CS540 Introduction to Artificial Intelligence

Convolutional Neural Networks (II)

University of Wisconsin-Madison
Spring 2026 Sections 1 & 2

Announcements

- **Homework 7 released**
- Course evaluation due on **Friday March 13th**
- Class roadmap and schedule:

Machine Learning: Deep Learning II
Machine Learning: Deep Learning III
Machine Learning: Deep Learning IV

Deep Learning

Midterm Information

- **Time: March 24th 5:45-7:15 PM**
- **Location (by section **):**
  - Section 001 (Tuesday/Thursday 11-12:15PM): 6210 Social Sciences Bldg
  - Section002 (Tuesday/Thursday 2:30-3:45PM): B10 Ingraham Hall
  - Students with McBurney accommodations should have received an email with additional information.
  - Students who cannot take the exam on the specified time should contact their instructor if they have not done it yet.
- **Topics: Topics covered up to and including Week 9**
- **Exclusion List (questions regarding the following topics will NOT appear on the midterm):**
  - **Logic (covered in sections 1 and 2)**
  - **SVM + Kernel Trick (covered in section 3)**
- **Format**: MCQ
- **Cheat sheet**: a handwritten single piece of paper, front and back
- **Calculator**: optional, if it doesn’t have an Internet connection
- **Bring**: your WISC ID, pencil (No 2 or softer), your 1-sheet notes.
- **Past exam questions**: on Canvas →Files → Past Exams

Today's goals

- Review (some of) convolutional computations.
  - 2D convolutions, multiple input channels, pooling.
- Understand how convolutions are used as layers in a (deep) neural network.
- Build intuition for output of convolutional layers.
- Overview the evolution of deeper convolutional networks

How to classify

Cats vs. dogs?

Dual

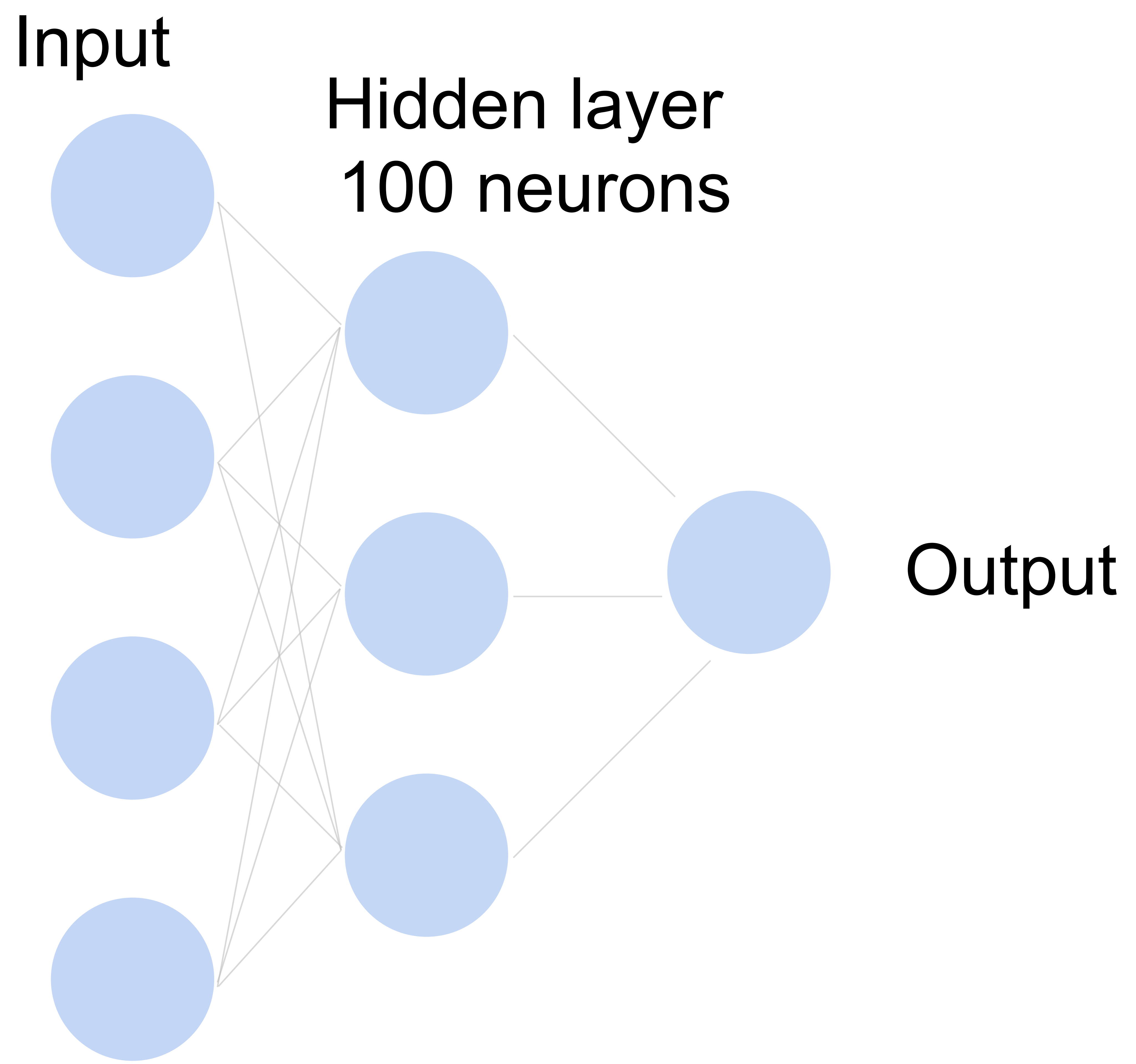
12MP

wide-angle and
telephoto cameras

36M floats in a RGB image!

Fully Connected Networks

Cats vs. dogs?



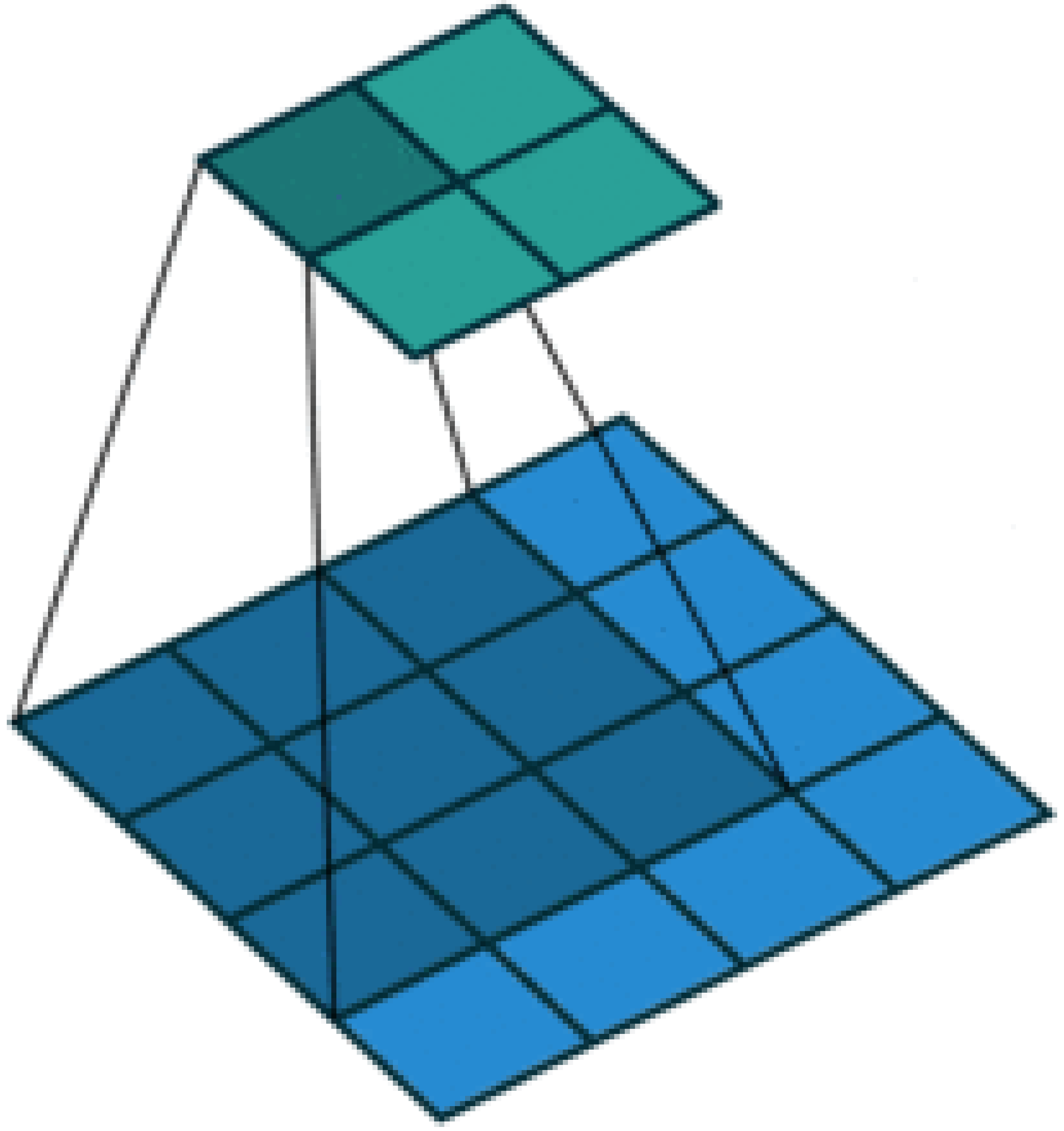
36M elements x 100 = **3.6B** parameters!

Review: 2-D Convolution

Input Kernel Output

$$\begin{bmatrix}0&1&2\\3&4&5\\6&7&8\end{bmatrix} * \begin{bmatrix}0&1\\2&3\end{bmatrix} = \begin{bmatrix}19&25\\37&43\end{bmatrix}$$

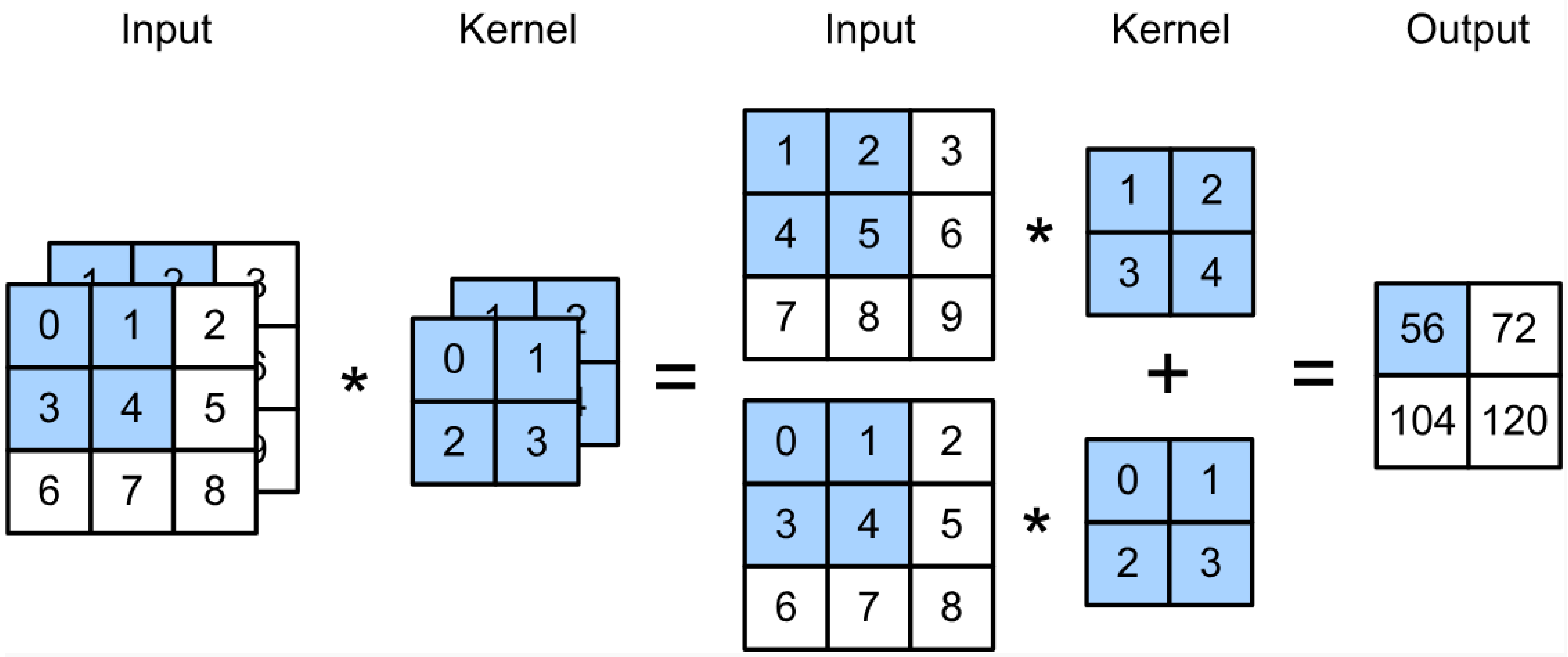
$$0\times0+1\times1+3\times2+4\times3=19,$$
$$1\times0+2\times1+4\times2+5\times3=25,$$
$$3\times0+4\times1+6\times2+7\times3=37,$$
$$4\times0+5\times1+7\times2+8\times3=43.$$



(vdumoulin@ Github)

Review: Multiple Input Channels

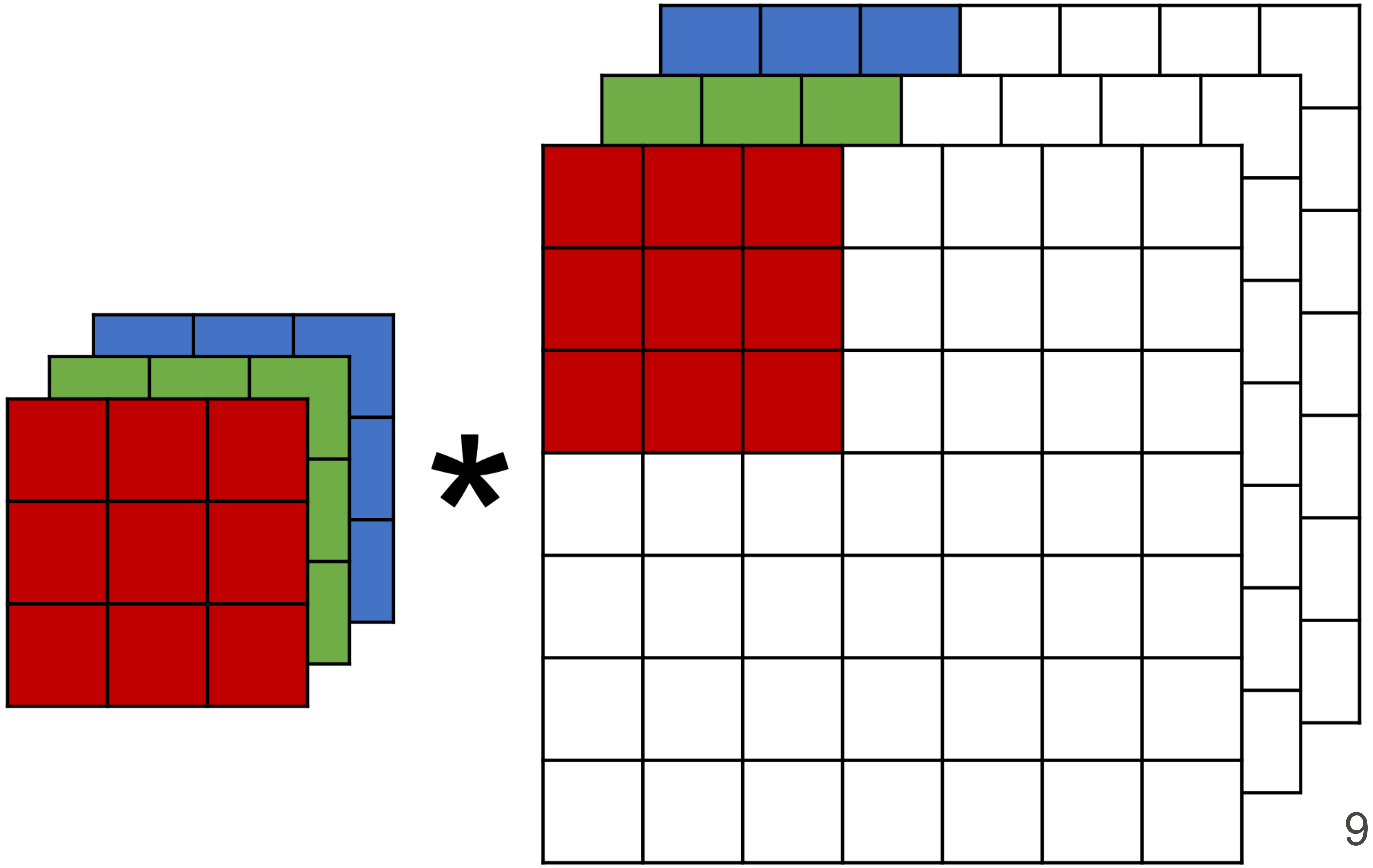
- Have a kernel matrix for each channel, and then sum results over channels



$$
\begin{gathered}
(1 \times 1 + 2 \times 2 + 4 \times 3 + 5 \times 4) \\
+(0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3) \\
= 56
\end{gathered}
$$

