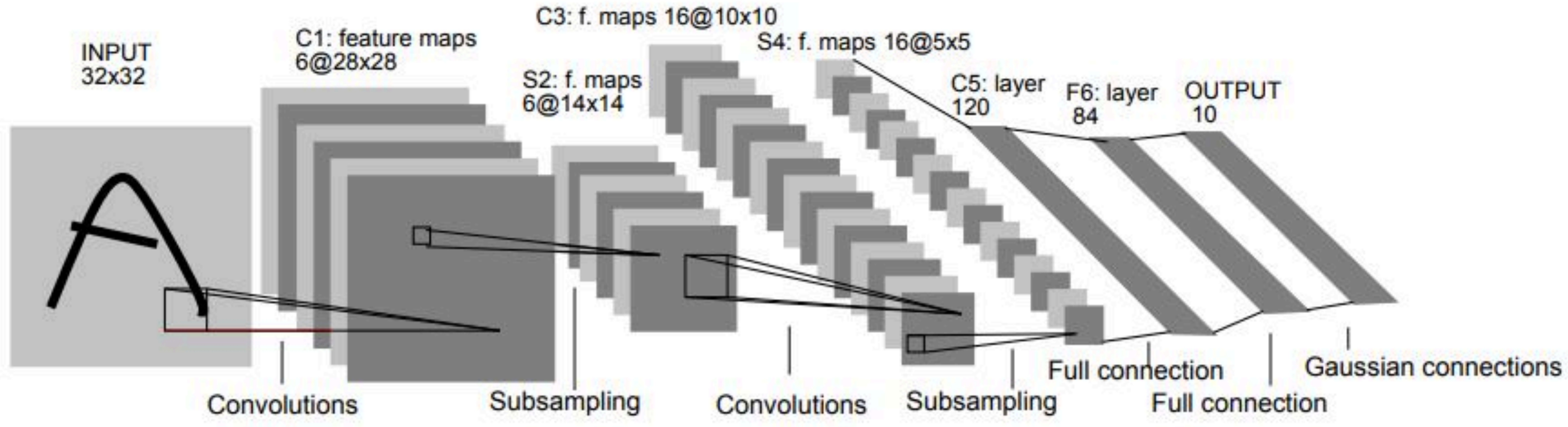
Convolutional Neural Networks

Cunjian Chen
Department of Computer Science and Engineering
Michigan State University
1/8/2020

Background

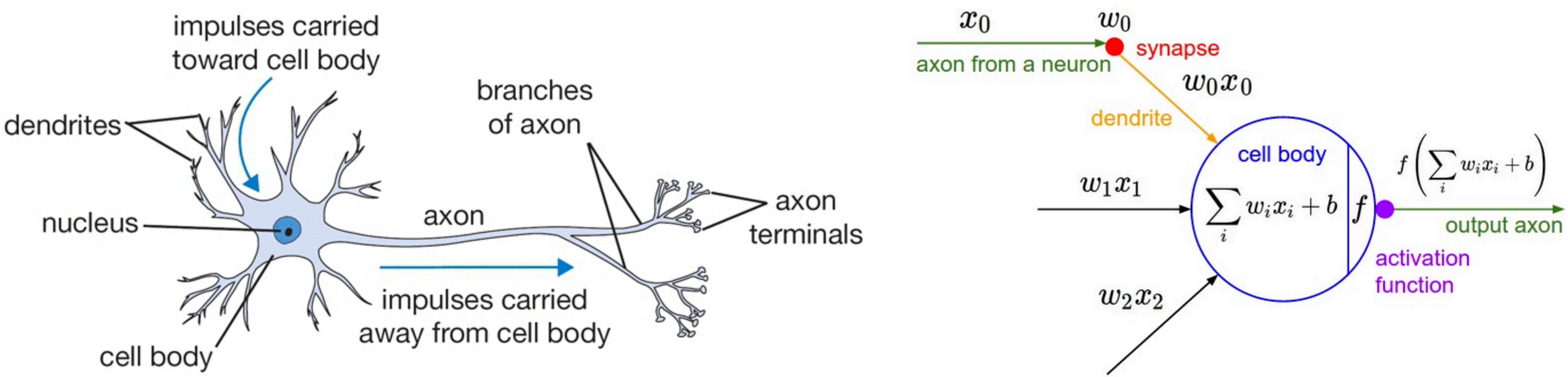
- A Convolutional Neural Network (CNN) consists of one or more convolutional layers (often with a subsampling step and an activation step) and then followed by one or more fully connected layers as observed in a standard multilayer neural network.

LeNet-5



LeCun et al., Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 1998

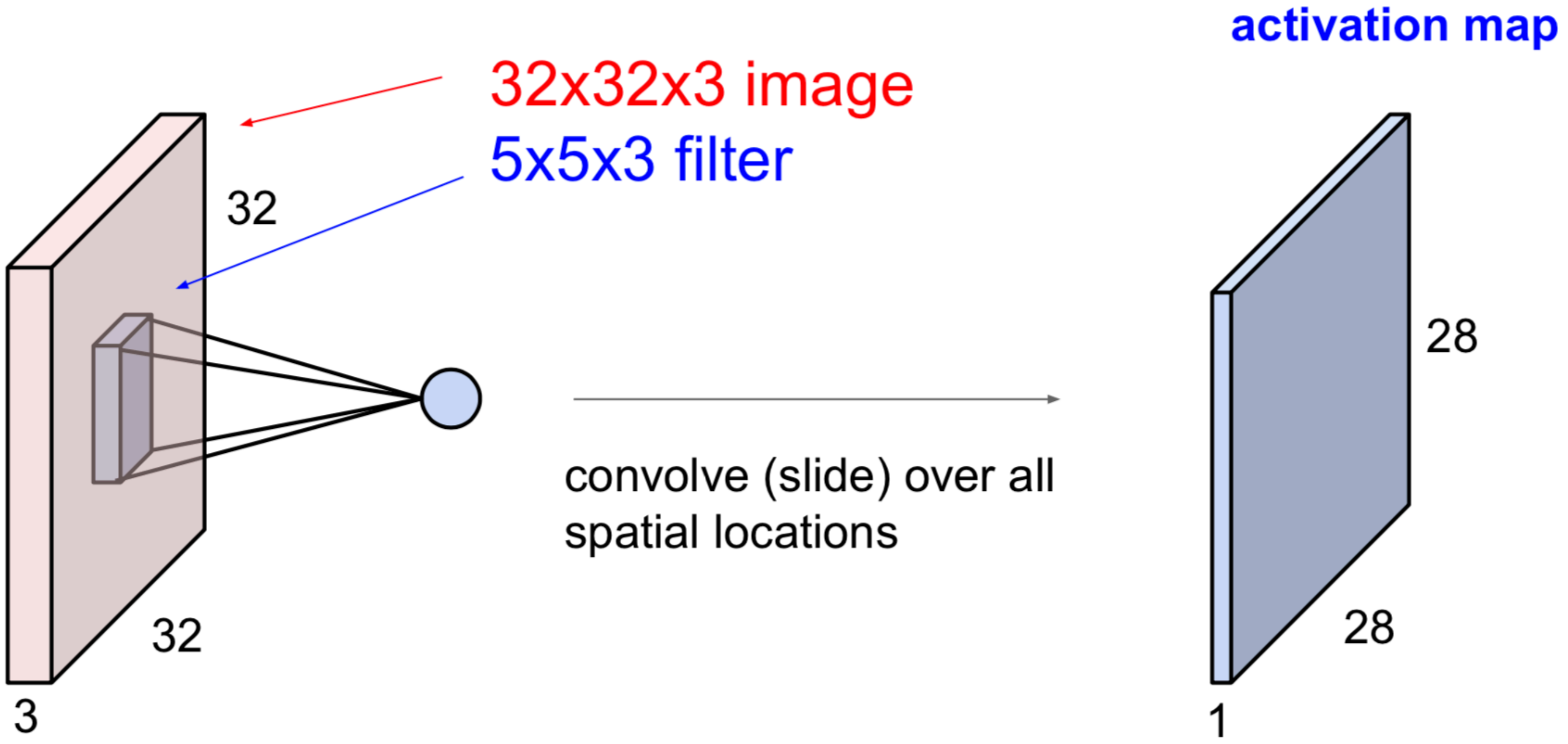
Background



A cartoon drawing of a biological neuron (left) and its mathematical model (right).

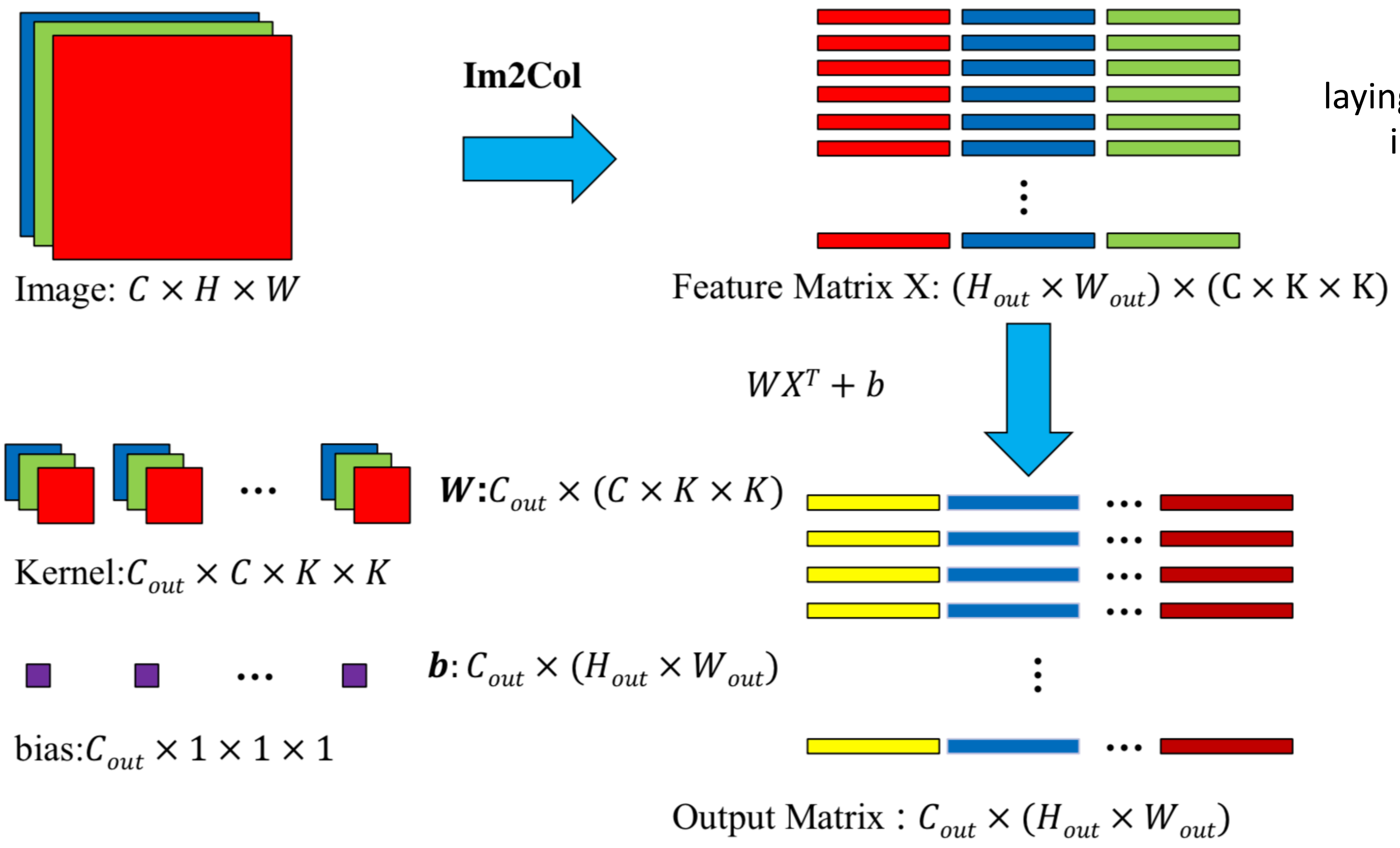
CS231n: Convolutional Neural Networks for Visual Recognition, Stanford, 2019

Convolution Layer



CS231n: Convolutional Neural Networks for Visual Recognition, Stanford, 2019

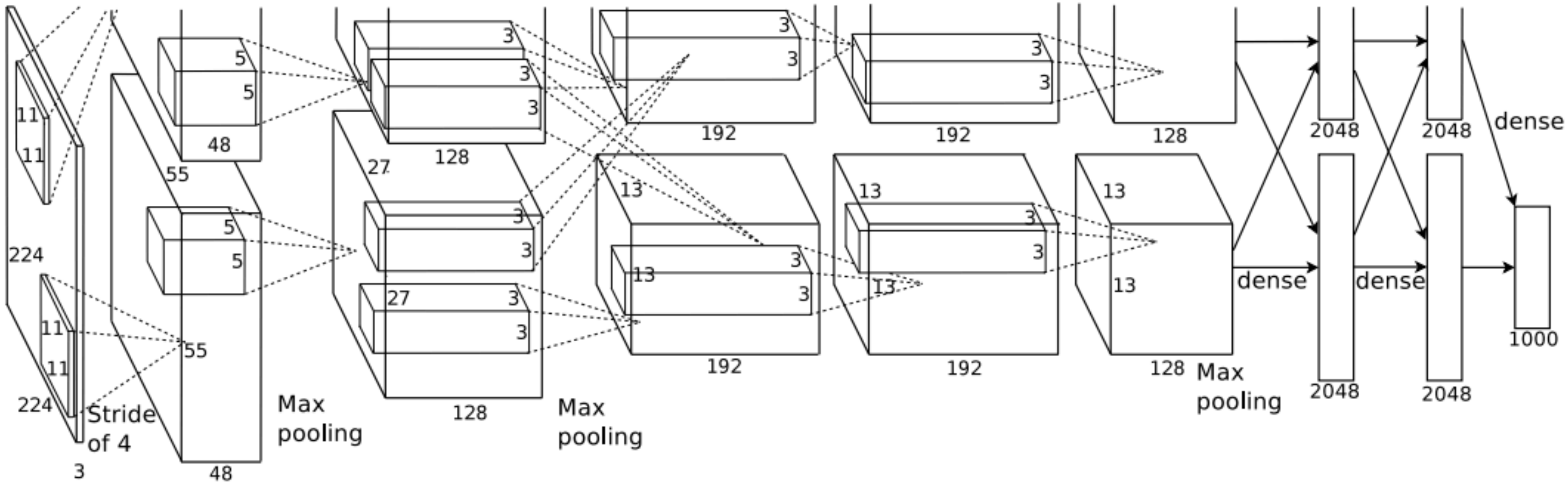
Convolution Layer



laying out all patches into a matrix

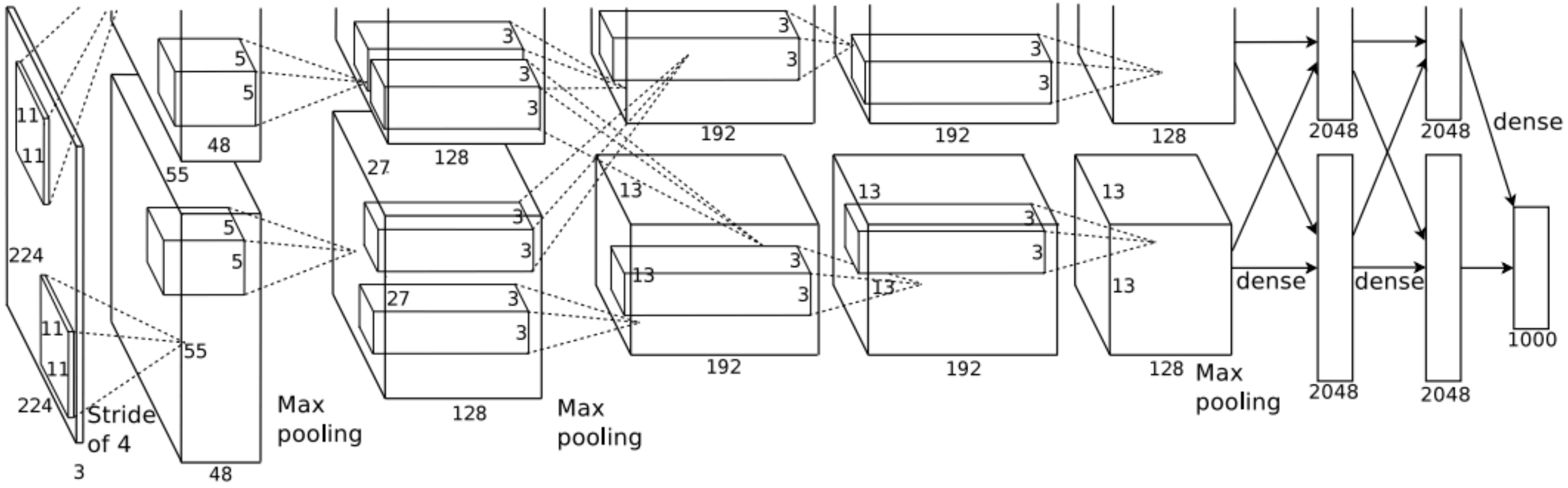
Figure courtesy of Xin Liu@CAS

First Breakthrough-AlexNet (2012)



