CNN Architectures

Neural Networks Design And Application

Input resolution issue

- Global average pooling

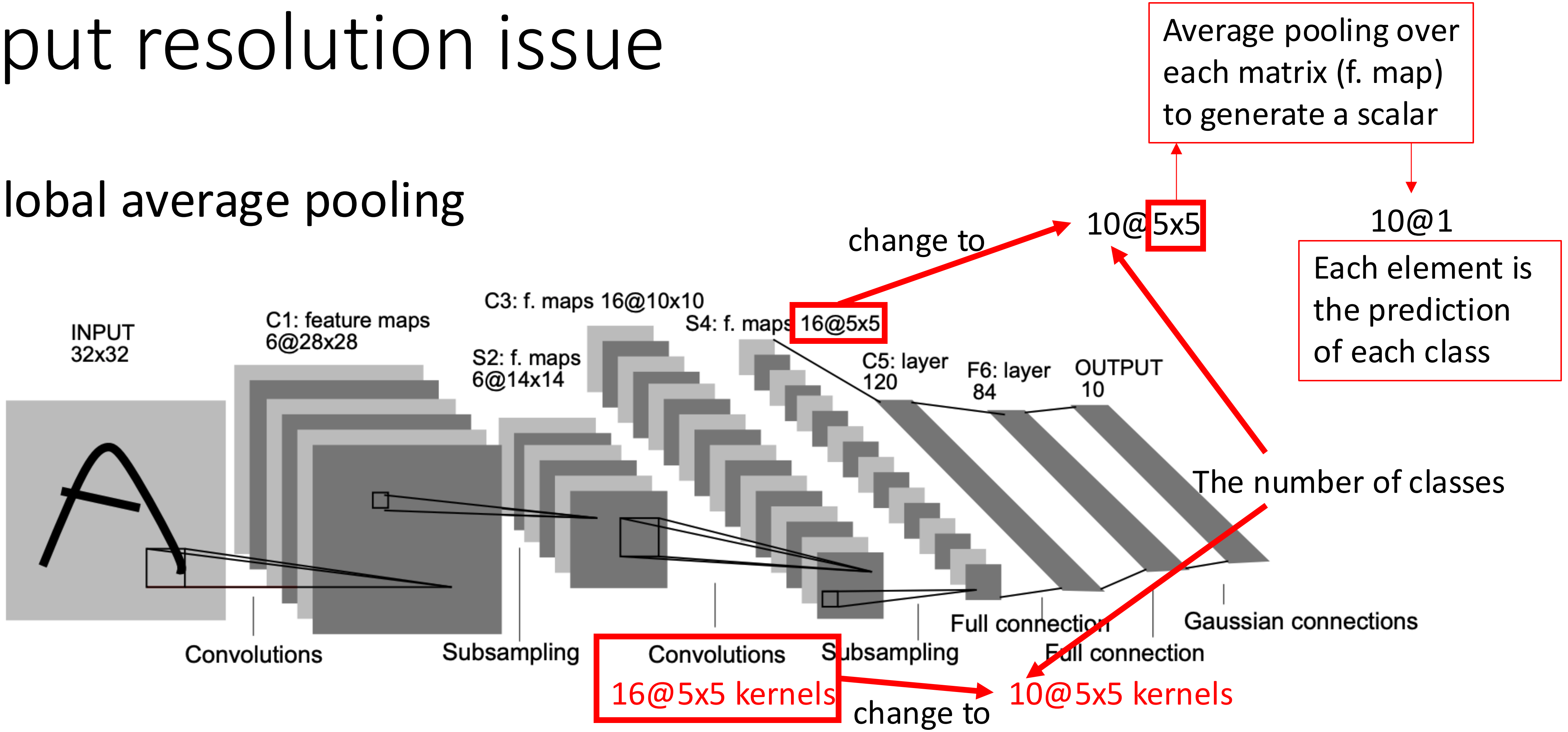
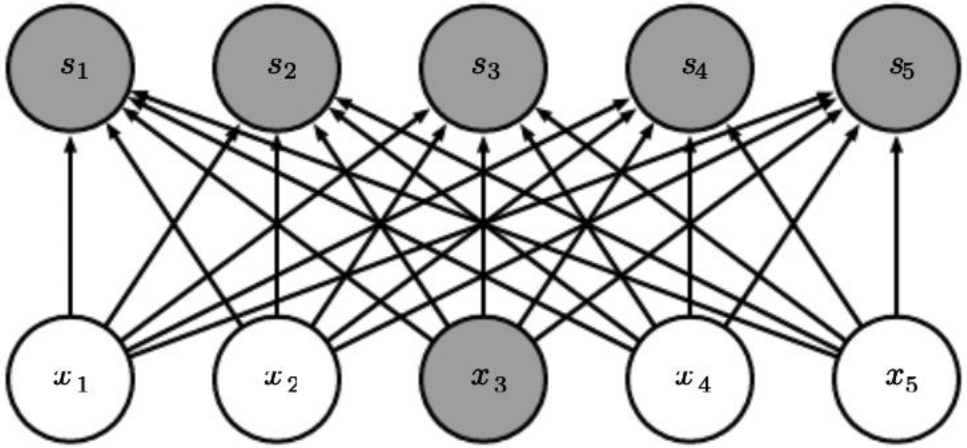


Fig. 1. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Difference between ConvNet and MLP

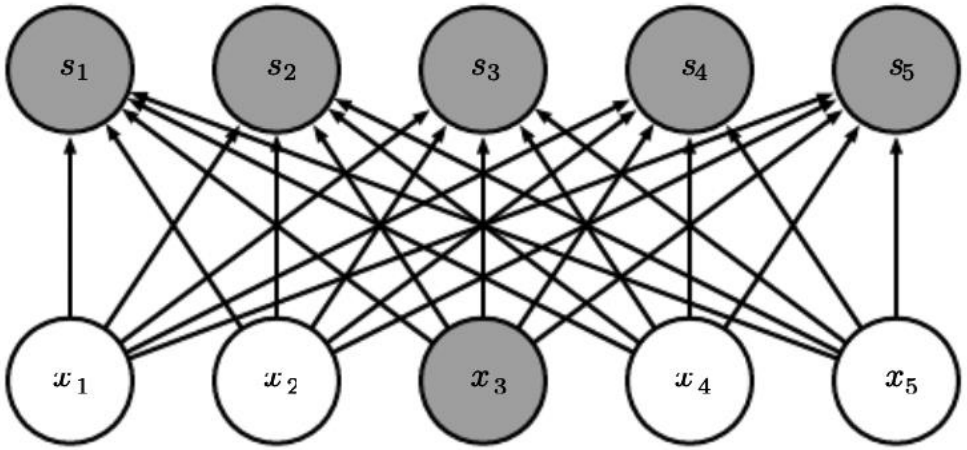
- Sparse connectivity
- Parameter sharing
- Equivariant representations

Sparse connectivity of convolution



Feedforward network (fully connected layer)

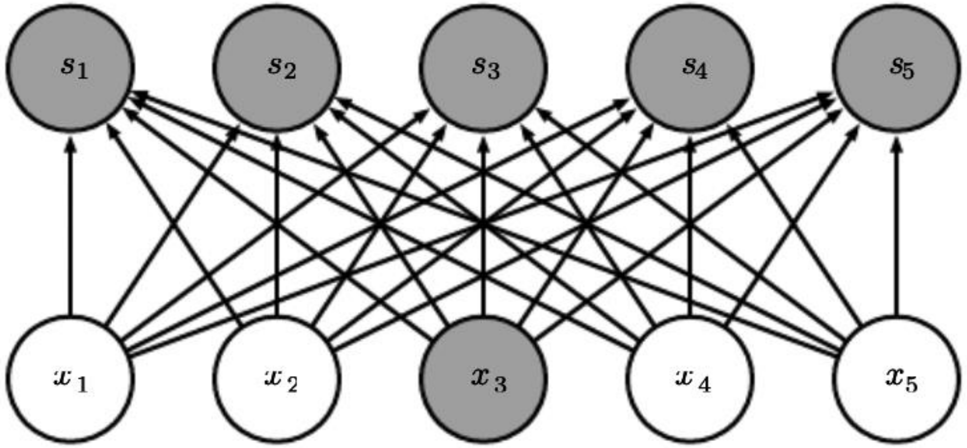
Sparse connectivity of convolution



Feedforward network (fully connected layer)

Q: how many arrows we have?

Sparse connectivity of convolution



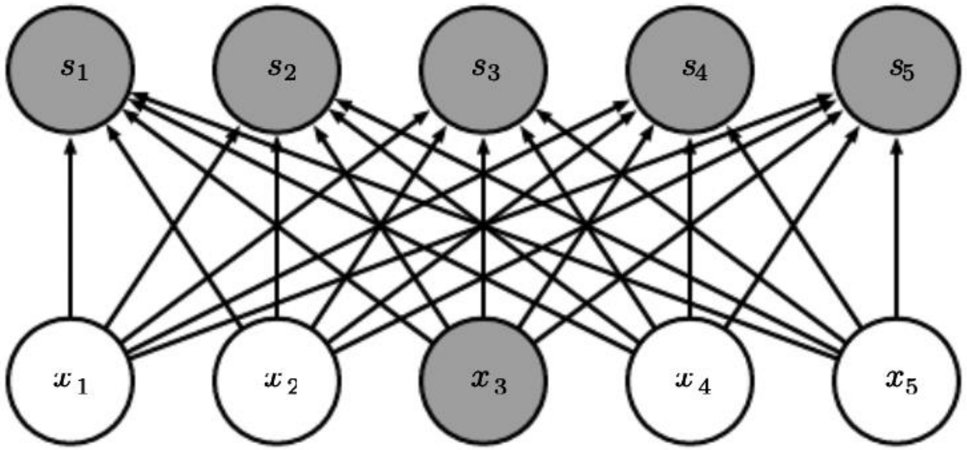
Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5 → 5 arrows
$x_1, \dots, x_5$ → 25 arrows

Sparse connectivity of convolution

$x'w \to s$



Feedforward network (fully connected layer)

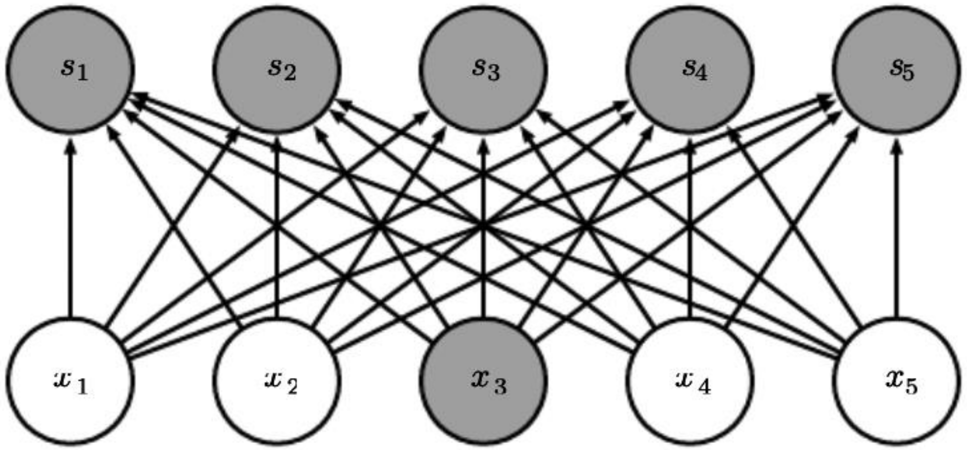
Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows
$x_1, \dots, x_5$→25 arrows

Sparse connectivity of convolution

x and w are vectors; s is a scalar number

$x'w \rightarrow s$



Feedforward network (fully connected layer)

Q: how many arrows we have?

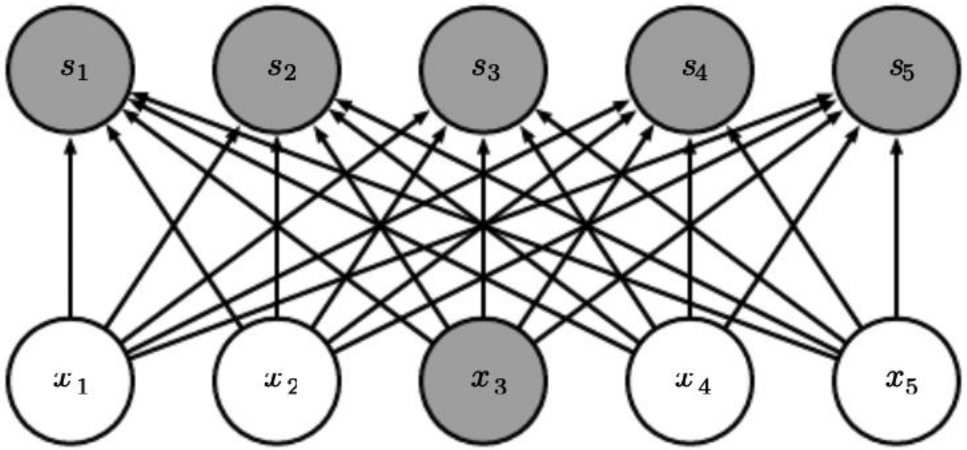
For each x_i: from s_1 to s_5→5 arrows
$x_1, \dots, x_5$→25 arrows

Sparse connectivity of convolution

x and w are vectors; s is a scalar number

$x'w \rightarrow s$

$$x'w_1 = \sum_{i=1}^{5} x_i w_{1,i}$$



Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5 → 5 arrows
$x_1, \dots, x_5$ → 25 arrows

Sparse connectivity of convolution

$x'w \rightarrow s$

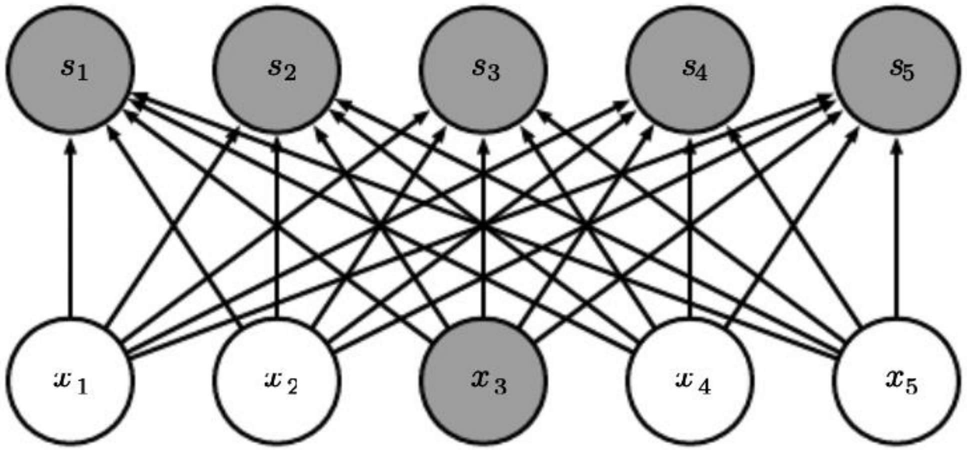
$$x'w_1 = \sum_{i=1}^{5} x_i w_{1,i}$$

$$x'w_2 = \sum_{i=1}^{5} x_i w_{2,i}$$

$$x'w_3 = \sum_{i=1}^{5} x_i w_{3,i}$$

$$x'w_4 = \sum_{i=1}^{5} x_i w_{4,i}$$

$$x'w_5 = \sum_{i=1}^{5} x_i w_{5,i}$$



Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows
$x_1, \ldots, x_5$→25 arrows

Sparse connectivity of convolution

$$x'w \to s$$

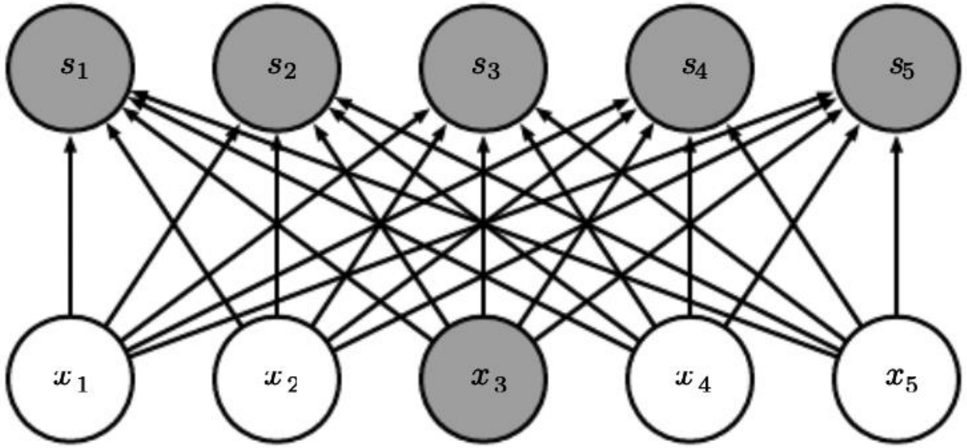
$$x'w_1 = \sum_{i=1}^{5} x_i w_{1,i}$$

$$x'w_2 = \sum_{i=1}^{5} x_i w_{2,i}$$

$$x'w_3 = \sum_{i=1}^{5} x_i w_{3,i}$$

$$x'w_4 = \sum_{i=1}^{5} x_i w_{4,i}$$

$$x'w_5 = \sum_{i=1}^{5} x_i w_{5,i}$$



Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows
$x_1, \dots, x_5$→25 arrows

Sparse connectivity of convolution

$x'w \to s$

$$x'w_1 = \sum_{i=1}^{5} x_i w_{1,i}$$

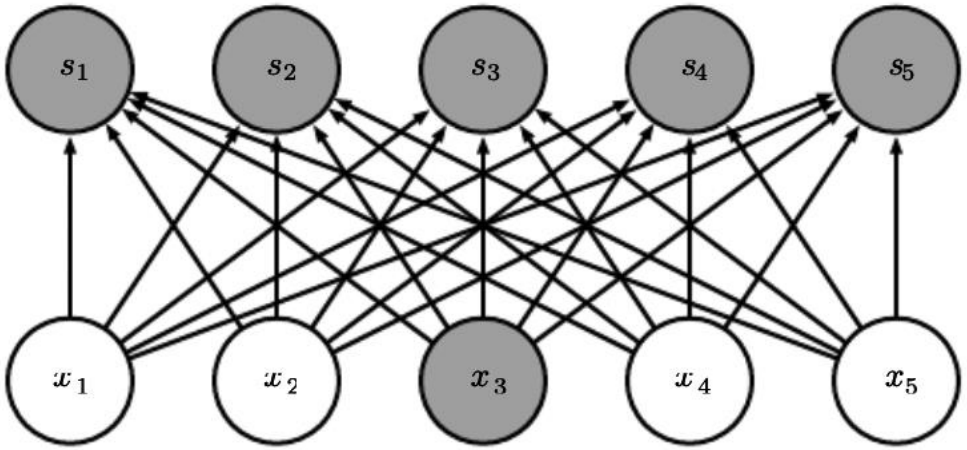
$$x'w_2 = \sum_{i=1}^{5} x_i w_{2,i}$$

$$x'w_3 = \sum_{i=1}^{5} x_i w_{3,i}$$

$$x'w_4 = \sum_{i=1}^{5} x_i w_{4,i}$$

$$x'w_5 = \sum_{i=1}^{5} x_i w_{5,i}$$

$W = [w_1, w_2, w_3, w_4, w_5]$



Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows
$x_1, \dots, x_5$→25 arrows

Sparse connectivity of convolution

$x'w \rightarrow s$

$$x'w_1 = \sum_{i=1}^{5} x_i w_{1,i}$$

$$x'w_2 = \sum_{i=1}^{5} x_i w_{2,i}$$

$$x'w_3 = \sum_{i=1}^{5} x_i w_{3,i}$$

$$x'w_4 = \sum_{i=1}^{5} x_i w_{4,i}$$

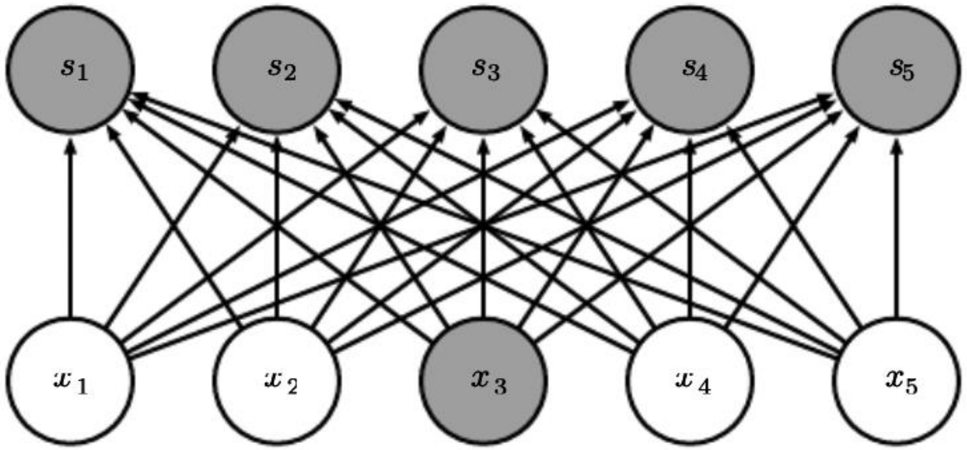
$$x'w_5 = \sum_{i=1}^{5} x_i w_{5,i}$$

W

$=$

$[w_1, w_2, w_3, w_4, w_5]$

(5x5 weight matrix)



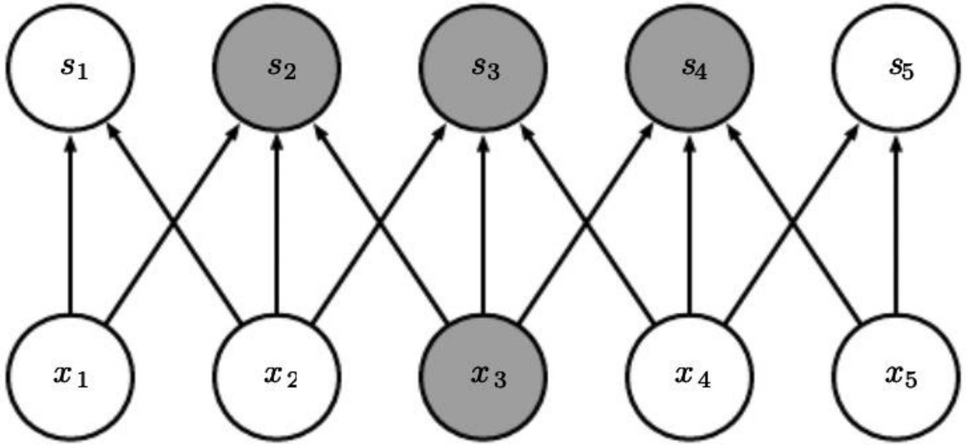
Feedforward network (fully connected layer)

Q: how many arrows we have?

For each x_i: from s_1 to s_5→5 arrows

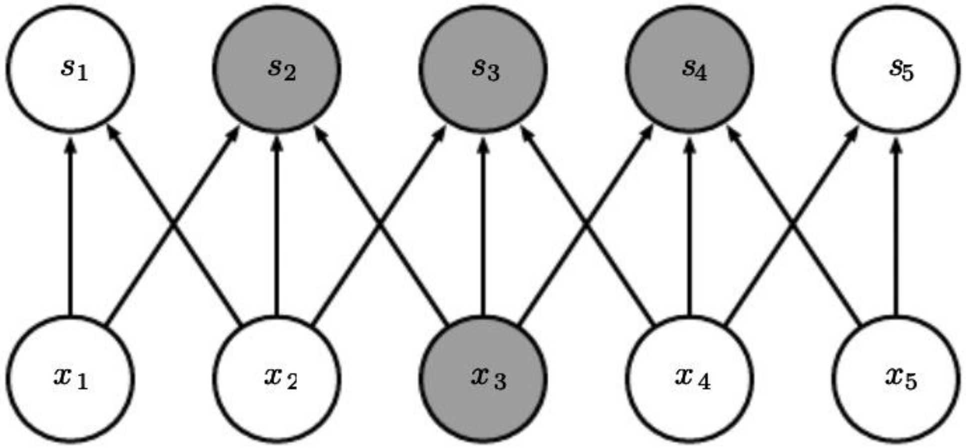
$x_1, \dots, x_5$→25 arrows

Sparse connectivity of convolution



Convolutional layers

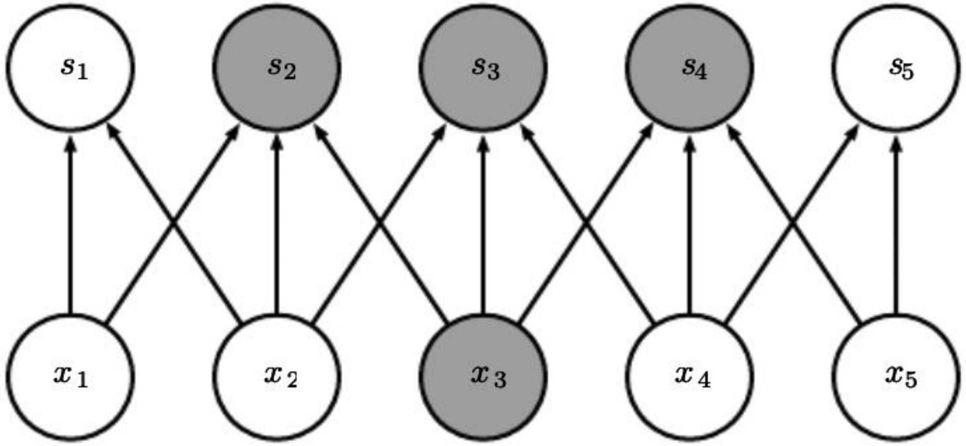
Sparse connectivity of convolution



Convolutional layers

Q: how many arrows we have?

Sparse connectivity of convolution

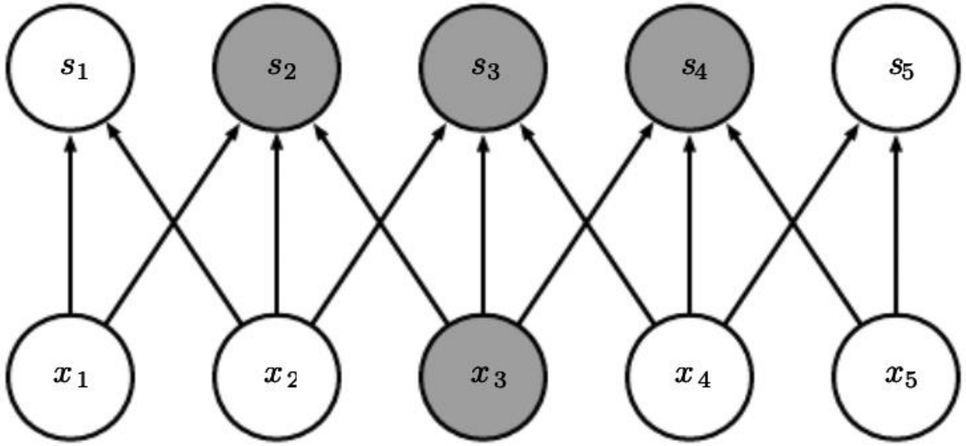


Convolutional layers

Q: how many arrows we have?

For each x_i: connect to 3 s outputs

Sparse connectivity of convolution

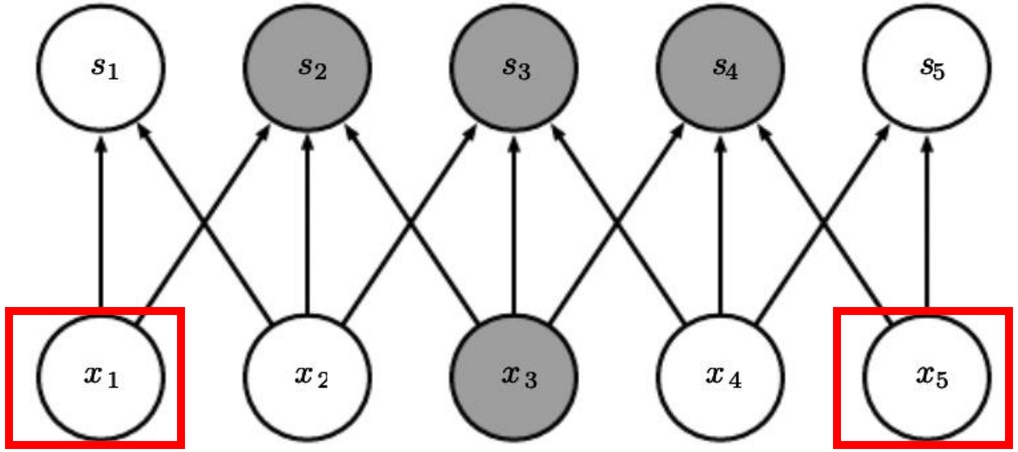


Convolutional layers

Q: how many arrows we have?

For each x_i: connect to 3 s outputs
$x_1, \ldots, x_5$ → 3x5-2=13 arrows

Sparse connectivity of convolution

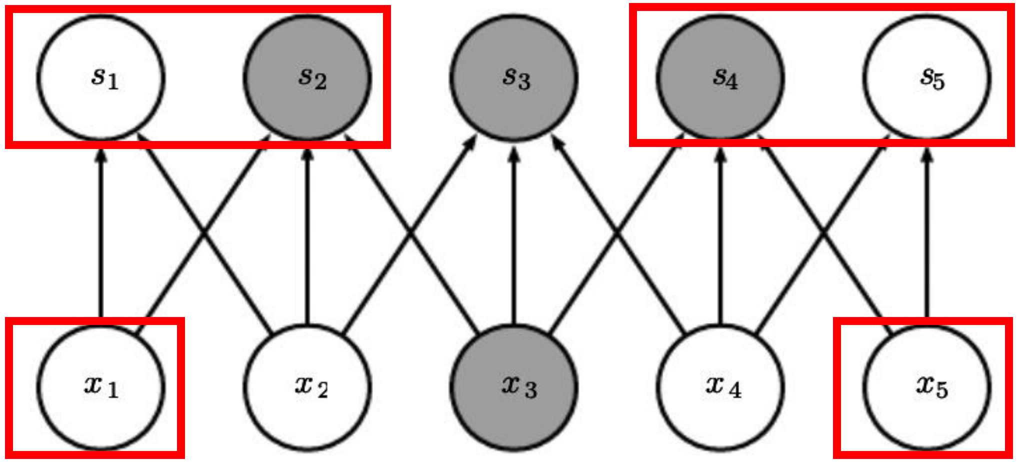


Convolutional layers

Q: how many arrows we have?

For each x_i: connect to 3 s outputs
$x_1, \dots, x_5$→3x5-2=13 arrows

Sparse connectivity of convolution



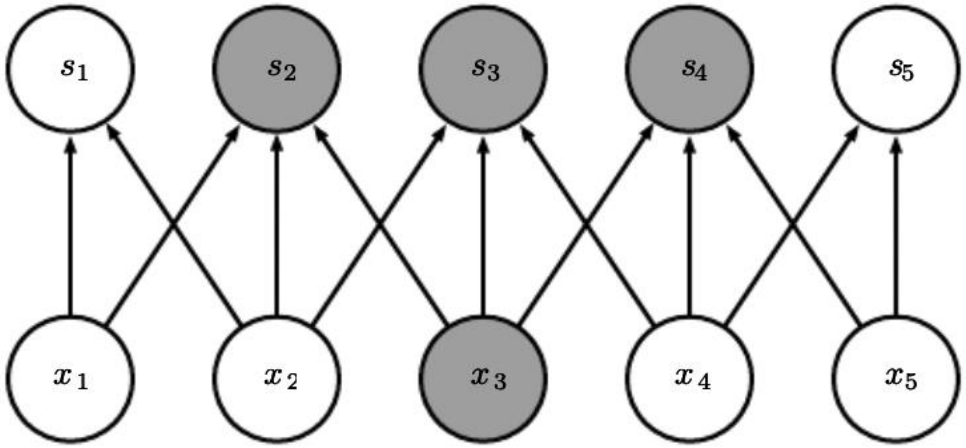
Convolutional layers

Q: how many arrows we have?

For each x_i: connect to 3 s outputs
$x_1, \dots, x_5 \rightarrow$ 3x5-2=13 arrows

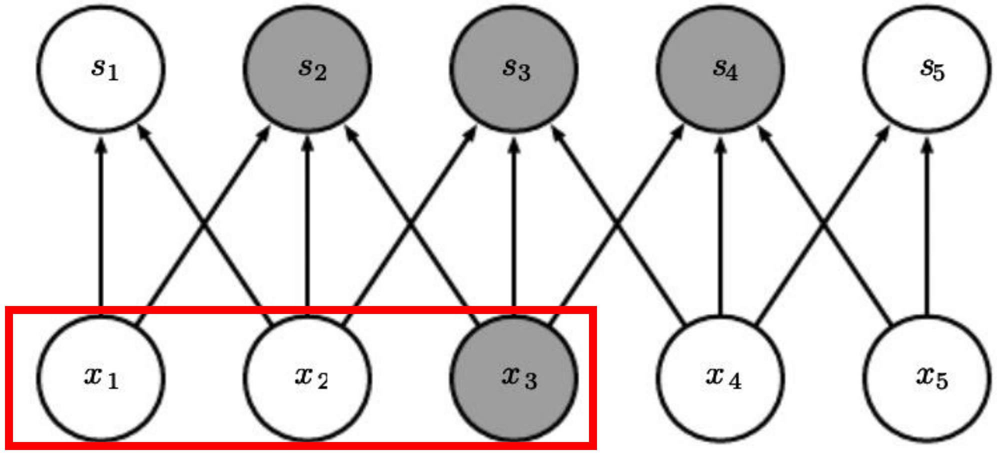
Sparse connectivity of convolution

The view of convolutional kernel/filter



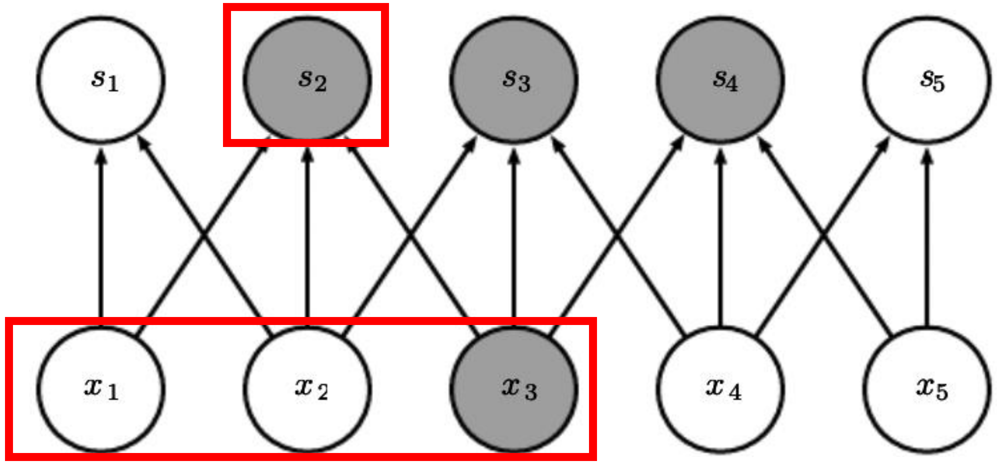
Sparse connectivity of convolution

The view of convolutional kernel/filter



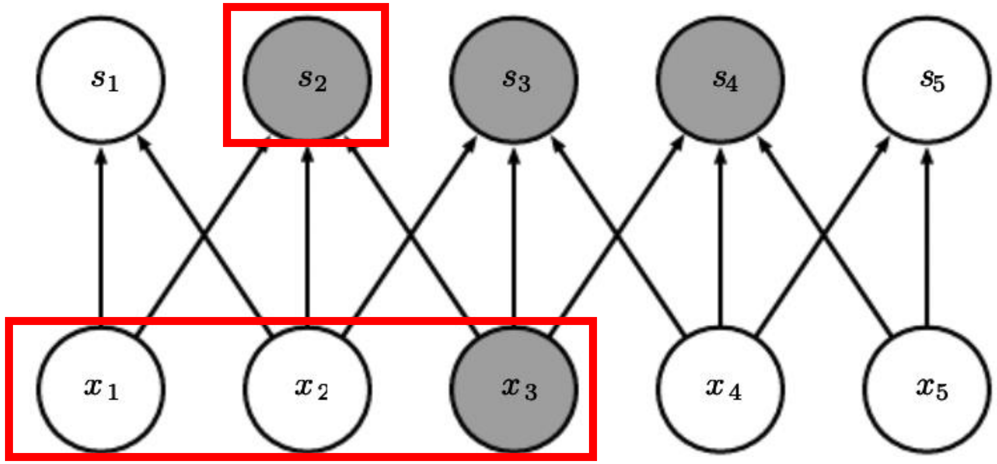
Sparse connectivity of convolution

The view of convolutional kernel/filter



Sparse connectivity of convolution

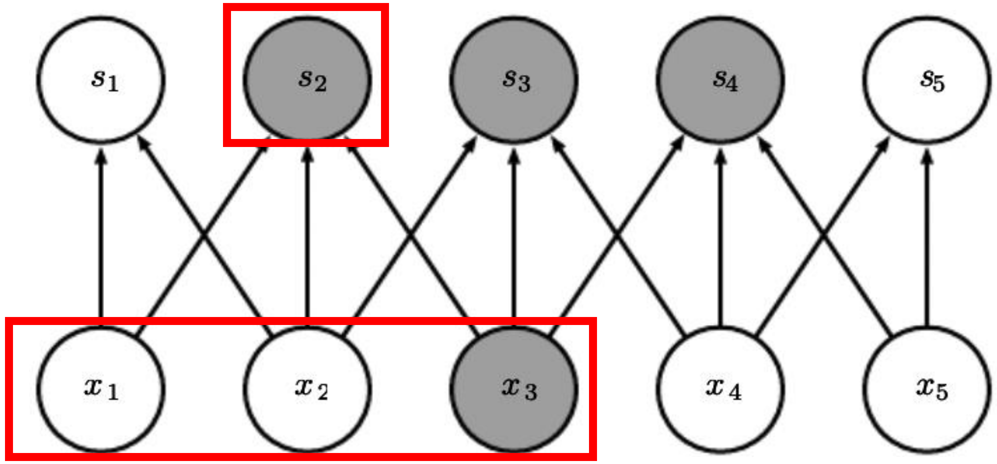
The view of convolutional kernel/filter



Q: filter size and stride?

Sparse connectivity of convolution

The view of convolutional kernel/filter

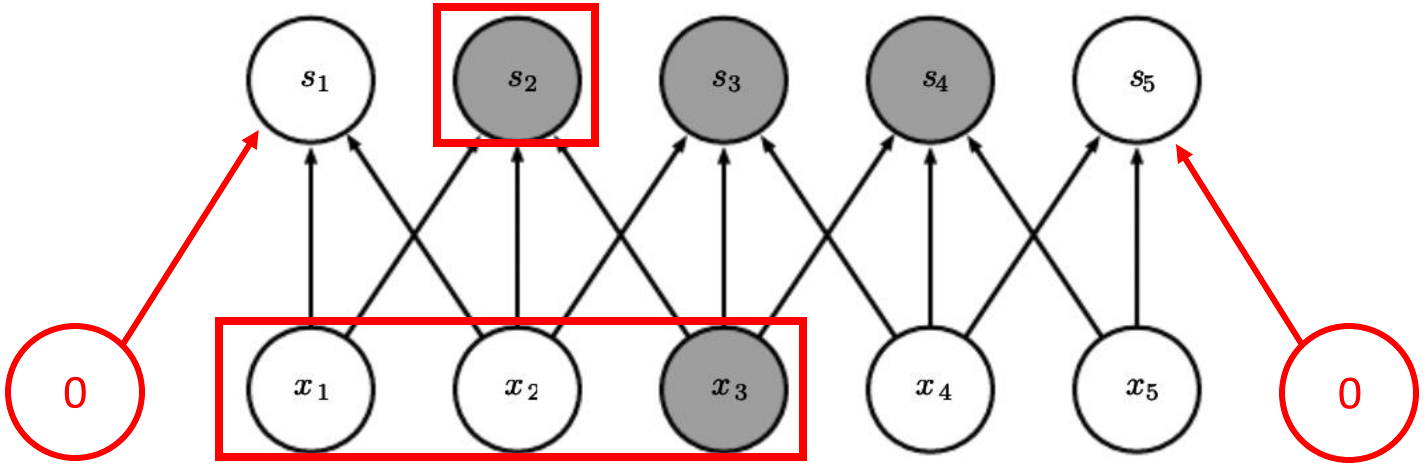


Q: filter size and stride?

Filter size = 3 + stride = 1 with 0-pading

Sparse connectivity of convolution

The view of convolutional kernel/filter

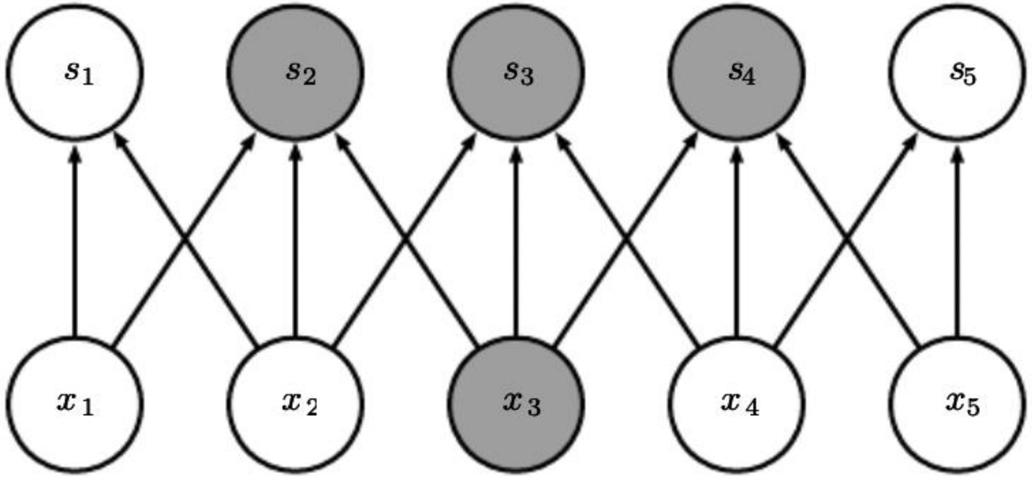


Q: filter size and stride?

