CS540 Introduction to Artificial Intelligence

Convolutional Neural Networks (II)

Josiah Hanna
University of Wisconsin-Madison

November 4, 2021

Slides created by Sharon Li [modified by Josiah Hanna]

Announcements

Tuesday, Nov 2	Machine Learning: Deep Learning I	Slides	
Thursday, Nov 4	Machine Learning: Deep Learning II	Slides	HW 6 Due, HW 7 Released
Everything below here is tentative and subject to change.			
Tuesday, Nov 9	Machine Learning: Deep Learning III		
Thursday, Nov 11	Machine Learning: Deep Learning and Neural Network's Summary		
Tuesday, Nov 16	Search I: Un-Informed search		HW 7 Due; HW 8 Released
Thursday, Nov 18	Search II: Informed search		
Tuesday, Nov 23	Games - Part I		HW 8 Due; HW 9 Released
Thursday, Nov 25	Happy Thanksgiving!		

Outline

Outline

- Brief review of convolutional computations

Outline

- Brief review of convolutional computations
- Convolutional Neural Networks

Outline

- Brief review of convolutional computations
- Convolutional Neural Networks
    - LeNet (first conv nets)

Outline

- Brief review of convolutional computations
- Convolutional Neural Networks
    - LeNet (first conv nets)
    - AlexNet

How to classify
Cats vs. dogs?

How to classify

Cats vs. dogs?

How to classify

Cats vs. dogs?

Dual

12MP

wide-angle and
telephoto cameras

How to classify

Cats vs. dogs?

Dual

12MP

wide-angle and
telephoto cameras

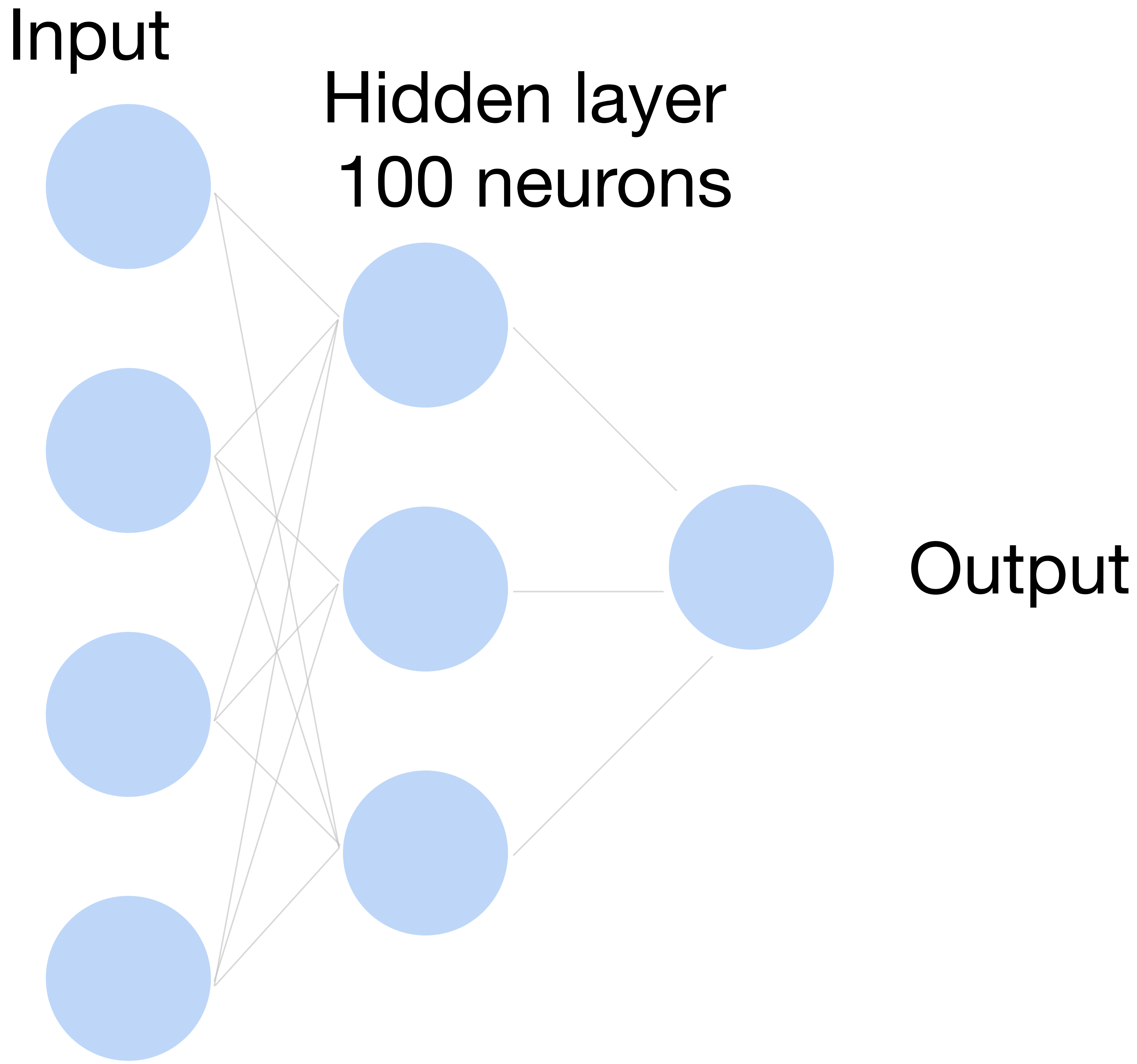
36M floats in a RGB image!

Fully Connected Networks

Cats vs. dogs?