Alex et al. ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

First Breakthrough-AlexNet (2012)

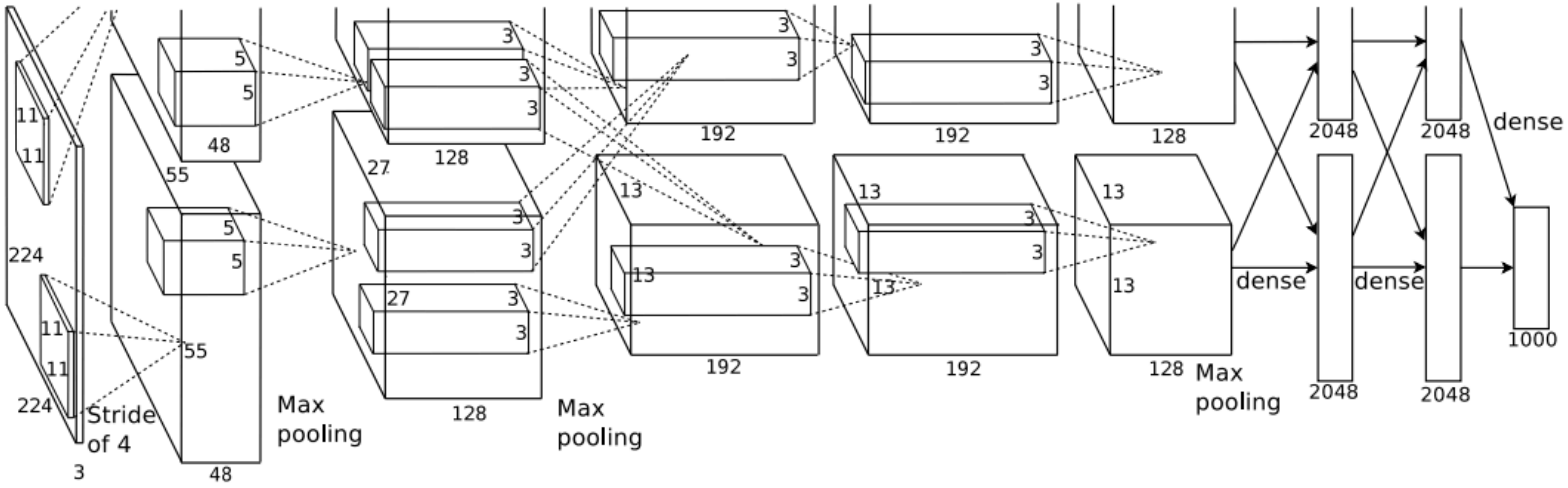


Input: 227*227*3

Conv1: 96 11*11*3 filters, with stride 4

Alex et al. ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

First Breakthrough-AlexNet (2012)



Input: 227*227*3

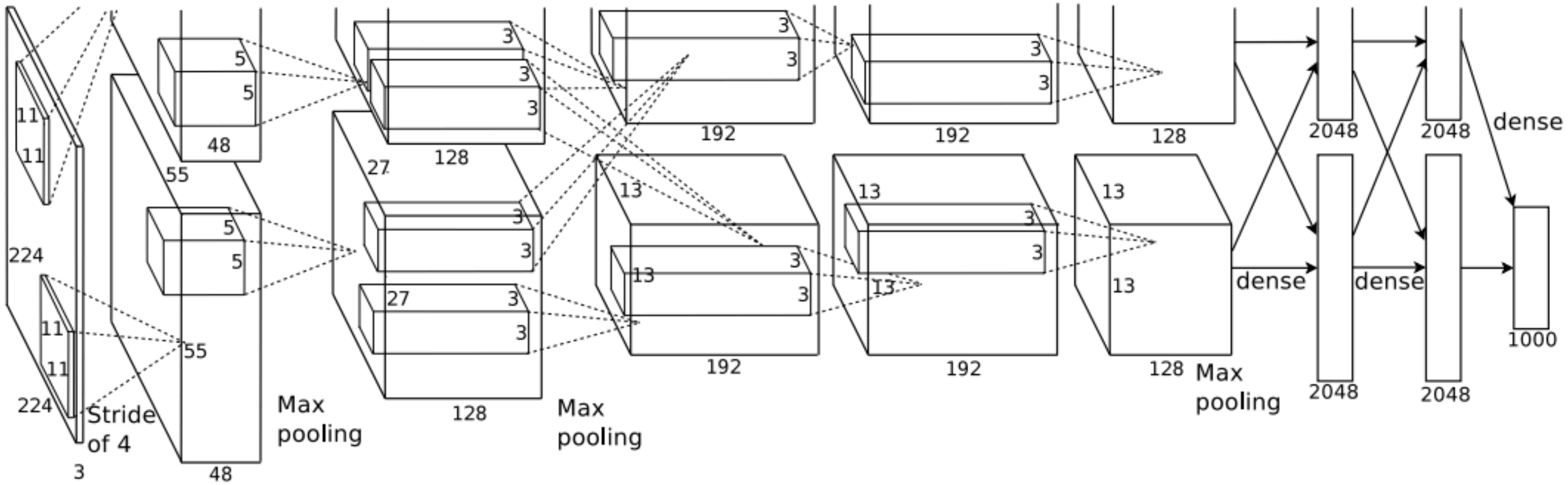
Conv1: 96 11*11*3 filters, with stride 4

Output: (227-11)/4+1=55, 55*55*96

$$(W-F+2P)/S+1$$

Alex et al. ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

First Breakthrough-AlexNet (2012)



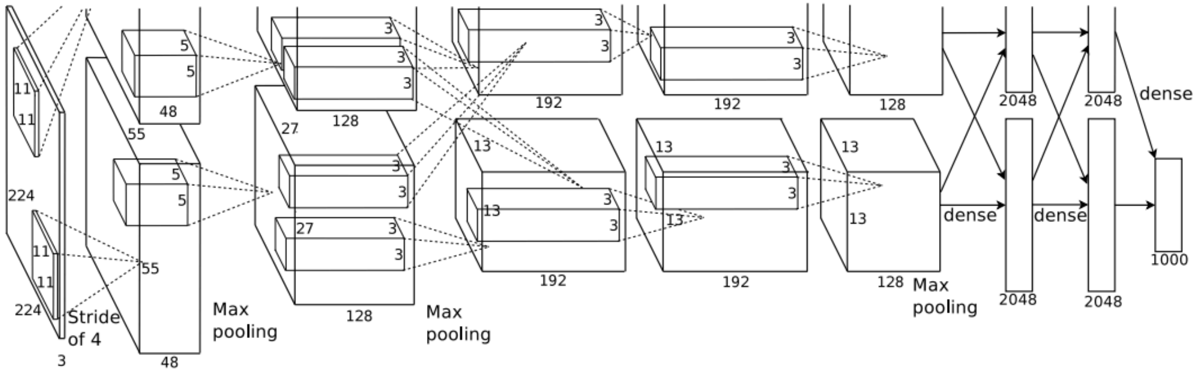
Input: 227*227*3
Conv1: 96 11*11*3 filters, with stride 4
Output: (227-11)/4+1=55
Parameters: 11*11*3*96 ~= 35K

Alex et al. ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

First Breakthrough-AlexNet (2012)

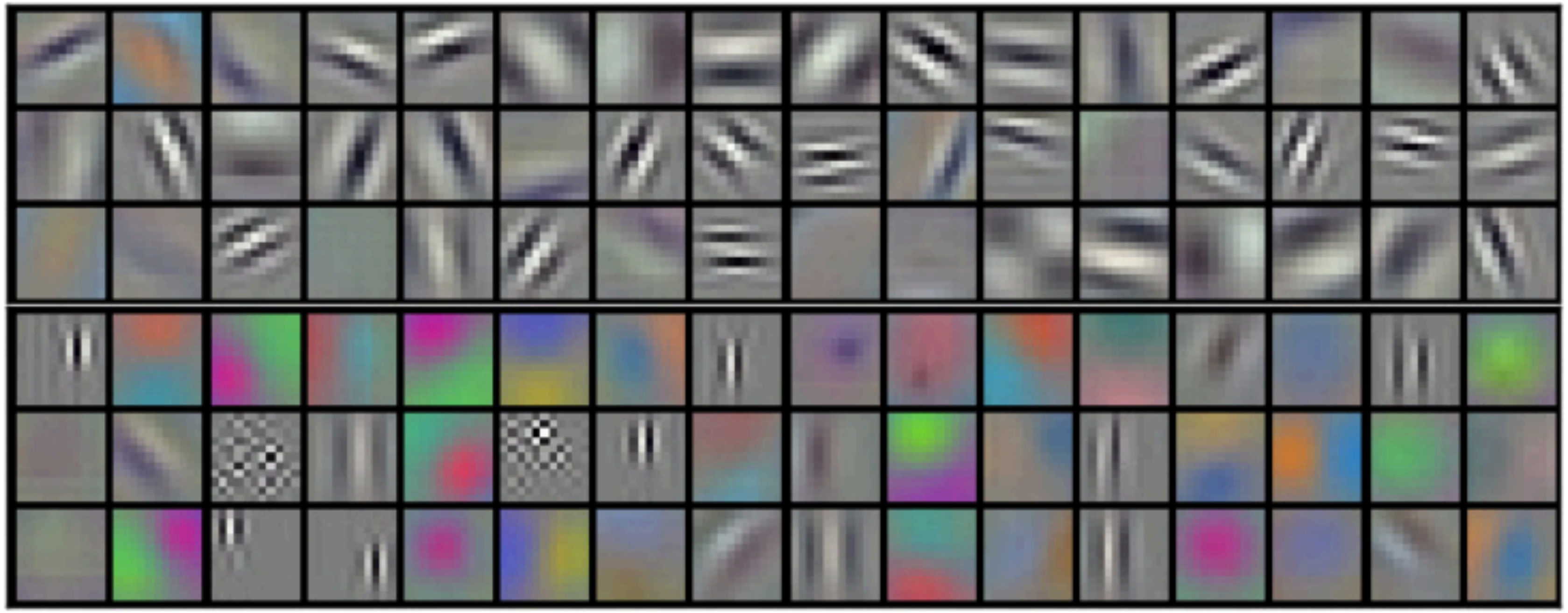
Details:

- first use of ReLU
- used Norm layers (not common anymore) - heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually w accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%



CS231n: Convolutional Neural Networks for Visual Recognition, Stanford, 2019

AlexNet



96 convolutional kernels of size 11×11×3 learned by the first convolutional layer on the 224×224×3 input images.

Alex et al. ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

AlexNet

```
class AlexNet(nn.Module):

    def __init__(self, num_classes=1000):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Linear(4096, num_classes),
        )
```

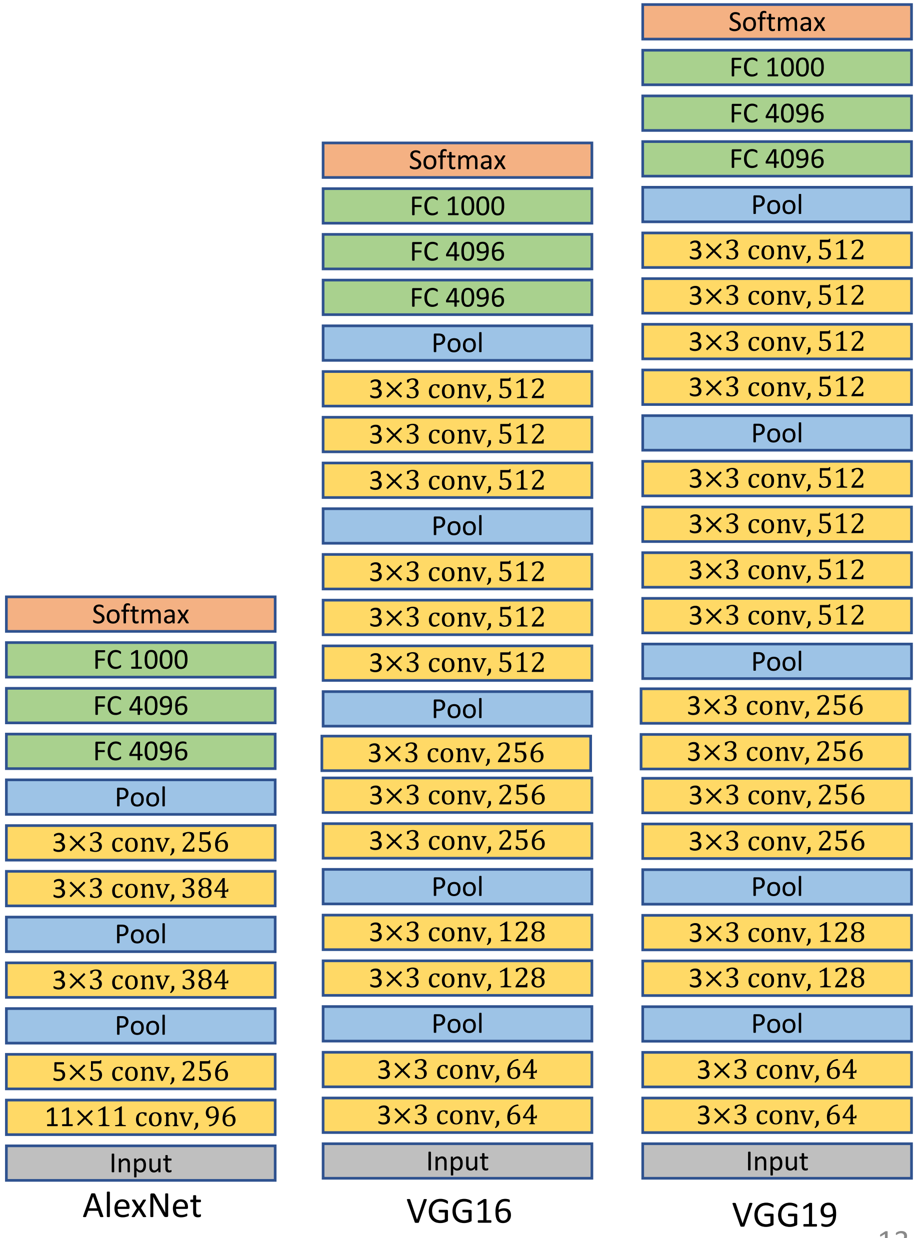
Feature Extractor

Classification

VGGNet

Small filters, Deeper networks

- 16 - 19 layers
- Only 3x3 CONV stride 1, pad 1 and 2x2 MAX POOL stride 2
- 16.4% top 5 error in ILSVRC'12 (AlexNet) -> 7.3% top 5 error in ILSVRC'14



