Artificial Intelligence: Convolutional Neural Network (CNN) Architectures

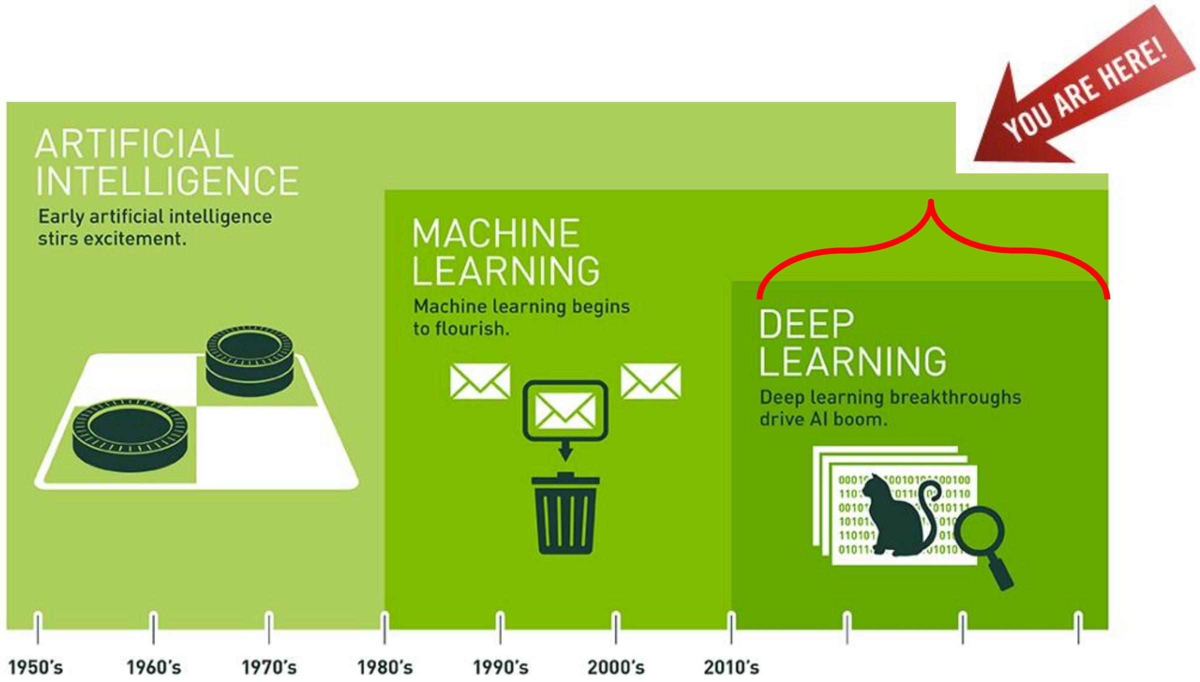
portion of slides from: Fei Fei Li

Today

1. Applications 

2. Backbone Models
   1. Serial Cascading
   2. Serial/Parallel Cascading
   3. Residual Connection
   4. Depthwise/Separable Convolutions
   5. Squeeze-Excitation

History of AI

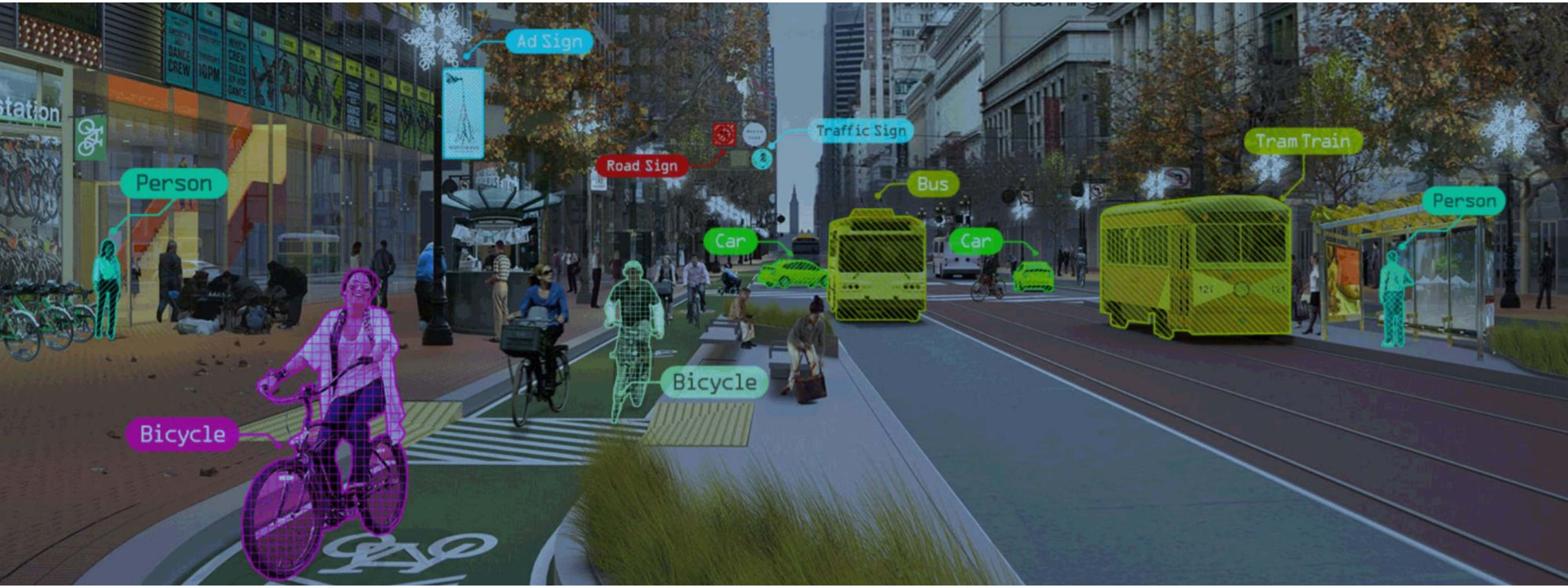


https://www.linkedin.com/pulse/goedels-incompleteness-theorem-emergence-ai-eberhard-schoeneburg/

How Computers Recognize Objects?

Question: Objects are anywhere in the scene (in any orientation, color hue, perspectives, illumination, etc), so how can we recognize them?

Answer: Learn a ton of features (millions) from the bottom up, by learning convolutional filters rather than pre-computing them



Feature Invariance to Perturbation is Hard

Viewpoint Variation
(Perspective Geometry)