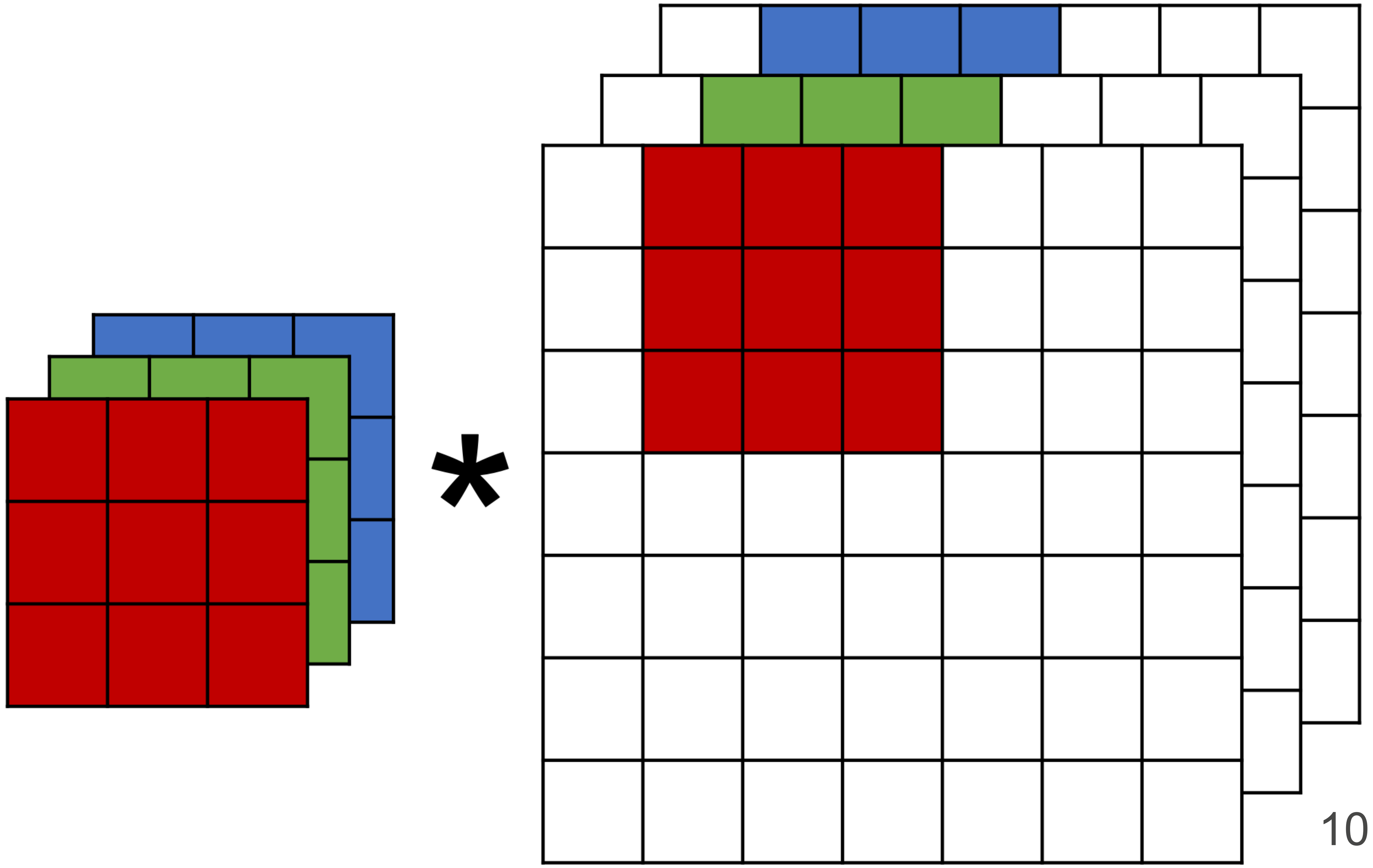
Review: Multiple Input Channels

- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels



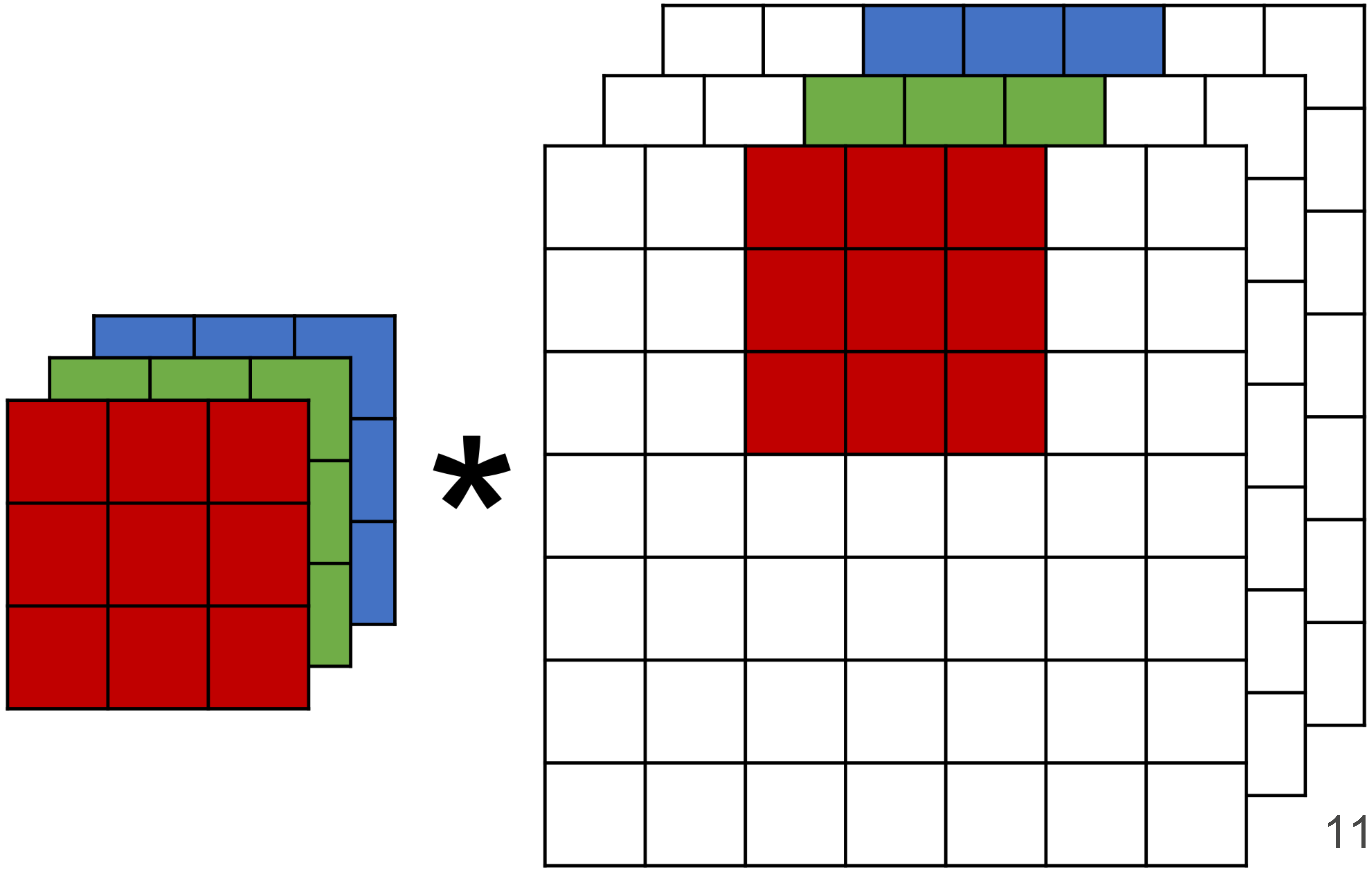
Review: Multiple Input Channels

- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels



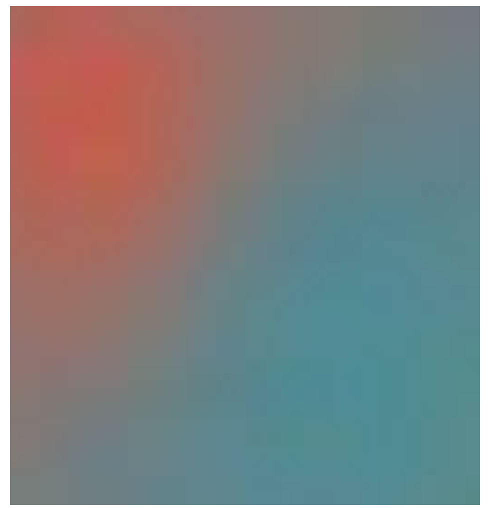
Review: Multiple Input Channels

- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels

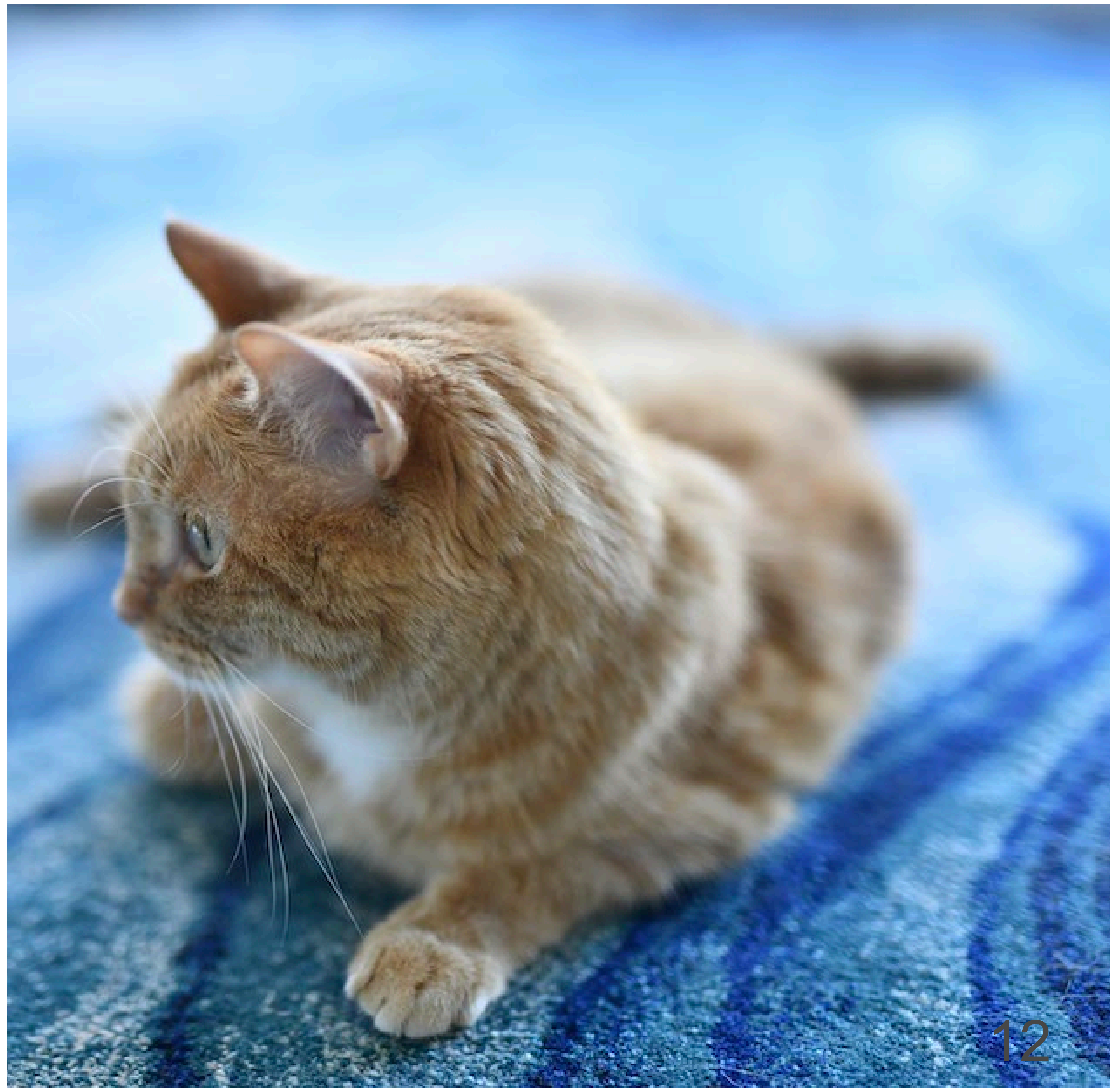


Multiple Input Channels

- Input and kernel can be 3D, e.g. RGB image has 3 channels
- Also call each 3D kernel a “**filter**”, which produces only **one** output channel (due to summation over channels)



*

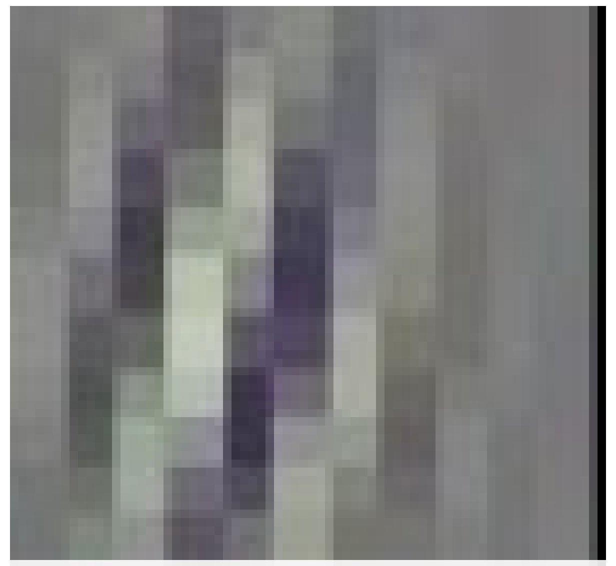


One filter
(3 channels)

RGB (3 input channels)

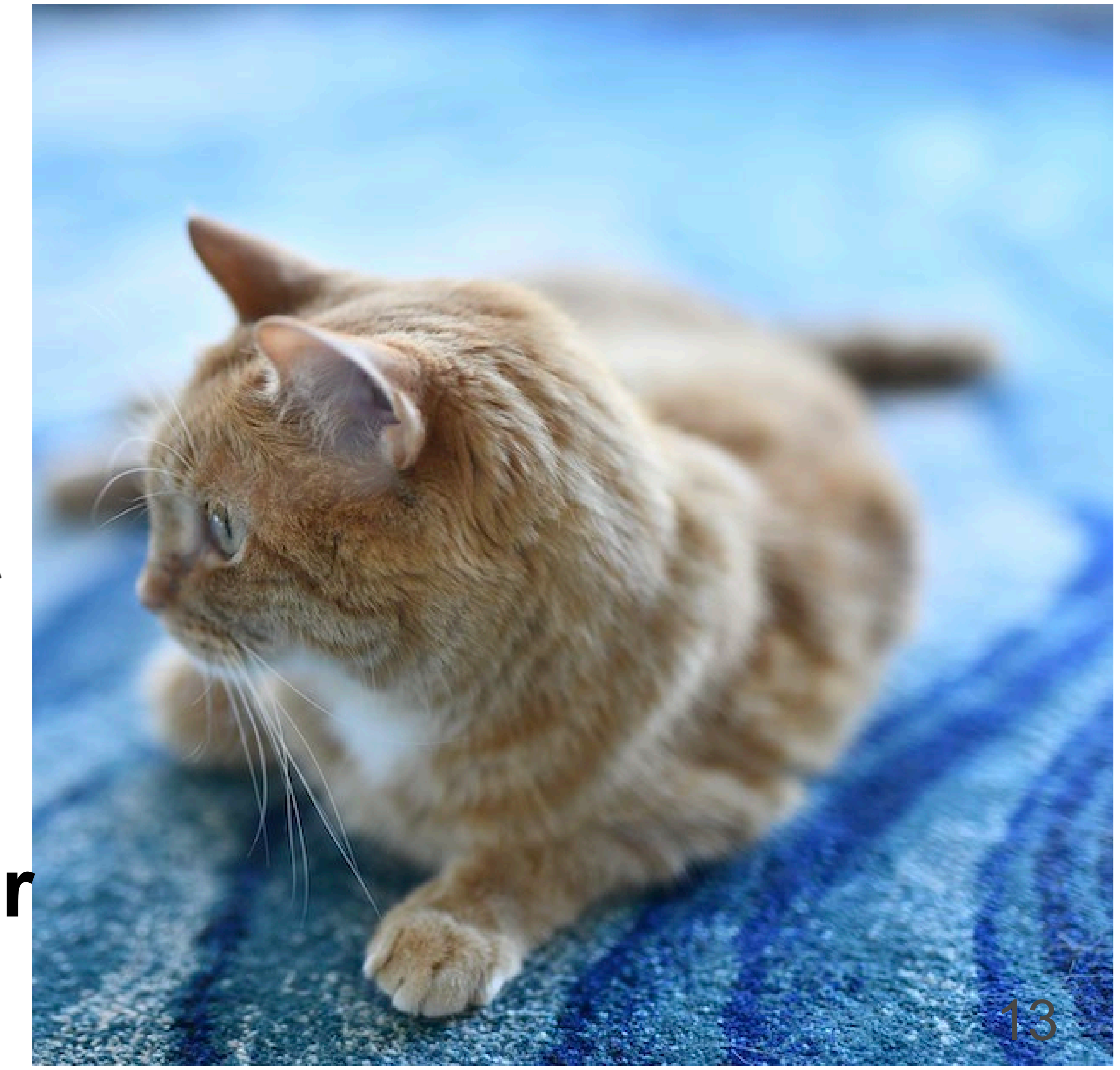
Multiple filters (in one layer)

- Apply multiple filters on the input
- Each filter may learn different features about the input
- Each filter (3D kernel) produces one output channel



A different filter

*



RGB (3 input channels)

Output shape

Kernel/filter size

$$\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$$

Input size **Pad** **Stride**

Q4. Suppose we want to perform convolution on a RGB image of size 224 x 224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. The convolution layer has bias parameters. Which is a reasonable estimate of the total number of learnable parameters?

A. 64 x 222 x 222

B. 64 x 3 x 3 x 222

C. 3 x 3 x 3 x 64

D. (3 x 3 x 3 + 1) x 64

Q4. Suppose we want to perform convolution on a RGB image of size 224 x 224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. The convolution layer has bias parameters. Which is a reasonable estimate of the total number of learnable parameters?

A. 64 x 222 x 222

B. 64 x 3 x 3 x 222

C. 3 x 3 x 3 x 64

D. (3 x 3 x 3 + 1) x 64

Q4. Suppose we want to perform convolution on a RGB image of size 224 x 224 (no padding) with 64 kernels, each with height 3 and width 3. Stride = 1. The convolution layer has bias parameters. Which is a reasonable estimate of the total number of learnable parameters?

A. 64 x 222 x 222

B. 64 x 3 x 3 x 222

C. 3 x 3 x 3 x 64

D. (3 x 3 x 3 + 1) x 64

Each kernel is 3D kernel across 3 input channels, so has 3x3x3 parameters. Each kernel has 1 bias parameter. So in total (3x3x3+1)x64

Consider a convolution layer with 16 filters. Each filter has a size of 11x11x3, a stride of 2x2. Given an input image of size 22x22x3, if we don't allow a filter to fall outside of the input, what is the output size?

- 11x11x16
- 6x6x16
- 7x7x16
- 5x5x16

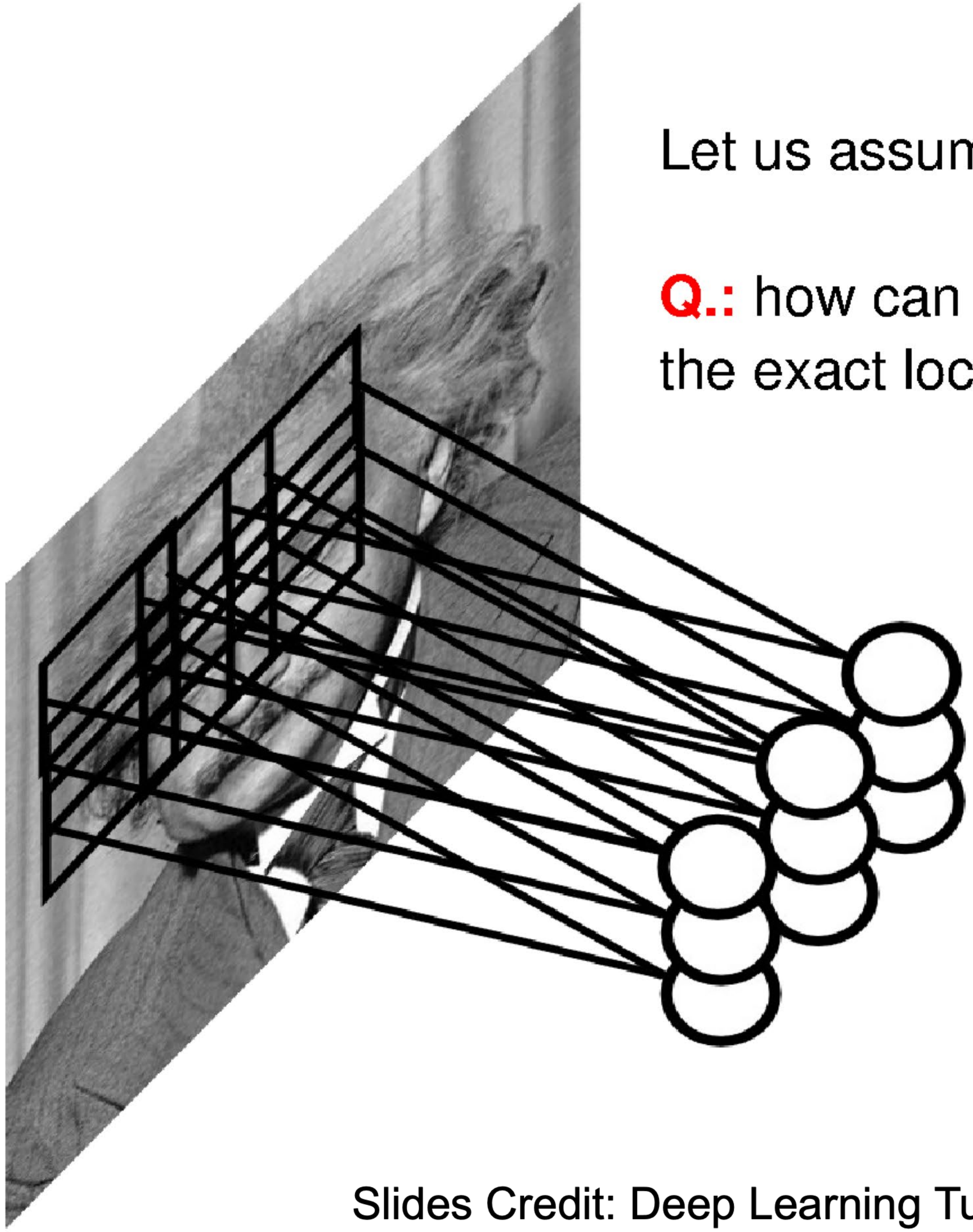
Consider a convolution layer with 16 filters. Each filter has a size of 11x11x3, a stride of 2x2. Given an input image of size 22x22x3, if we don't allow a filter to fall outside of the input, what is the output size?