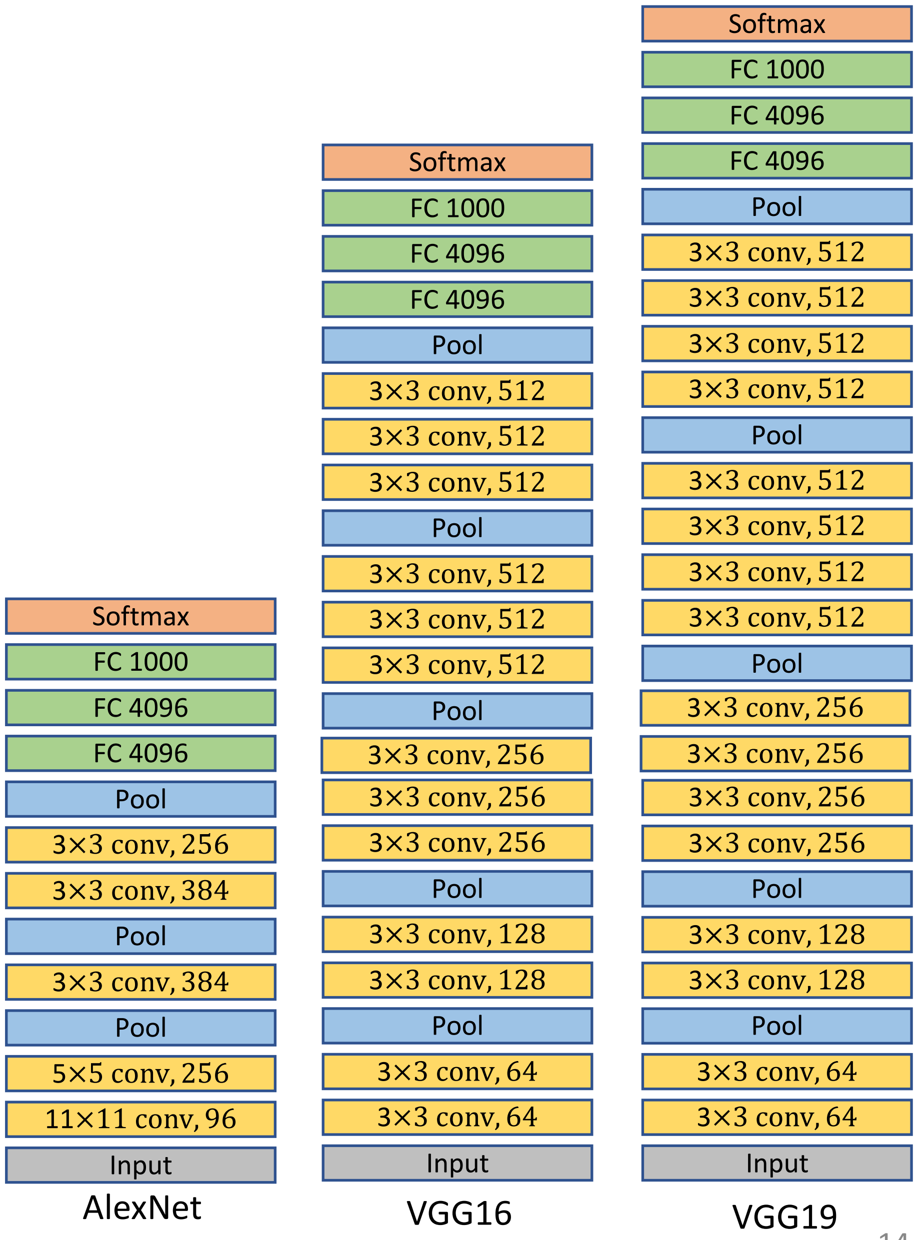
VGGNet

Why smaller filters?

Stack of two 3x3 conv (stride 1) layers has same **effective receptive field** as one 5x5 conv layer.

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer.

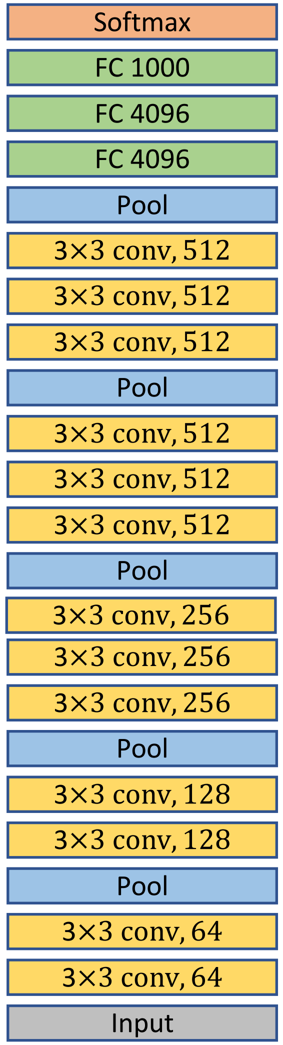
$$C\times(7\times7\times C)=49C^2 \longrightarrow 3\times(C\times(3\times3\times C))=27C^2$$



VGGNet

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000



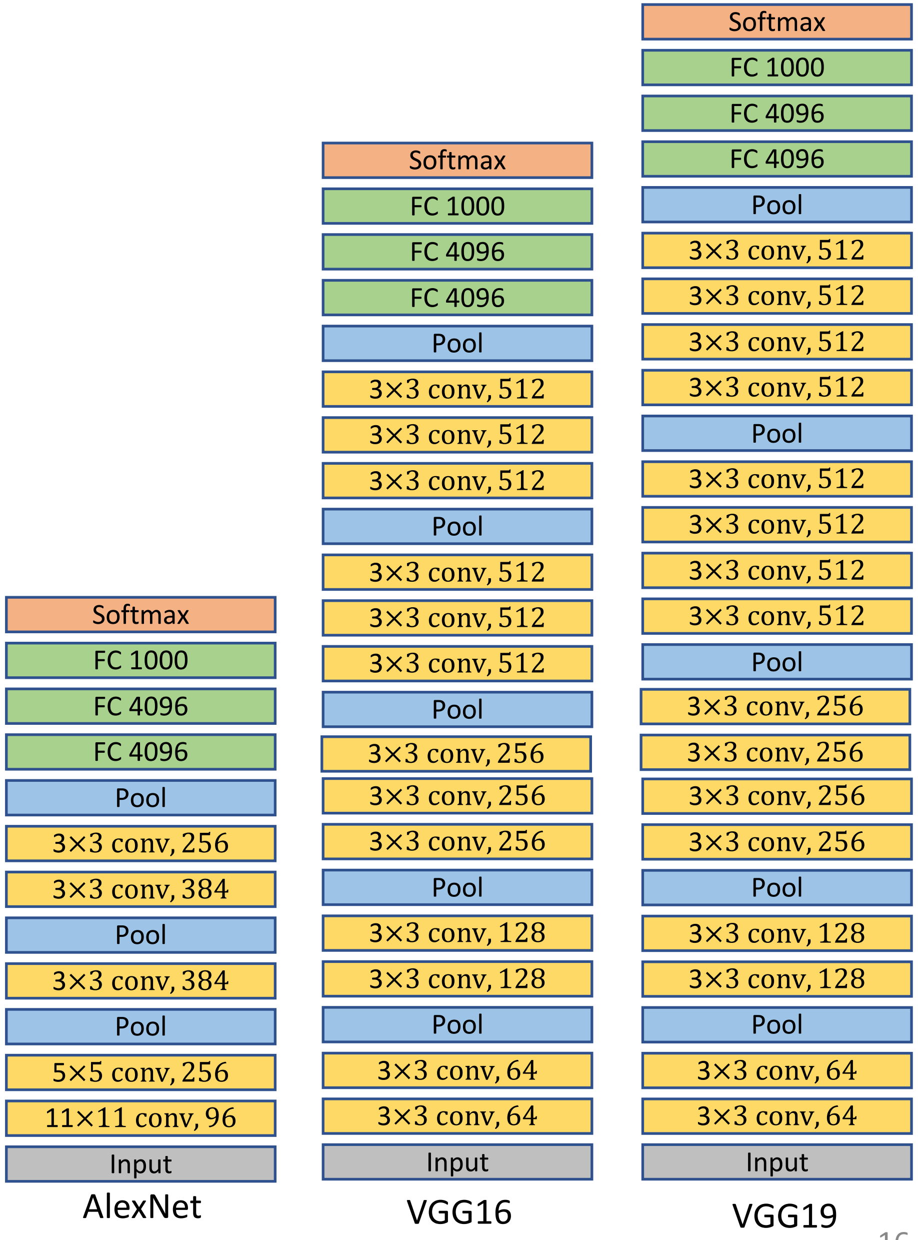
VGG16

CS231n: Convolutional Neural Networks for Visual Recognition, Stanford, 2019

VGGNet

Details:

- ILSVRC'14 2nd in classification, 1st in localization
- Similar training procedure as Krizhevsky 2012
- No Local Response Normalisation (LRN)
- Use VGG16 or VGG19 (VGG19 only slightly better, more memory)
- Use ensembles for best results
- FC7 features generalize well to other tasks



GoogLeNet

The most straightforward way of improving the performance of deep neural networks is by increasing their size.

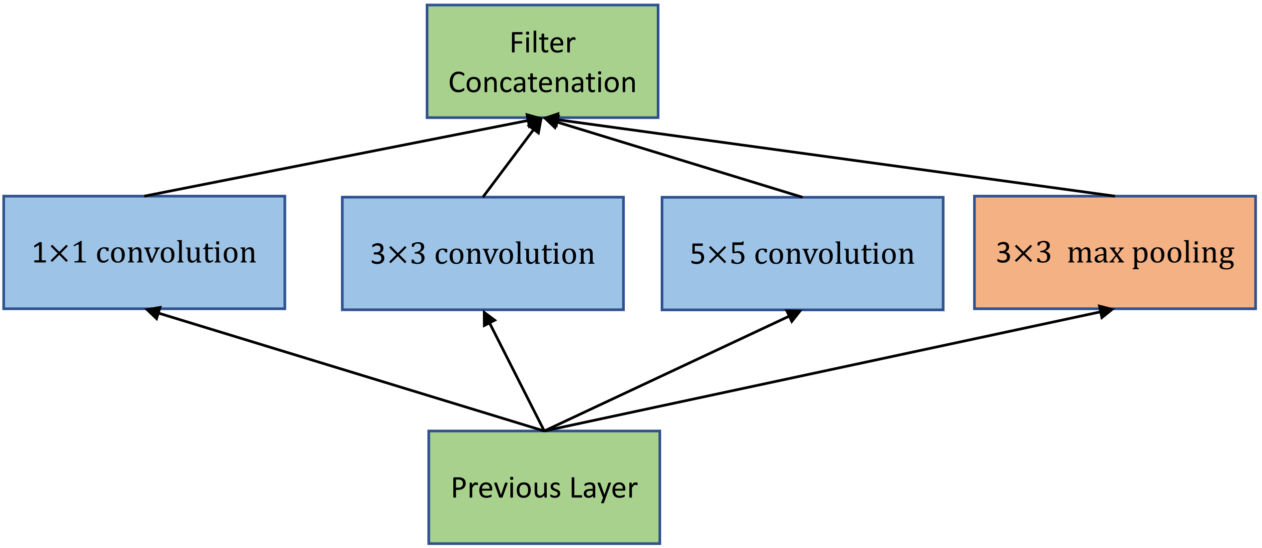
- Bigger size means a larger number of parameters, making the network more prone to the overfitting, if the number of the labeled samples in the training set is scarce.

- Increased network size means the dramatically increased use of computational resources.

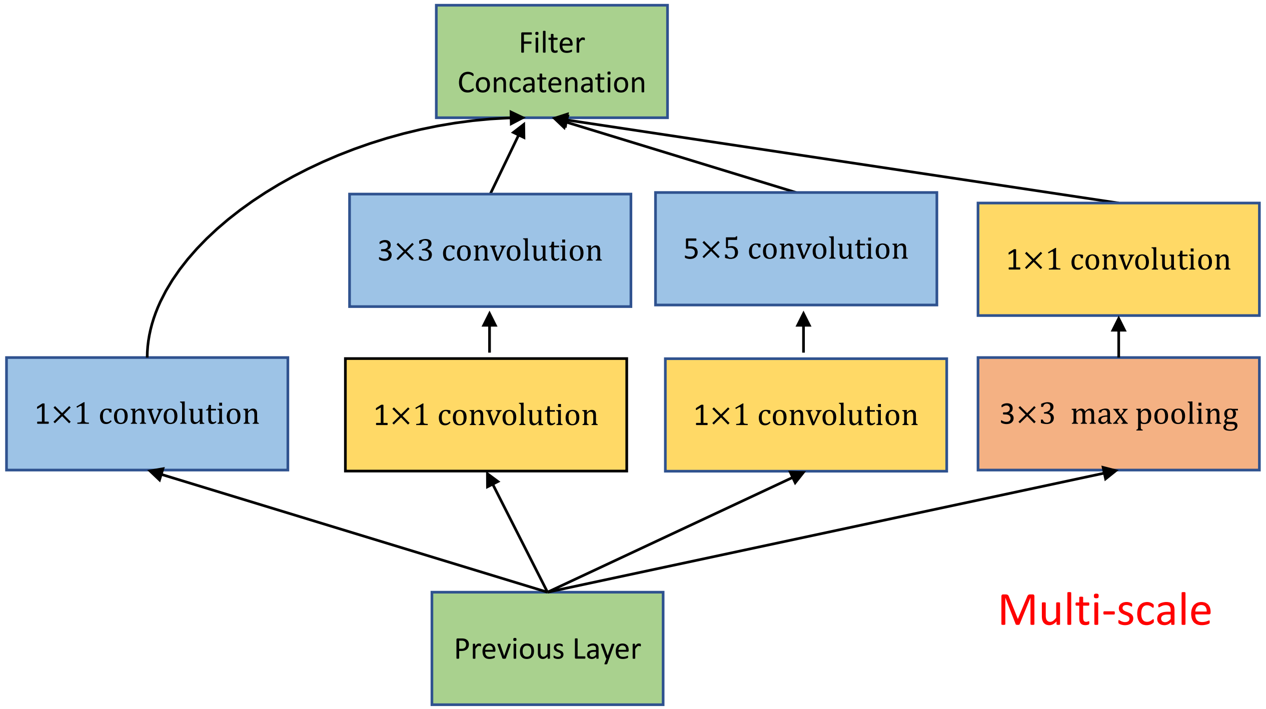
By adding auxiliary classifiers connected to these intermediate layers, discrimination in the lower stages in the classifier was expected. It was used to combat the vanishing gradient problem while providing regularization.

Szegedy et al., Going Deeper with Convolutions, CVPR , 2015

GoogLeNet



(a) Inception Module, naïve version



Multi-scale

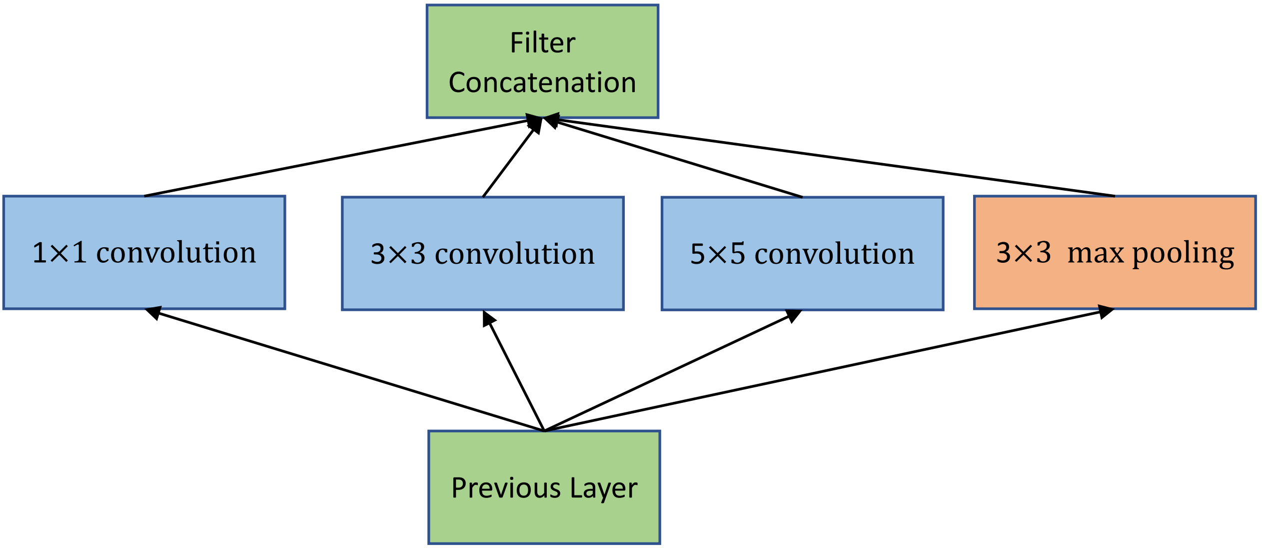
(b) Inception Module with dimension reduction

Increase the depth and width of the network while keeping the computational budget constant.