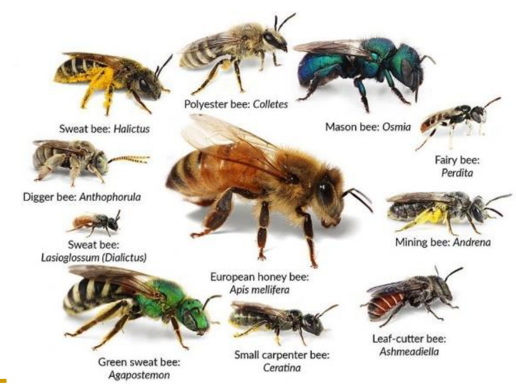
Scale Variation

Deformation

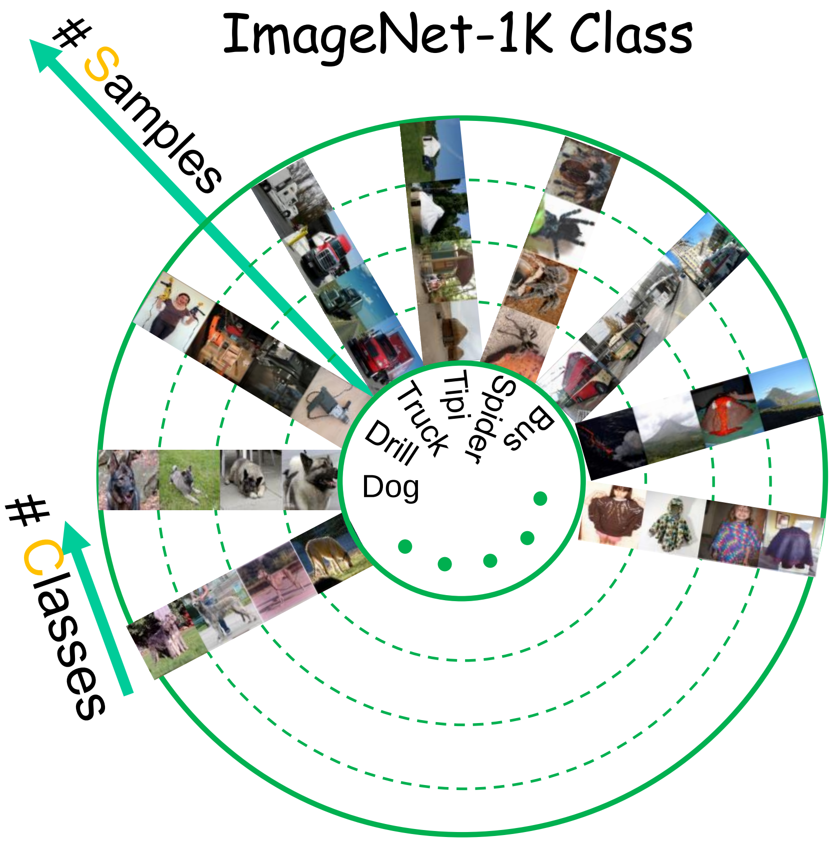
Illumination Conditions

Background Clutter

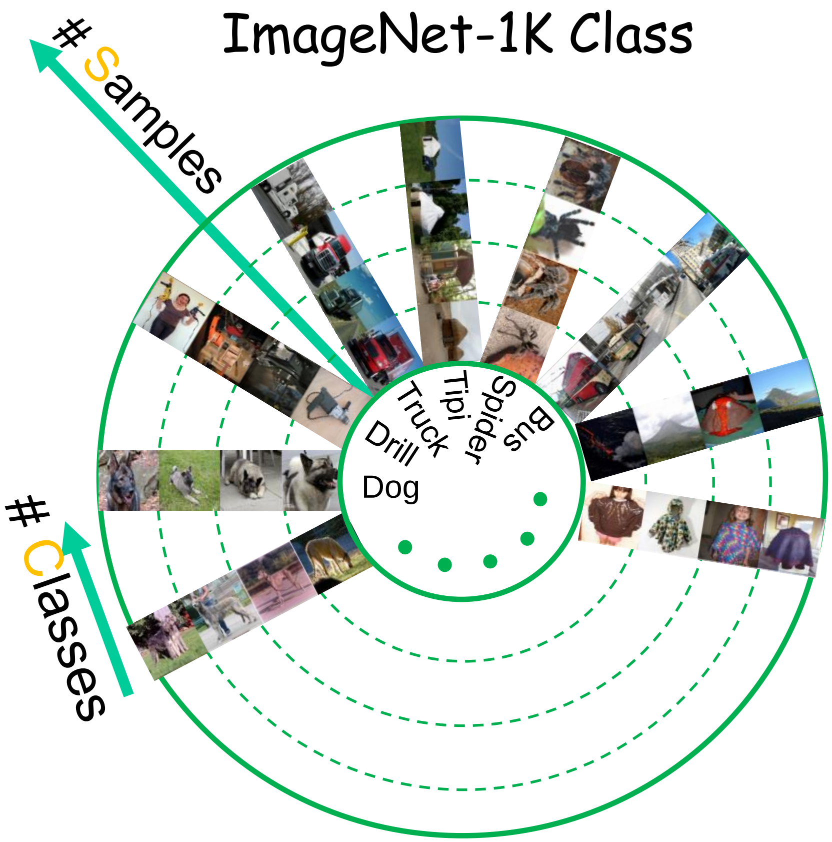
Intra-Class Variation

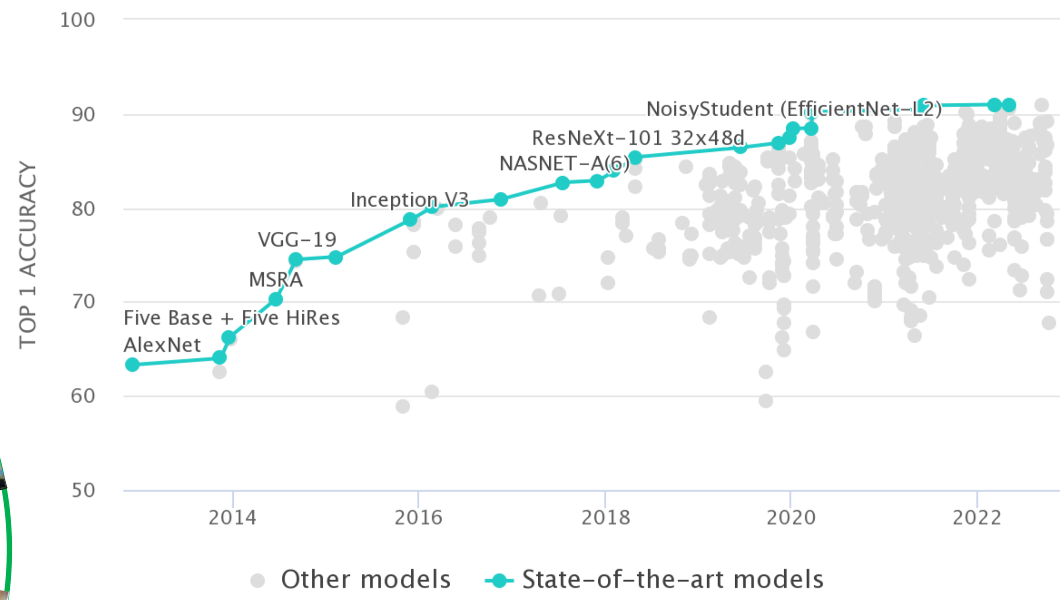


ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

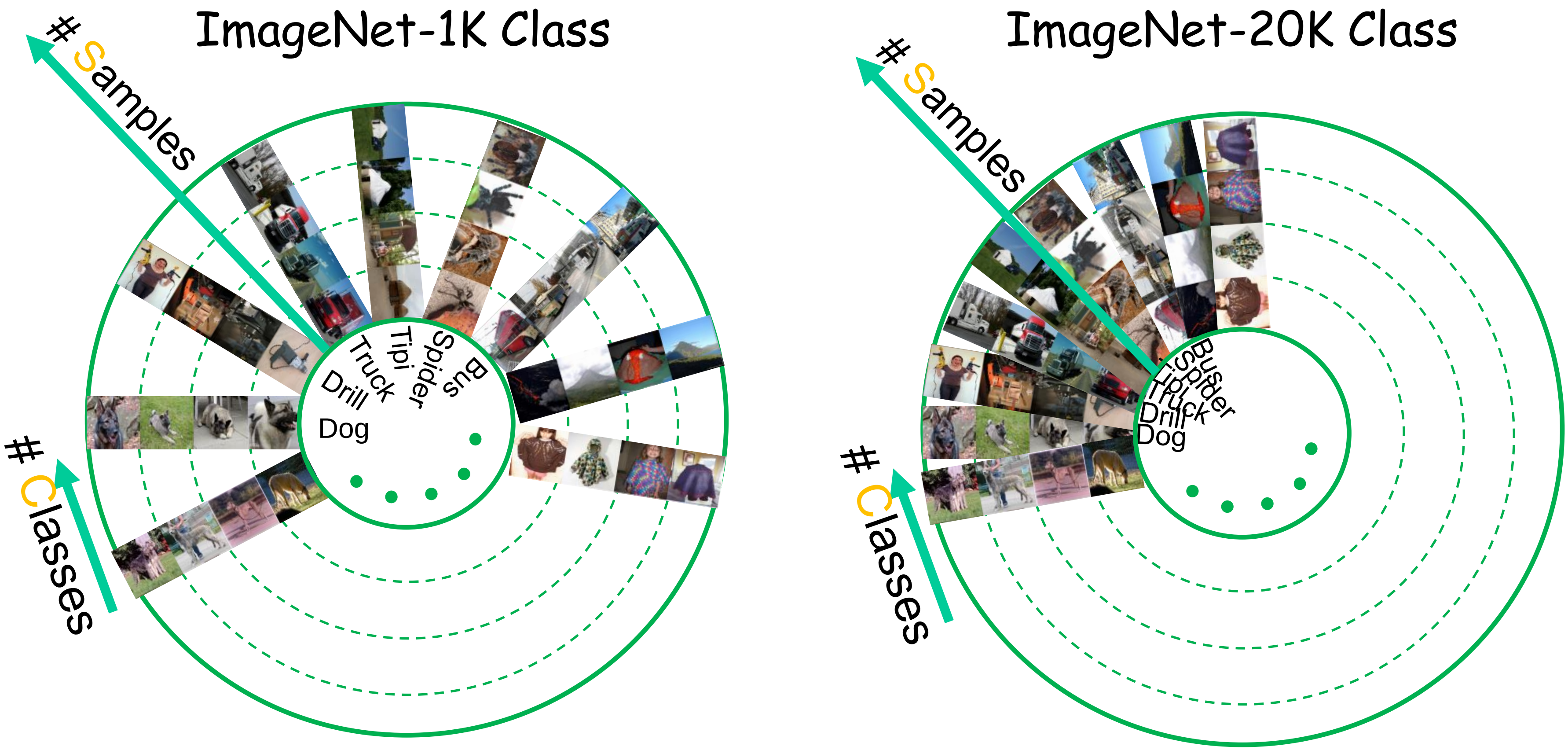


ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



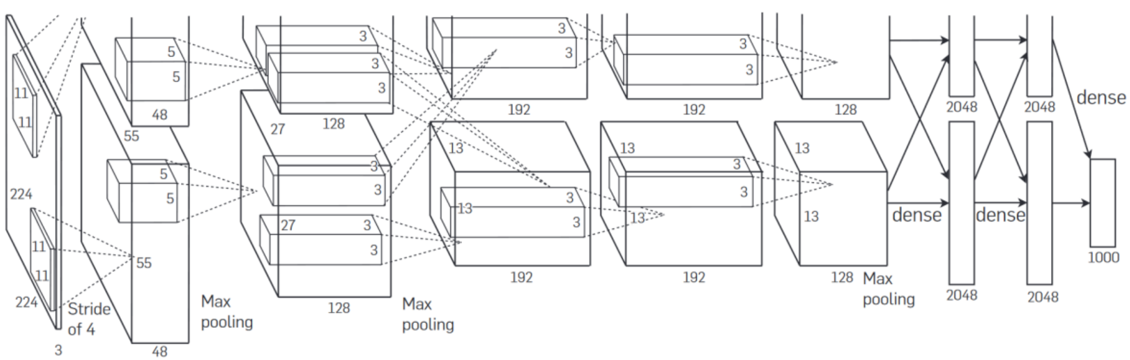


ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Serial Cascade: AlexNet

- 5 Conv and 3 FC layers
- ReLU Activation
- Training on Multiple GPUs (GTX 580 with 3GB memory)
- Response Normalization
- Overlapping Pooling
- Heavy data augmentation
  - Image translation/horizontal reflection
  - Altering intensities of RGB channels using PCA



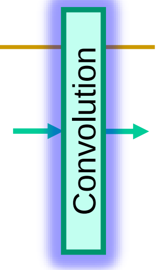
CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

ImageNet Classification with Deep Convolutional Neural Networks

Part of Advances in Neural Information Processing Systems 25 (NIPS 2012)

Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

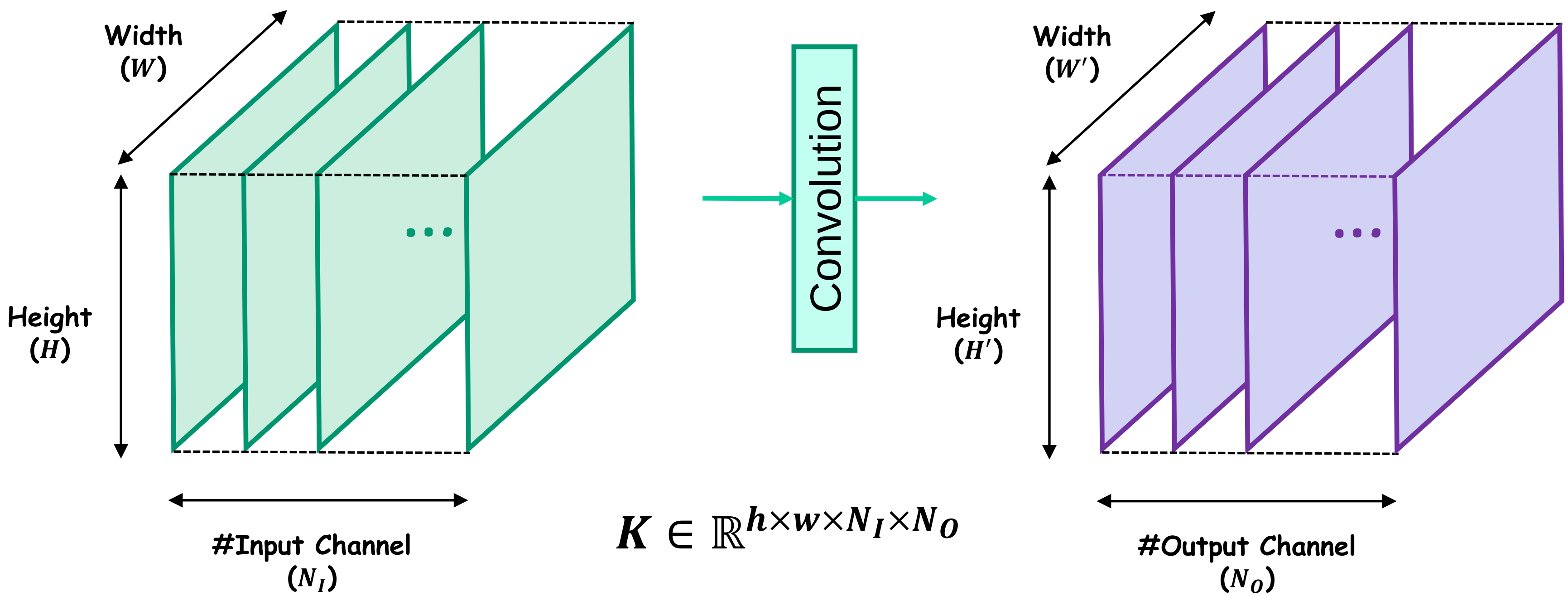
Serial Cascade: AlexNet



Reminder from Convolution Feature mapping

$\boldsymbol{F^I} \in \mathbb{R}^{\boldsymbol{H \times W \times N_I}}$

$\boldsymbol{F^O} \in \mathbb{R}^{\boldsymbol{H' \times W' \times N_O}}$

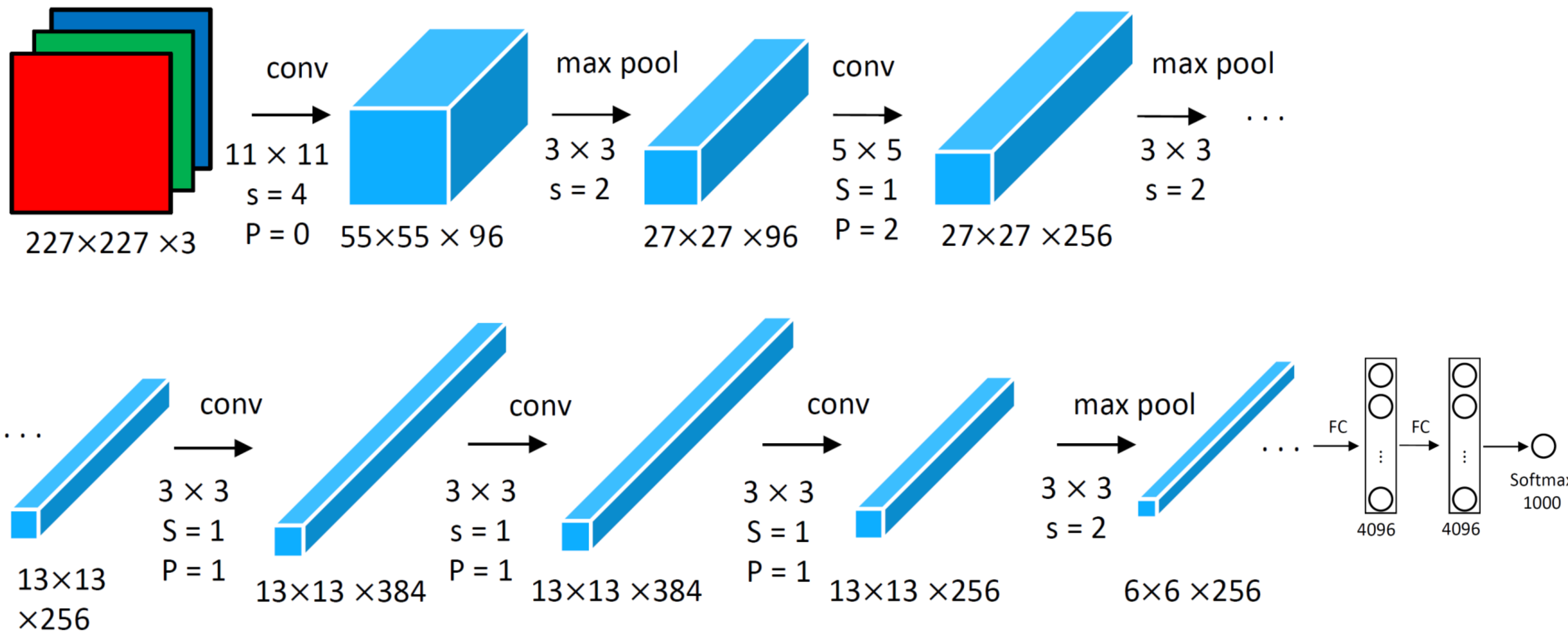


$\boldsymbol{K} \in \mathbb{R}^{\boldsymbol{h \times w \times N_I \times N_O}}$

$$\boxed{\boldsymbol{F^O}(:,:,\boldsymbol{j}) = \left[\sum_{\boldsymbol{i=1}}^{\boldsymbol{N_I}} \boldsymbol{F^I}(:,:,\boldsymbol{i}) * \boldsymbol{K}(:,:,\boldsymbol{i},\boldsymbol{j})\right] + \boldsymbol{b_j}}$$

Serial Cascade: AlexNet

- Feature mapping via cascaded convolutional layers



ImageNet Classification with Deep Convolutional Neural Networks

Part of Advances in Neural Information Processing Systems 25 (NIPS 2012)

Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

Serial Cascade: AlexNet

- AlexNet was the coming out party for CNNs in the computer vision community. This was the first time a model performed so well on a historically difficult ImageNet dataset.

Table 1. Comparison of results on ILSVRC-2010 test set.

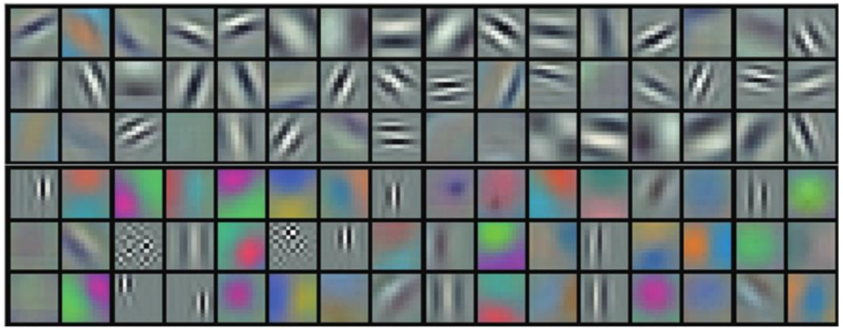
Model	Top-1 (%)	Top-5 (%)
Sparse coding[2]	*47.1*	*28.2*
SIFT + FVs[29]	*45.7*	*25.7*
CNN	**37.5**	**17.0**

Table 2. Comparison of error rates on ILSVRC-2012 validation and test sets.