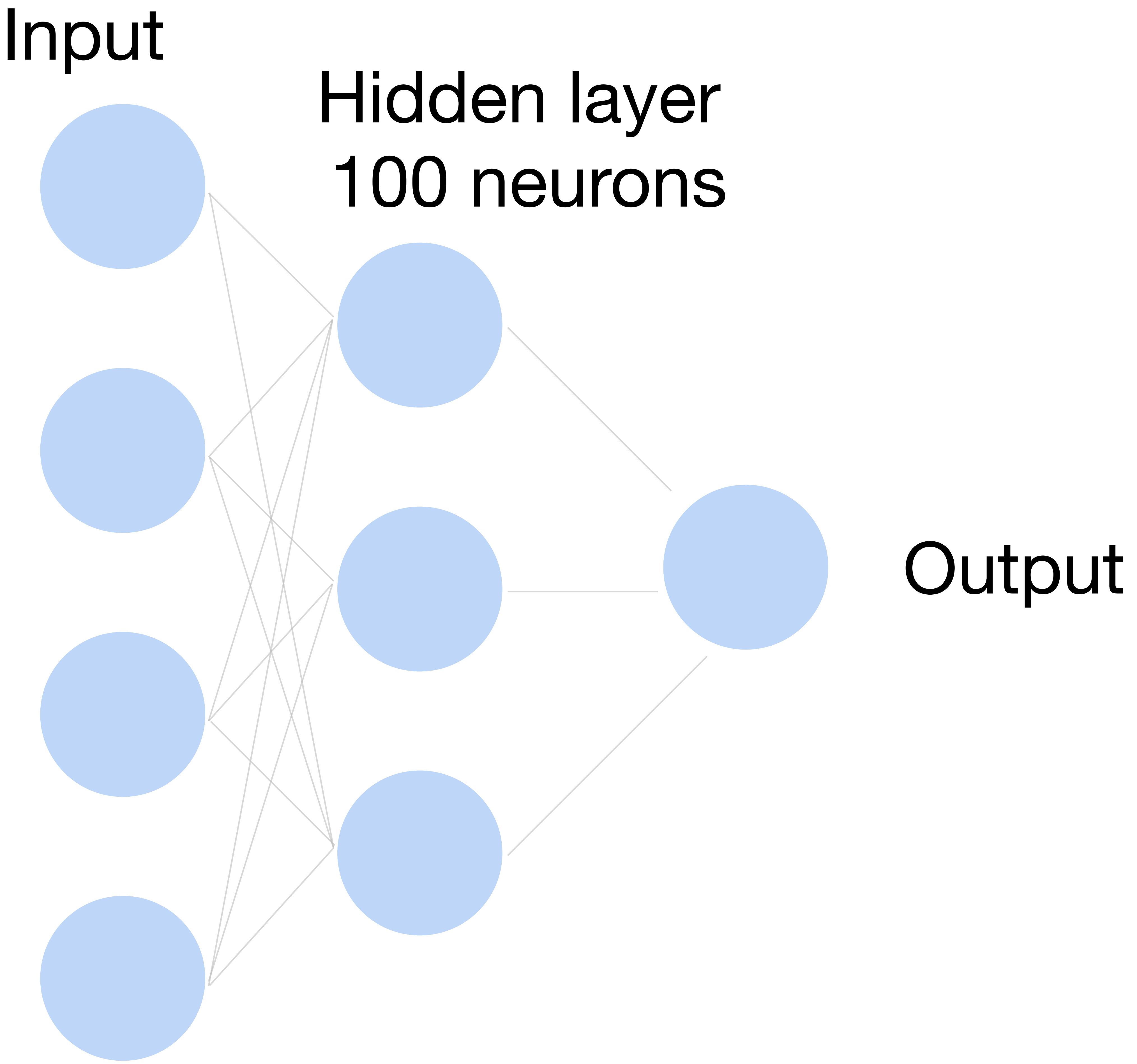
Fully Connected Networks

Cats vs. dogs?



Fully Connected Networks

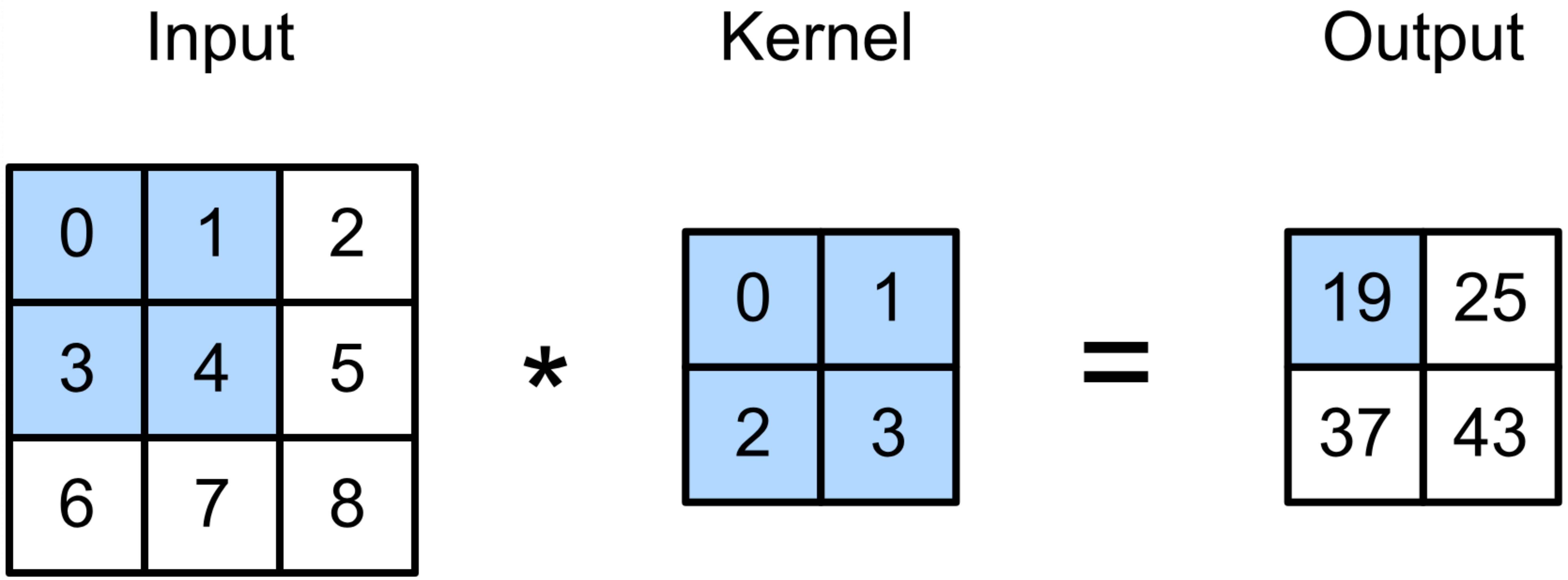
Cats vs. dogs?



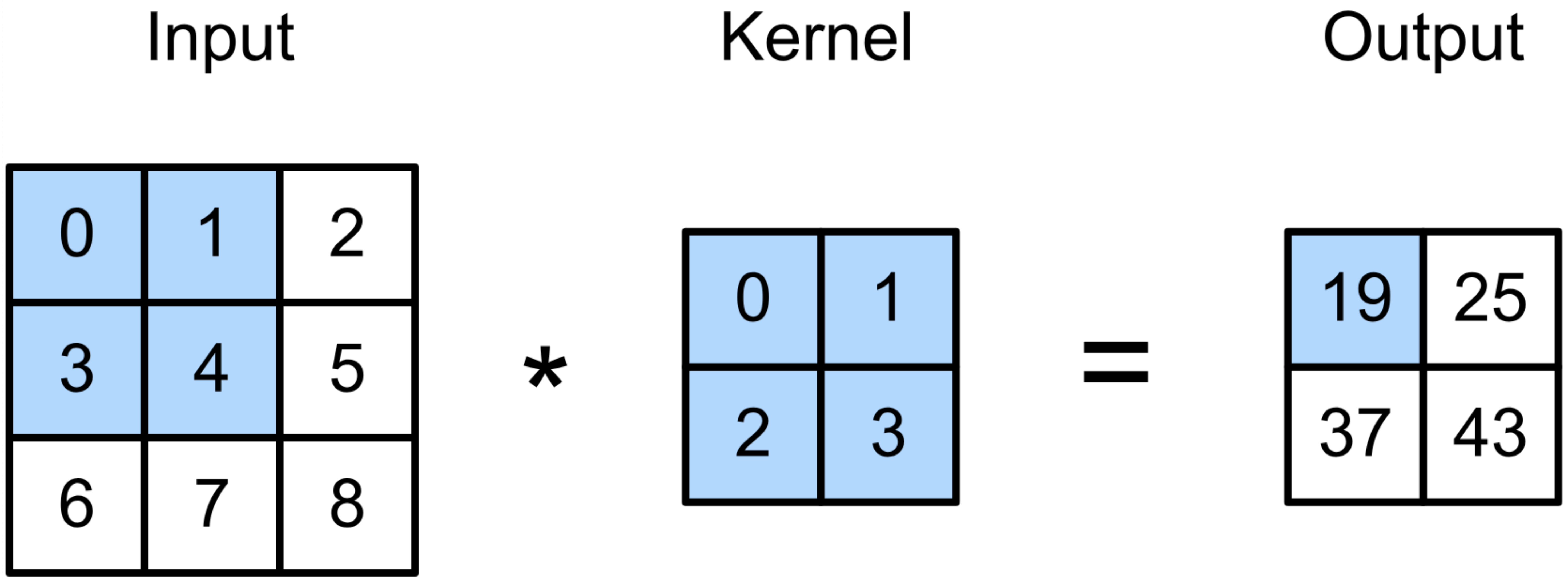
36M elements x 100 = **3.6B** parameters!