- 11x11x16
- $\boxed{\text{6x6x16}}$
- 7x7x16
- 5x5x16

$$\lfloor (n_h - k_h + p_h + s_h)/s_h \rfloor \times \lfloor (n_w - k_w + p_w + s_w)/s_w \rfloor$$

Pooling Layer

Pooling

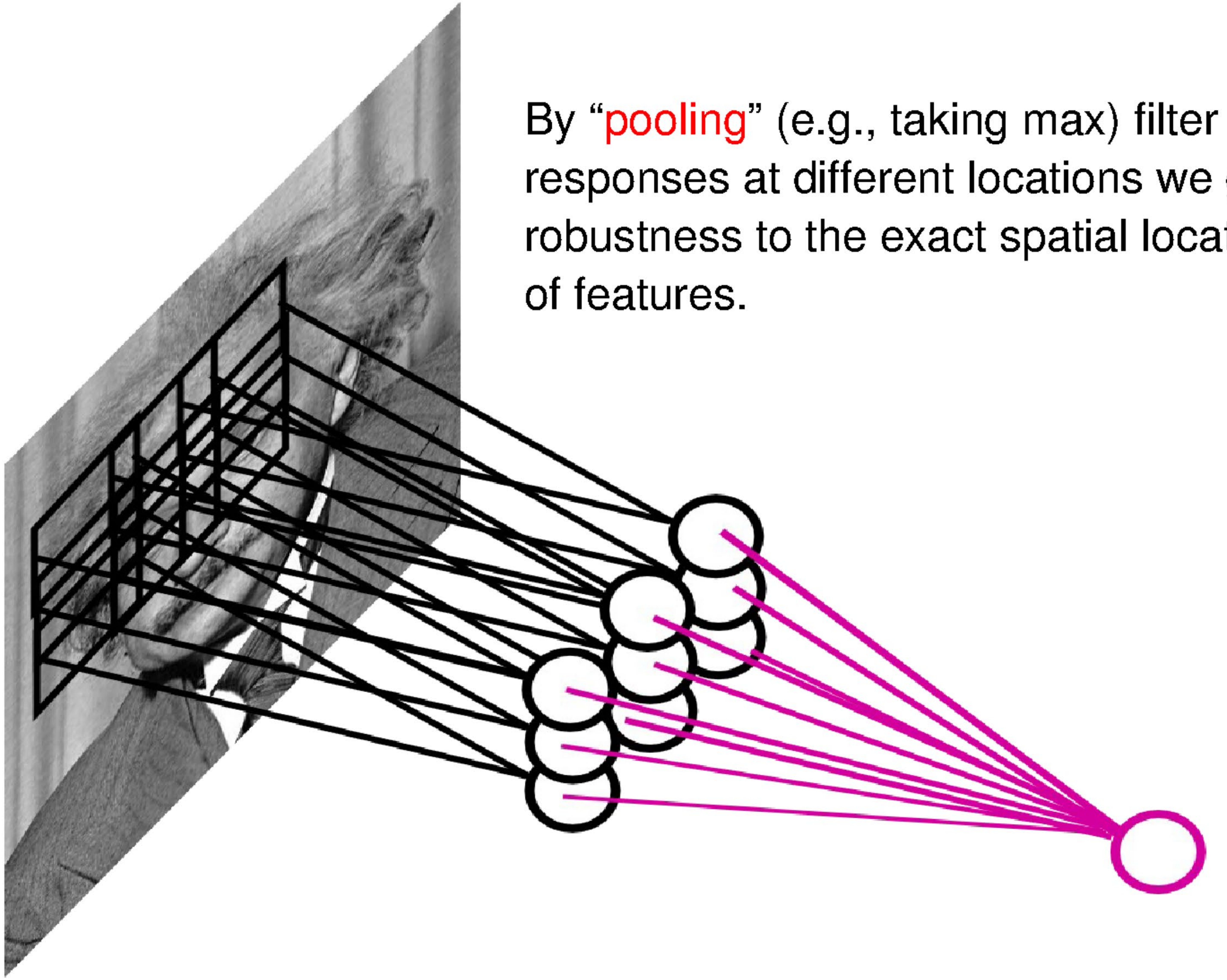


Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?

Slides Credit: Deep Learning Tutorial by Marc’Aurelio Ranzato

Pooling

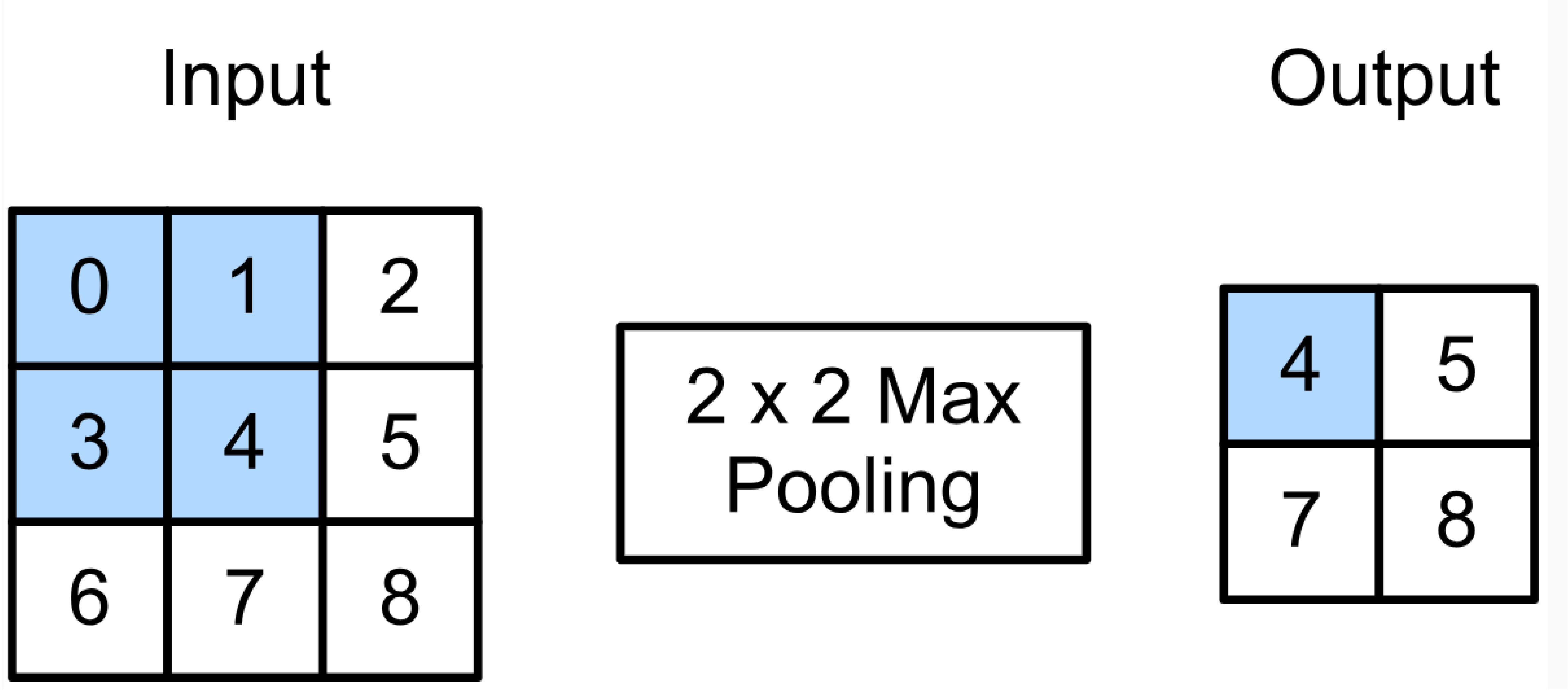


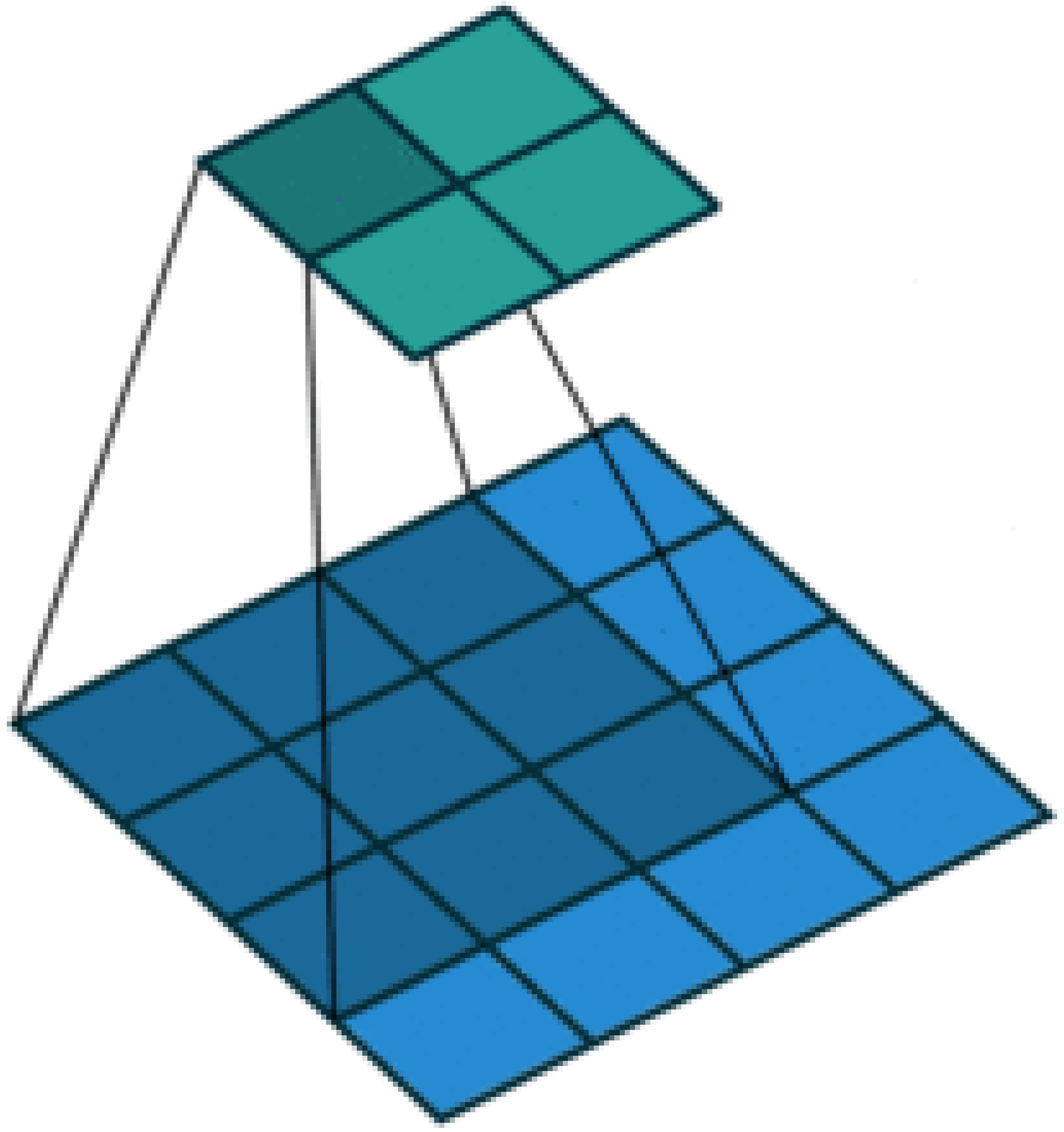
By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

Slides Credit: Deep Learning Tutorial by Marc'Aurelio Ranzato

2-D Max Pooling

- Returns the maximal value in the sliding window



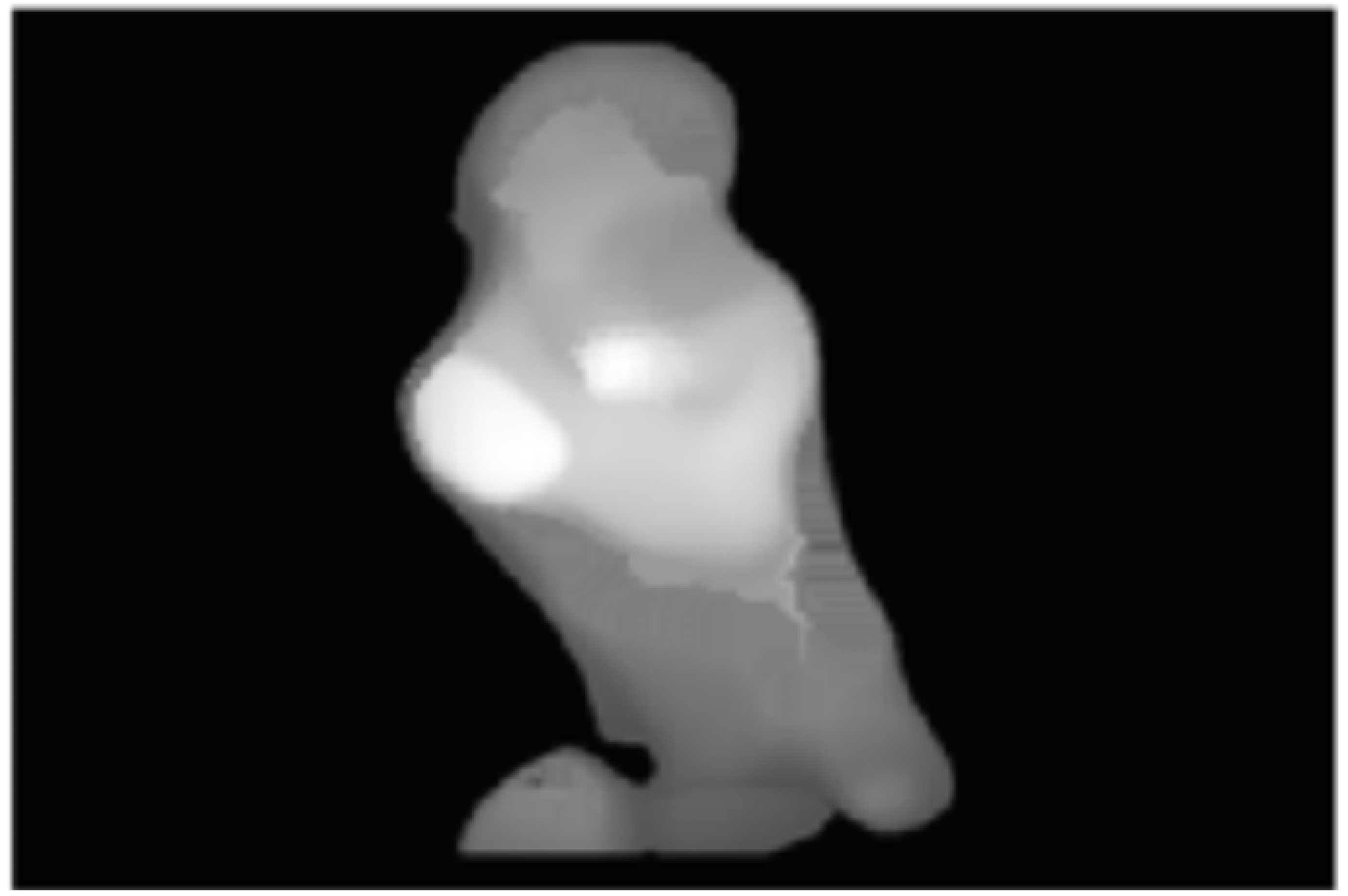


$$max(0,1,3,4) = 4$$

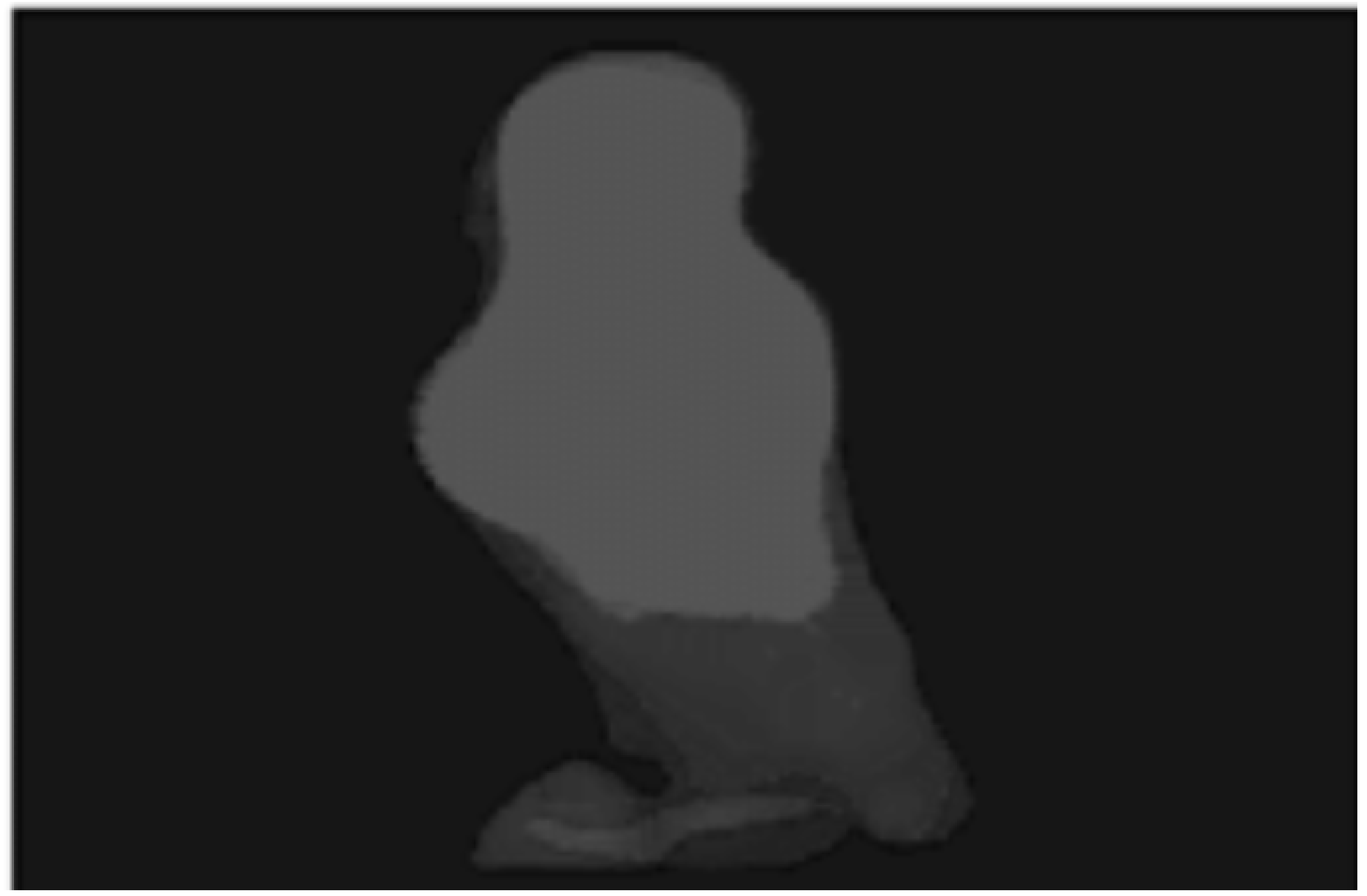
Average Pooling

- Max pooling: the strongest pattern signal in a window
- Average pooling: replace max with mean in max pooling
  - The average signal strength in a window

Max pooling



Average pooling



Q2. Suppose we want to perform 2x2 average pooling on the following single channel feature map of size 4x4 (no padding), and stride = 2. What is the output?

12	**20**	**30**	**0**
20	12	2	0
0	70	5	2
8	2	90	3

A.

20	**30**
70	90

B.

16	**8**
20	25

20	**30**
20	25

C.

12	**2**
70	5

Q2. Suppose we want to perform 2x2 average pooling on the following single channel feature map of size 4x4 (no padding), and stride = 2. What is the output?

12	**20**	**30**	**0**
20	12	2	0
0	70	5	2
8	2	90	3

A.

20	**30**
70	90

B.

16	**8**
20	25

20	**30**
20	25

C.

12	**2**
70	5

Q3. What is the output if we replace average pooling with 2 x 2 max pooling (other settings are the same)?

12	**20**	**30**	**0**
20	12	2	0
0	70	5	2
8	2	90	3

A.

20	**30**
70	90

16	**8**
20	25

B.

20	**30**
20	25

C.

12	**2**
70	5

Q3. What is the output if we replace average pooling with 2 x 2 max pooling (other settings are the same)?

12	**20**	**30**	**0**
20	12	2	0
0	70	5	2
8	2	90	3

A.

20	**30**
70	90

16	**8**
20	25

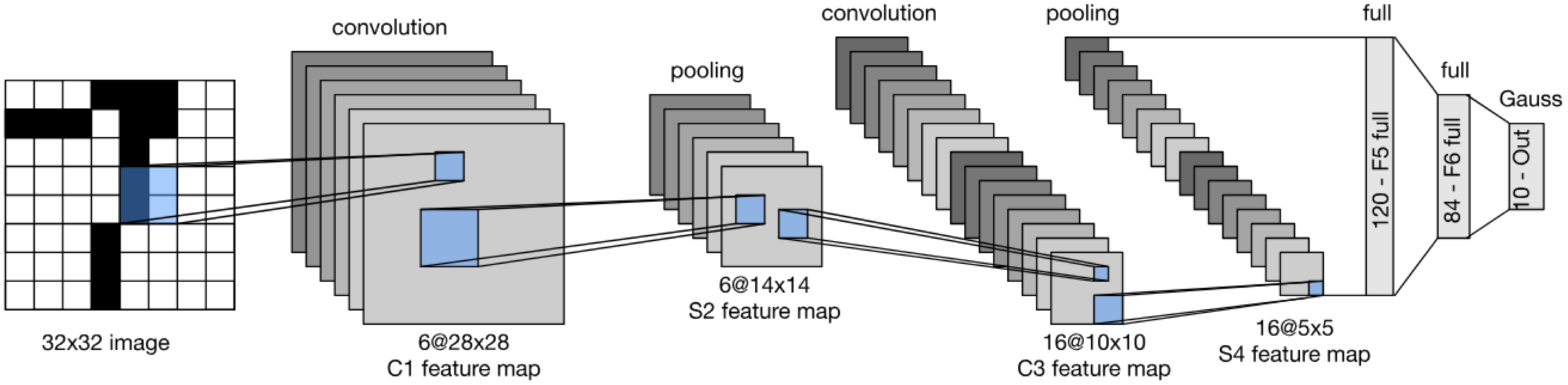
B.

20	**30**
20	25

C.

12	**2**
70	5

Convolutional Neural Network Architecture



Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner

Convolutional Neural Network Intuition

Early layers recognize simple visual features, later layers recognize more complex visual features.

