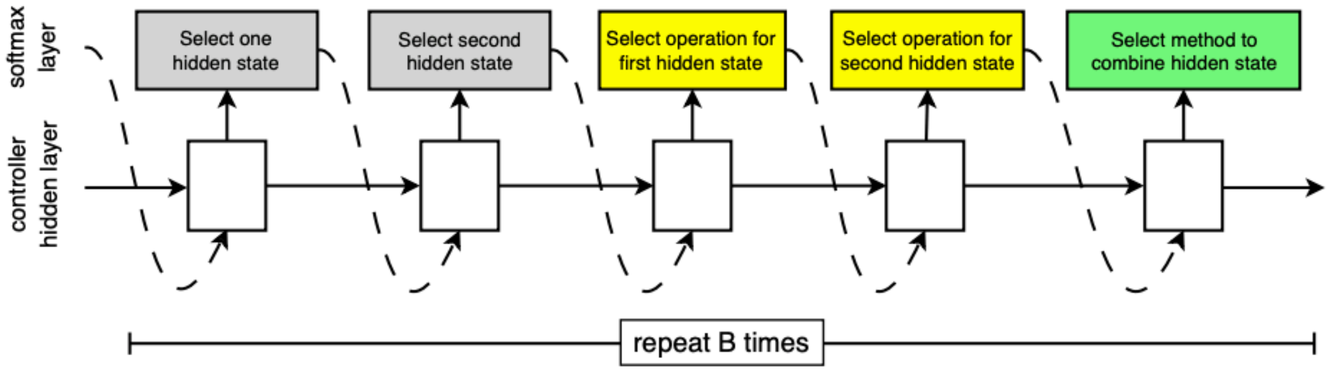
- identity
- 1x7 then 7x1 convolution
- 3x3 average pooling
- 5x5 max pooling
- 1x1 convolution
- 3x3 depthwise-separable conv
- 7x7 depthwise-separable conv
- 1x3 then 3x1 convolution
- 3x3 dilated convolution
- 3x3 max pooling
- 7x7 max pooling
- 3x3 convolution
- 5x5 depthwise-seperable conv



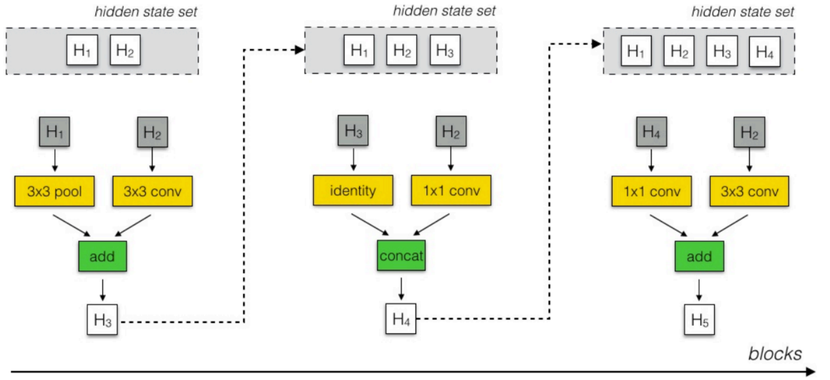
Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8697-8710).

Cell-level Search Space: NASNet

- 🤔 Assuming that we have 2 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 blocks?



Construction of one block



Schematic diagram of the NASNet search space

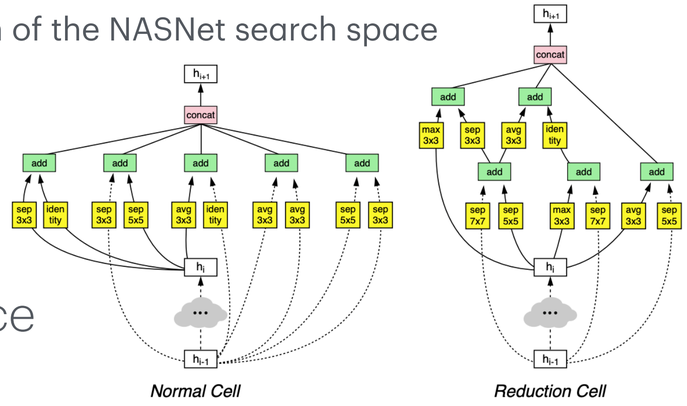
Step 1: 2 options
Step 2: 2 options
Step 3: M options
Step 4: M options
Step 5: N options
Repeat 5 steps B times

Size of the search space: $(2 \times 2 \times M \times M \times N)^B$

Assume $M = 13, N = 2, B = 5$,

There are 4.5×10^{15} candidates in the design space

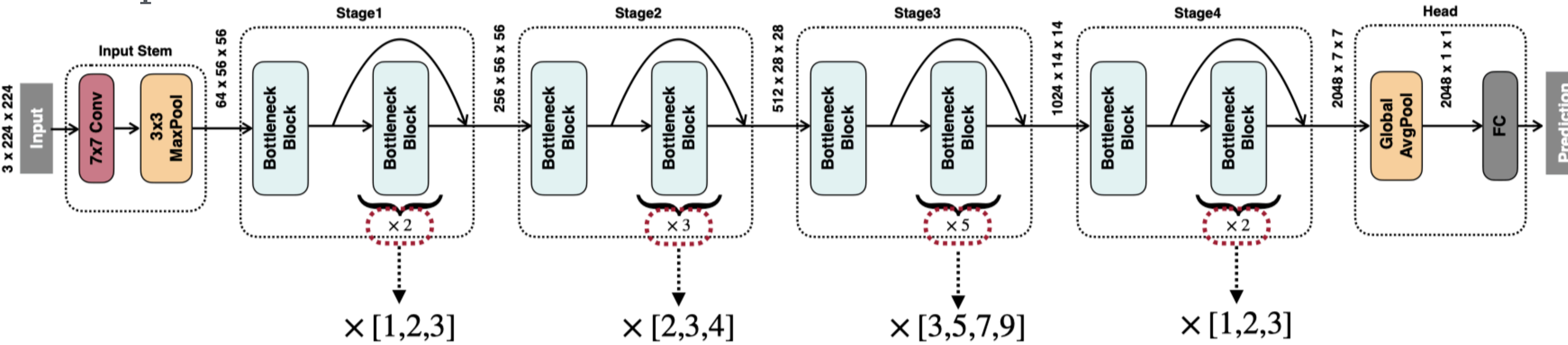
+ 😖 Very complicated dependencies

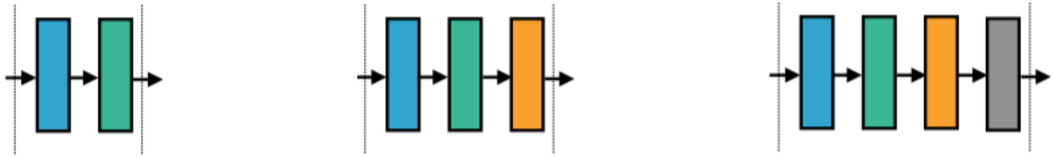


Architecture of the best convolutional cells

Network-level Search Space

Depth dimension

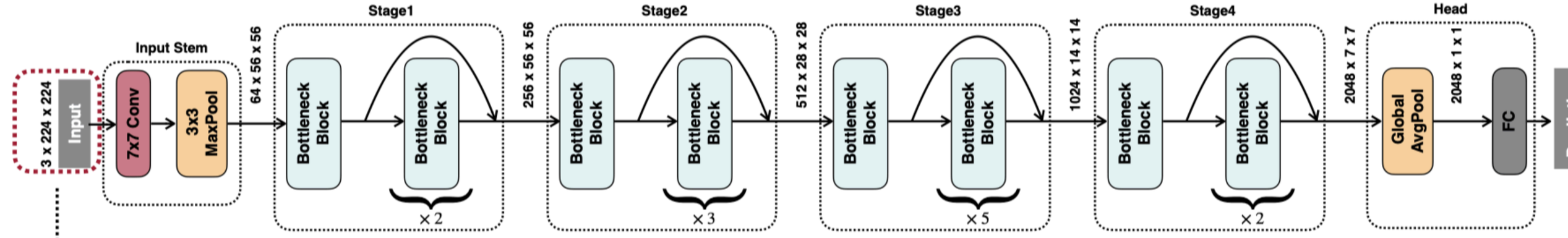




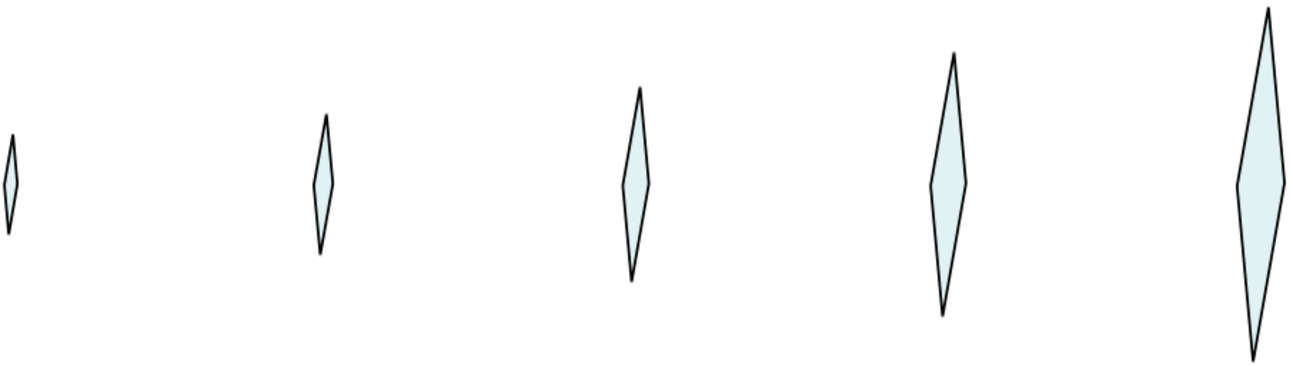
Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Network-level Search Space