Review: 2-D Convolution

Review: 2-D Convolution



Review: 2-D Convolution

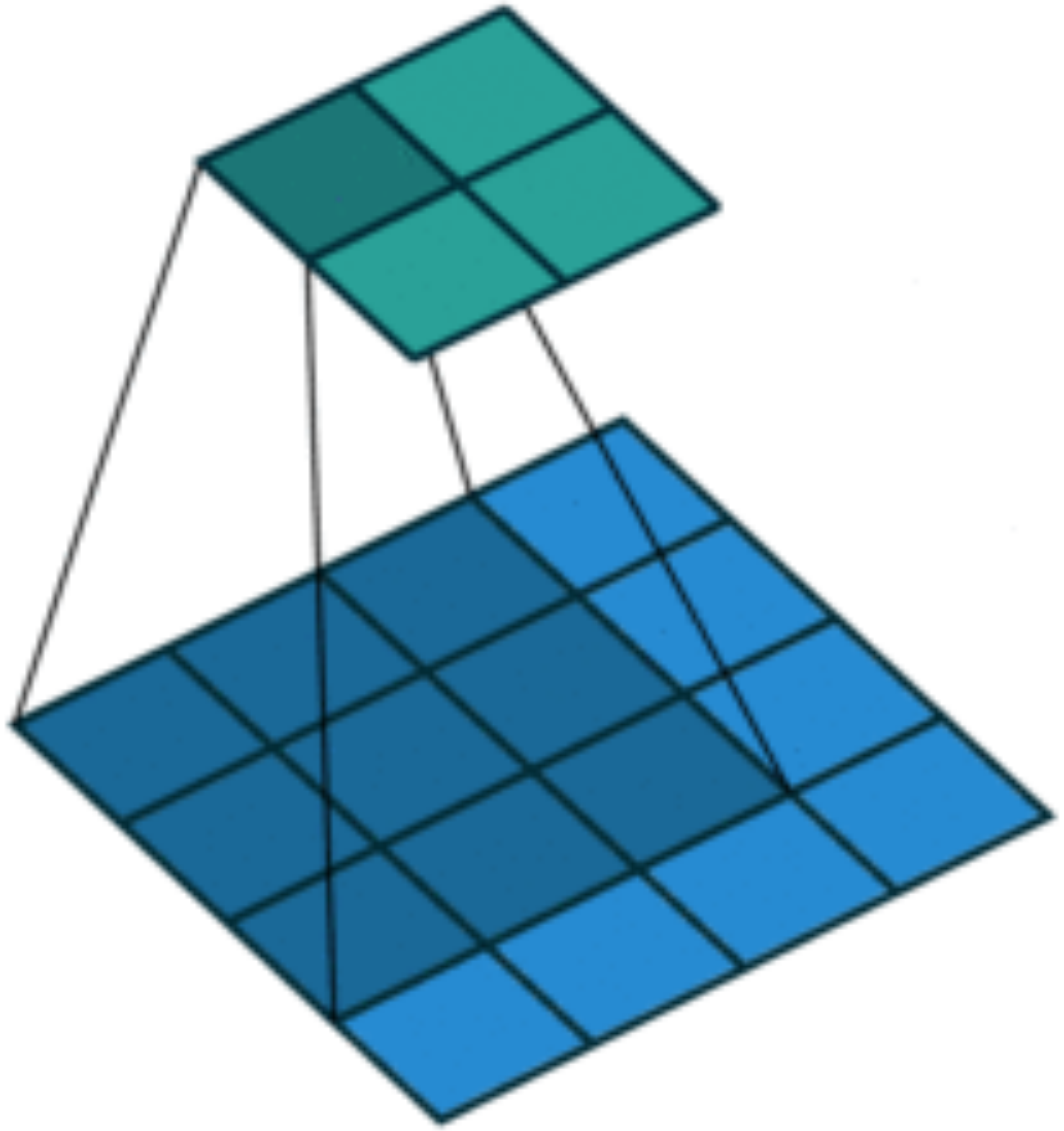


$$
\begin{aligned}
0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 &= 19,\\
1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 &= 25,\\
3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 &= 37,\\
4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 &= 43.
\end{aligned}
$$

Review: 2-D Convolution

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

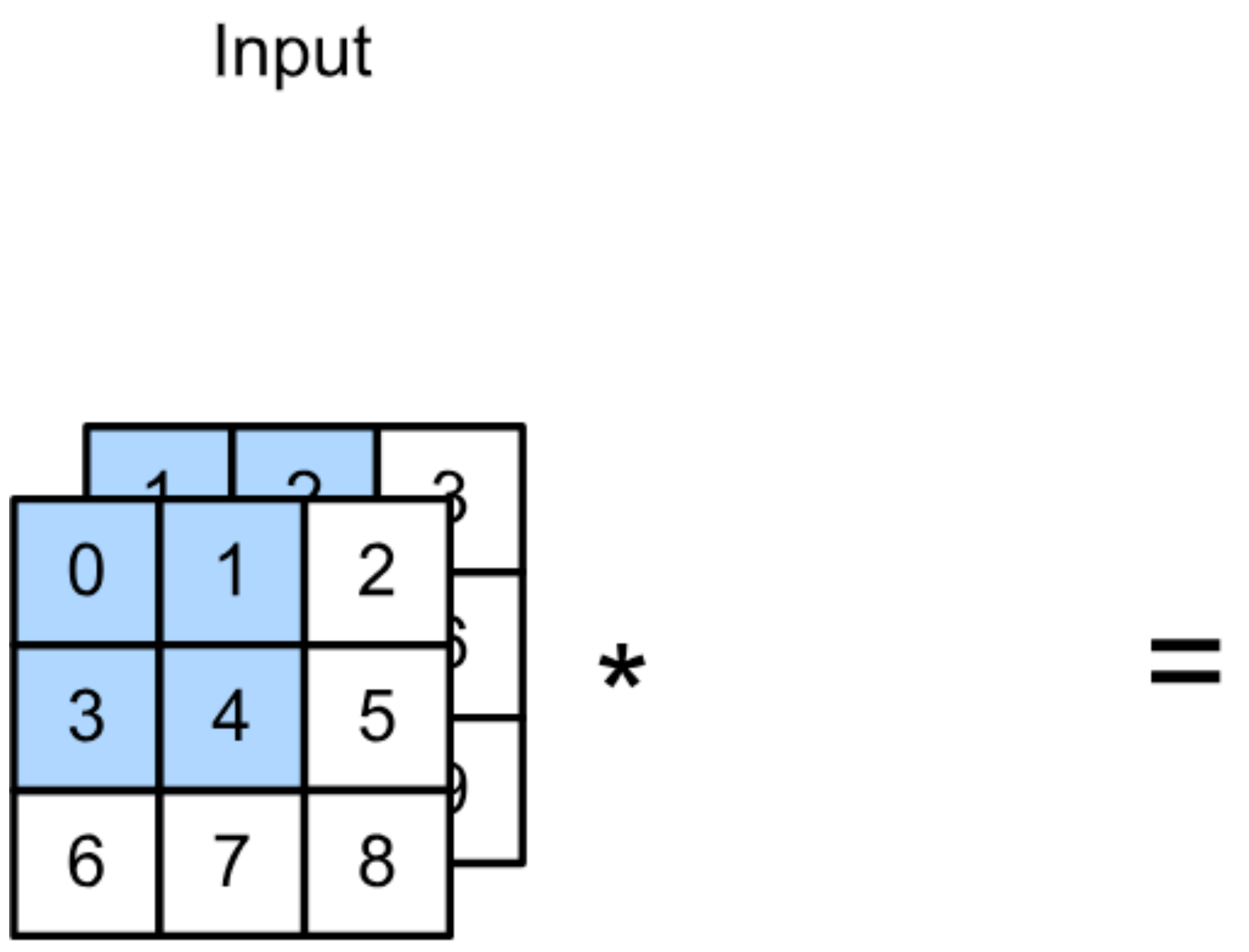
$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$



(vdumoulin@ Github)