Model	Top-1 (val, %)	Top-5 (val, %)	Top-5 (test, %)
SIFT + FVs[6]	–	–	*26.2*
1 CNN	40.7	18.2	–
5 CNNs	38.1	16.4	**16.4**
1 CNN*	39.0	16.6	–
7 CNNs*	36.7	15.4	**15.3**

- How did the learned kernels responses look like?

Figure 3. Ninety-six convolutional kernels of size 11 × 11 × 3 learned by the first convolutional layer on the 224 × 224 × 3 input images. The top 48 kernels were learned on GPU 1 while the bottom 48 kernels were learned on GPU 2 (see Section 7.1 for details).



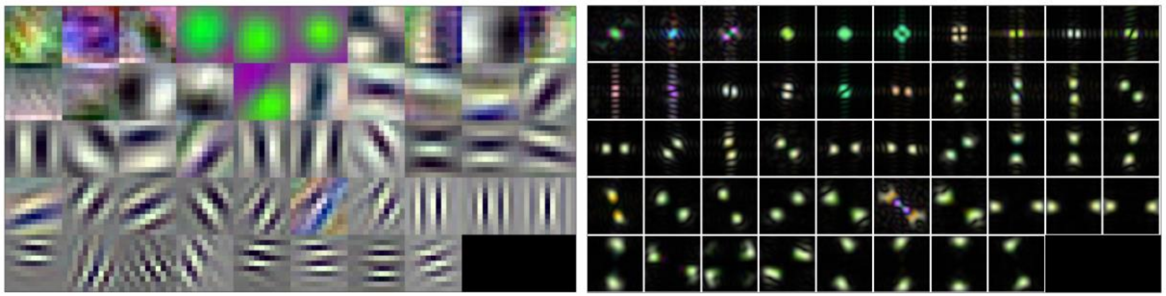
ImageNet Classification with Deep Convolutional Neural Networks

Part of Advances in Neural Information Processing Systems 25 (NIPS 2012)

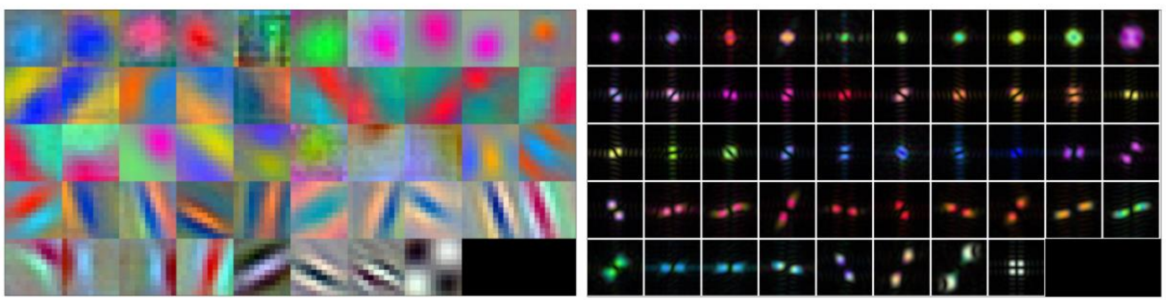
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

Serial Cascade: AlexNet

- Further analysis on AlexNet pretrained kernels (e.g. in 1st layer) reveals that convolution kernels encode features in different orientations, frequencies, and colors



(a) impulse resp. (color-agnostic) (b) filter resp. (color-agnostic)



(c) impulse resp. (color-specific) (d) filter resp. (color-specific)

Figure 1. AlexNet first layer convolution kernels for color-agnostic and color-specific sets.

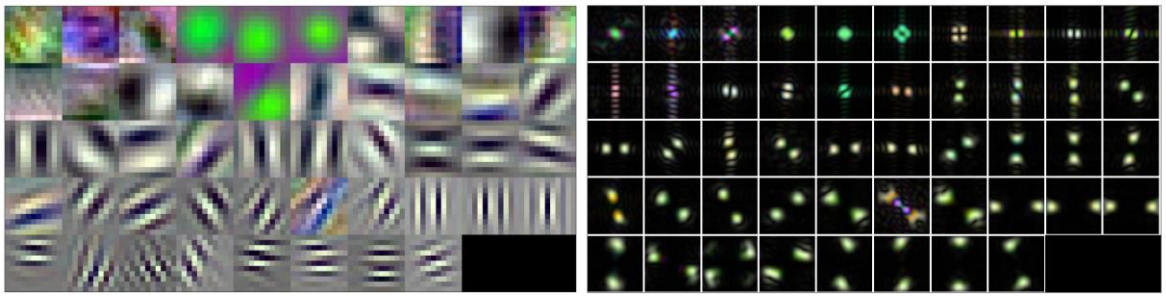
ImageNet Classification with Deep Convolutional Neural Networks

Part of Advances in Neural Information Processing Systems 25 (NIPS 2012)

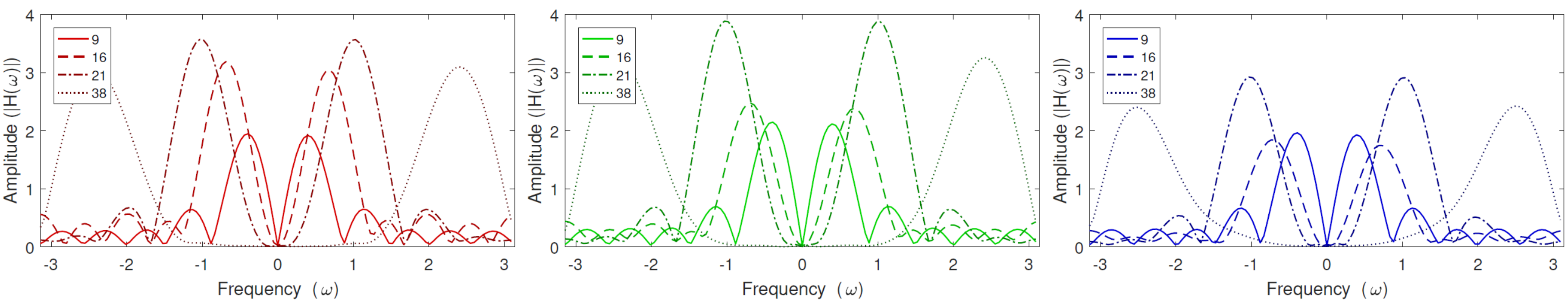
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

Serial Cascade: AlexNet

- Further analysis on AlexNet pretrained kernels (e.g. in 1st layer) reveals that convolution kernels encode features in different orientations, frequencies, and colors



(a) impulse resp. (color-agnostic) (b) filter resp. (color-agnostic)



(e) Red channel (f) Green channel (g) Blue channel

ImageNet Classification with Deep Convolutional Neural Networks

Part of Advances in Neural Information Processing Systems 25 (NIPS 2012)

Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

Serial Cascade: AlexNet

Figure 4. (Left) Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5). (Right) Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

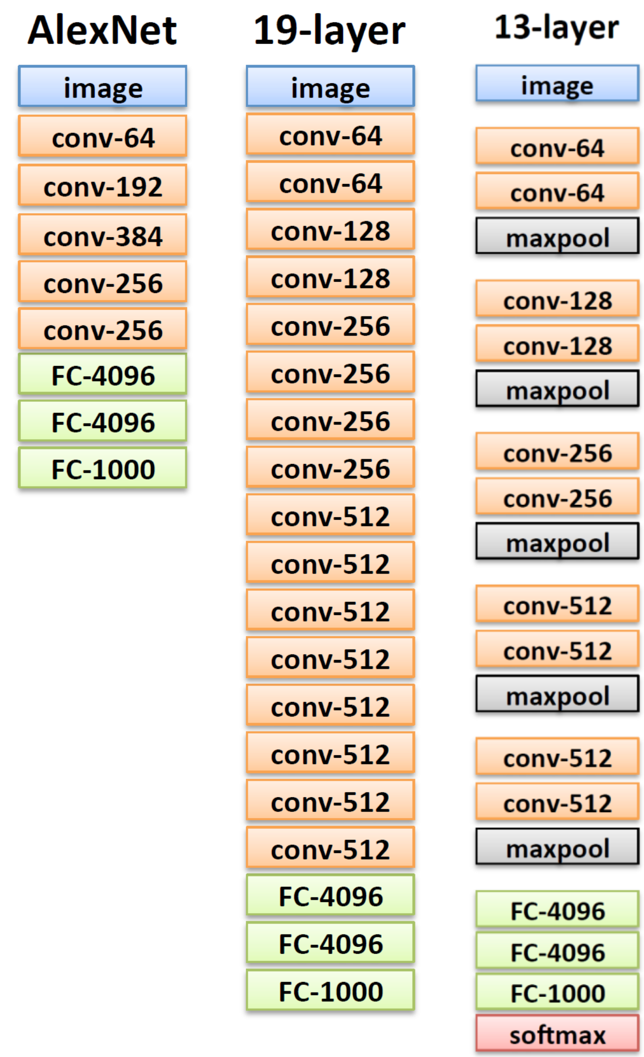
ImageNet Classification with Deep Convolutional Neural Networks

Part of Advances in Neural Information Processing Systems 25 (NIPS 2012)

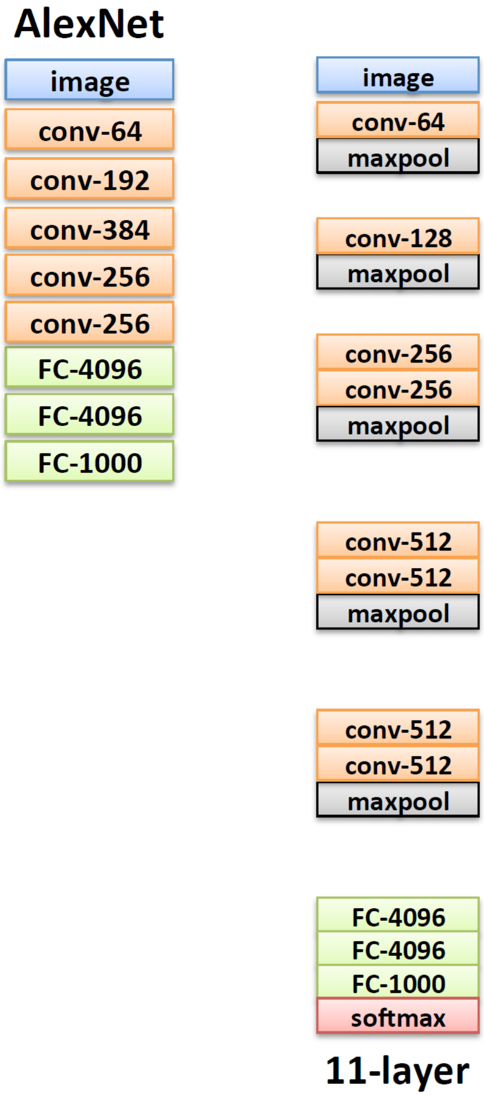
Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton

Serial Cascade: Going Deeper with VGG

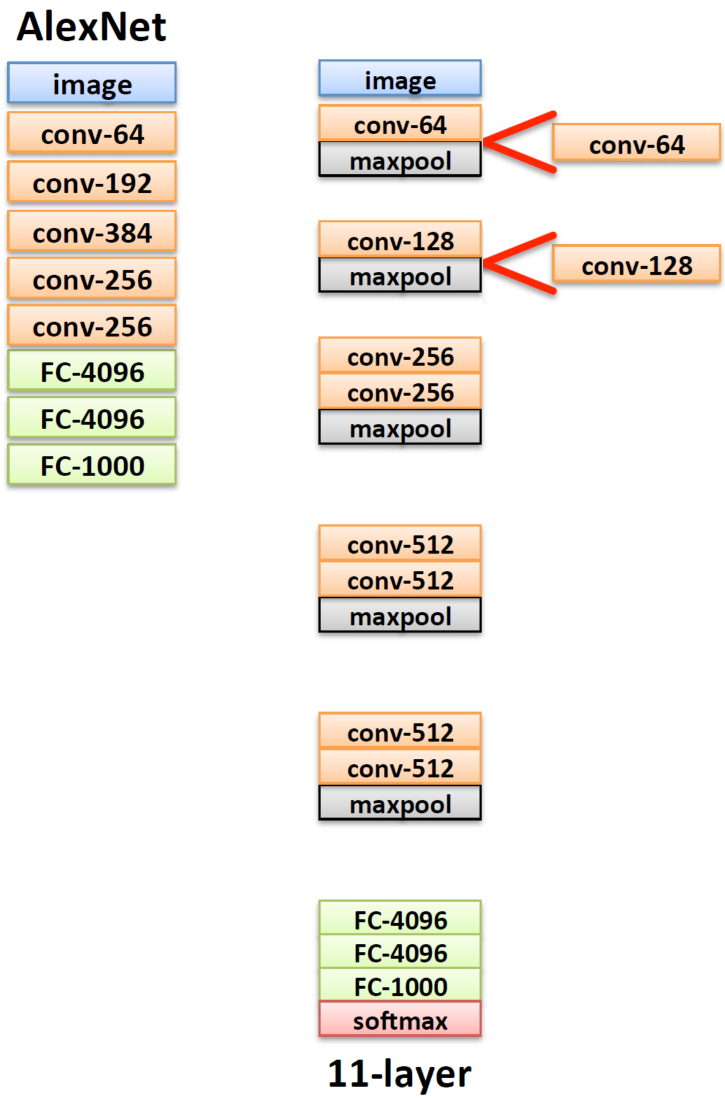
- Investigate the effect of the convolutional network depth
- Great boost is achieved by increasing #layers to 16-19
- Won ILSVRC2014 challenge
- Key design choice
  - 3x3 kernel size
  - Stack of conv layers w/o pooling
  - Conv stride=1 (no skipping)
  - ReLU activation
  - 5 Max-pooling (x2 downsampling)
  - 3 FC layers
- Later designs added Batch Normalization