Suppose we want to classify pictures of cats or dogs. How would you do this?

Look for features of cats or dogs in the image and use for decision.

- Example: cats have cat-like faces, dogs have dog-like faces.
- How do you determine what is a “cat-like” face vs a “dog-like” face?

Look for features of “cat-like” faces and “dog-like” faces.

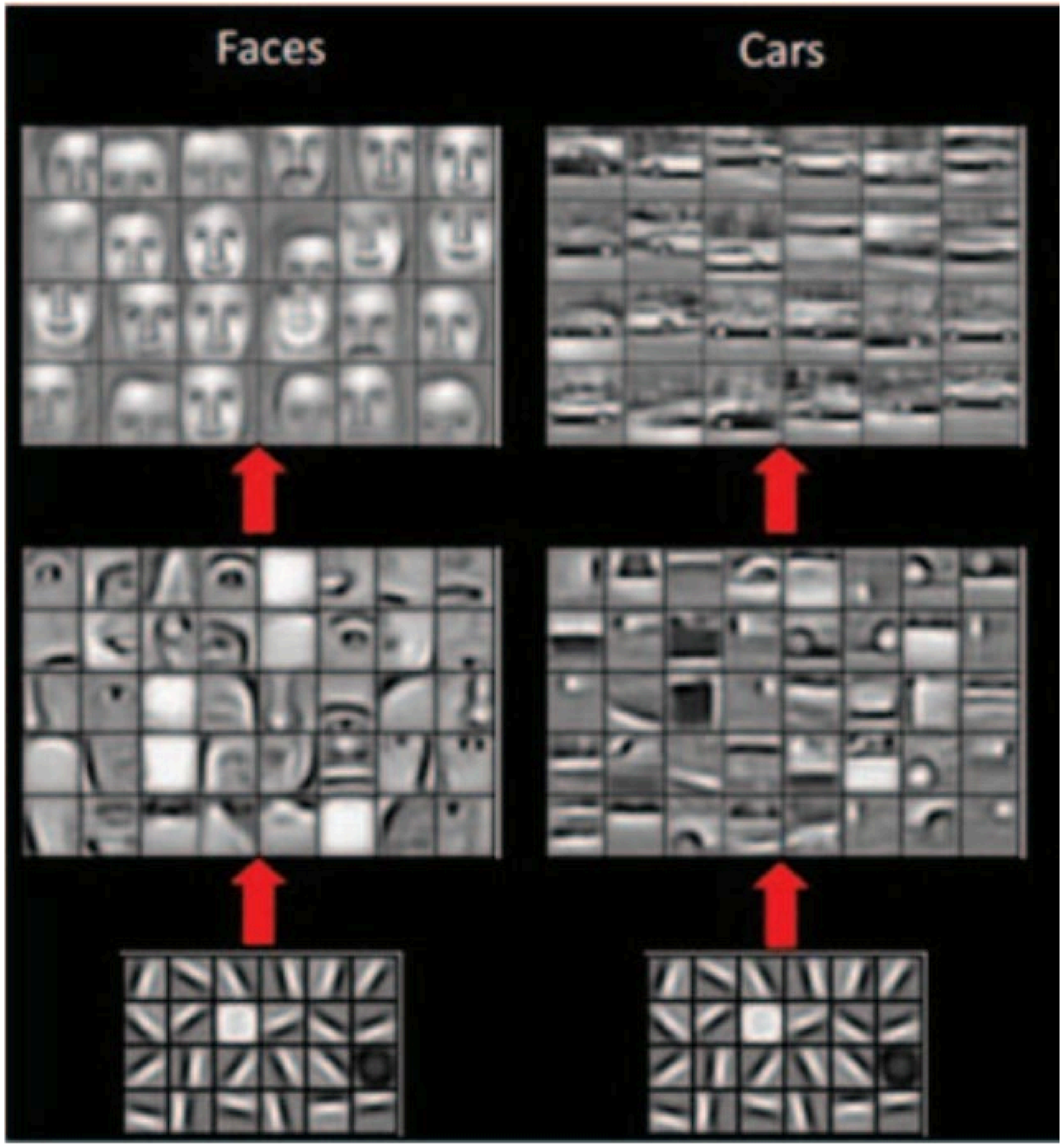
- Example: Dogs have longer snouts.
- How do you determine what is a long snout?

Feature Learning

Later layers recognize complete objects

Middle layers recognize parts of objects

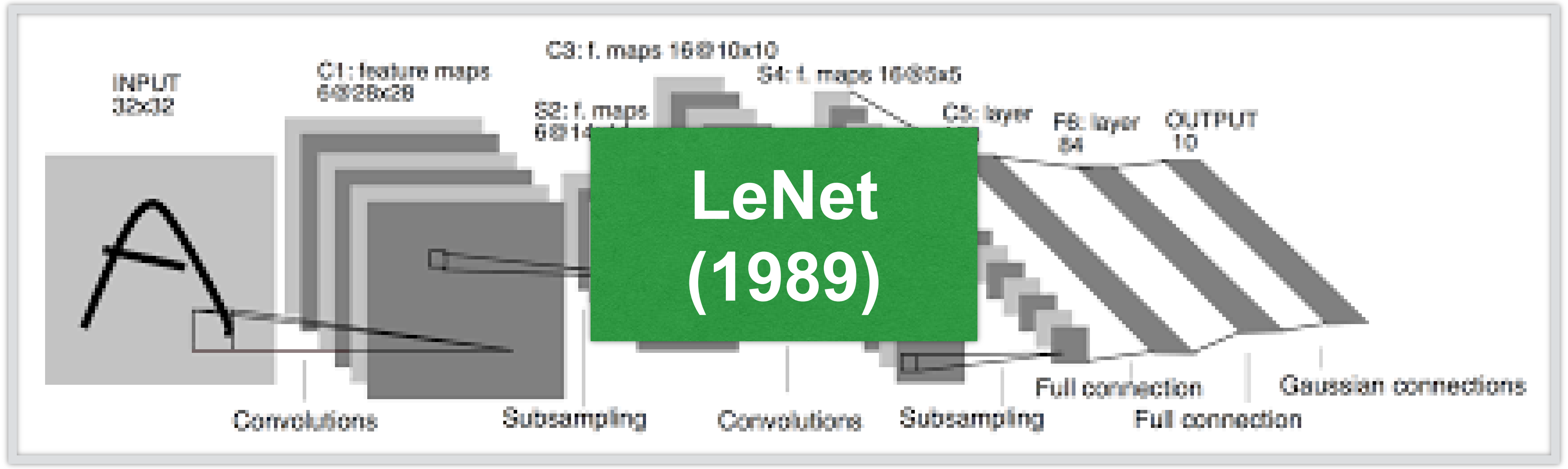
Early layers recognize simple patterns

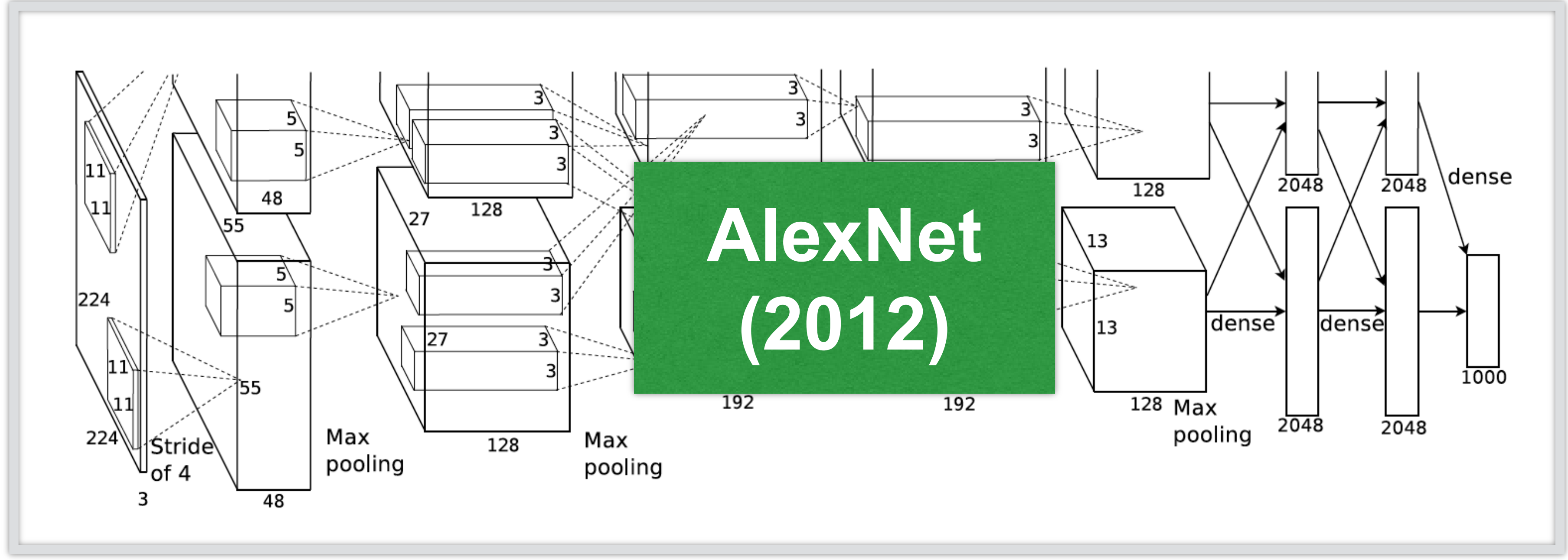


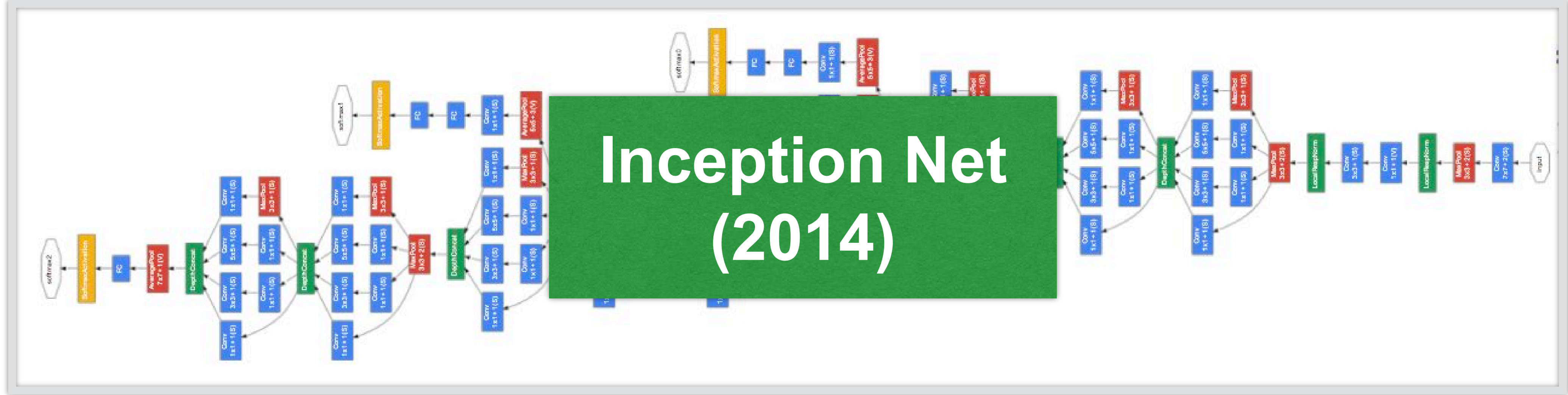
Adaptive Neuron Apoptosis for Accelerating Deep Learning on Large Scale Systems. Seigel et al. 2016.