Resolution dimension



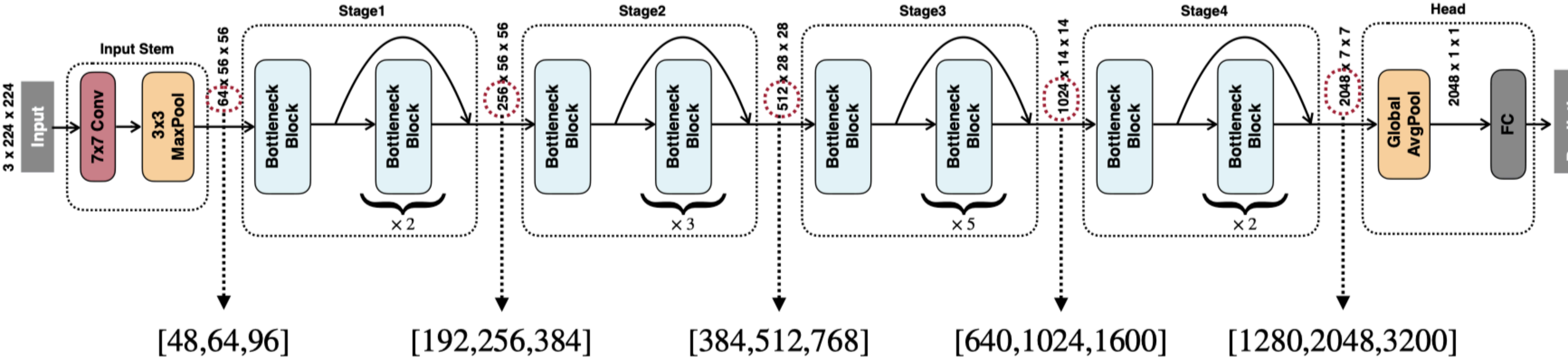
[(3,128,128); (3,160,160); (3,192,192); (3,224,224); (3,256,256)]

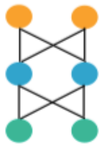


Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Network-level Search Space

Width dimension

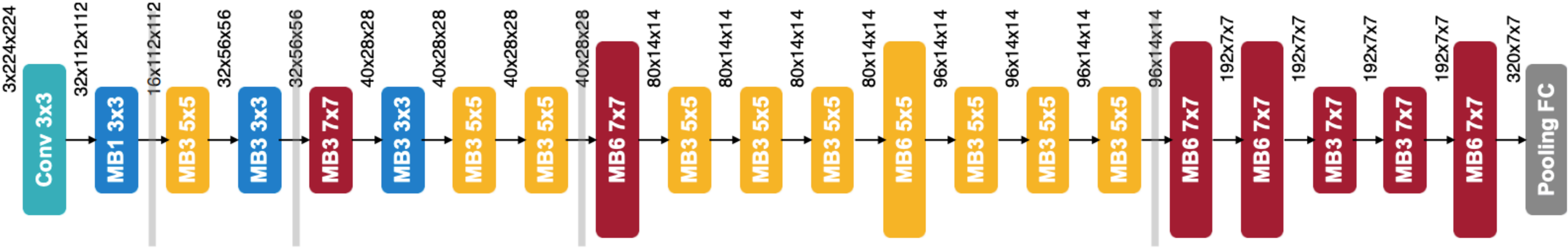


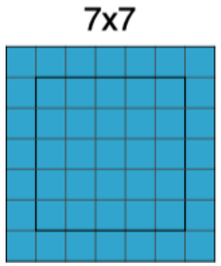


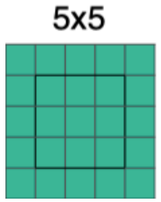
Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

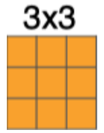
Network-level Search Space

Kernel size dimension

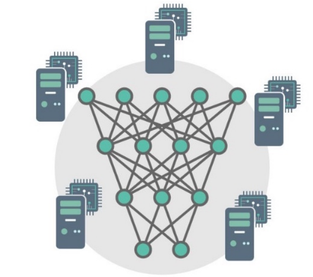








🤔 **2 3x3 conv, vs. 1 5x5 conv, assume the #channels are the same, which choice is better?**

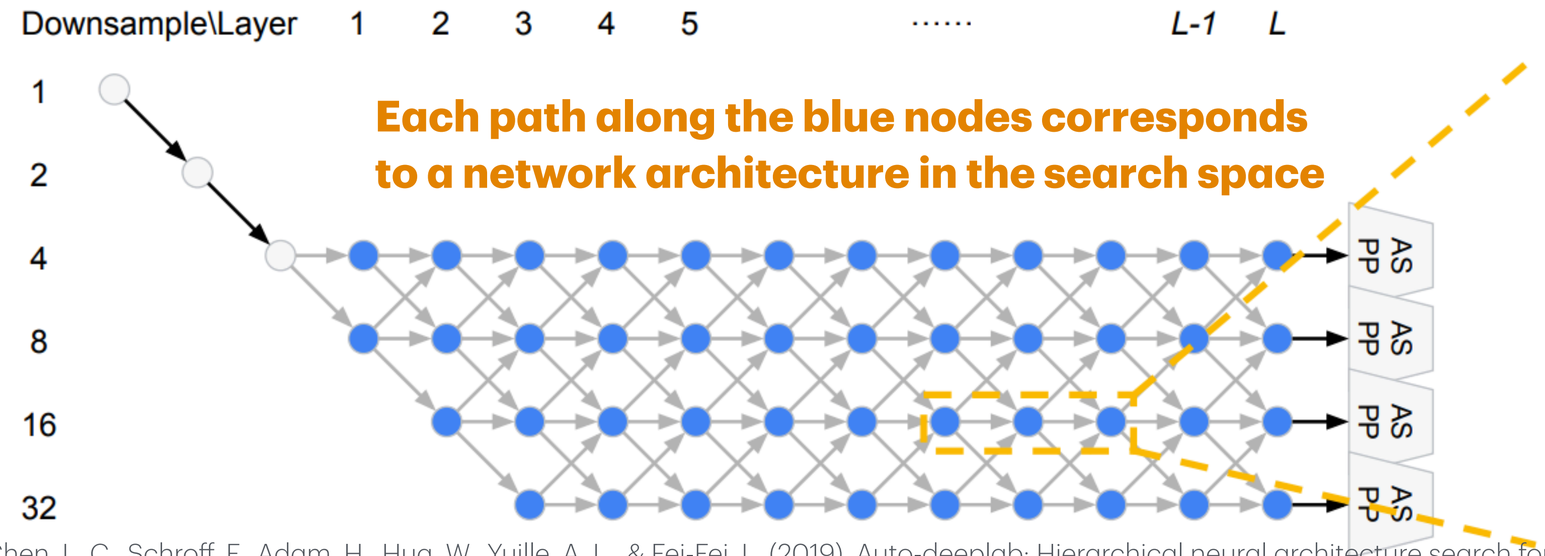


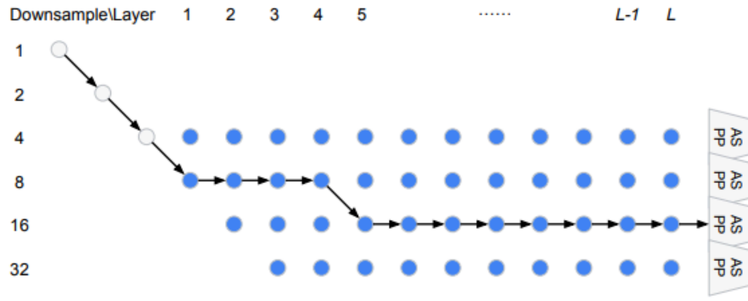
Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Network-level Search Space