Filter size = 3 + stride = 1 with 0-pading

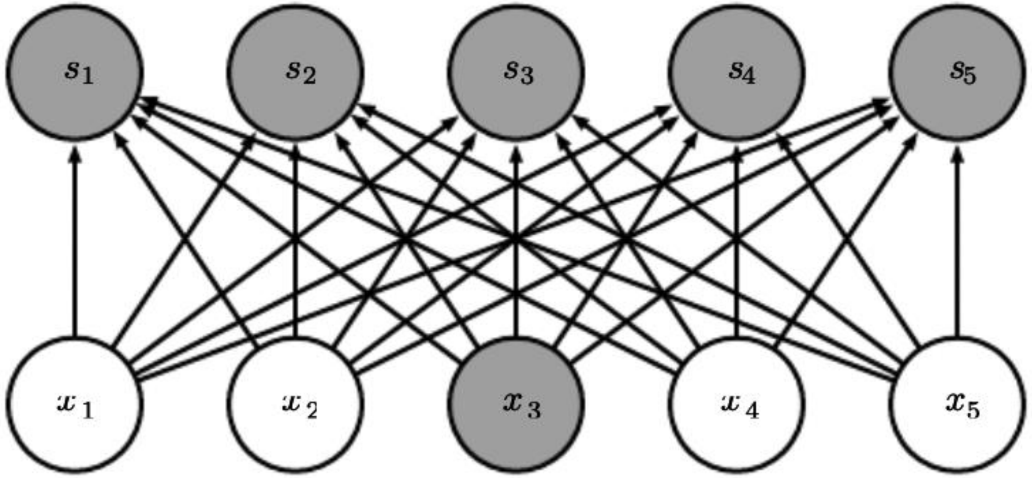
Sparse connectivity of convolution

Sparse connection



Weights→13 scalar numbers

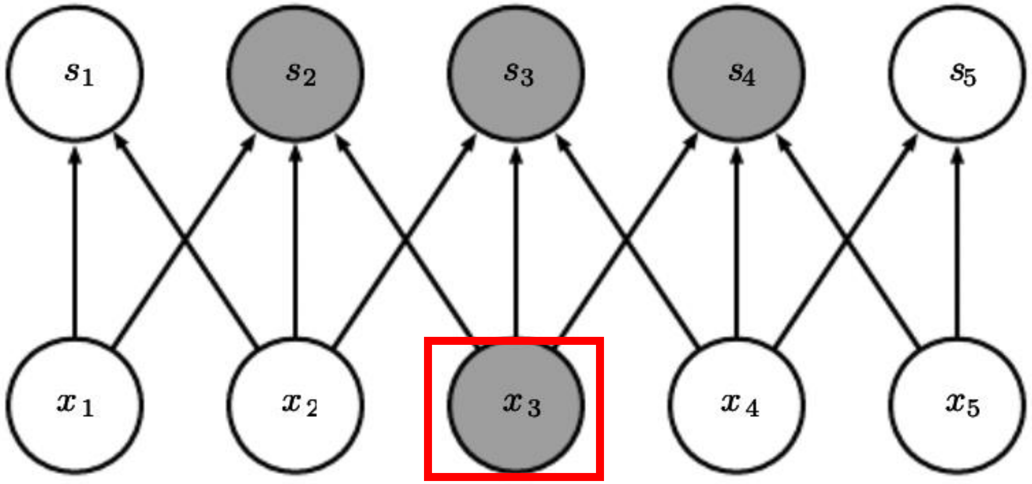
Dense connection



Weights→25 scalar numbers

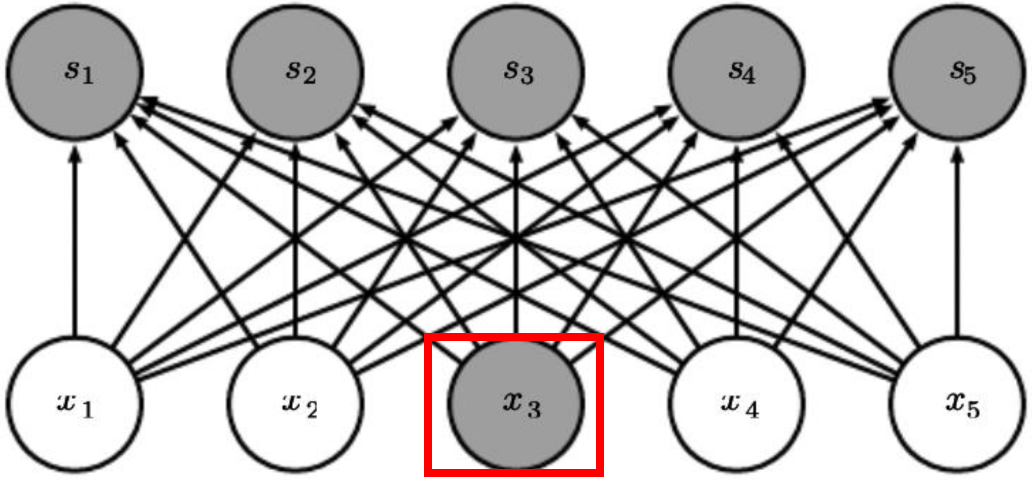
Sparse connectivity of convolution

Sparse connection



Weights→13 scalar numbers

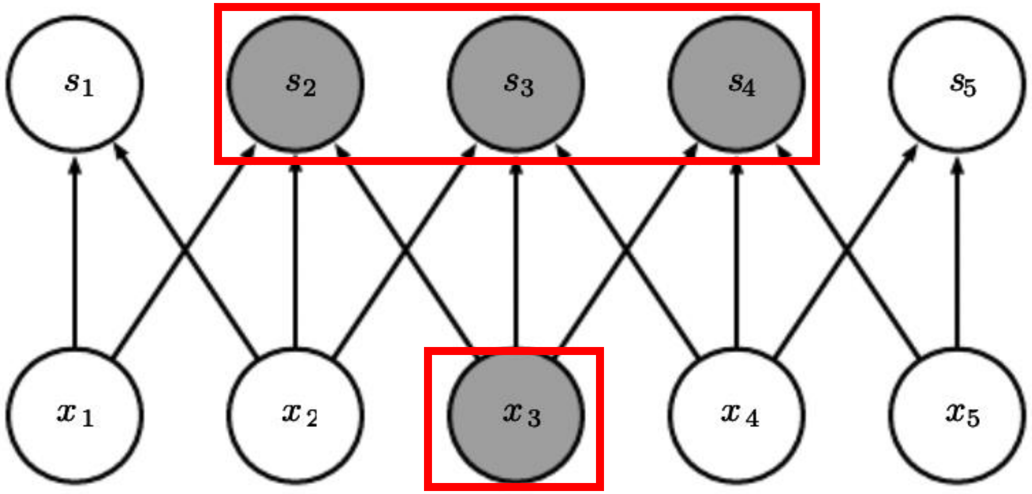
Dense connection



Weights→25 scalar numbers

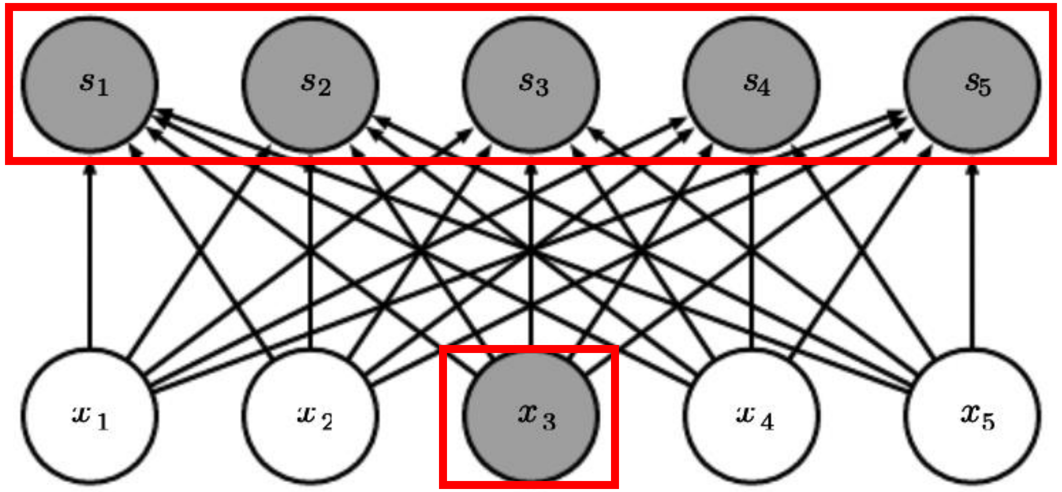
Sparse connectivity of convolution

Sparse connection



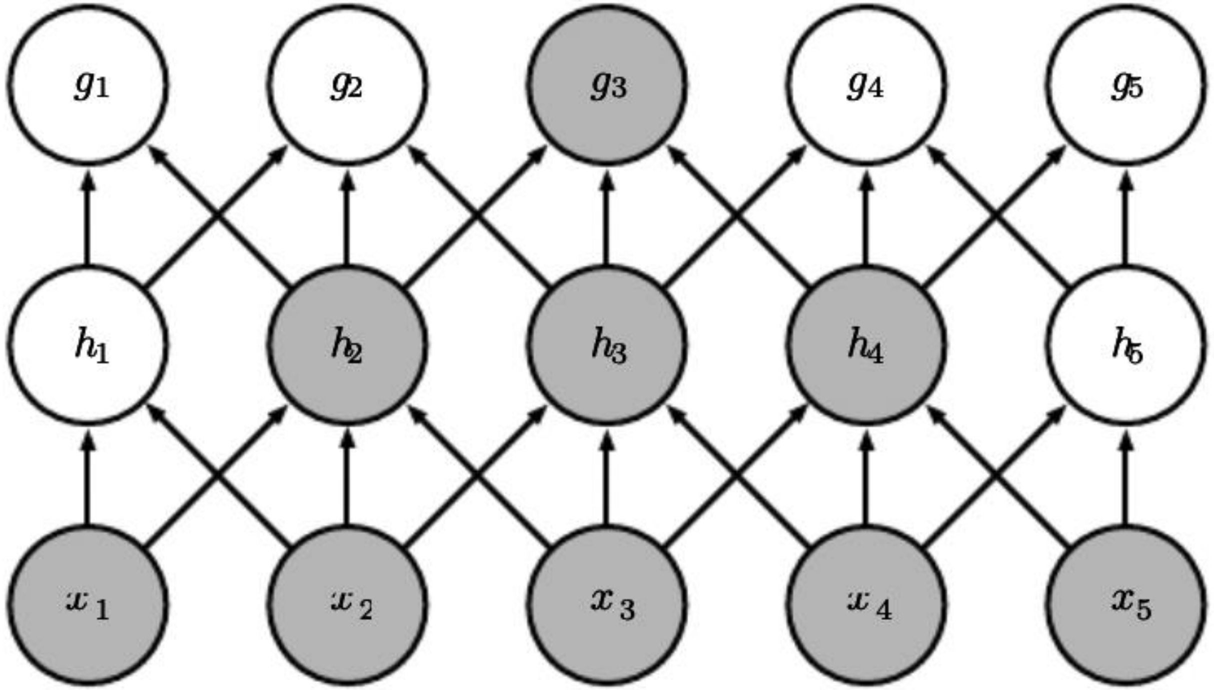
Weights→13 scalar numbers

Dense connection



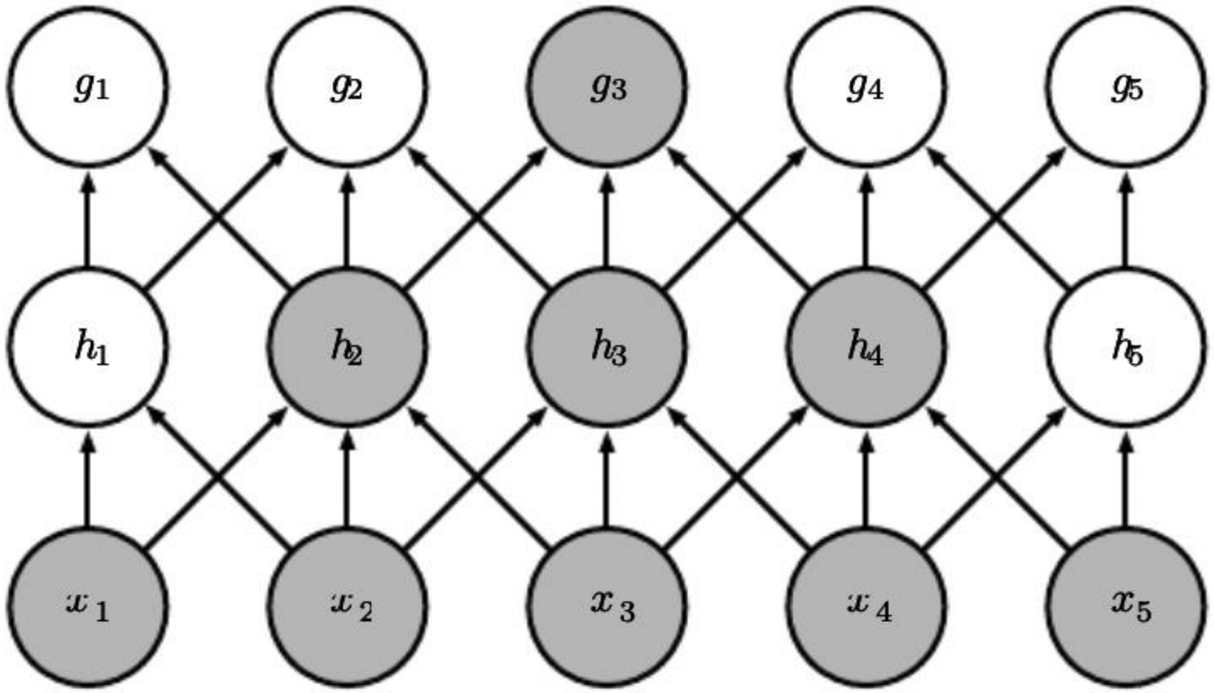
Weights→25 scalar numbers

Sparse connectivity of convolution



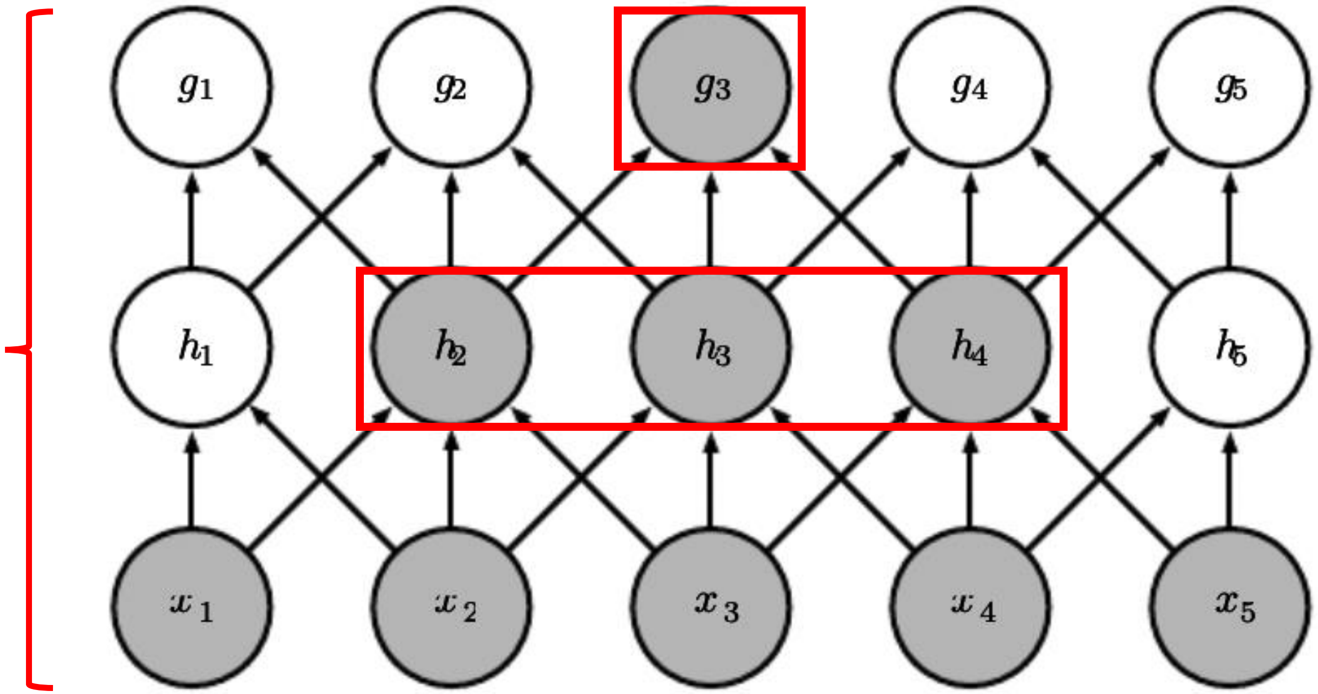
Sparse connectivity of convolution

3 layers

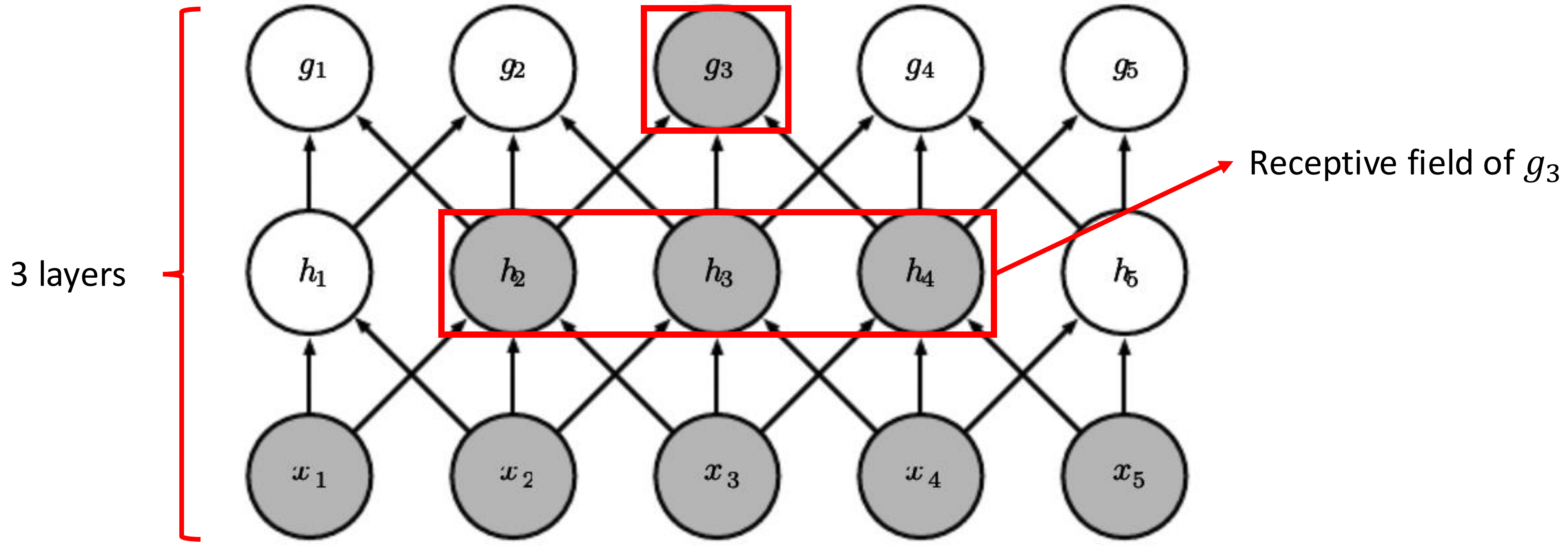


Sparse connectivity of convolution

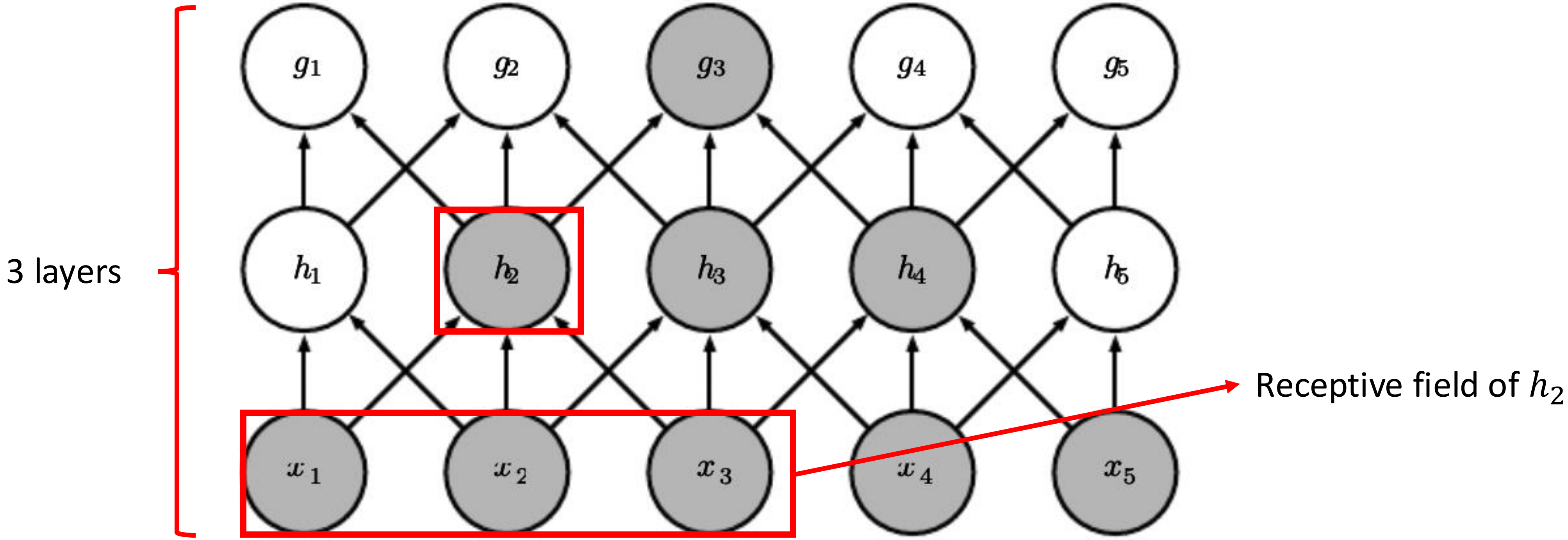
3 layers



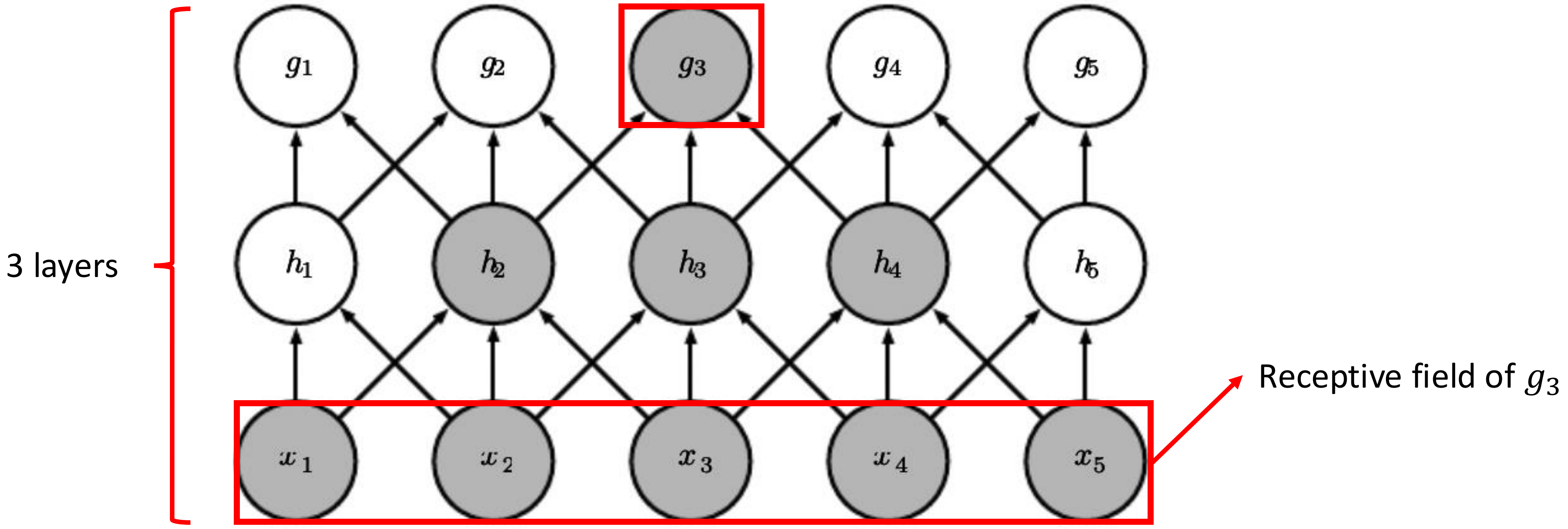
Sparse connectivity of convolution



Sparse connectivity of convolution

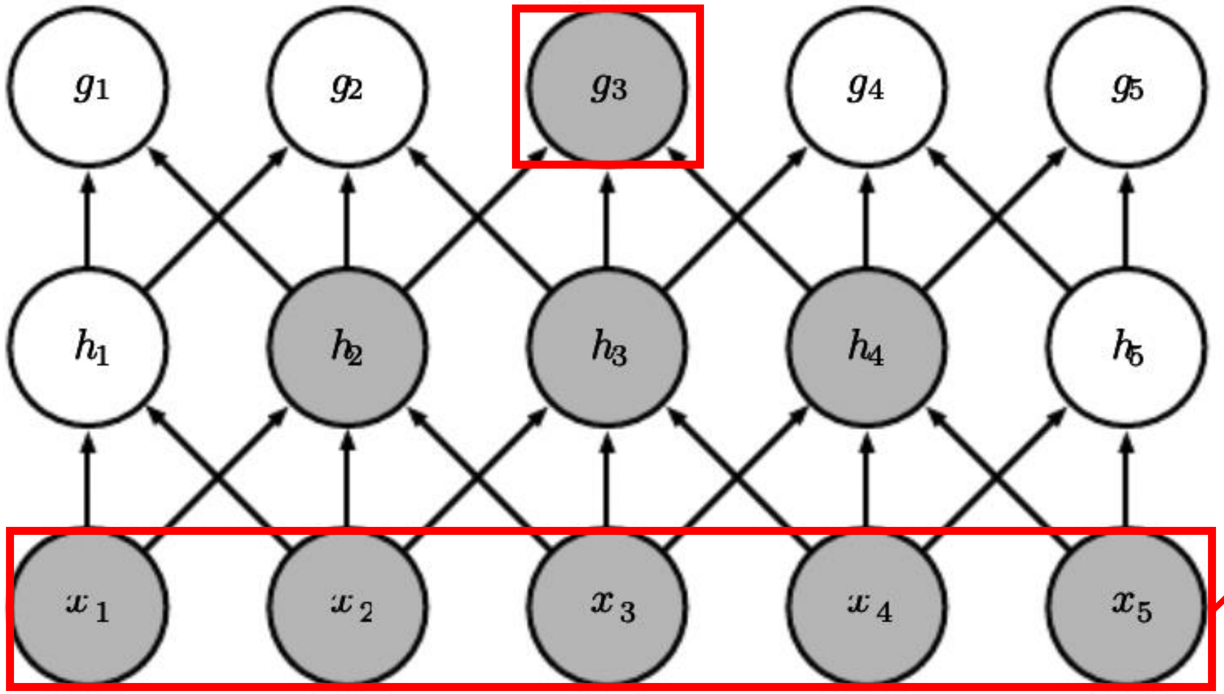


Sparse connectivity of convolution



Sparse connectivity of convolution

3 layers

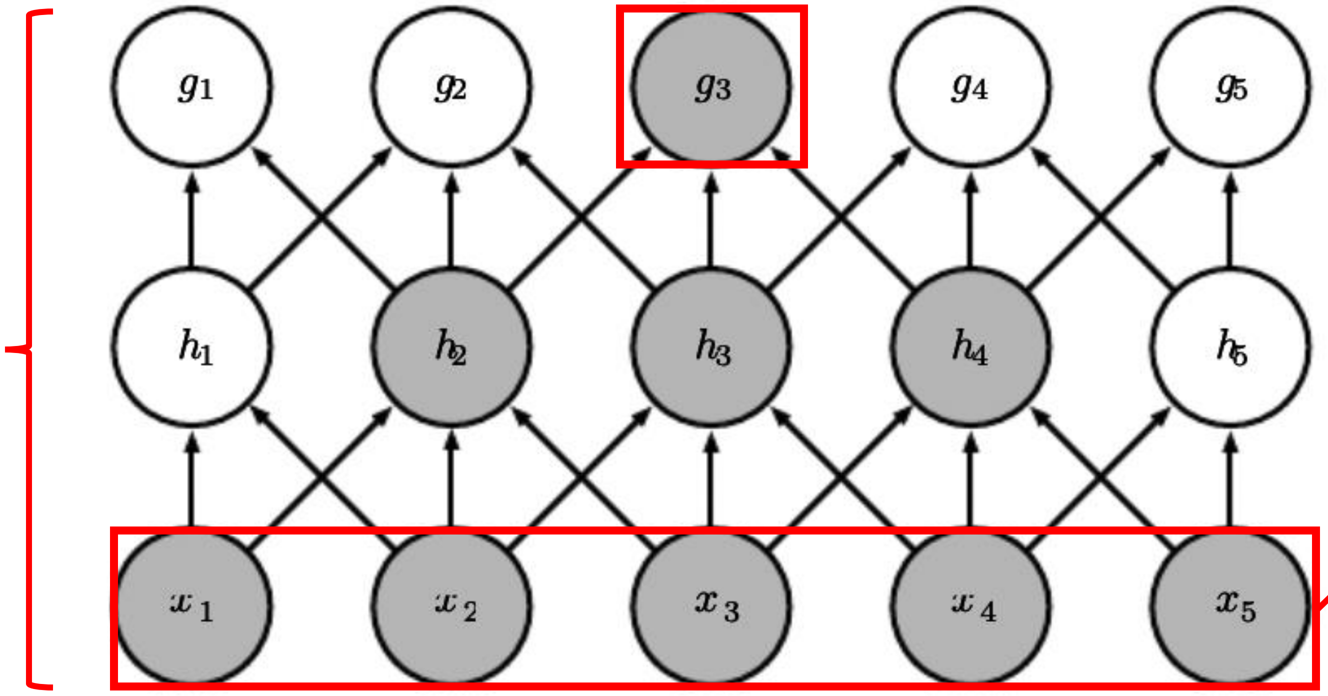


Receptive field of g_3

Deep layers has larger receptive field than shallow layers

Sparse connectivity of convolution

3 layers

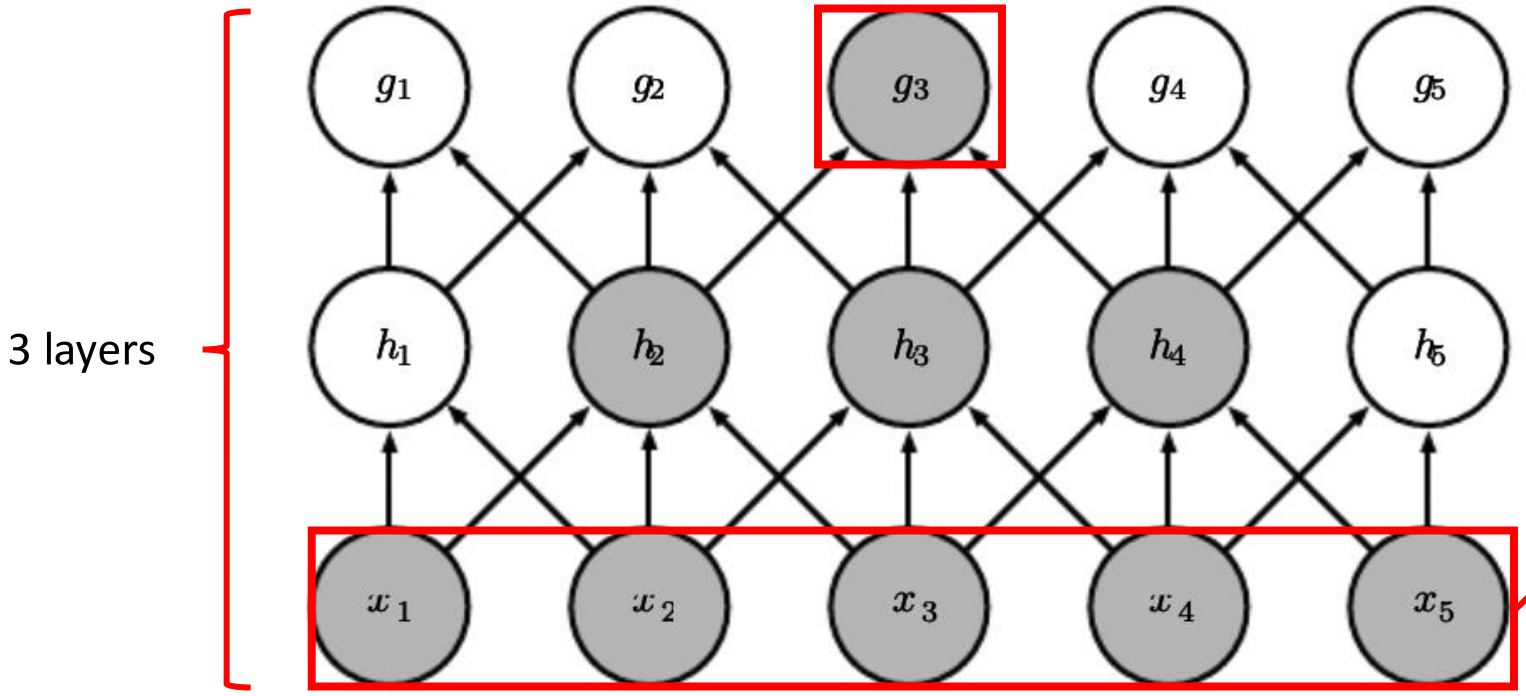


Receptive field of g_3

Units in deep layer indirectly connect to all/most input image

Deep layers has larger receptive field than shallow layers

Sparse connectivity of convolution



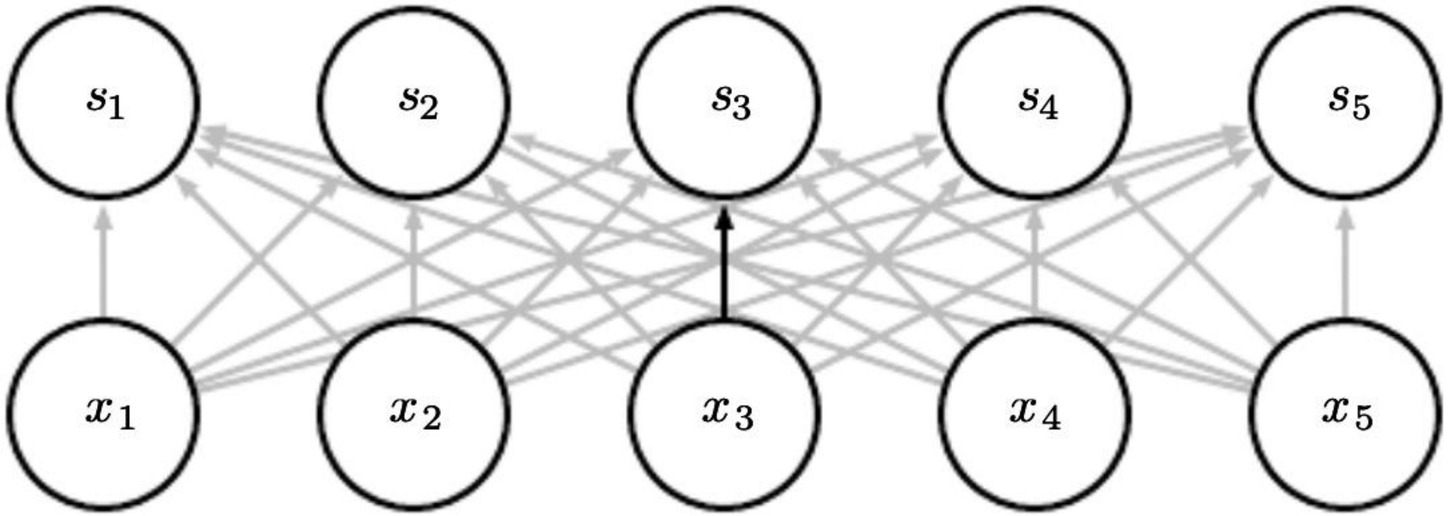
Receptive field of g_3

Units in deep layer indirectly connect to all/most input image

Deep layers has larger receptive field than shallow layers

Q: larger stride of convolution filter → increase receptive field?

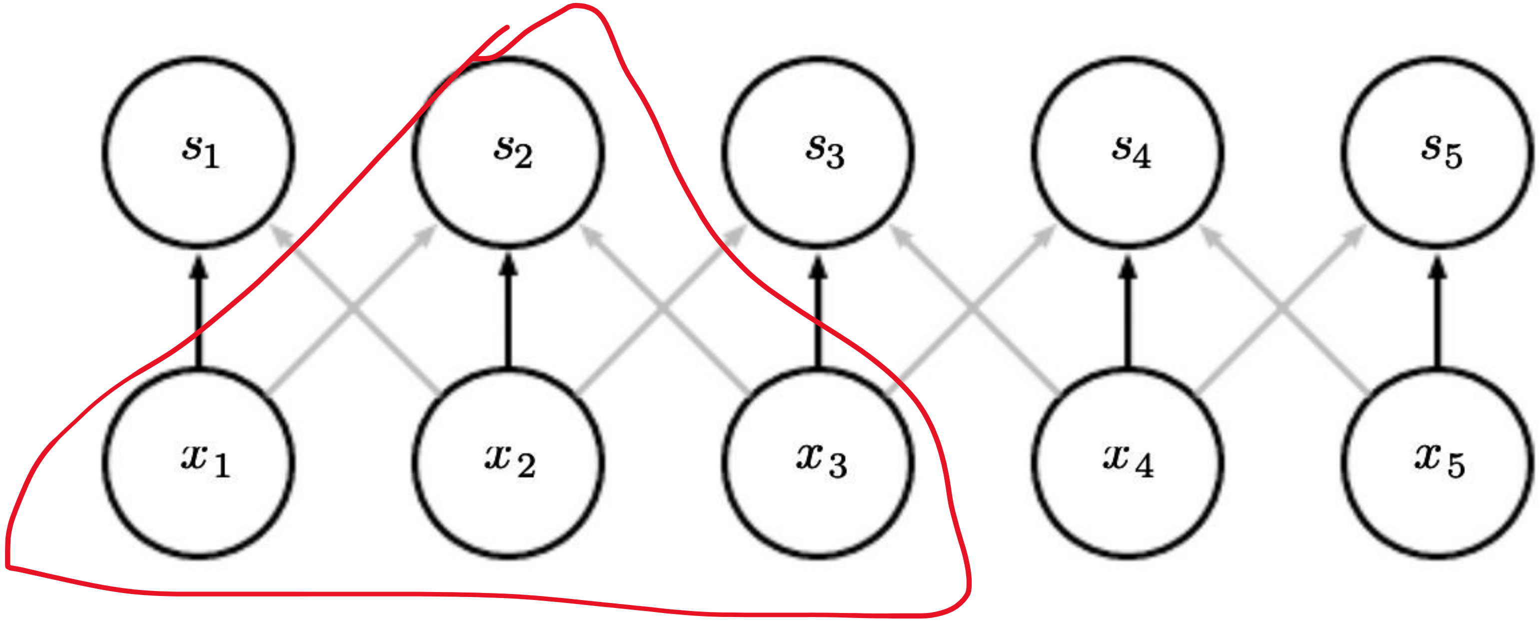
Parameter sharing



In MLP (FC layer): $w^T x$

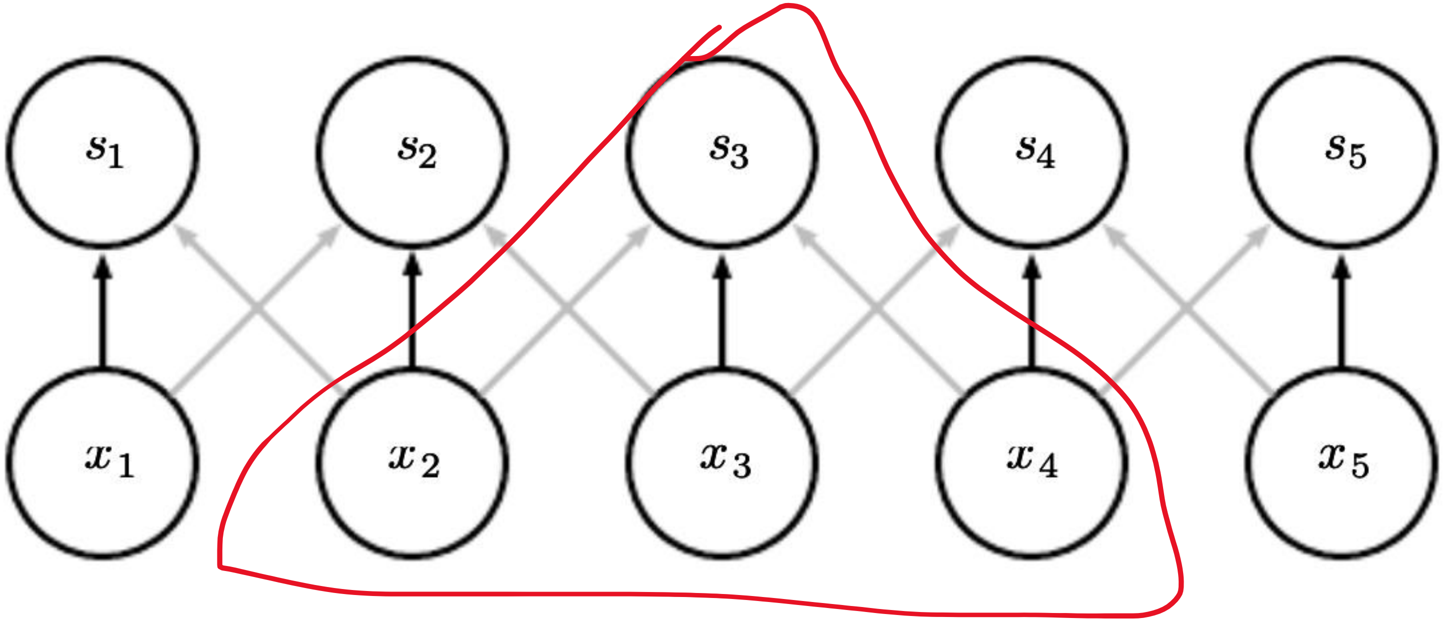
Parameter sharing

Consider the same filter



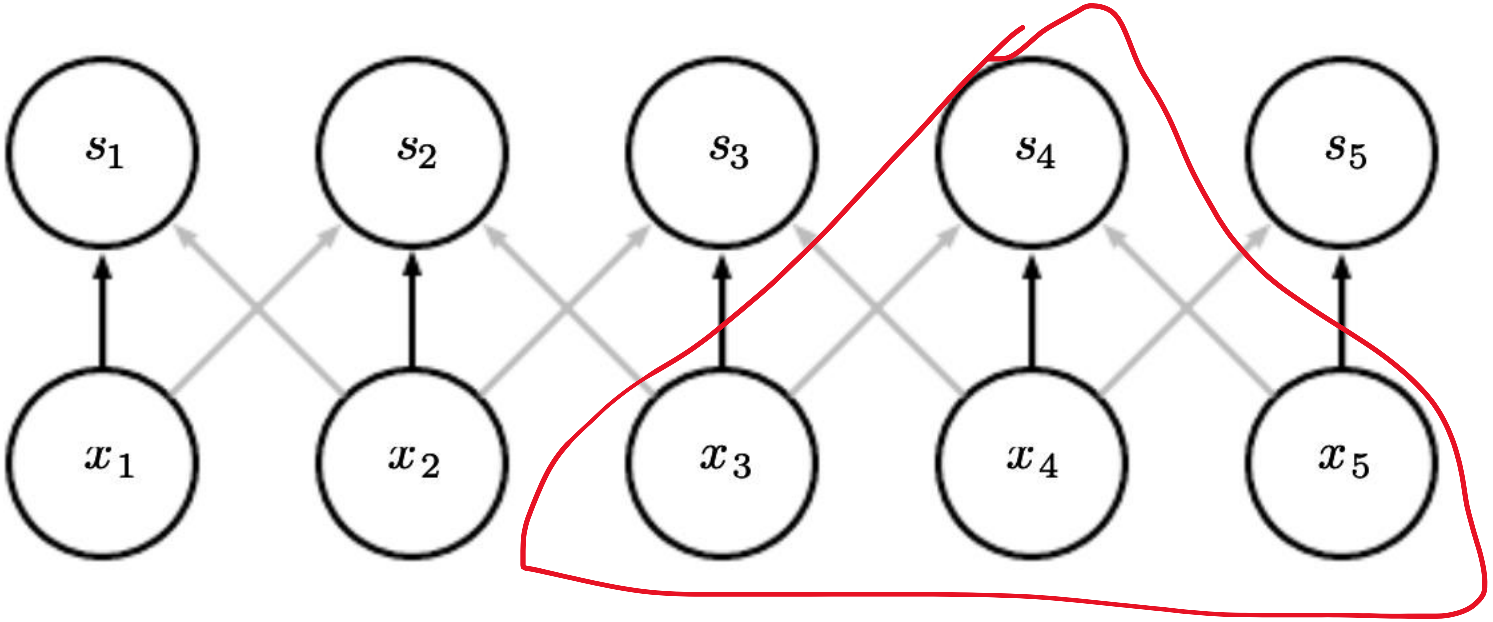
Parameter sharing

Consider the same filter



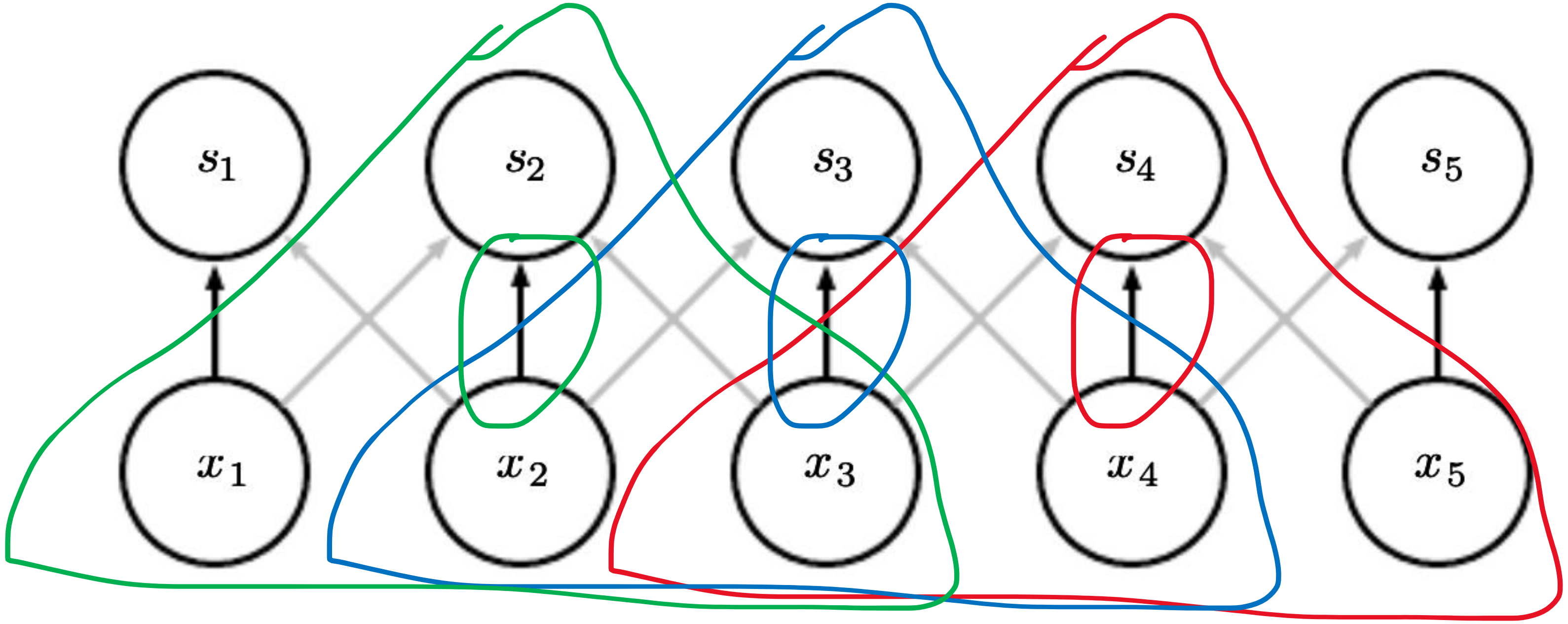
Parameter sharing

Consider the same filter



Parameter sharing

Consider the same filter (but different part of input feat. map)