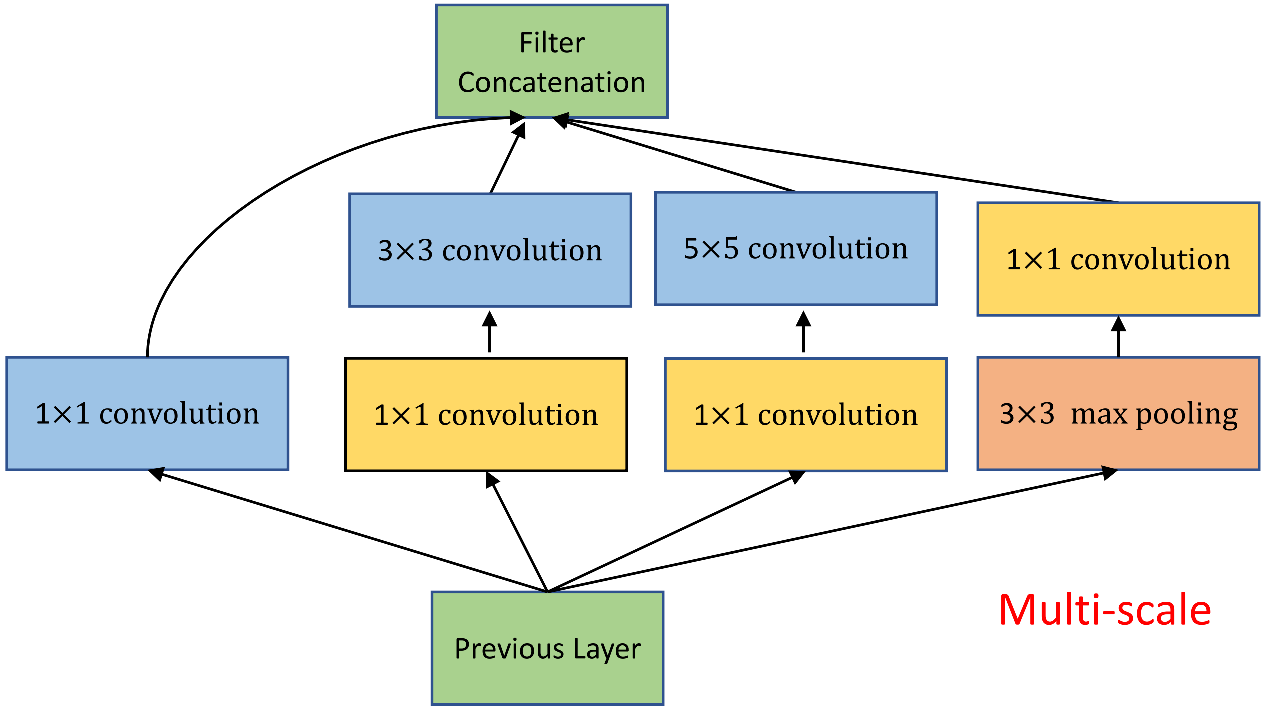
- Depth: the number of network levels
- Width: the number of units at each level

Szegedy et al., Going Deeper with Convolutions, CVPR , 2015

GoogLeNet



(a) Inception Module, naïve version

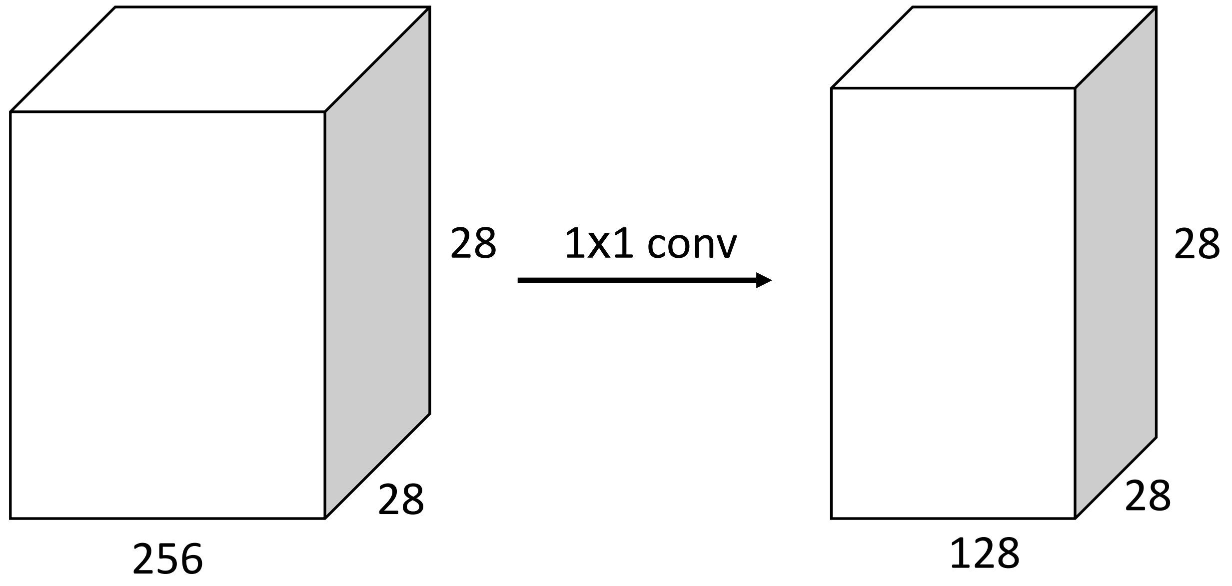


(b) Inception Module with dimension reduction

The design follows the practical intuition that visual information should be processed at various scales and then aggregated so that the next stage can abstract features from the different scales simultaneously.

Szegedy et al., Going Deeper with Convolutions, CVPR , 2015

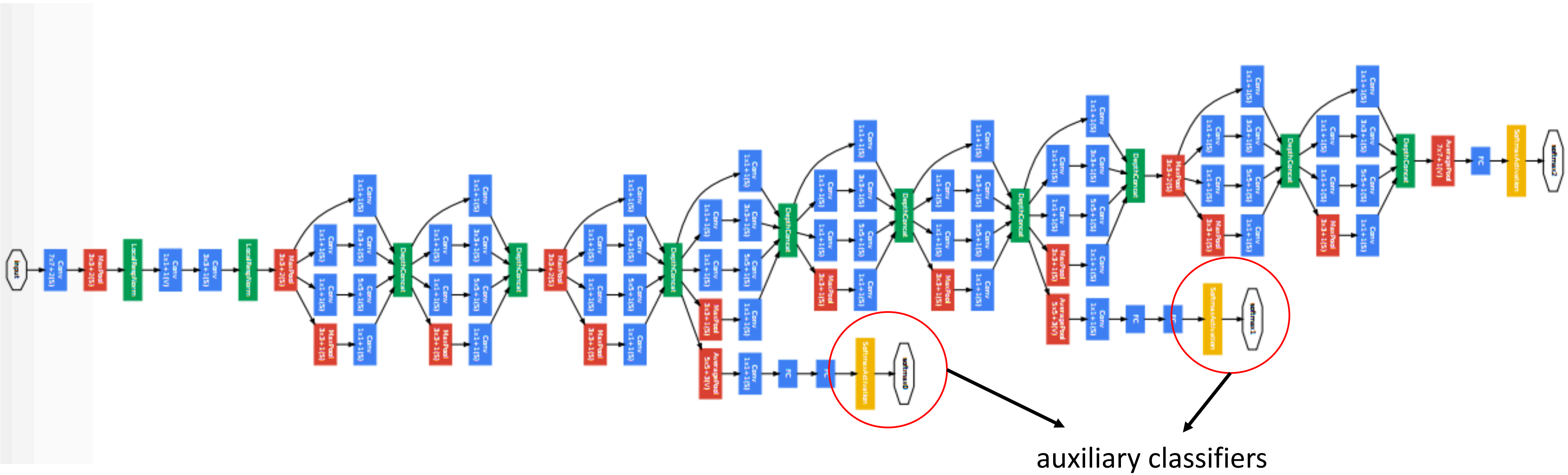
GoogLeNet



1x1 CONV

- 1x1 convolutions are used mainly as dimension reduction modules to remove computational bottleneck.
- 1×1 convolutions are used to compute reductions before the expensive 3×3 and 5×5 convolutions.

GoogLeNet



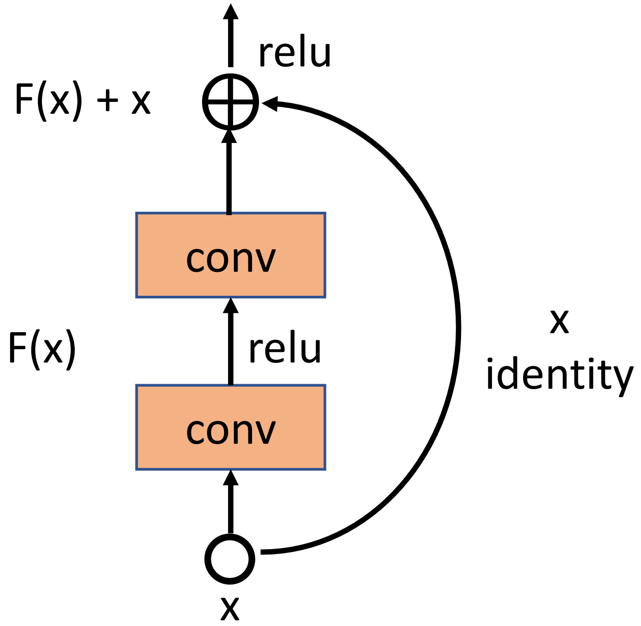
auxiliary classifiers

Szegedy et al., Going Deeper with Convolutions, CVPR , 2015

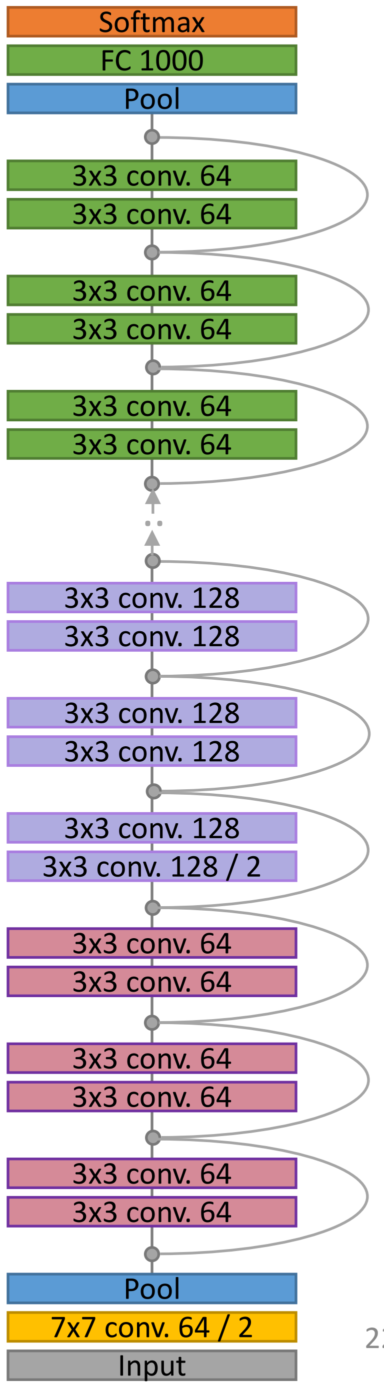
ResNet

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!

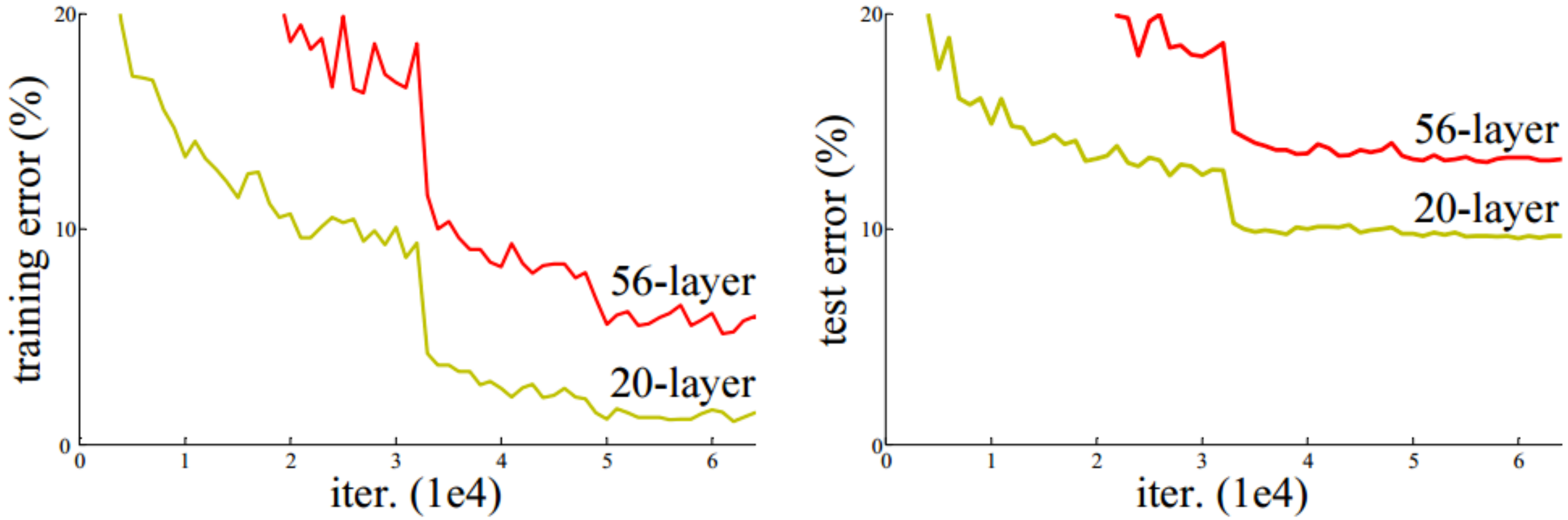


Residual block



ResNet

Is learning better networks as easy as stacking more layers?

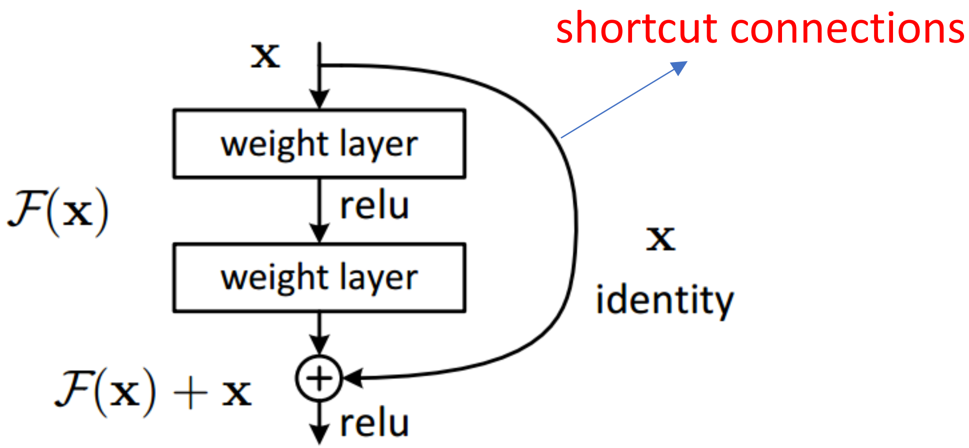


Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error.

Deeper neural networks are more difficult to train. Such degradation is not caused by overfitting.