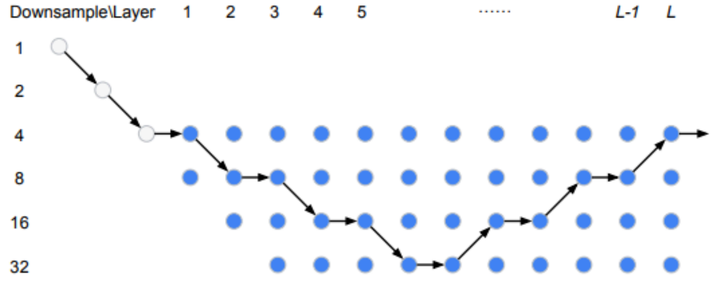
Topology connection

- Propose the network-level search space following **two common principles**:
  - The spatial resolution of the next layer is either twice as large, or twice as small, or remains the same
  - The smallest spatial resolution is downsampled by 32

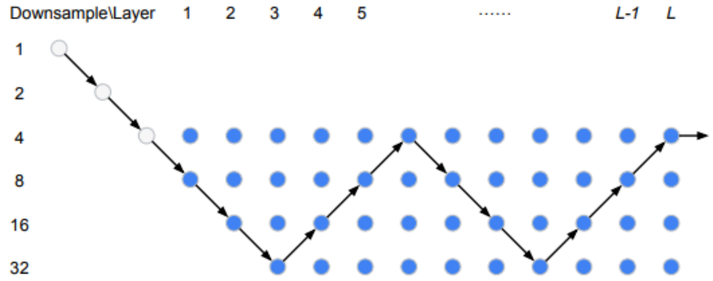




(a) Network level architecture used in DeepLabv3 [9].



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



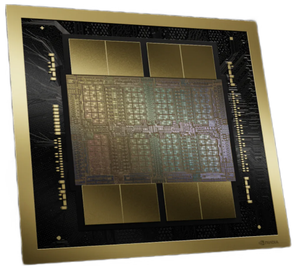
(c) Network level architecture used in Stacked Hourglass [55].

Such network level search space is general and includes various existing designs.

Liu, C., Chen, L. C., Schroff, F., Adam, H., Hua, W., Yuille, A. L., & Fei-Fei, L. (2019). Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 82-92).

Search space design is crucial for NAS performance

Design the Search Space for Edge AI



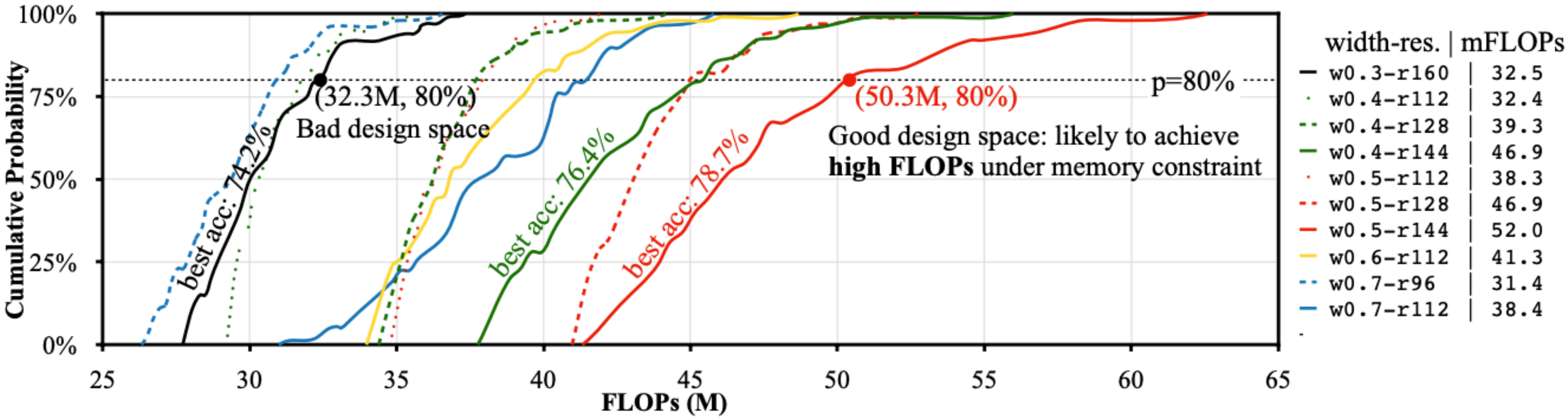
	Cloud AI	Mobile AI	Tiny AI
Memory (Activation)	80GB	4GB	320KB
Storage (Weights)	~TB/PB	256GB	1MB

- Design the search space for mobile phones: **latency** constraint, **energy** constraint
- Design the search space for microcontrollers: + **memory** constraint

Design the Search Space for Edge AI

Larger FLOPs → larger model capacity → more likely to give higher accuracy

- Analyzing **FLOPs distribution** of satisfying models

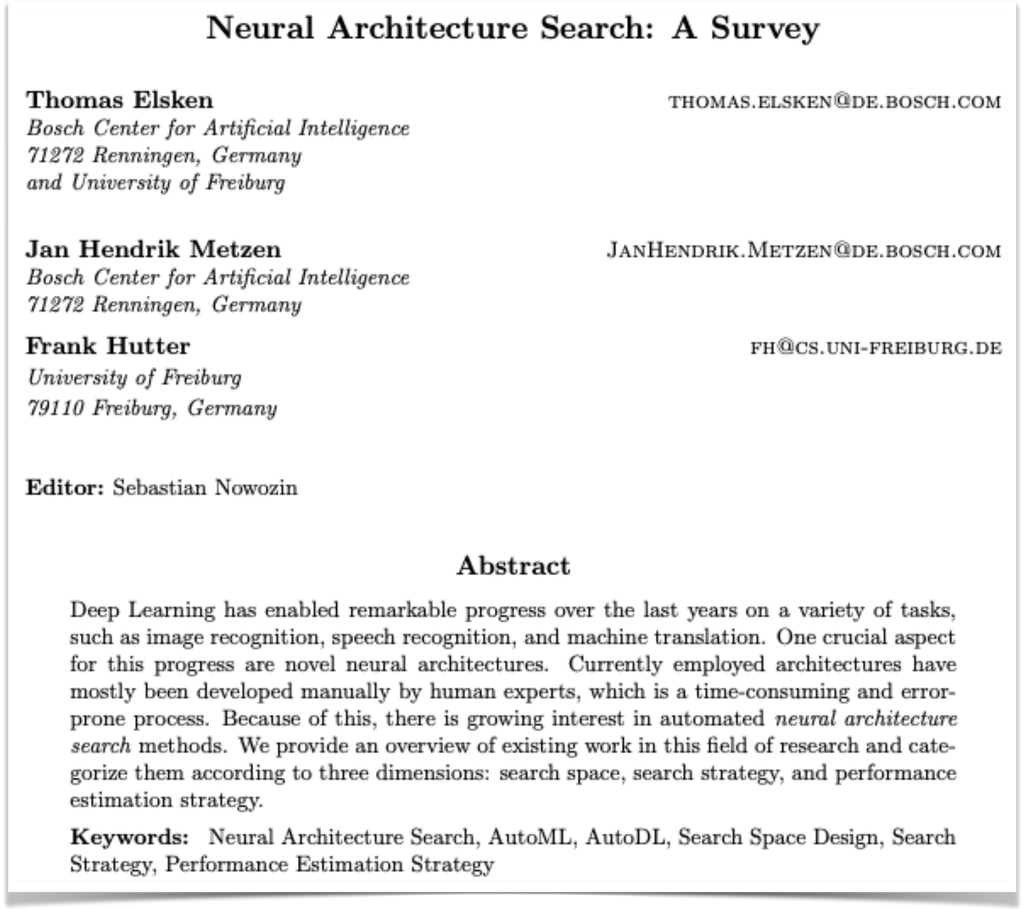


FLOPs CDF of different search spaces. Each curve represents a design space.

Lin, J., Chen, W. M., Lin, Y., Gan, C., & Han, S. (2020). Mcunet: Tiny deep learning on iot devices. Advances in neural information processing systems, 33, 11711-11722.