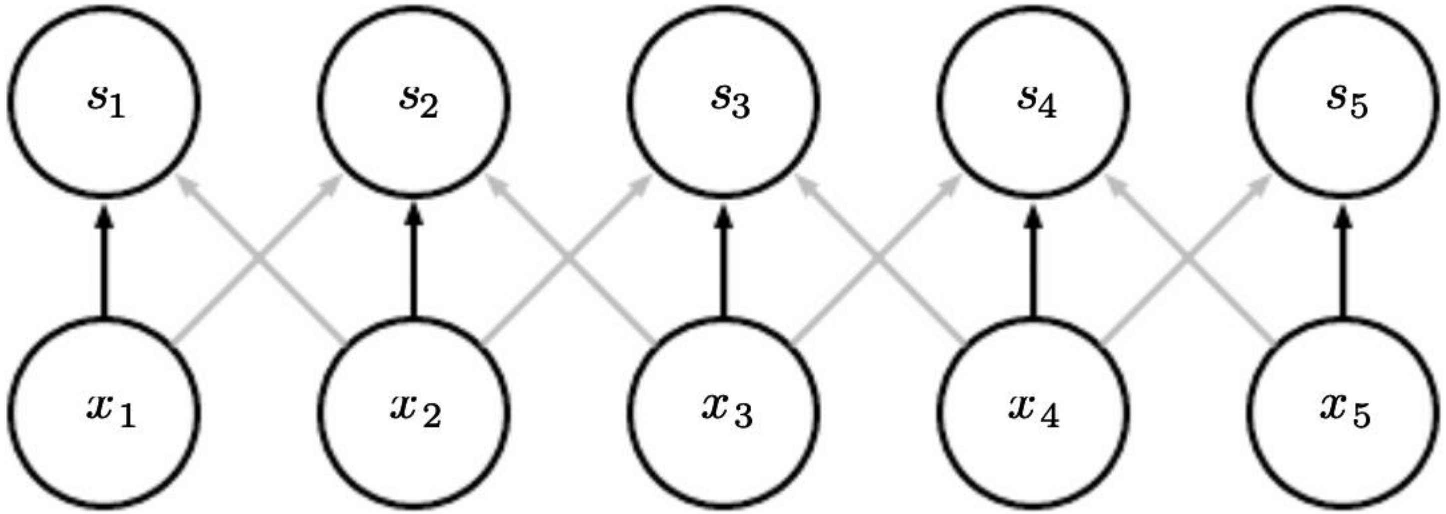
In convlayer:

$w^T(x_1; x_2; x_3)$

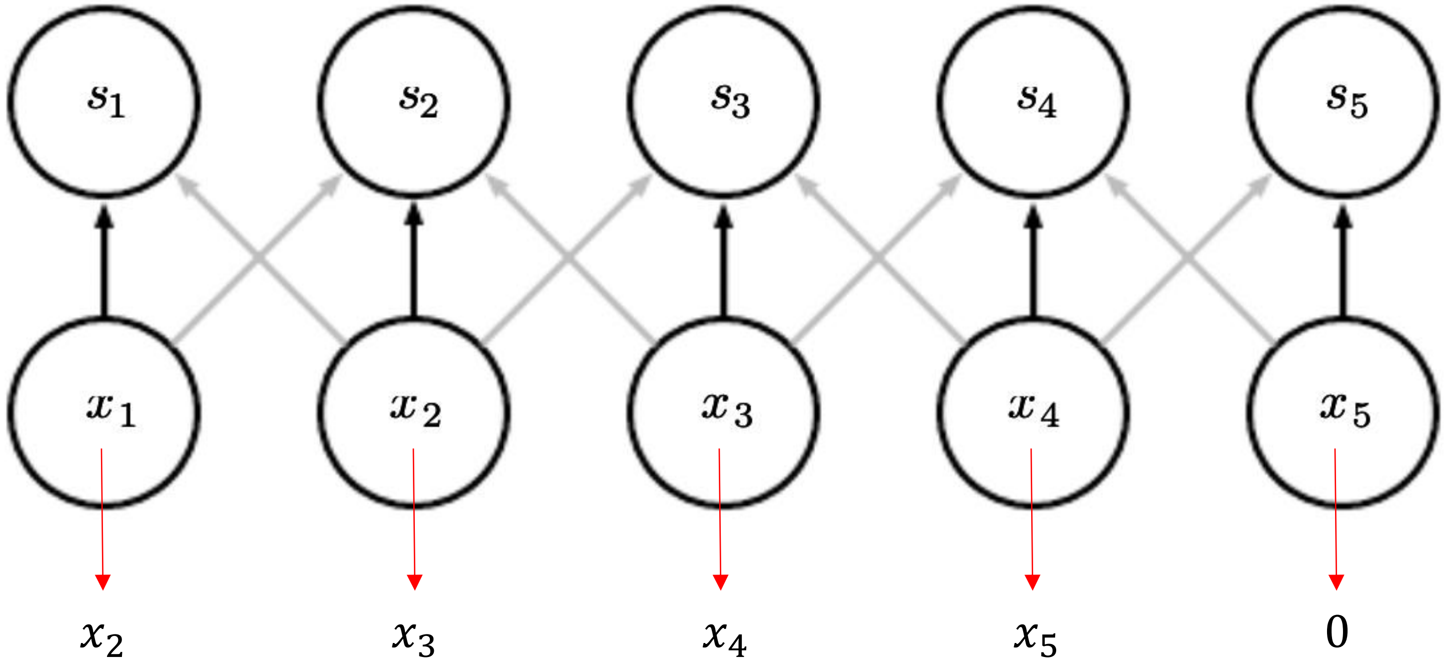
$w^T(x_2; x_3; x_4)$

$w^T(x_3; x_4; x_5)$

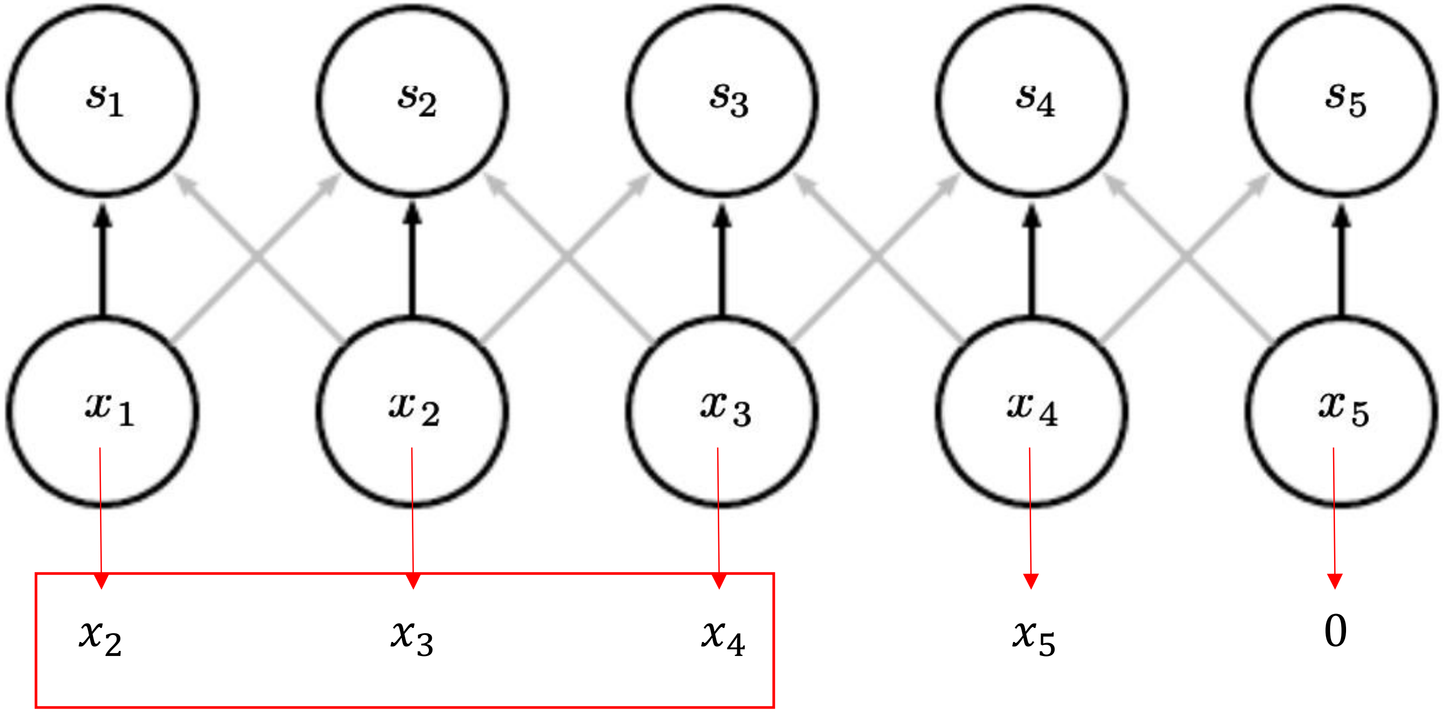
Equivariance



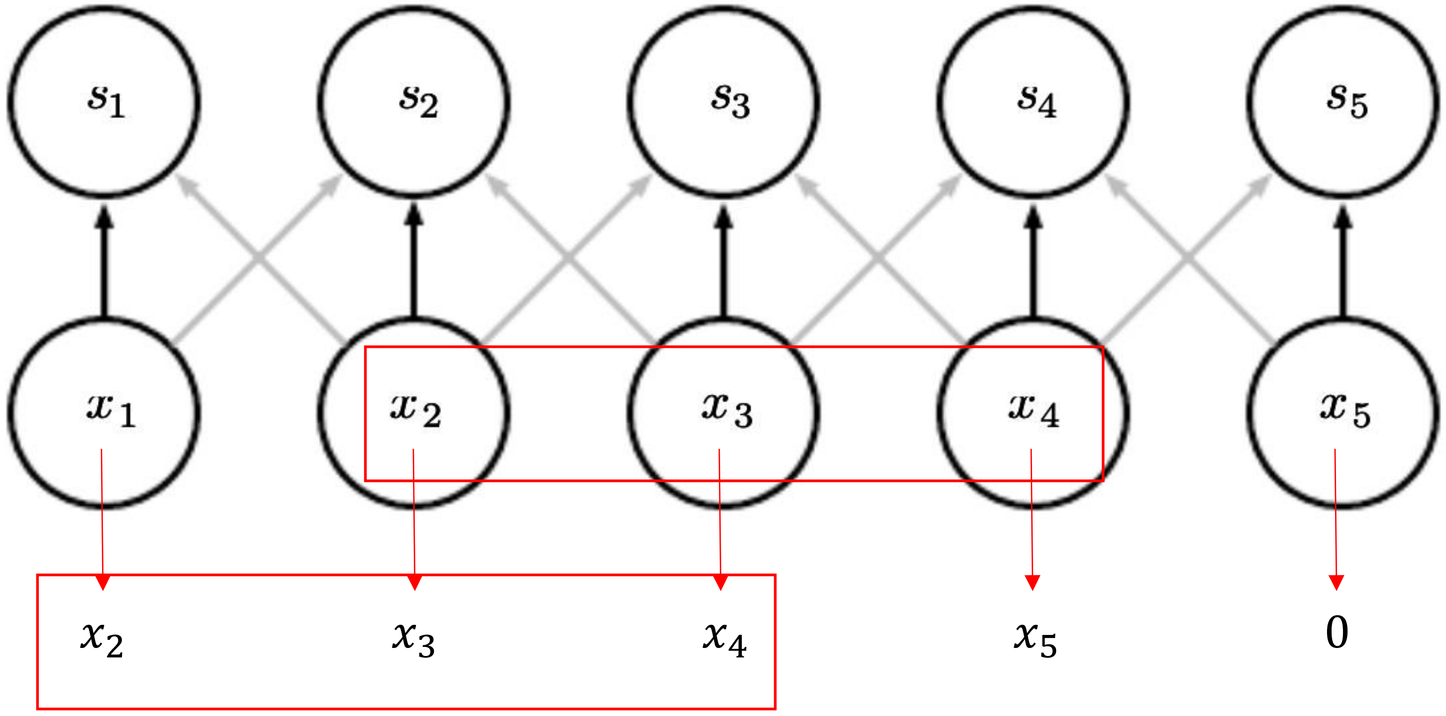
Equivariance



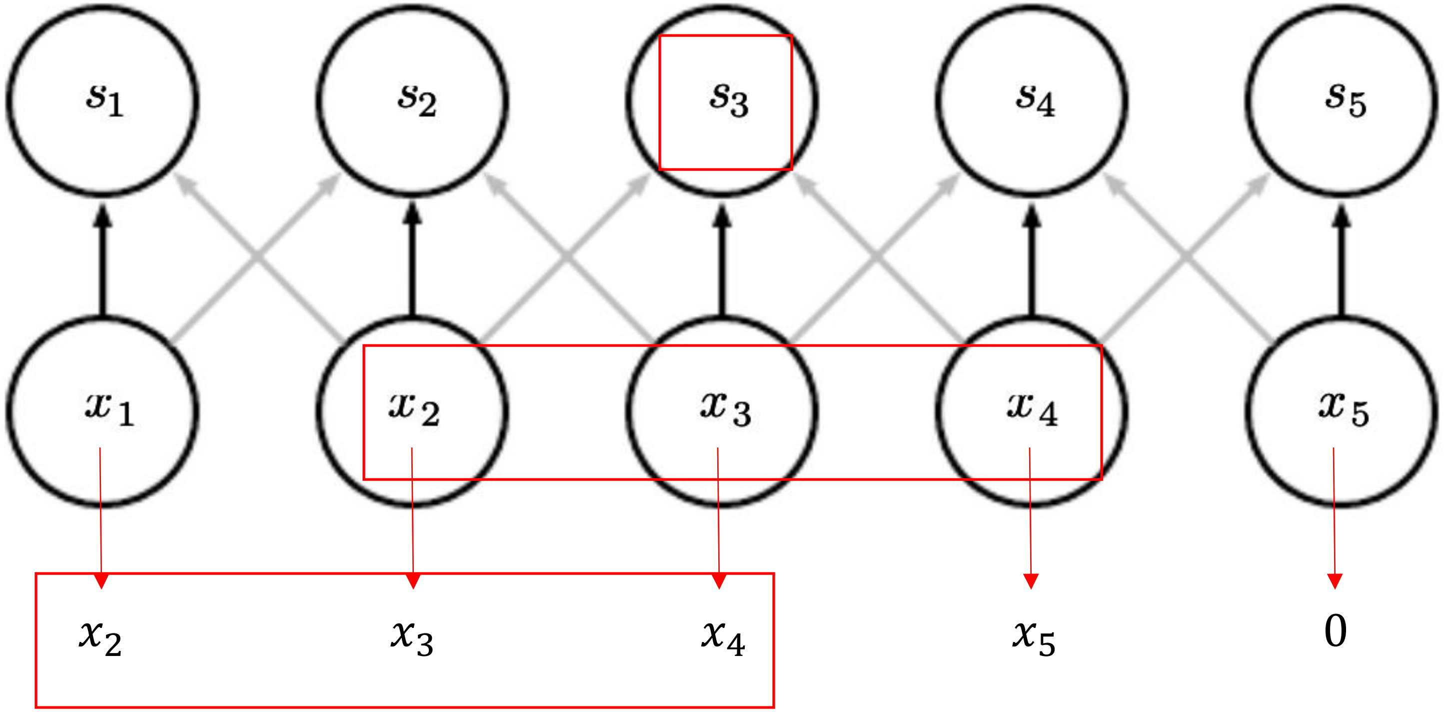
Equivariance



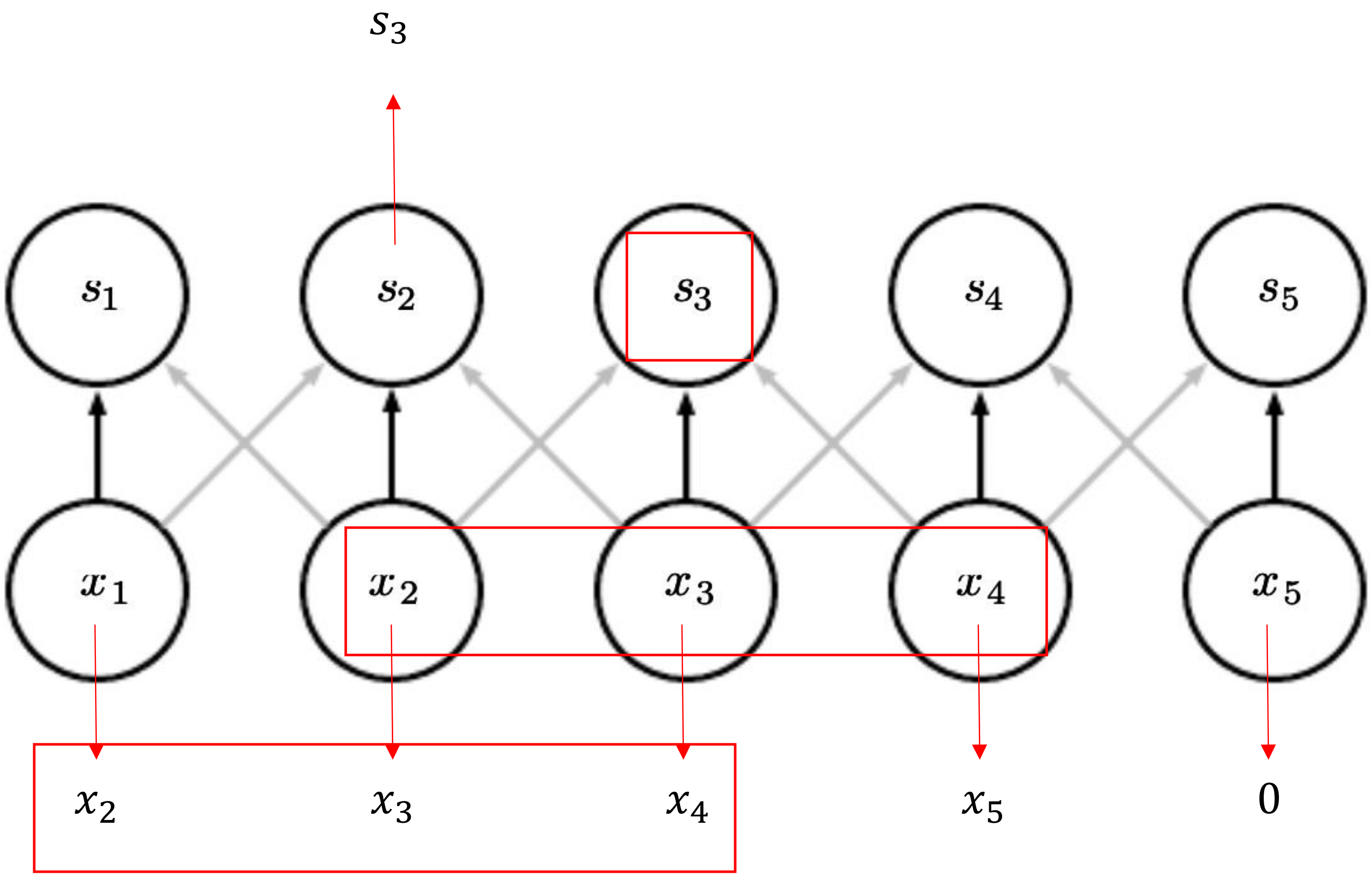
Equivariance



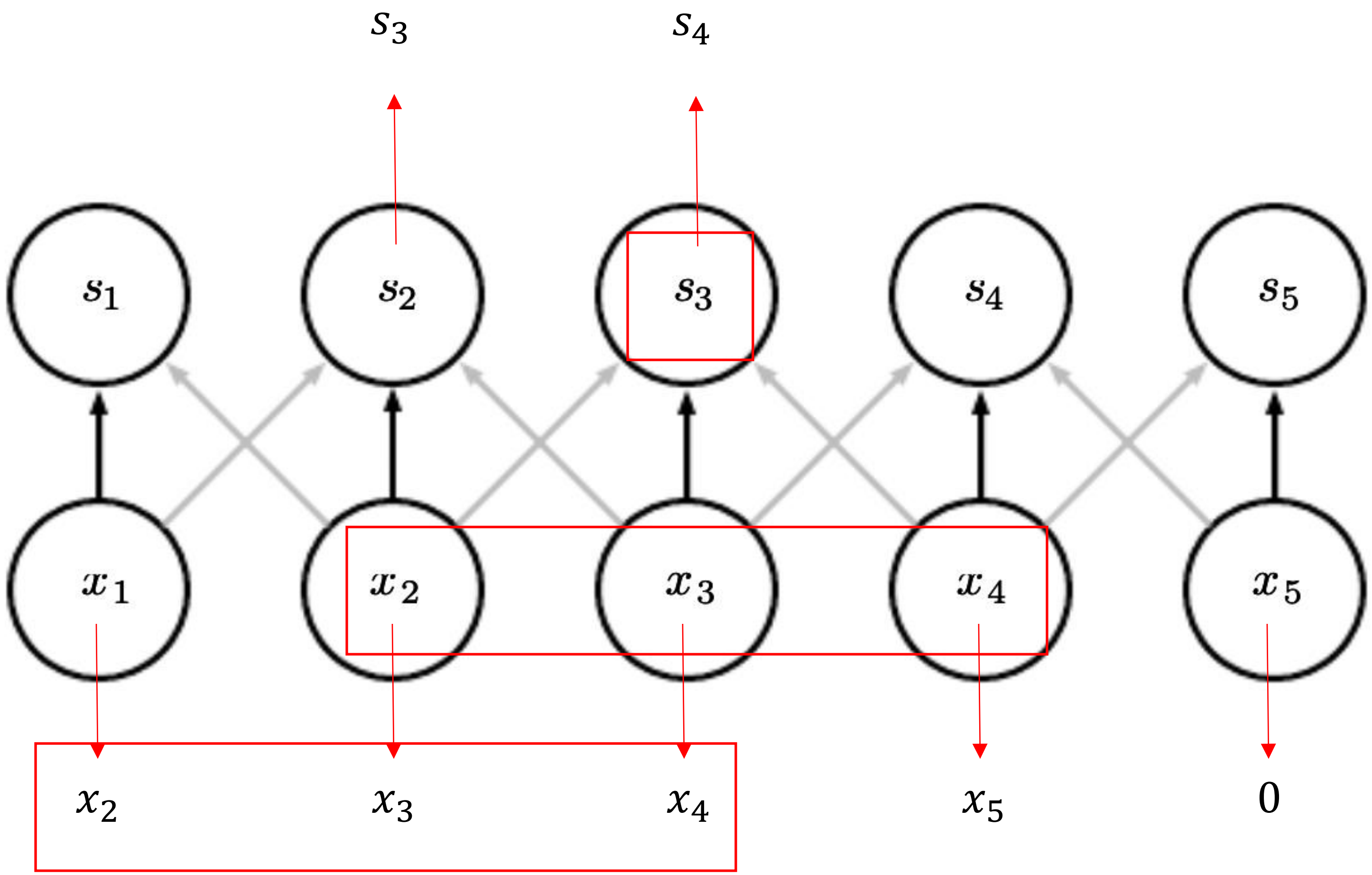
Equivariance



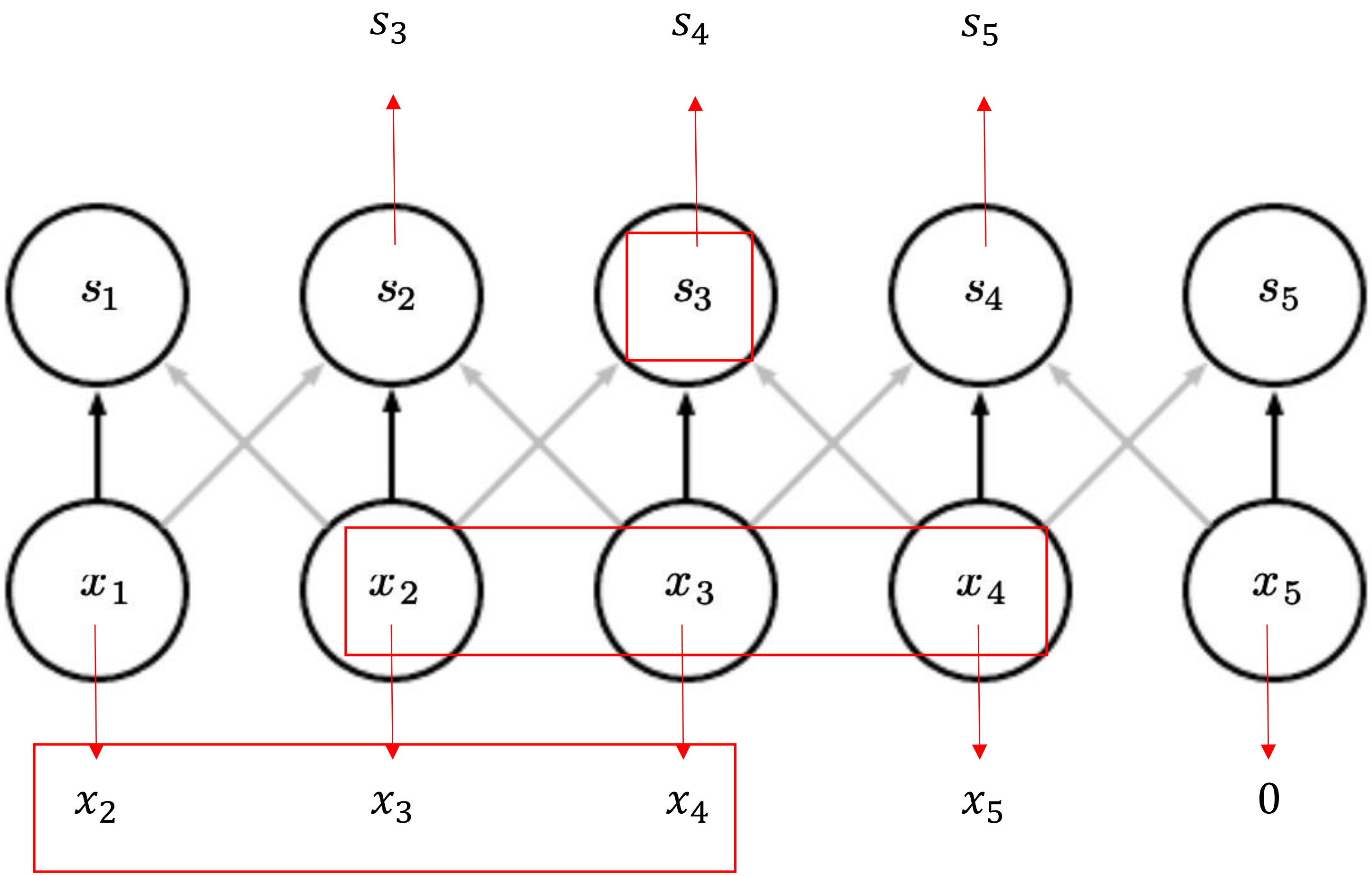
Equivariance



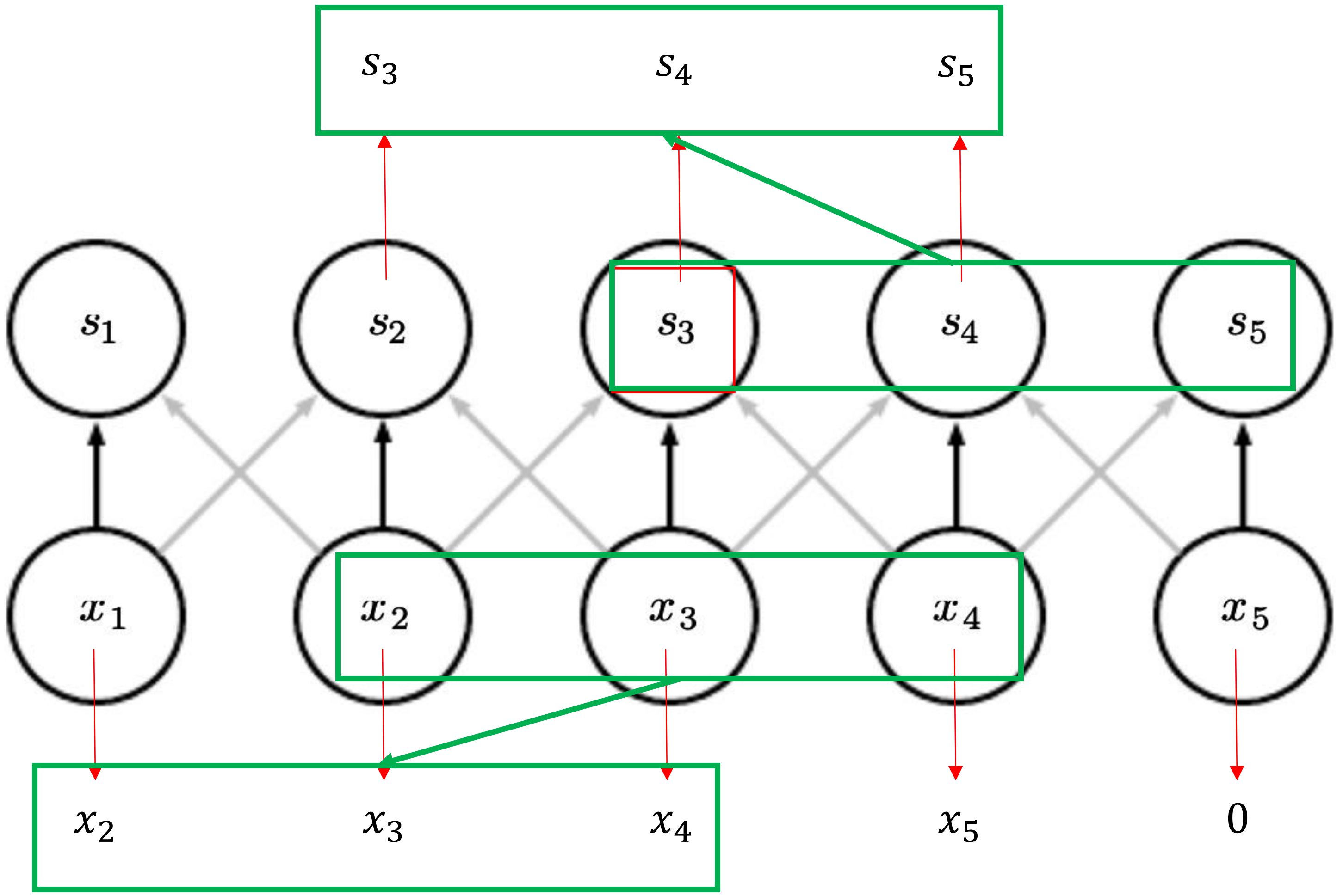
Equivariance



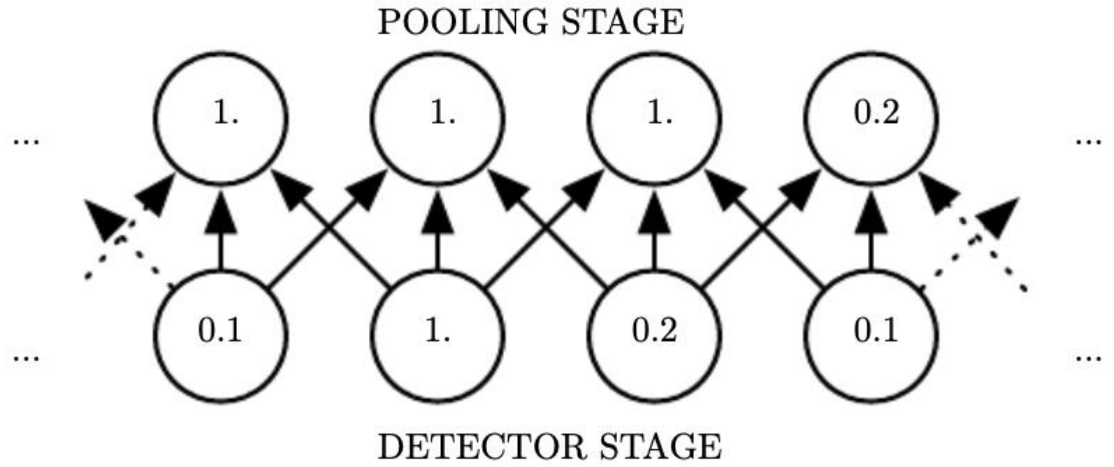
Equivariance

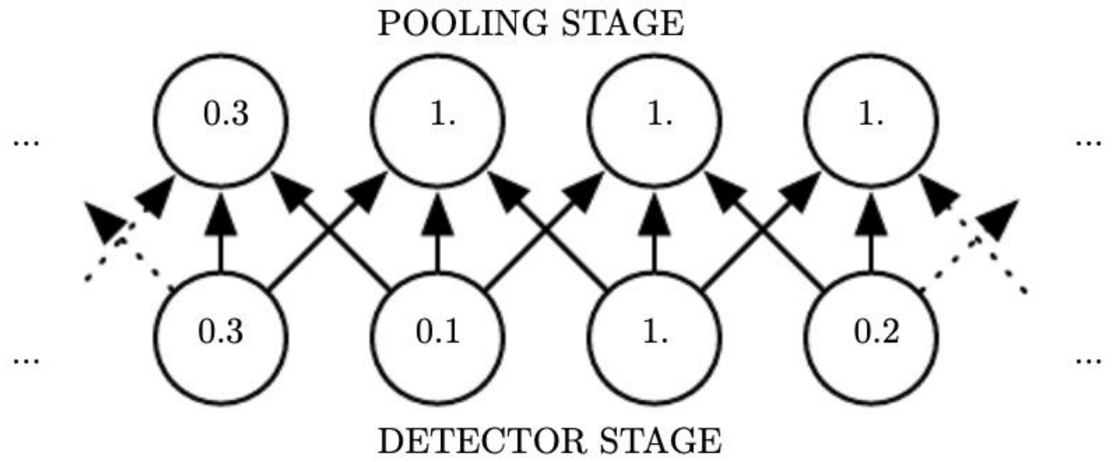


Equivariance

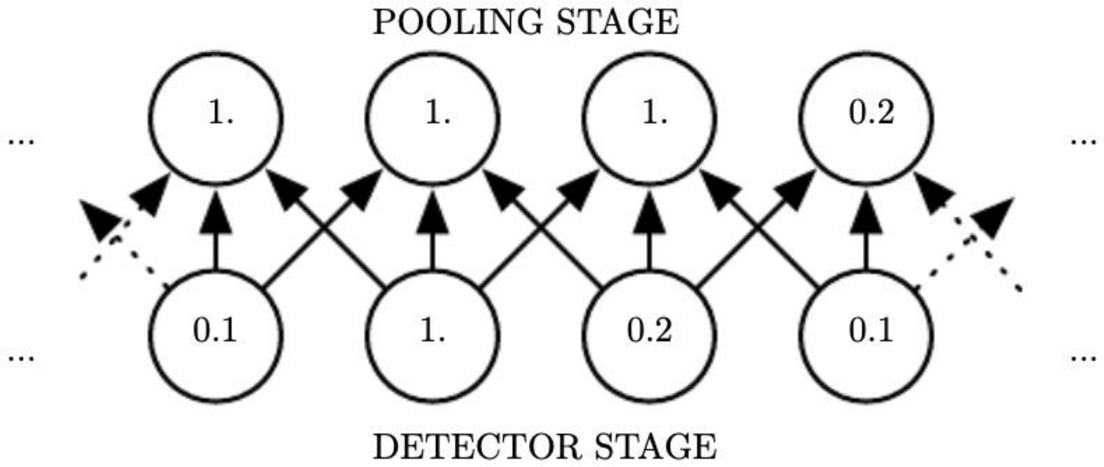


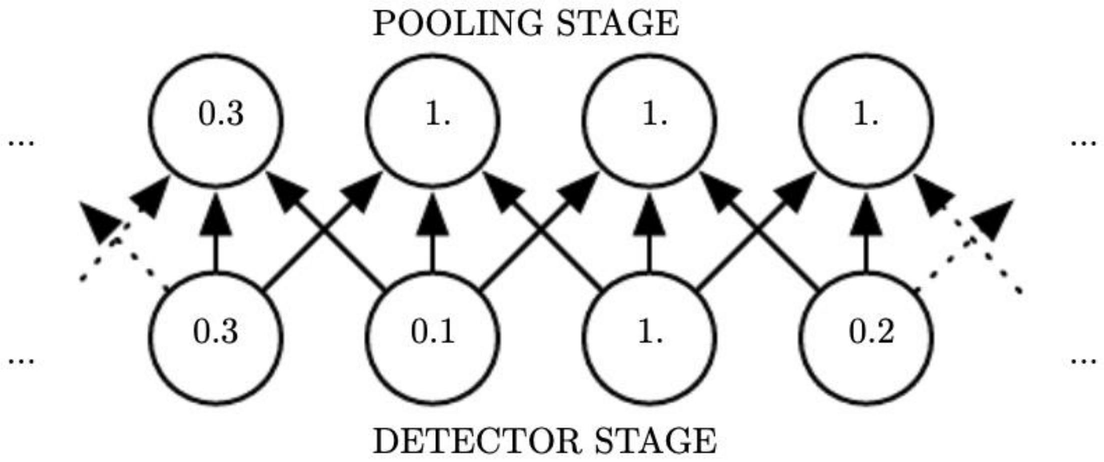
Pooling: invariance to small translation





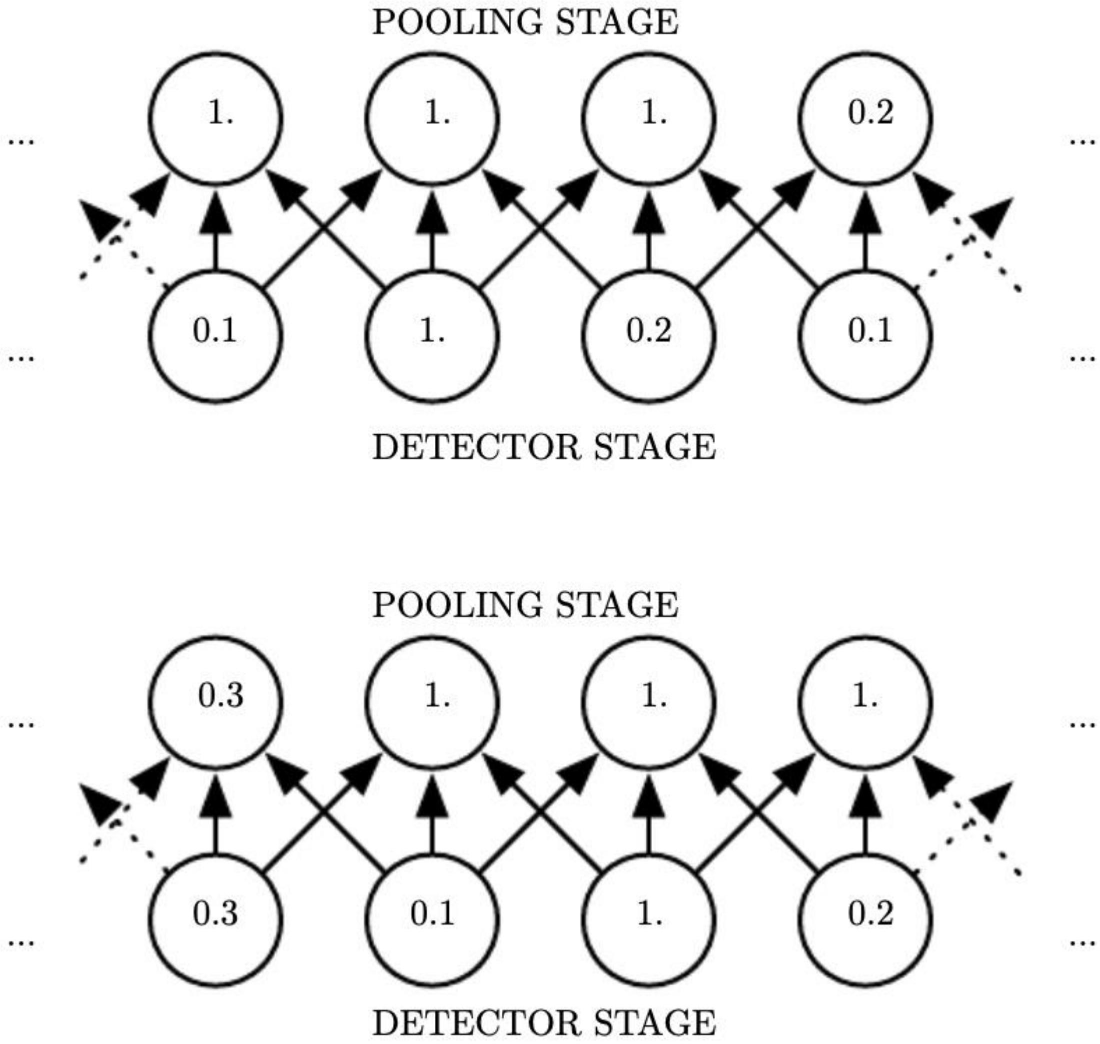
Pooling: invariance to small translation





Q: what is type of pooling?
Max or average pooling?

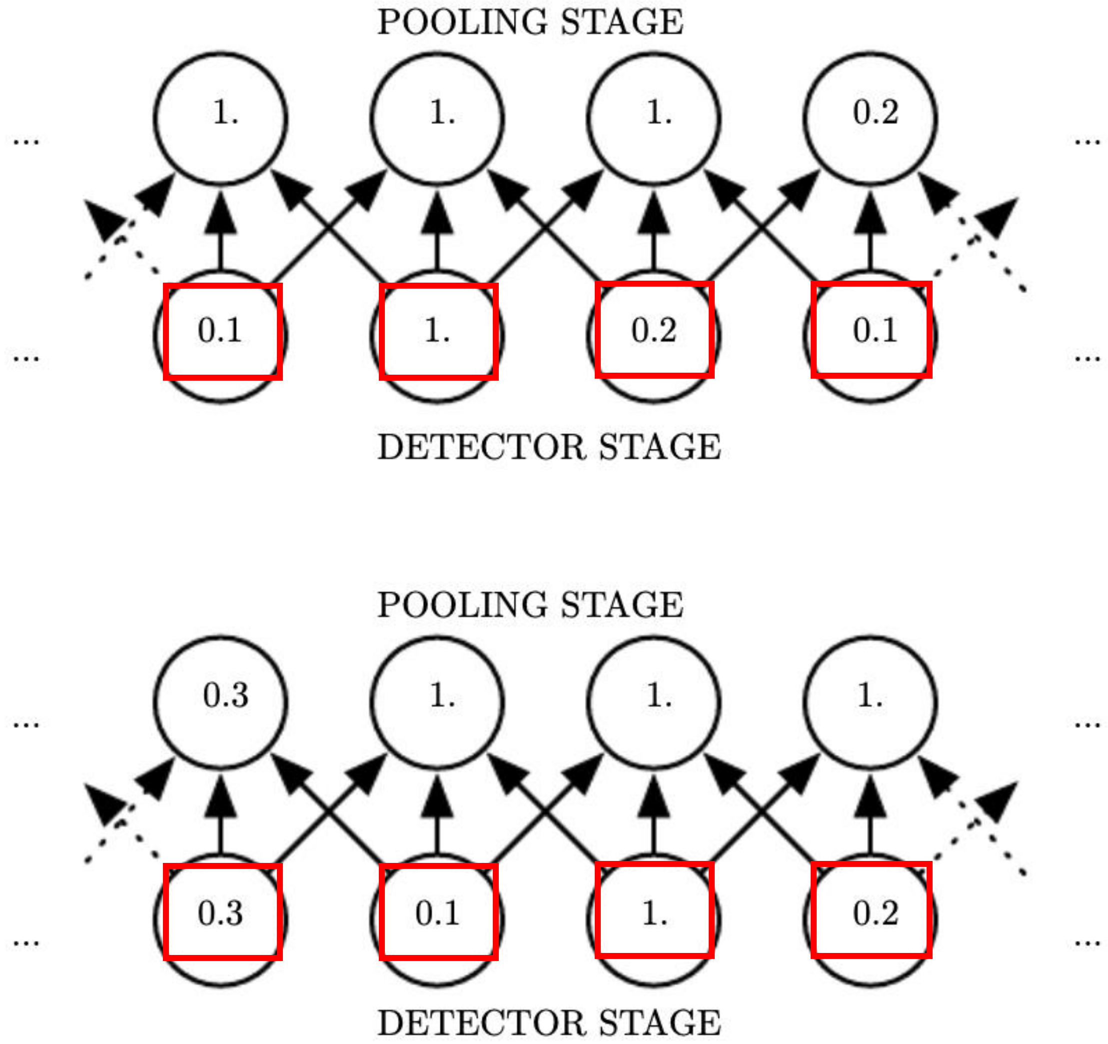
Pooling: invariance to small translation



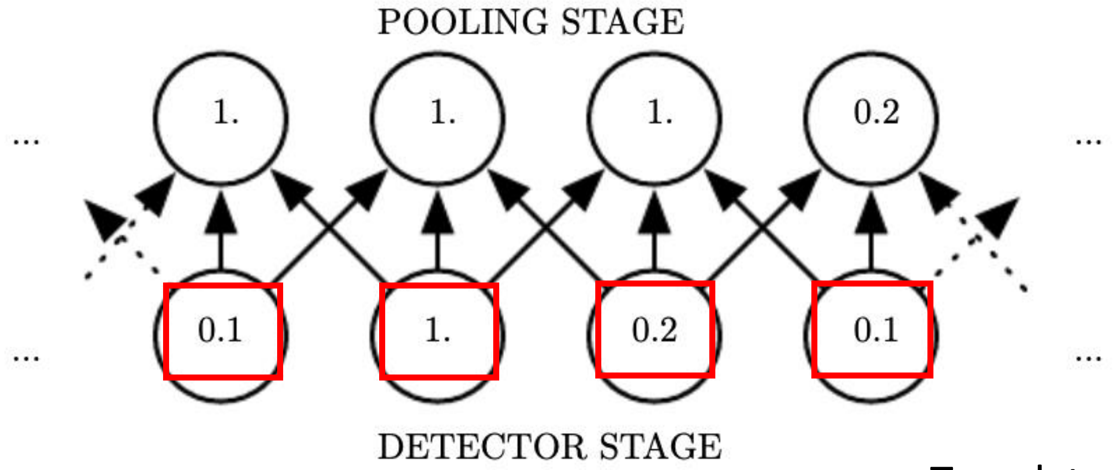
Q: what is type of pooling?
Max or average pooling?