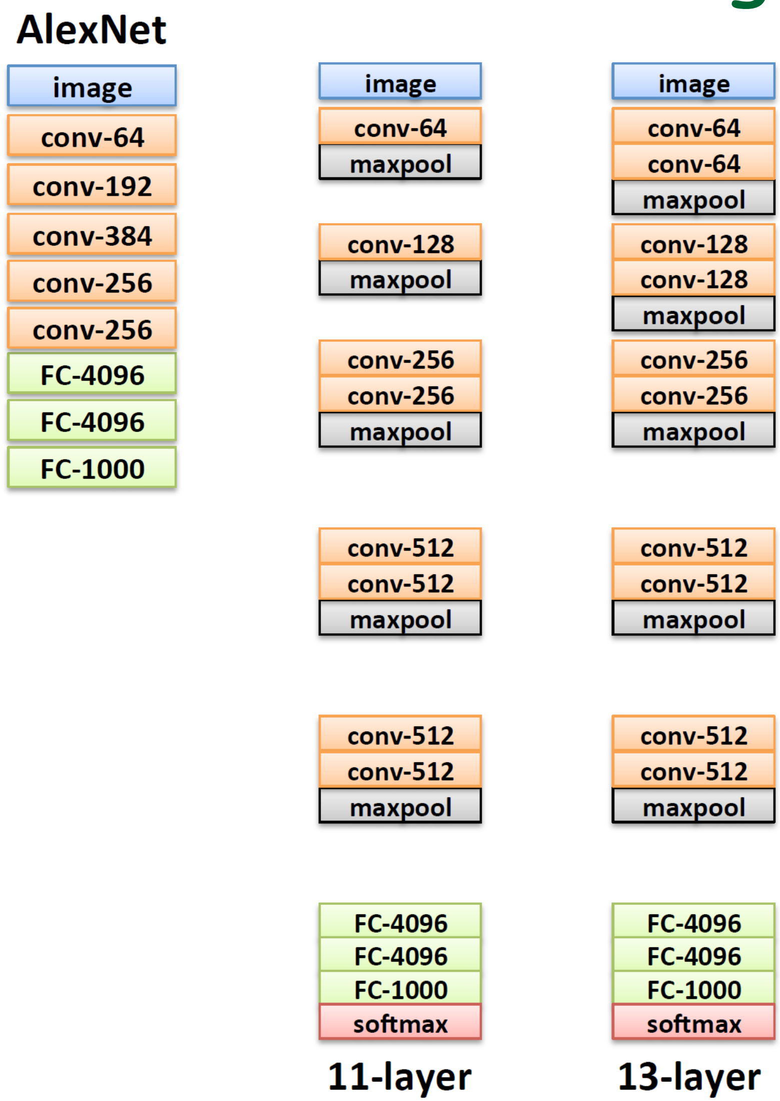
Serial Cascade: Going Deeper with VGG



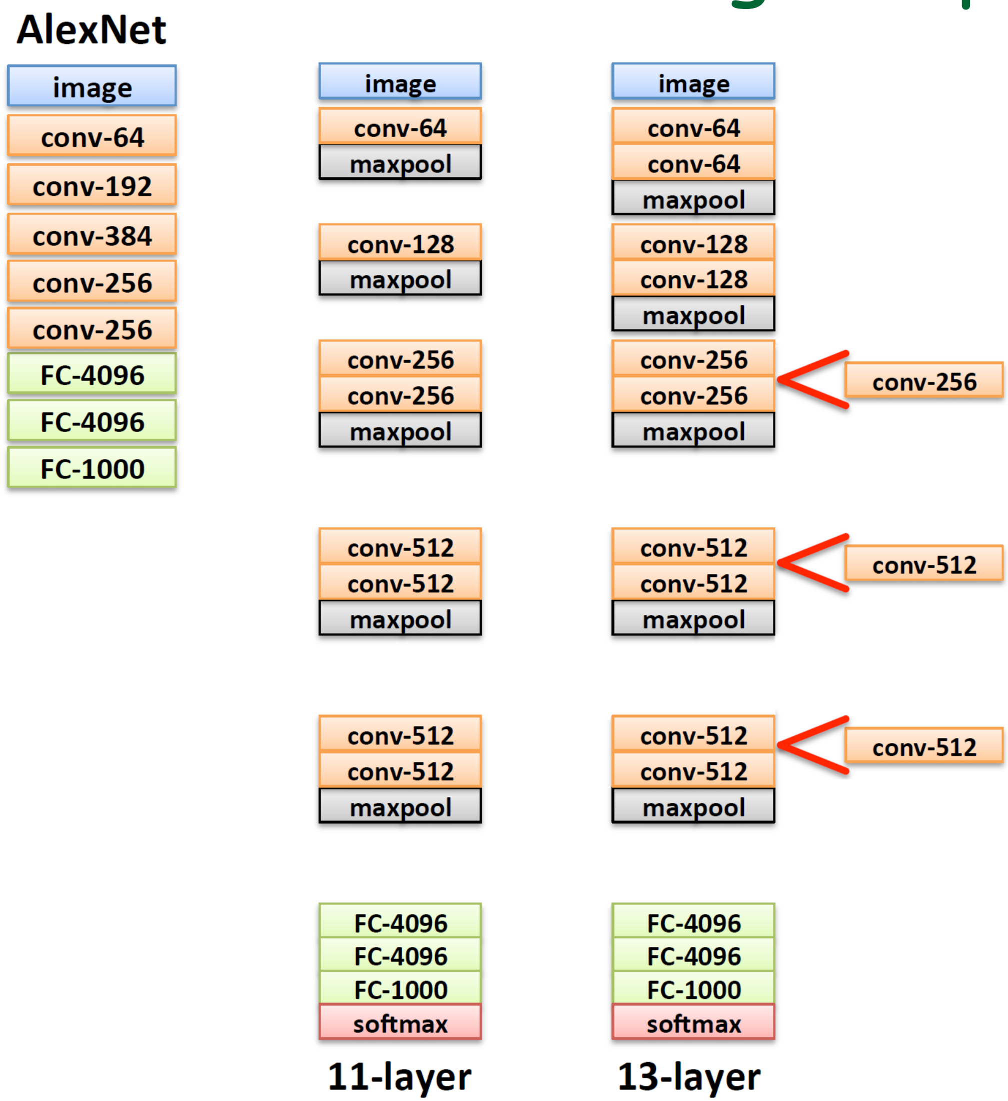
Serial Cascade: Going Deeper with VGG



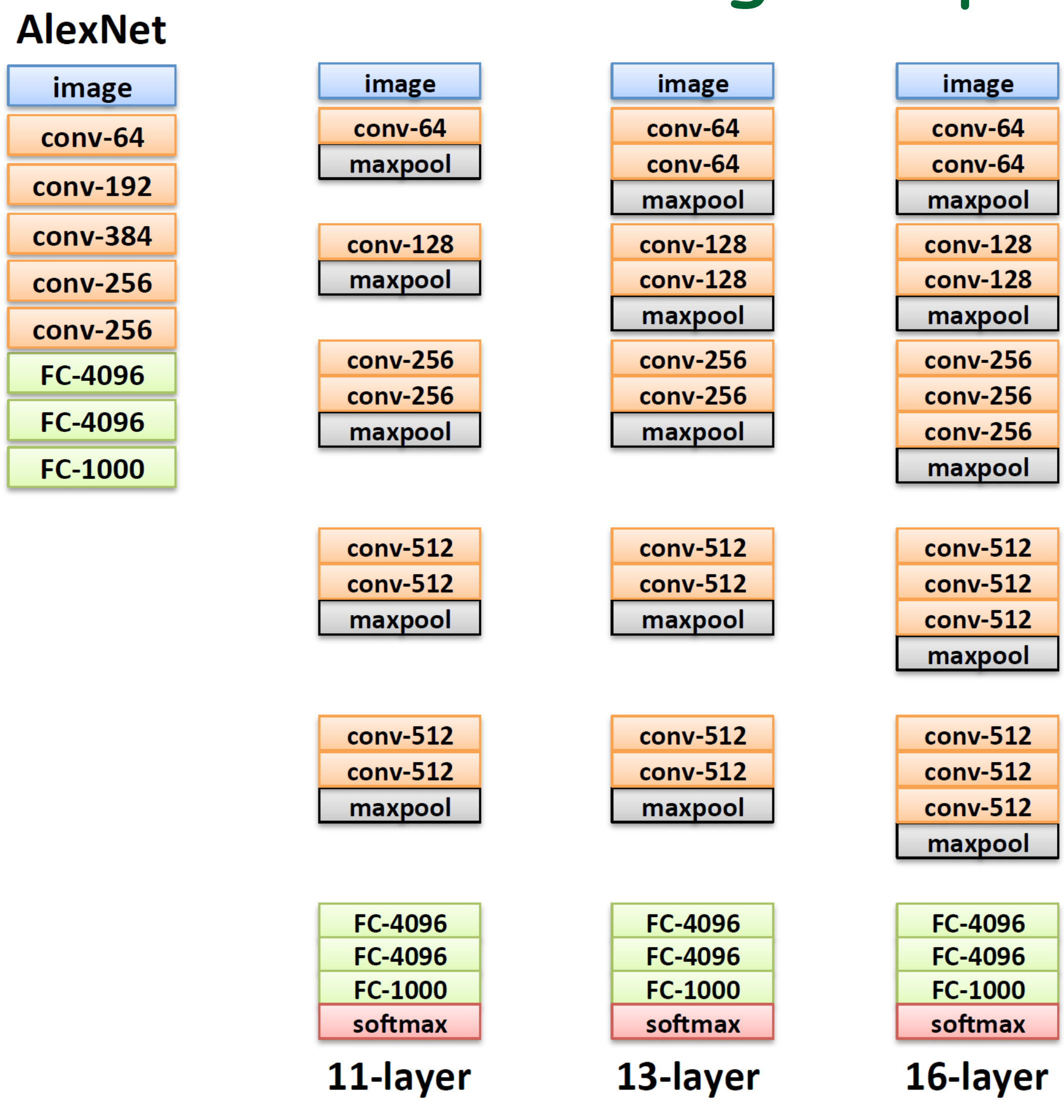
Serial Cascade: Going Deeper with VGG



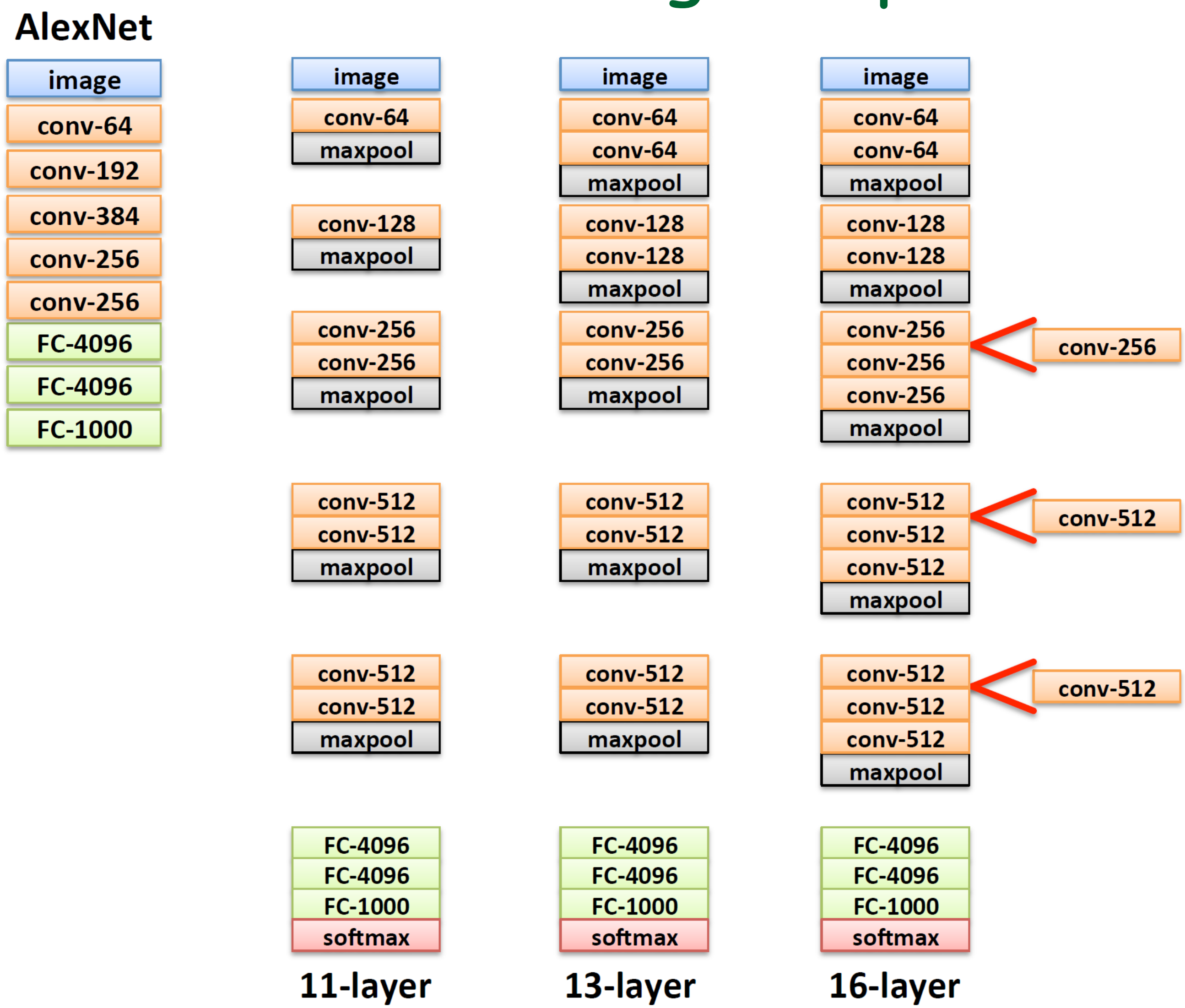
Serial Cascade: Going Deeper with VGG



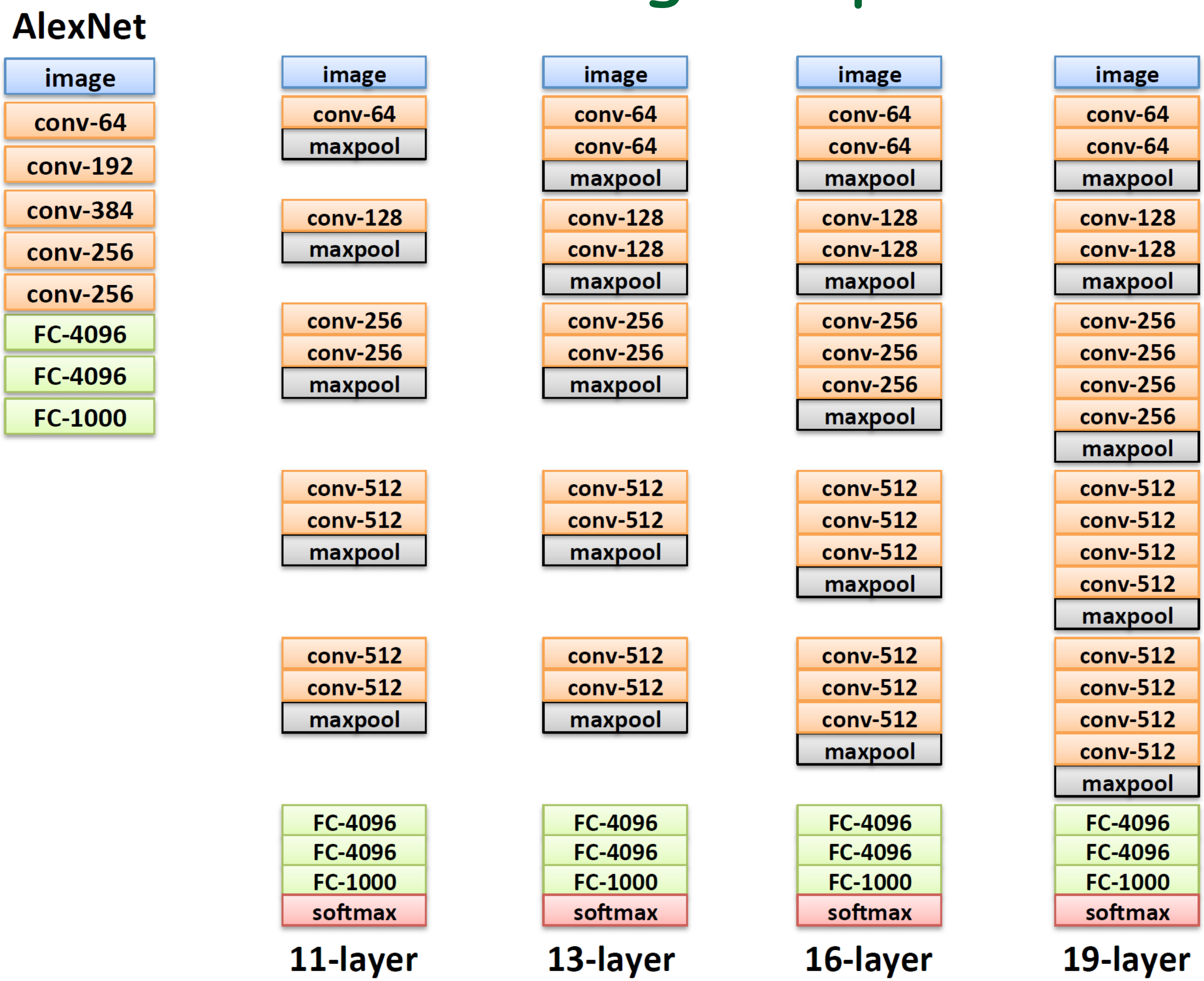
Serial Cascade: Going Deeper with VGG



Serial Cascade: Going Deeper with VGG



Serial Cascade: Going Deeper with VGG

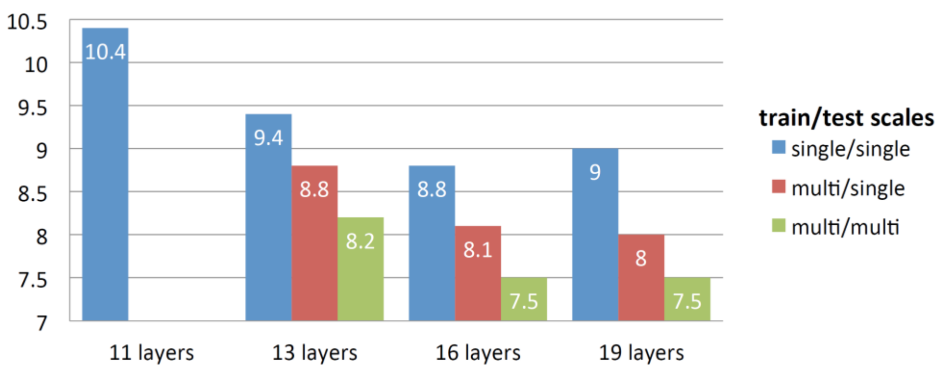


Serial Cascade: Going Deeper with VGG—Training Phase

- Input training image: fixed size of 224x224 crop
- Images have varying size, so upscale to e.g. 384x(N>384)
- Random crop 224x224
- Standard augmentation: random flip and RGB shift
- SGD-Momentum (next lecture)
- Regularization: dropout and weight decay
- Fast convergence (74 training epochs)
- Initialization (some sort of transfer-learning)
  - Deeper networks are prone to vanishing-gradients
  - 11-layer net: random initialization from N(0;0.01)
  - Deeper nets: Top & bottom layers initialized with 11-layer. Other layers: random initialization

Serial Cascade: Going Deeper with VGG— Testing Phase (ImageNet-1k)

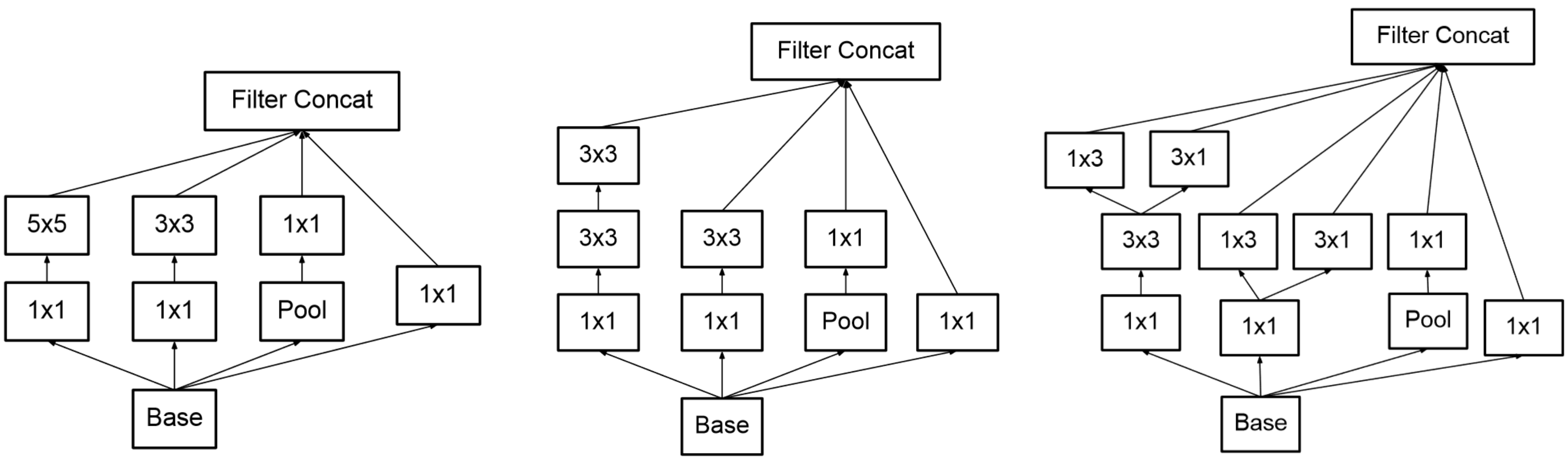
- Evaluation on variable size images
- Testing on multiple 224x224 crops [AlexNet]
- Multiple scales are tested (256xN, 384xN, 512xN) and class score averaged



- Error decreases with depth
- Using multiple scales is important
  - Multi-scale training outperforms single scale training =