Framework of Neural Architecture Search

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

Thomas Elsken THOMAS.ELSKEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany
and University of Freiburg

Jan Hendrik Metzen JANHENDRIK.METZEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany

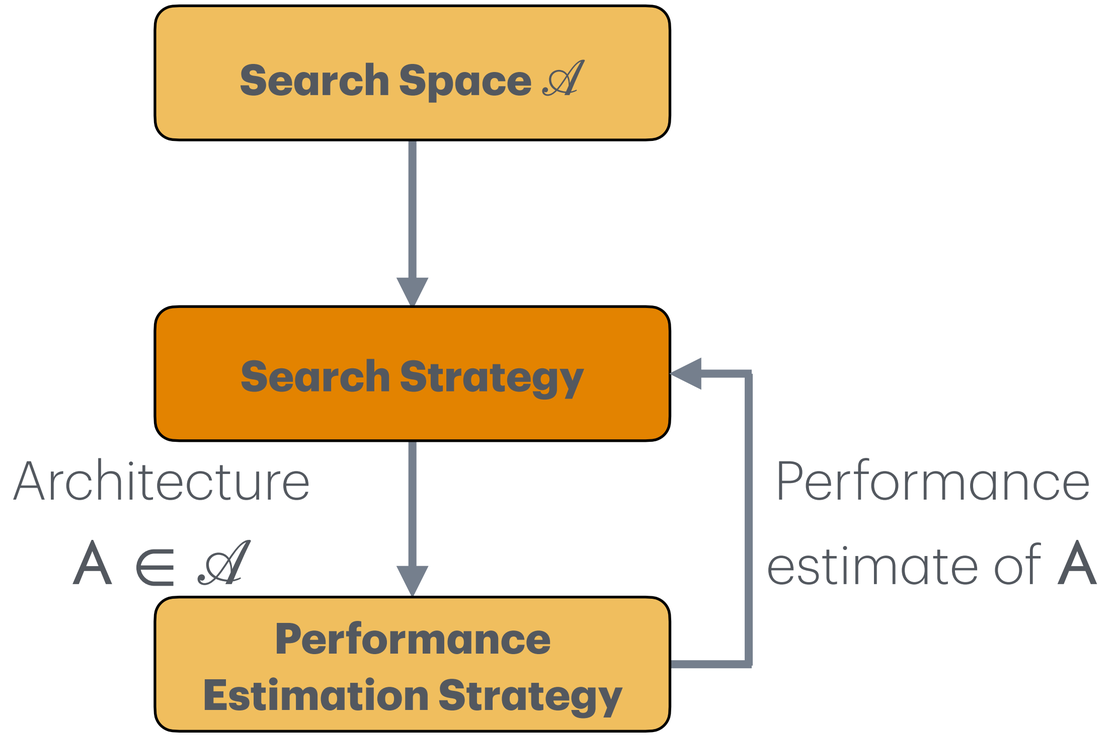
Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg
79110 Freiburg, Germany

Editor: Sebastian Nowozin

Abstract

Deep Learning has enabled remarkable progress over the last years on a variety of tasks, such as image recognition, speech recognition, and machine translation. One crucial aspect for this progress are novel neural architectures. Currently employed architectures have mostly been developed manually by human experts, which is a time-consuming and error-prone process. Because of this, there is growing interest in automated *neural architecture search* methods. We provide an overview of existing work in this field of research and categorize them according to three dimensions: search space, search strategy, and performance estimation strategy.

Keywords: Neural Architecture Search, AutoML, AutoDL, Search Space Design, Search Strategy, Performance Estimation Strategy



Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.

Search Strategy in NAS

Search strategy defines how to explore the search space

- Grid search
- Random search
- Reinforcement learning
- Gradient descent
- **Evolutionary search**

Search Strategy: Grid Search

- Grid search is the traditional way of hyper parameter optimization. The entire design space is represented as the Catesian 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

Width \ Resolution	**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%

Values in the grids correspond to the accuracy of candidate networks

Satisfies the latency constraint

Breaks the latency constraint

Search Strategy: Grid Search

EfficientNet

😣 **An exhaustive search strategy**

- Target: maximize the model accuracy for any given resource constraints

$$\max_{d,w,r} \quad Accuracy(\mathcal{N}(d,w,r))$$

$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\left(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\right)$$

$$\text{Memory}(\mathcal{N}) \leq \text{target\_memory}$$

$$\text{FLOPS}(\mathcal{N}) \leq \text{target\_flops}$$

- Apply **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

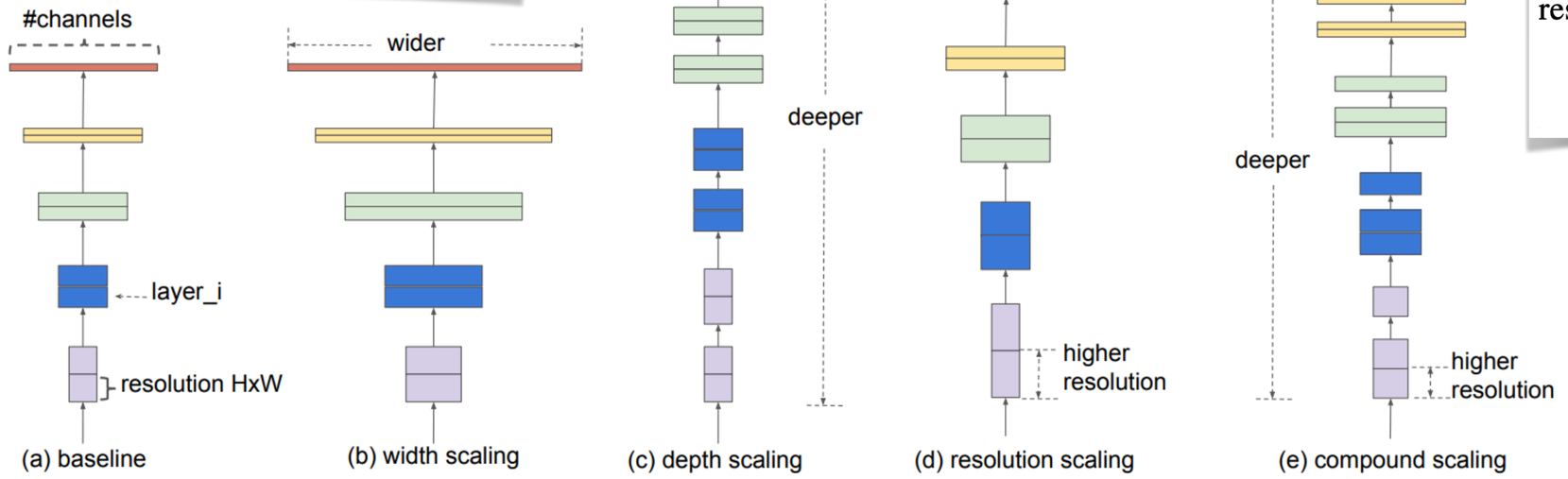
$$\text{depth: } d = \alpha^\phi$$

$$\text{width: } w = \beta^\phi$$

$$\text{resolution: } r = \gamma^\phi$$

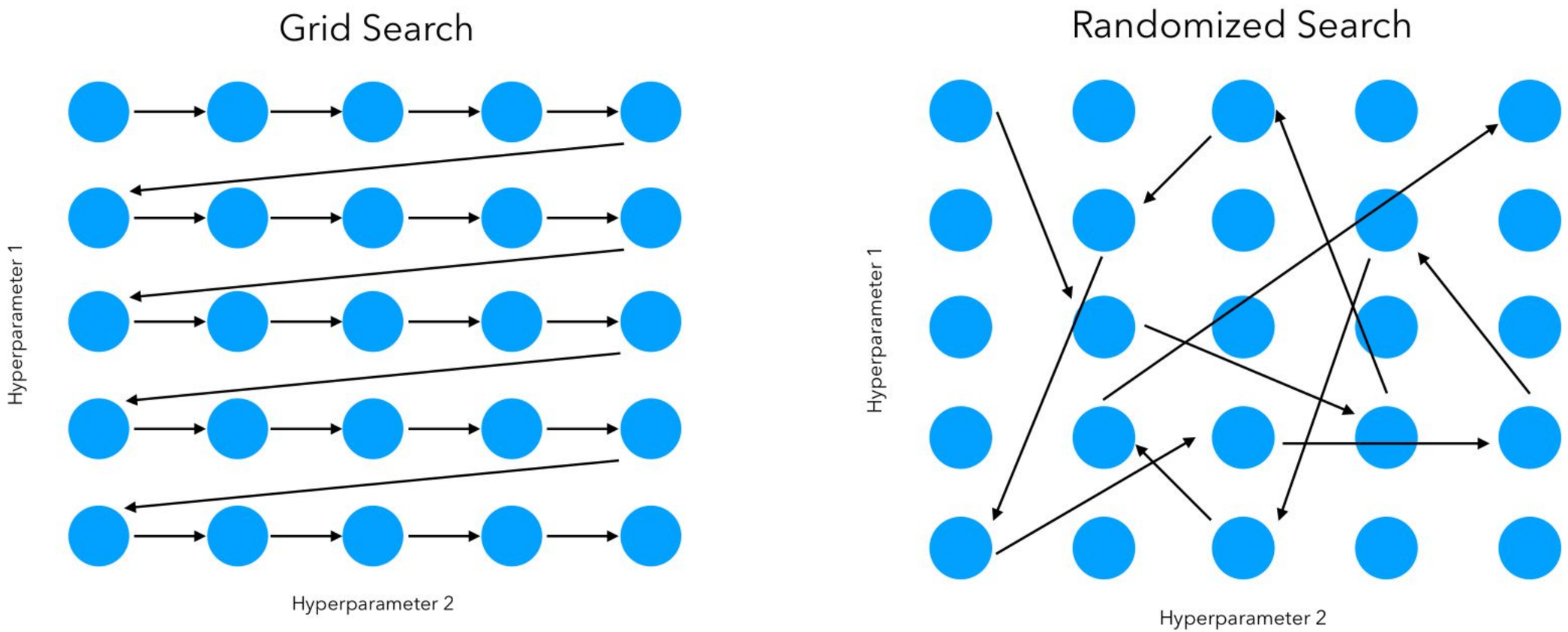
$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$



Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.