Max pooling

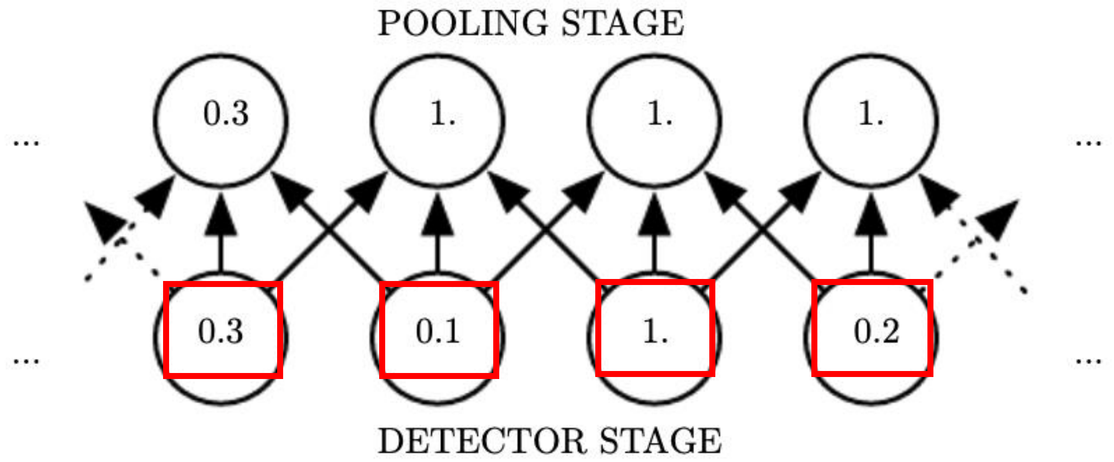
Pooling: invariance to small translation



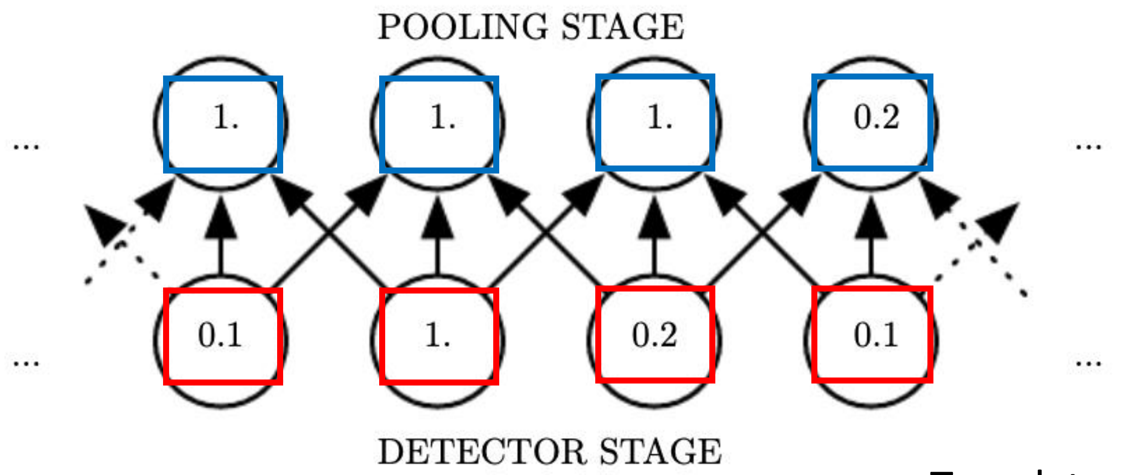
Pooling: invariance to small translation



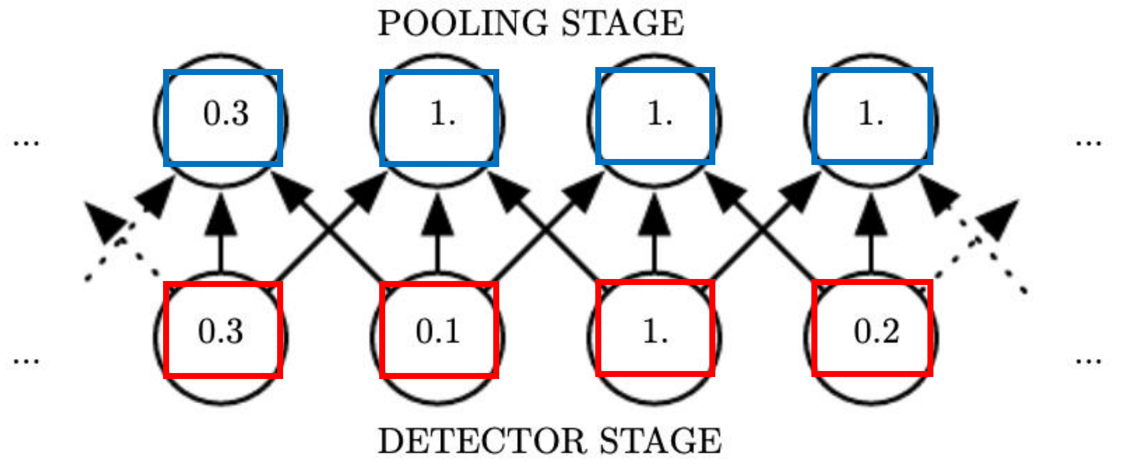
Translate: change the input value a little bit
+ change their positions



Pooling: invariance to small translation



Translate: change the input value a little bit
+ change their positions



Seagull = 1?

Seagull = 1?

Seagull is in the center

Seagull = 1?

Seagull = 1?

Seagulls are present, but not in the center

Seagull = 1?

Seagulls are present, but not in the center

Seagull = 1?

Seagulls are present, but not in the center

CNNs can tell:
whether seagulls are present
Not tell:
their positions in the image

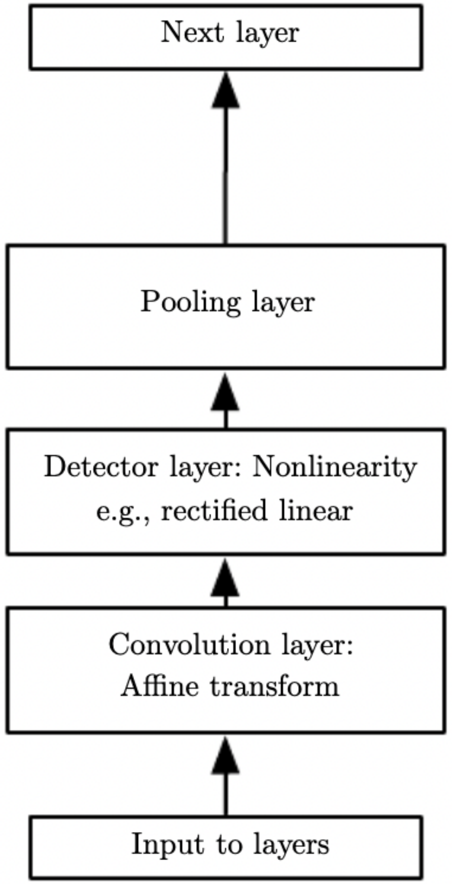
Seagull = 1?

Seagulls are present, but not in the center

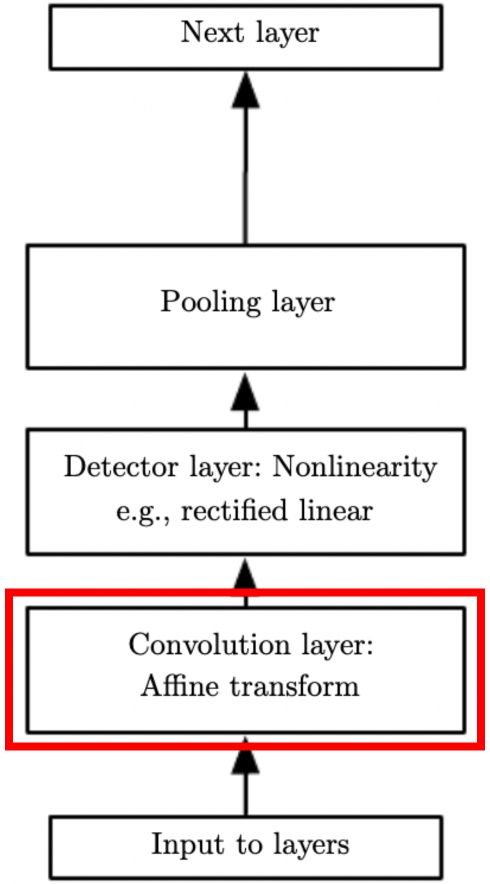
CNNs can tell:
whether seagulls are present
Not tell:
their positions in the image

Invariant to small translations

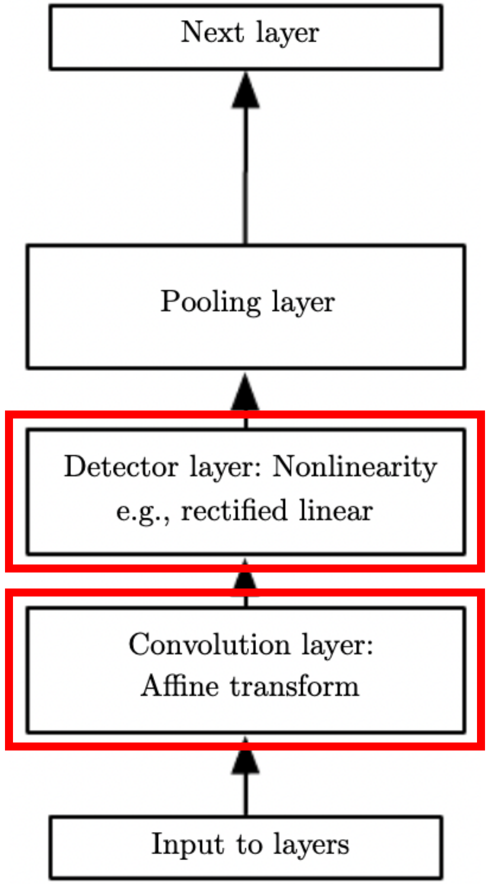
A typical convolutional layer



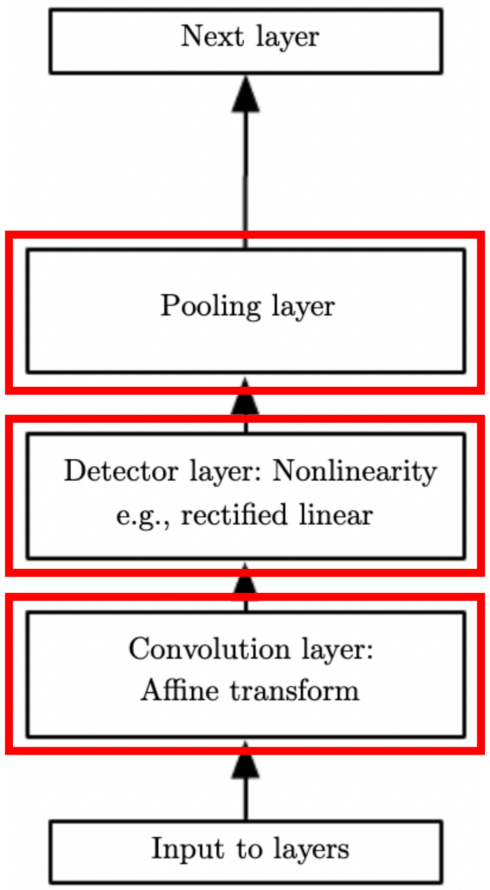
A typical convolutional layer



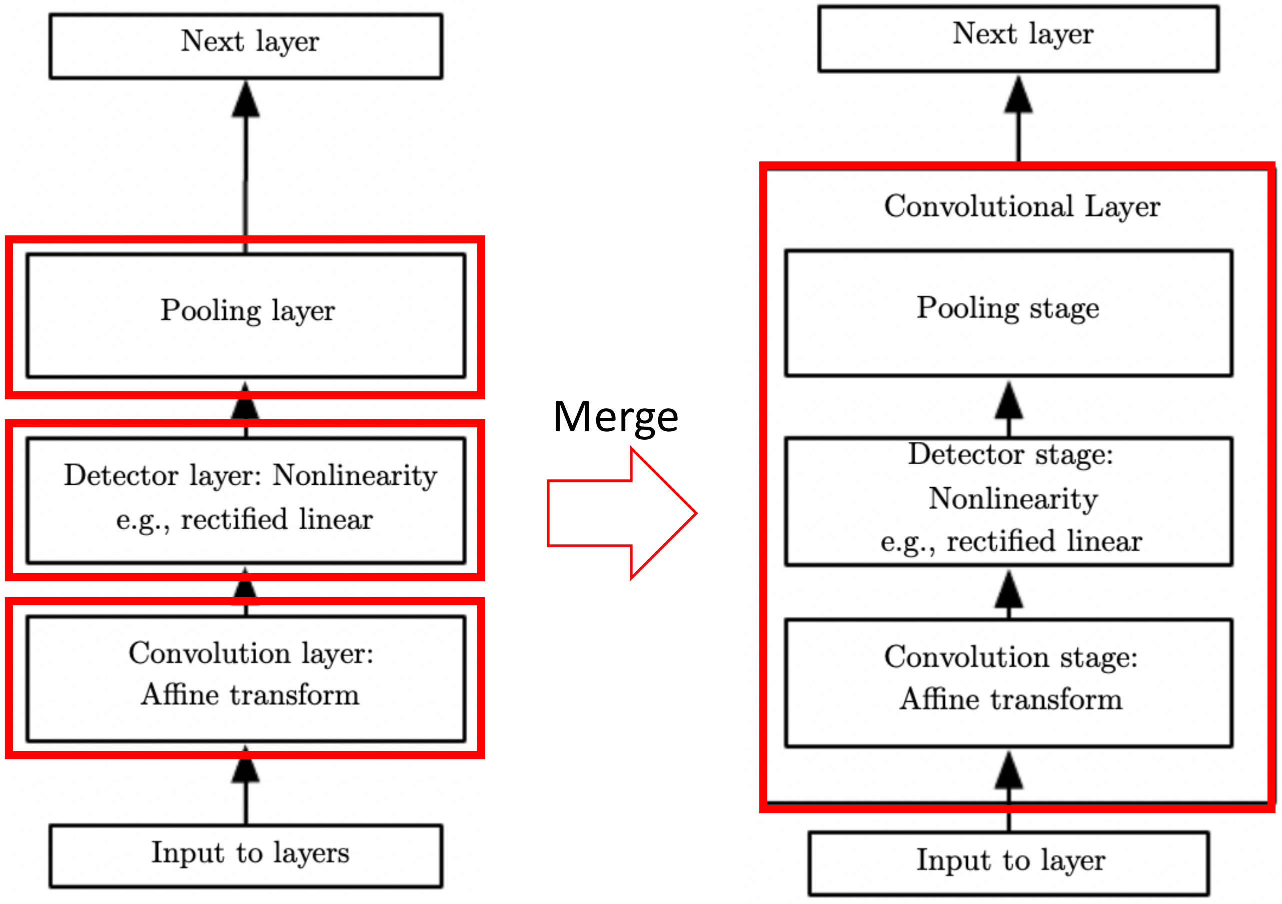
A typical convolutional layer

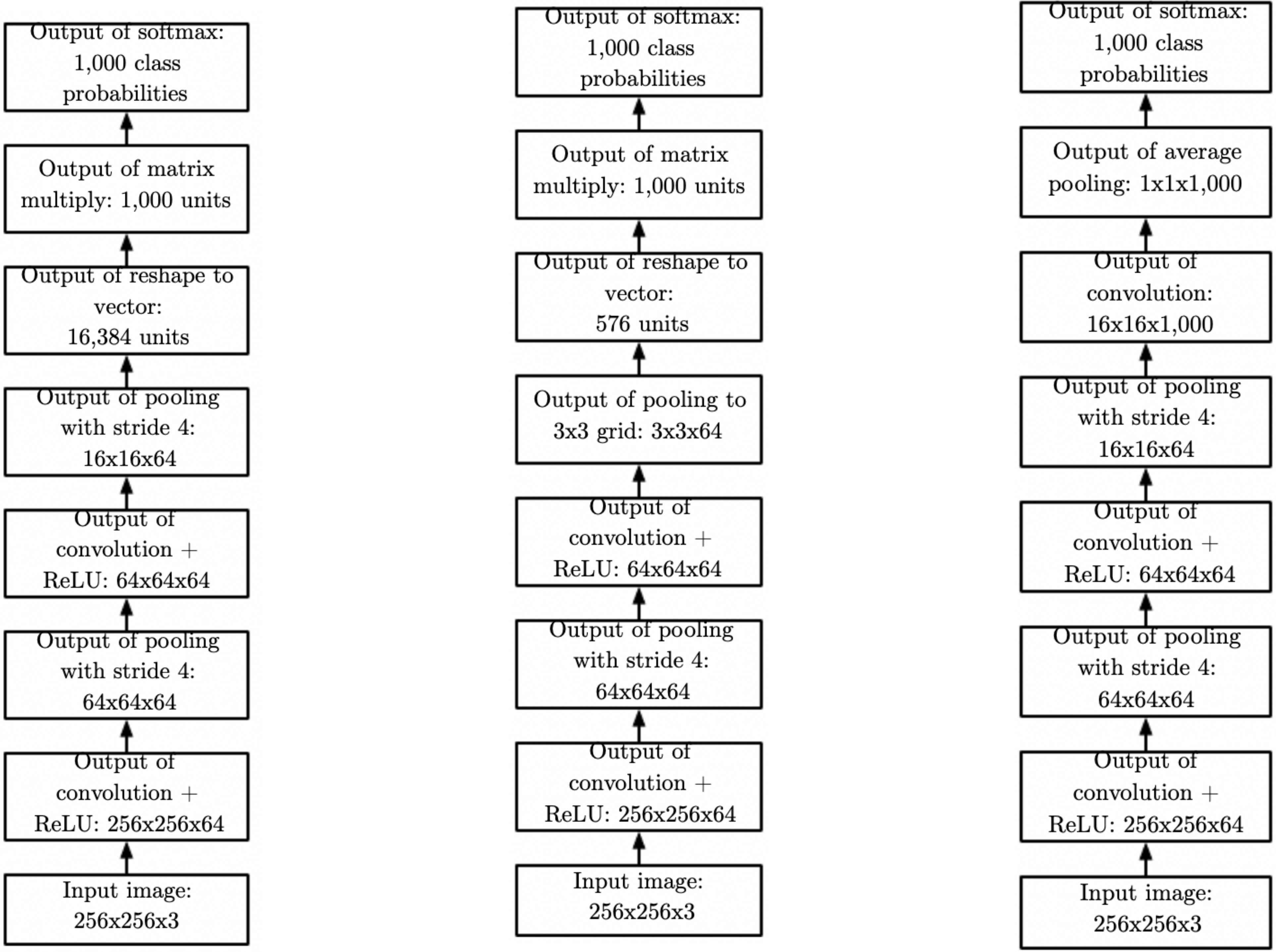


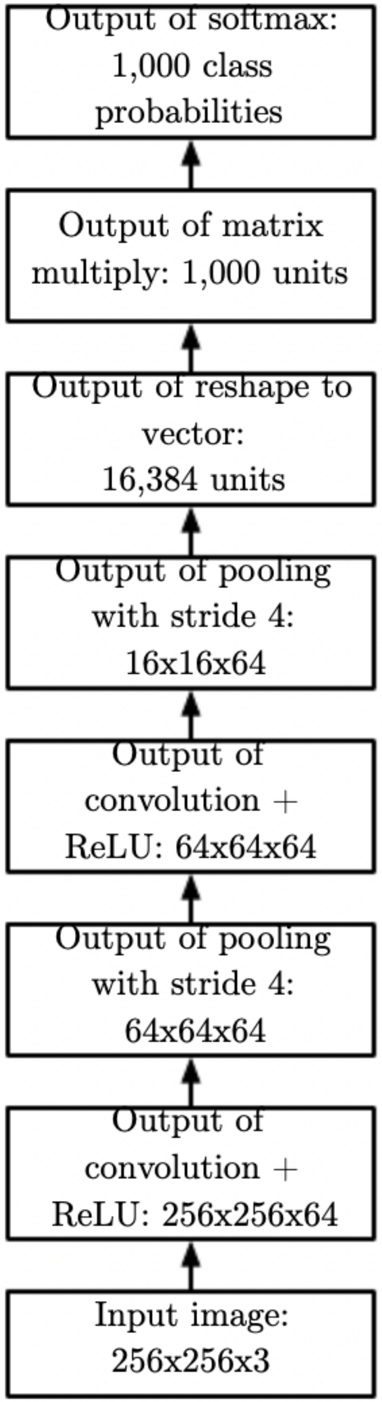
A typical convolutional layer

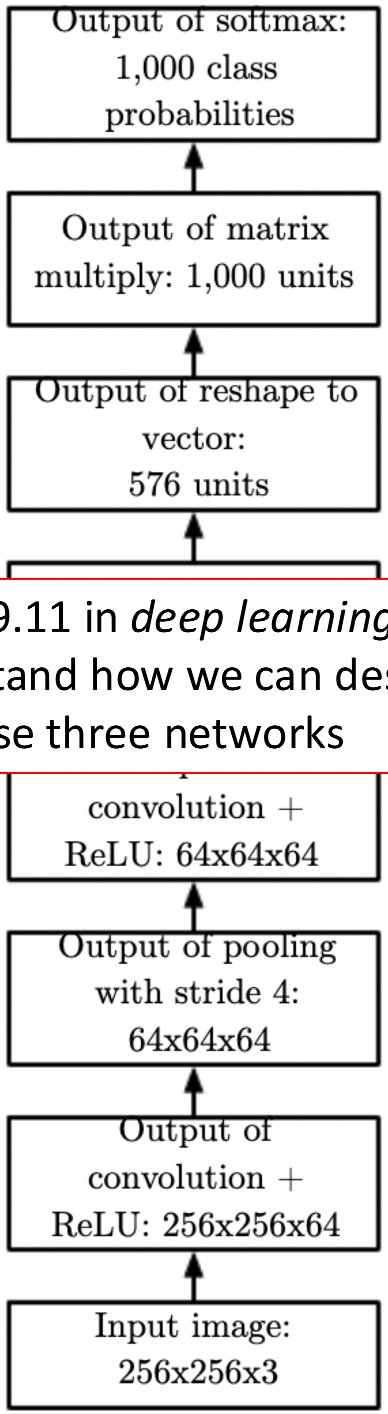


A typical convolutional layer

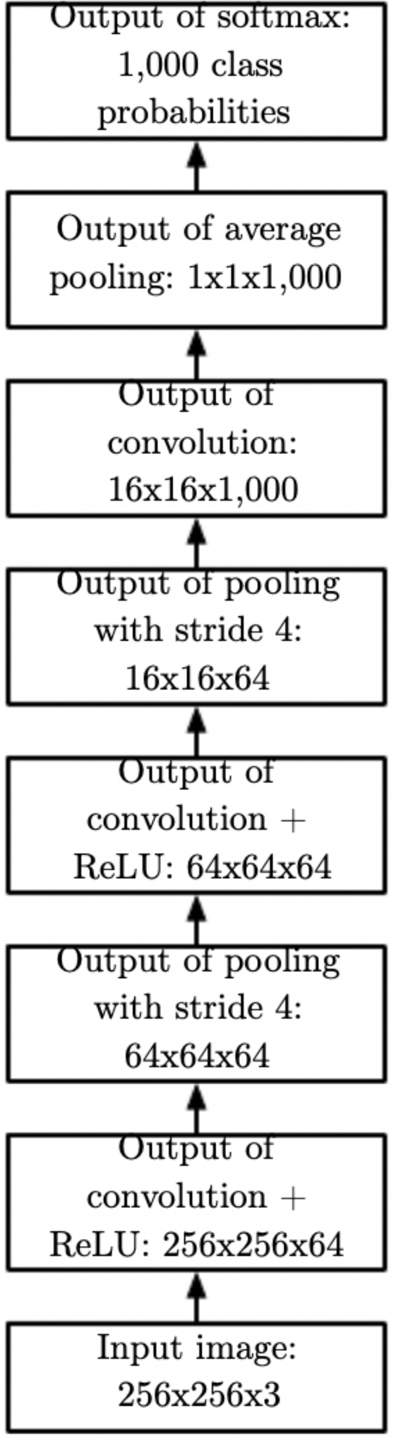



Output of softmax: 1,000 class probabilities
Output of matrix multiply: 1,000 units
Output of reshape to vector: 16,384 units
Output of pooling with stride 4: 16x16x64
Output of convolution + ReLU: 64x64x64
Output of pooling with stride 4: 64x64x64
Output of convolution + ReLU: 256x256x64
Input image: 256x256x3
Output of softmax: 1,000 class probabilities
Output of matrix multiply: 1,000 units
Output of reshape to vector: 576 units
Output of pooling to 3x3 grid: 3x3x64
Output of convolution + ReLU: 64x64x64
Output of pooling with stride 4: 64x64x64
Output of convolution + ReLU: 256x256x64
Input image: 256x256x3
Output of softmax: 1,000 class probabilities
Output of average pooling: 1x1x1,000
Output of convolution: 16x16x1,000
Output of pooling with stride 4: 16x16x64
Output of convolution + ReLU: 64x64x64
Output of pooling with stride 4: 64x64x64
Output of convolution + ReLU: 256x256x64
Input image: 256x256x3

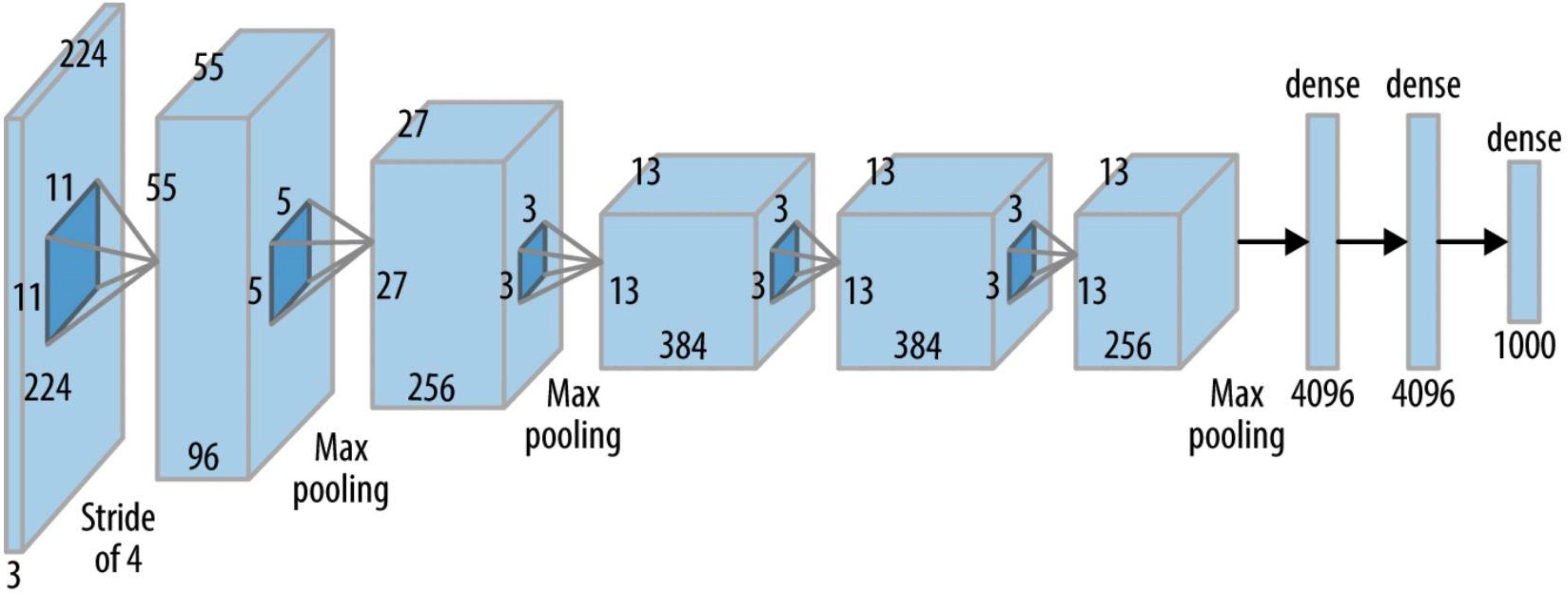




Read: Figure 9.11 in *deep learning* book
Try to understand how we can design architecture and build these three networks

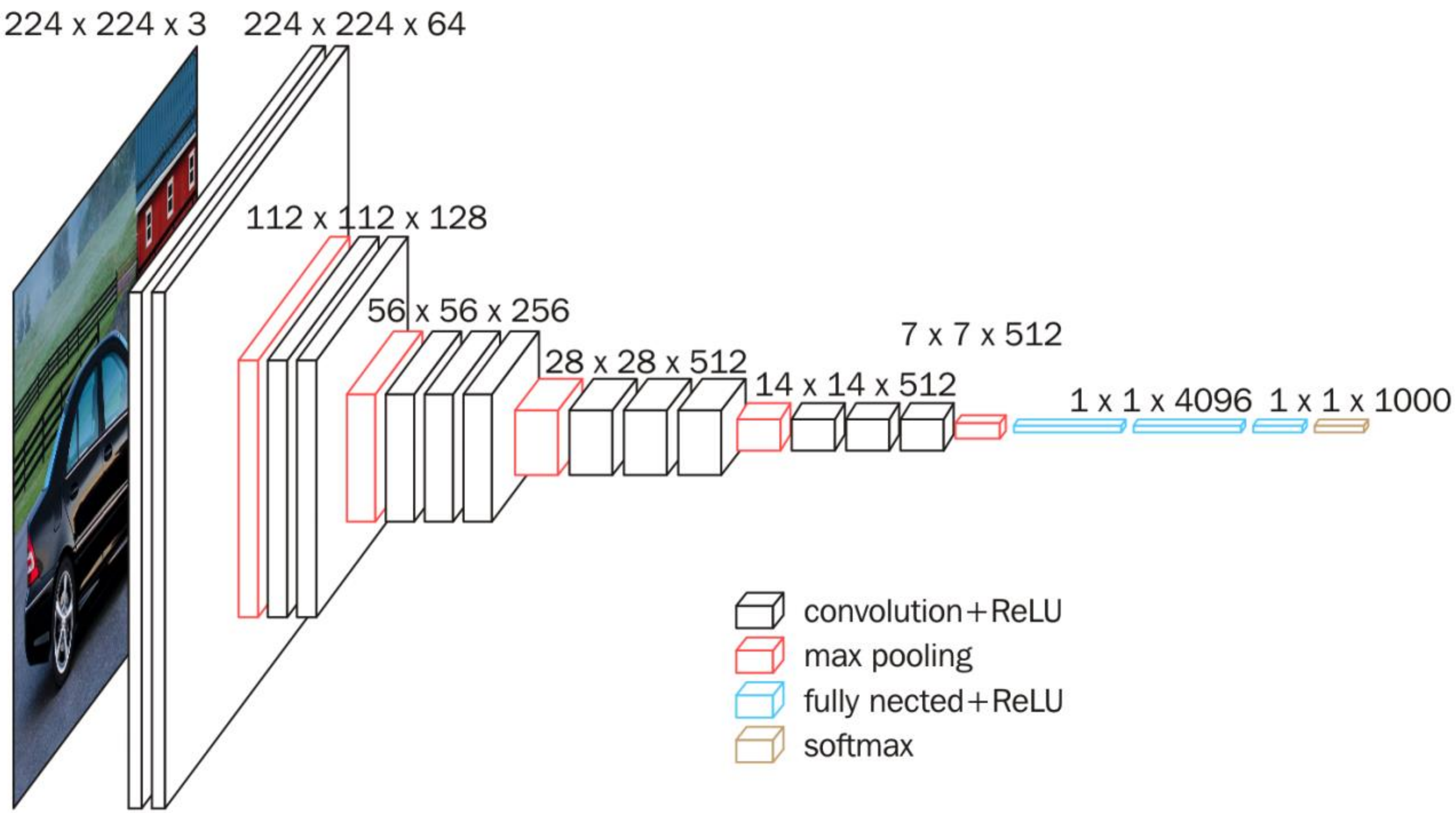


AlexNet

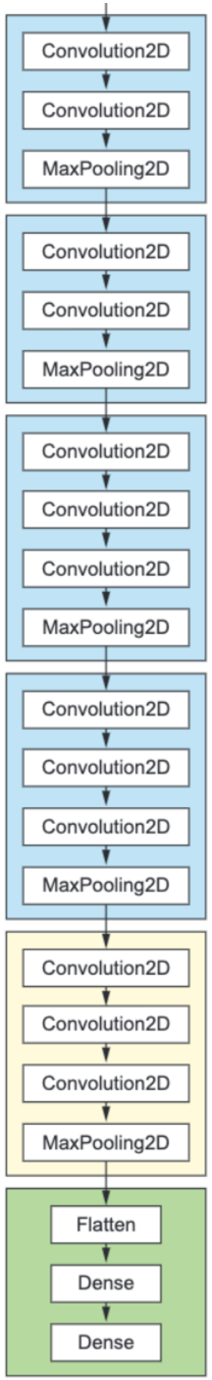


224
224
11
11
3
Stride
of 4
55
55
96
5
5
Max
pooling
27
27
256
3
3
Max
pooling
13
13
384
3
3
13
13
384
3
3
13
13
256
Max
pooling
dense
dense
dense
4096
4096
1000

VGG-16



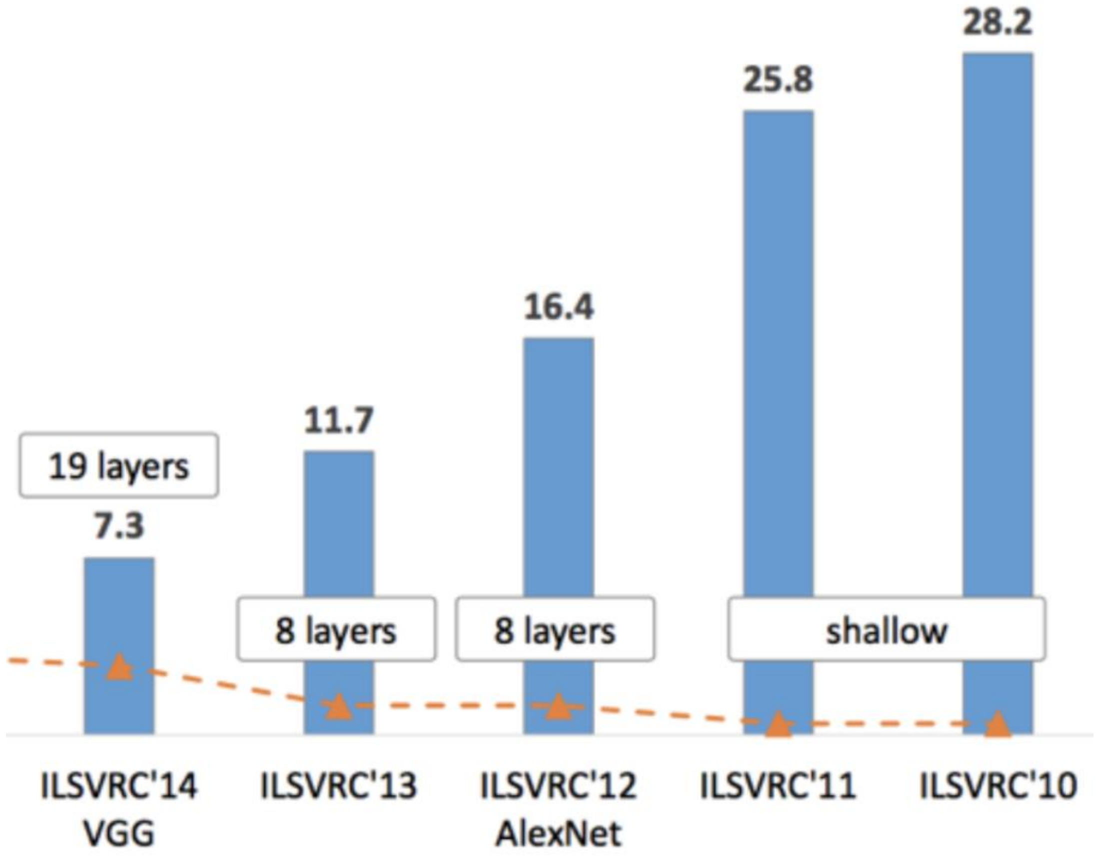
224 x 224 x 3
224 x 224 x 64
112 x 112 x 128
56 x 56 x 256
28 x 28 x 512
14 x 14 x 512
7 x 7 x 512
1 x 1 x 4096
1 x 1 x 1000
convolution+ReLU
max pooling
fully nected+ReLU
softmax



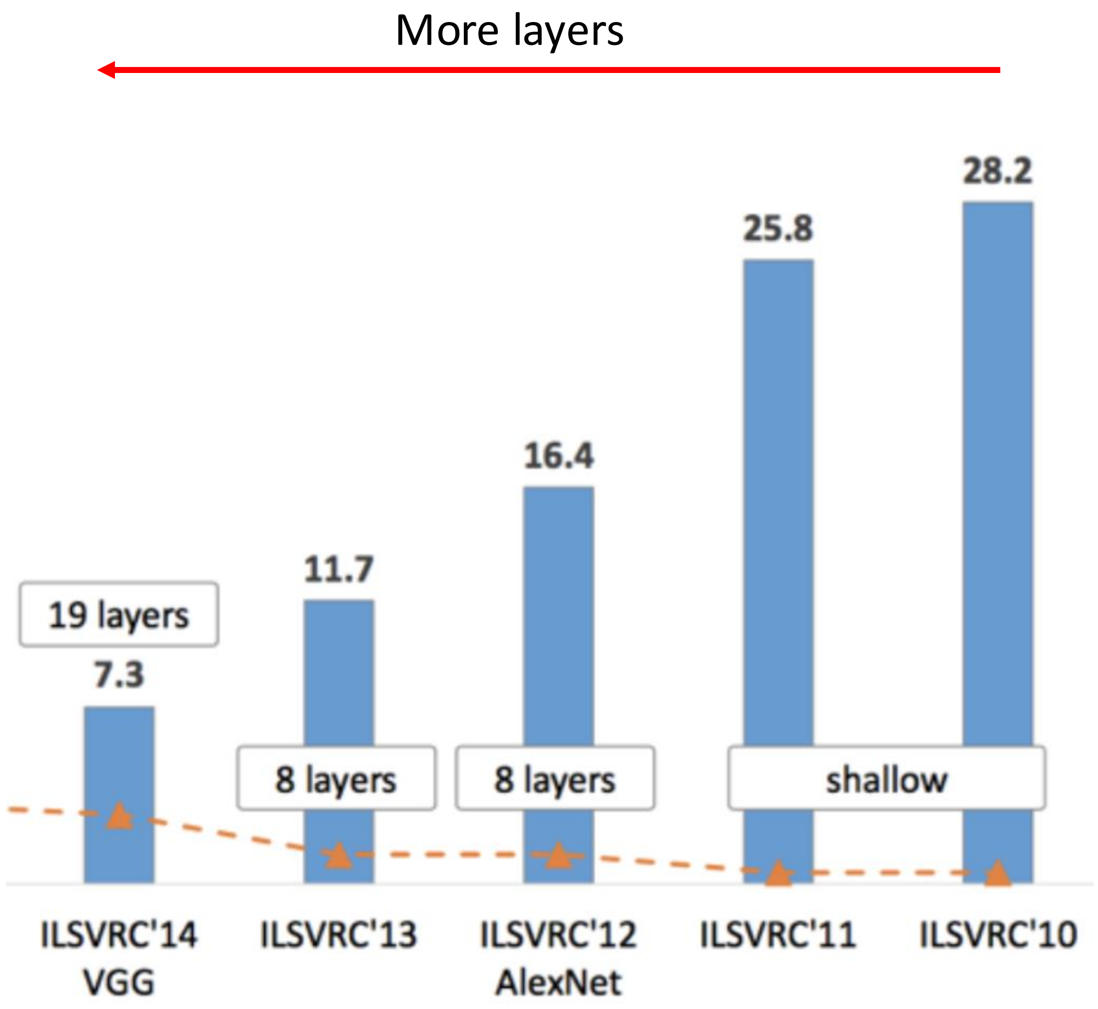
Convolution2D
Convolution2D
MaxPooling2D
Convolution2D
Convolution2D
MaxPooling2D
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Convolution2D
Convolution2D
Convolution2D
MaxPooling2D
Flatten
Dense
Dense

[VGG]