  - Mutli-scale testing further improves the results

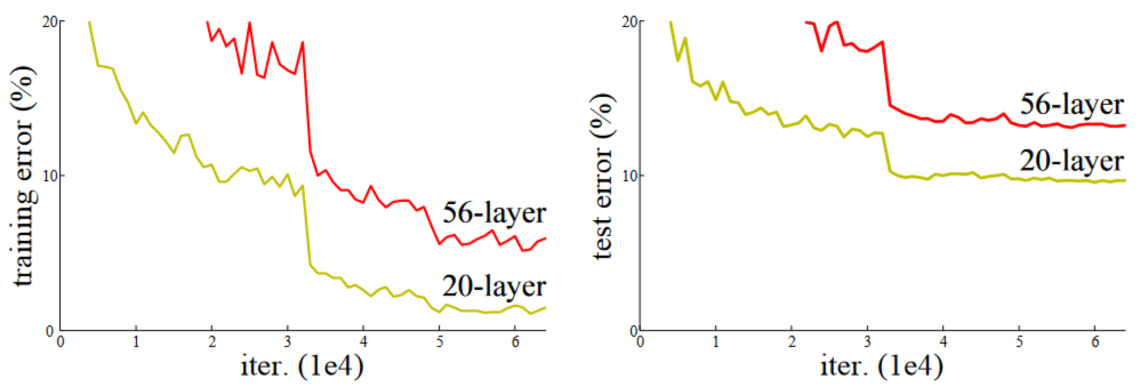
Serial/Parallel Cascade: Inception Net

- Multiple scales are encoded in parallel and cascaded to next layers



Residual Connection: ResNet Architecture

- Is learning better networks as easy as stacking more layers?
- Obstacle: deeper networks are difficult to train because of the notorious problem of vanishing/exploding gradients
- Early solutions were proposed by introducing
  - normalized initialization
  - intermediate normalization layer
- Still accuracy gets saturated with increasing depth and then degrades rapidly (this is not caused by overfitting!)
- Adding more layers leads to higher training error



Residual Connection: ResNet Architecture

- **Residual learning:** instead of hoping each few stacked layers directly fit a desired underling mapping, we explicitly let these layers fit a residual mapping
- Define underlying forward mapping by H(x) := F(x) + x

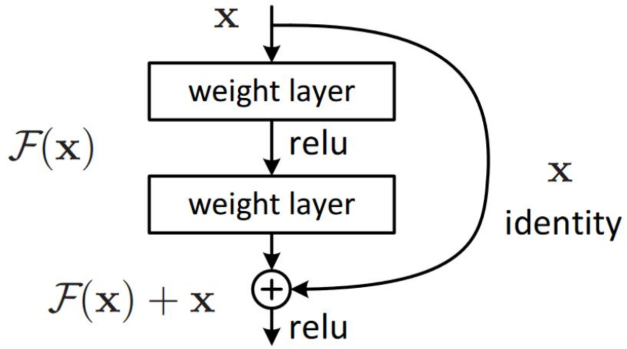


Figure 2. Residual learning: a building block.

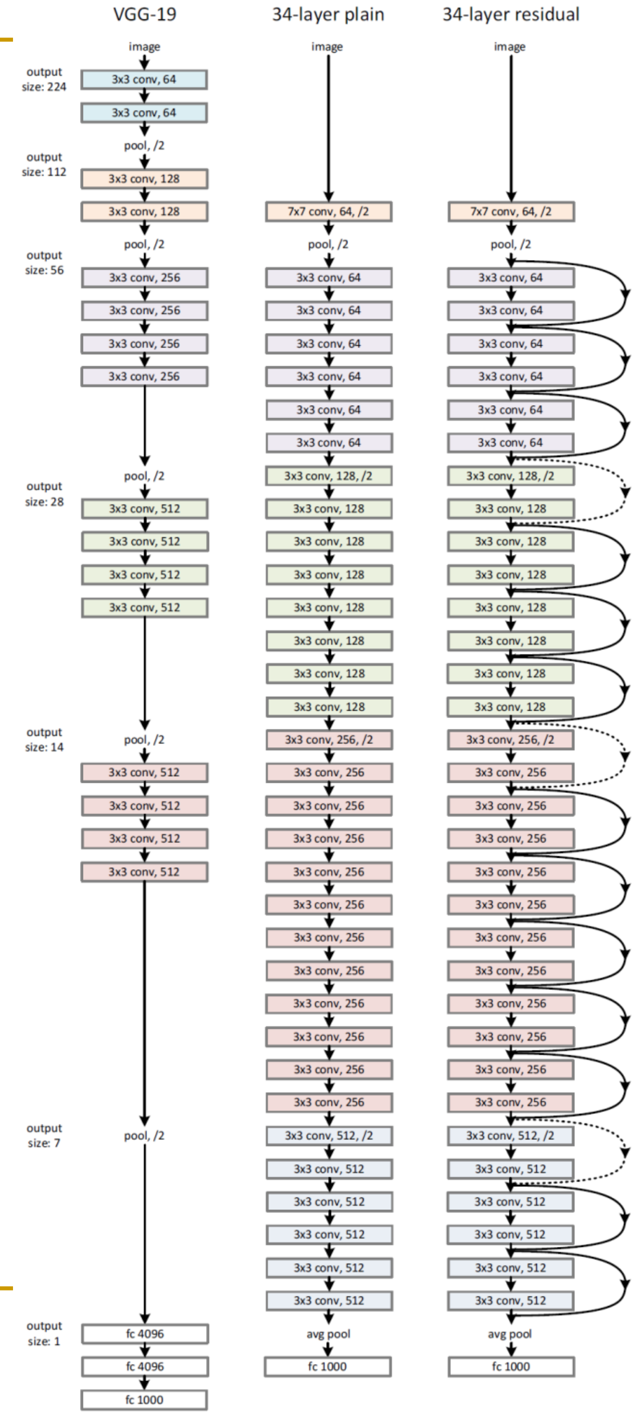
- Corresponding gradient back-propagation:

$$\frac{\partial \epsilon}{\partial x} = \frac{\partial \epsilon}{\partial H(x)} \cdot \frac{\partial H(x)}{\partial x} = \frac{\partial \epsilon}{\partial H(x)} \cdot \left[\frac{\partial F}{\partial x} + 1\right] = \frac{\partial \epsilon}{\partial H(x)} \cdot \frac{\partial F}{\partial x} + \frac{\partial \epsilon}{\partial H(x)}$$