Search Strategy: Random Search



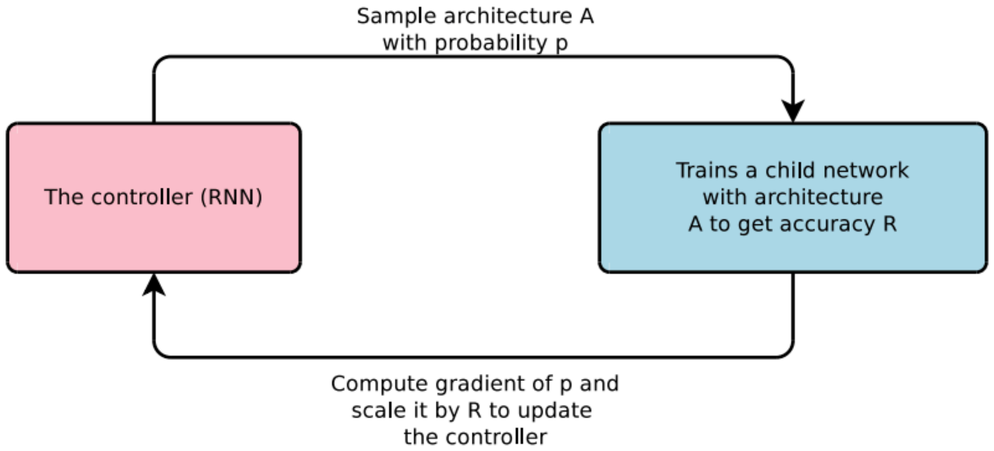
- Random search often finds better-performing models faster than grid search, especially when only a subset of hyperparameters is important for performance

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(2).

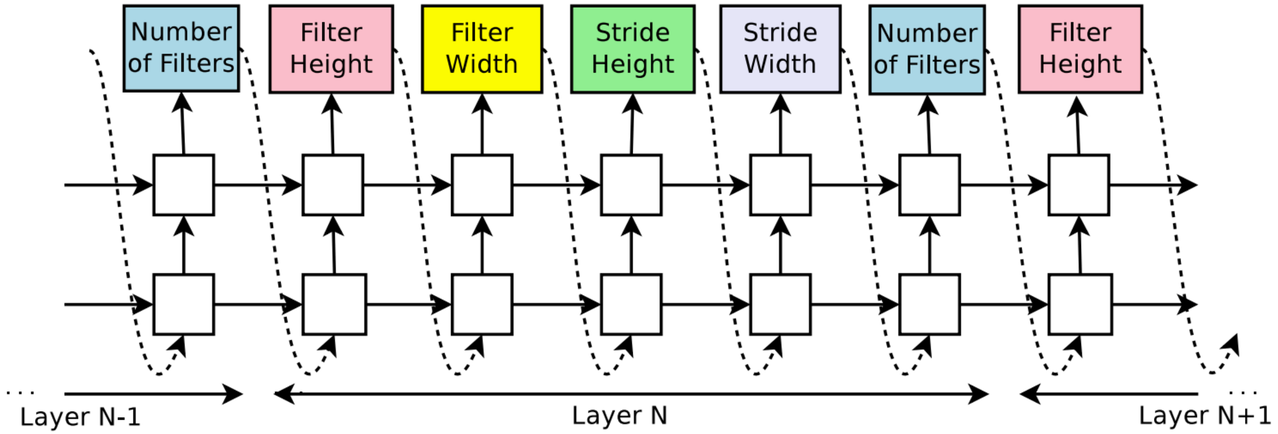
Figure source

Search Strategy: Reinforcement Learning

- Model neural architecture design as a **sequential decision-making problem**
- Use reinforcement learning to **train the controller** that is implemented with an RNN



Overview of RL-based NAS



The RNN controller samples a simple convolutional network

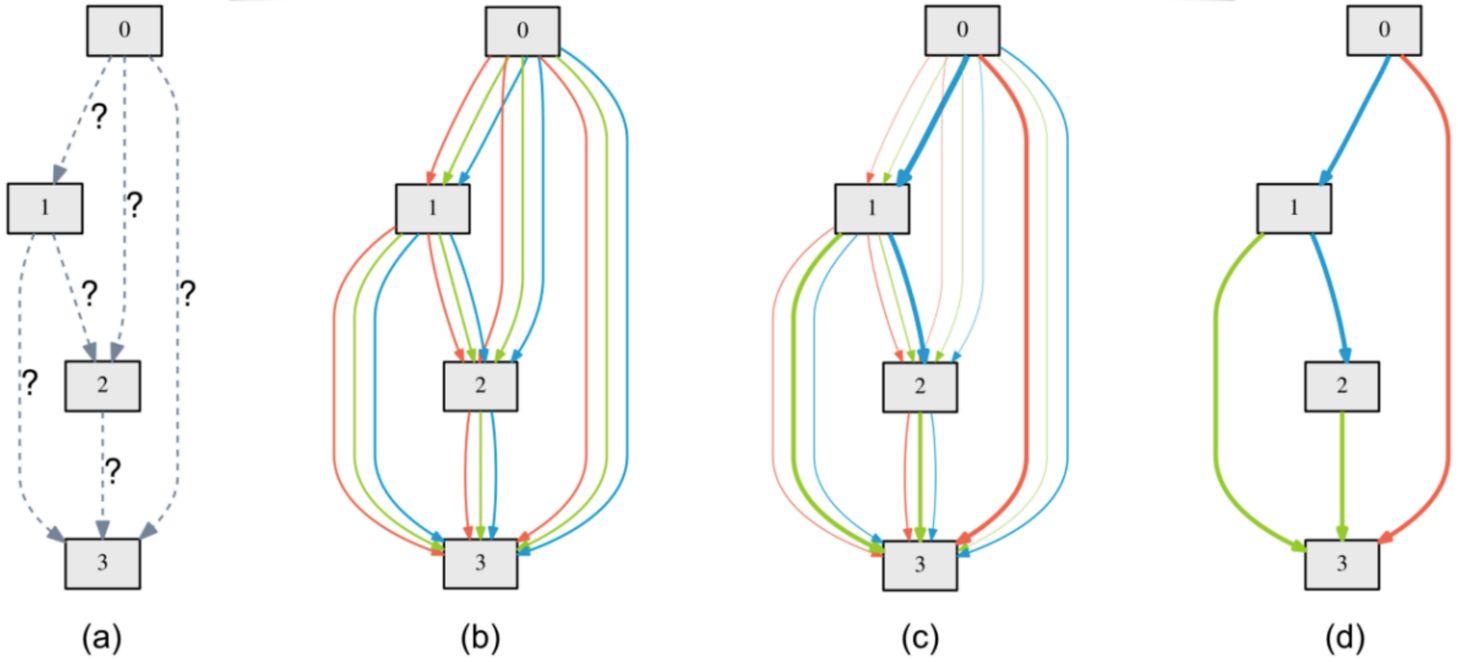
Zoph, B. (2016). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.

Search Strategy: Gradient Descent

DARTS

- Represent the output at each node as a weighted sum of outputs from different edges

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



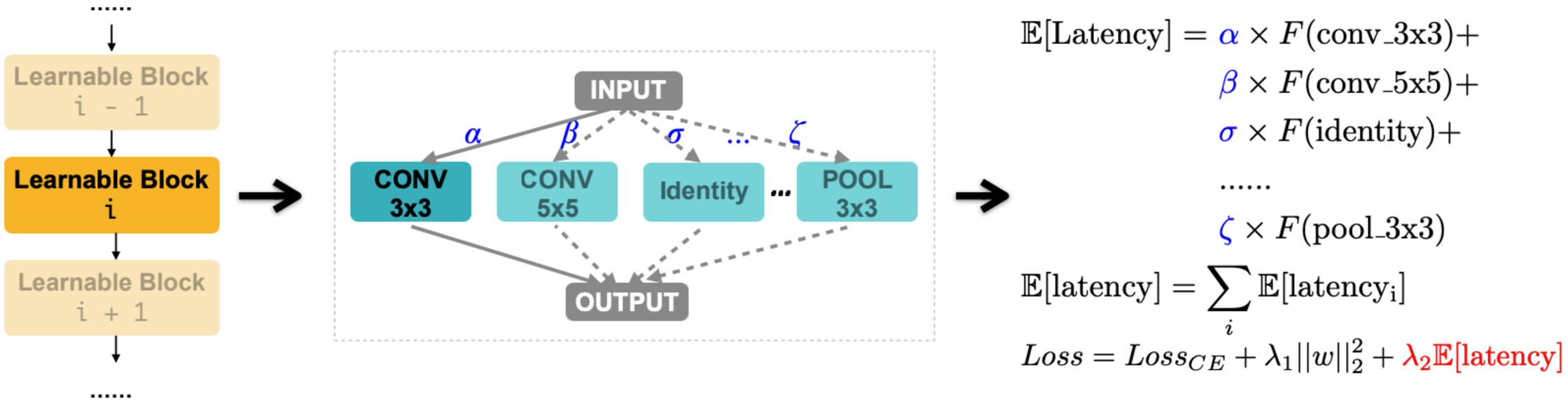
Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055.