ImageNet competition

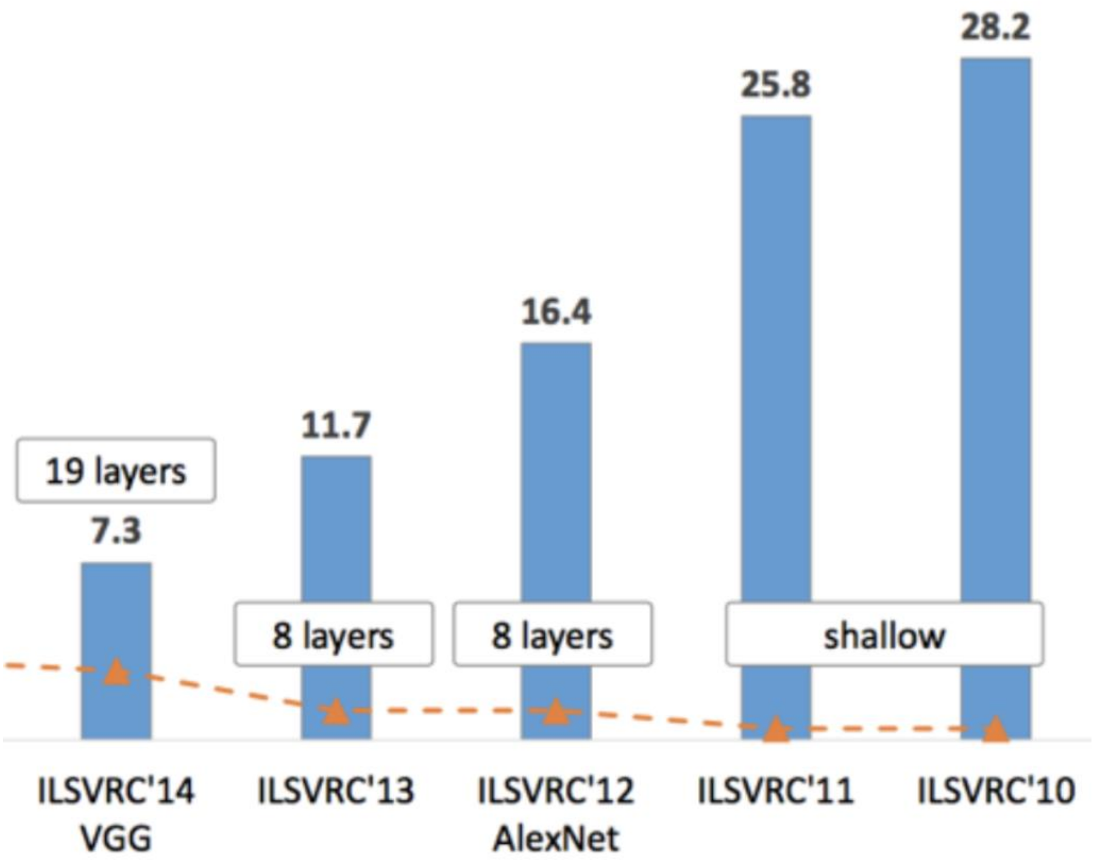


ImageNet competition



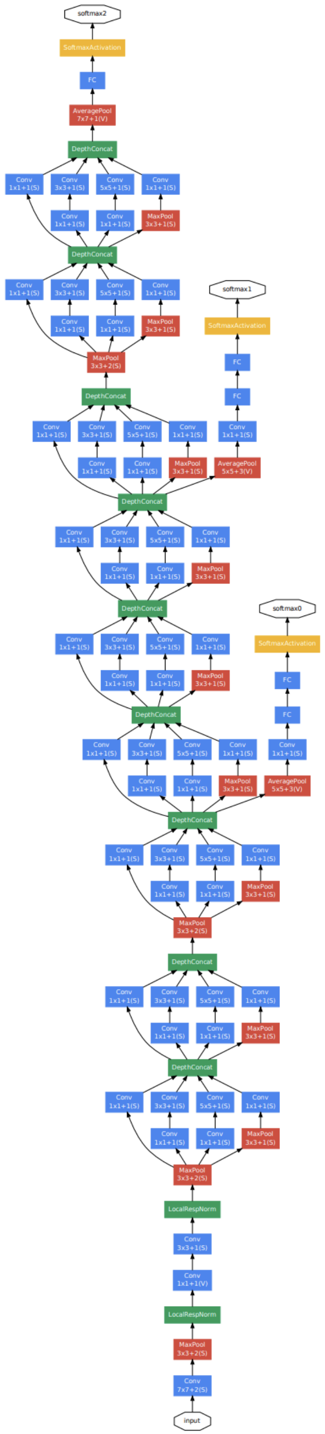
ImageNet competition

More layers

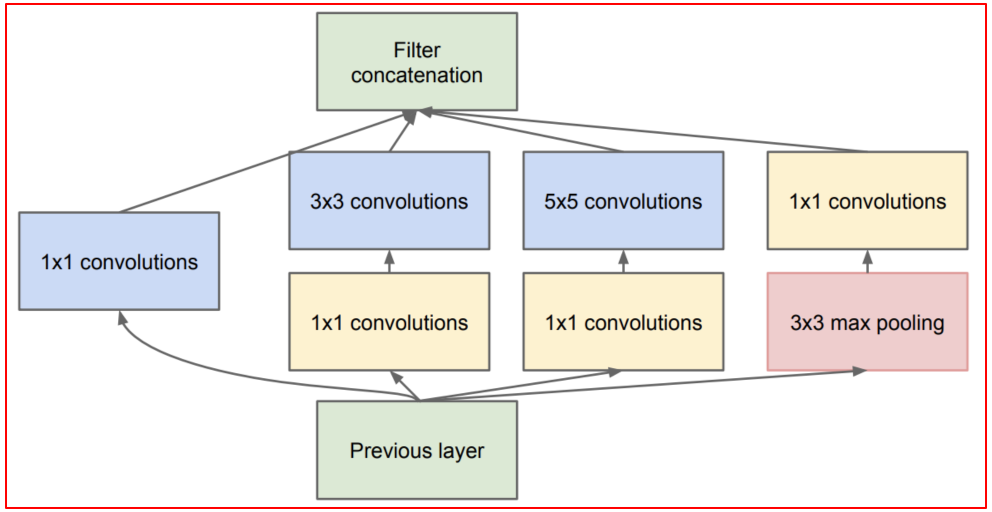


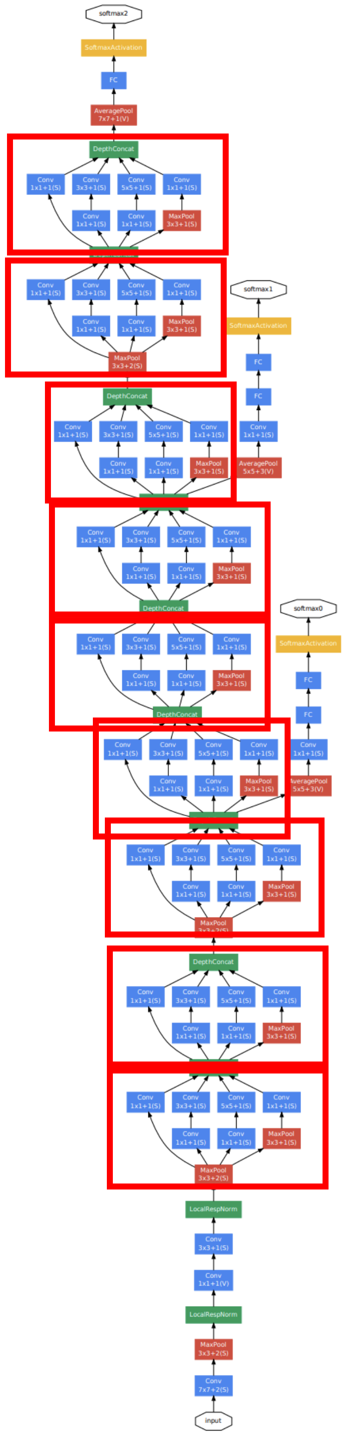
Better performance

Inception (GoogLeNet)

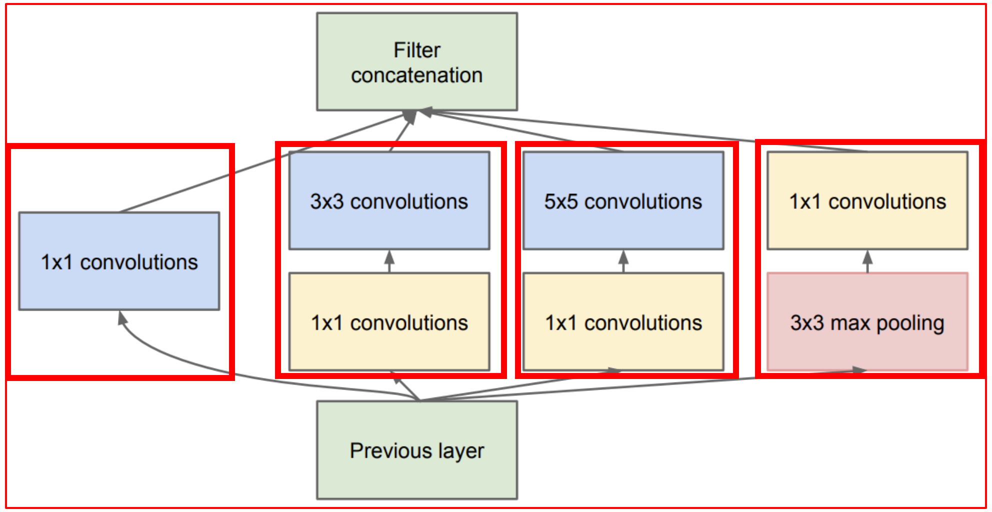


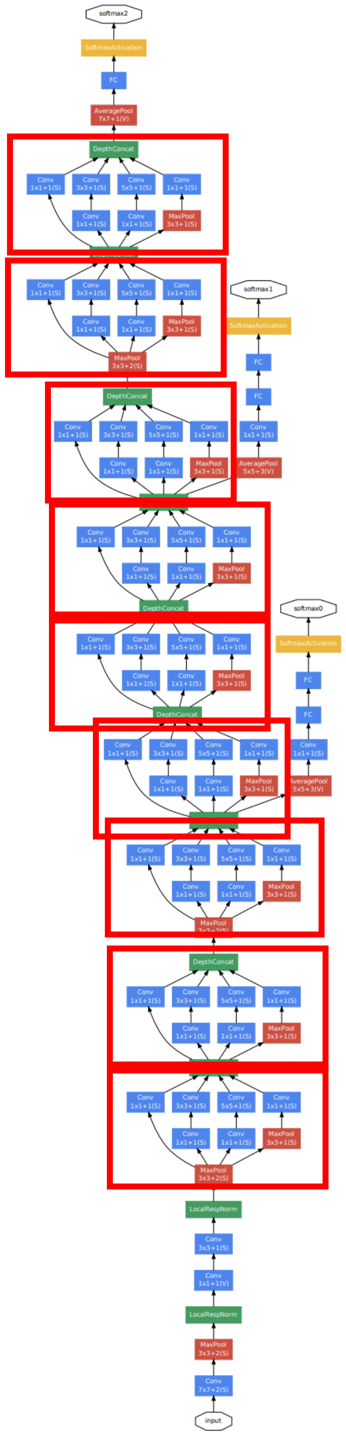
Inception (GoogLeNet)



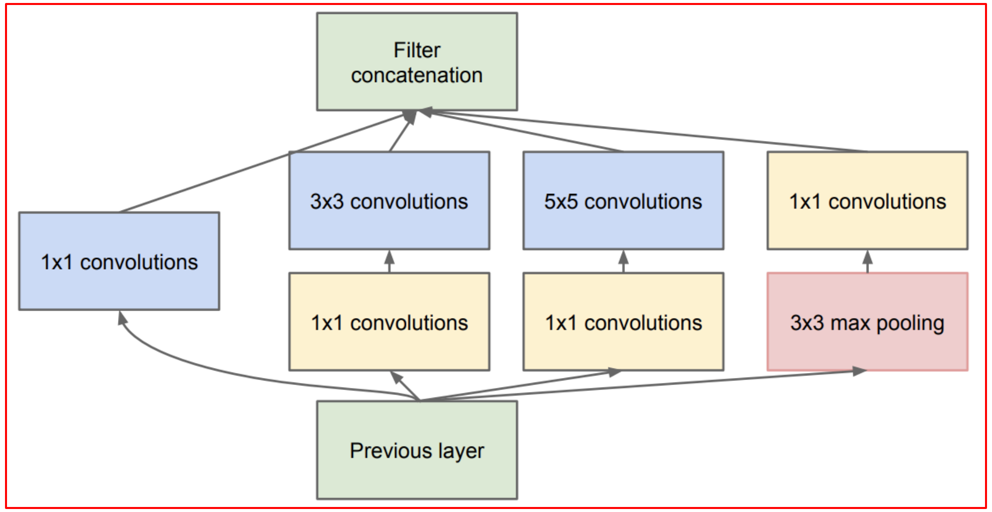


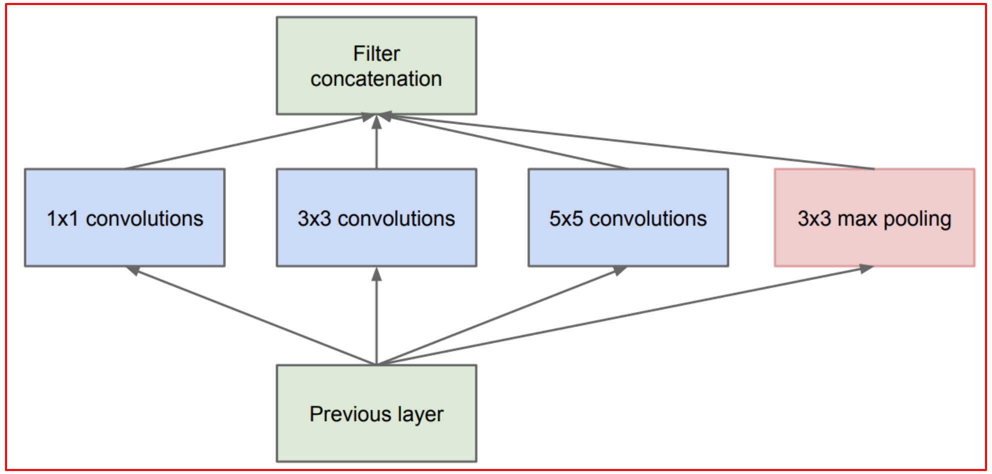
Inception (GoogLeNet)

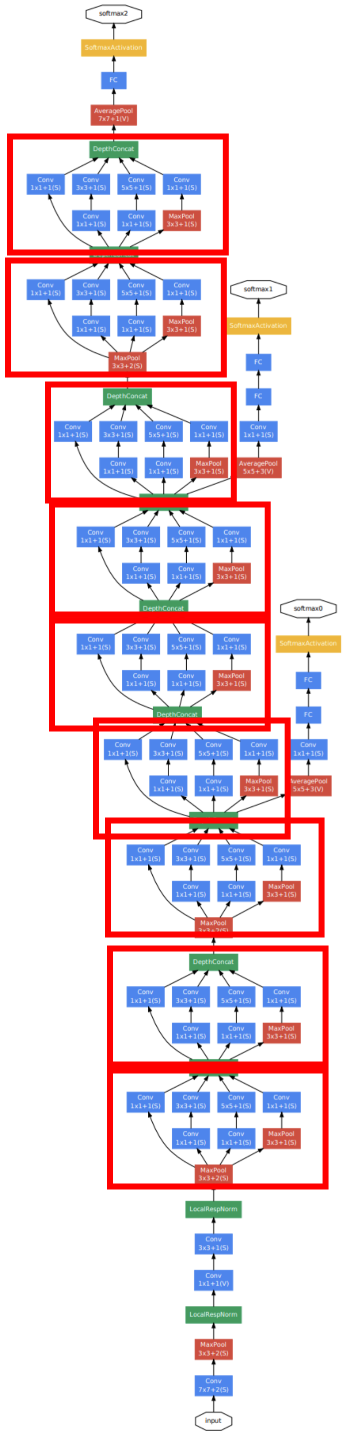




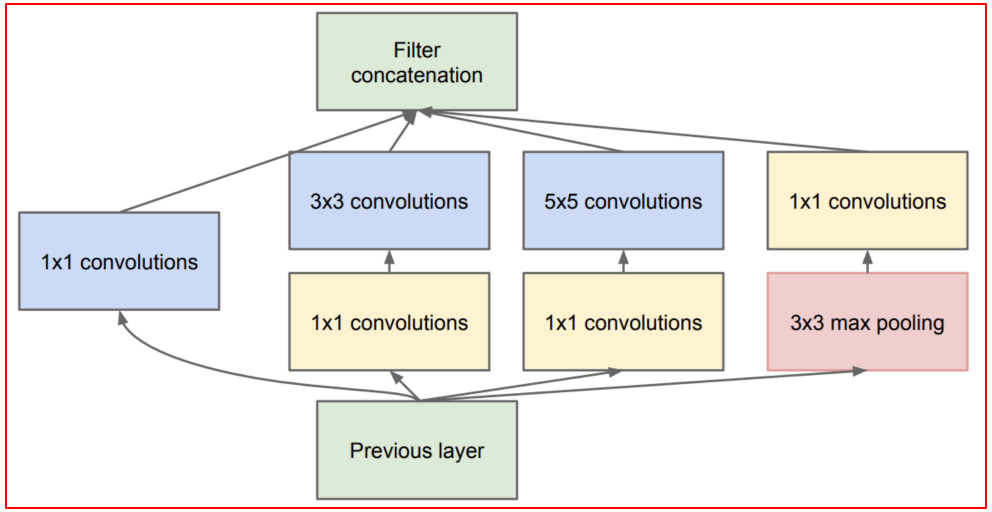
Inception (GoogLeNet)



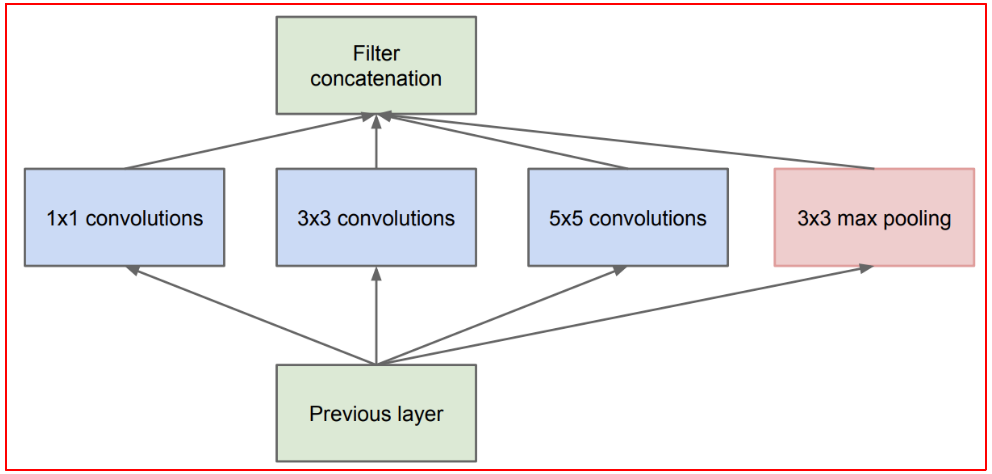


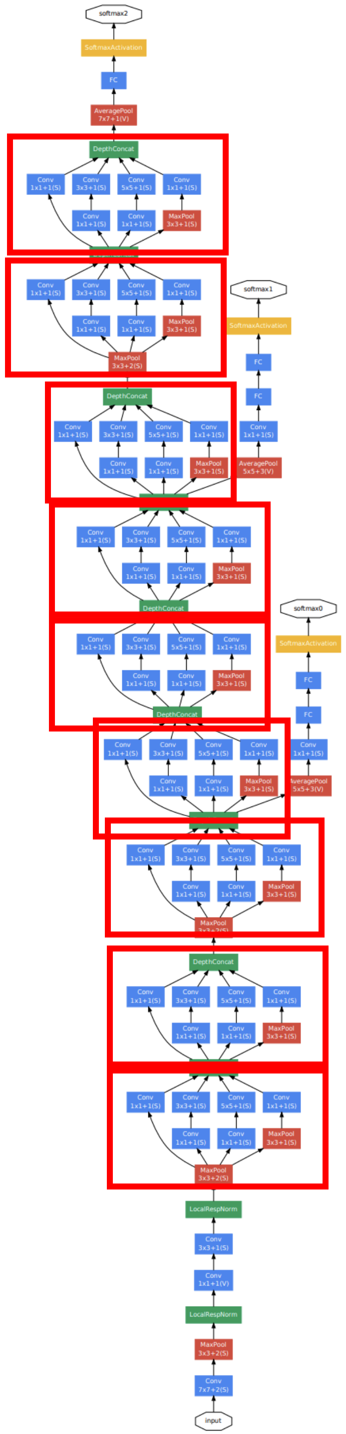


Inception (GoogLeNet)

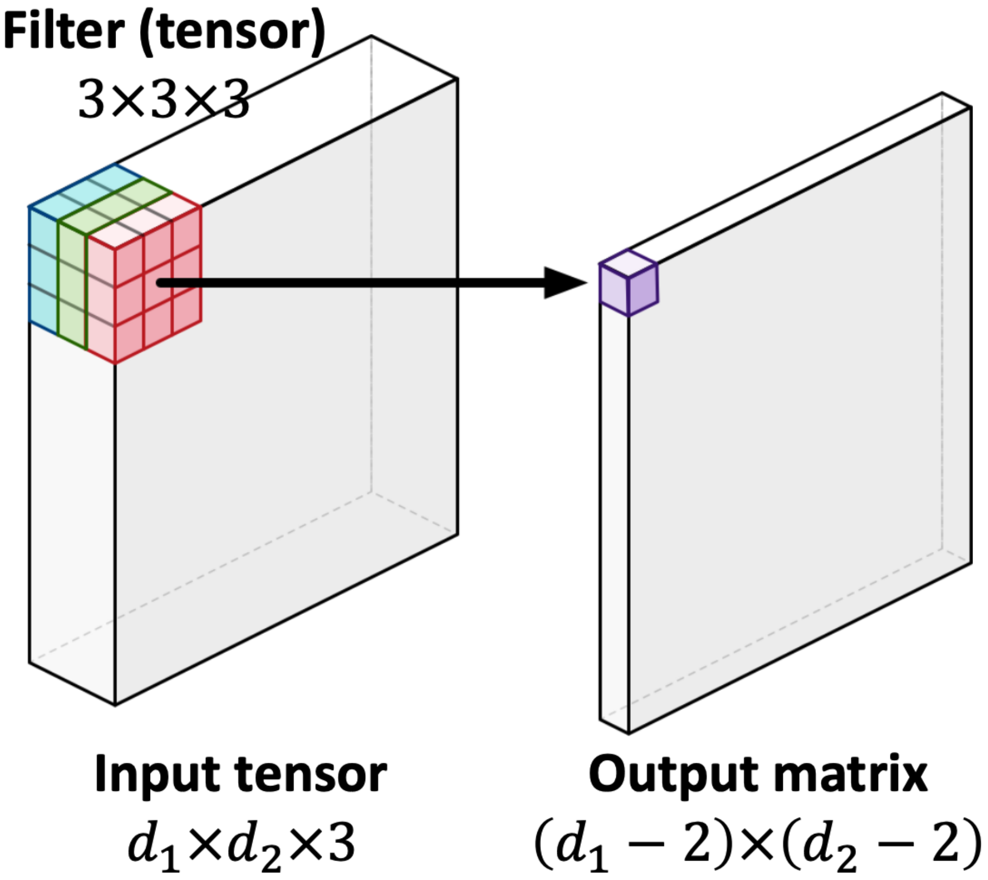


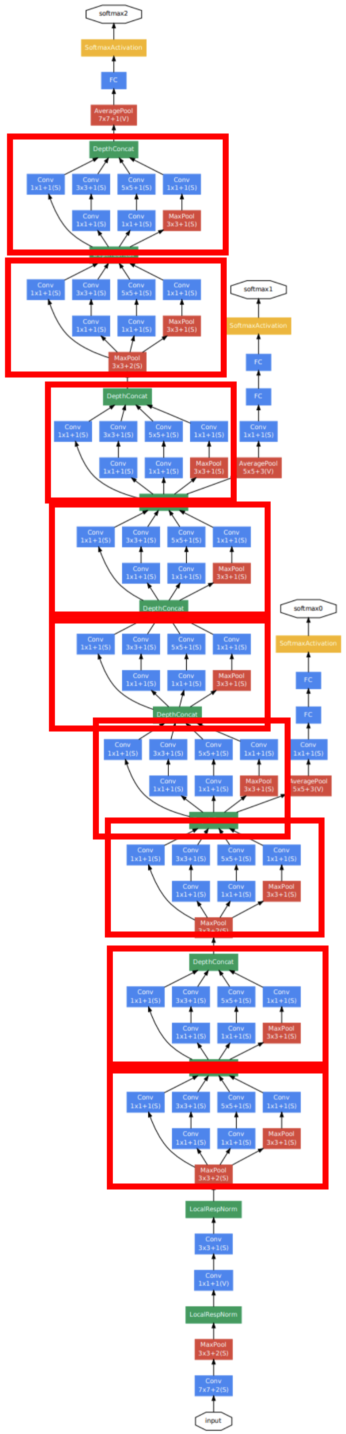
Q: difference between those two variants?



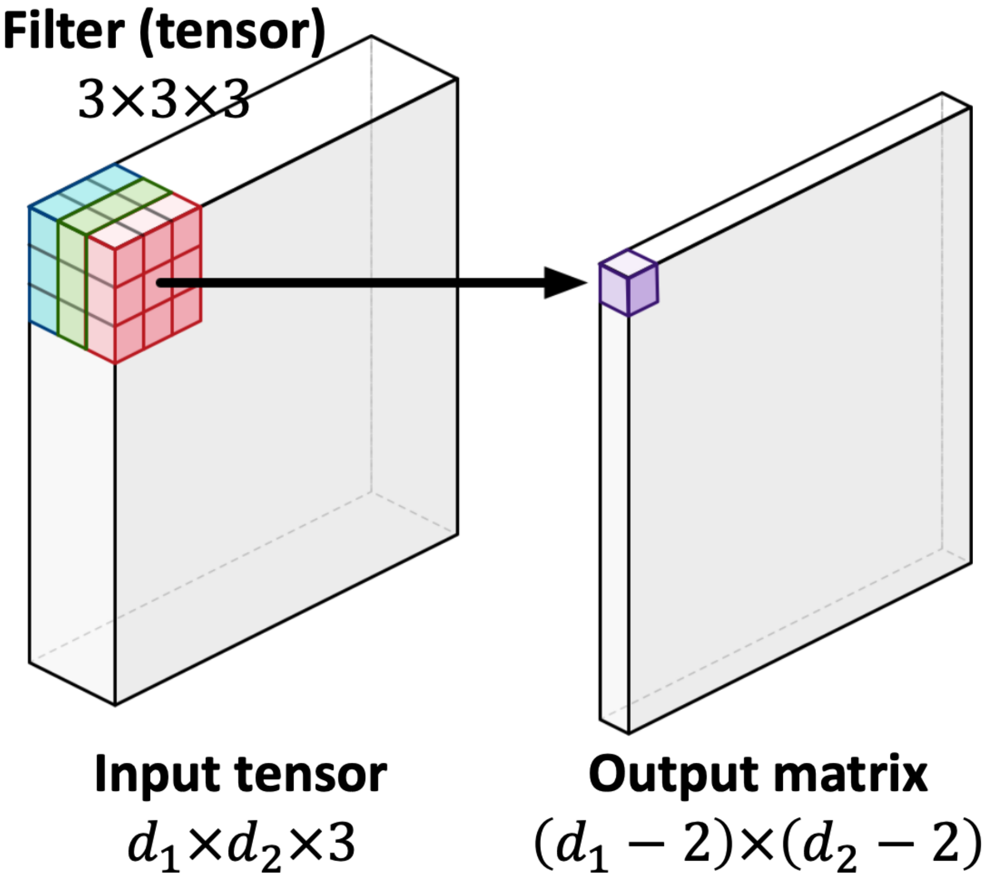


Inception (GoogLeNet)





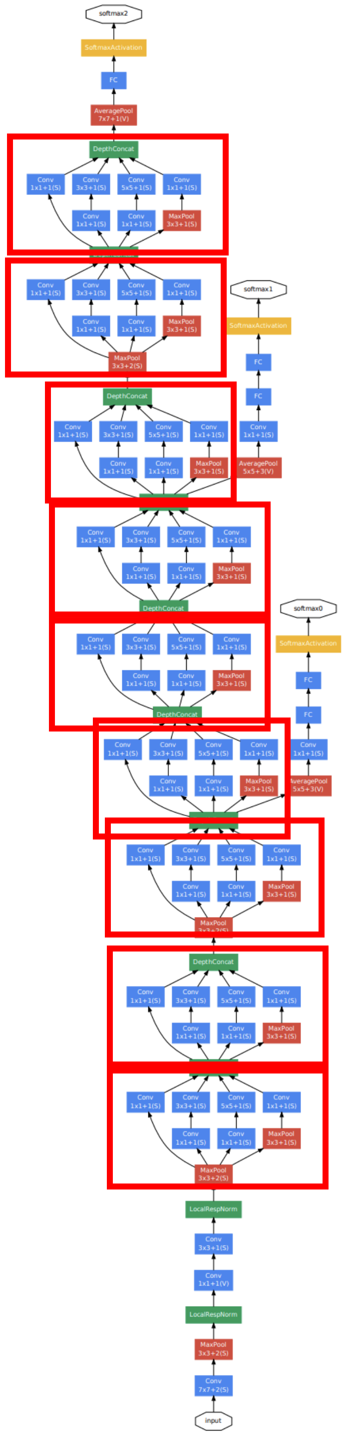
Inception (GoogLeNet)



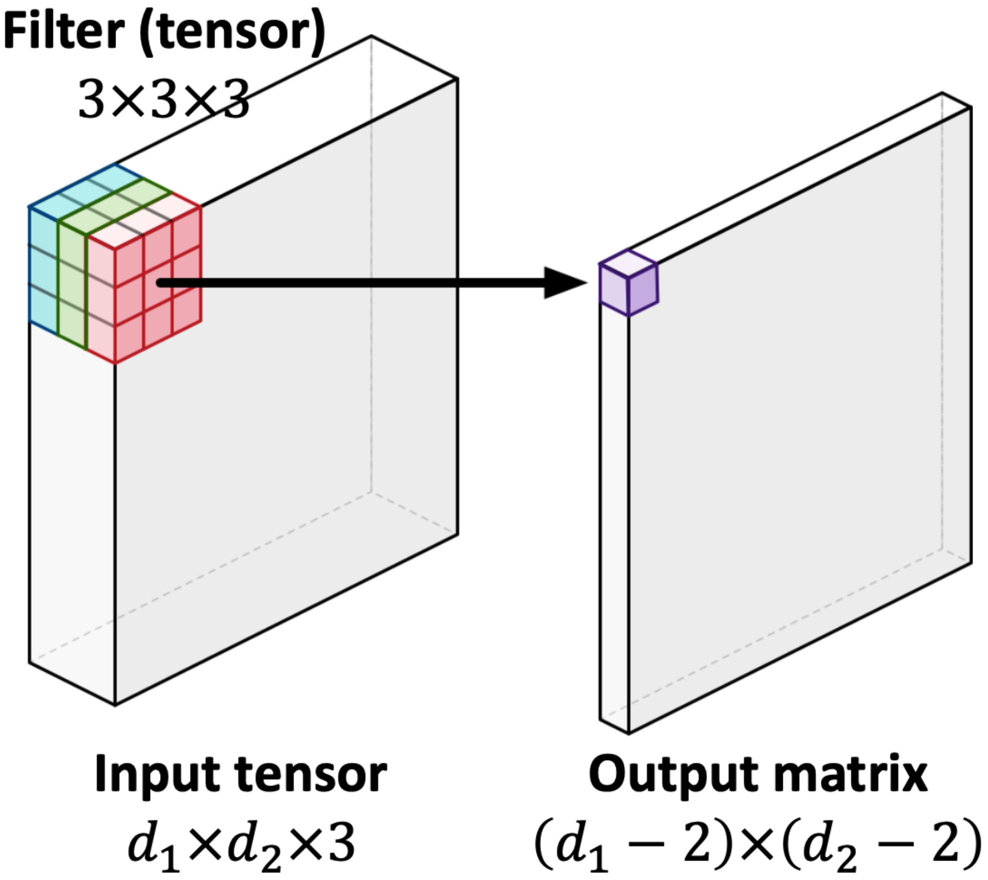
a tensor → a matrix (channel)

↑

a filter/kernel

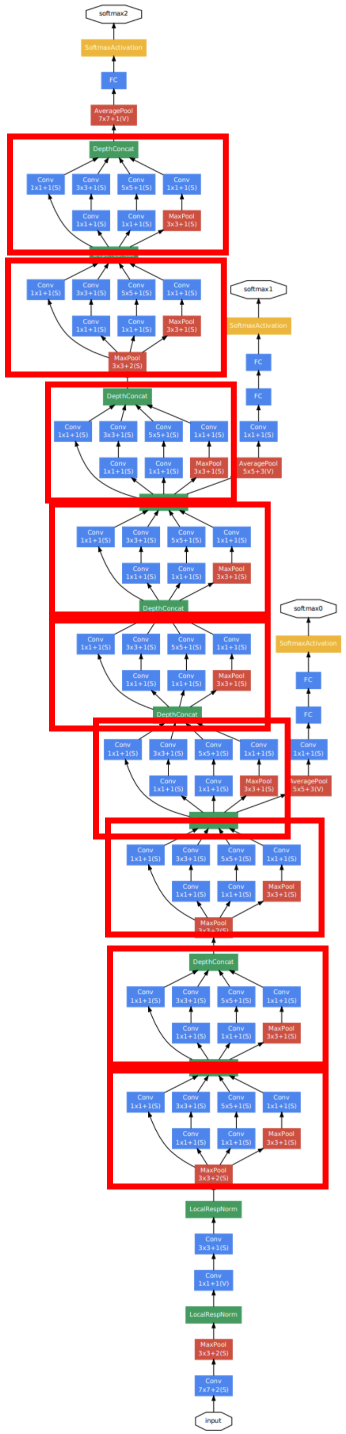


Inception (GoogLeNet)



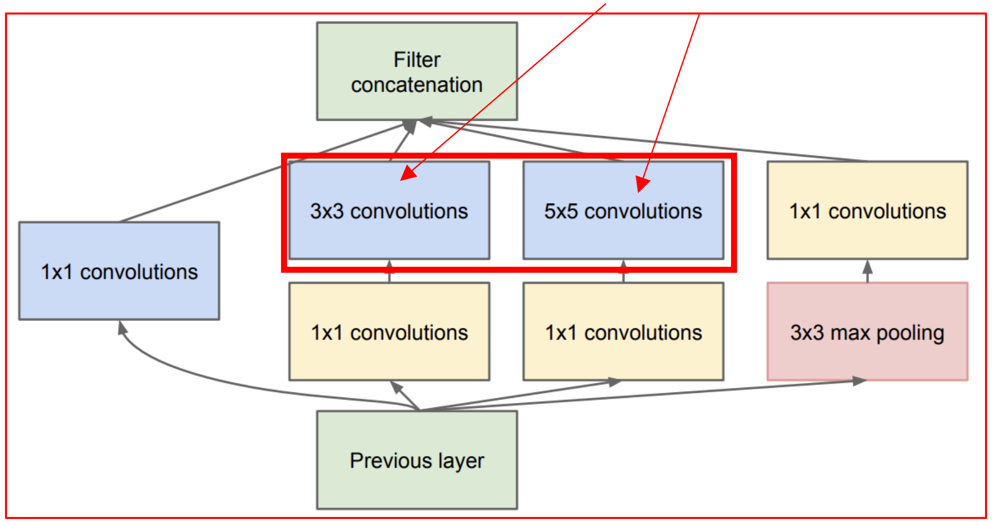
a tensor → m matrices (channels)

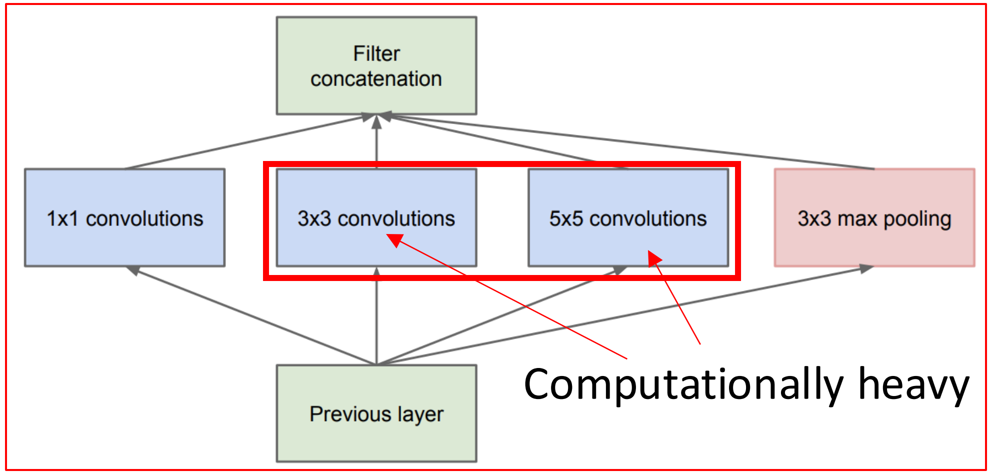
m filter/kernel

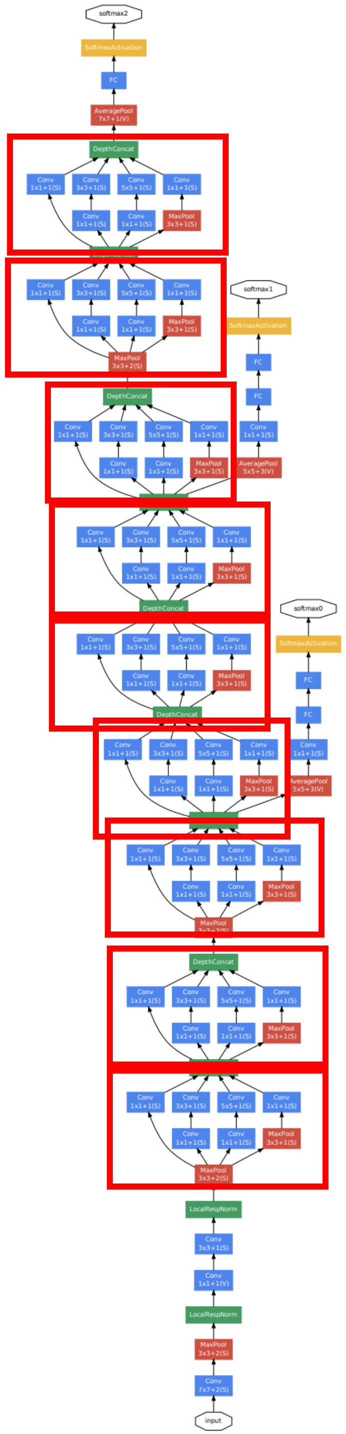


Inception (GoogLeNet)

Computationally heavy

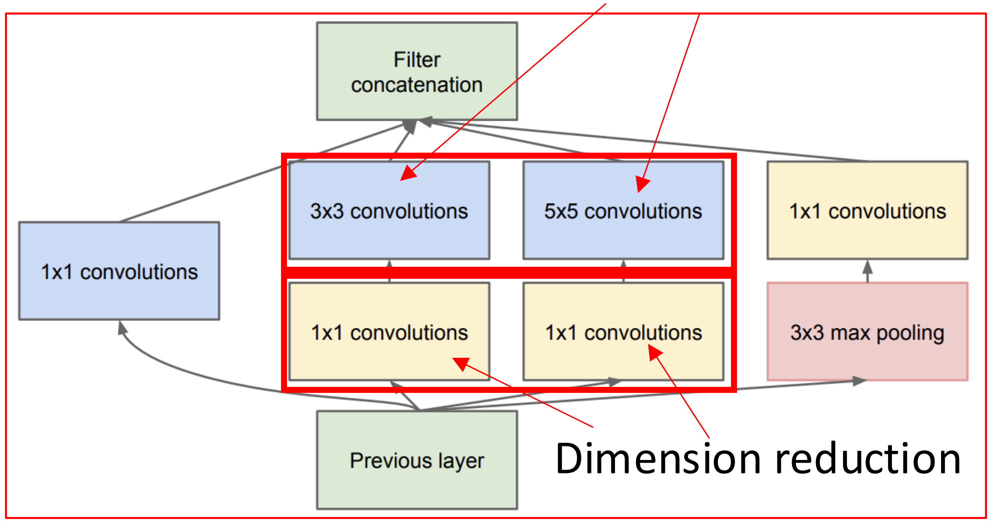


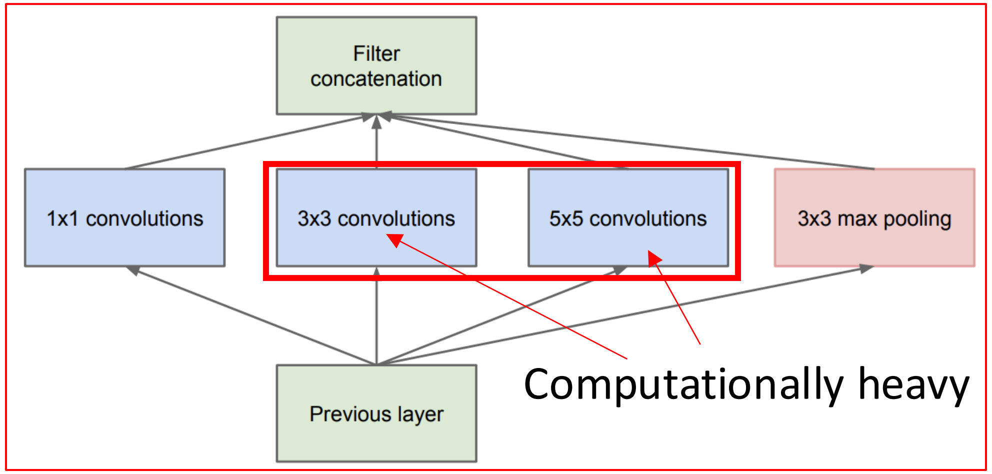


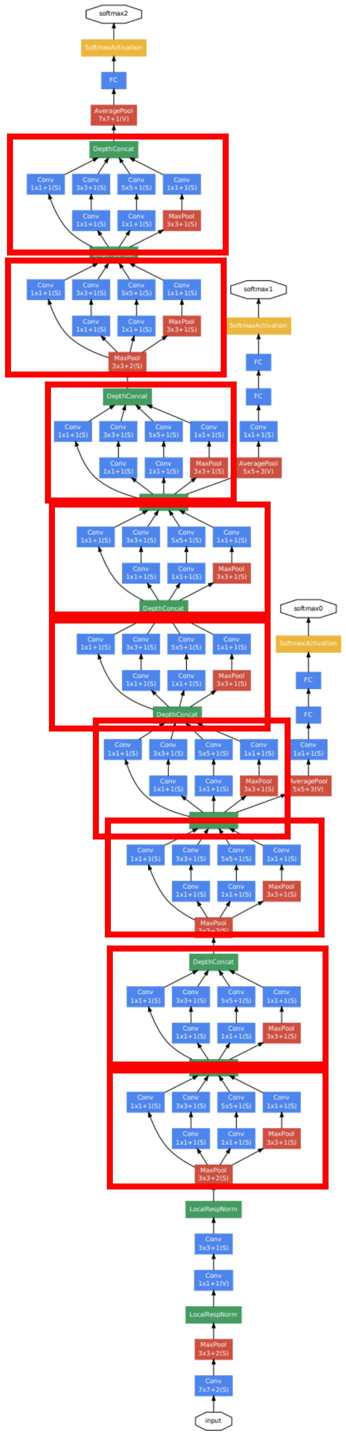


Inception (GoogLeNet)

Computationally heavy

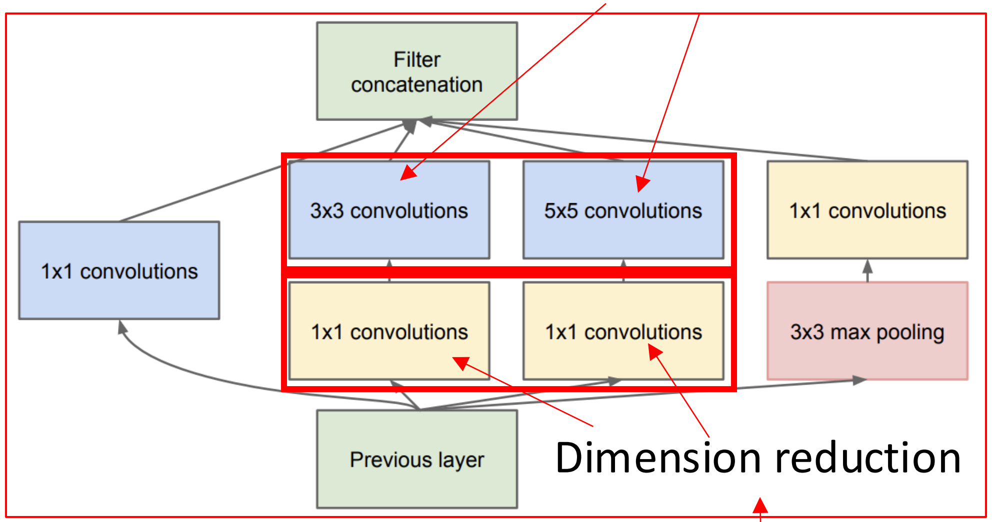




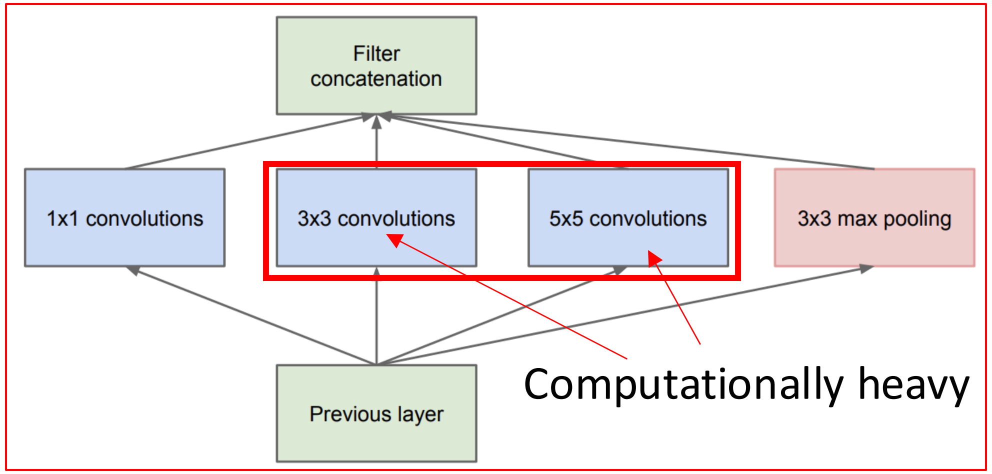


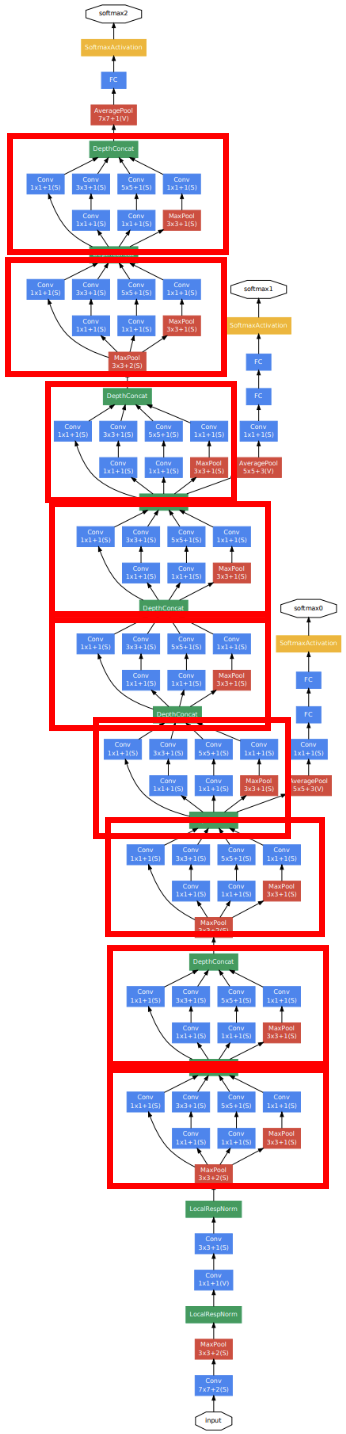
Inception (GoogLeNet)