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

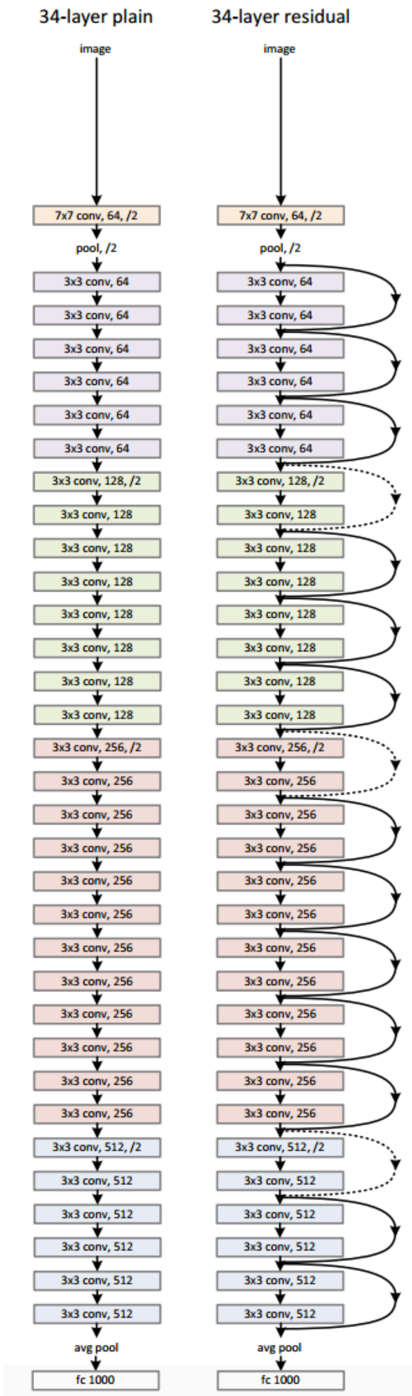
ResNet



Residual Mapping: $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$

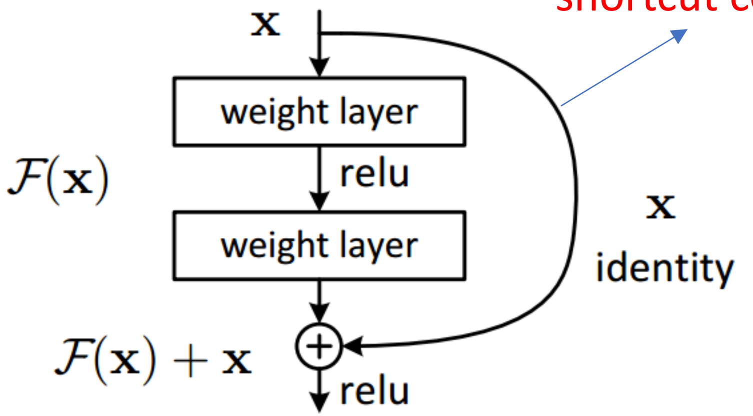
The formulation of F(x)+x can be realized by feedforward neural networks with “shortcut connections”.

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



ResNet

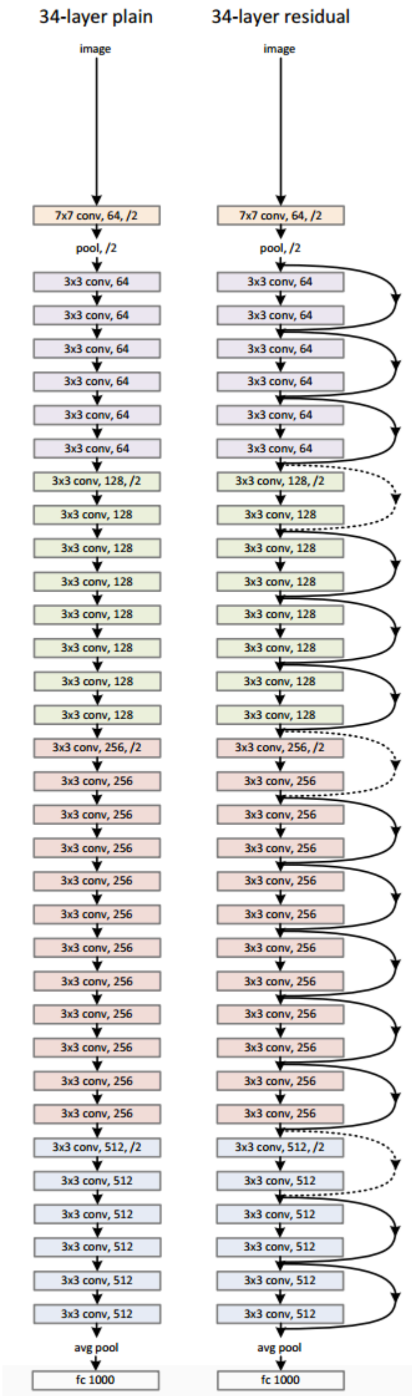
shortcut connections



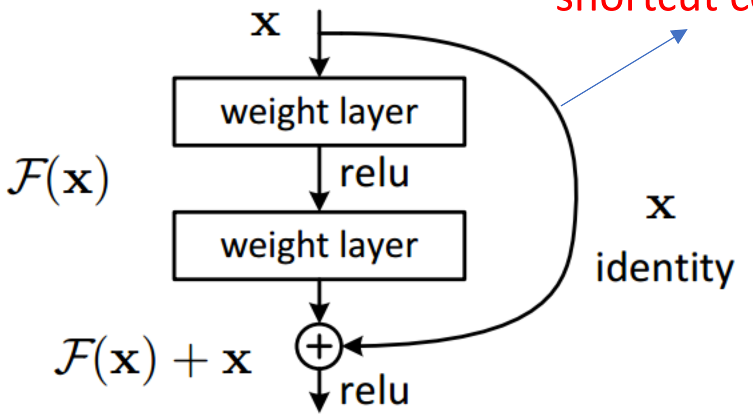
Residual Mapping: $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$

Identity shortcut connections add no extra parameter.

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



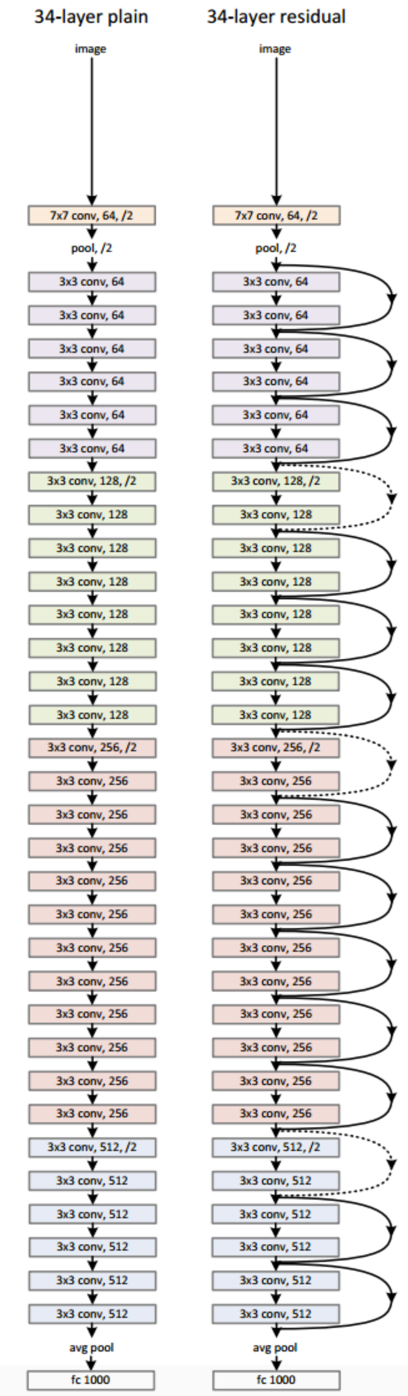
ResNet



shortcut connections

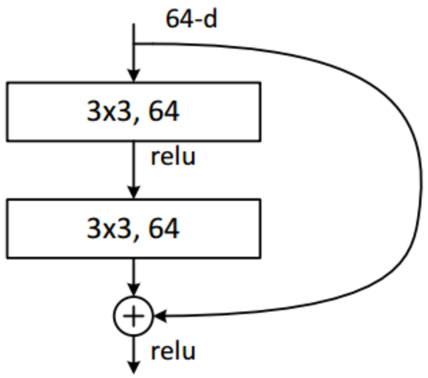
Residual Mapping: $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$

The form of the residual function F is flexible. If F has only a single layer, equation is similar to a linear layer (no observed advantage). F is applicable to convolutional layers.



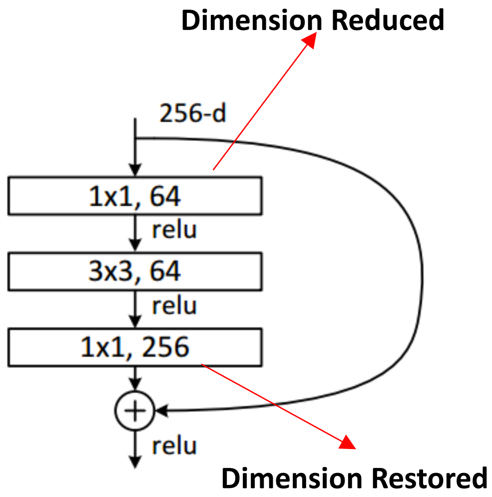
He et al., Deep Residual Learning for Image Recognition, CVPR 2016

ResNet



Used in ResNet-34

```python
def forward(self, x):
    identity = x

    out = self.conv1(x)
    out = self.bn1(out)
    out = self.relu(out)

    out = self.conv2(out)
    out = self.bn2(out)

    if self.downsample is not None:
        identity = self.downsample(x)

    out += identity
    out = self.relu(out)

    return out
```



Used in ResNet-50/101/152

```python
def forward(self, x):
    identity = x

    out = self.conv1(x)
    out = self.bn1(out)
    out = self.relu(out)

    out = self.conv2(out)
    out = self.bn2(out)
    out = self.relu(out)

    out = self.conv3(out)
    out = self.bn3(out)

    if self.downsample is not None:
        identity = self.downsample(x)

    out += identity
    out = self.relu(out)

    return out
```

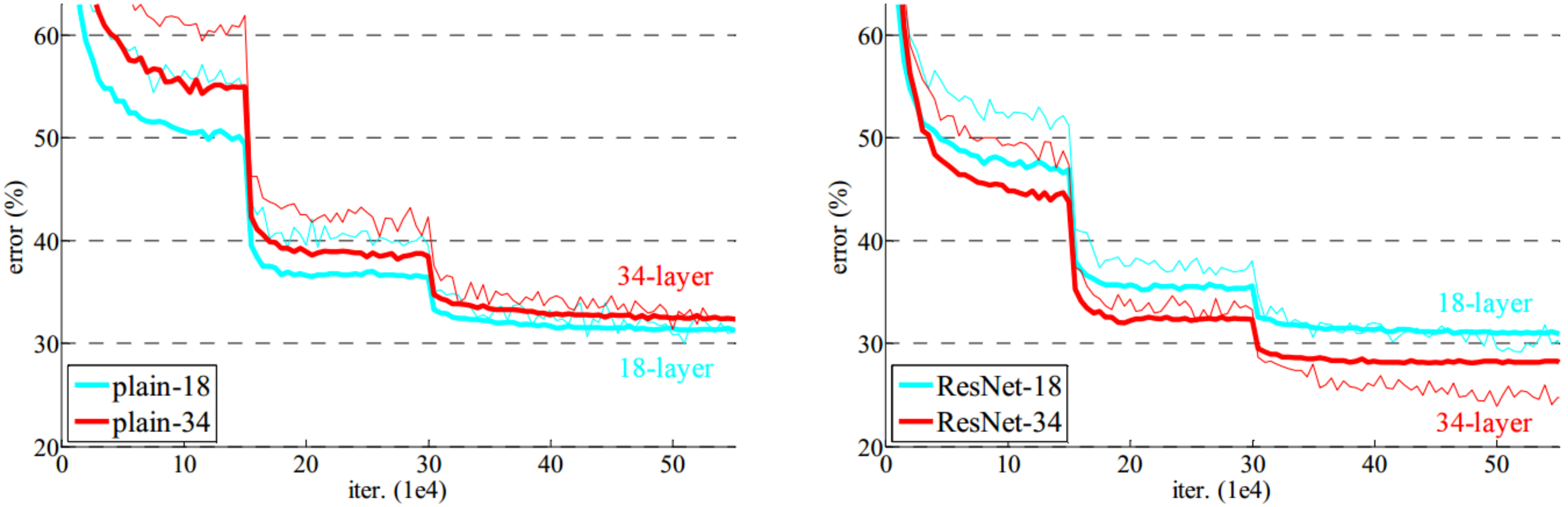
ResNet

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

Building blocks are shown in brackets, with the numbers of blocks stacked.

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

ResNet



Training on ImageNet. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

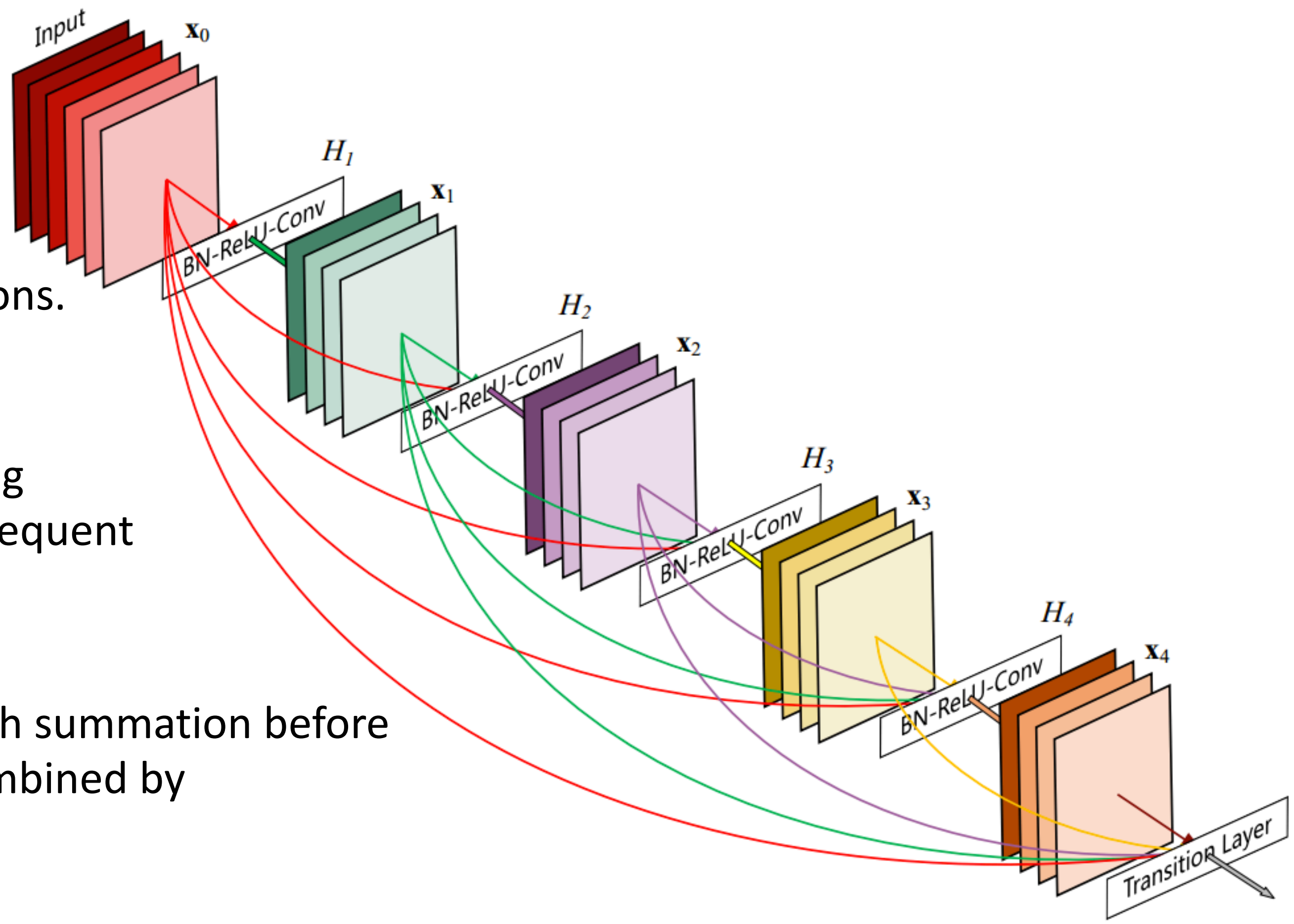
He et al., Deep Residual Learning for Image Recognition, CVPR 2016

DenseNet

- Traditional convolutional networks with L layers have connections, while DenseNet has L(L +1)/2 connections.