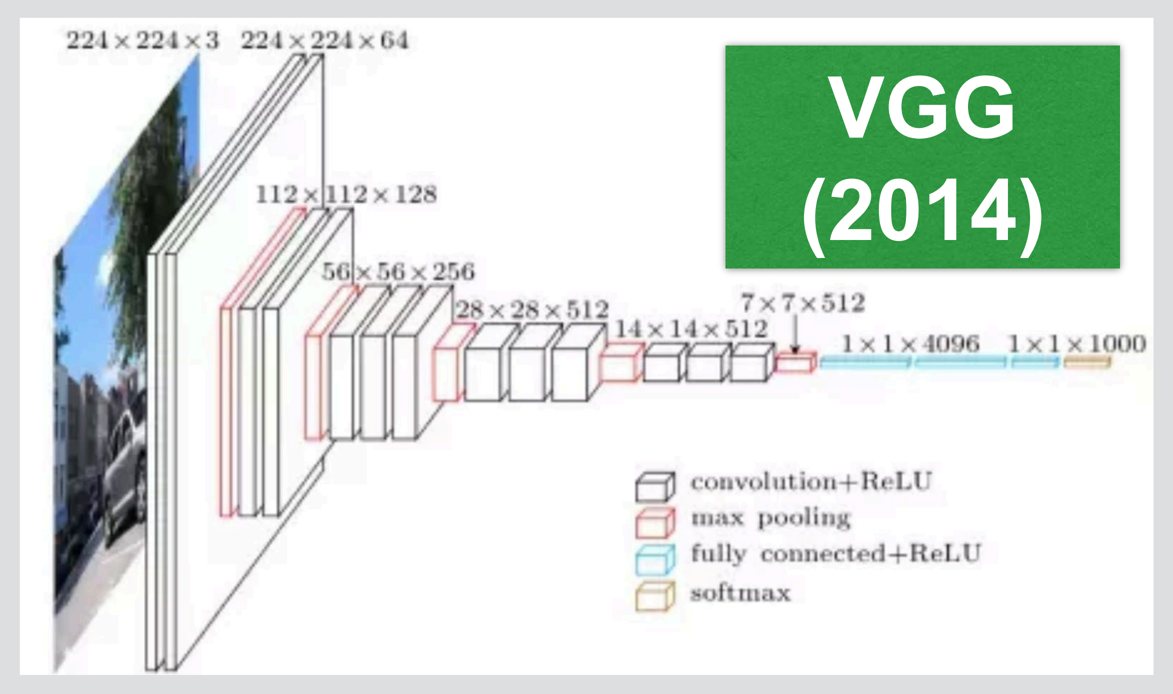
Convolutional Neural Networks Examples

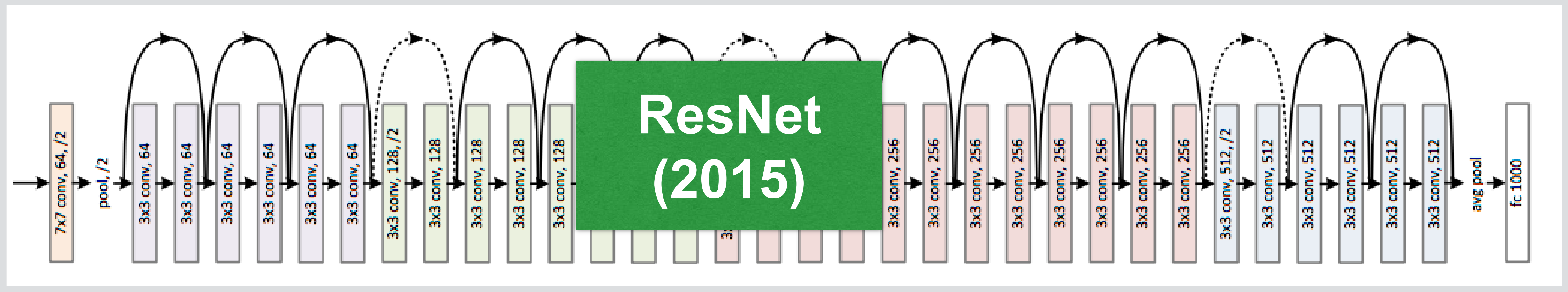
Evolution of neural net architectures

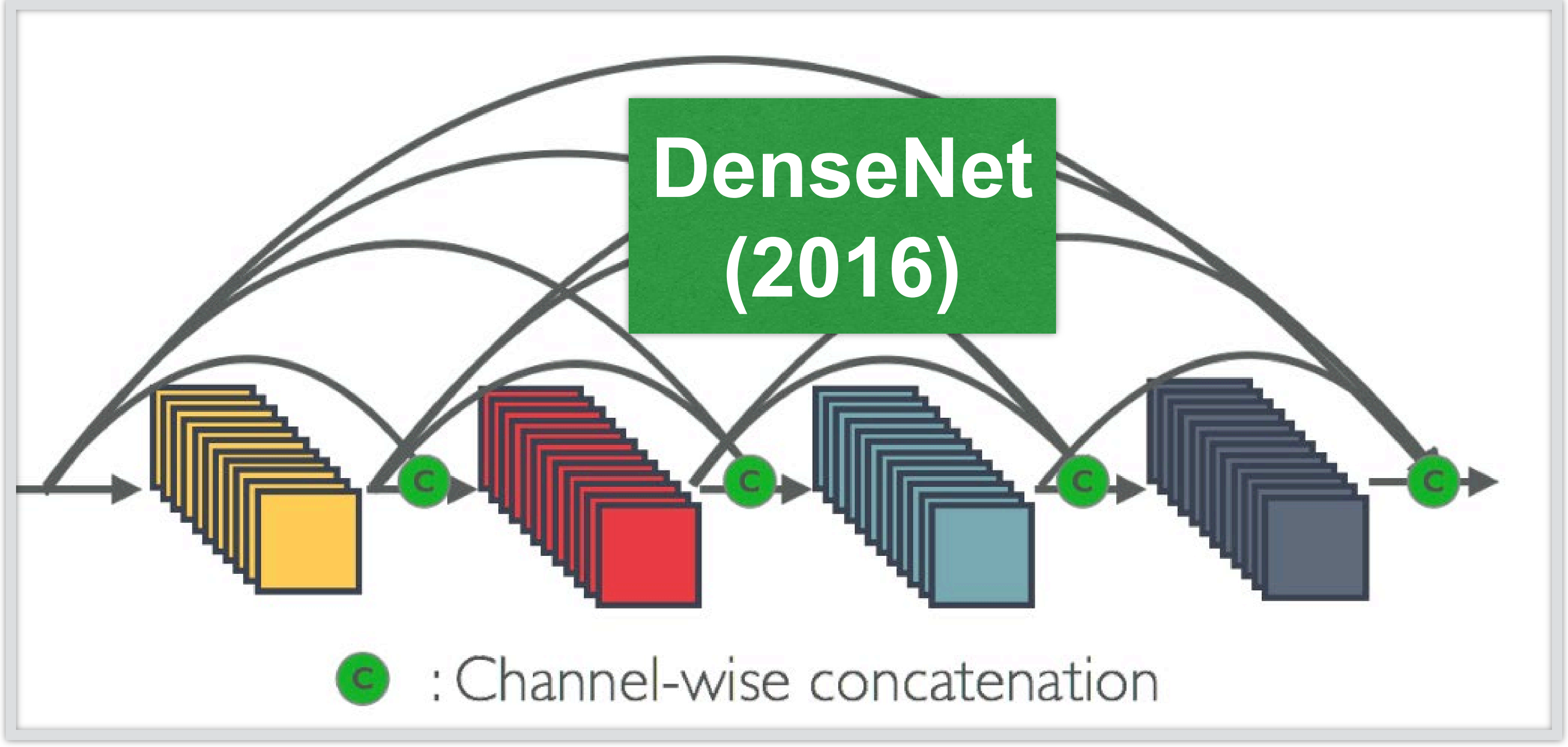








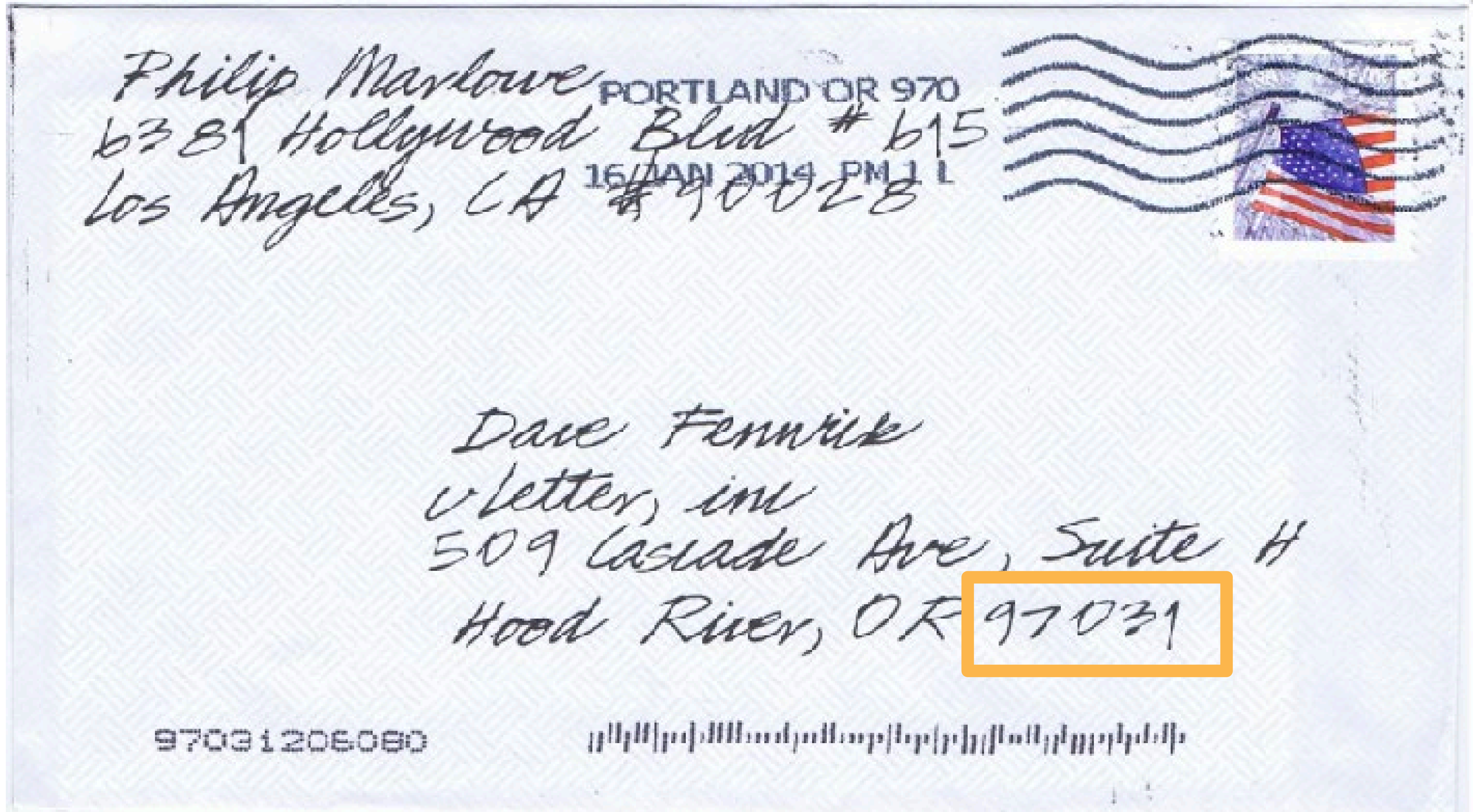


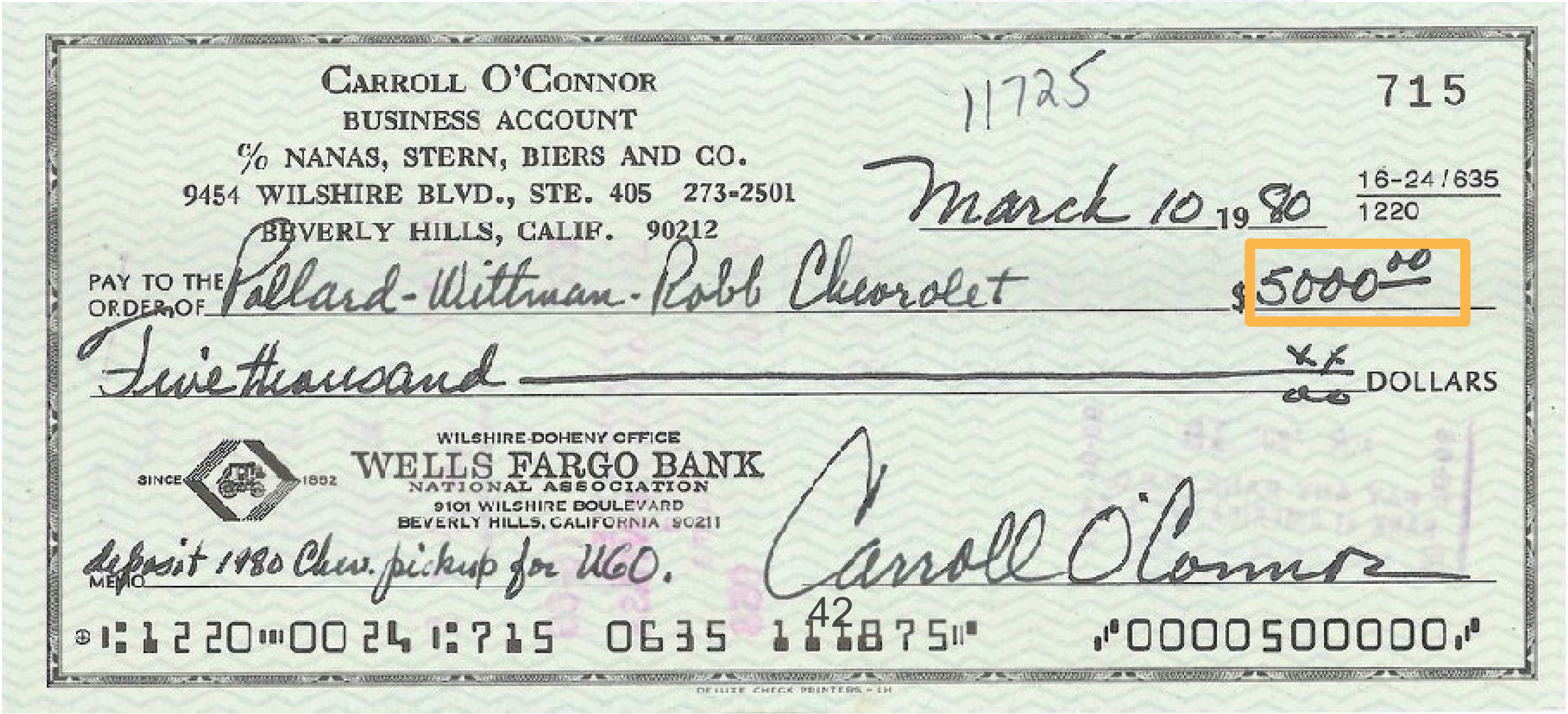


LeNet Architecture
(first convolutional neural net; 1989)

Gradient-based learning applied to document recognition,
by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner

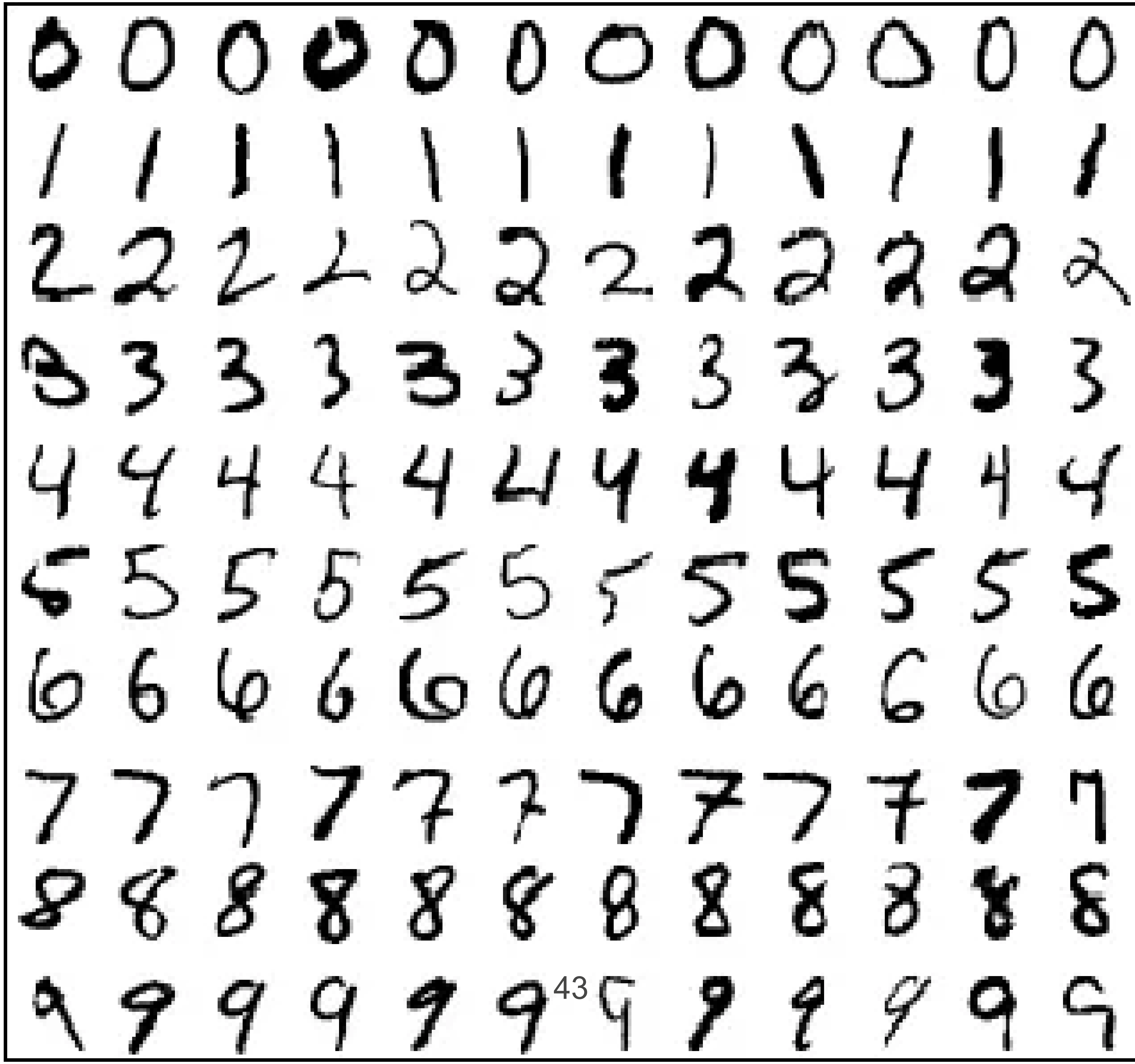
Handwritten Digit Recognition

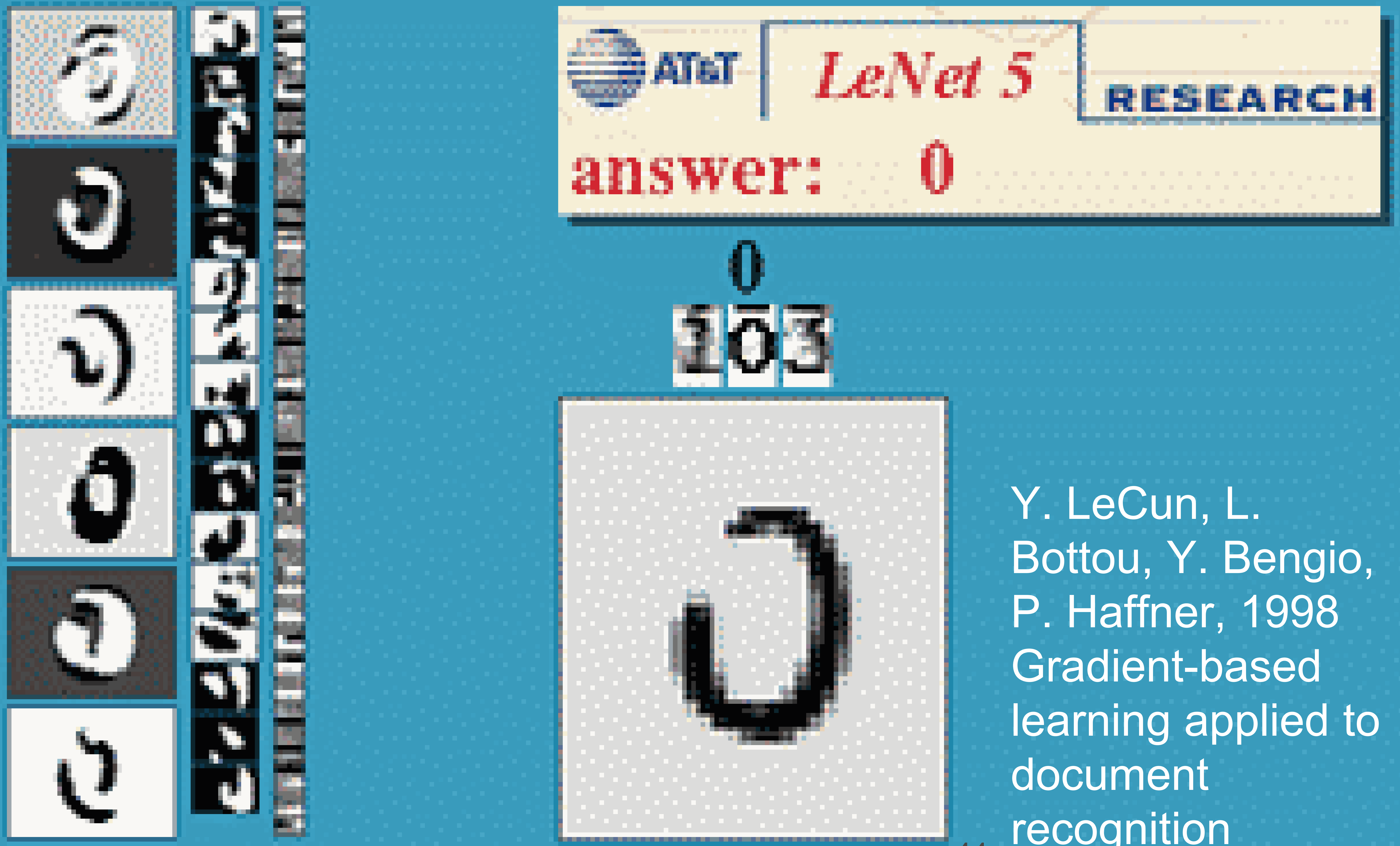




MNIST

- Centered and scaled
- 50,000 training data
- 10,000 test data
- 28 x 28 images
- 10 classes





Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, 1998 Gradient-based learning applied to document recognition

LeNet Architecture

Gradient-based learning applied to document recognition, by Y. LeCun, L. Bottou, Y. Bengio and P. Haffner

LeNet in Pytorch

```
def __init__(self):
    super(LeNet5, self).__init__()
    # Convolution (In LeNet-5, 32x32 images are given as input. Hence padding of 2 is done below)
    self.conv1 = torch.nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5, stride=1, padding=2, bias=True)
    # Max-pooling
    self.max_pool_1 = torch.nn.MaxPool2d(kernel_size=2)
    # Convolution
    self.conv2 = torch.nn.Conv2d(in_channels=6, out_channels=16, kernel_size=5, stride=1, padding=0, bias=True)
    # Max-pooling
    self.max_pool_2 = torch.nn.MaxPool2d(kernel_size=2)
    # Fully connected layer
    self.fc1 = torch.nn.Linear(16*5*5, 120)   # convert matrix with 16*5*5 (= 400) features to a matrix of 120 features (columns
    self.fc2 = torch.nn.Linear(120, 84)       # convert matrix with 120 features to a matrix of 84 features (columns)
    self.fc3 = torch.nn.Linear(84, 10)        # convert matrix with 84 features to a matrix of 10 features (columns)
```

https://github.com/bollakarthikeya/LeNet-5-PyTorch/blob/master/lenet5_gpu.py

LeNet in Pytorch

```
def forward(self, x):
    # convolve, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.conv1(x))
    # max-pooling with 2x2 grid
    x = self.max_pool_1(x)
    # convolve, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.conv2(x))
    # max-pooling with 2x2 grid
    x = self.max_pool_2(x)
    # first flatten 'max_pool_2_out' to contain 16*5*5 columns
    # read through https://stackoverflow.com/a/42482819/7551231
    x = x.view(-1, 16*5*5)
    # FC-1, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.fc1(x))
    # FC-2, then perform ReLU non-linearity
    x = torch.nn.functional.relu(self.fc2(x))
    # FC-3
    x = self.fc3(x)

    return x
```

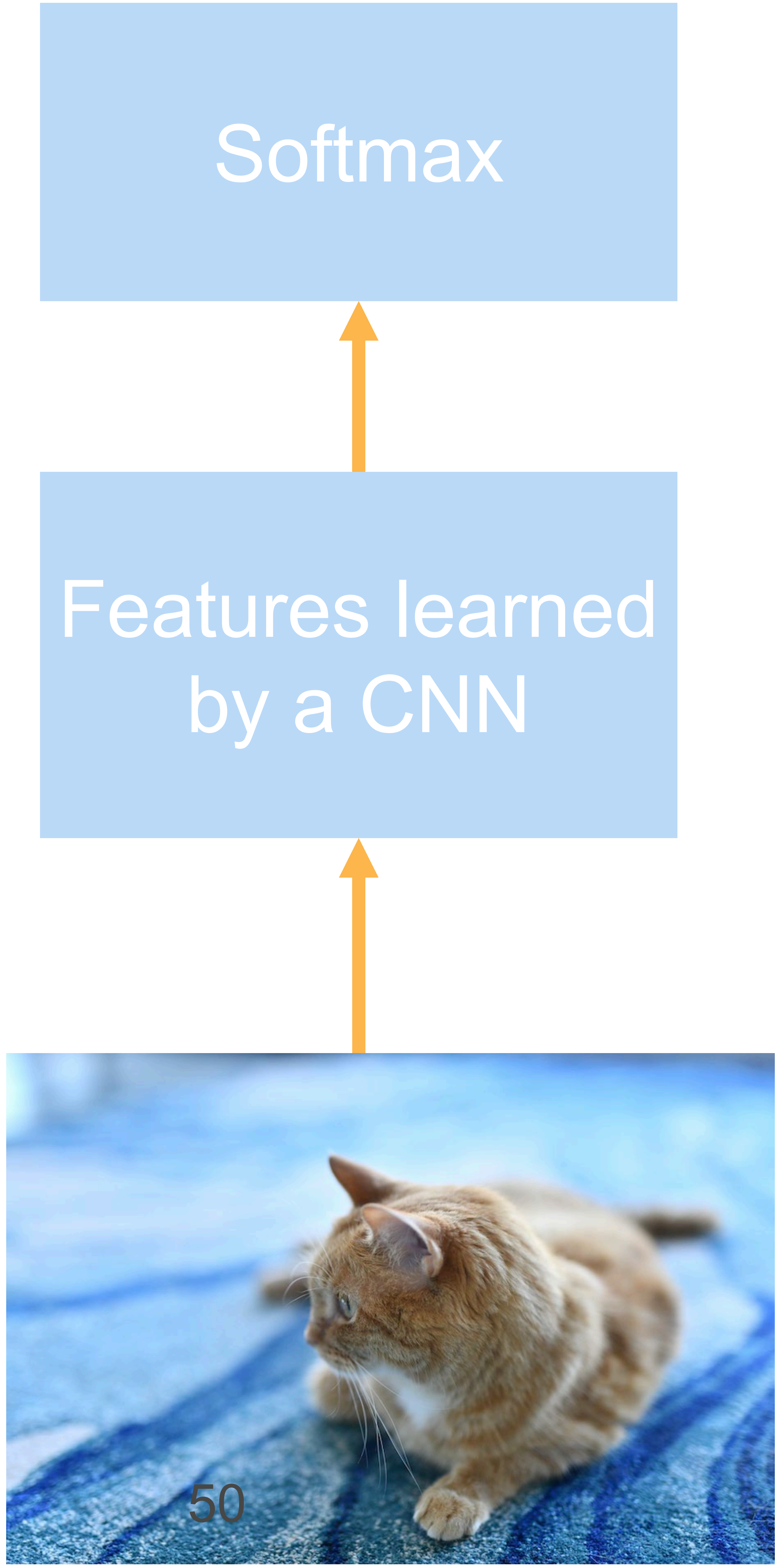
AlexNet

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems.*

Deng et al. 2009

AlexNet

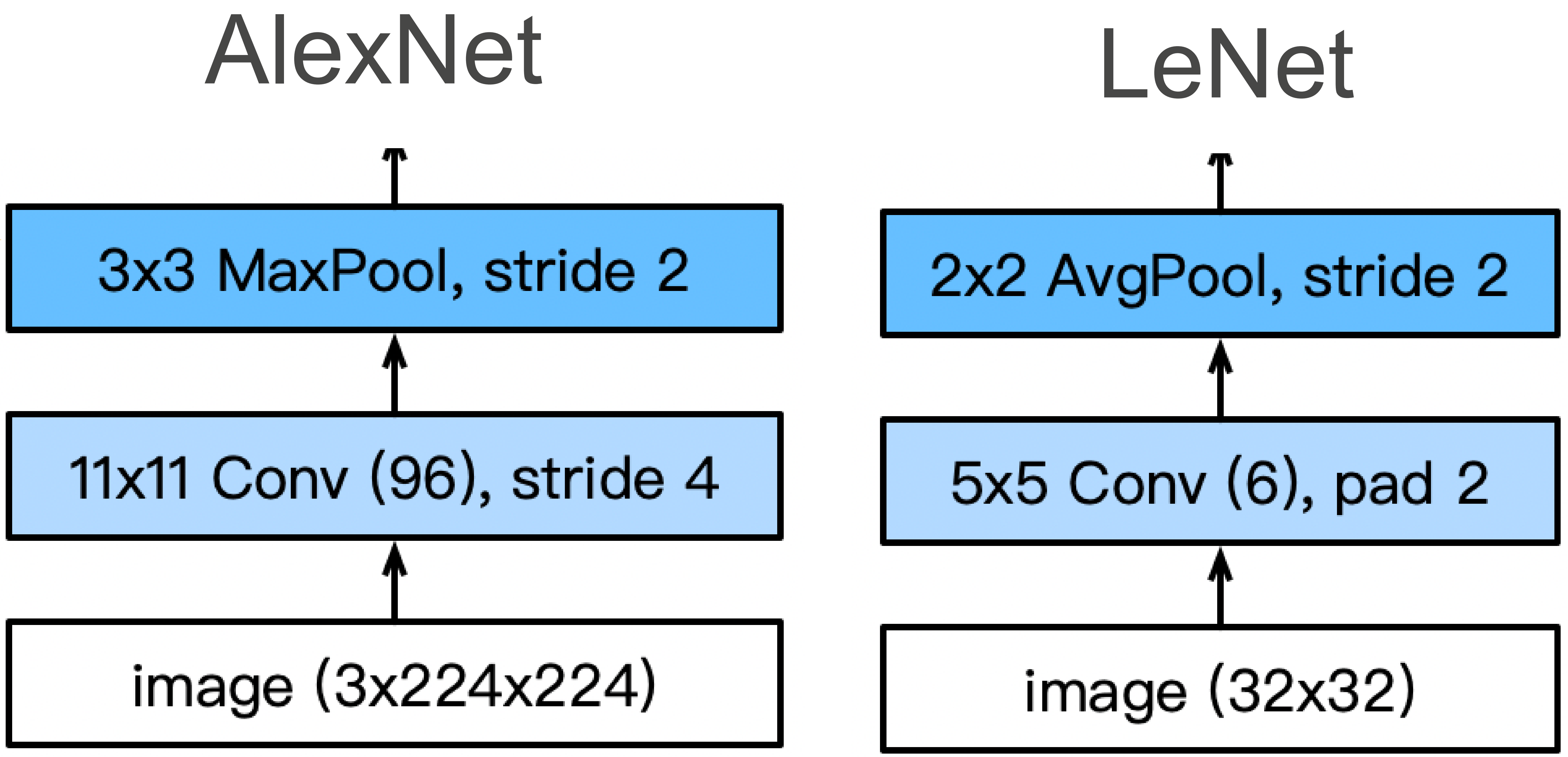
- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Paradigm shift for computer vision