Search Strategy: Gradient Descent

ProxylessNAS

- Is it 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



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

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

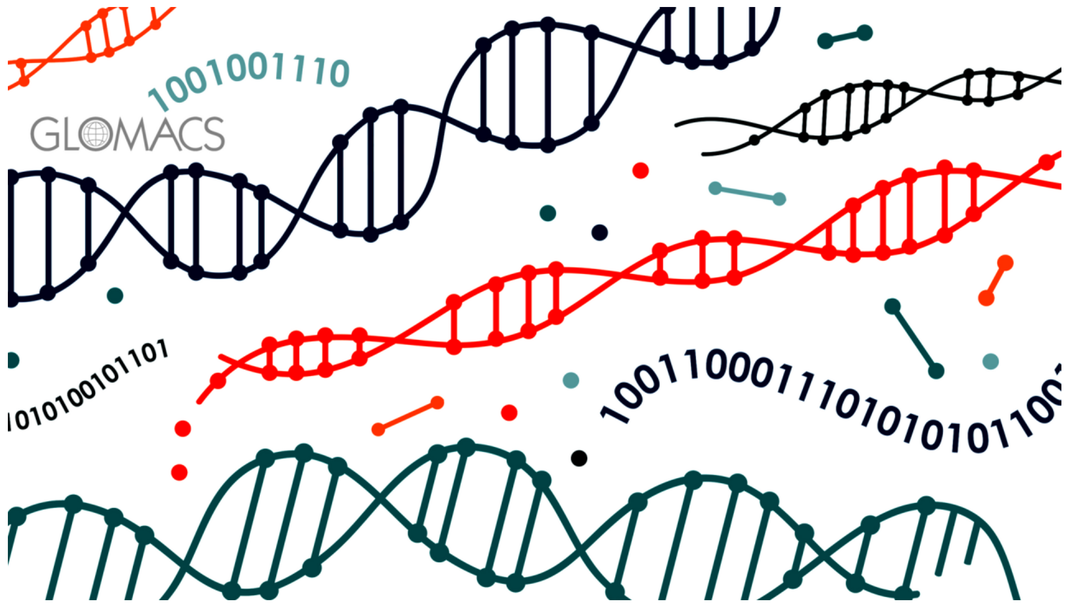
$$Loss = Loss_{CE} + \lambda_1 ||w||_2^2 + \lambda_2 \mathbb{E}[\text{latency}]$$

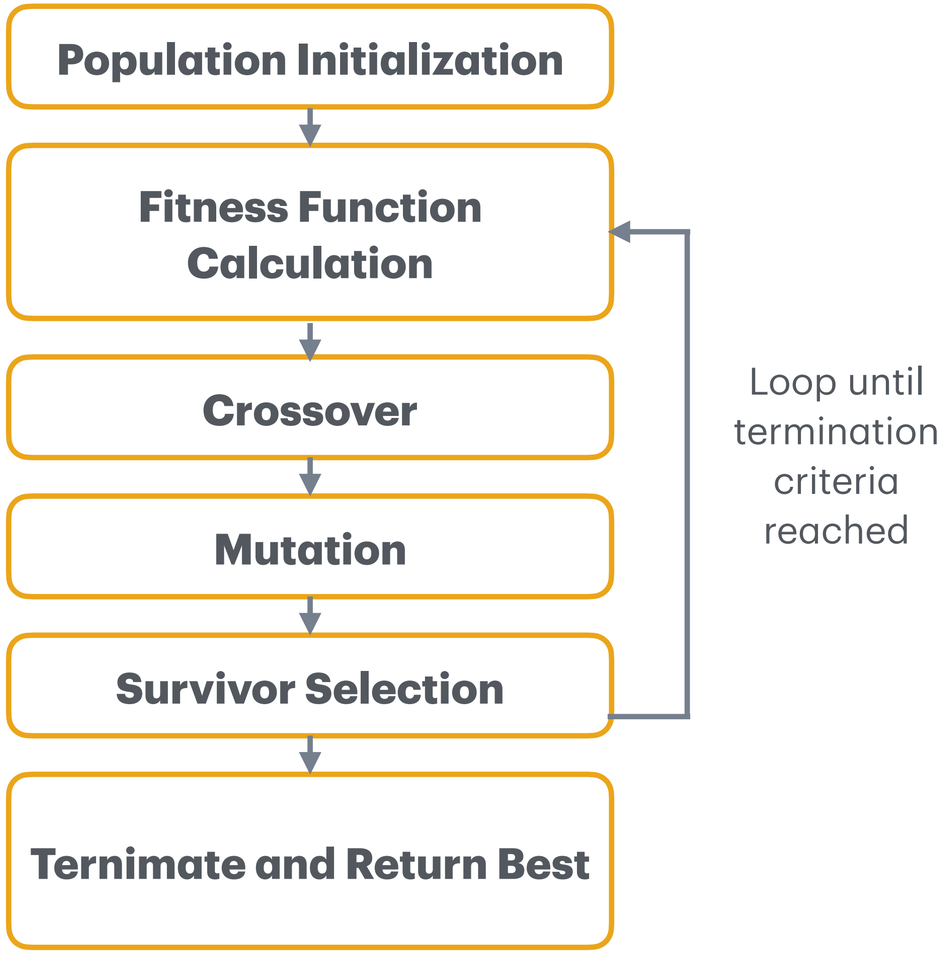
Making latency differentiable by introducing latency regularization loss

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Search Strategy: Evolutionary Search

Mimic the evolution process in biology

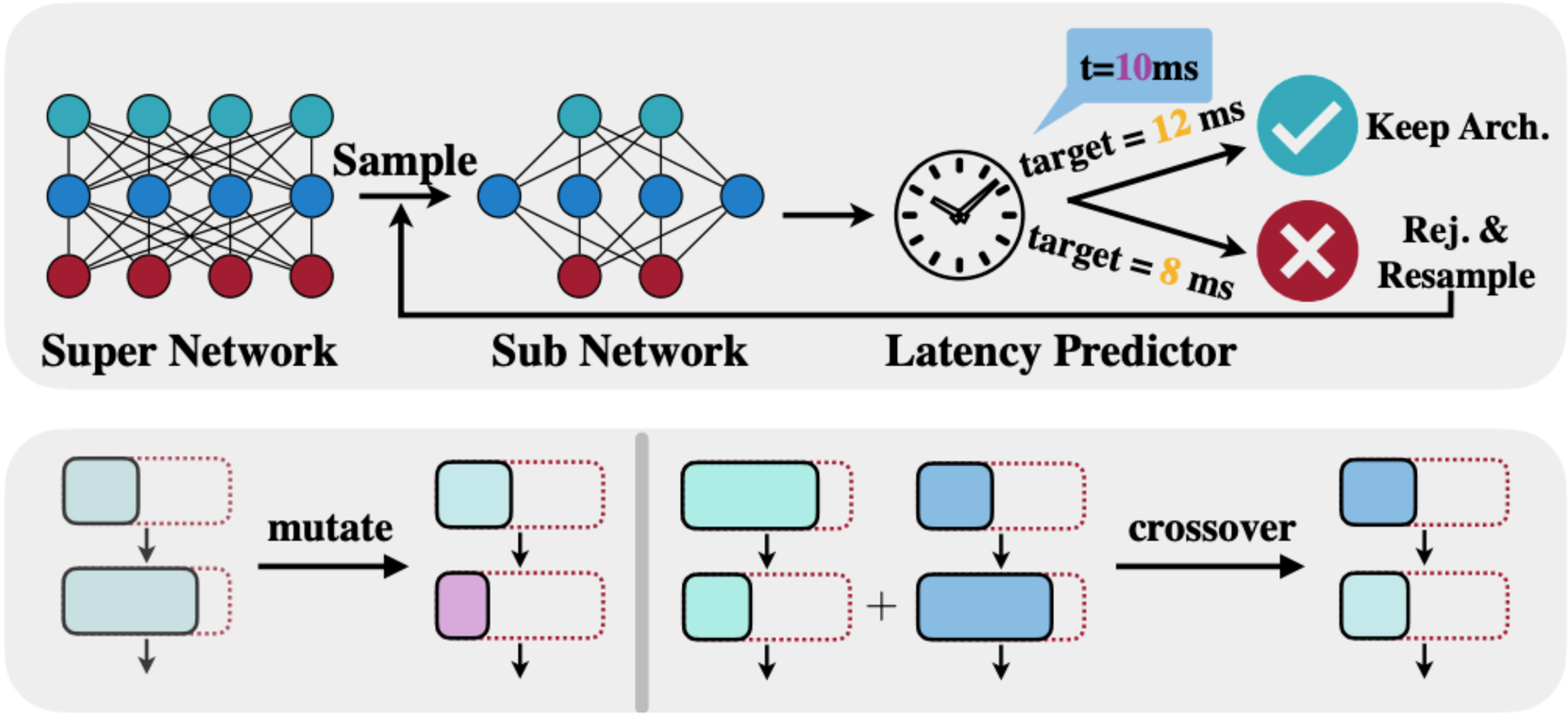




Search Strategy: Evolutionary Search

Fitness function F(accuracy, efficiency)