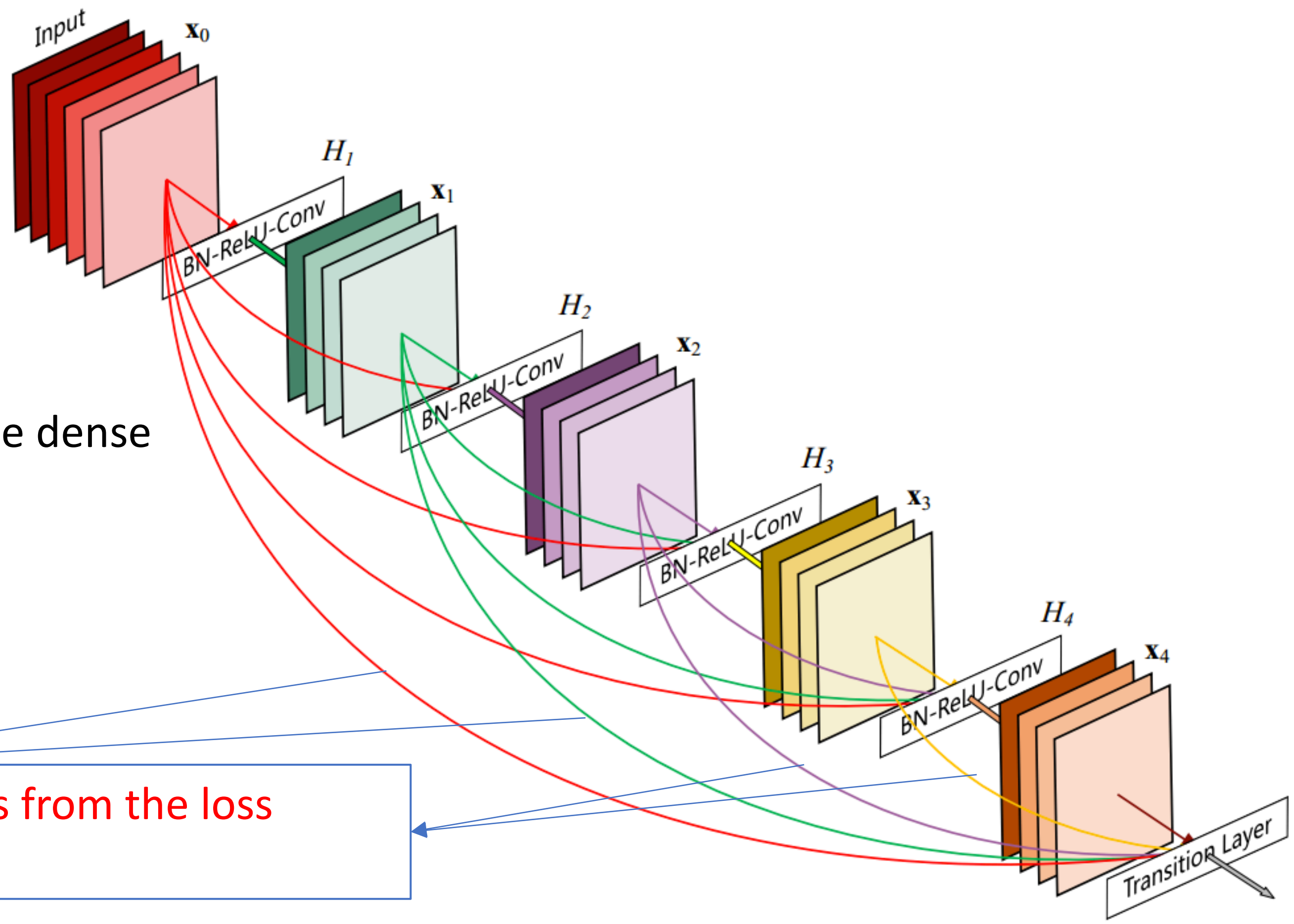
- Each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers.

- In contrast to ResNets, features are combined through summation before they are passed into a layer; instead, features are combined by concatenating them.



Huang et al., Densely connected convolutional networks, CVPR 2017

DenseNet

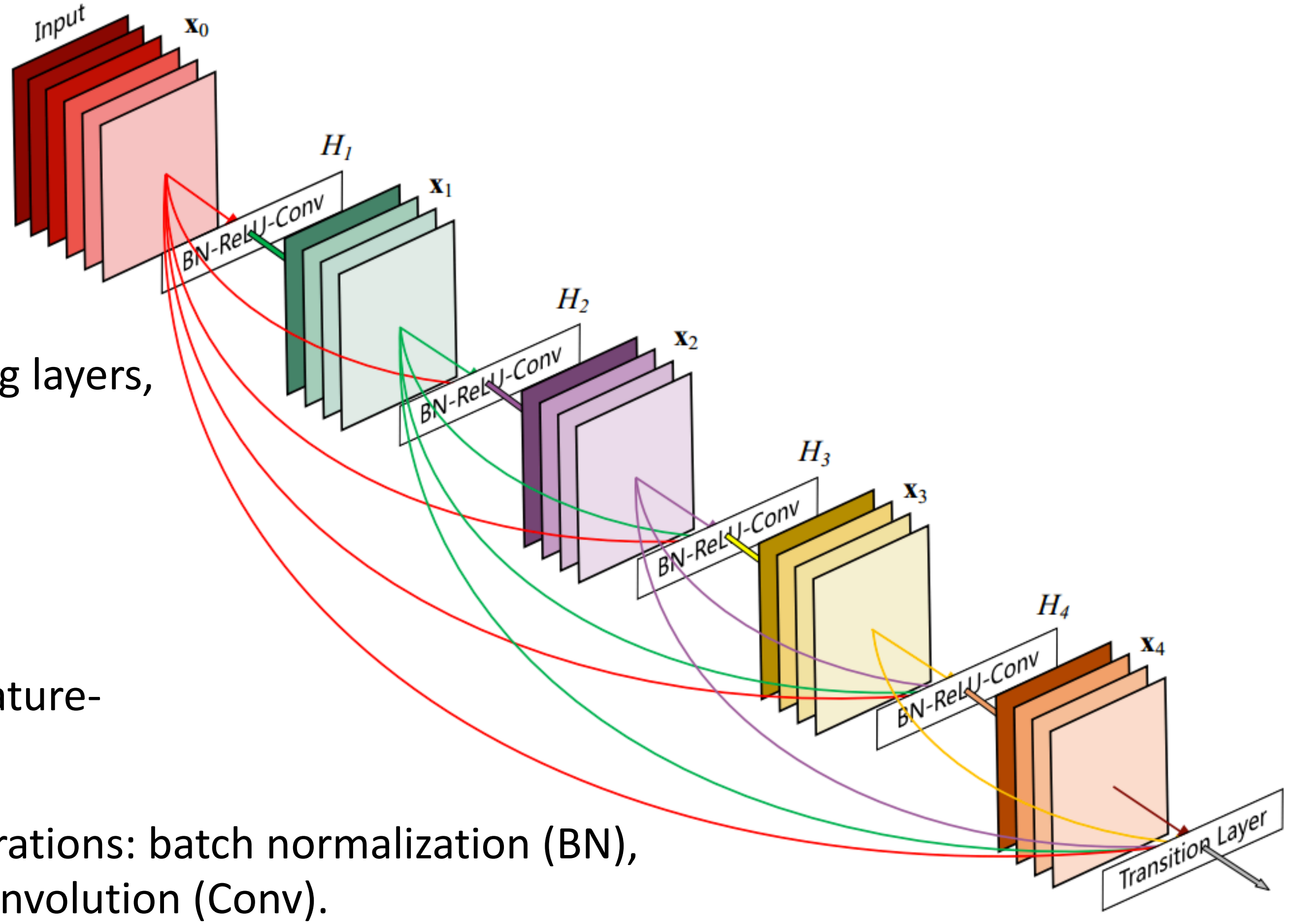


DenseNet improves the gradient flow through the dense connections.

Each layer has direct access to the gradients from the loss function.

Huang et al., Densely connected convolutional networks, CVPR 2017

DenseNet



The ℓth layer receives the feature-maps of all preceding layers, $x_0 \cdots x_{\ell\text{-}1}$, as input:

$$\mathbf{x}_\ell = H_\ell([\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{\ell-1}])$$

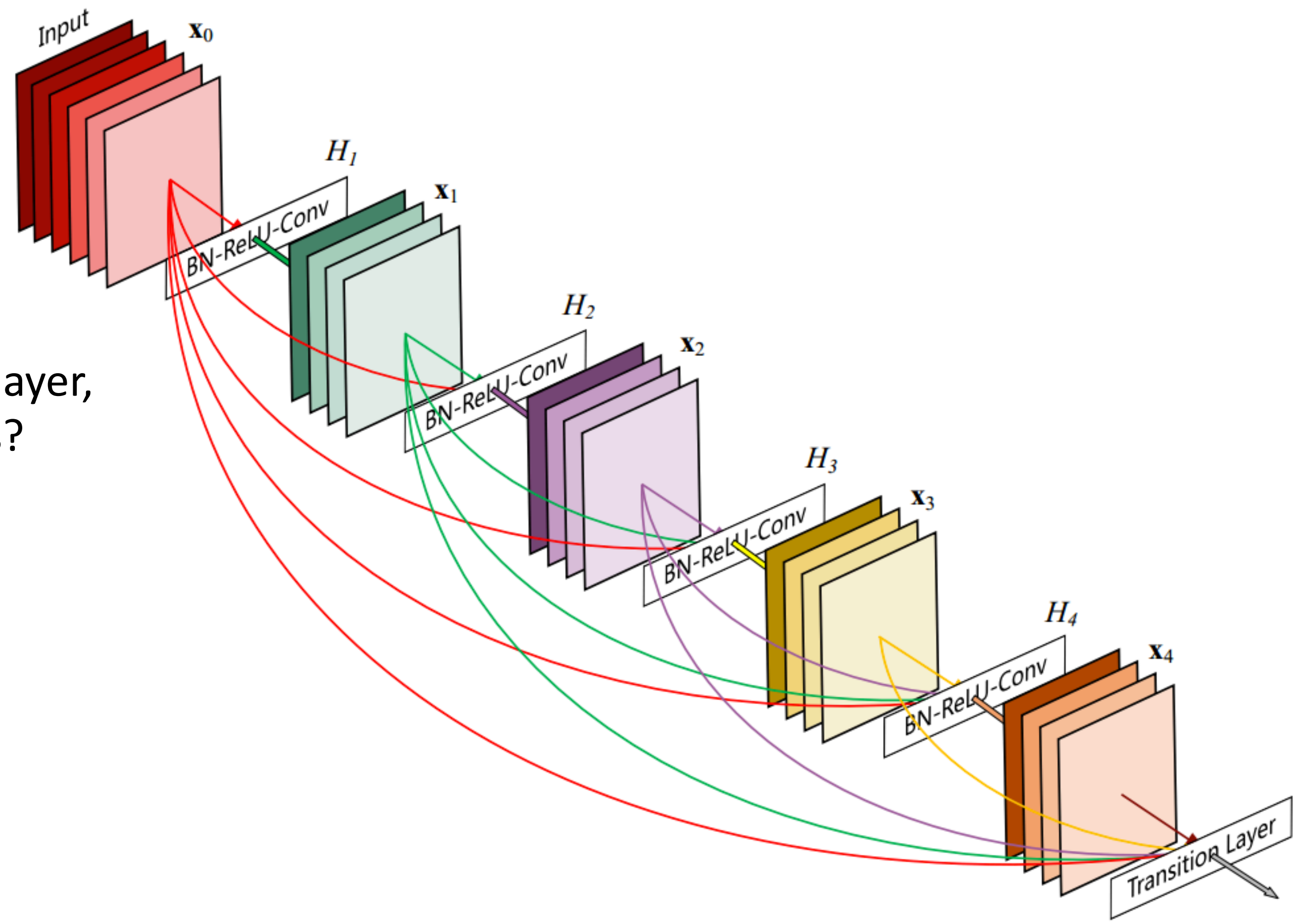
where $[x_0 \cdots x_{\ell\text{-}1}]$ refers to the concatenation of the feature-maps produced in layers $0 \cdots \ell$-1.

H_ℓ (·) is a composite function of three consecutive operations: batch normalization (BN), followed by a rectified linear unit (ReLU) and a 3 × 3 convolution (Conv).

Huang et al., Densely connected convolutional networks, CVPR 2017

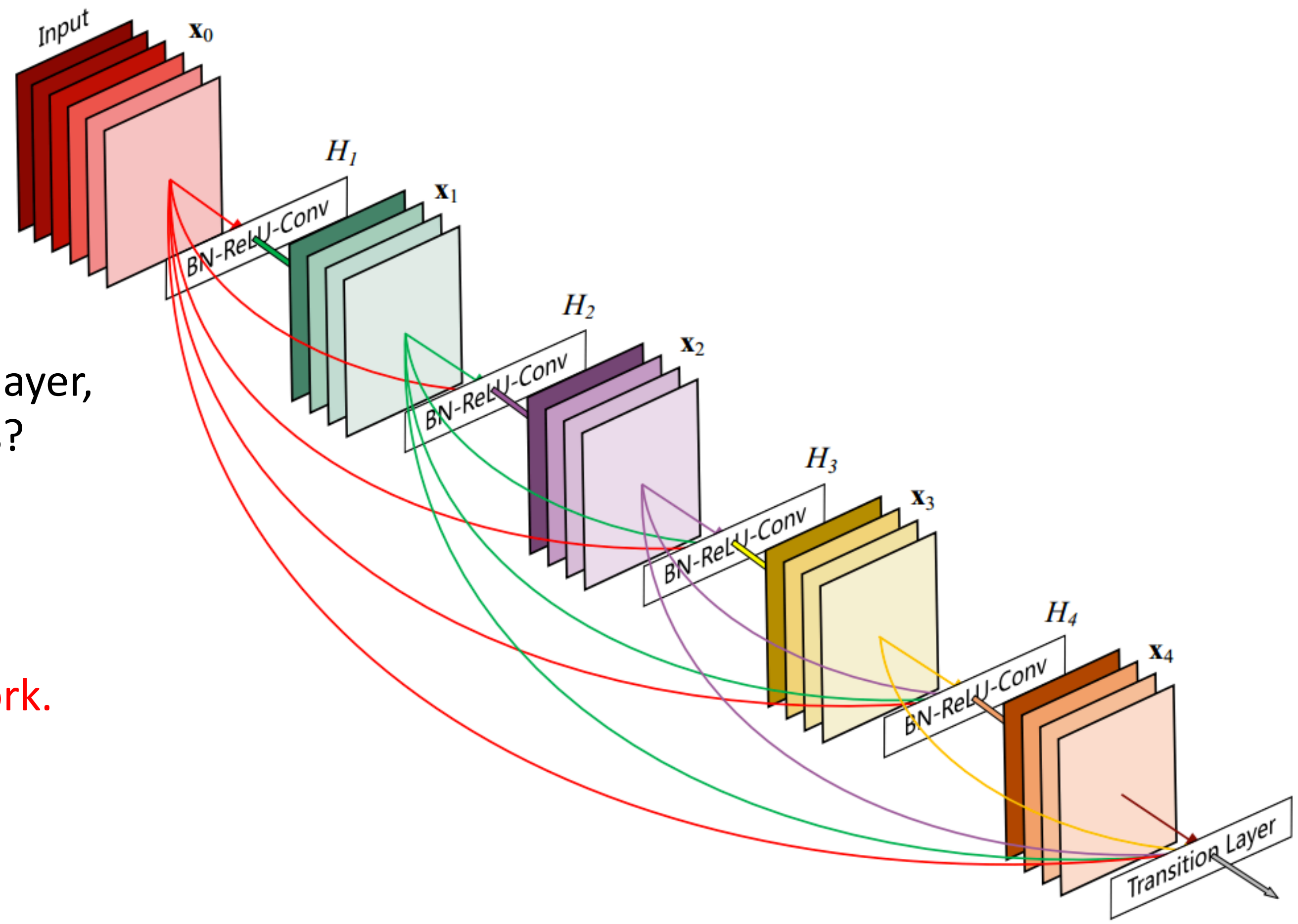
DenseNet

What is the number of input feature maps for the ℓth layer, suppose that each function H produces k feature maps?



Huang et al., Densely connected convolutional networks, CVPR 2017

DenseNet



What is the number of input feature maps for the ℓth layer, suppose that each function H produces k feature maps?

$$k_0 + k \times (\ell - 1)$$

The hyper-parameter k is the growth rate of the network.