Computationally heavy

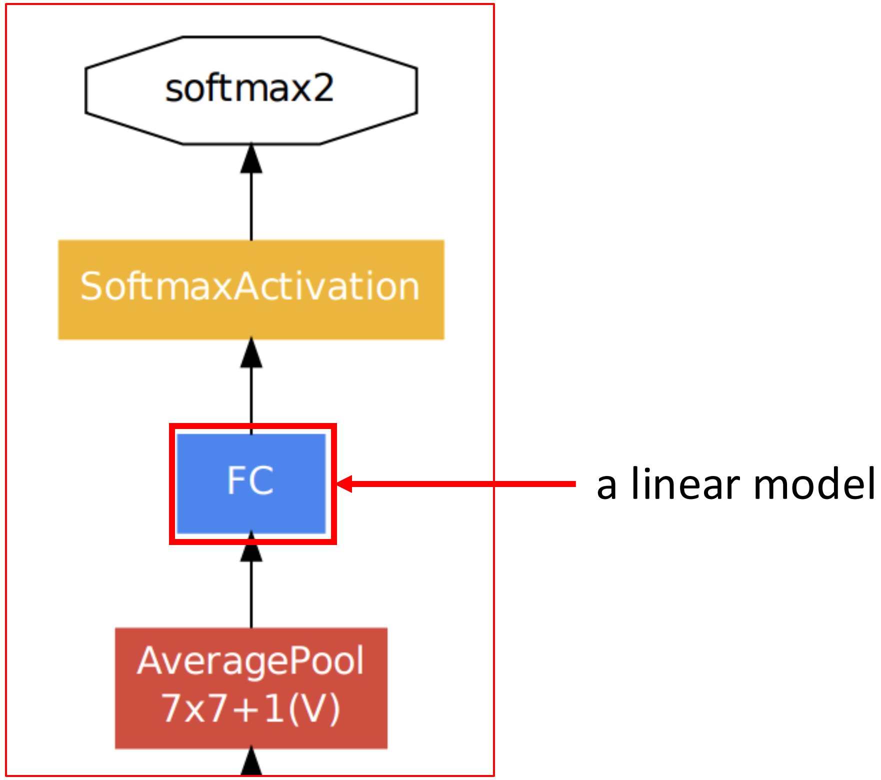


Less channels

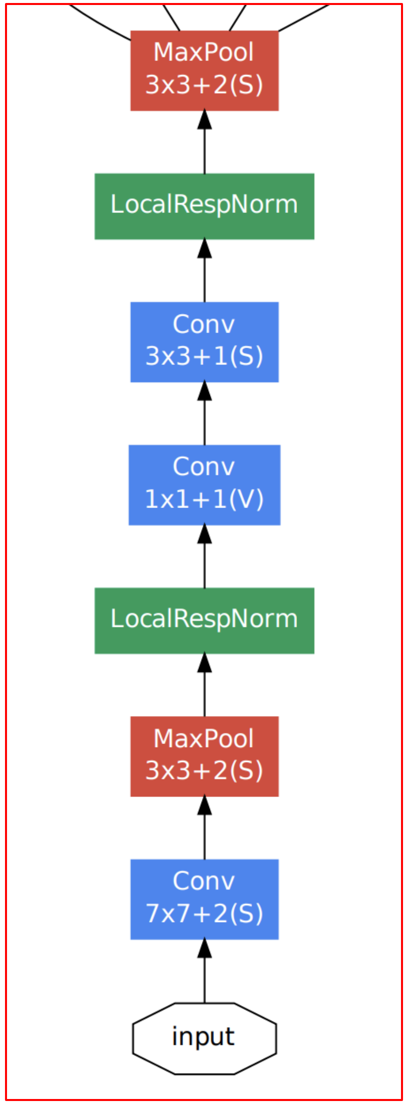


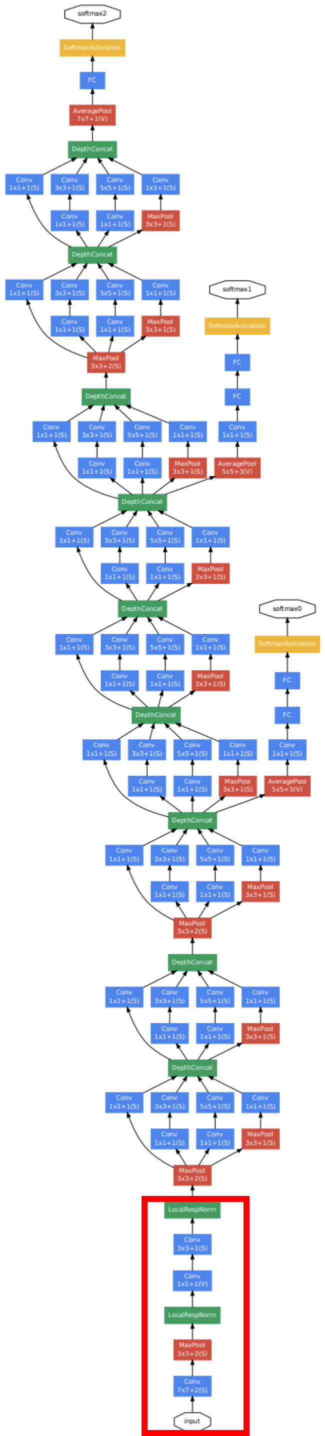


Inception (GoogLeNet)

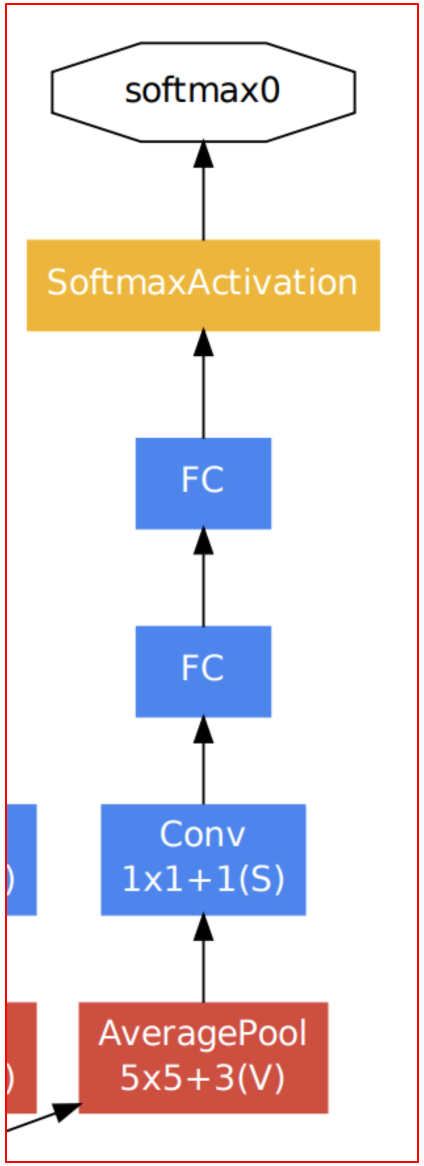


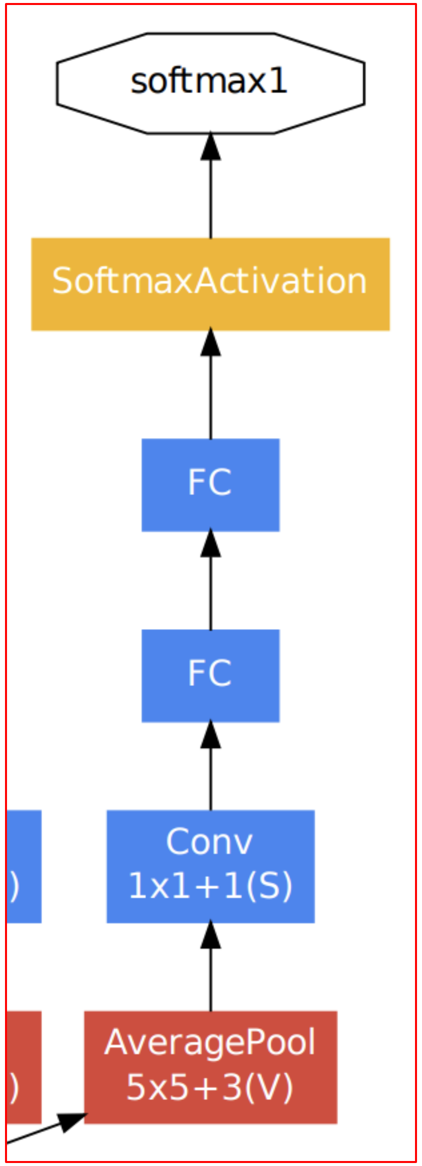
Inception (GoogLeNet)

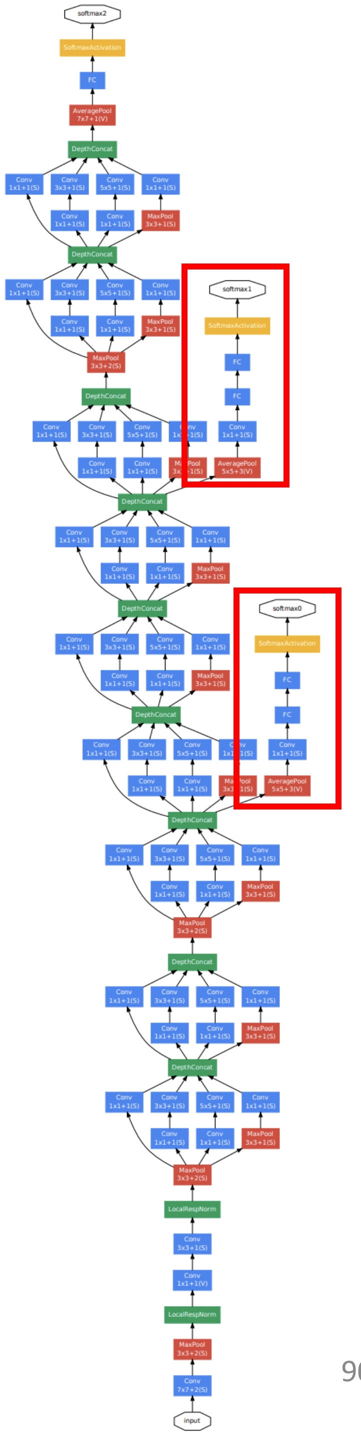




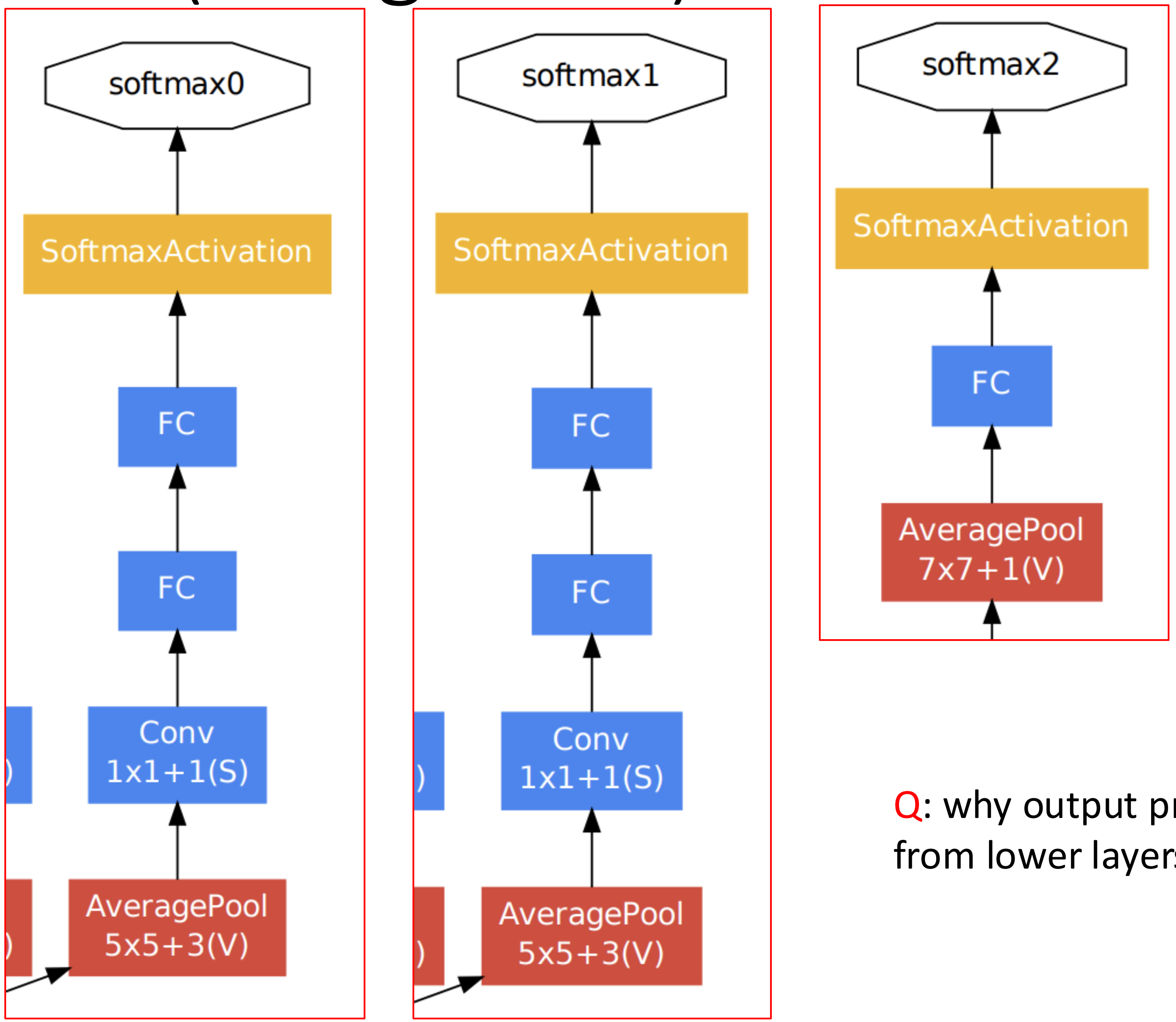
Inception (GoogLeNet)



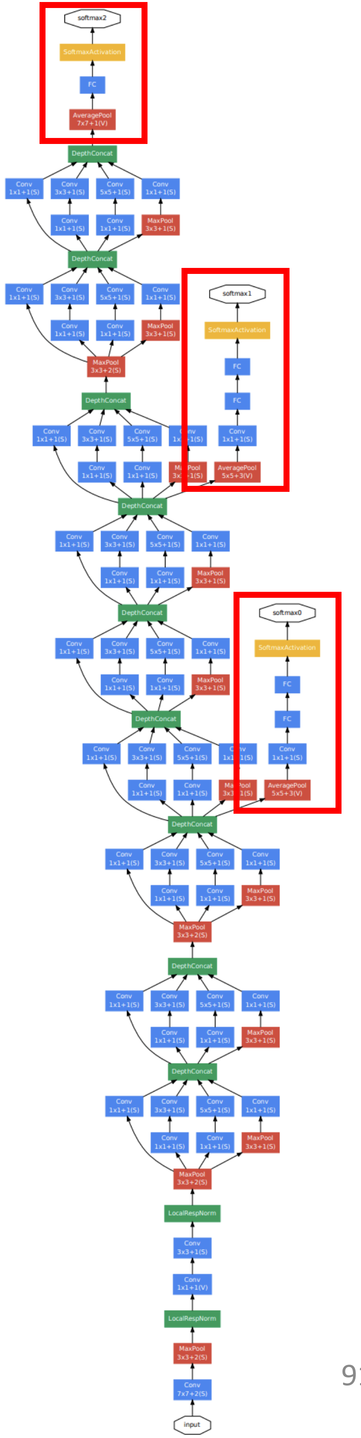




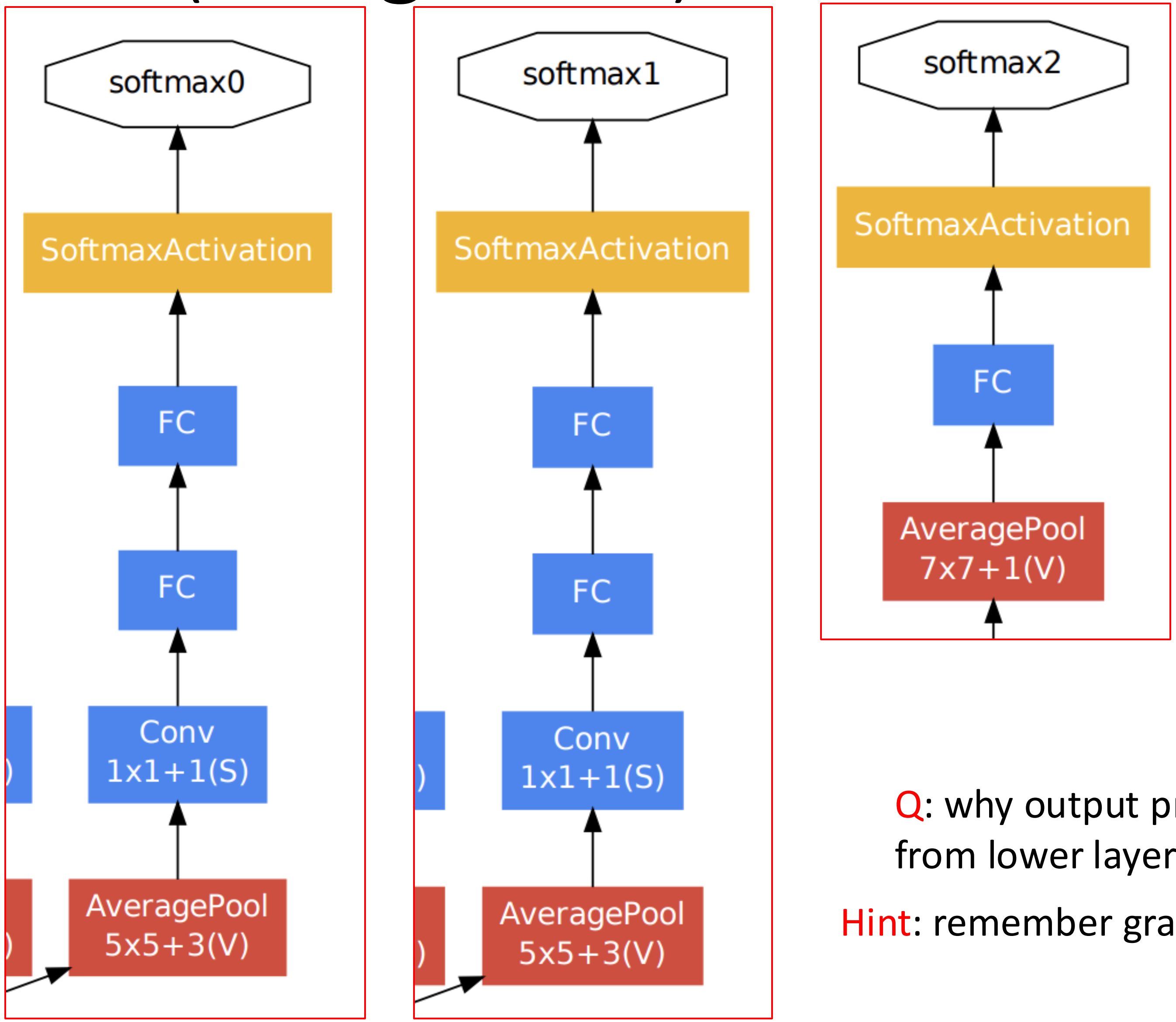
Inception (GoogLeNet)

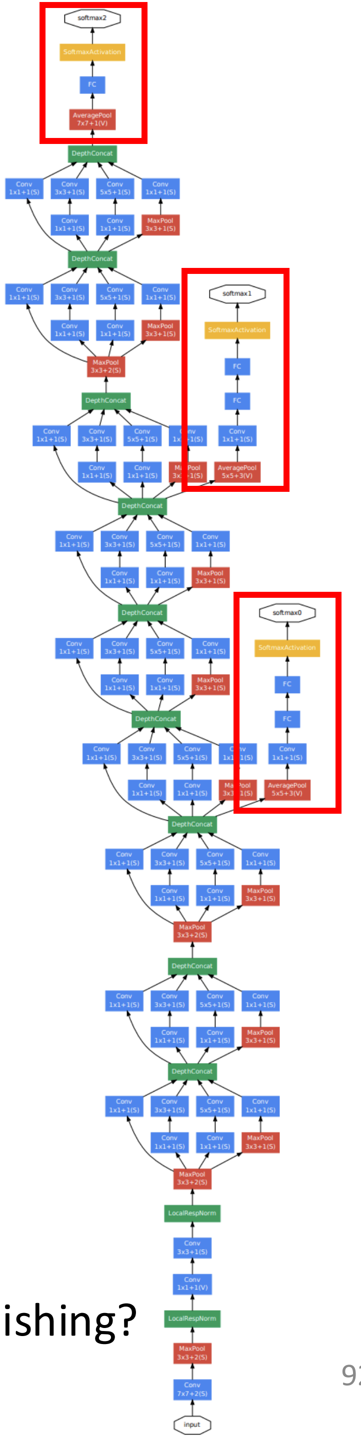


Q: why output prediction from lower layers?



Inception (GoogLeNet)





Q: why output prediction from lower layers?

Hint: remember gradient vanishing?

of layers

- LeNet-5: 3 conv + 2 fc
- AlexNet: 5 conv + 2 fc
- VGG-16: 13 conv + 2 fc

More conv layers

of layers

- LeNet-5: 3 conv + 2 fc
- AlexNet: 5 conv + 2 fc
- VGG-16: 13 conv + 2 fc

More conv layers

Q: why they did not develop some architecture with more layers?

of layers

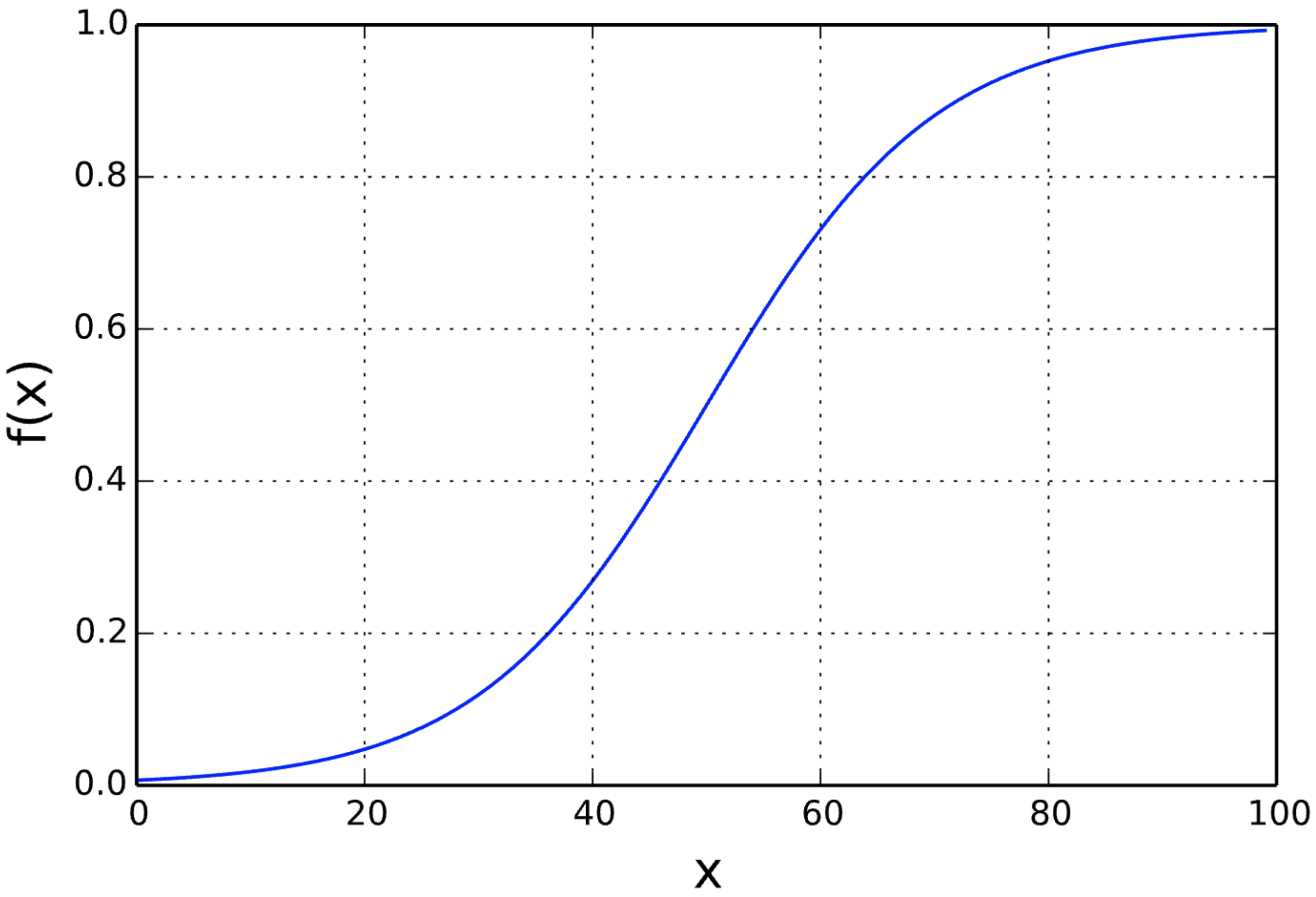
- LeNet-5: 3 conv + 2 fc
- AlexNet: 5 conv + 2 fc
- VGG-16: 13 conv + 2 fc

More conv layers

Q: why they did not develop some architecture with more layers?

Hint (again): remember gradient vanishing?

Gradient vanish



Sigmoid function

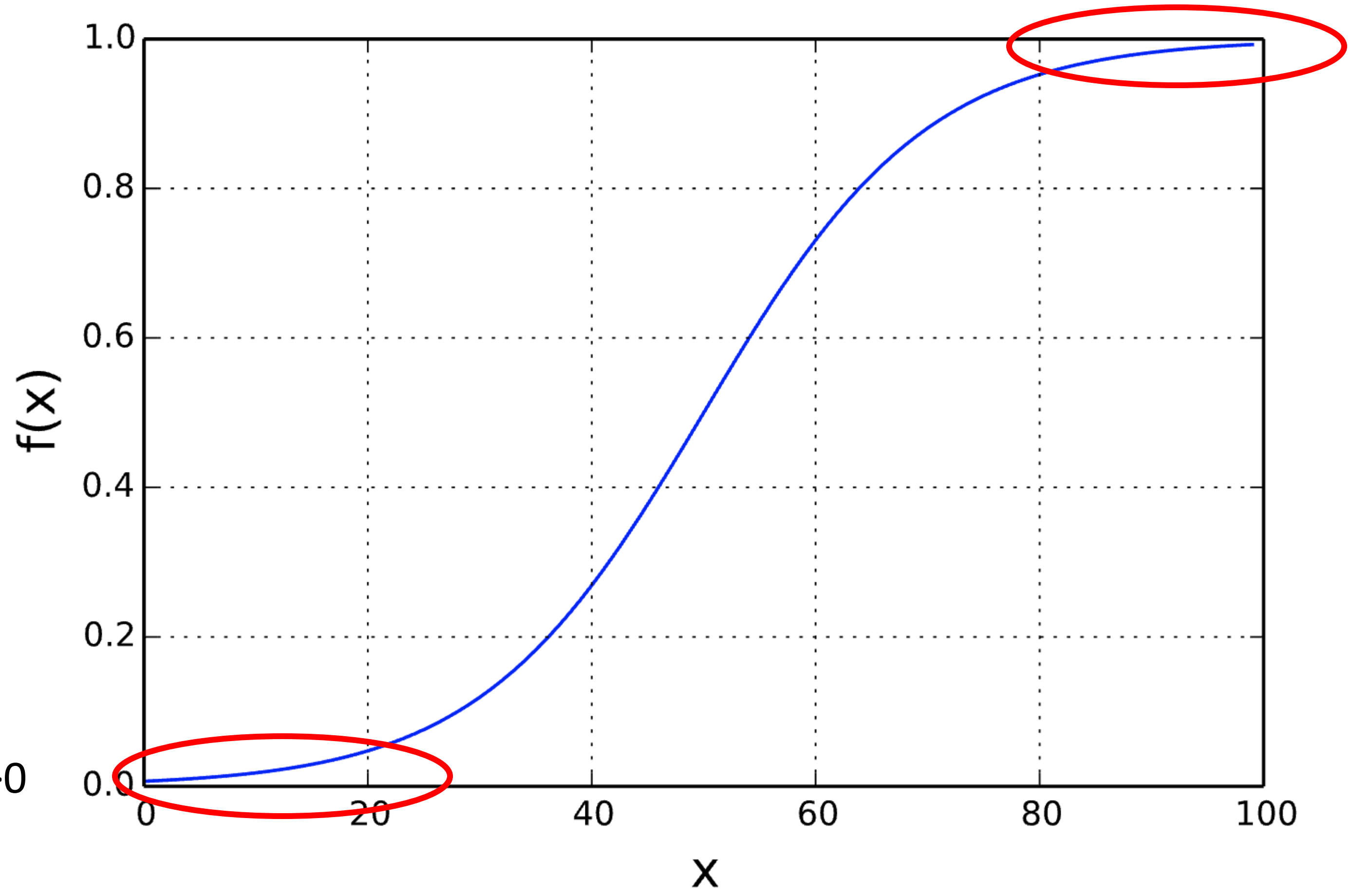
Gradient vanish

$$f_n\left(\dots\left(f_2\left(f_1(x)\right)\right)\right) \to ?$$

$$f_i \to x_i$$

$$\frac{dx_n}{dx_1} = \frac{dx_n}{dx_{n-1}} \cdot \dots \cdot \frac{dx_2}{dx_1} \cdot \frac{dx_1}{dx}$$

gradients->0



gradients->0

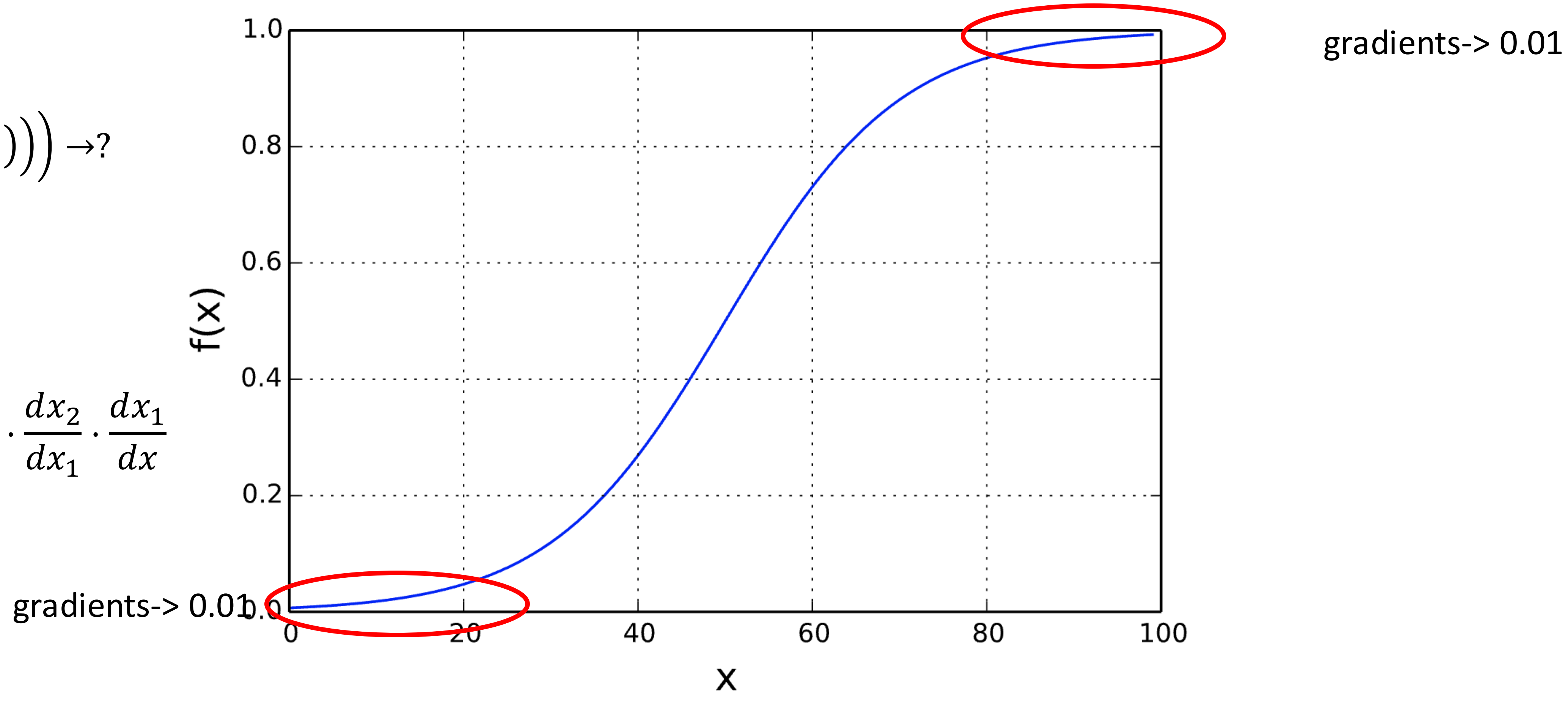
Sigmoid function

Gradient vanish

$$f_n\left(\ldots\left(f_2\left(f_1(x)\right)\right)\right) \to ?$$

$$f_i \to x_i$$

$$\frac{dx_n}{dx_1} = \frac{dx_n}{dx_{n-1}} \cdot \ldots \cdot \frac{dx_2}{dx_1} \cdot \frac{dx_1}{dx}$$



Sigmoid function

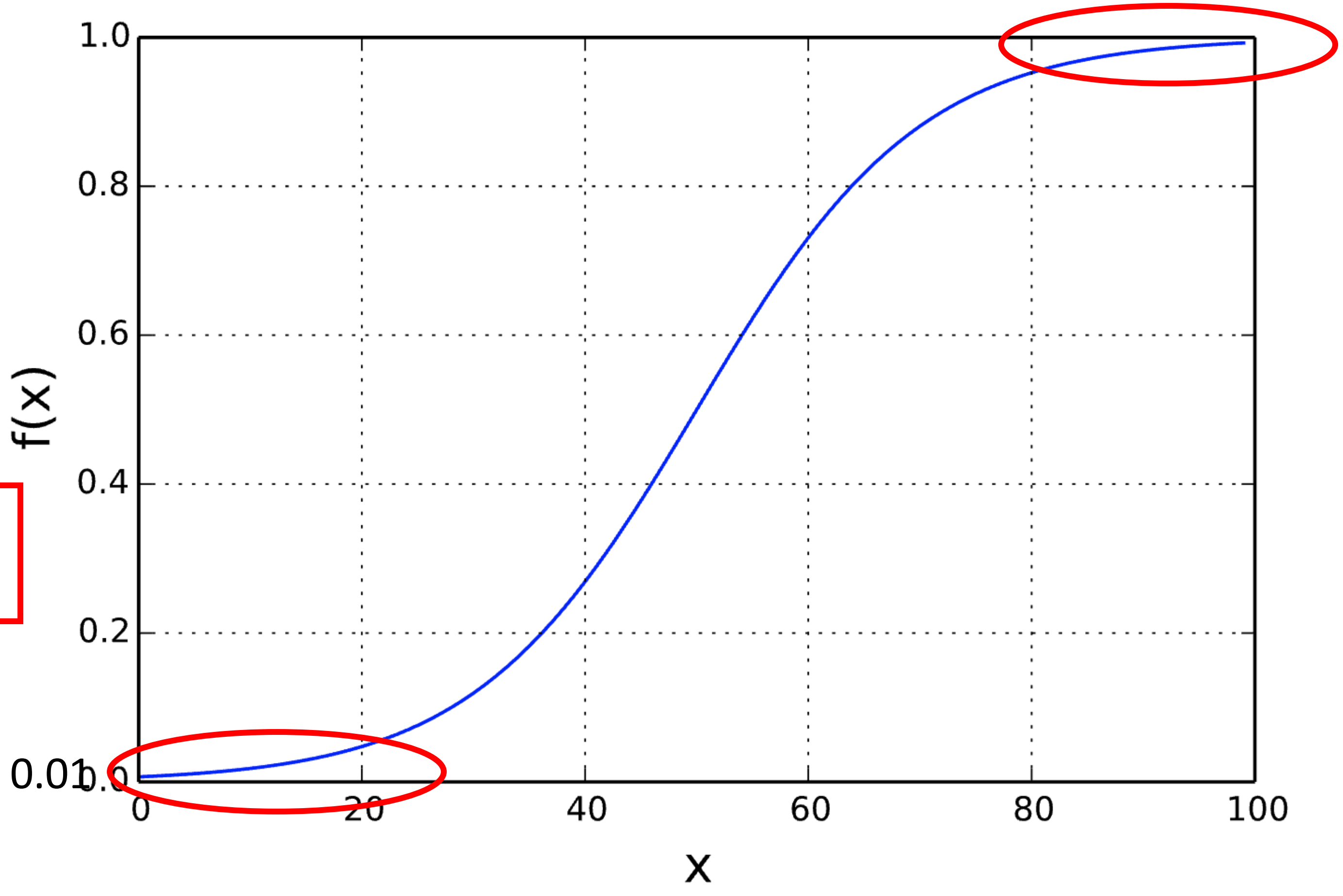
Gradient vanish

$$f_n\left(\dots\left(f_2\left(f_1(x)\right)\right)\right) \to ?$$

$$f_i \to x_i$$

$$\frac{dx_n}{dx_1} = \boxed{\frac{dx_n}{dx_{n-1}}} \cdot \dots \cdot \boxed{\frac{dx_2}{dx_1}} \cdot \boxed{\frac{dx_1}{dx}}$$

gradients-> 0.01



gradients-> 0.01

Sigmoid function

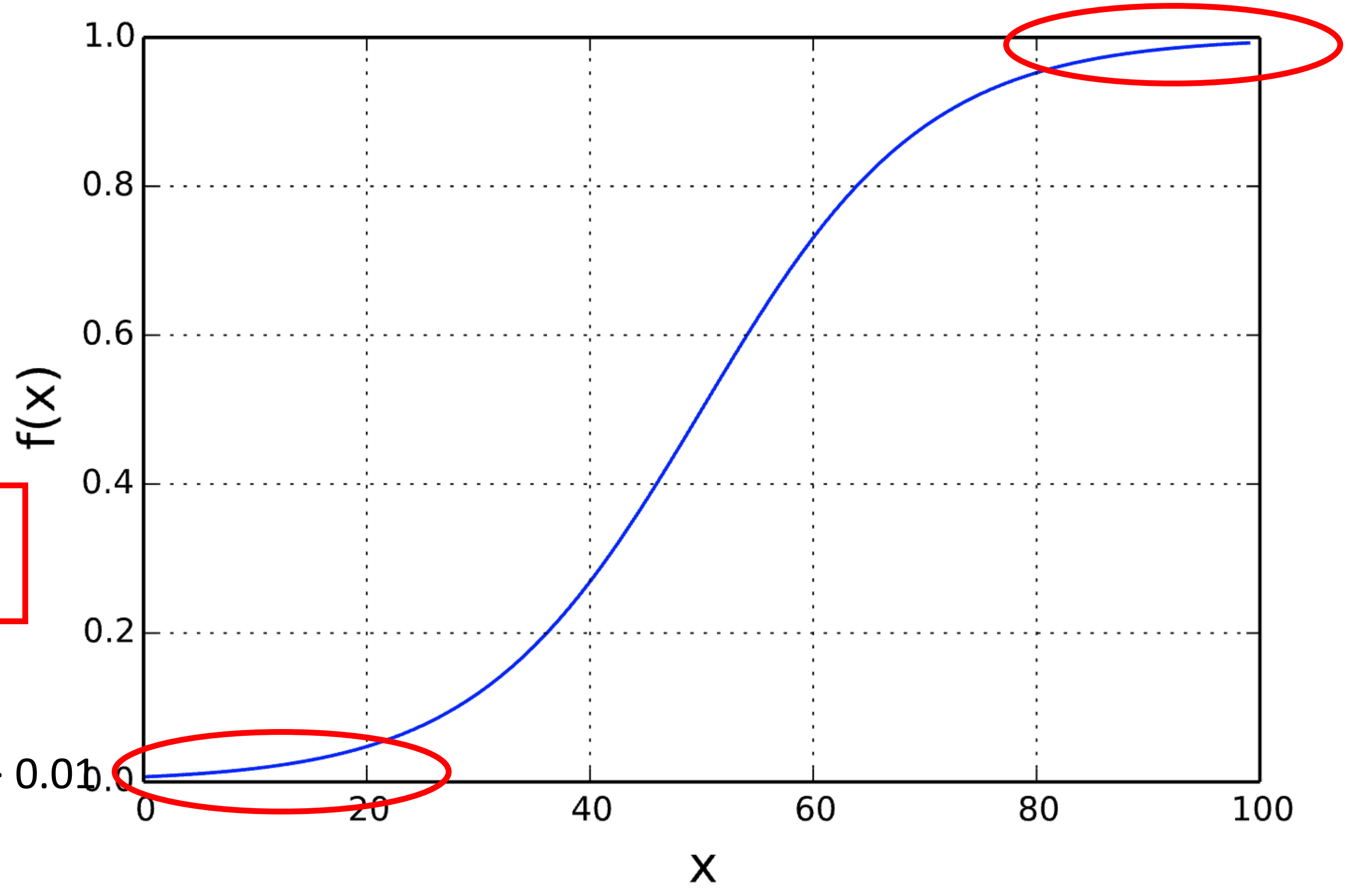
Gradient vanish

$$f_n\left(\ldots\left(f_2\left(f_1(x)\right)\right)\right) \to ?$$

$$f_i \to x_i$$

$$\frac{dx_n}{dx_1} = \boxed{\frac{dx_n}{dx_{n-1}}} \cdot \ldots \cdot \boxed{\frac{dx_2}{dx_1}} \cdot \boxed{\frac{dx_1}{dx}}$$

$\to 0.01^n$



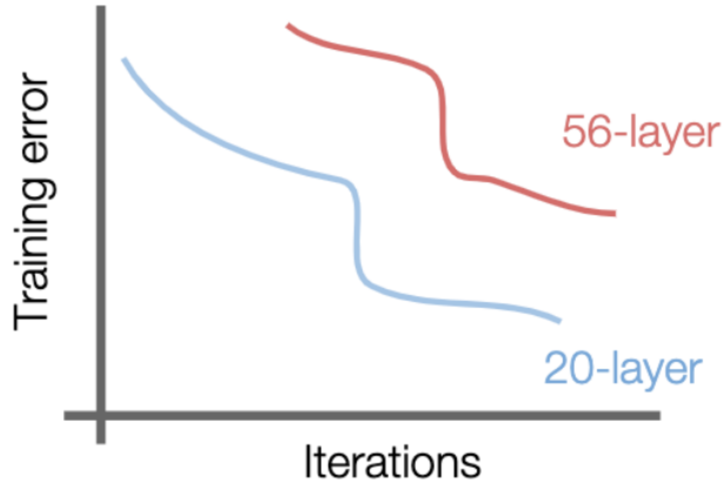
Sigmoid function

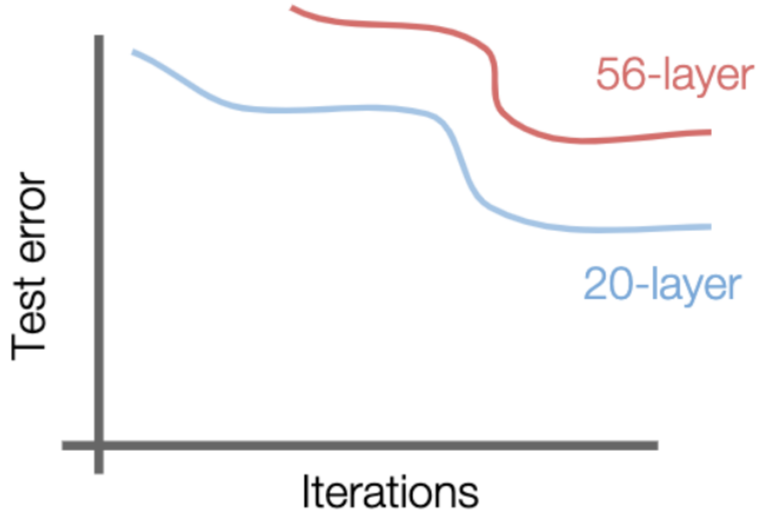
of layers

- LeNet-5: 3 conv + 2 fc
- AlexNet: 5 conv + 2 fc
- VGG-16: 13 conv + 2 fc

Q: why they did not develop some architecture with more layers?

Optimization may be difficult.