- Gradient from output layer transfers directly to the input layer and avoids vanishing

Residual Connection: ResNet

- Plain baseline is inspired by VGG nets
- Shortcuts introduced on plain baseline
- Same number of filters are used for the same output feature map size
- If the feature map size is halved, the number of filters is doubled
- Down-sampling by stride=2
- Network ends with global-average-pooling
- 1000-way FC layer with softmax
- ResNet34 has 3.6 BFLOPS (18% of VGG19 i.e. 19.6 BFLOPS)



Residual Connection: ResNet Architecture

- Different ResNet architectures of ResNet18, ResNet34, ResNet50, ResNet101, ResNet152

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Residual Connection: ResNet Architecture

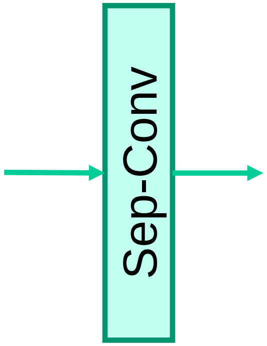
- ResNet performance on ImageNet

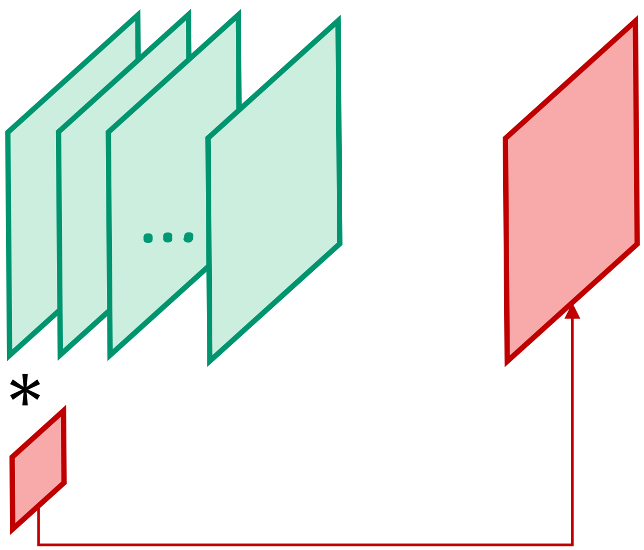
model	top-1 err.	top-5 err.
VGG-16 [40]	28.07	9.33
GoogLeNet [43]	-	9.15
PReLU-net [12]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

Table 3. Error rates (%, **10-crop** testing) on ImageNet validation. VGG-16 is based on our test. ResNet-50/101/152 are of option B that only uses projections for increasing dimensions.

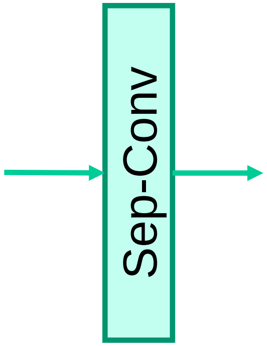
Depthwise/Separable Convolutions

- Keep input channel convolution separate from mapping to output feature map

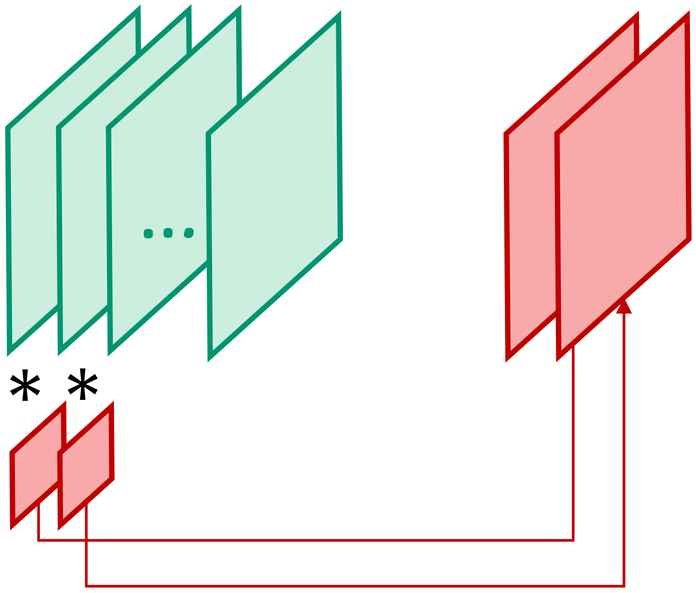




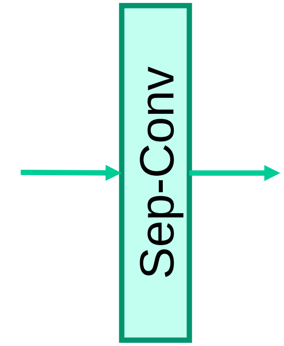
Depthwise/Separable Convolutions



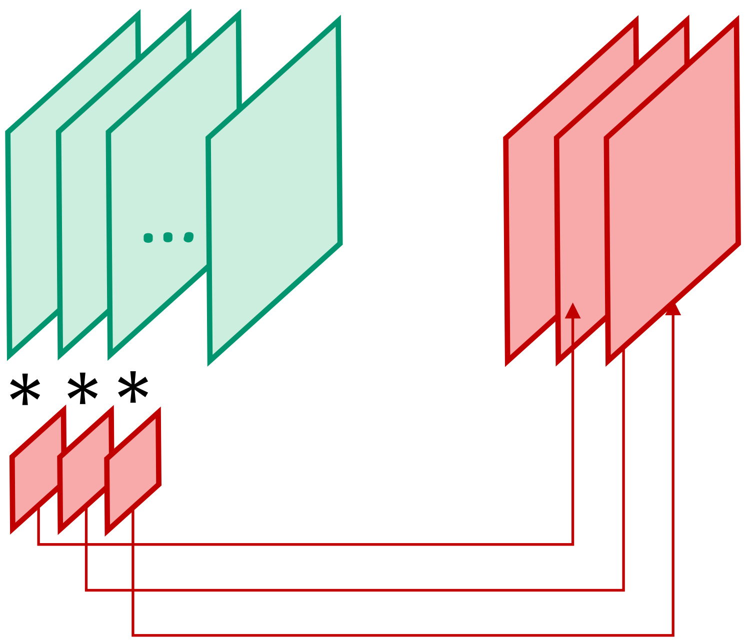
- Keep input channel convolution separate from mapping to output feature map



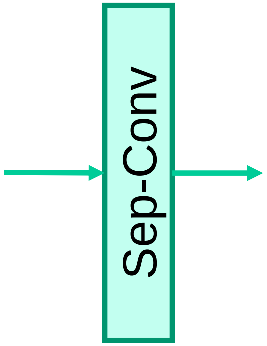
Depthwise/Separable Convolutions



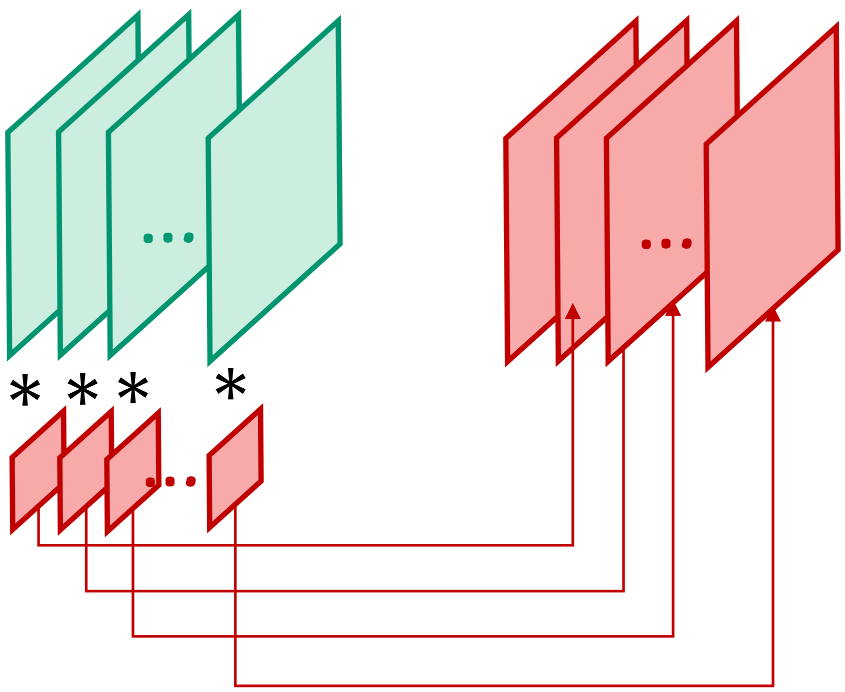
- Keep input channel convolution separate from mapping to output feature map



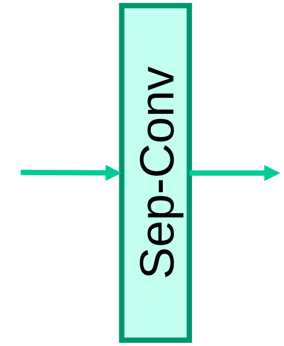
Depthwise/Separable Convolutions



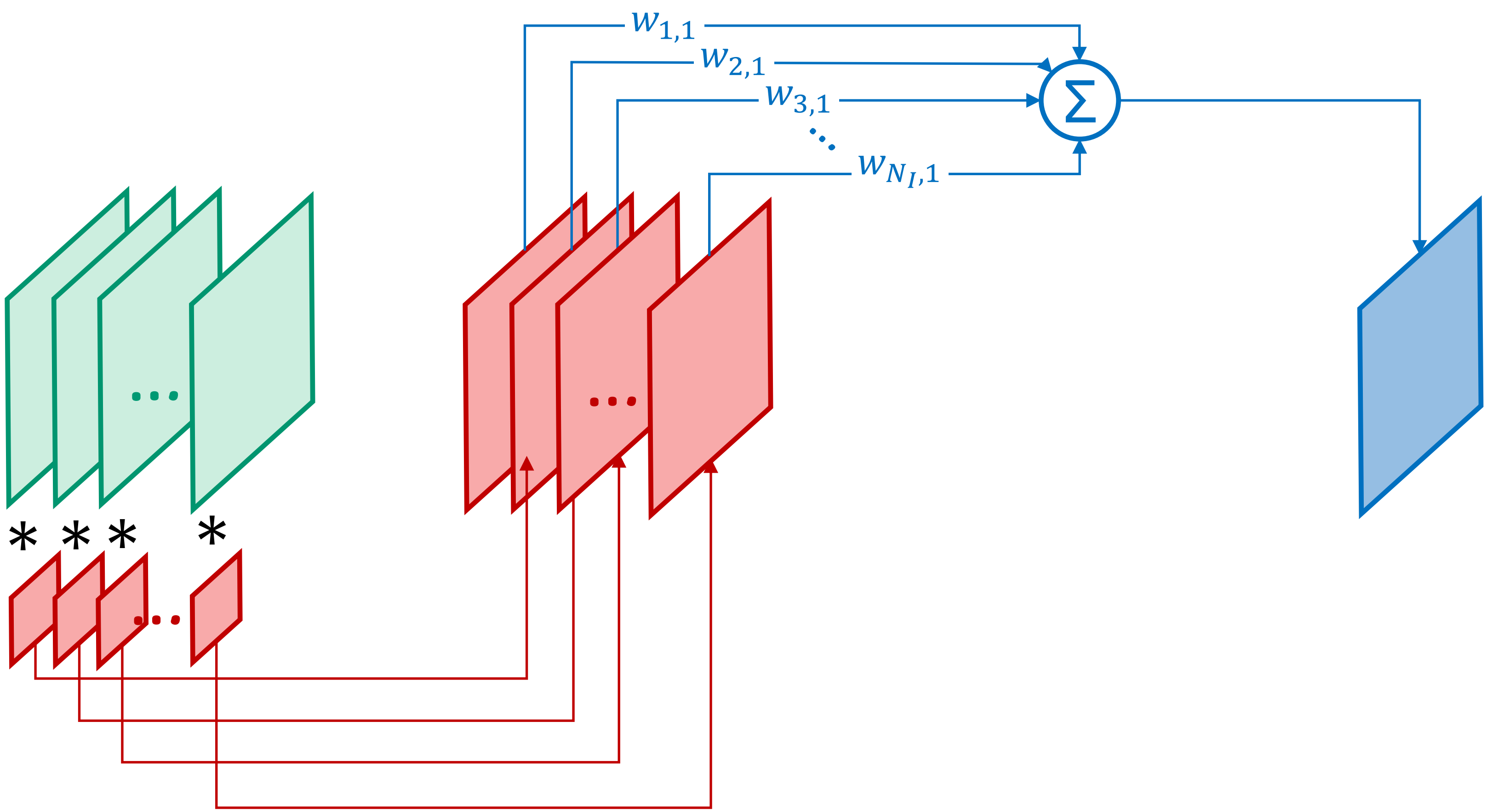
- Keep input channel convolution separate from mapping to output feature map