Huang et al., Densely connected convolutional networks, CVPR 2017

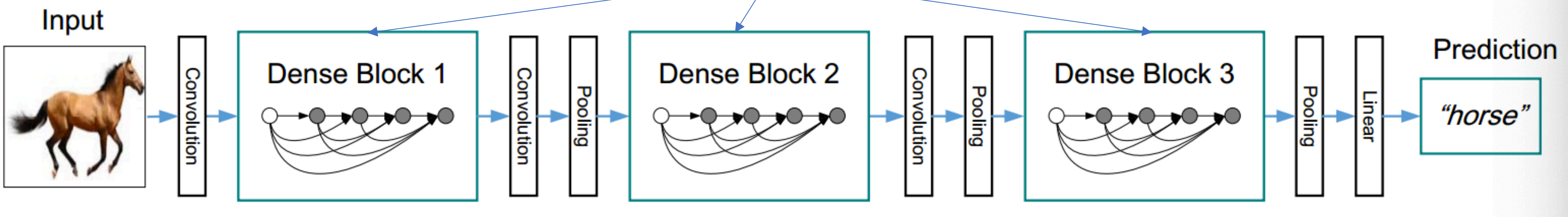
DenseNet

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112×112	7×7 conv, stride 2			
Pooling	56×56	3×3 max pool, stride 2			
Dense Block (1)	56×56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56×56	1×1 conv			
	28×28	2×2 average pool, stride 2			
Dense Block (2)	28×28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28×28	1×1 conv			
	14×14	2×2 average pool, stride 2			
Dense Block (3)	14×14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14×14	1×1 conv			
	7×7	2×2 average pool, stride 2			
Dense Block (4)	7×7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1×1	7×7 global average pool			
		1000D fully-connected, softmax			

Huang et al., Densely connected convolutional networks, CVPR 2017

DenseNet

To facilitate down-sampling in our architecture, the network was divided into multiple densely connected dense blocks.

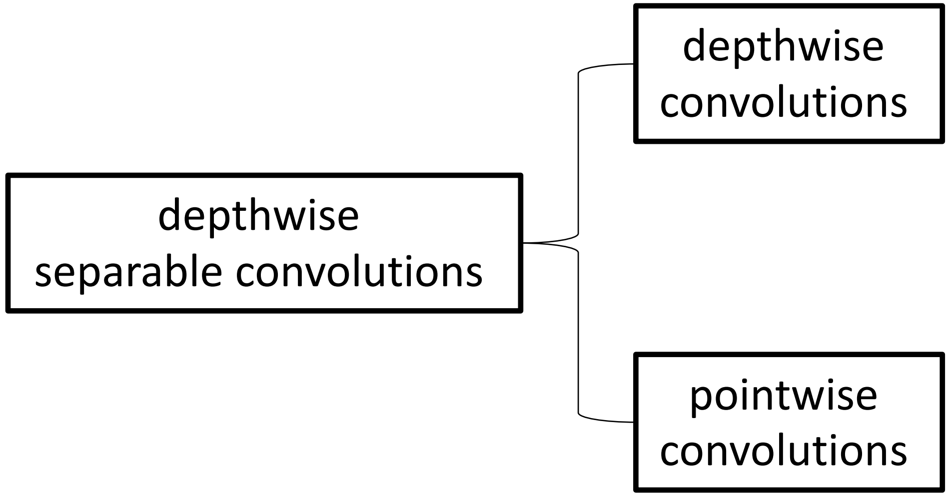


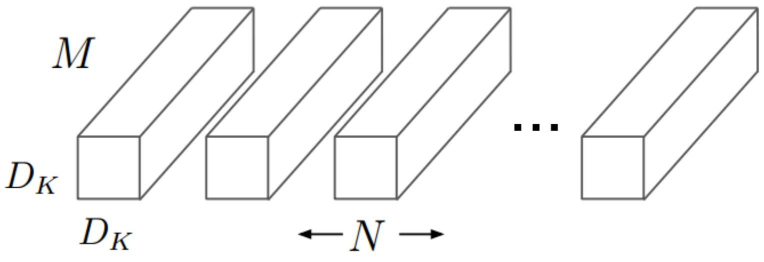
A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

Huang et al., Densely connected convolutional networks, CVPR 2017

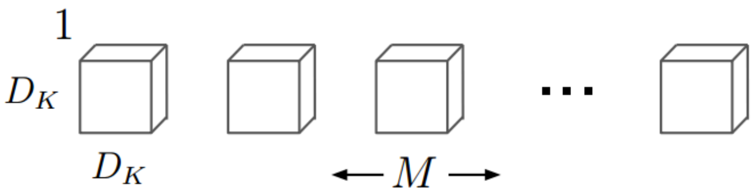
MobileNet

The MobileNet model is based on **depthwise separable convolutions** which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1× 1 convolution called a pointwise convolution.

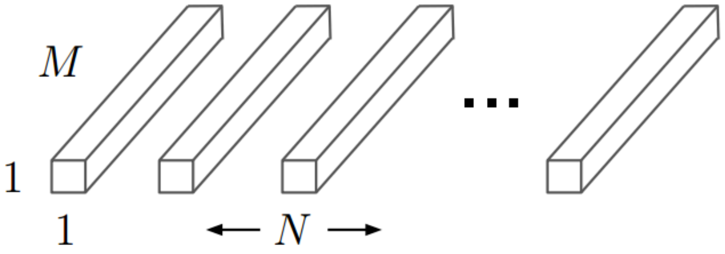




(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

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

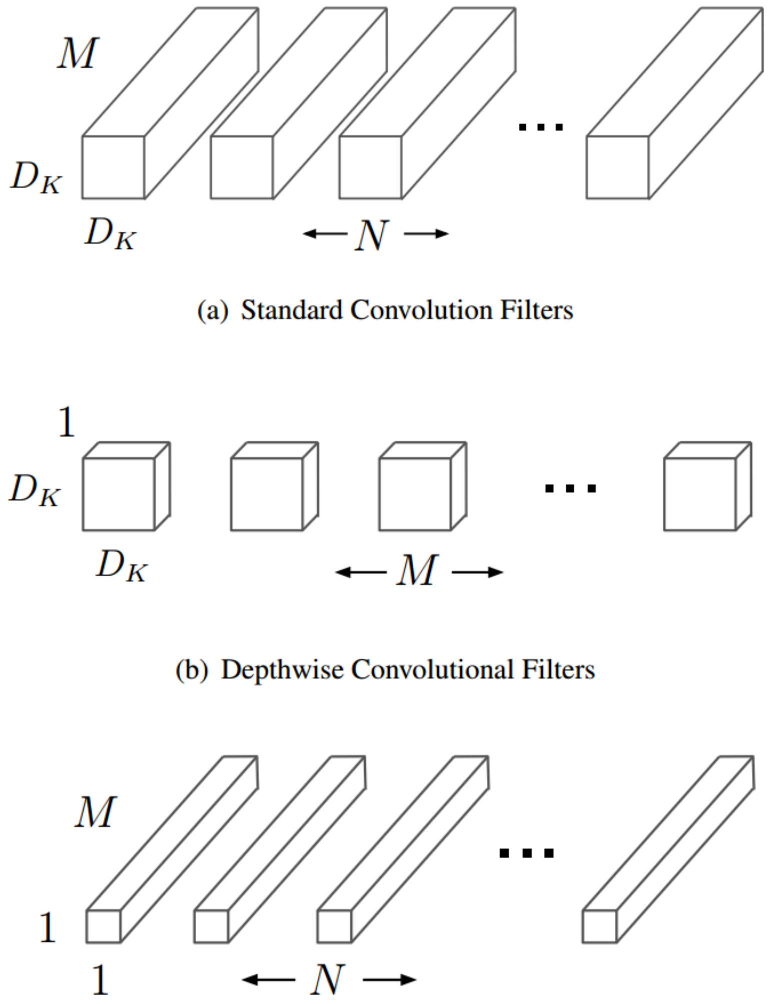
MobileNet

$Input\ feature\ map\ F\colon\ D_F{\times}D_F{\times}M$

$Ouput\ feature\ map\ G\colon\ D_G{\times}D_G{\times}N$

$Convolution\ kernel\ K\colon\ D_K{\times}D_K{\times}M\ {\times}N$

What is the parameter and computational cost for standard convolution?



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

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

MobileNet

$Input\ feature\ map\ F\colon\ D_F{\times}D_F{\times}M$

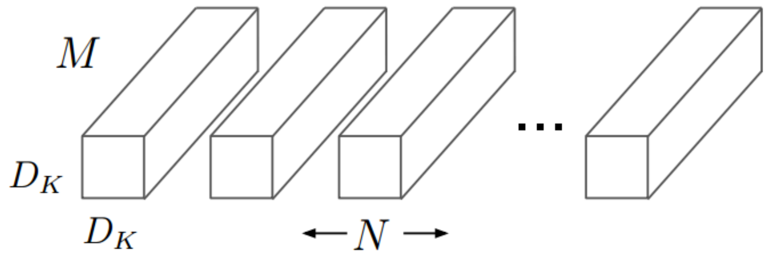
$Ouput\ feature\ map\ G\colon\ D_G{\times}D_G{\times}N$

$Convolution\ kernel\ K\colon\ D_K{\times}D_K{\times}M\ {\times}N$

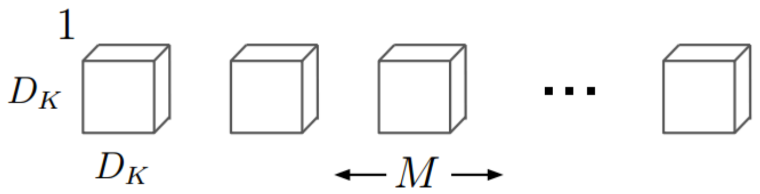
What is the parameter and computational cost for standard convolution?

$Parameter\colon\ D_K{\times}D_K{\times}M\ {\times}N$

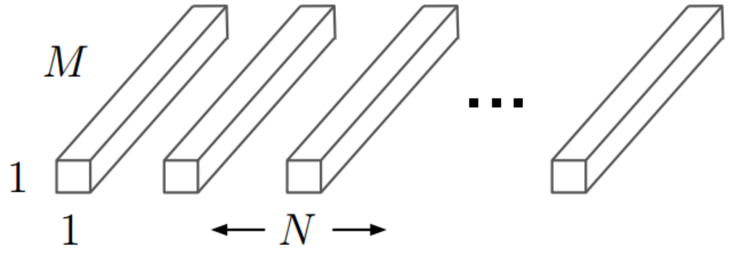
$Computational\ Cost\colon\ D_K{\times}D_K{\times}M\ {\times}N{\times}D_G{\times}D_G$



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters

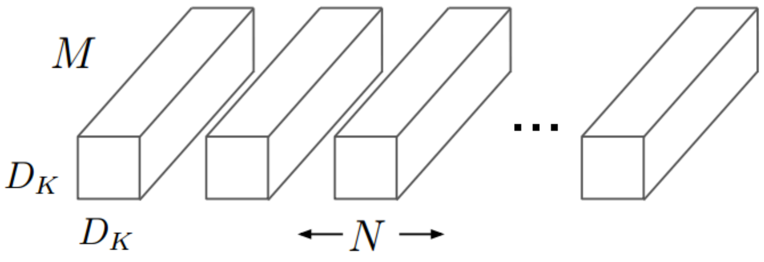


(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

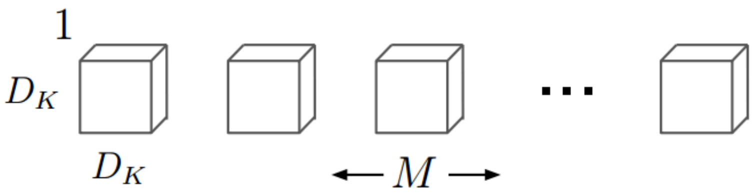
Howard et al., MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv 2017

MobileNet

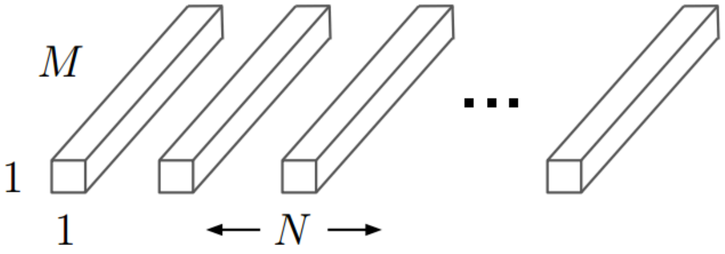
What is the parameter and computational cost for depthwise separable convolution?



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

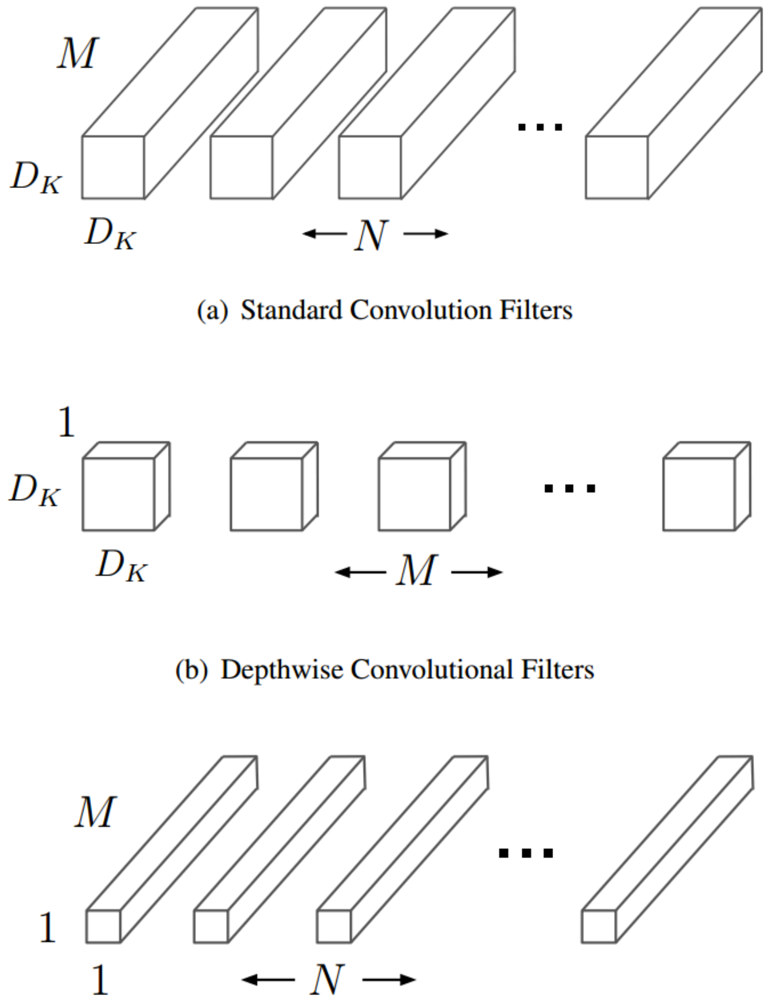
Howard et al., MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv 2017

MobileNet

What is the parameter and computational cost for depthwise separable convolution?