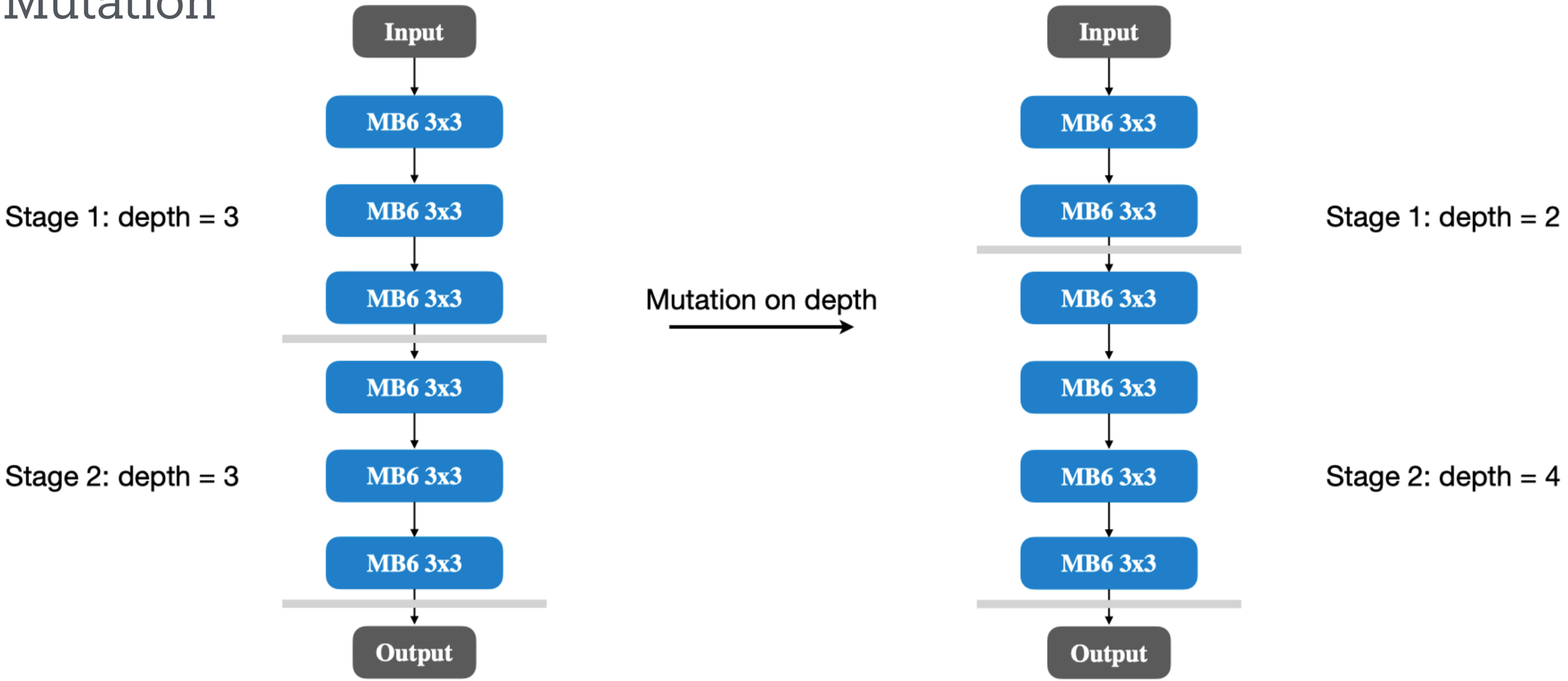
- Given a specific resource constraint, find the best candidate network



Liu, Z., Tang, H., Zhao, S., Shao, K., & Han, S. (2021). Pvnas: 3d neural architecture search with point-voxel convolution. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11), 8552-8568.

Search Strategy: Evolutionary Search

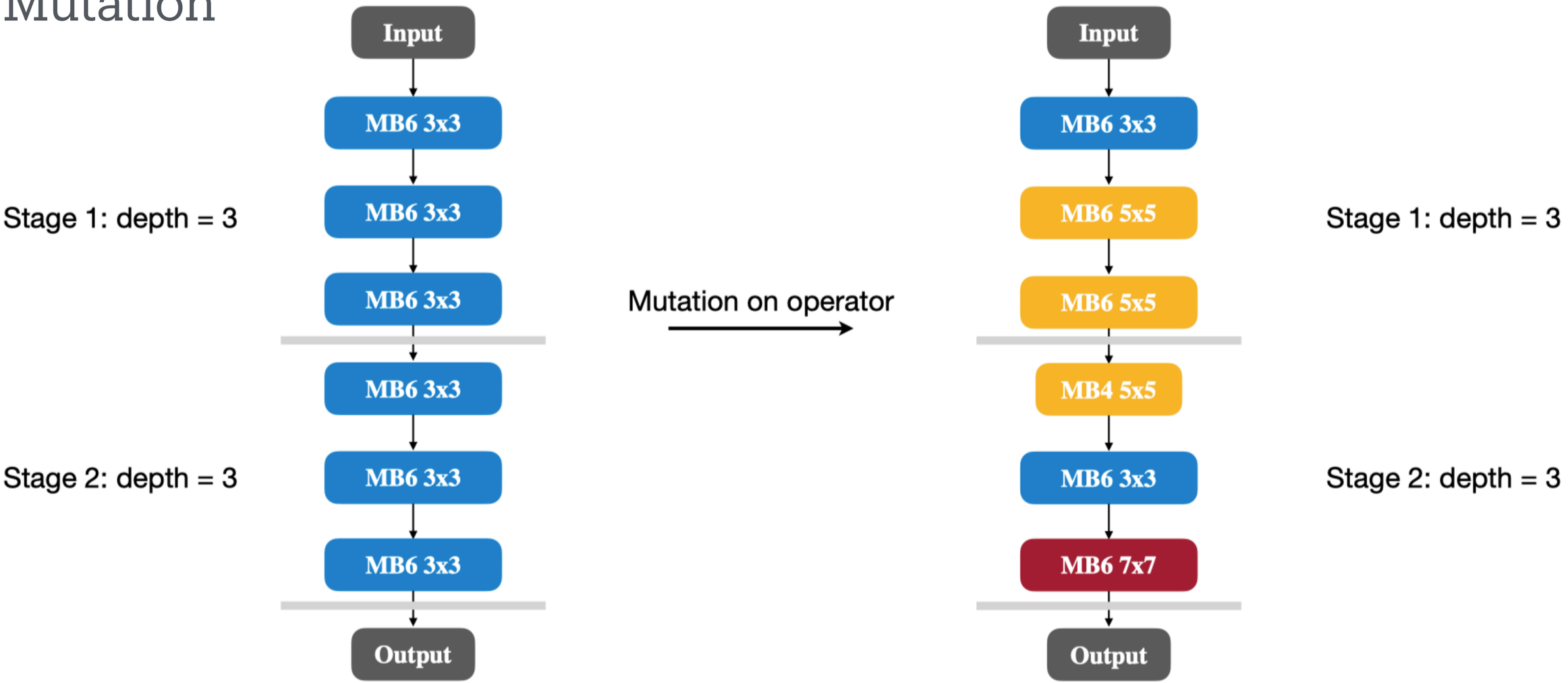
Mutation



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Search Strategy: Evolutionary Search