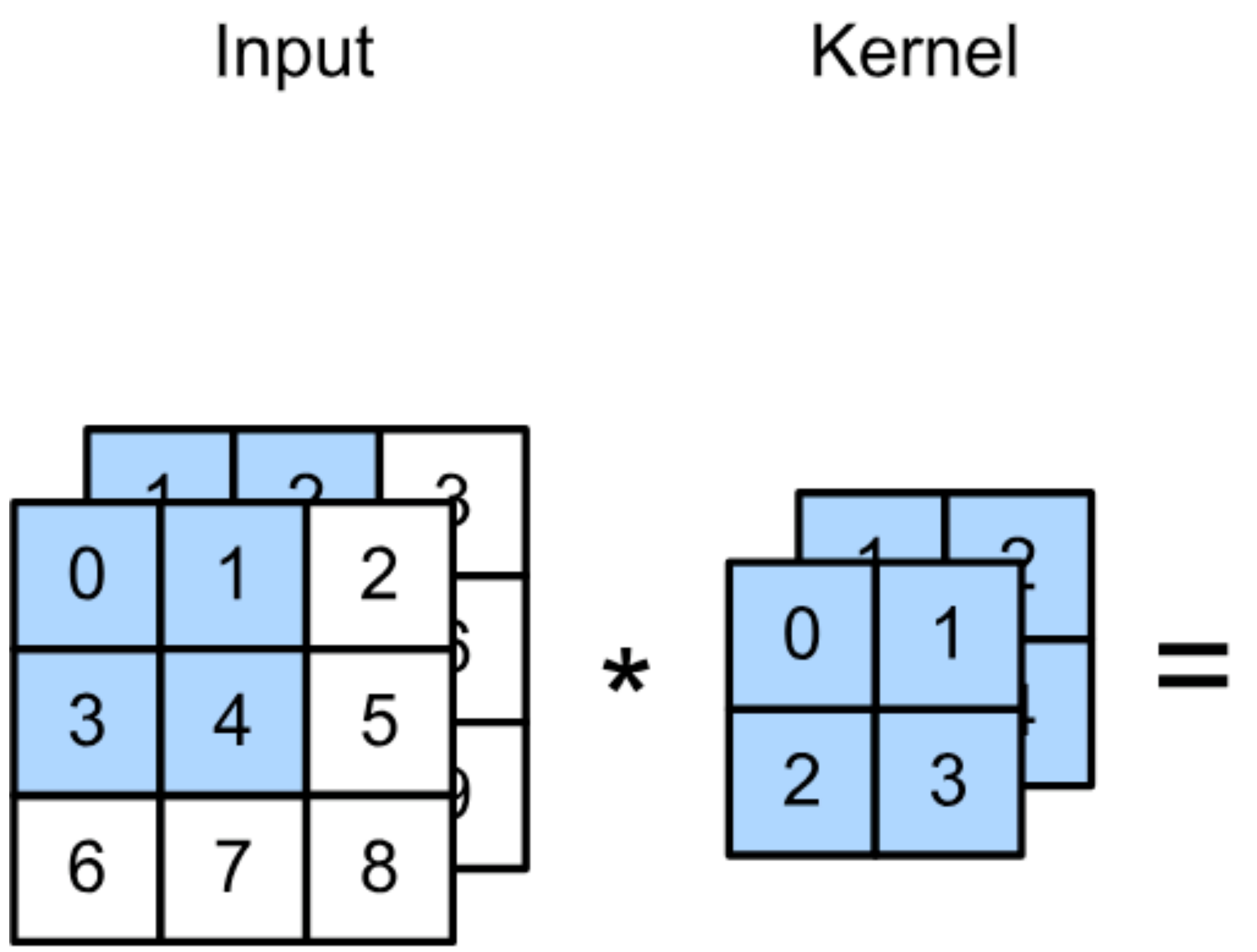
Review: Multiple Input Channels

- 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



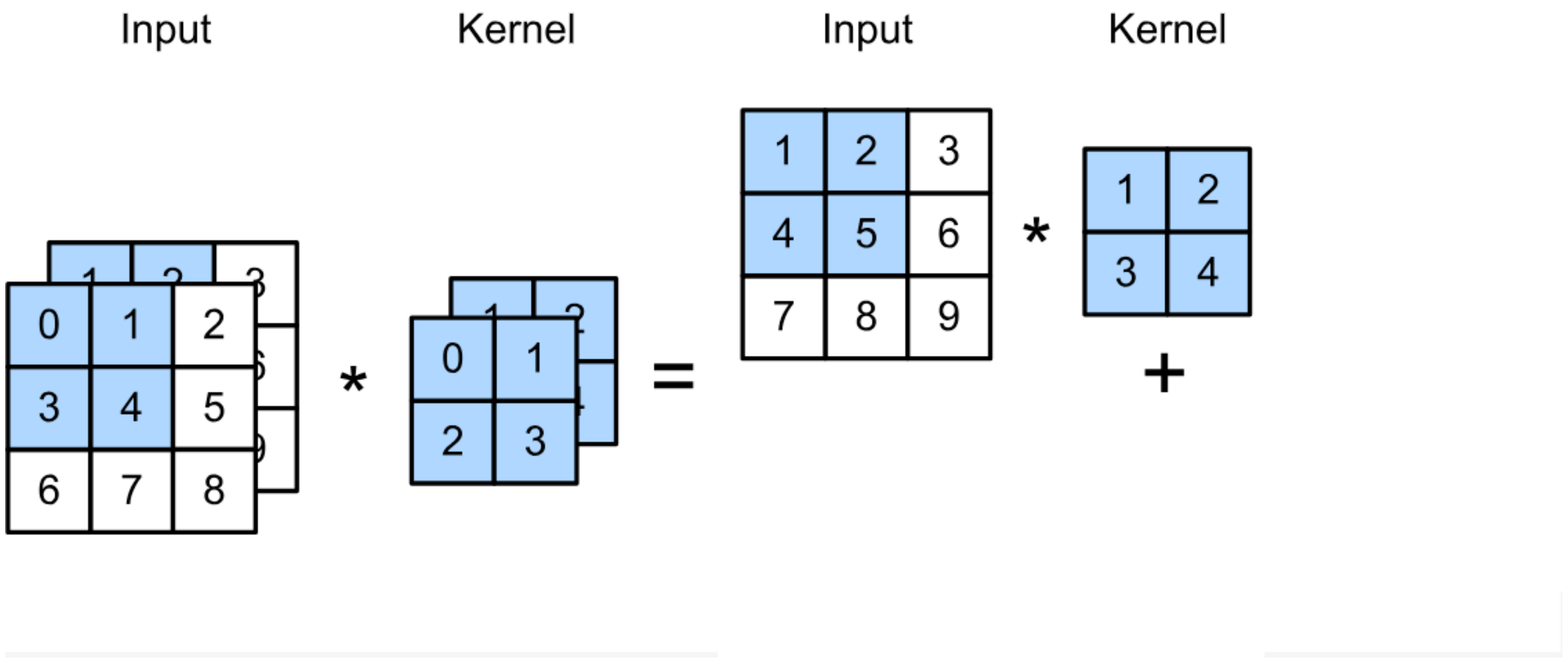
Review: Multiple Input Channels

- 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

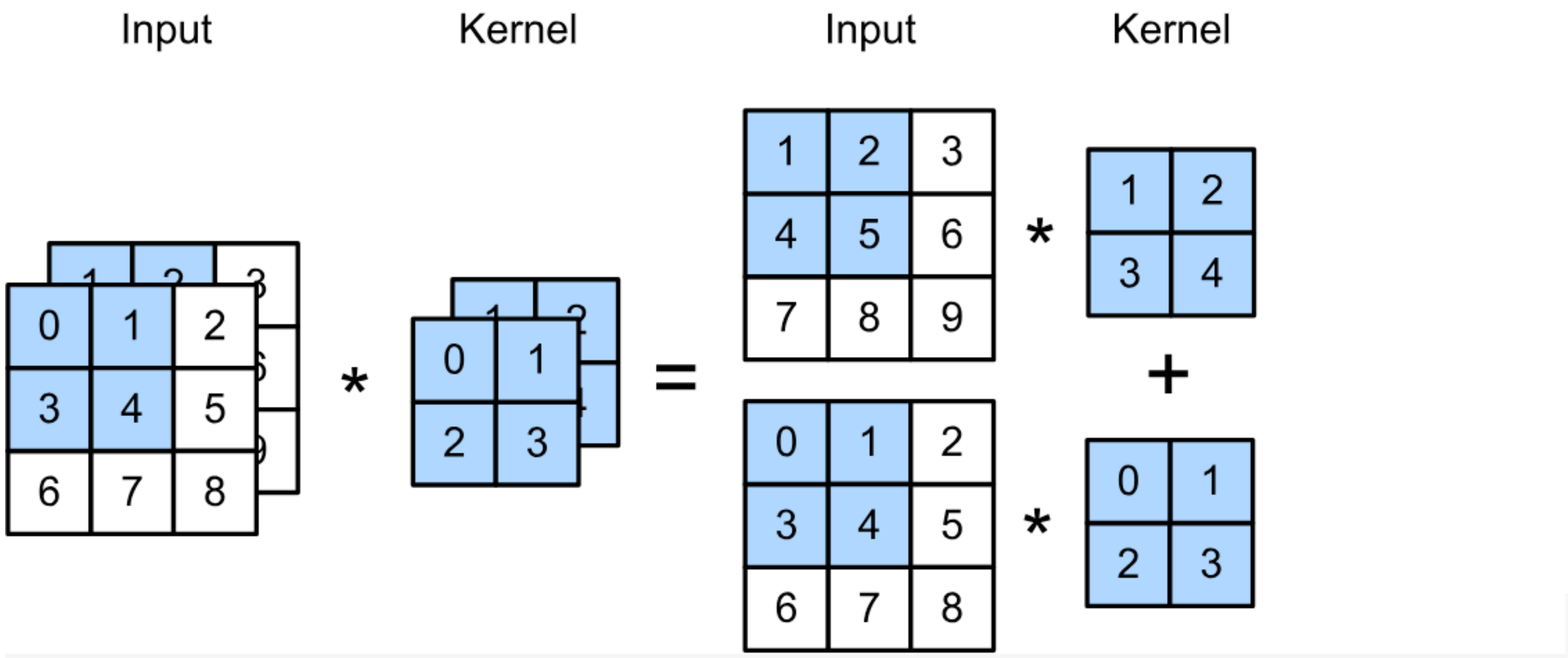
- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels



)