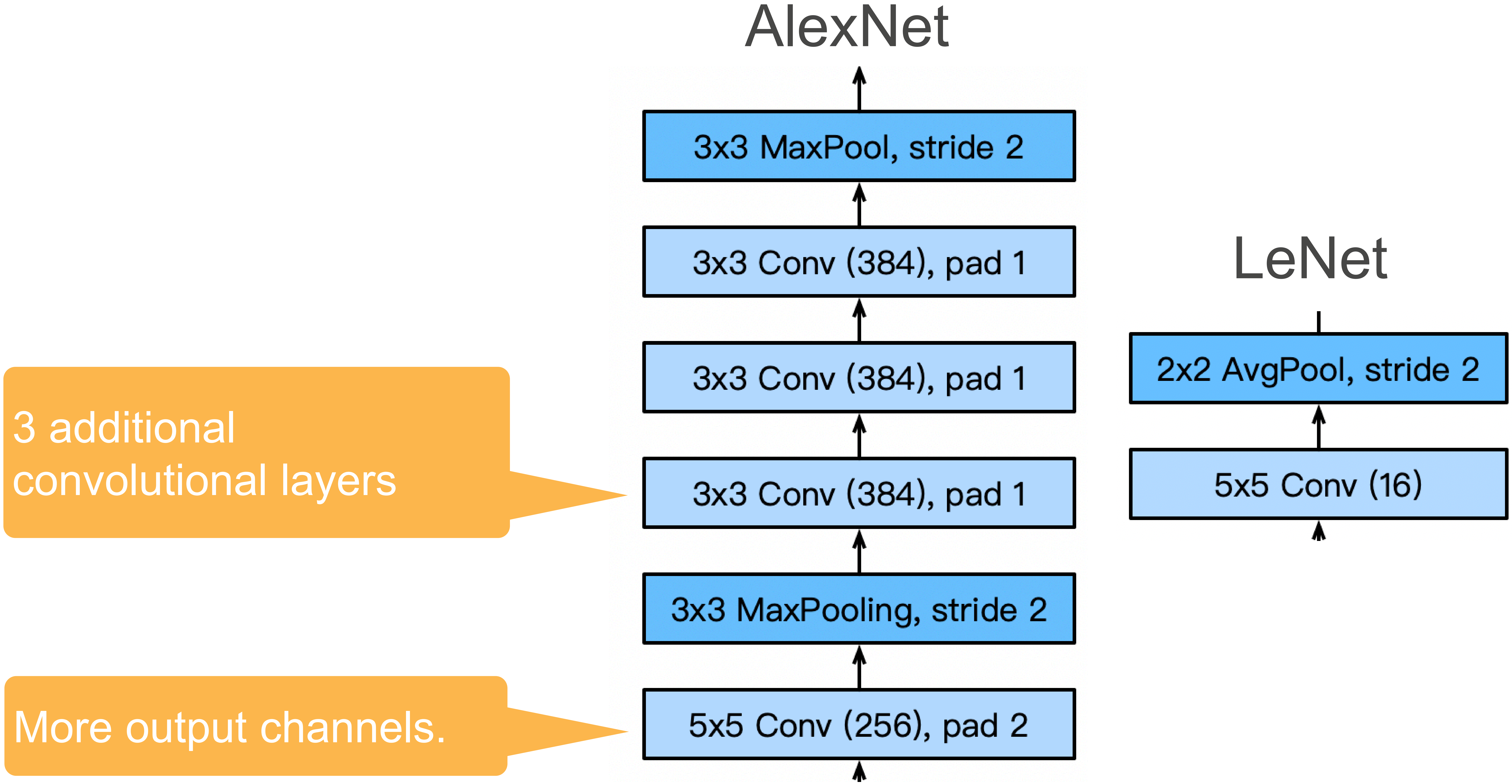
AlexNet Architecture

Larger pool size

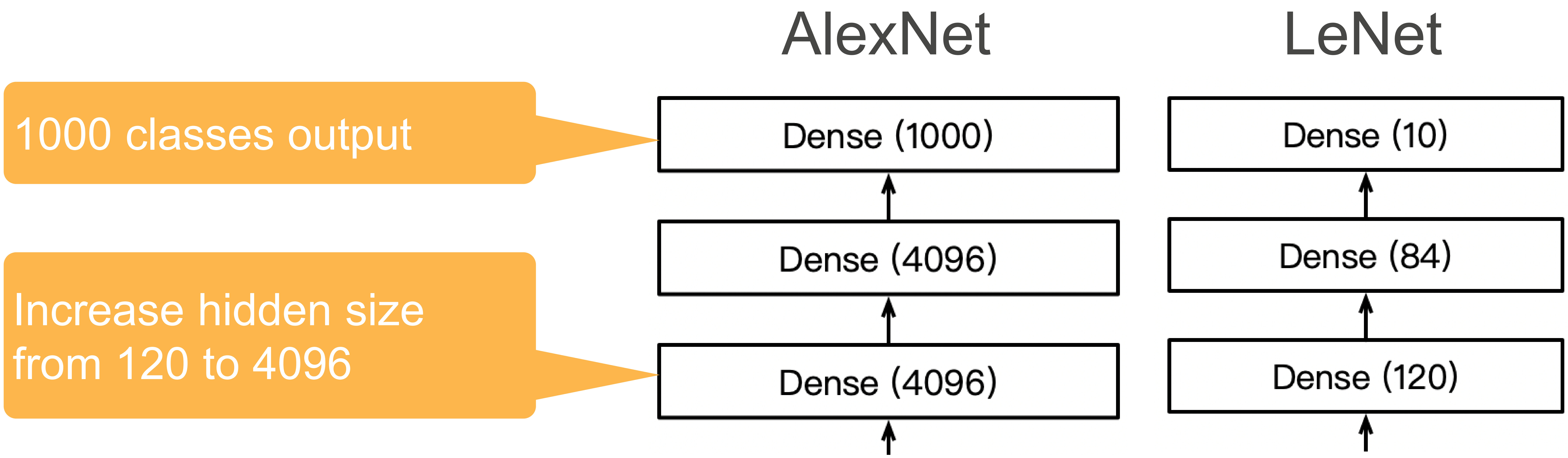
Larger kernel size, stride because of the increased image size, and more output channels.



AlexNet Architecture

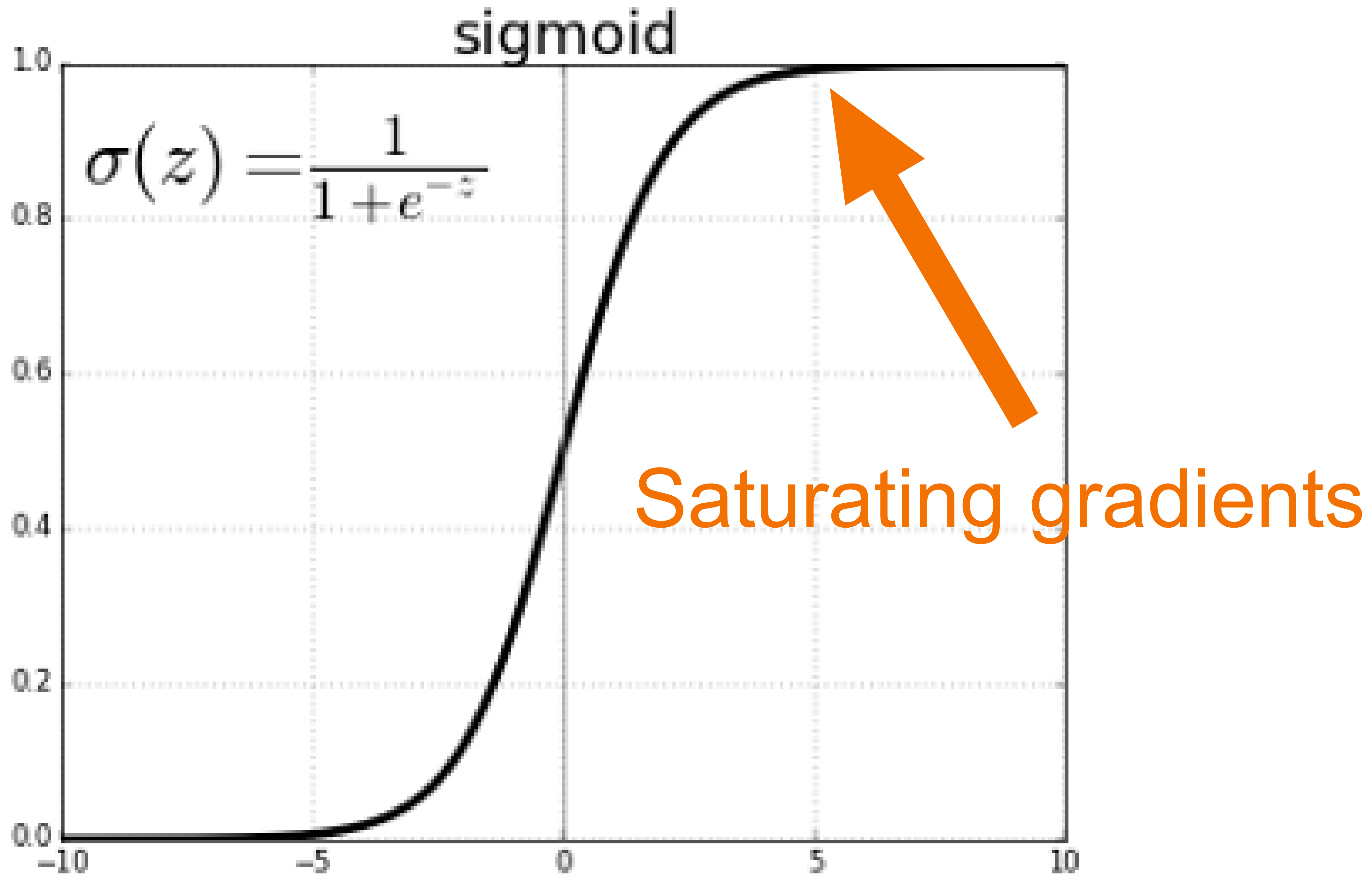


AlexNet Architecture



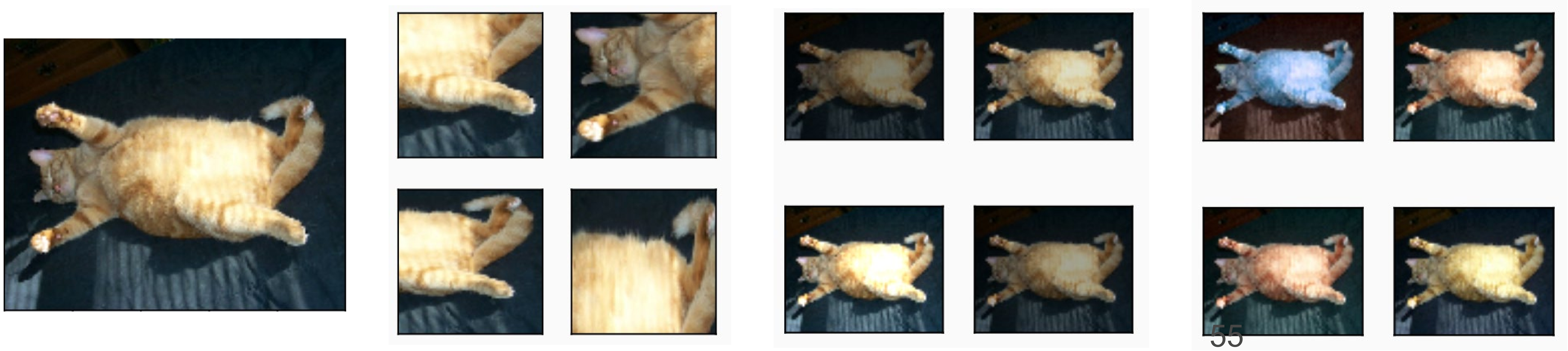
More Differences…

- Change activation function from sigmoid to ReLu (no more vanishing gradient)



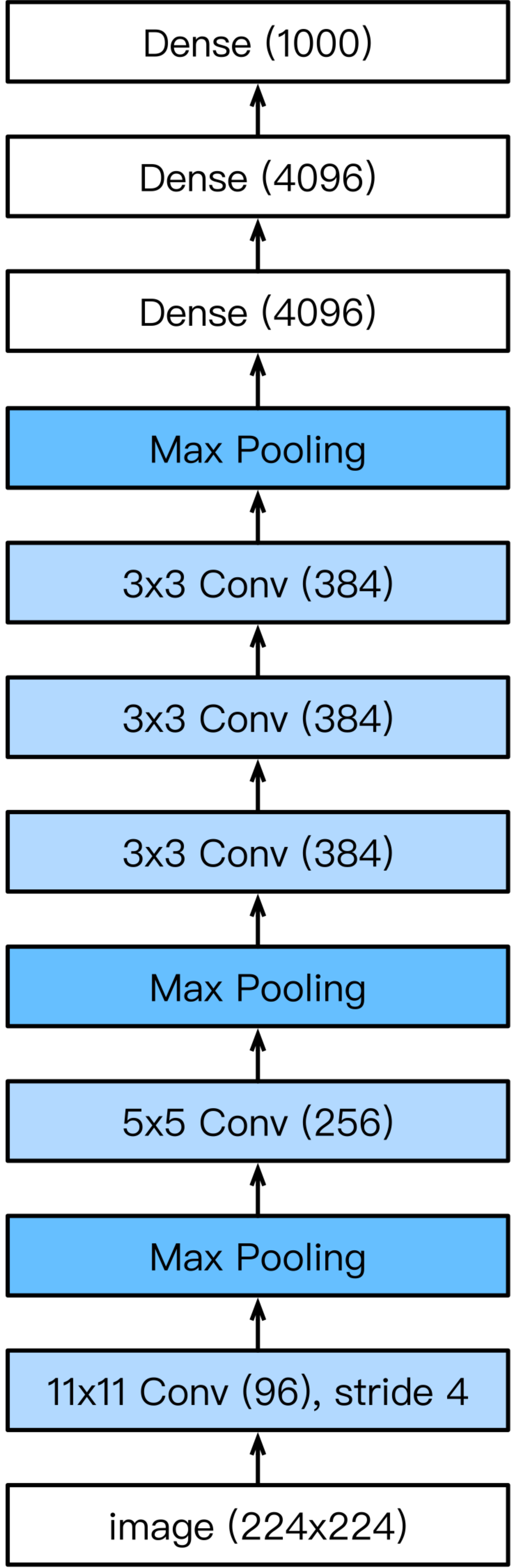
More Differences…

- Change activation function from sigmoid to ReLu (no more vanishing gradient)
- Data augmentation



Complexity

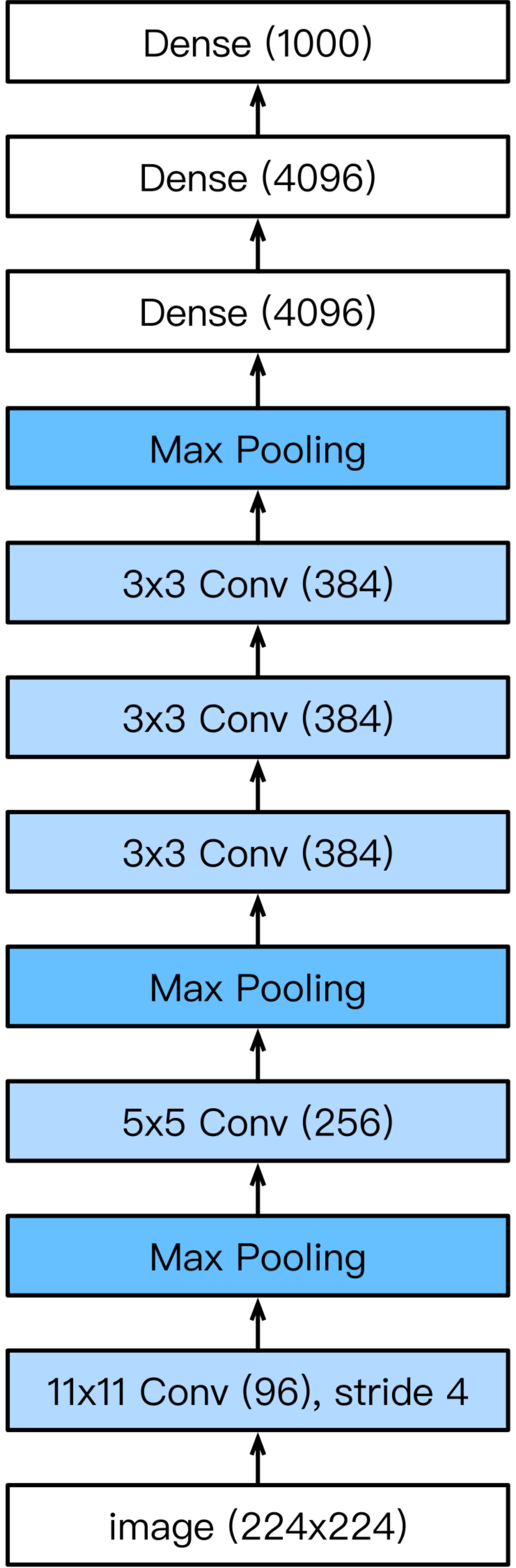
	#parameters	
	AlexNet	**LeNet**
Conv1	35K	150
Conv2	614K	2.4K
Conv3-5	3M	
Dense1	26M	0.048M
Dense2	16M	0.01M
Total	46M	0.06M