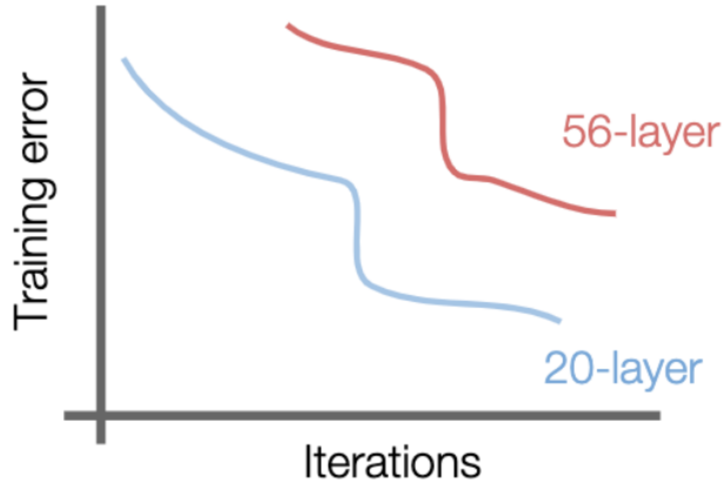


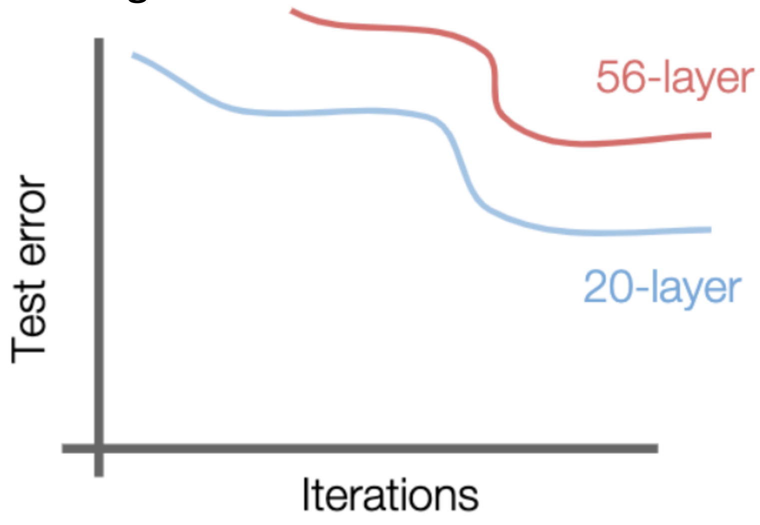
of layers

- LeNet-5: 3 conv + 2 fc
- AlexNet: 5 conv + 2 fc
- VGG-16: 13 conv + 2 fc

Q: why they did not develop some architecture with more layers?

Optimization may be difficult. We do not have a good solution as our model.



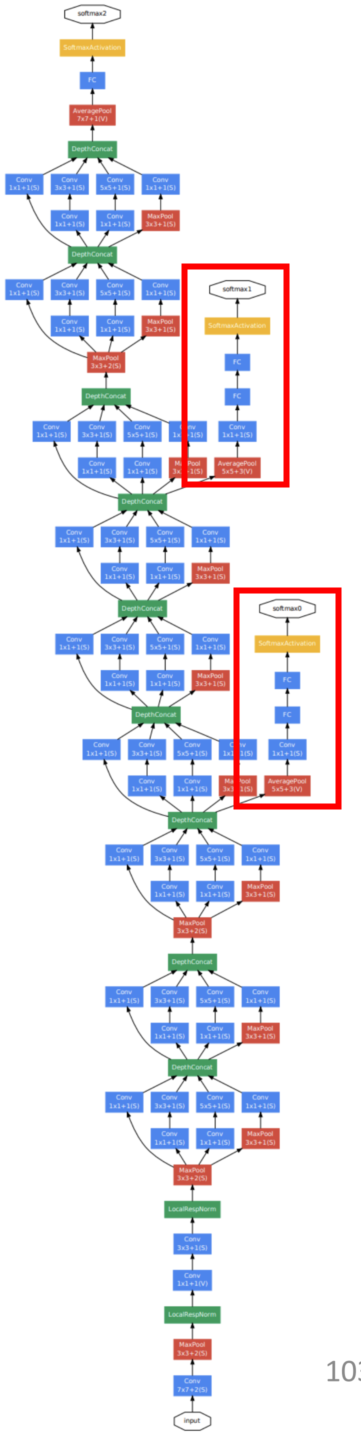


Inception

$$f_n\Big(\dots\big(f_2(f_1(x))\big)\Big) \rightarrow ?$$

$$\frac{dx_n}{dx_1} = \boxed{\frac{dx_n}{dx_{n-1}}} \cdot \dots \cdot \boxed{\frac{dx_2}{dx_1}} \boxed{\frac{dx_1}{dx}}$$

$\rightarrow 0.01^n$

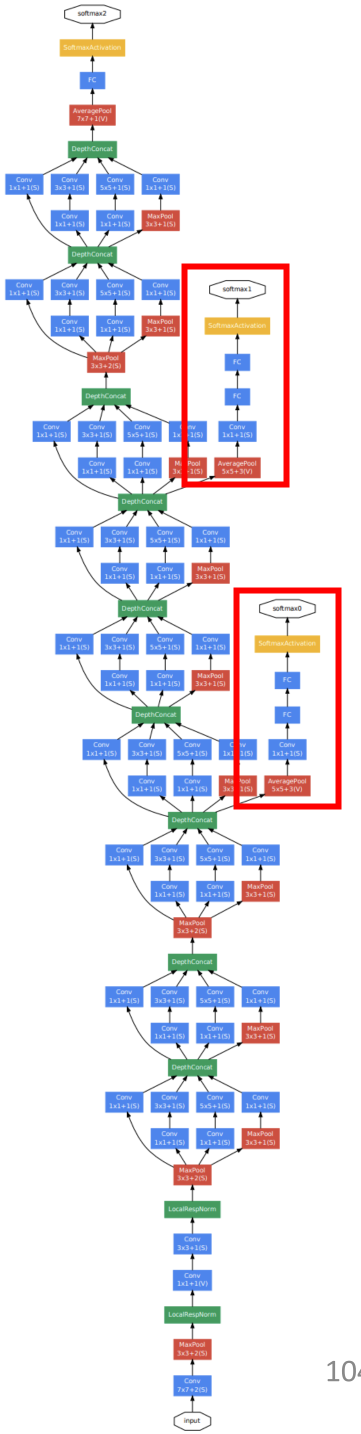


Inception

$$f_n\left(\dots\left(f_2\left(f_1(x)\right)\right)\right) + f_m\left(\dots\left(f_{n+2}\left(f_{n+1}(x)\right)\right)\right)$$

$$\frac{dx_n}{dx_1} = \boxed{\frac{dx_n}{dx_{n-1}}} \cdot \dots \cdot \boxed{\frac{dx_2}{dx_1}} \cdot \boxed{\frac{dx_1}{dx}}$$

$\rightarrow 0.01^n$



Inception

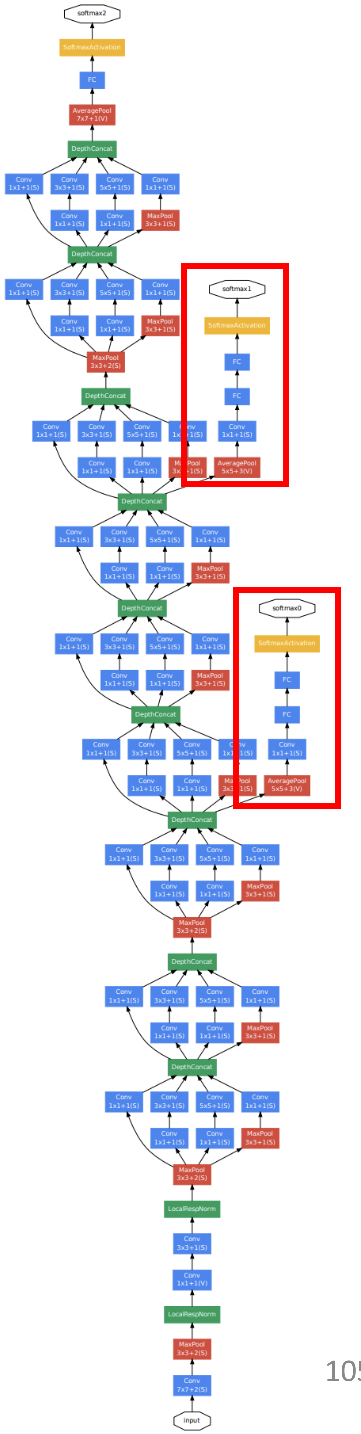
$$f_n\left(\dots\left(f_2\left(f_1(x)\right)\right)\right) + f_m\left(\dots\left(f_{n+2}\left(f_{n+1}(x)\right)\right)\right)$$

$$\boxed{\frac{dx_n}{dx_1}} = \boxed{\frac{dx_n}{dx_{n-1}} \cdot \dots \cdot \frac{dx_2}{dx_1} \cdot \frac{dx_1}{dx}} + \boxed{\frac{dx_m}{dx_{m-1}} \cdot \dots \cdot \frac{dx_{n+2}}{dx_{n+1}} \cdot \frac{dx_{n+1}}{dx}}$$

$\to 0.01^n$

>> 0

Will not be very small



Residual neural networks (ResNet)

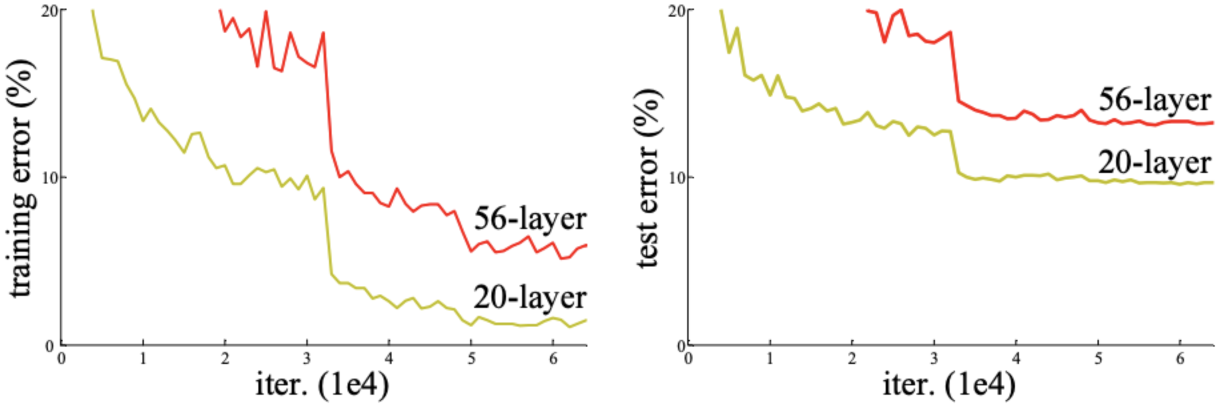


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Residual neural networks (ResNet)

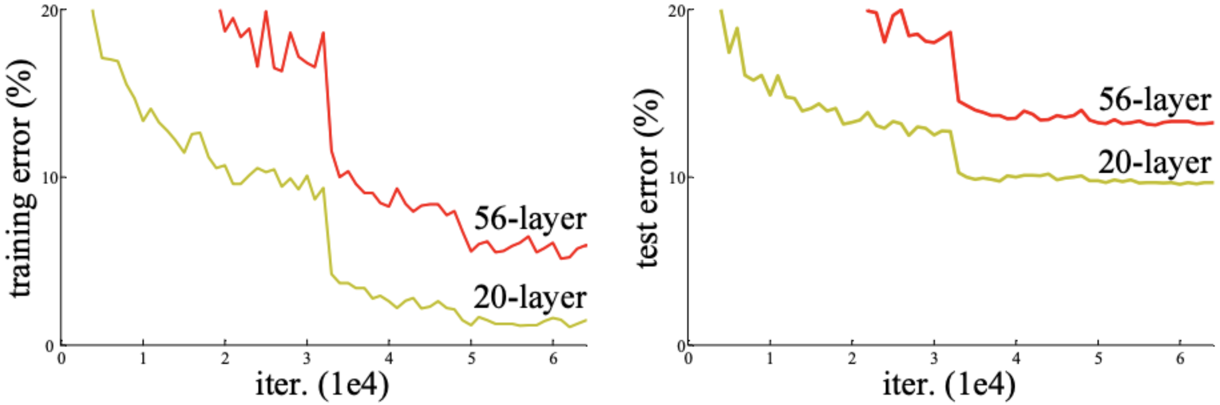


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Without special structure other than conv/fc layers

ResNet: shortcut connection

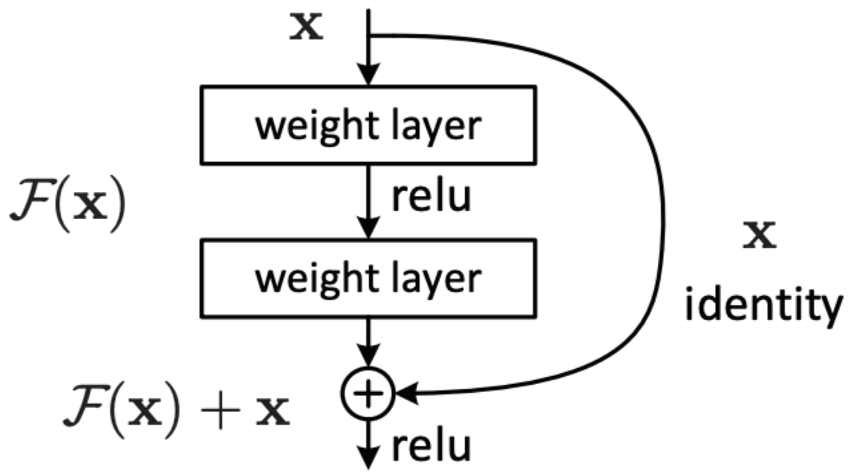


Figure 2. Residual learning: a building block.

ResNet: shortcut connection

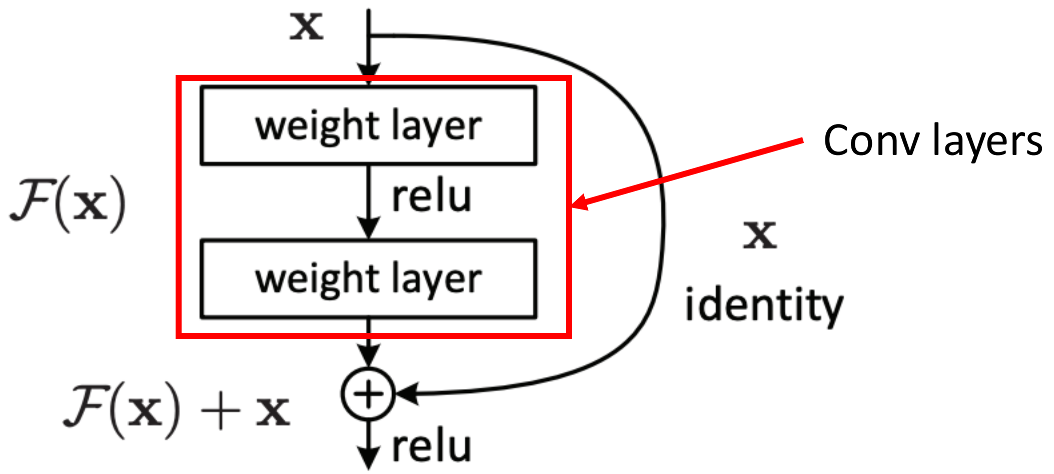


Figure 2. Residual learning: a building block.

ResNet: shortcut connection

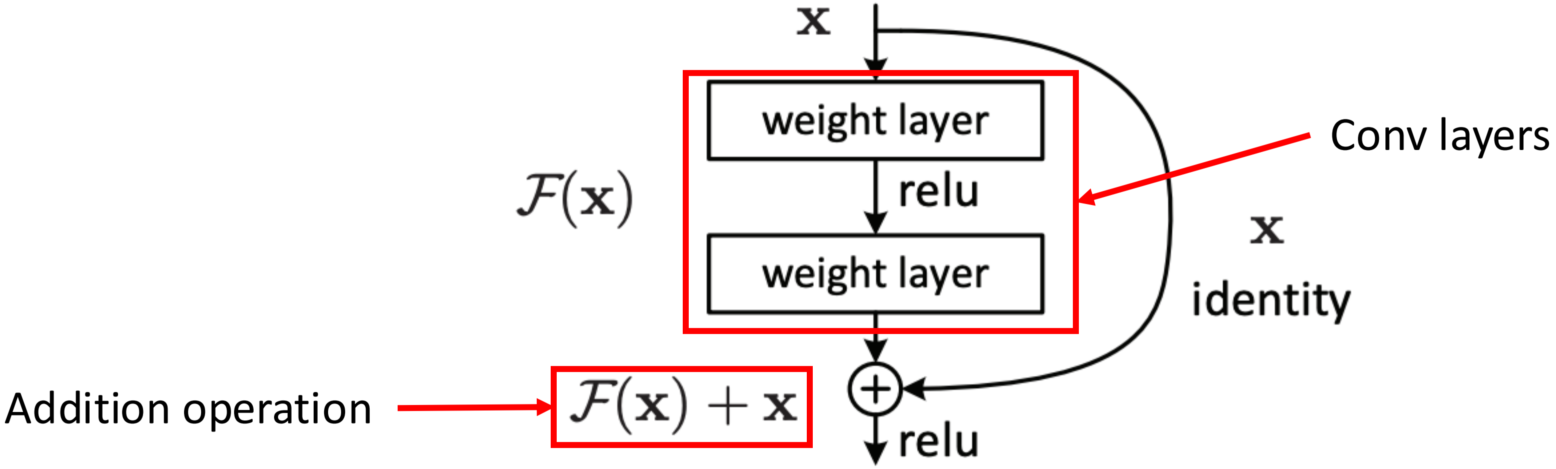
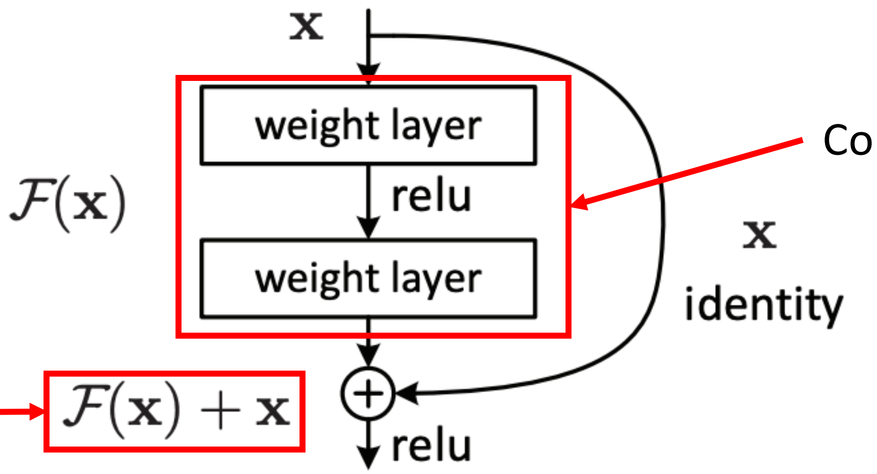


Figure 2. Residual learning: a building block.

ResNet: shortcut connection



Conv layers

Addition operation

implication: same dimension

Figure 2. Residual learning: a building block.

ResNet

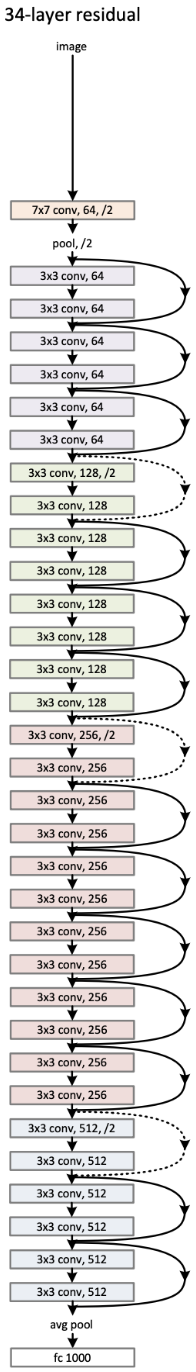


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet

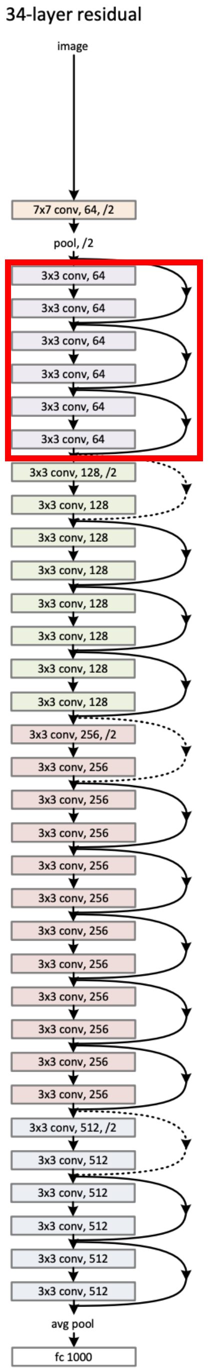
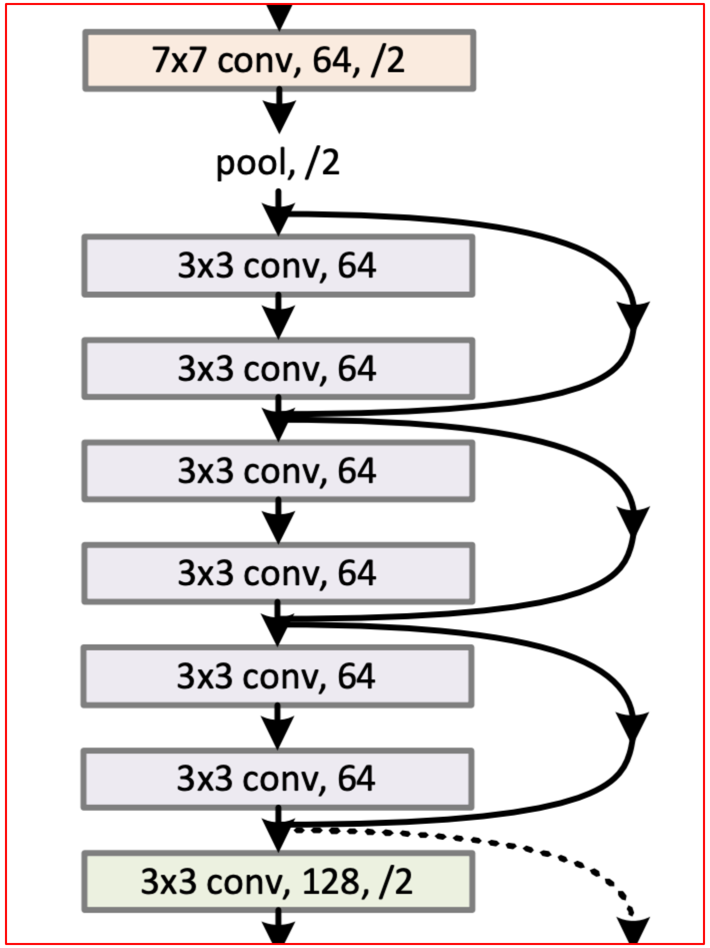


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet



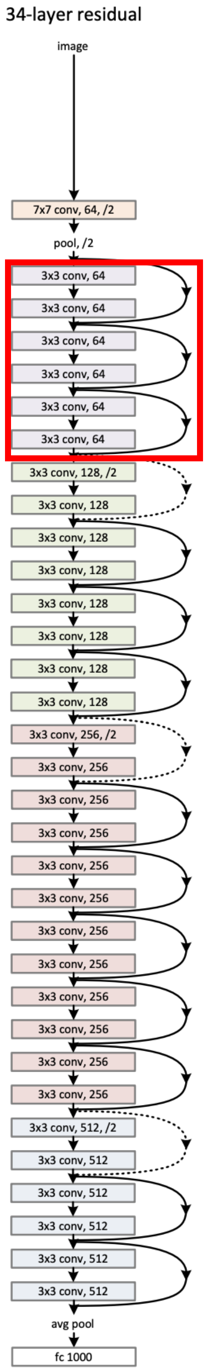
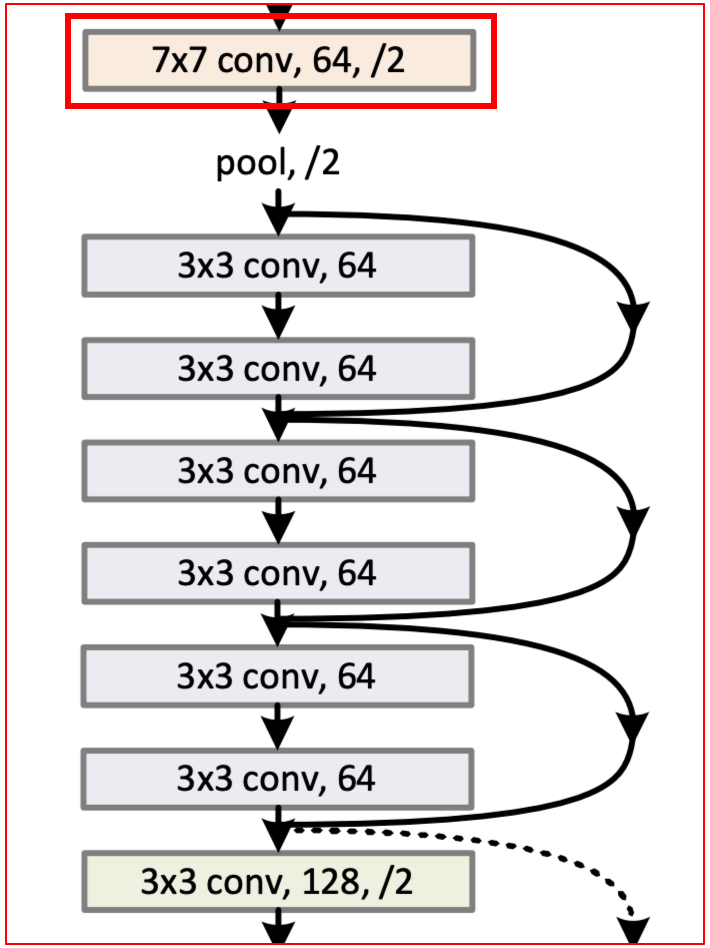


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet



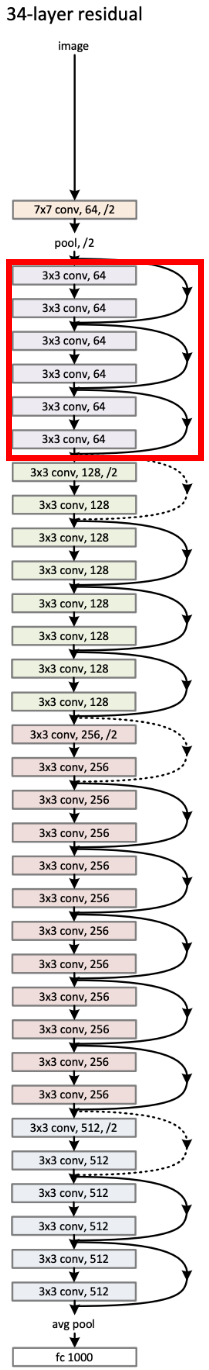
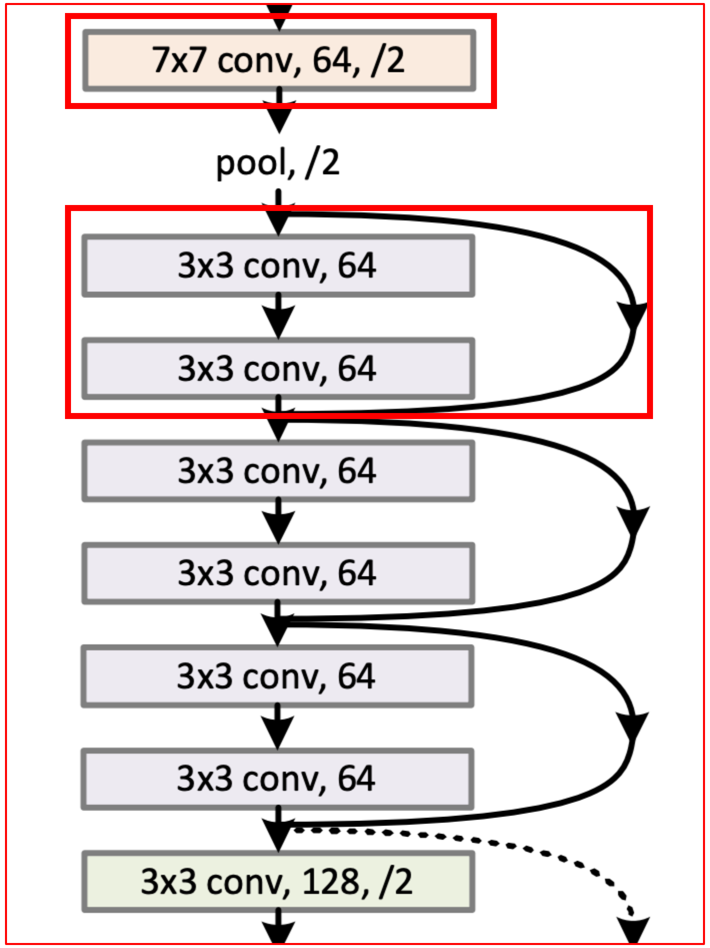


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet



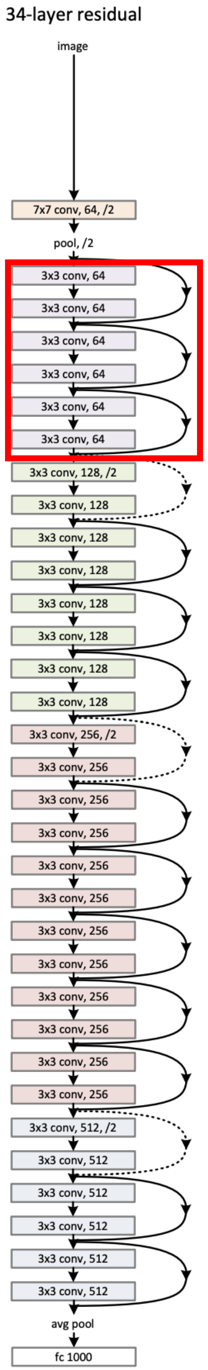
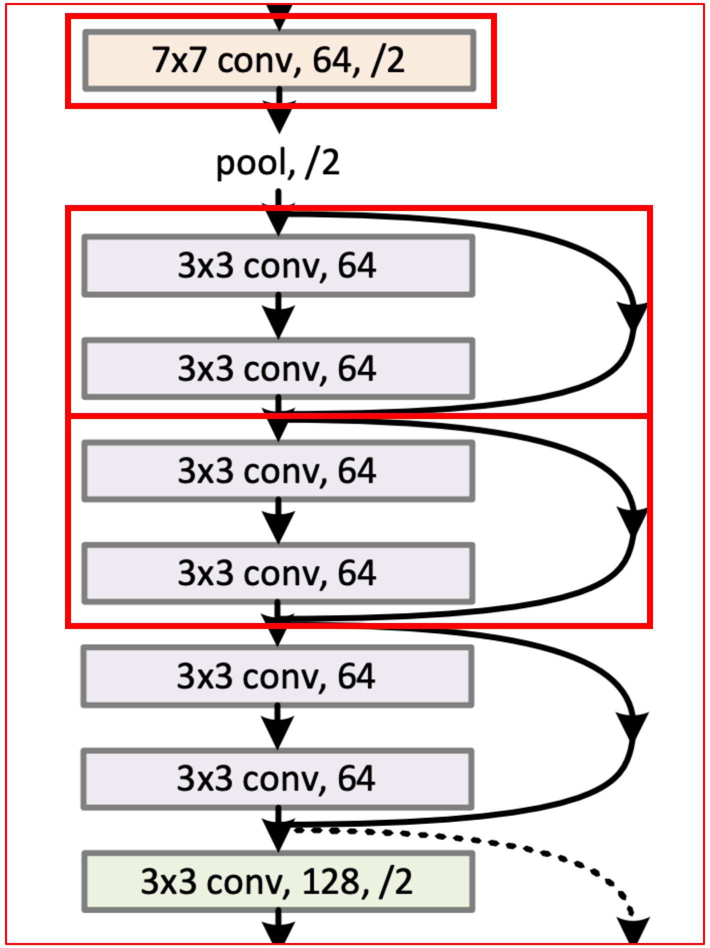


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet



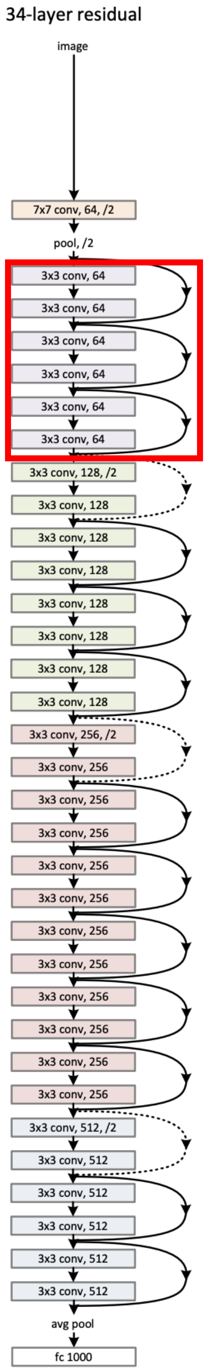
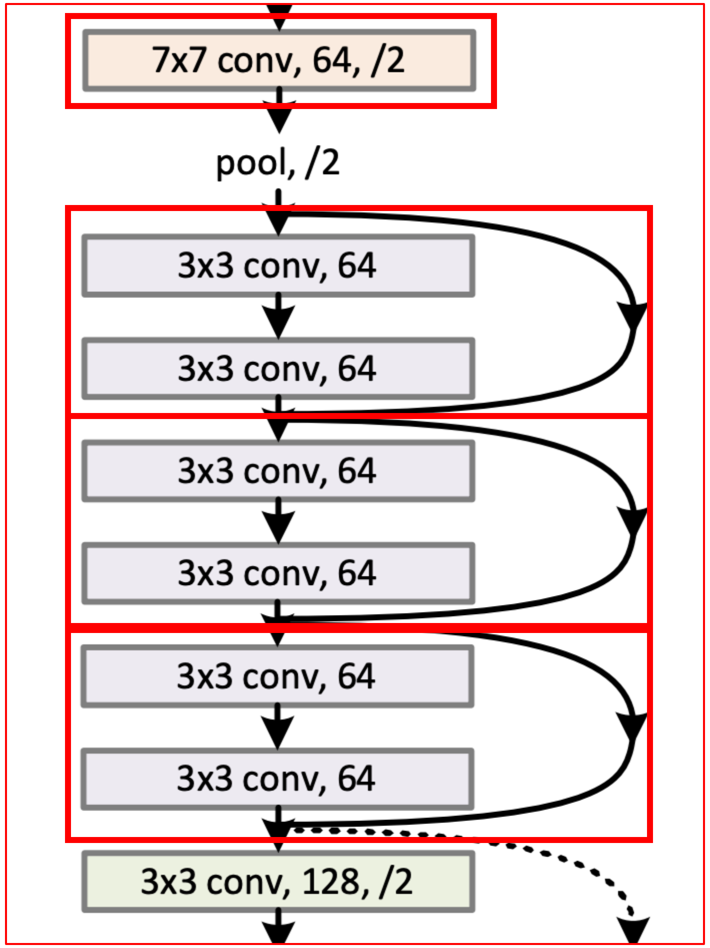


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet



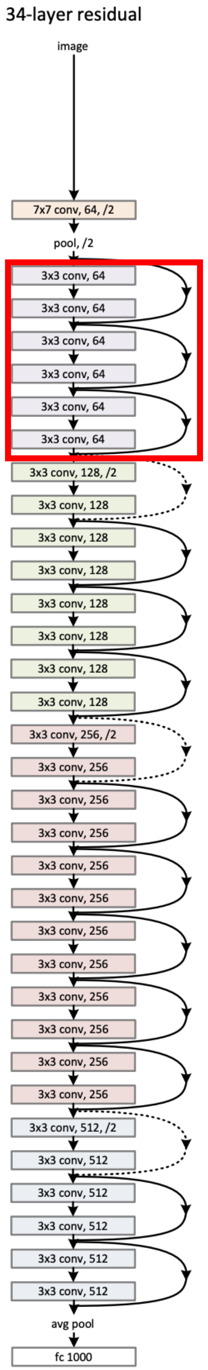
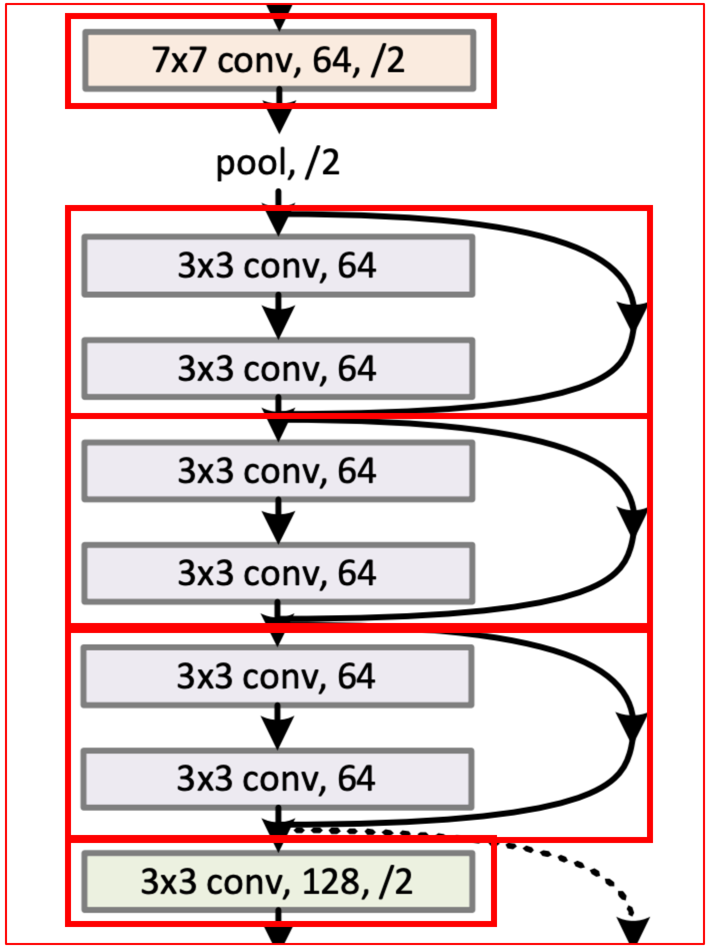


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet



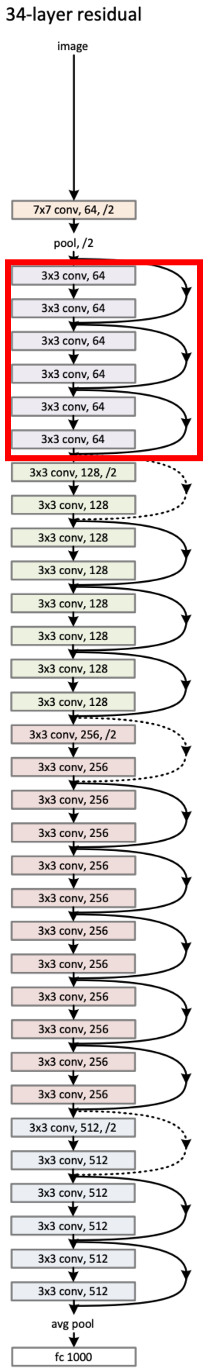


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet

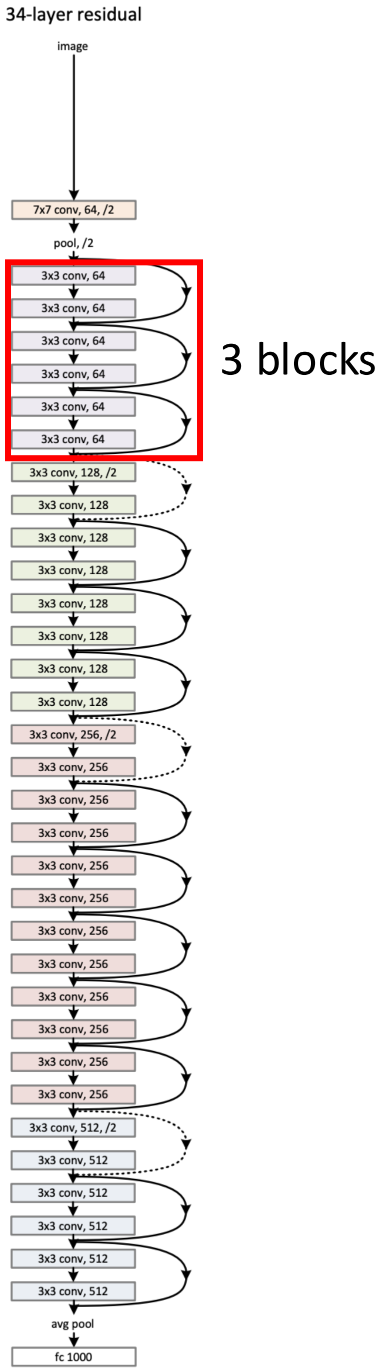
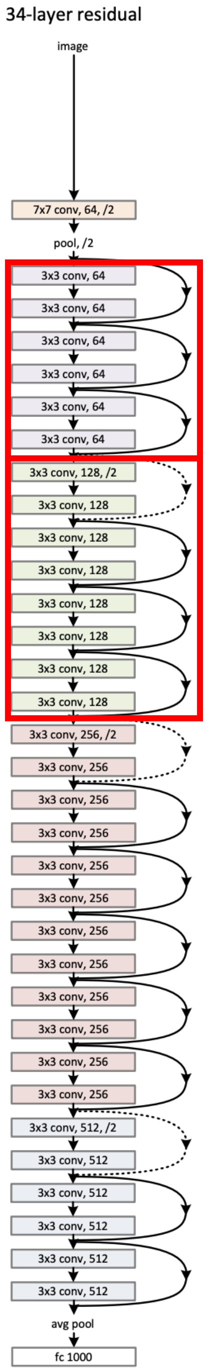