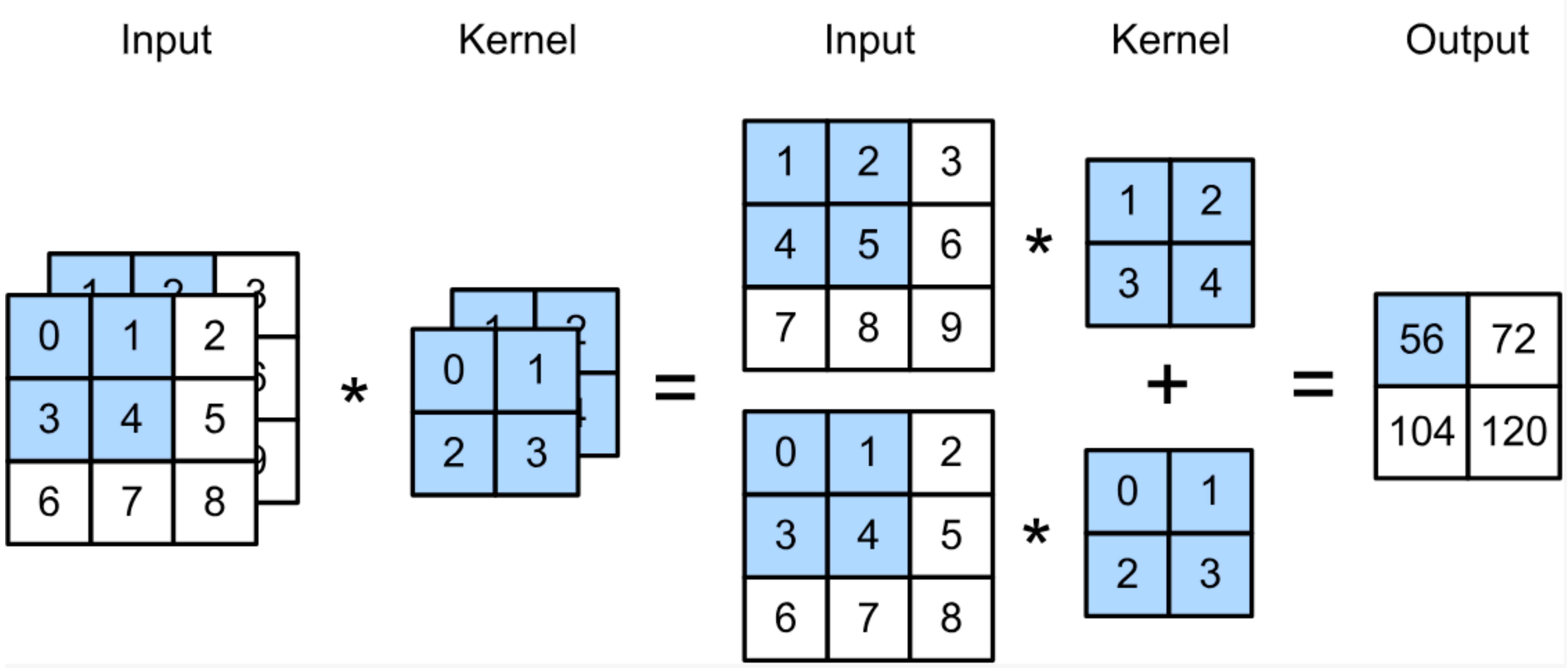
Review: Multiple Input Channels

- 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



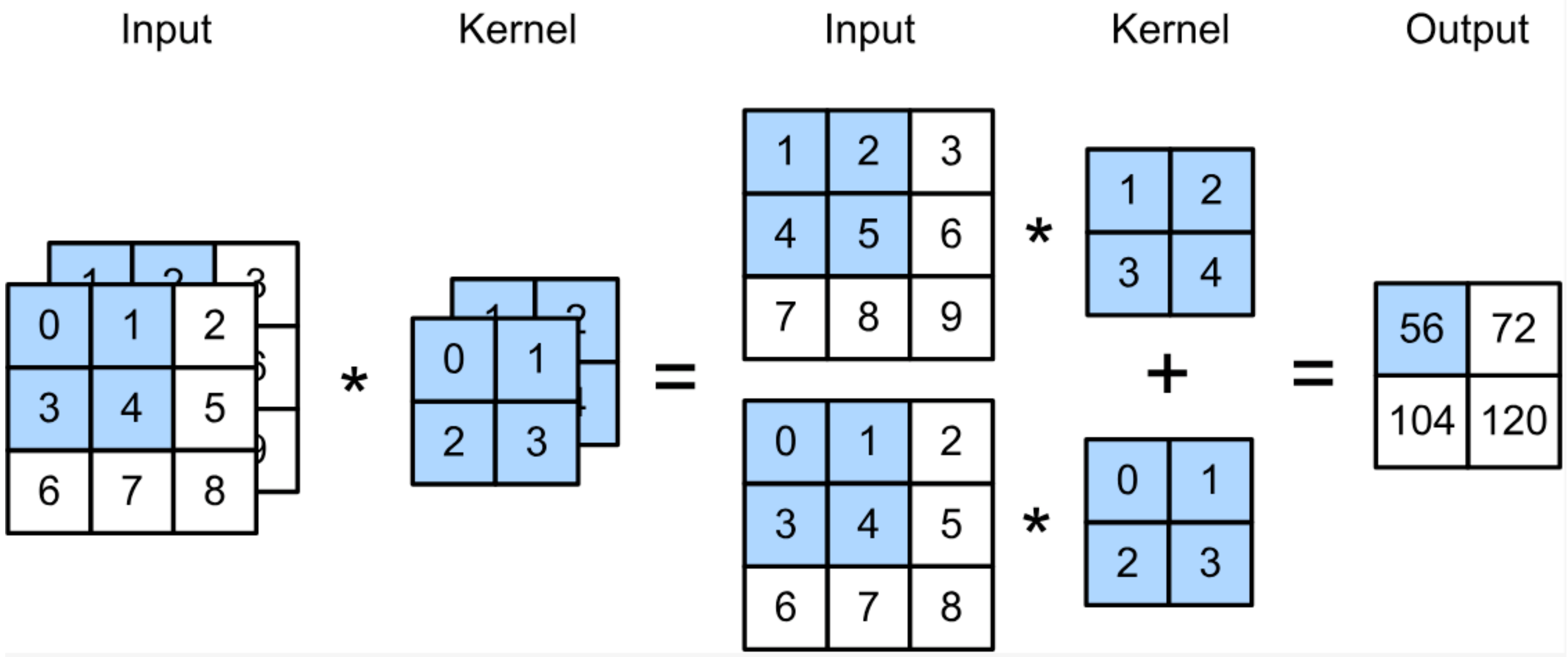
Review: Multiple Input Channels

- 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

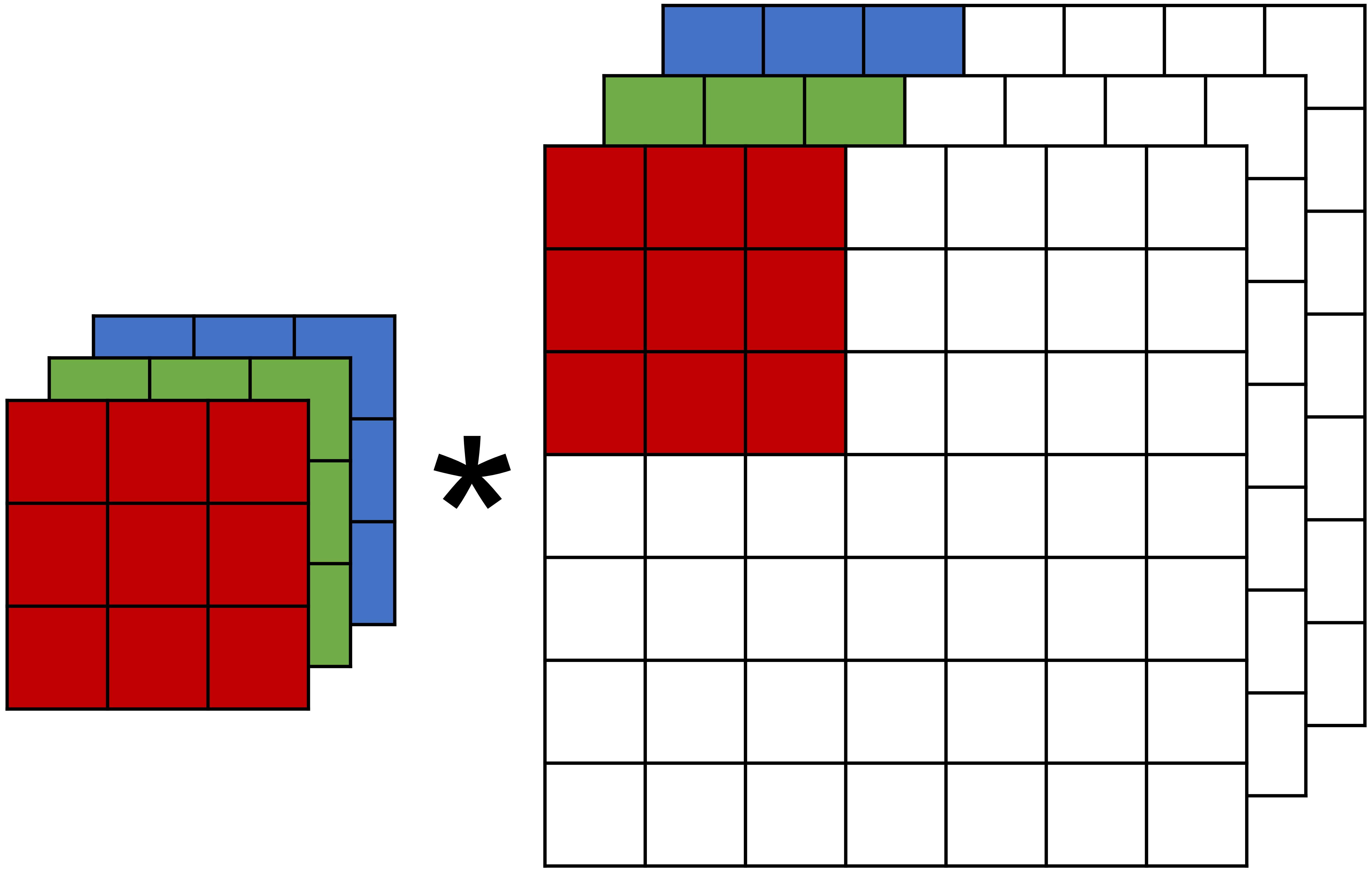
- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels



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

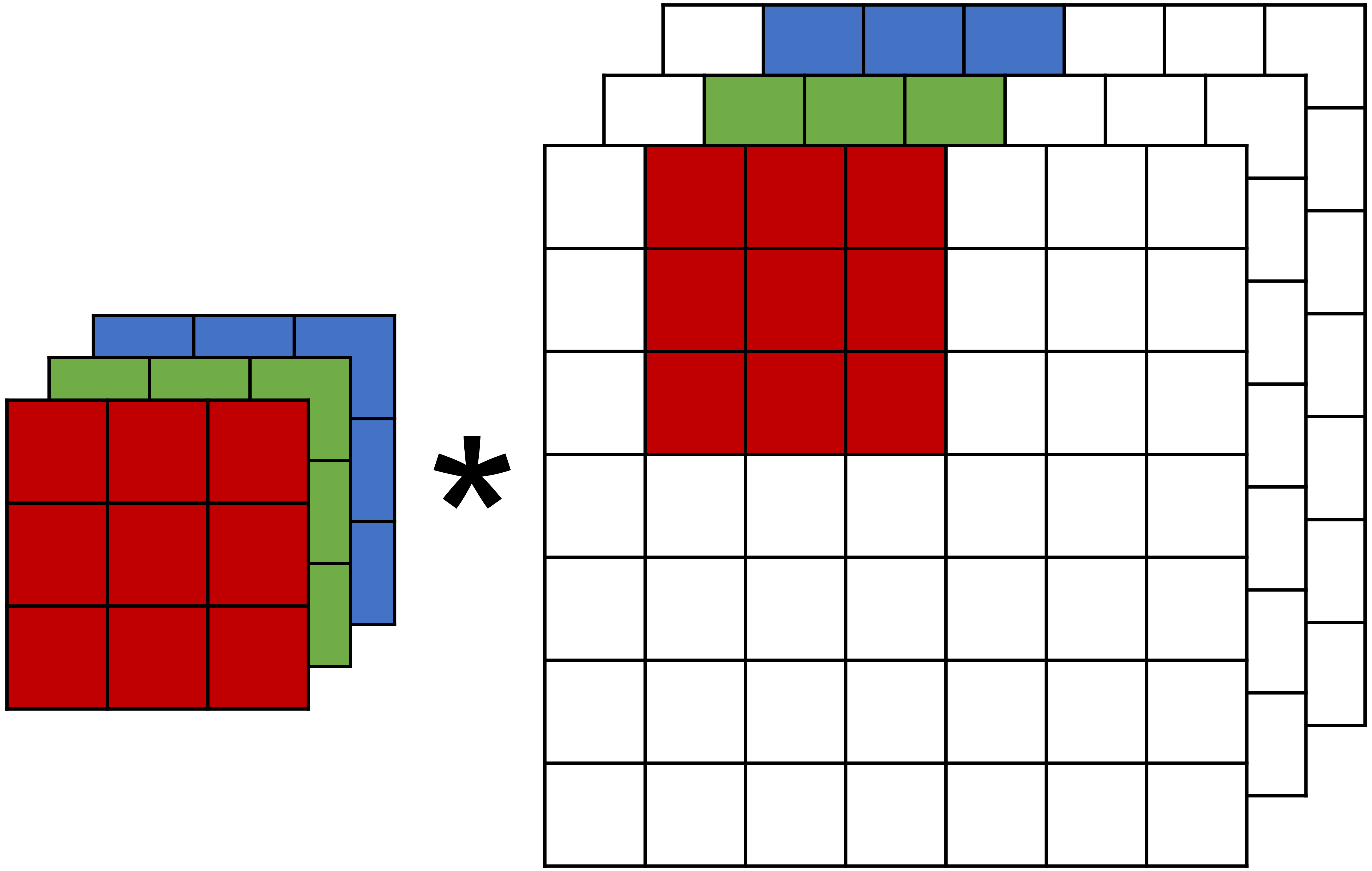
Review: Multiple Input Channels

- 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



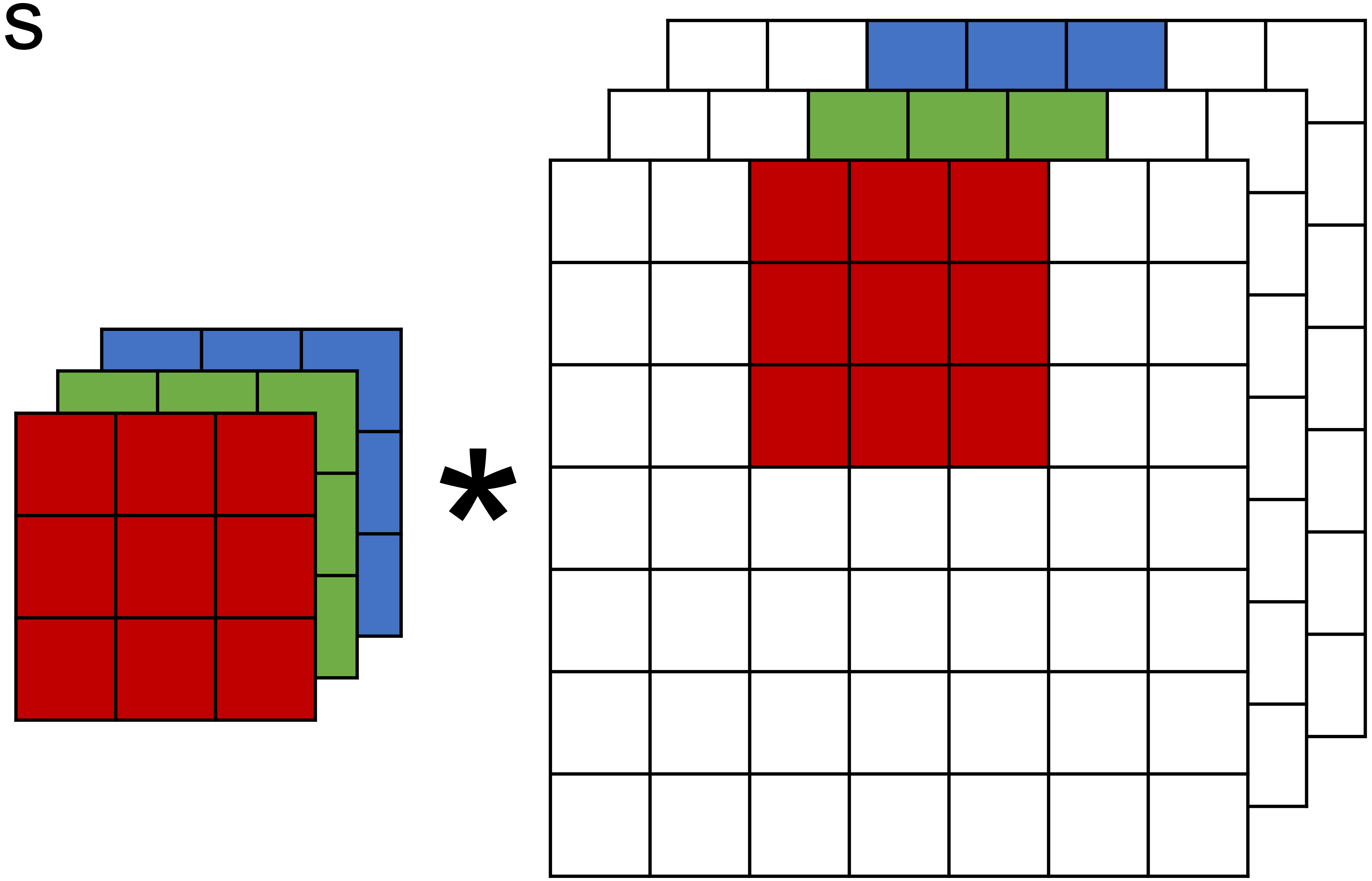
Review: Multiple Input Channels

- 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