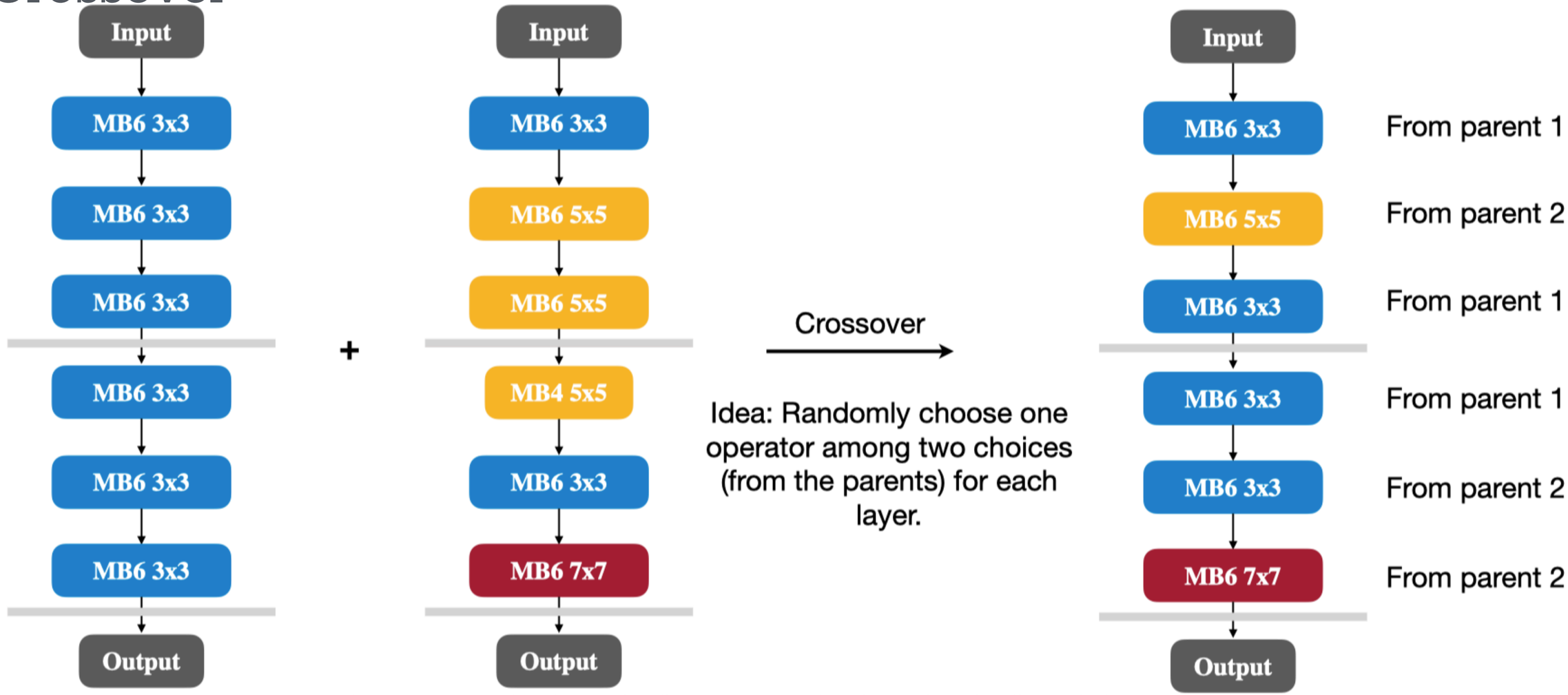
Mutation



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Search Strategy: Evolutionary Search

Crossover



Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.

Framework of Neural Architecture Search

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

Thomas Elsken THOMAS.ELSKEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany
and University of Freiburg

Jan Hendrik Metzen JANHENDRIK.METZEN@DE.BOSCH.COM
Bosch Center for Artificial Intelligence
71272 Renningen, Germany

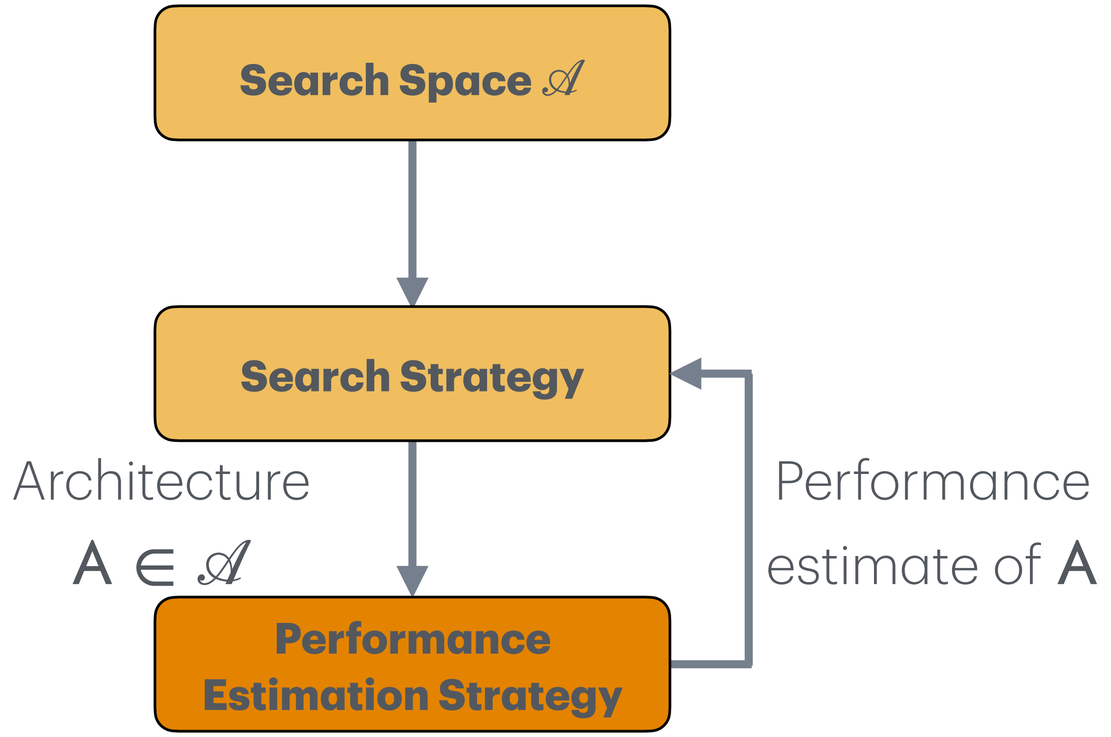
Frank Hutter FH@CS.UNI-FREIBURG.DE
University of Freiburg
79110 Freiburg, Germany

Editor: Sebastian Nowozin

Abstract

Deep Learning has enabled remarkable progress over the last years on a variety of tasks, such as image recognition, speech recognition, and machine translation. One crucial aspect for this progress are novel neural architectures. Currently employed architectures have mostly been developed manually by human experts, which is a time-consuming and error-prone process. Because of this, there is growing interest in automated *neural architecture search* methods. We provide an overview of existing work in this field of research and categorize them according to three dimensions: search space, search strategy, and performance estimation strategy.

Keywords: Neural Architecture Search, AutoML, AutoDL, Search Space Design, Search Strategy, Performance Estimation Strategy



Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.

Reference

- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
- Srinivas, A., Lin, T. Y., Parmar, N., Shlens, J., Abbeel, P., & Vaswani, A. (2021). Bottleneck transformers for visual recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 16519-16529).
- Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1492-1500).
- Howard, A. G. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.
- Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).
- Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 6848-6856).
- Vaswani, A. (2017). Attention is all you need. Advances in Neural Information Processing Systems.
- Deng, L., Li, G., Han, S., Shi, L., & Xie, Y. (2020). Model compression and hardware acceleration for neural networks: A comprehensive survey. Proceedings of the IEEE, 108(4), 485-532.
- Elsken, T., Metzen, J. H., & Hutter, F. (2019). Neural architecture search: A survey. Journal of Machine Learning Research, 20(55), 1-21.
- Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2018). Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 8697-8710).

Reference

- Cai, H., Gan, C., Wang, T., Zhang, Z., & Han, S. (2019). Once-for-all: Train one network and specialize it for efficient deployment. arXiv preprint arXiv:1908.09791.
- Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.
- Liu, C., Chen, L. C., Schroff, F., Adam, H., Hua, W., Yuille, A. L., & Fei-Fei, L. (2019). Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 82-92).
- Lin, J., Chen, W. M., Lin, Y., Gan, C., & Han, S. (2020). Mcunet: Tiny deep learning on iot devices. Advances in neural information processing systems, 33, 11711-11722.
- Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning (pp. 6105-6114). PMLR.
- Zoph, B. (2016). Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578.
- Liu, H., Simonyan, K., & Yang, Y. (2018). Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055.
- Liu, Z., Tang, H., Zhao, S., Shao, K., & Han, S. (2021). Pvnas: 3d neural architecture search with point-voxel convolution. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11), 8552-8568.
- Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(2).
- A Guide to Hyperparameter Optimization [link]
- Neural Architecture Search [MIT 6.5940]