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet



3 blocks

4 blocks

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet

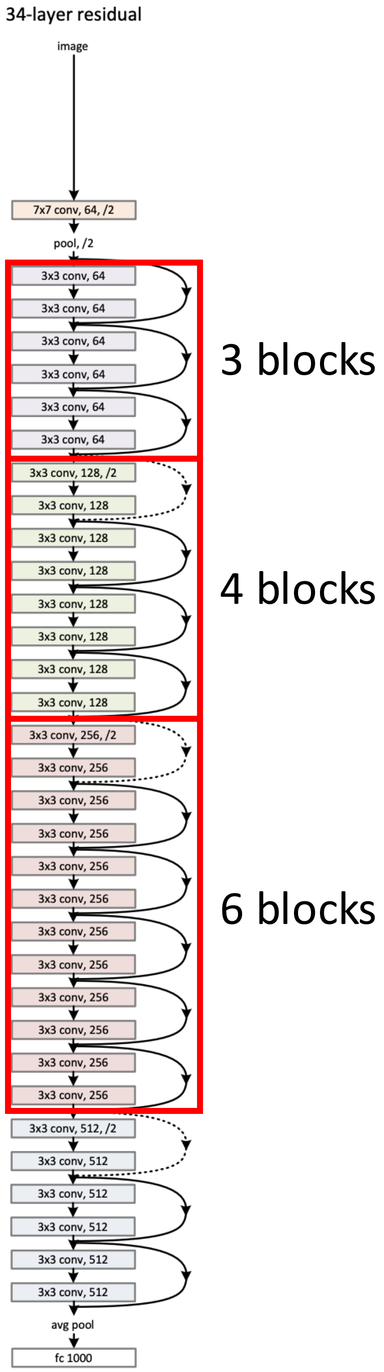


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet

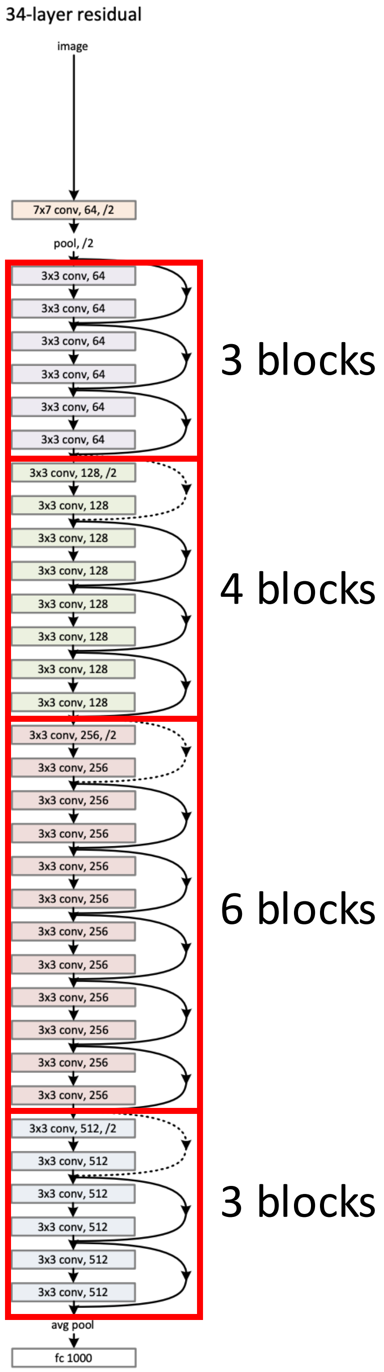


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet

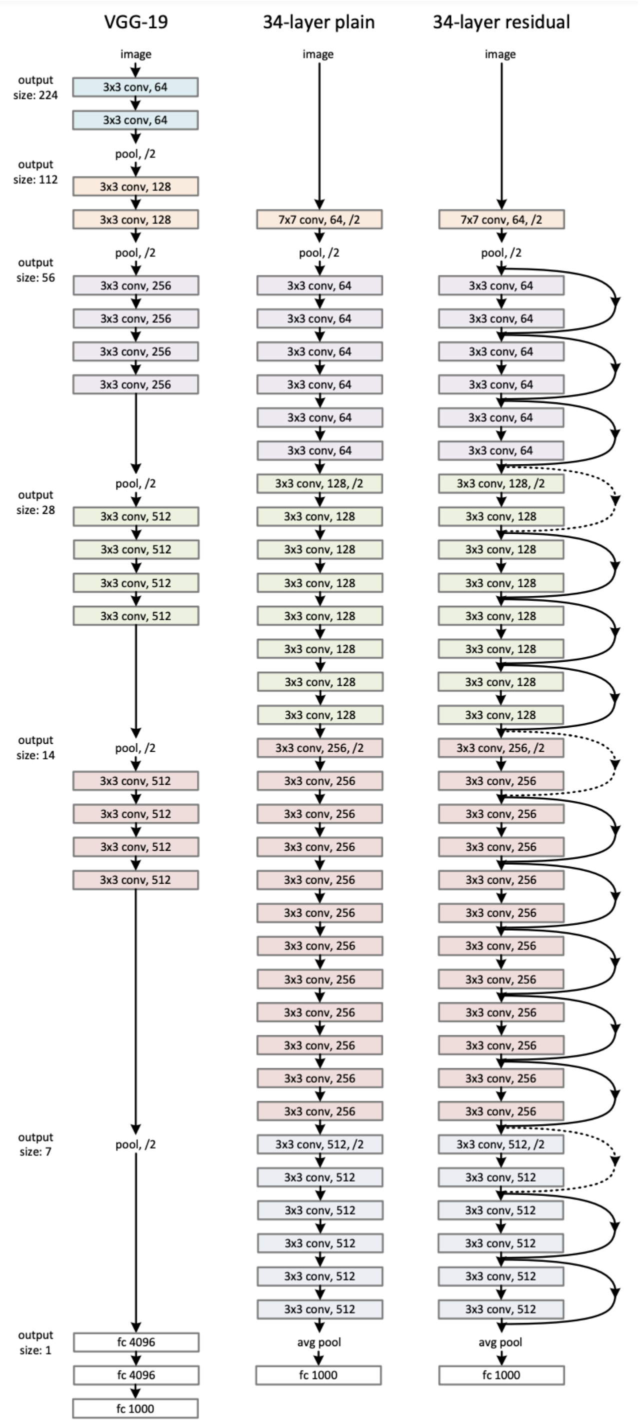


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet

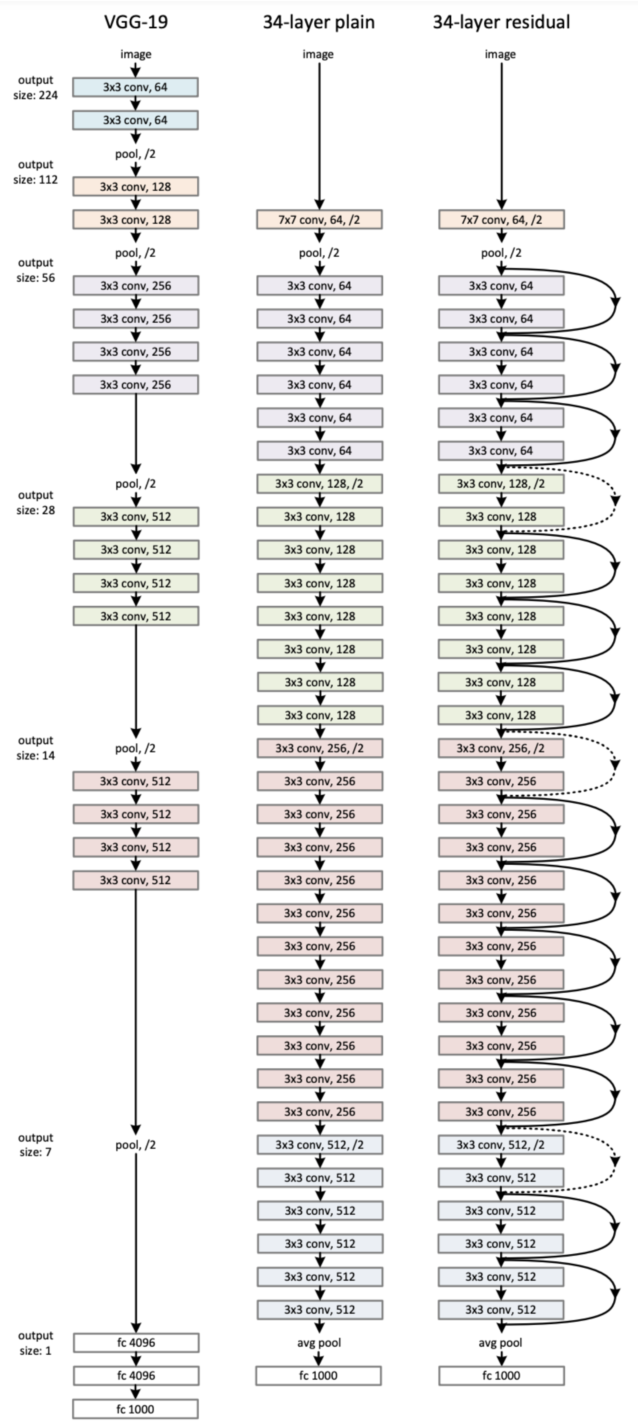
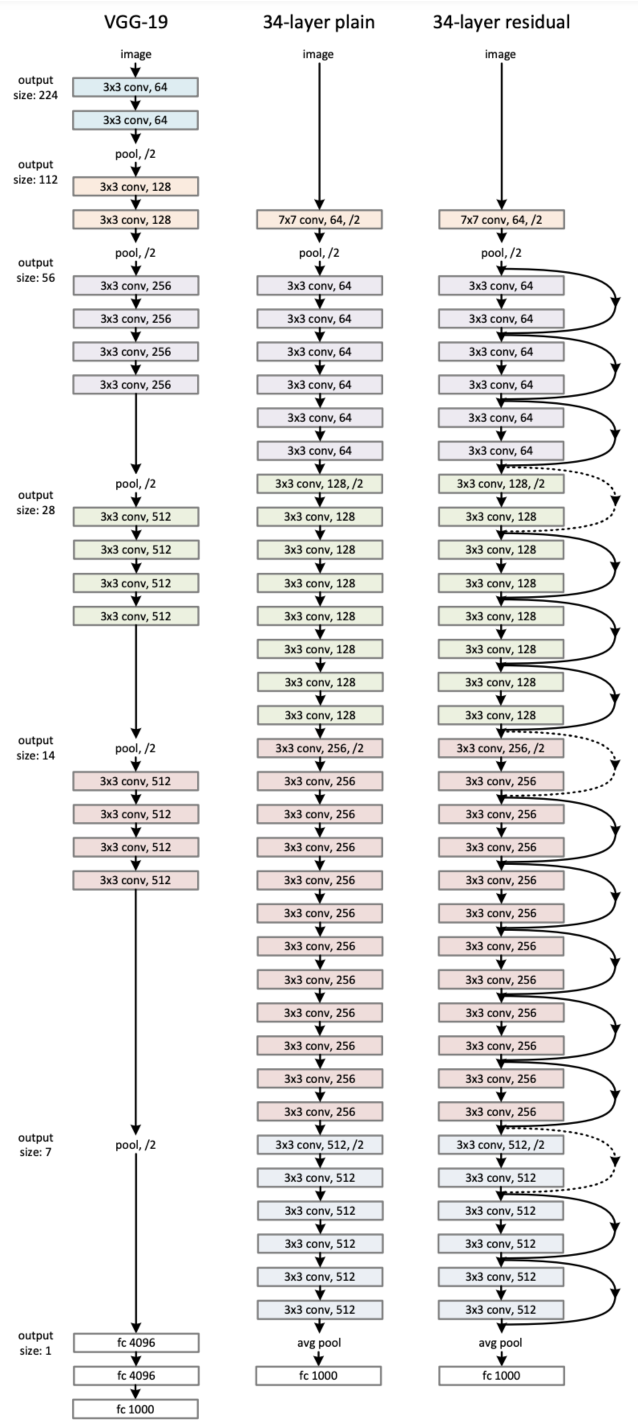


Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

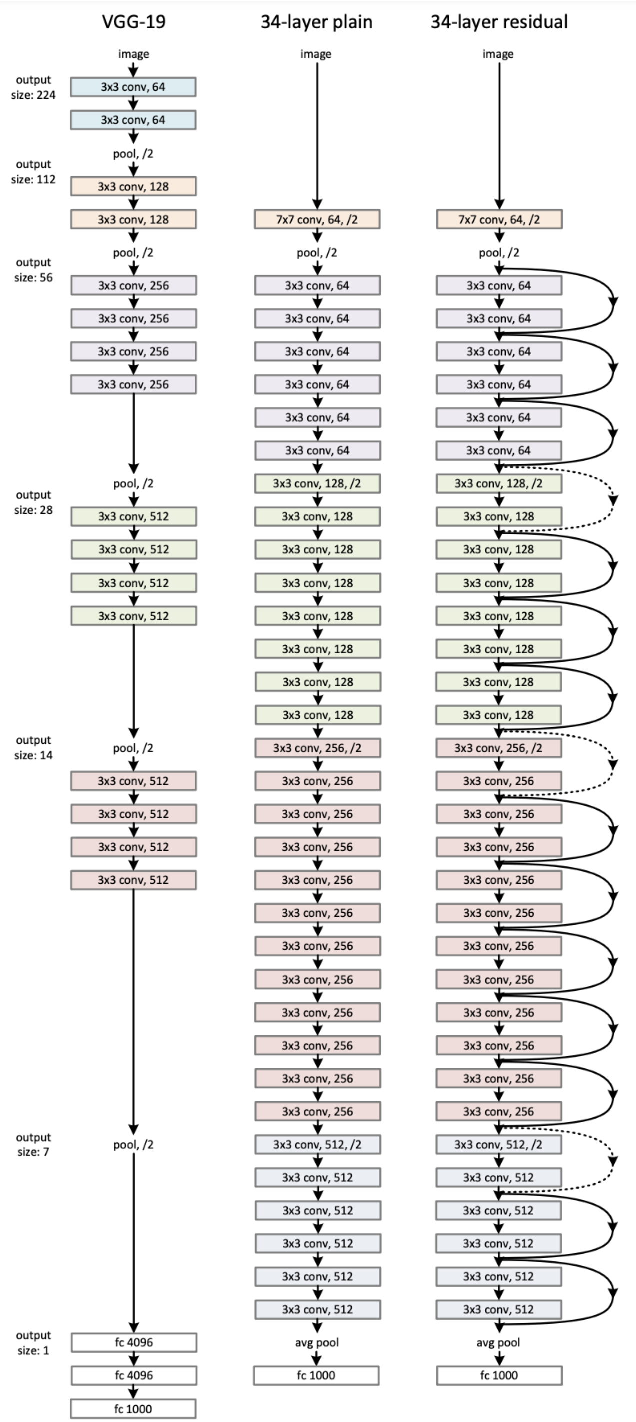
ResNet



floating point operations per second

Figure 3. Example network architectures for ImageNet. **Left**: the $\boxed{\text{VGG-19 model [41] (19.6 billion FLOPs)}}$ as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

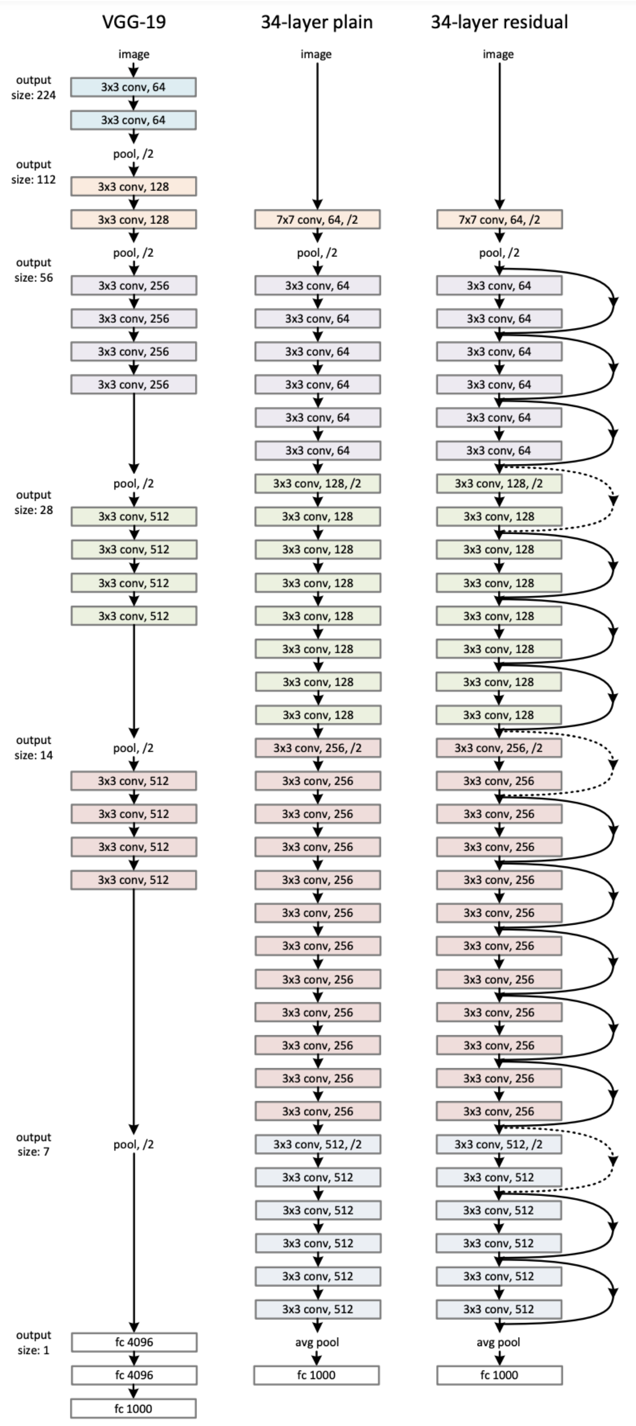
ResNet



Measure: how complicated the model is

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

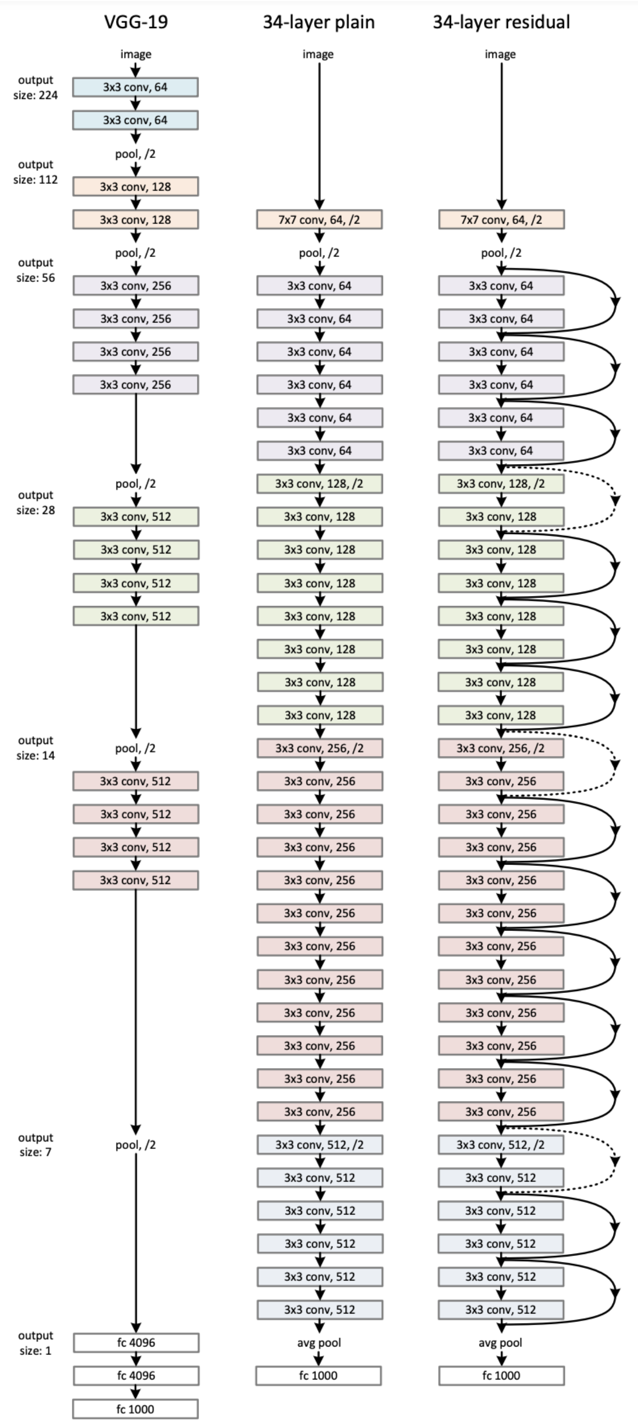
ResNet



Measure: how complicated the model is

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

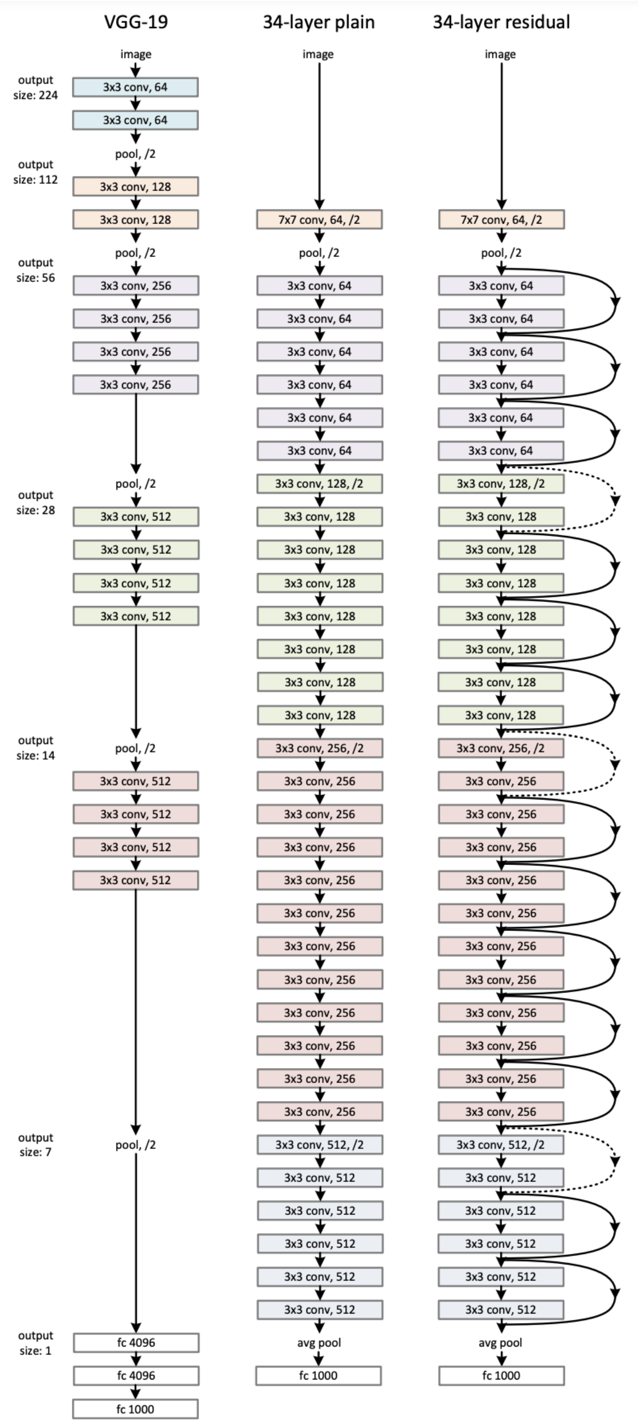
ResNet



Measure: how complicated the model is

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

ResNet

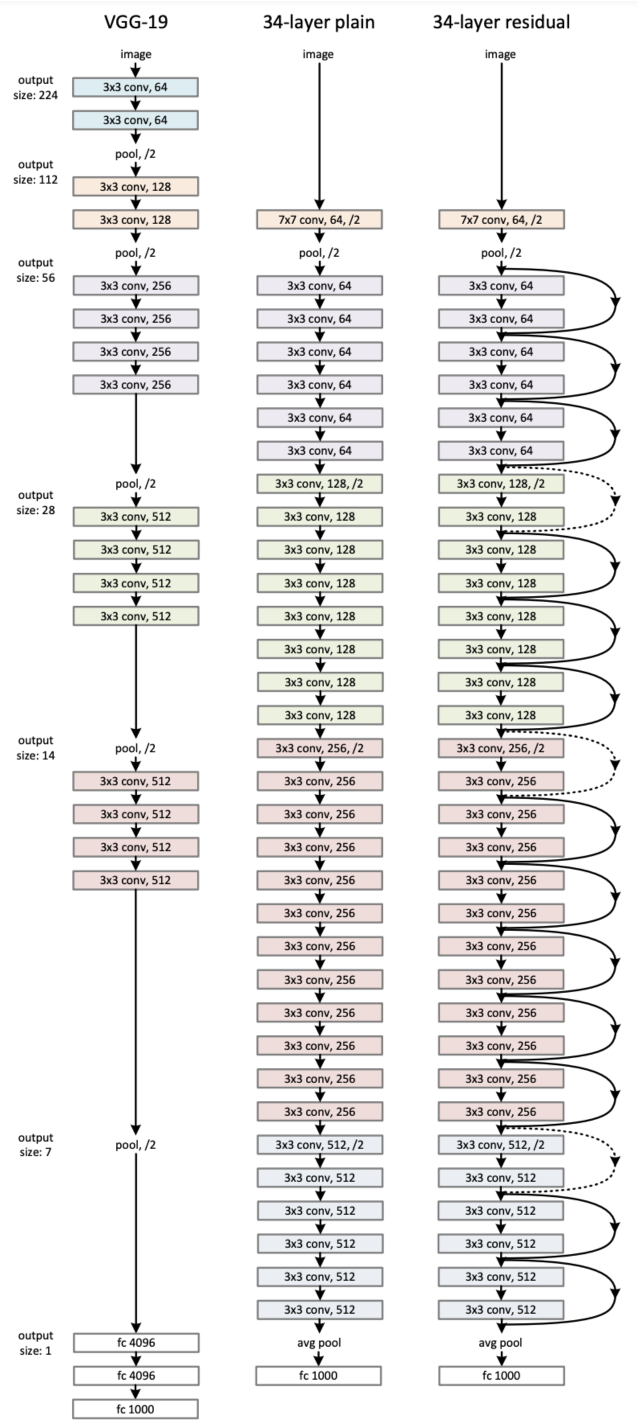


Measure: how complicated the model is

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Q: VGG-19 has much more FLOPS than 34-layer plain network and 34-layer ResNet?

ResNet



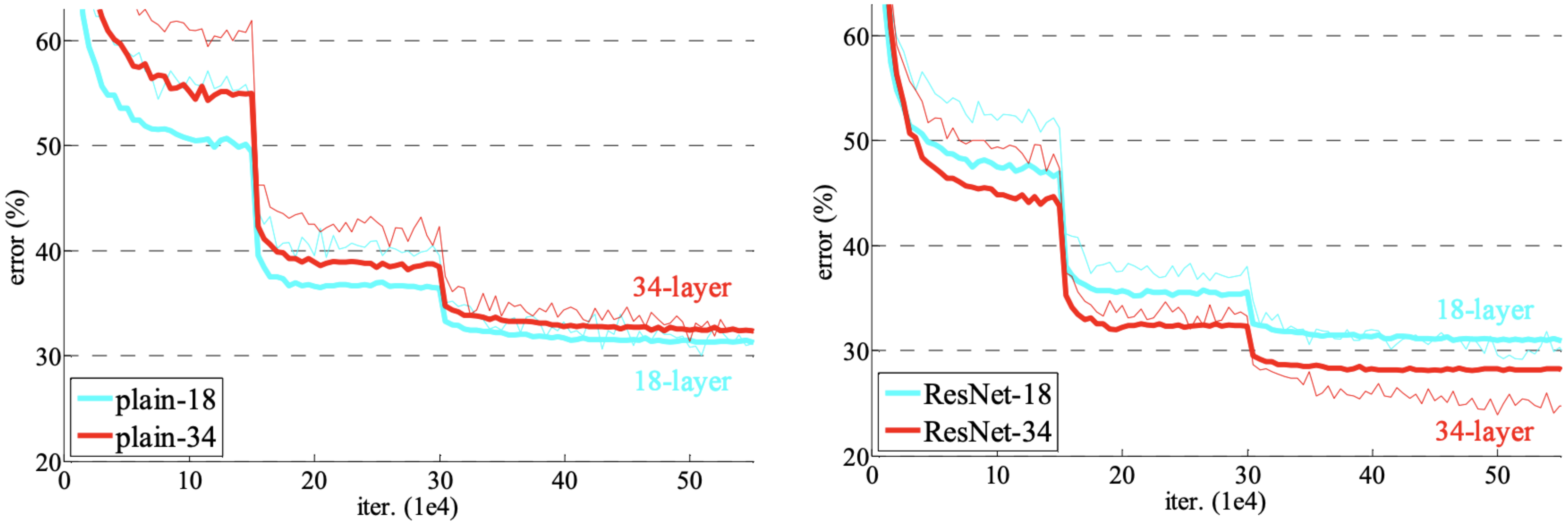
Measure: how complicated the model is

Figure 3. Example network architectures for ImageNet. **Left**: the VGG-19 model [41] (19.6 billion FLOPs) as a reference. **Middle**: a plain network with 34 parameter layers (3.6 billion FLOPs). **Right**: a residual network with 34 parameter layers (3.6 billion FLOPs). The dotted shortcuts increase dimensions. **Table 1** shows more details and other variants.

Q: VGG-19 has much more FLOPS than 34-layer plain network and 34-layer ResNet?

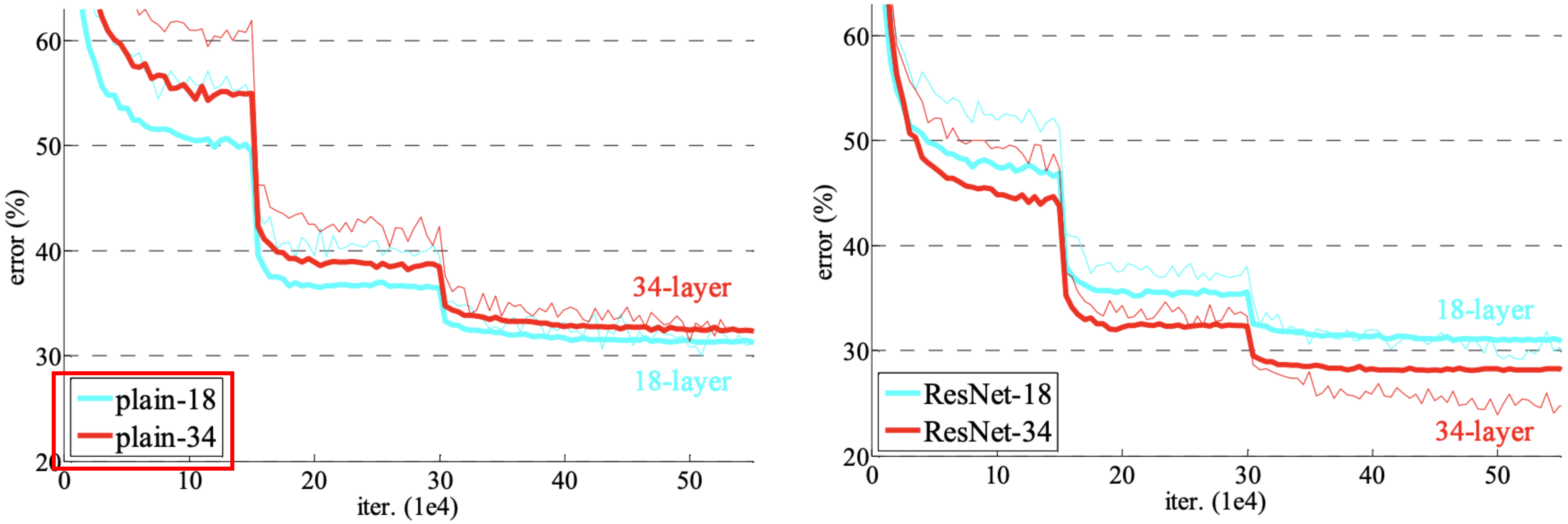
[Reading material]

ResNet



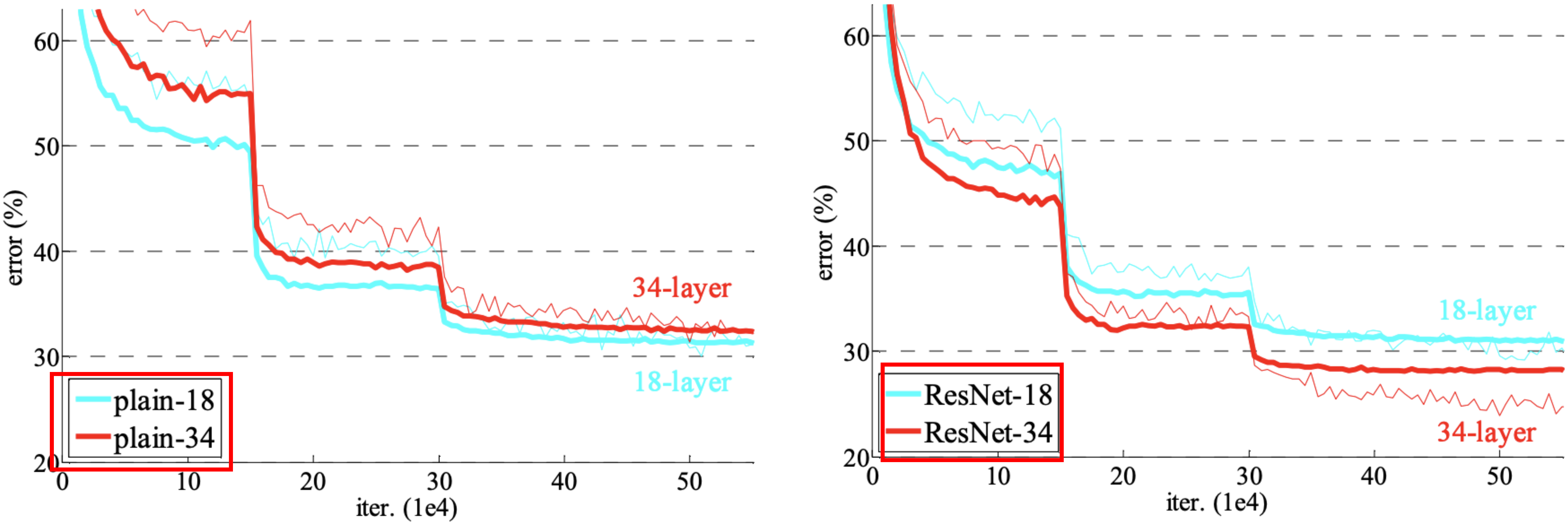
60
50
40
30
20
error (%)
0
10
20
30
40
50
iter. (1e4)
34-layer
18-layer
plain-18
plain-34
60
50
40
30
20
error (%)
0
10
20
30
40
50
iter. (1e4)
18-layer
34-layer
ResNet-18
ResNet-34

ResNet



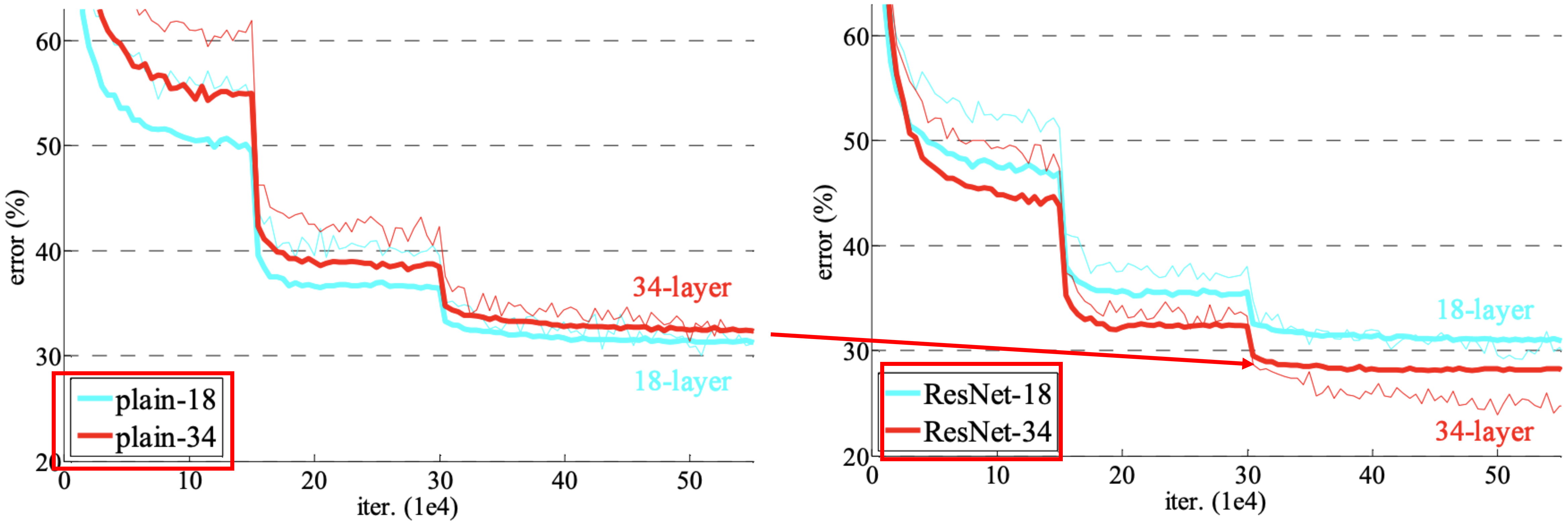
60
50
40
30
20
error (%)
0
10
20
30
40
50
iter. (1e4)
34-layer
18-layer
plain-18
plain-34
60
50
40
30
20
error (%)
0
10
20
30
40
50
iter. (1e4)
18-layer
34-layer
ResNet-18
ResNet-34

ResNet



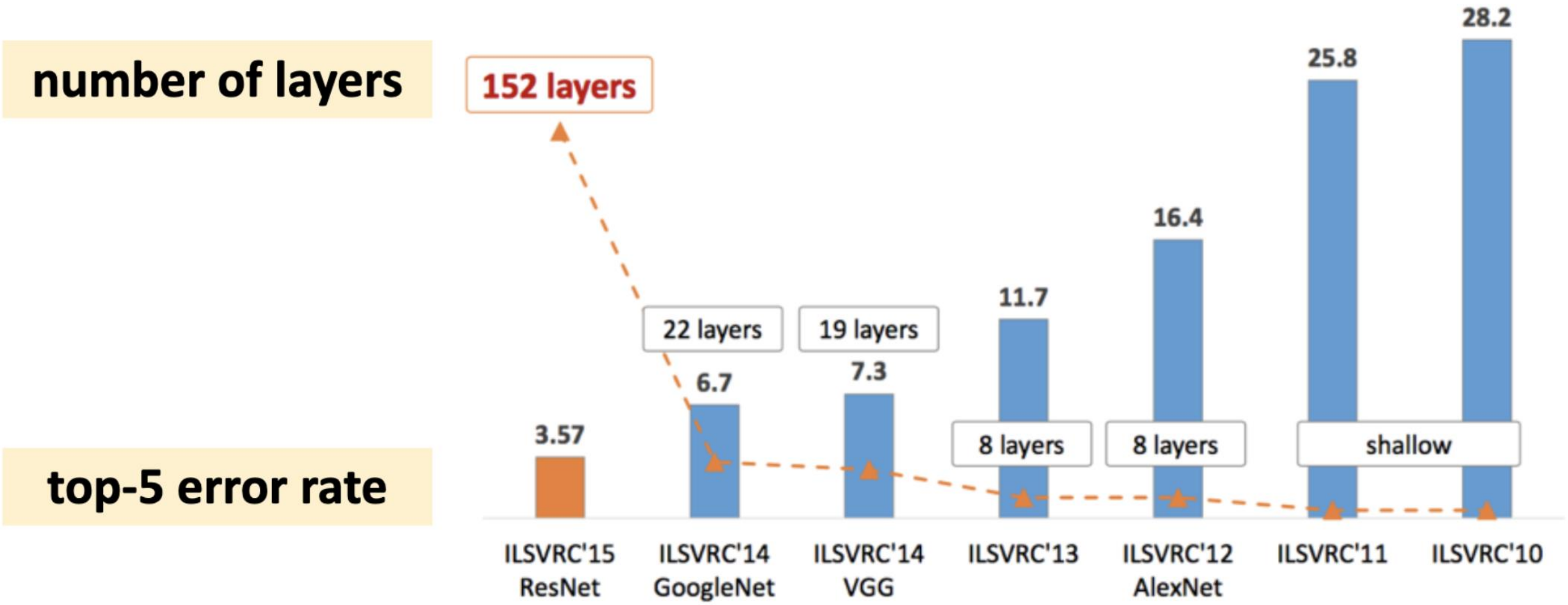
error (%)
60
50
40
30
20
0
10
20
30
40
50
iter. (1e4)
34-layer
18-layer
plain-18
plain-34
error (%)
60
50
40
30
20
0
10
20
30
40
50
iter. (1e4)
18-layer
34-layer
ResNet-18
ResNet-34

ResNet

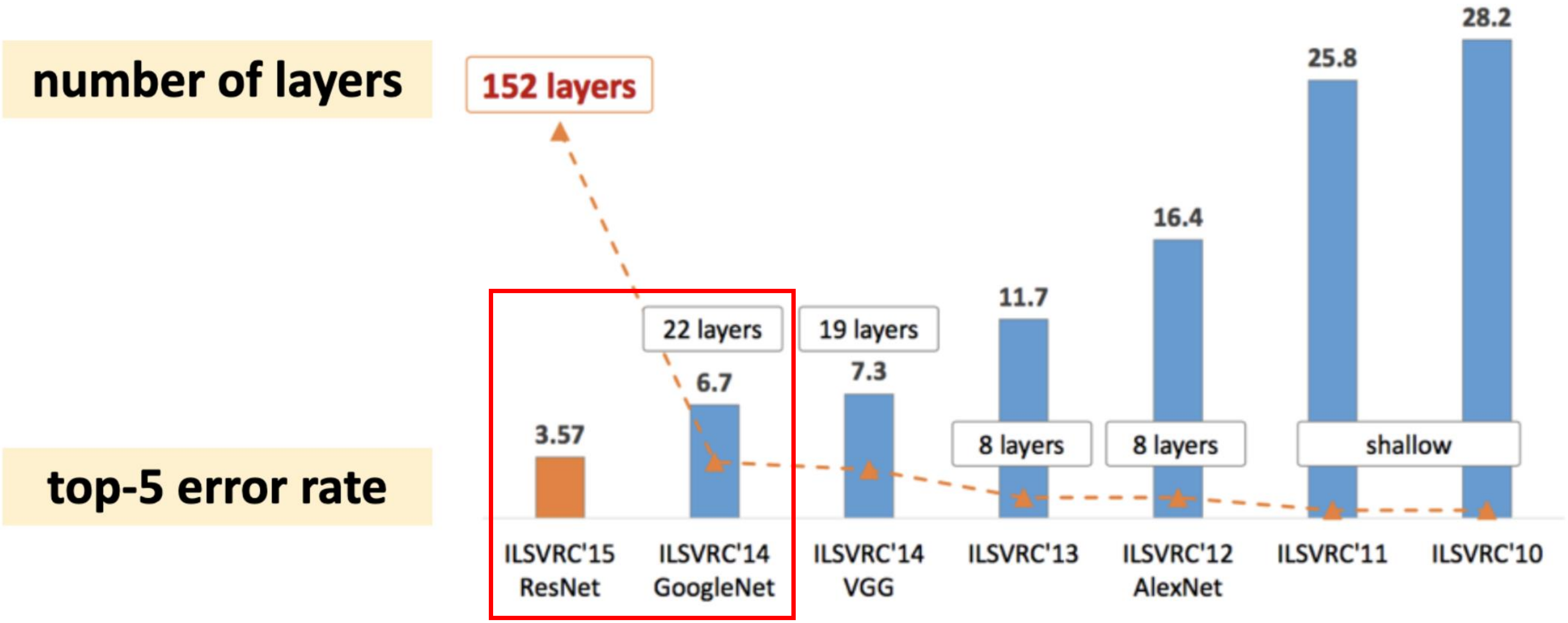


60
50
40
30
20
error (%)
0
10
20
30
40
50
iter. (1e4)
34-layer
18-layer
plain-18
plain-34
60
50
40
30
20
error (%)
0
10
20
30
40
50
iter. (1e4)
18-layer
34-layer
ResNet-18
ResNet-34

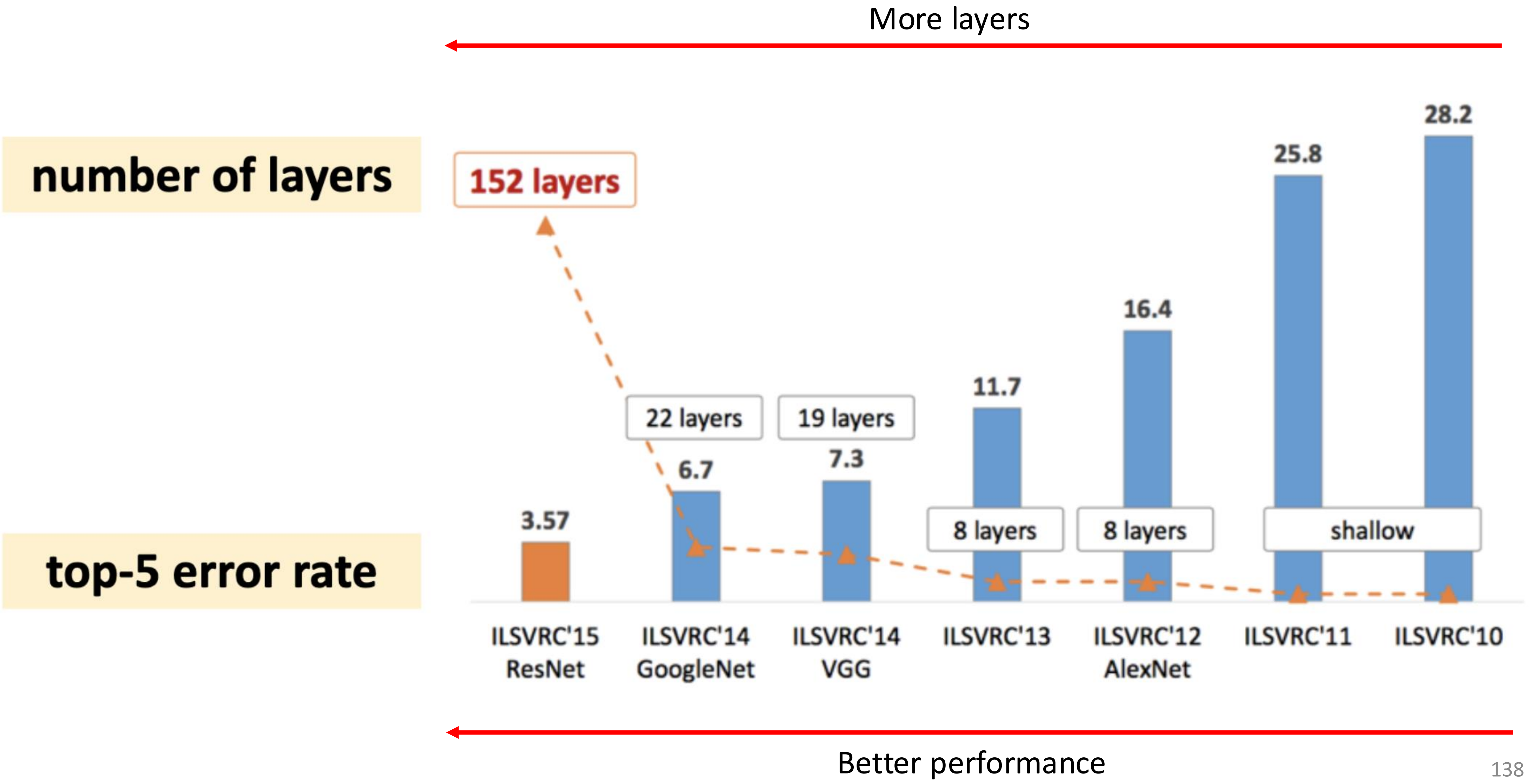
ImageNet competition winners



ImageNet competition winners



ImageNet competition winners



References

- [LetNet-5] LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.
- [AlexNet] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems* 25 (2012): 1097-1105.
- [VGG] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014). Arxiv at https://arxiv.org/pdf/1409.1556.pdf

References

- [Inception] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9. 2015. ArXiv at https://arxiv.org/pdf/1409.4842.pdf (*Section 4 and 5)*
- [ResNet] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016. ArXiv at https://arxiv.org/pdf/1512.03385.pdf (*Section 3.1, 3.2 and 3.3)*