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

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?

A. 11 x 11 x 16
B. 6 x 6 x 16
C. 7 x 7 x16
D. 5 x 5 x16

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?

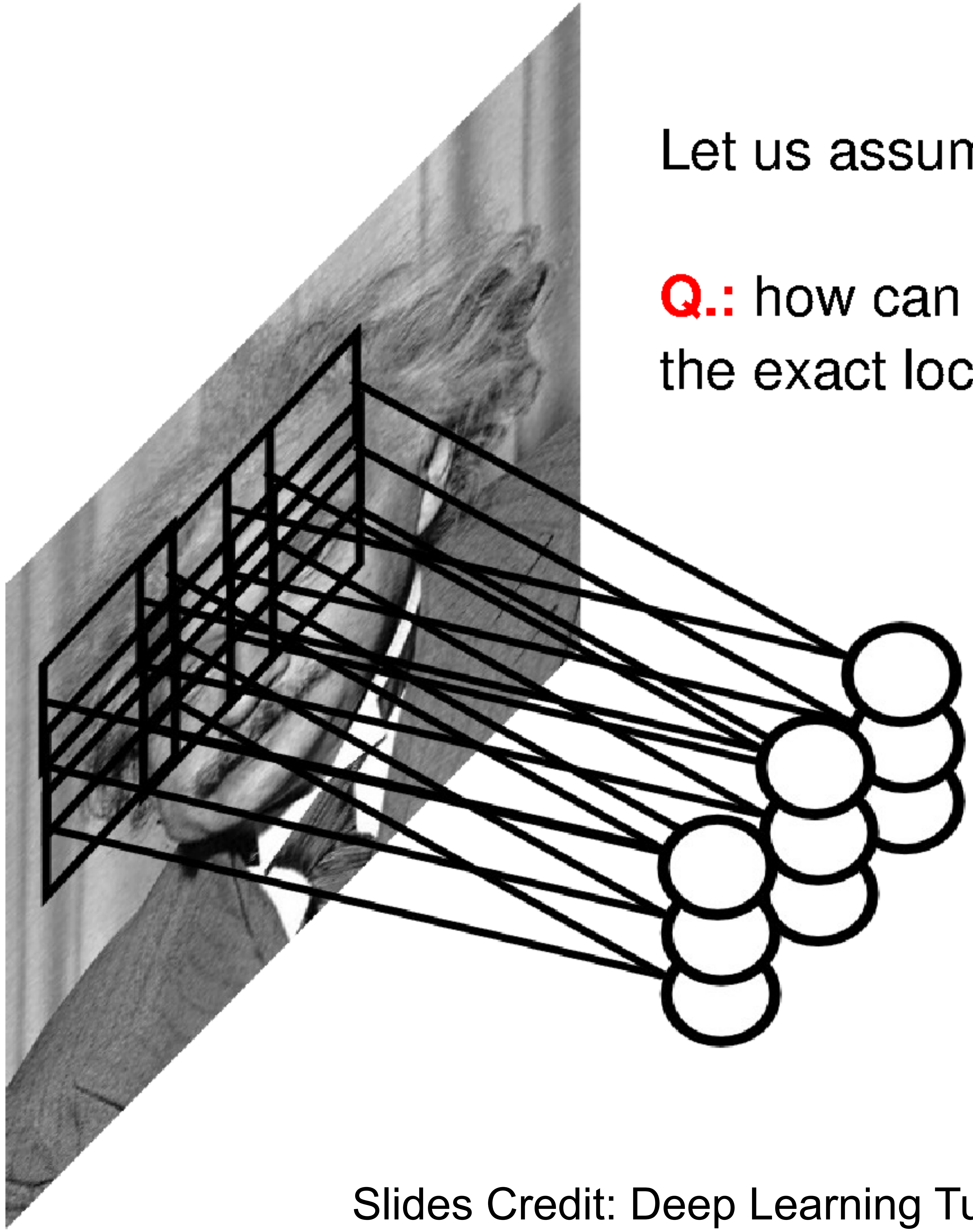
A. 11 x 11 x 16
B. 6 x 6 x 16
C. 7 x 7 x16
D. 5 x 5 x16