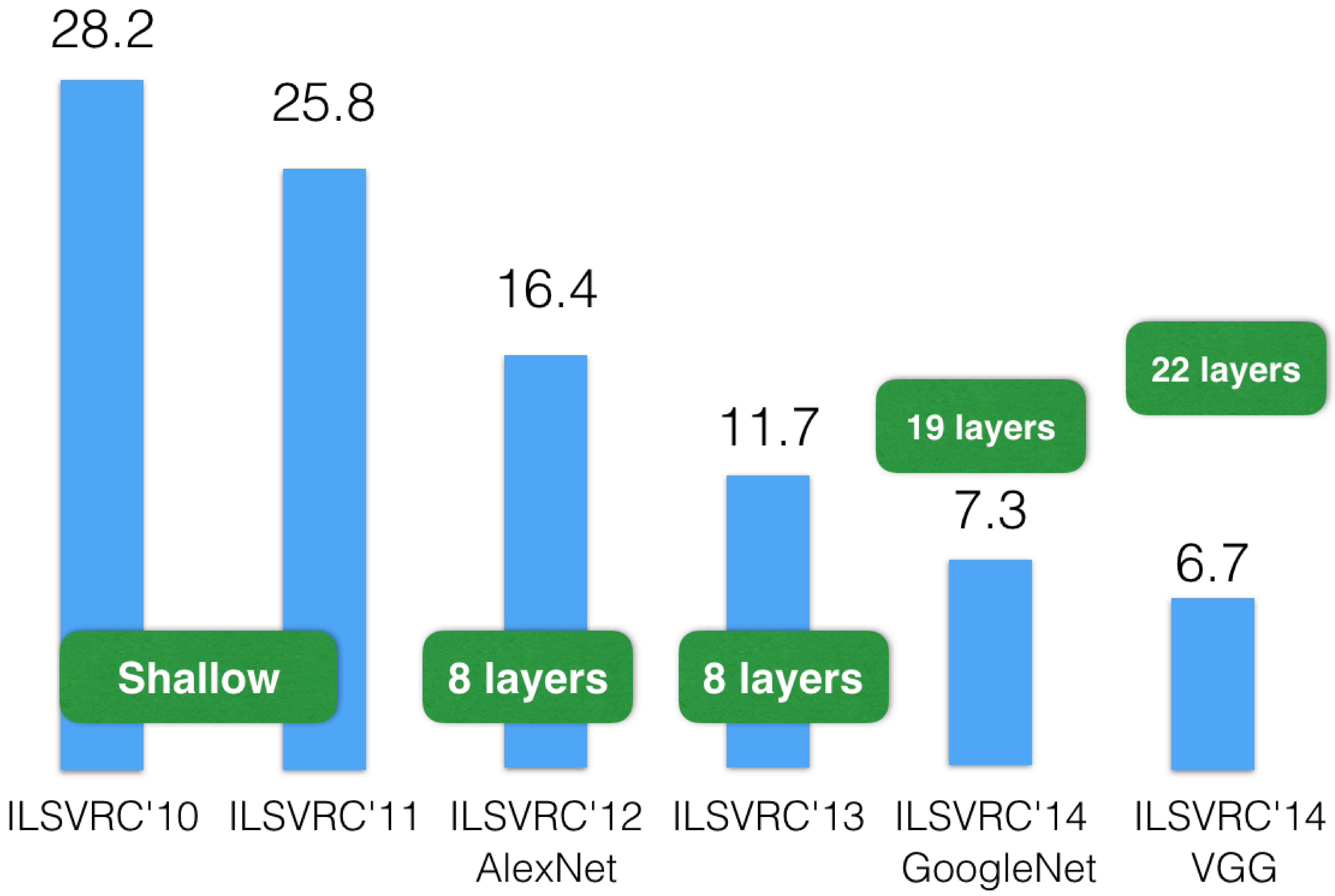


Complexity

	#parameters	
	AlexNet	**LeNet**
Conv1	35K	150
Conv2	614K	2.4K
Conv3-5	3M	
Dense1	26M	0.048M
Dense2	16M	0.01M
Total	46M	0.06M

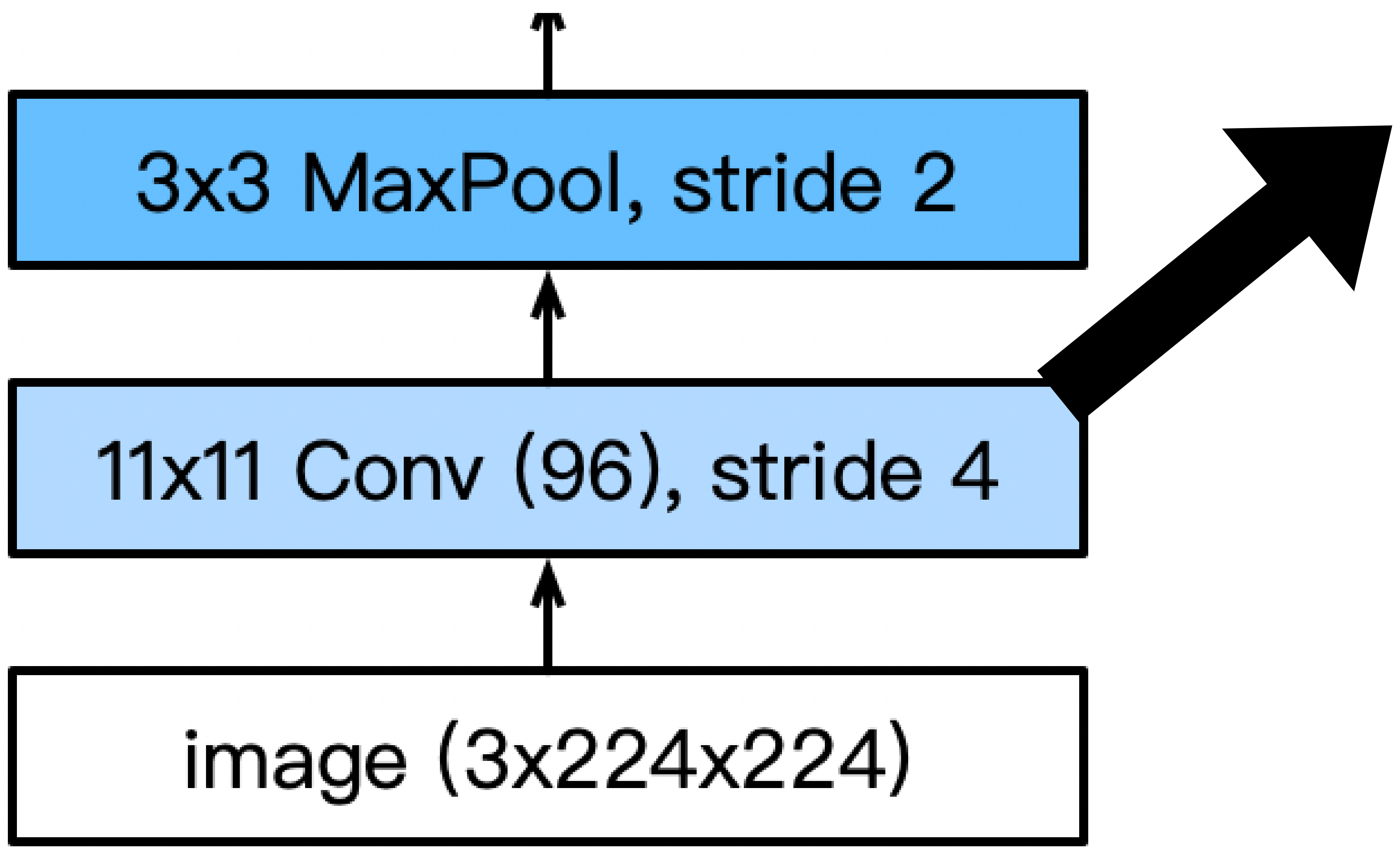
11x11x3x96=35k



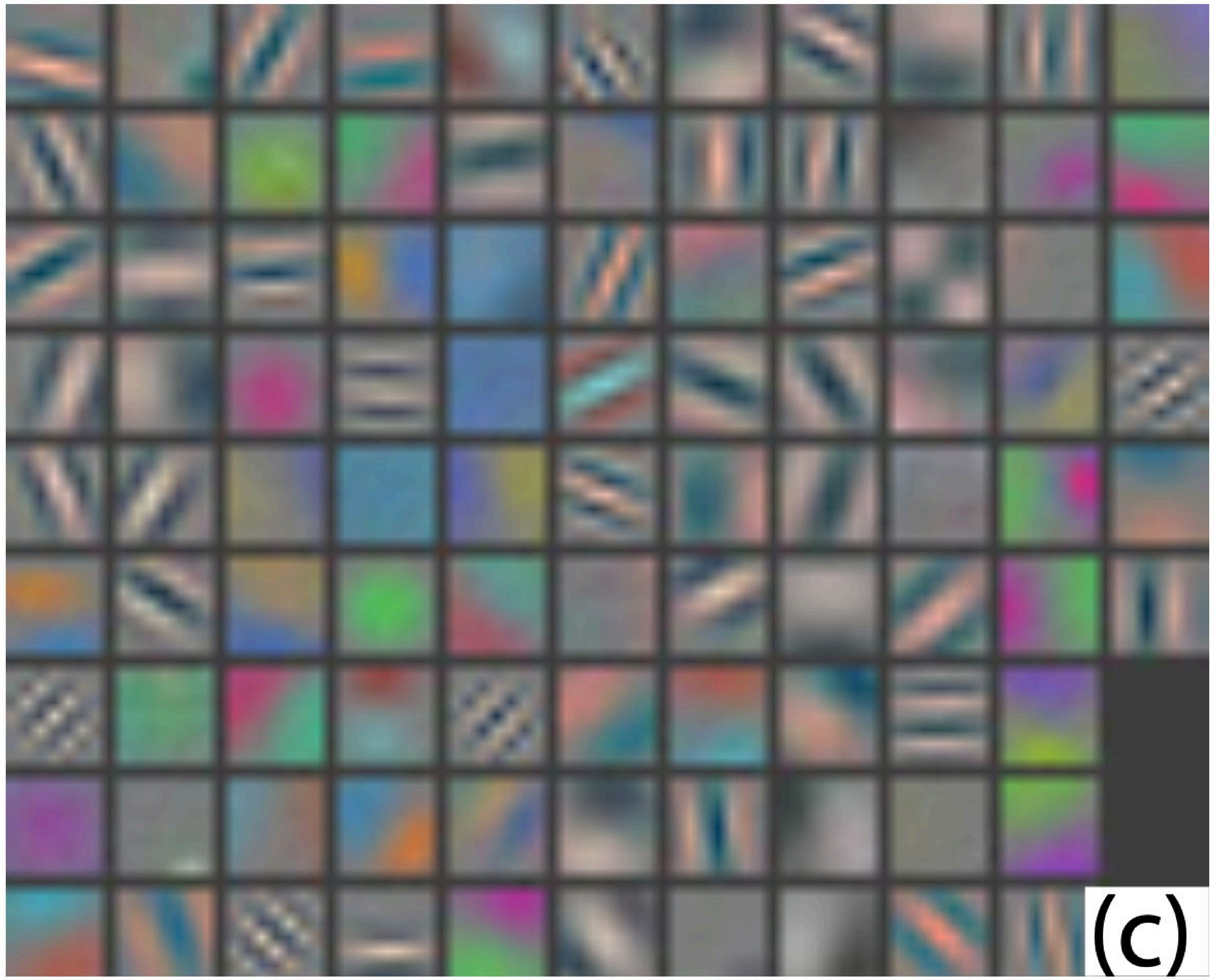


ImageNet Top-5 Classification Error (%)

AlexNet



Each Conv1 kernel is 3x11x11, can be visualized as an RGB patch:



[Visualizing and Understanding Convolutional Networks. M Zeiler & R Fergus 2013]

Which of the following are true about AlexNet? Select all that apply.

A. AlexNet contains 8 conv/fc layers. The first five are convolutional layers.

B. The last three layers are fully connected layers.

C. some of the convolutional layers are followed by max-pooling (layers).

D. AlexNet achieved excellent performance in the 2012 ImageNet challenge.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* (pp. 1097–1105).

Which of the following are true about AlexNet? Select all that apply.

A. AlexNet contains 8 conv/fc layers. The first five are convolutional layers.

B. The last three layers are fully connected layers.

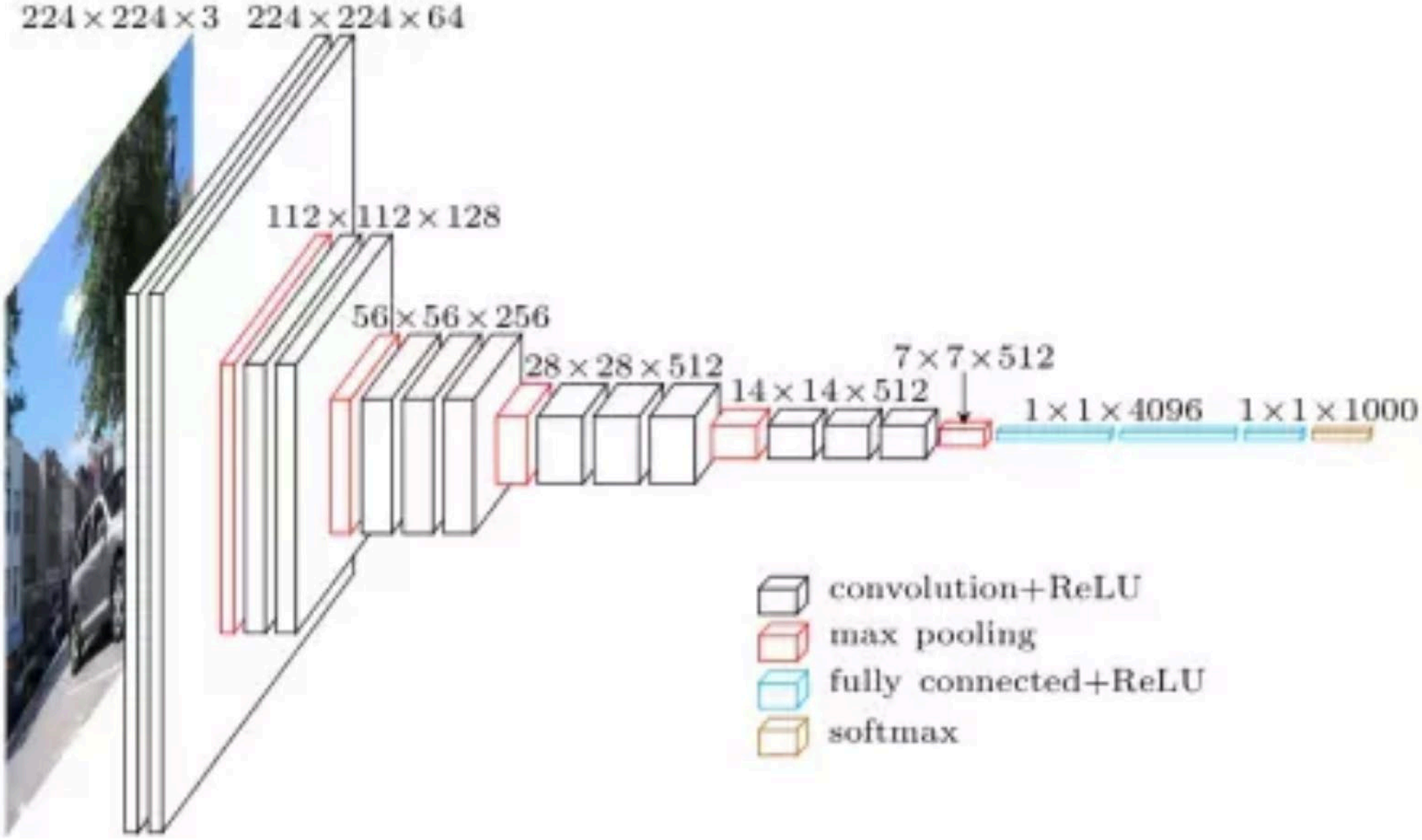
C. some of the convolutional layers are followed by max-pooling (layers).

D. AlexNet achieved excellent performance in the 2012 ImageNet challenge.

All options are true!

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* (pp. 1097–1105).

VGG



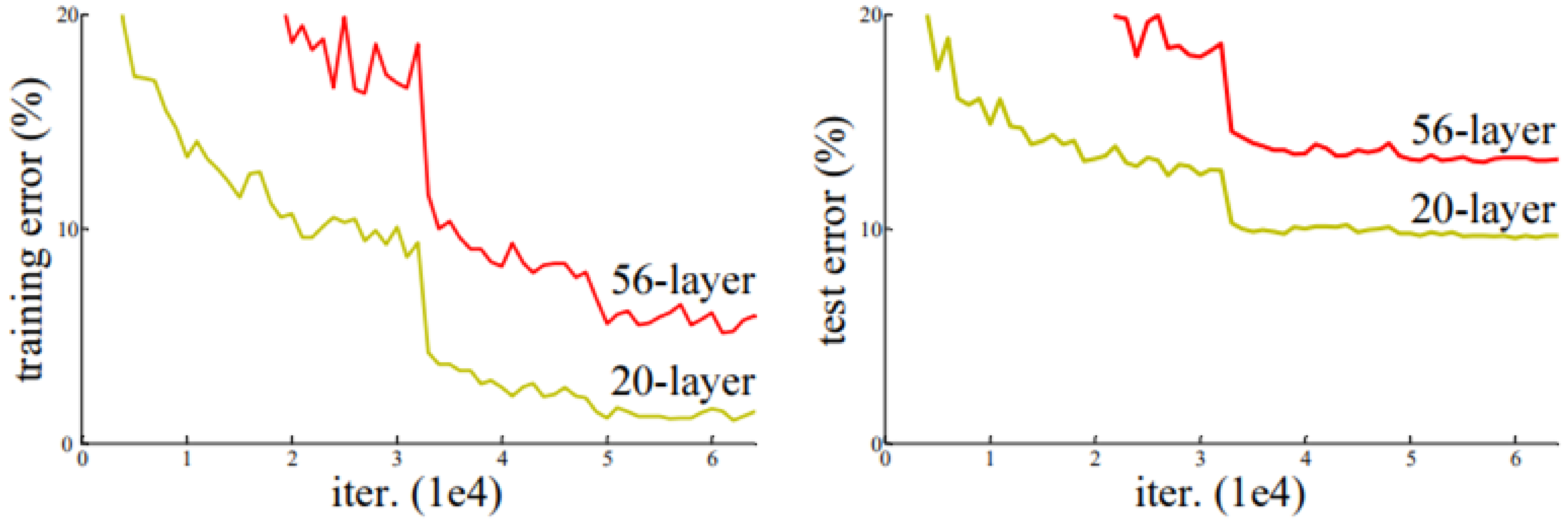
VGG Block: Multiple convolution layers followed by pooling.

Simple Idea: Add More Layers

VGG: 19 layers. ResNet: 152 layers. **Add more layers…** sufficient?

- No! Some problems:
  - i) Vanishing gradients: more layers ➔ more likely
  - ii) Instability: deeper models are harder to optimize

Reflected in training error:

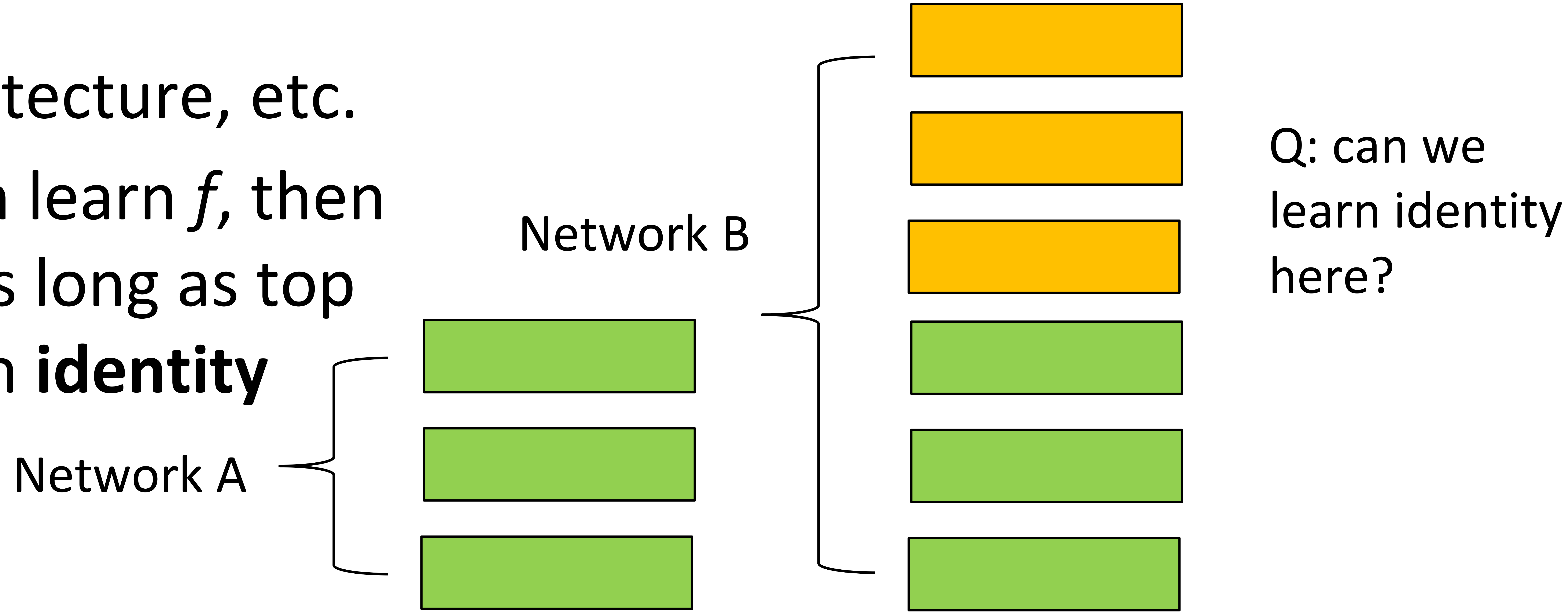


He et al: “Deep Residual Learning for Image Recognition”

Depth Issues & Learning Identity

Why would more layers result in **worse** performance?