Depthwise/Separable Convolutions

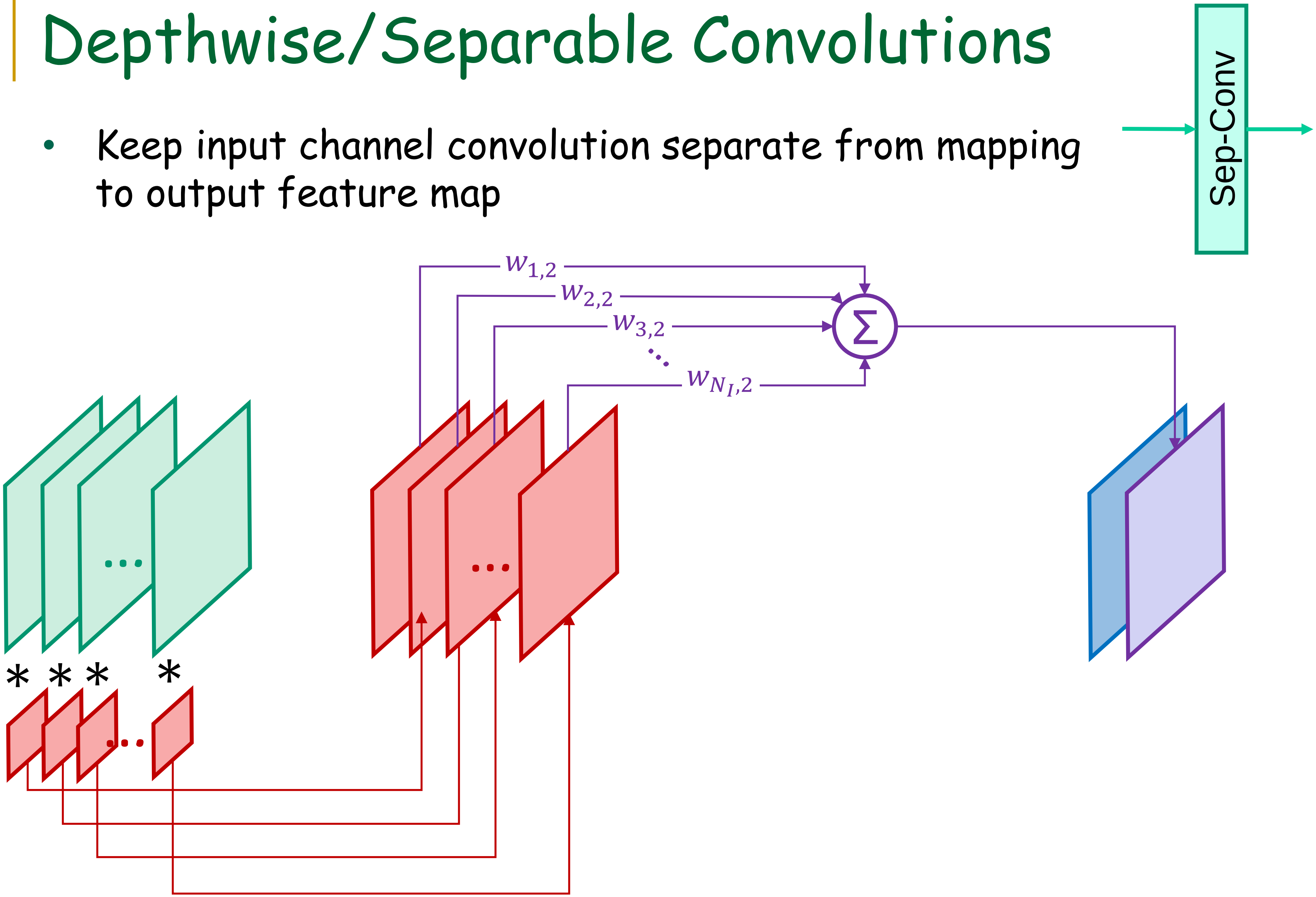


- Keep input channel convolution separate from mapping to output feature map



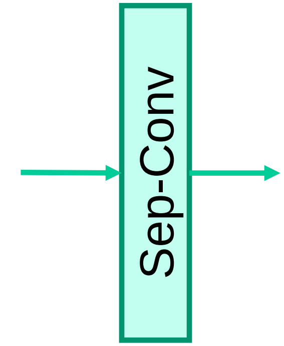
Depthwise/Separable Convolutions

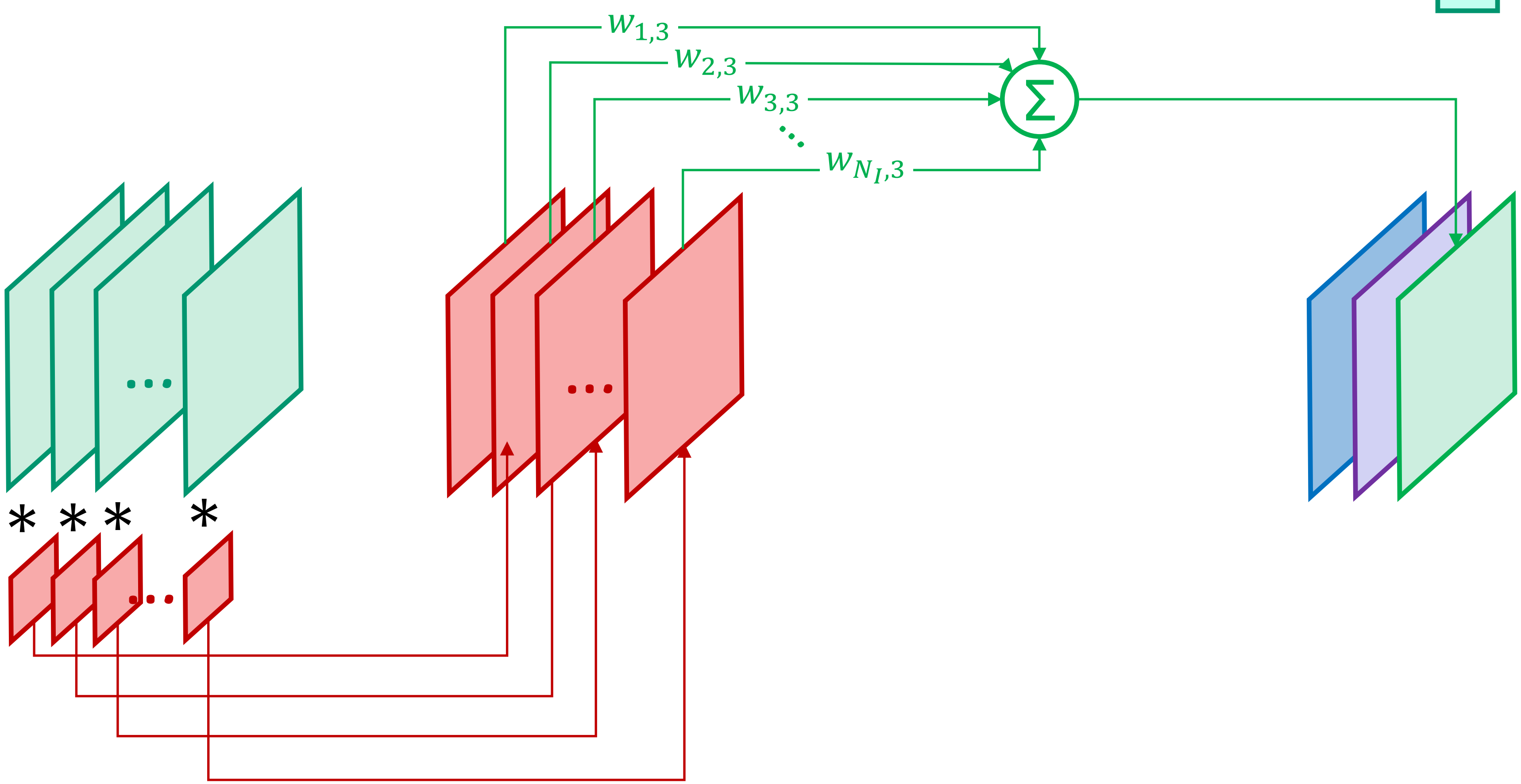
- Keep input channel convolution separate from mapping to output feature map