$Parameter: D_K \times D_K \times M + M \times N$

$Computational\ Cost: D_K \times D_K \times M \times D_G \times D_G$+M $\times N \times D_G \times D_G$



(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

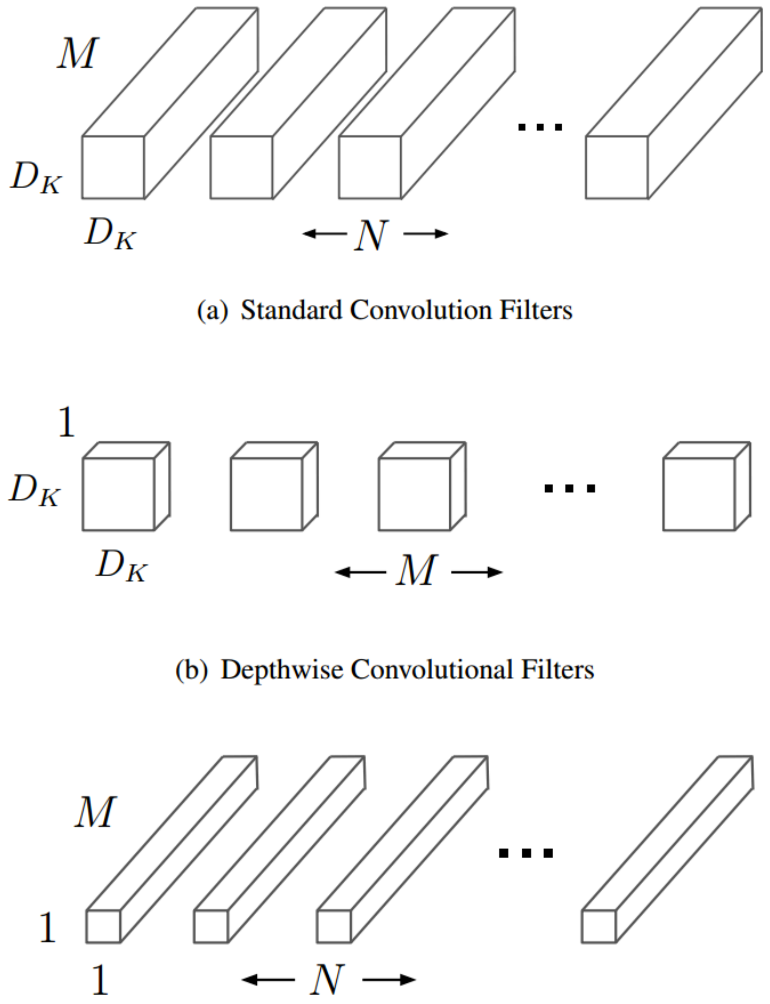
Howard et al., MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv 2017

MobileNet

What is reduction in computational cost?

$$\frac{D_K \times D_K \times M \times D_G \times D_G + M \times N \times D_G \times D_G}{D_K \times D_K \times M \times N \times D_G \times D_G}$$

$$= \frac{1}{N} + \frac{1}{D_K \times D_K}$$



(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

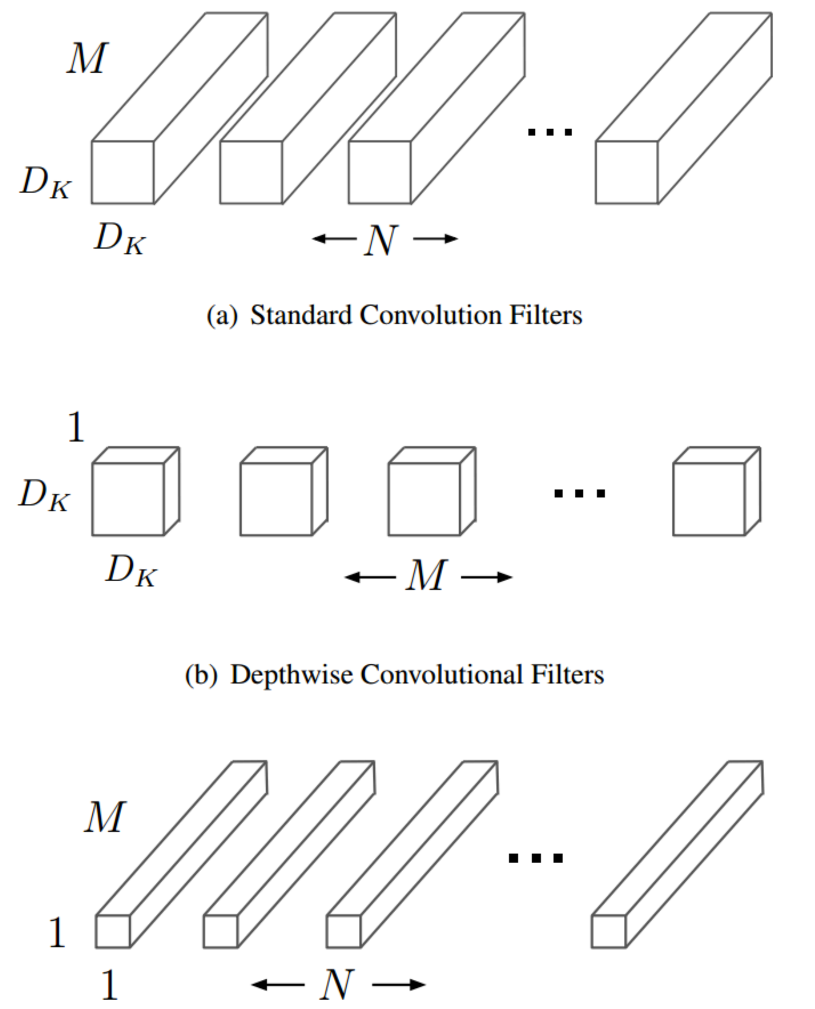
Howard et al., MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv 2017

MobileNet

What is reduction in parameter size?

$$\frac{D_K \times D_K \times M + M \times N}{D_K \times D_K \times M \times N}$$

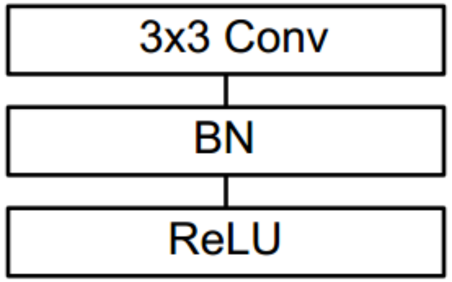
$$= \frac{1}{N} + \frac{1}{D_K \times D_K}$$



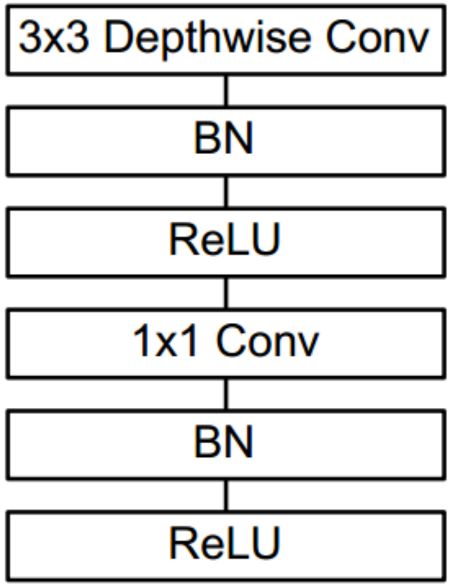
(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

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

MobileNet



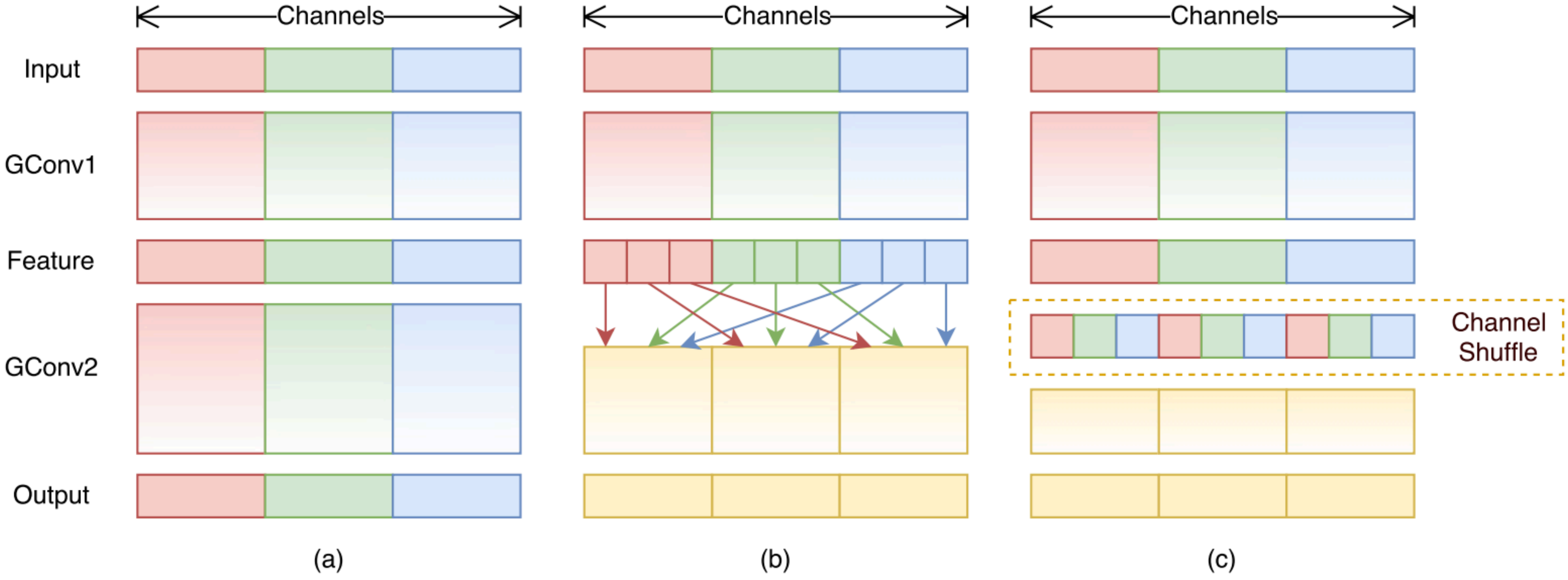
Standard convolutional layer



Depthwise + Pointwise layers

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

ShuffleNet



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

Model Comparisons

Model	Input Resolution	Params(M)	MACs(G)	Top-1 error	Top-5 error
alexnet	224x224	61.1	0.72	43.45	20.91
vgg11	224x224	132.86	7.63	30.98	11.37
vgg13	224x224	133.05	11.34	30.07	10.75
vgg16	224x224	138.36	15.5	28.41	9.62
vgg19	224x224	143.67	19.67	27.62	9.12
vgg11_bn	224x224	132.87	7.64	29.62	10.19
vgg13_bn	224x224	133.05	11.36	28.45	9.63
vgg16_bn	224x224	138.37	15.53	26.63	8.50
vgg19_bn	224x224	143.68	19.7	25.76	8.15

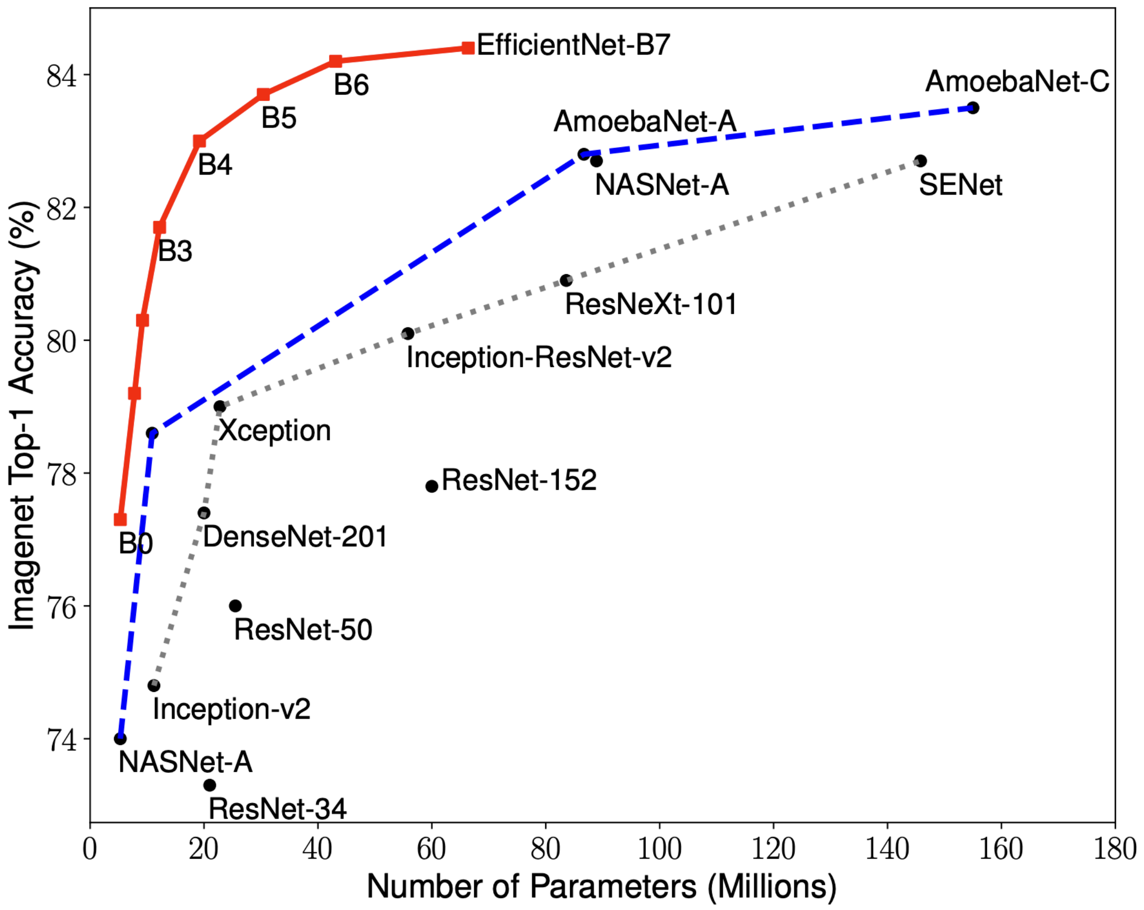
Model	Input Resolution	Params(M)	MACs(G)	Top-1 error	Top-5 error
resnet18	224x224	11.69	1.82	30.24	10.92
resnet34	224x224	21.8	3.68	26.70	8.58
resnet50	224x224	25.56	4.12	23.85	7.13
resnet101	224x224	44.55	7.85	22.63	6.44
resnet152	224x224	60.19	11.58	21.69	5.94
squeezenet1_0	224x224	1.25	0.83	41.90	19.58
squeezenet1_1	224x224	1.24	0.36	41.81	19.38
densenet121	224x224	7.98	2.88	25.35	7.83
densenet169	224x224	14.15	3.42	24.00	7.00
densenet201	224x224	20.01	4.37	22.80	6.43
densenet161	224x224	28.68	7.82	22.35	6.20
inception_v3	224x224	27.16	2.85	22.55	6.44

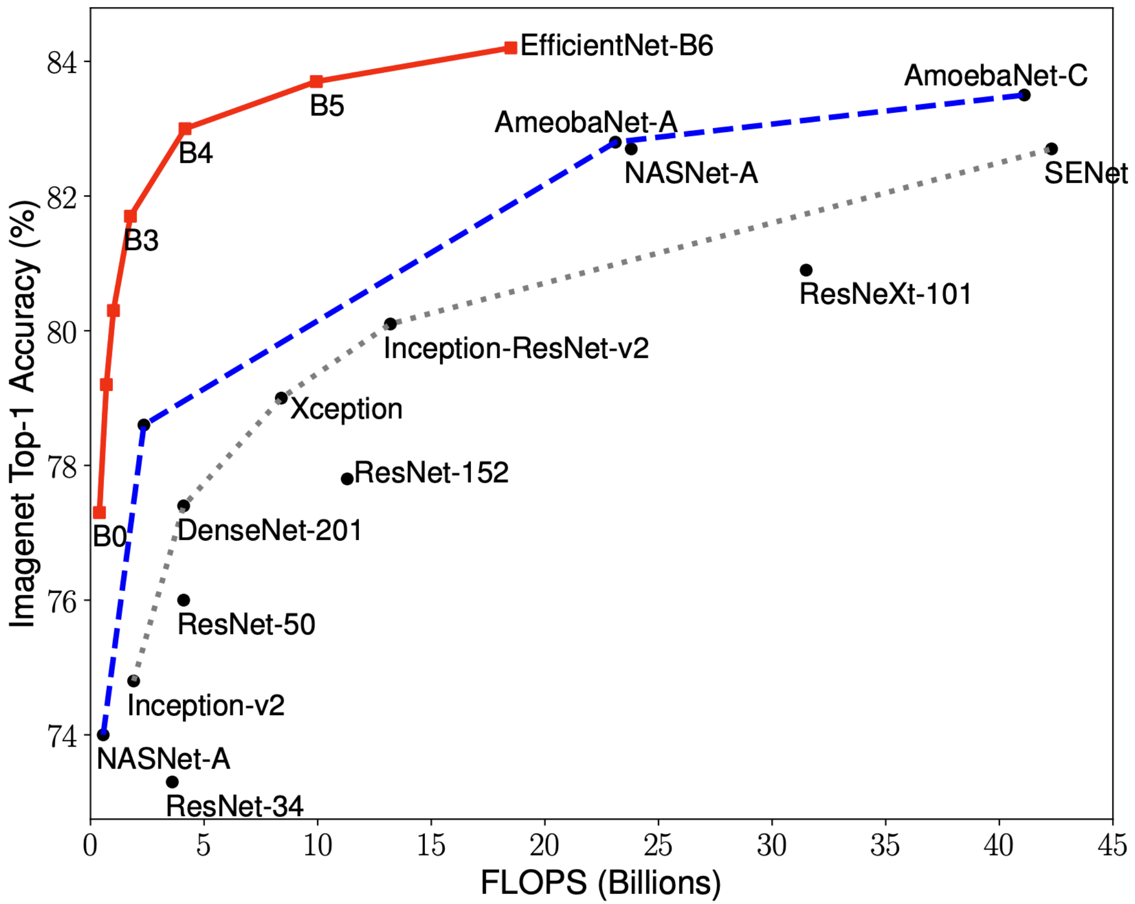
GMACs = 2 * GFLOPs

TORCHVISION.MODELS

https://github.com/sovrasov/flops-counter.pytorch

Model Comparisons





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