- Same architecture, etc.
- If the A can learn *f*, then so can B, as long as top layers learn **identity**

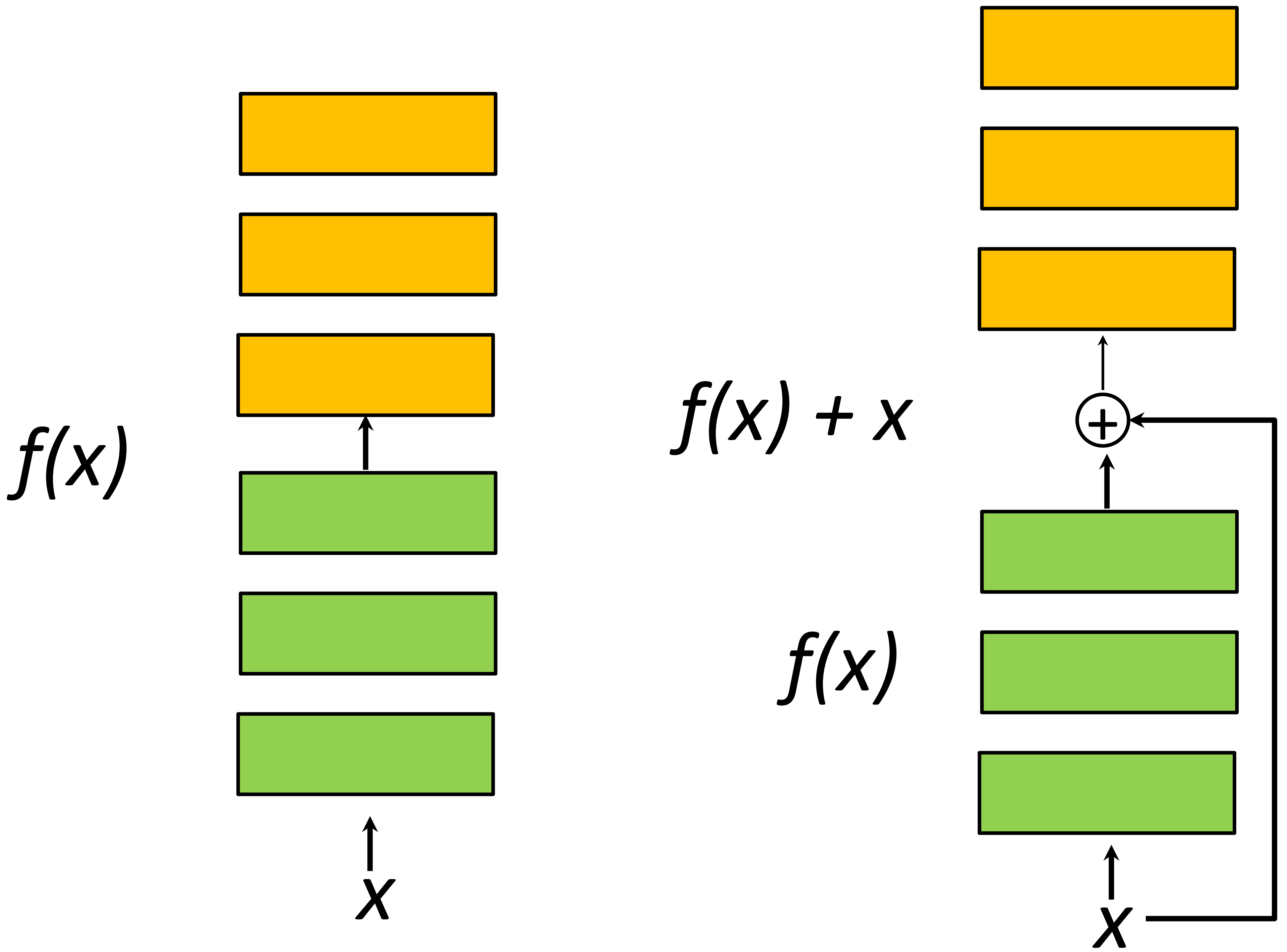


Idea: if layers can learn identity, **can't get** worse.

Residual Connections

Idea: Identity might be hard to learn, but zero is easy!

- Make all the weights tiny, produces zero for output
- Can easily transform learning identity to learning zero:



Left: Conventional layers block

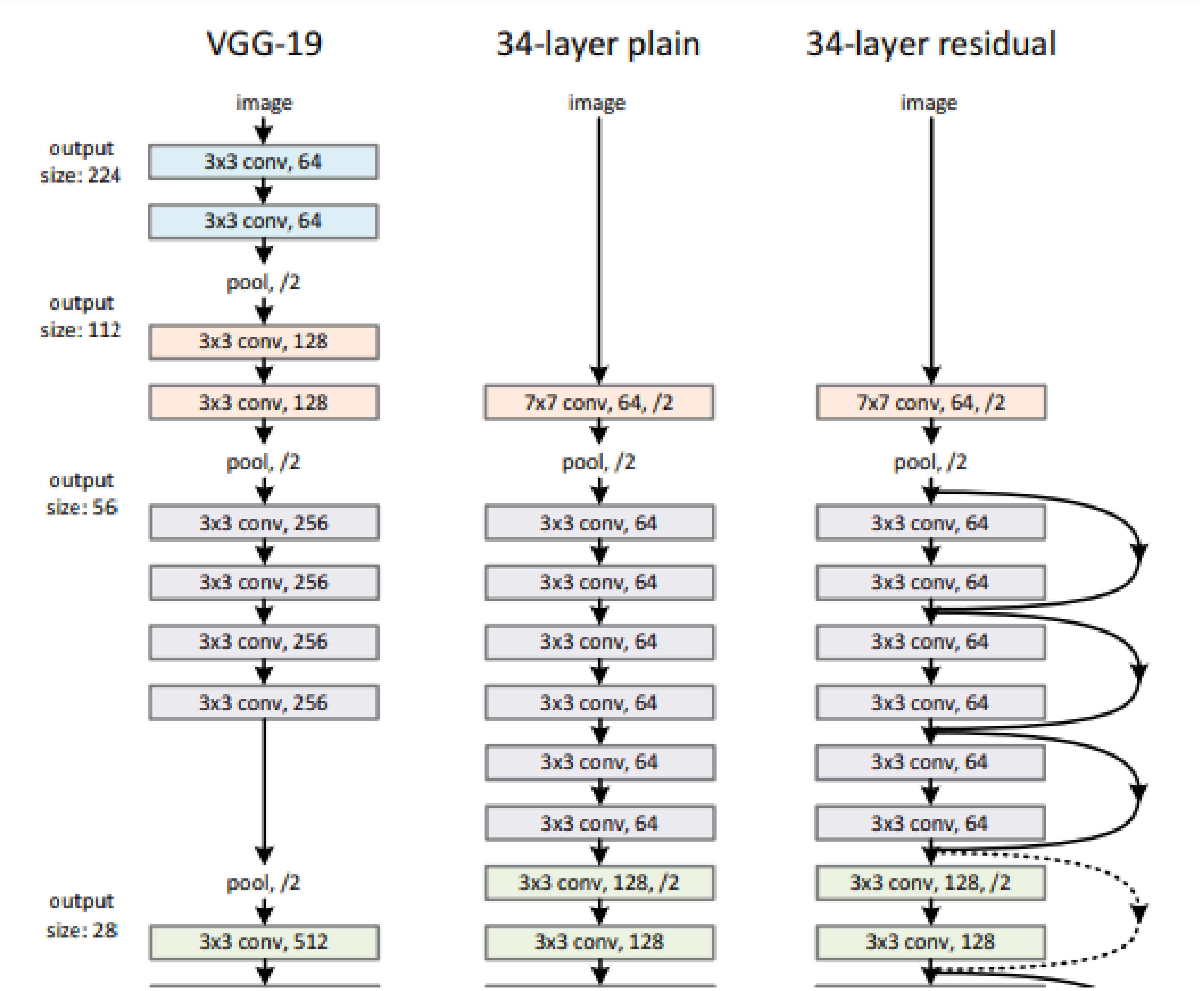
Right: **Residual** layer block

To learn identity $f(x) = x$, layers now need to learn $f(x) = 0$ ➔ easier

ResNet Architecture

Idea: Residual (skip) connections help make learning easier

- Example architecture:
- Note: residual connections
  - Every two layers for ResNet34
- **Vastly better** performance
  - No additional parameters!
  - Records on many benchmarks

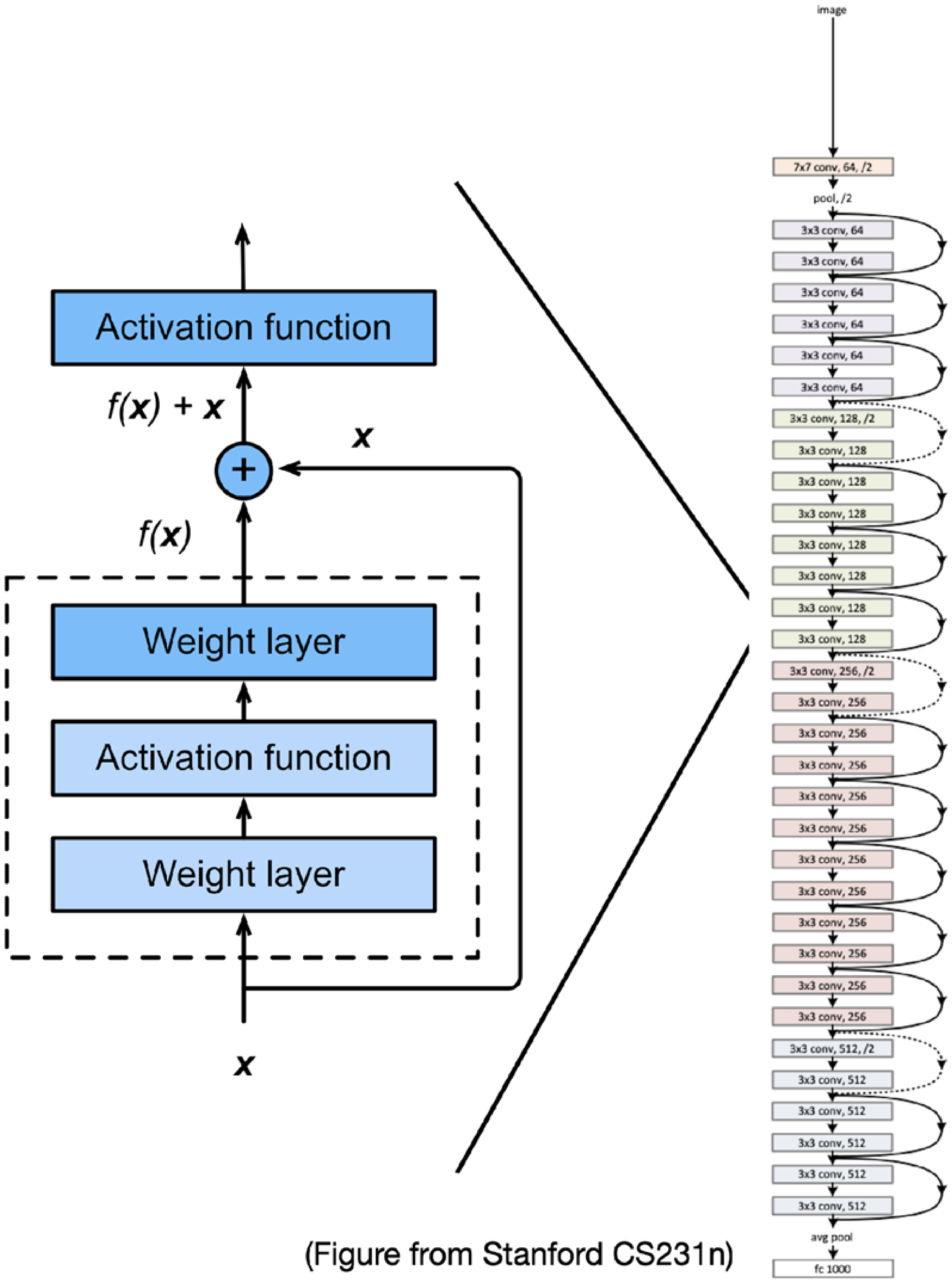


He et al: "Deep Residual Learning for Image Recognition"

Full ResNet Architecture

[He et al. 2015]

- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride of 2 (/2 in each dimension)



(Figure from Stanford CS231n)

ResNet Architecture

Various depth

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

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Architecture

Various depth

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

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Architecture

Various depth

Repeat x3 times

# of filters

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

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Architecture

Various depth

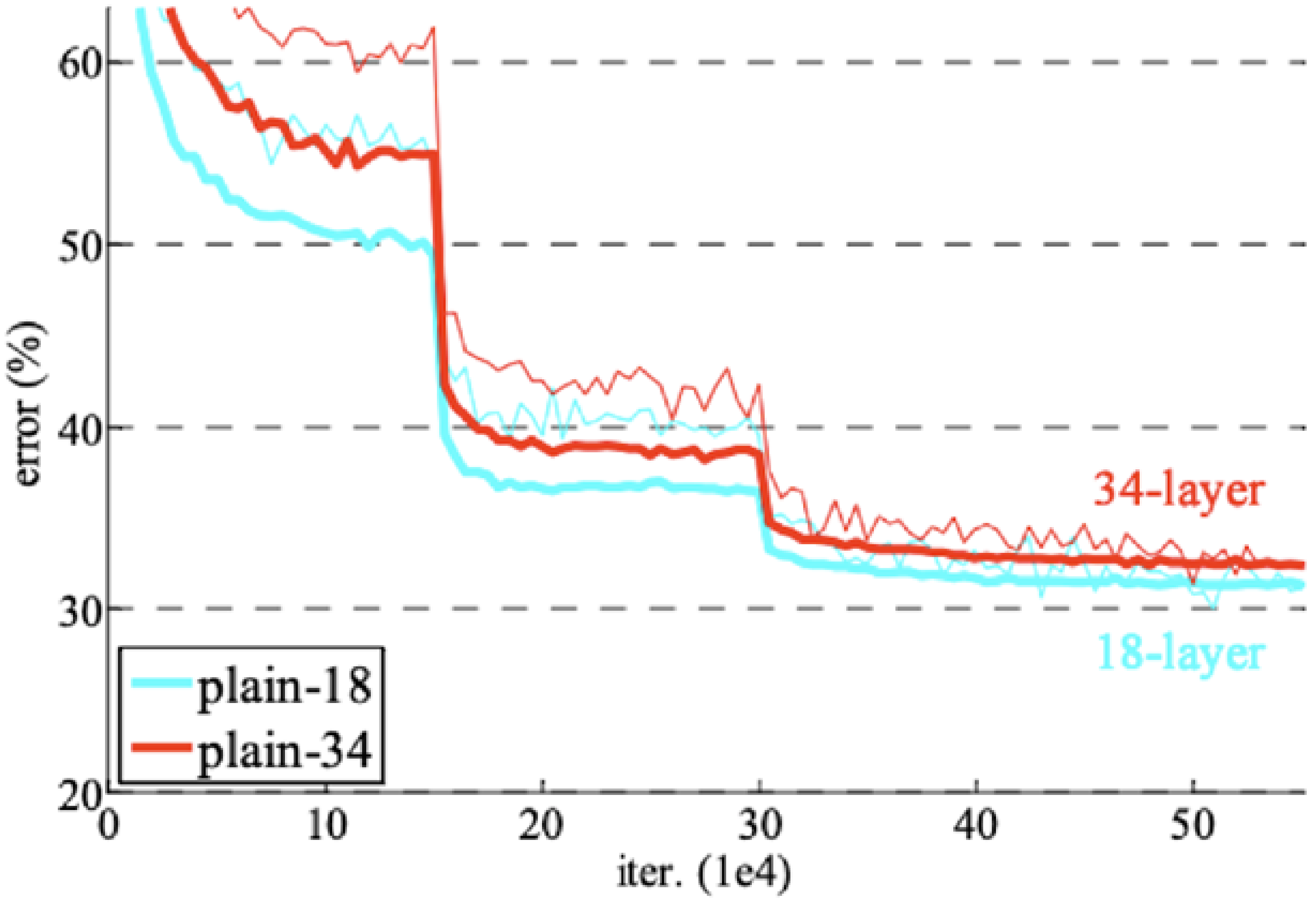
1+ 2x3 + 2x4 + 2x6 + 2x3 +1 = 34

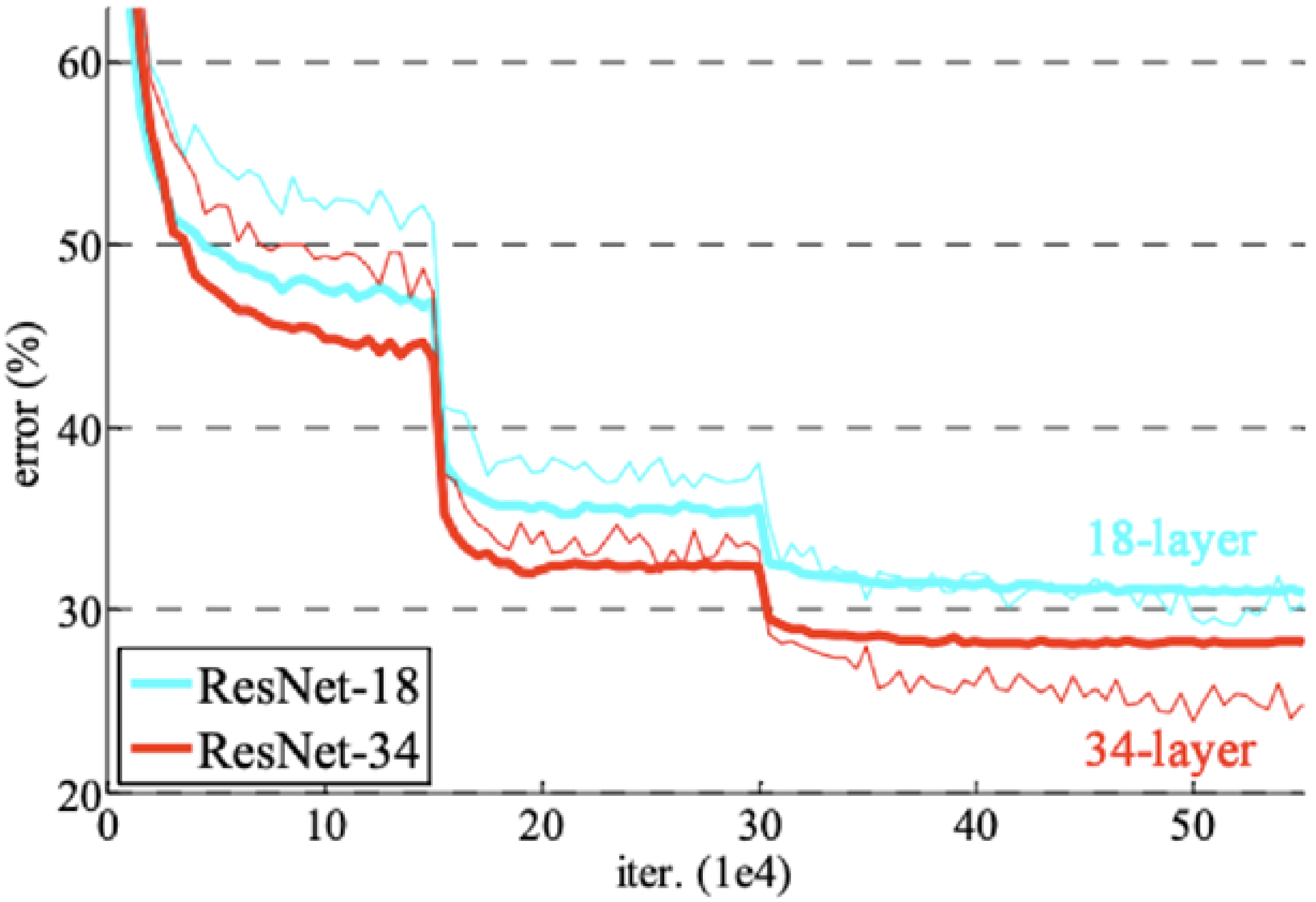
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\ 3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\ 3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\ 3\times3, 64\\ 1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\ 3\times3, 64\\ 1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\ 3\times3, 64\\ 1\times1, 256\end{bmatrix}\times3$
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

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Training Curves on ImageNet

[He et al., 2015]





Progress

- LeNet (1995)
  - 2 convolution + pooling layers
  - 2 hidden dense layers
- AlexNet
  - Bigger and deeper LeNet
  - ReLu, preprocessing
- VGG
  - Bigger and deeper AlexNet (repeated VGG blocks)
- Resnet
  - Residual (Skip) connections

Which of the following statement is True for the success of deep models?

- Better design of the neural networks
- Large scale training dataset
- Available computing power
- All of the above

Which of the following statement is True for the success of deep models?

- Better design of the neural networks
- Large scale training dataset
- Available computing power
- All of the above

Summary of today

- Reviewed (some of) convolutional computations.
  - 2D convolutions, multiple input channels, pooling.
- Shown how convolutions are used as layers in a (deep) neural network.
- Built intuition for output of convolutional layers.
- Evolution of deeper convolutional networks

Suggested Reading

- Example using PyTorch:

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

- Textbook: Artificial Intelligence: A Modern Approach (4th edition). Stuart Russell and Peter Norvig. Pearson, 2020. Section 21.3

Acknowledgement:

Some of the slides in these lectures have been adapted/borrowed from materials developed by Yin Li (https://happyharrycn.github.io/CS540-Fall20/schedule/), Alex Smola and Mu Li: https://courses.d2l.ai/berkeley-stat-157/index.html