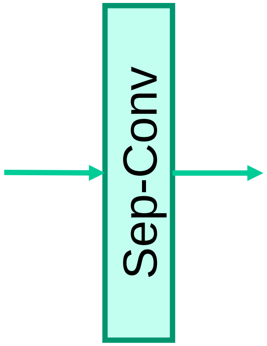
Depthwise/Separable Convolutions

- Keep input channel convolution separate from mapping to output feature map

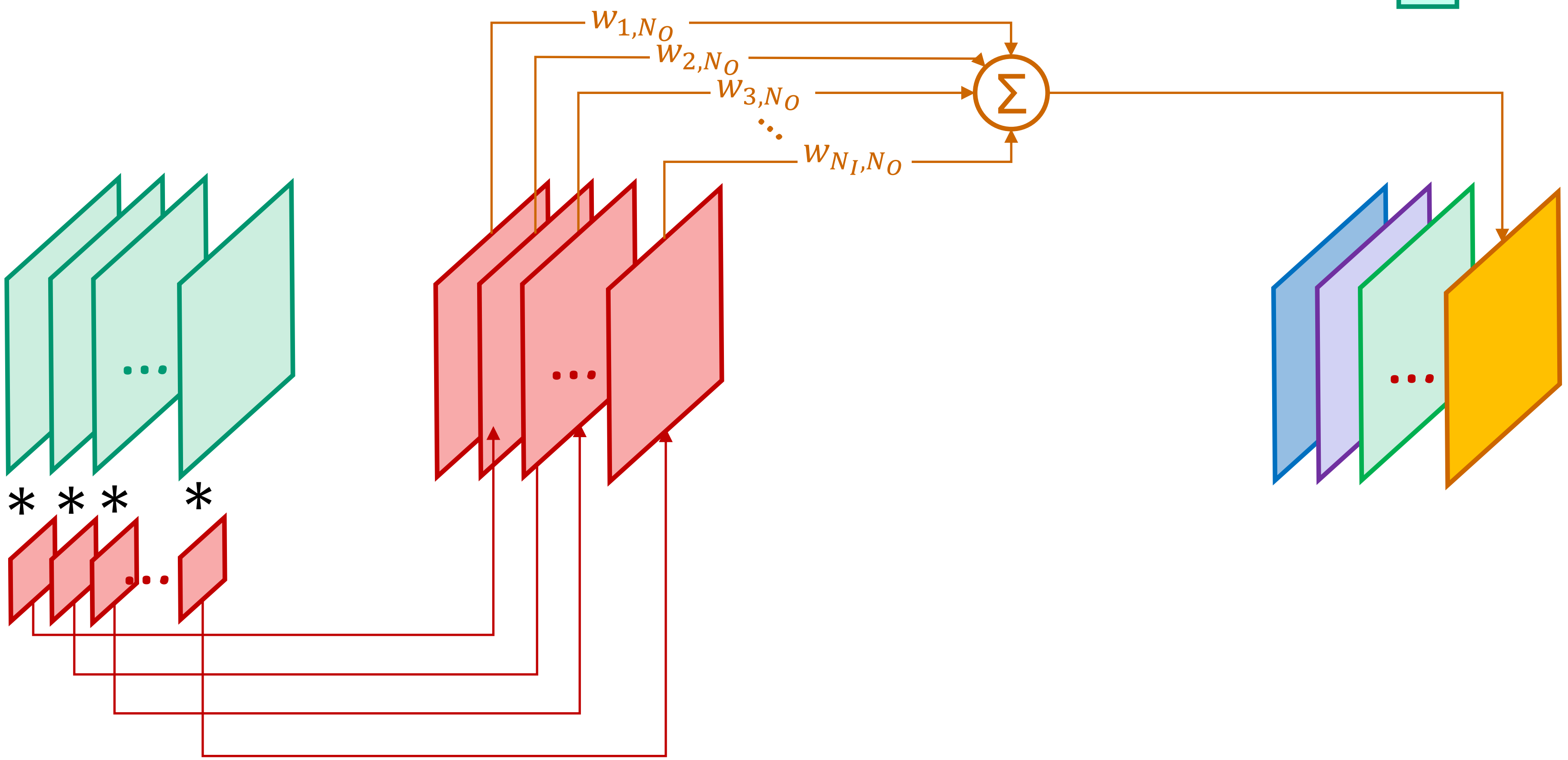




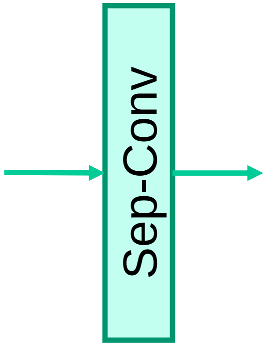
Depthwise/Separable Convolutions



- Keep input channel convolution separate from mapping to output feature map

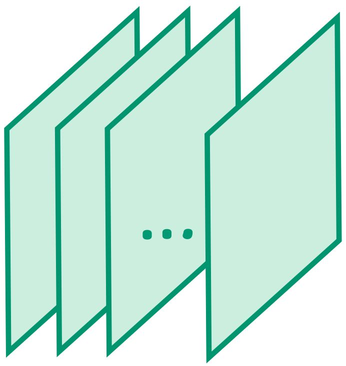


Depthwise/Separable Convolutions

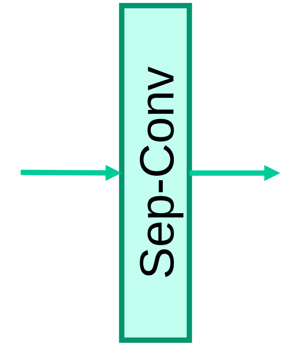


- Keep input channel convolution separate from mapping to output feature map

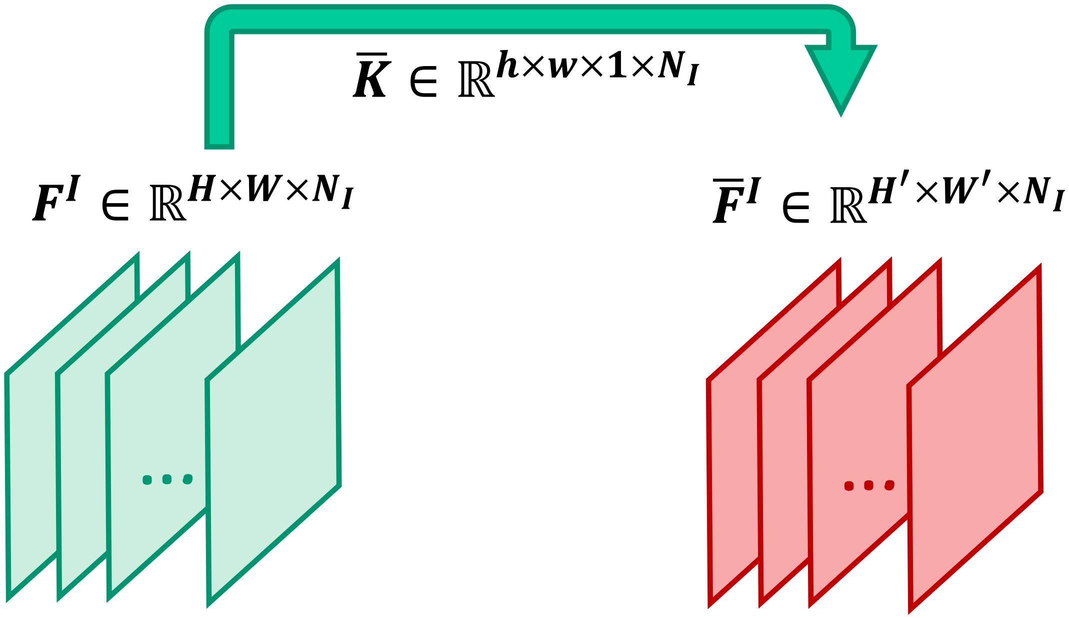
$\boldsymbol{F^I} \in \mathbb{R}^{\boldsymbol{H} \times \boldsymbol{W} \times \boldsymbol{N_I}}$



Depthwise/Separable Convolutions

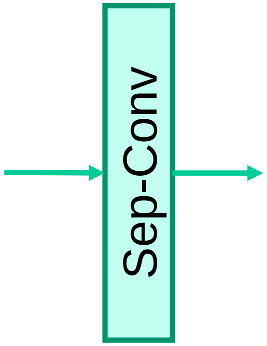


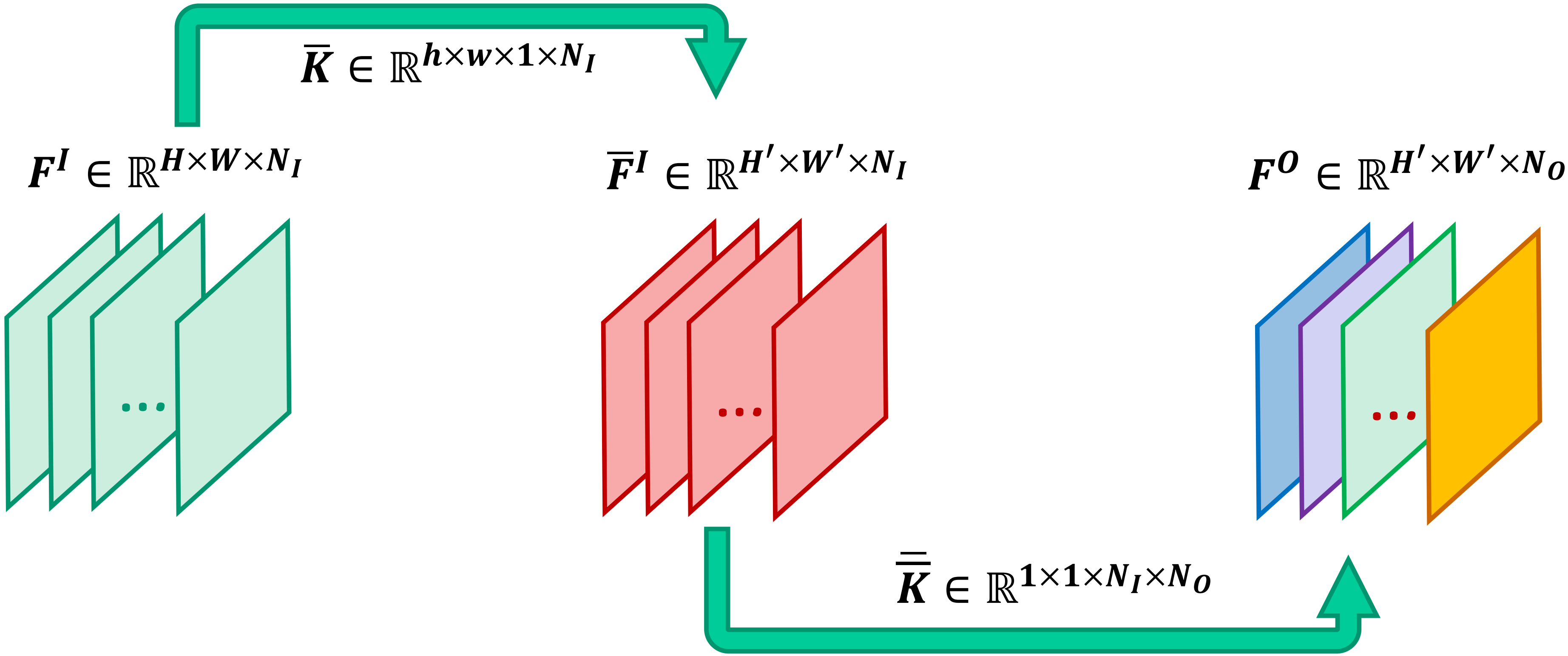
- Keep input channel convolution separate from mapping to output feature map



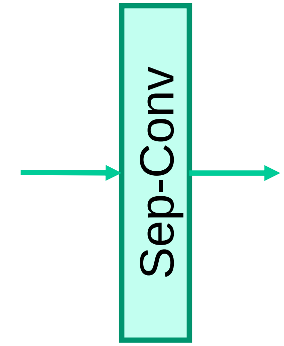
Depthwise/Separable Convolutions

- Keep input channel convolution separate from mapping to output feature map





Depthwise/Separable Convolutions



- What are the benefits?
- Standard convolution has the computation cost of

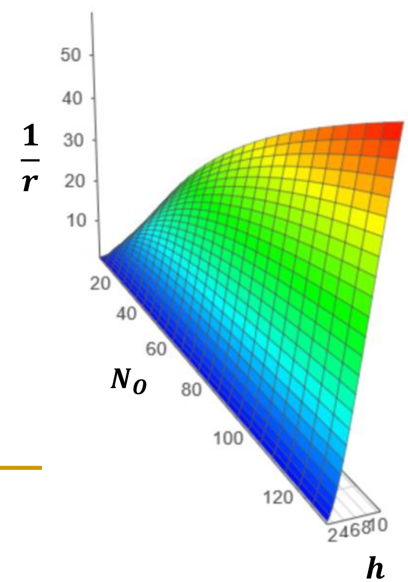
$$O(\underbrace{h \cdot w \cdot N_I \cdot N_O}_{\text{Convolution Kernel}} \cdot \underbrace{H \cdot W}_{\text{Input Feature Map}})$$

- Depthwise-separable convolution has the computation cost of

$$O(\underbrace{h \cdot w \cdot N_I}_{\text{Conv Kernel-1}} \cdot \underbrace{H \cdot W}_{\text{Input Feature}} + \underbrace{N_I \cdot N_O}_{\text{Conv Kernel-2}} \cdot \underbrace{H \cdot W}_{\text{Input Feature}})$$

- Reduction in computation ratio

$$r = \frac{hwN_IHW + N_IN_OHW}{hwN_IN_OHW} = \frac{1}{N_O} + \frac{1}{hw}$$



Depthwise/Separable Convolutions

Sep-Conv

- Activation and Batch-Normalization are used in between
- 1x1 convolution also referred by *Pointwise Convolution*

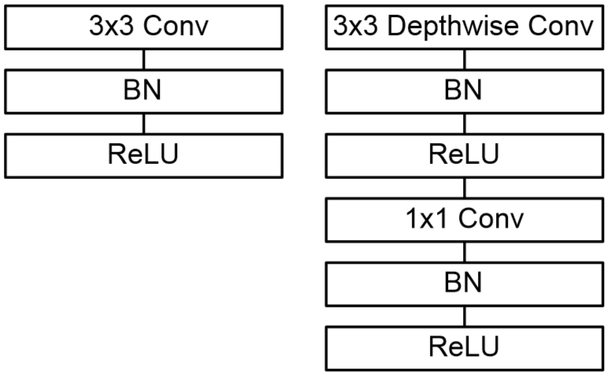


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

MobileNet by Depthwise Separable Convolutions

- All layers are followed by BN and ReLU
- MobileNet has 28 separate layers

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

MobileNet by Depthwise Separable Convolutions

- Significant parameter reduction while maintaining performance accuracy

Table 12. Face attribute classification using the MobileNet architecture. Each row corresponds to a different hyper-parameter setting (width multiplier α and image resolution).

Width Multiplier / Resolution	Mean AP	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	88.7%	568	3.2
0.5 MobileNet-224	88.1%	149	0.8
0.25 MobileNet-224	87.2%	45	0.2
1.0 MobileNet-128	88.1%	185	3.2
0.5 MobileNet-128	87.7%	48	0.8
0.25 MobileNet-128	86.4%	15	0.2
Baseline	86.9%	1600	7.5

Squeeze-Excitation Block

- The idea is to boost the representation power of the network
- SE: channel relationships are adaptively recalibrated in channel-wise feature response by explicit modelling interdependencies between channels

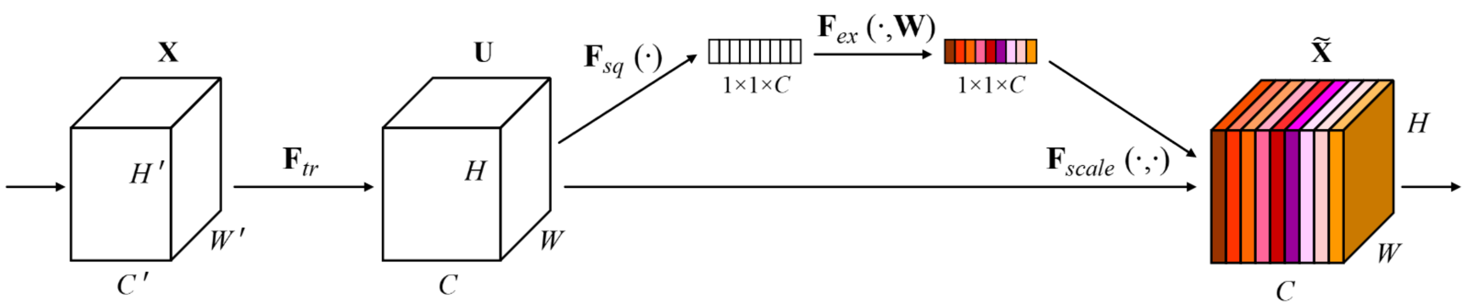


Figure 1: A Squeeze-and-Excitation block.

Squeeze-Excitation Block

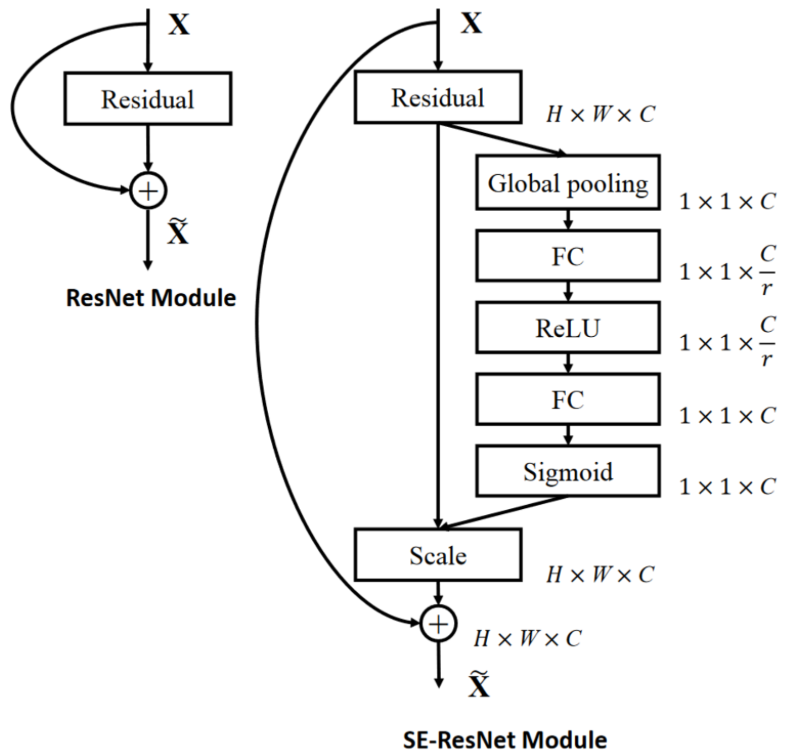


Figure 3: The schema of the original Residual module (left) and the SE-ResNet module (right).

Today

1. Applications
2. Backbone Models
    1. Serial Cascading
    2. Serial/Parallel Cascading
    3. Residual Connection
    4. Depthwise/Separable Convolutions
    5. Squeeze-Excitation