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

0 because filter not outside of input

Pooling Layer

Pooling

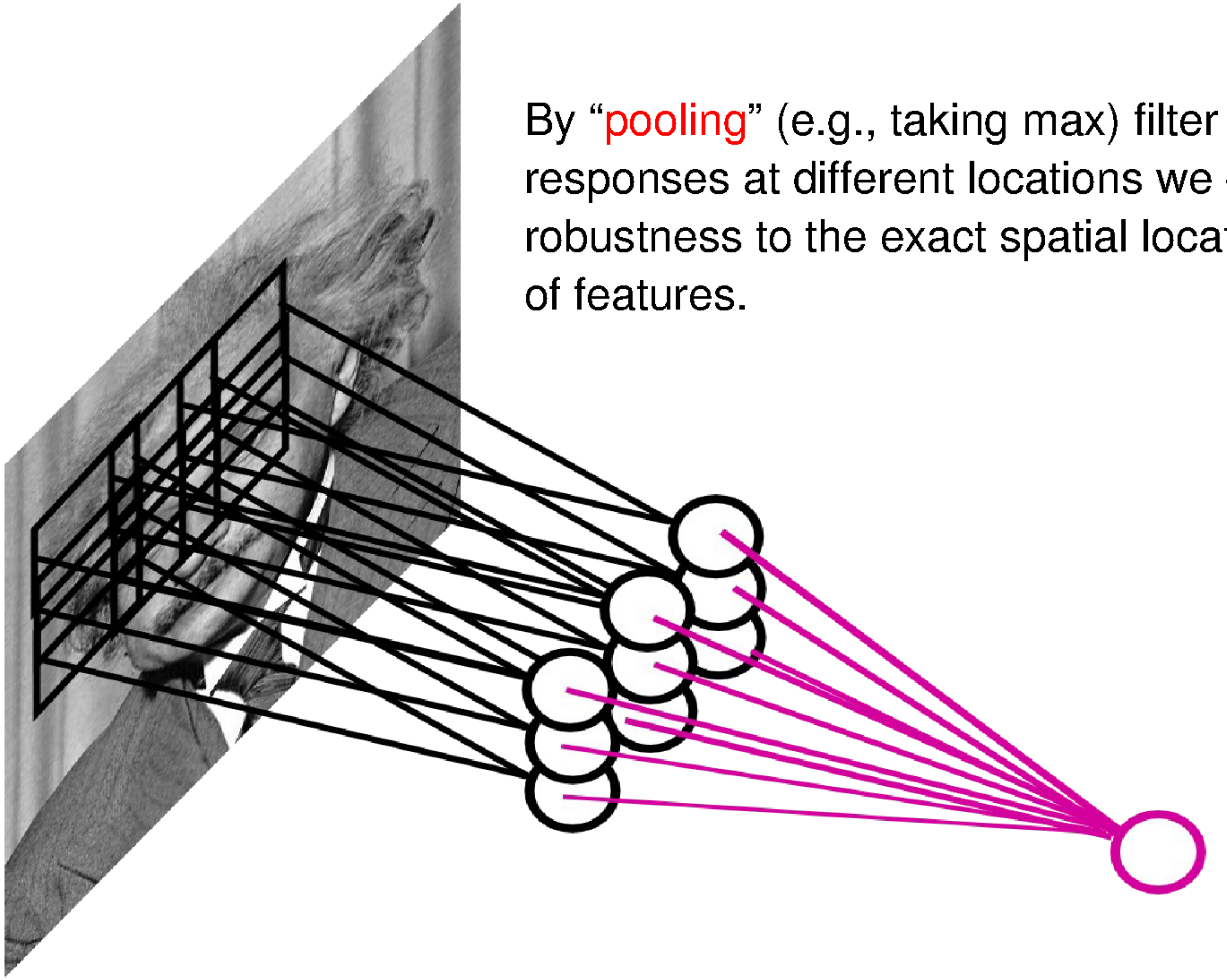


Let us assume filter is an “eye” detector.

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

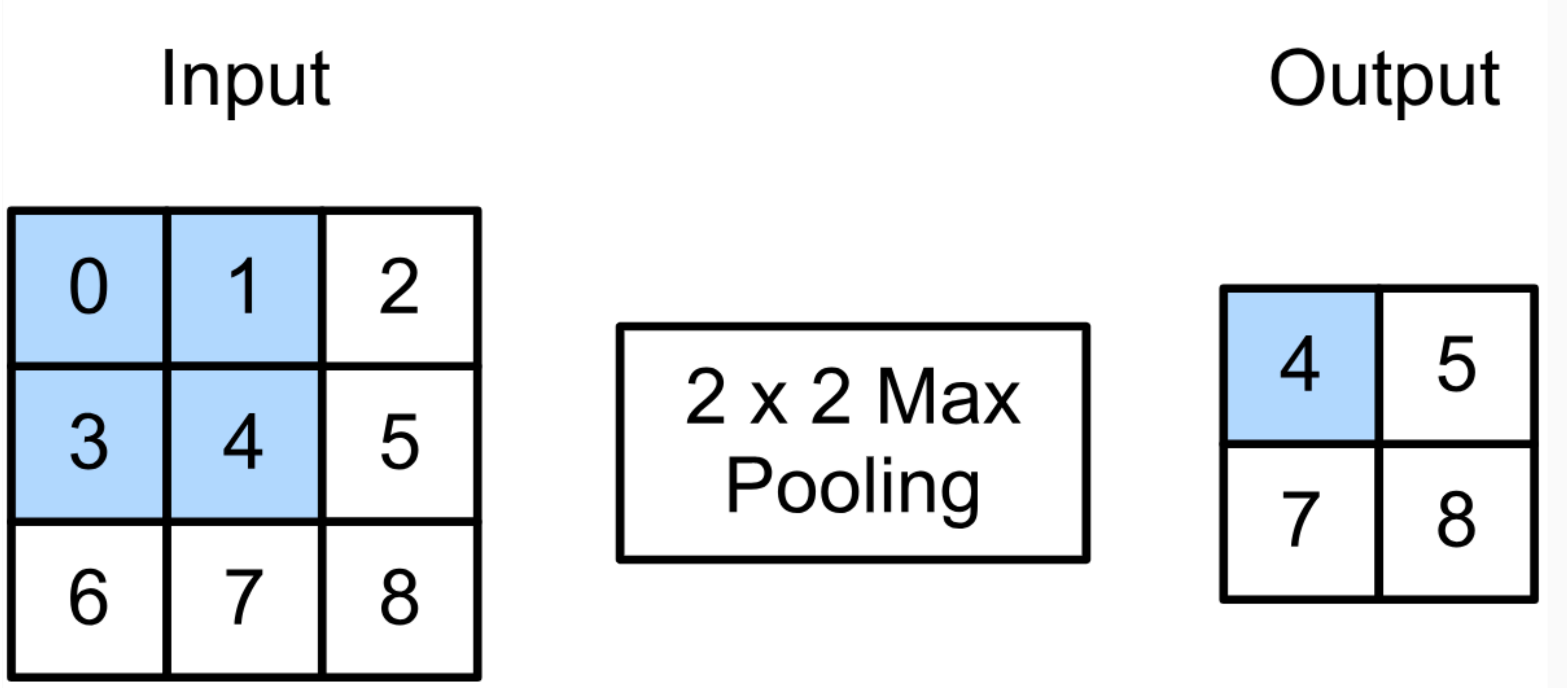


By "pooling" (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

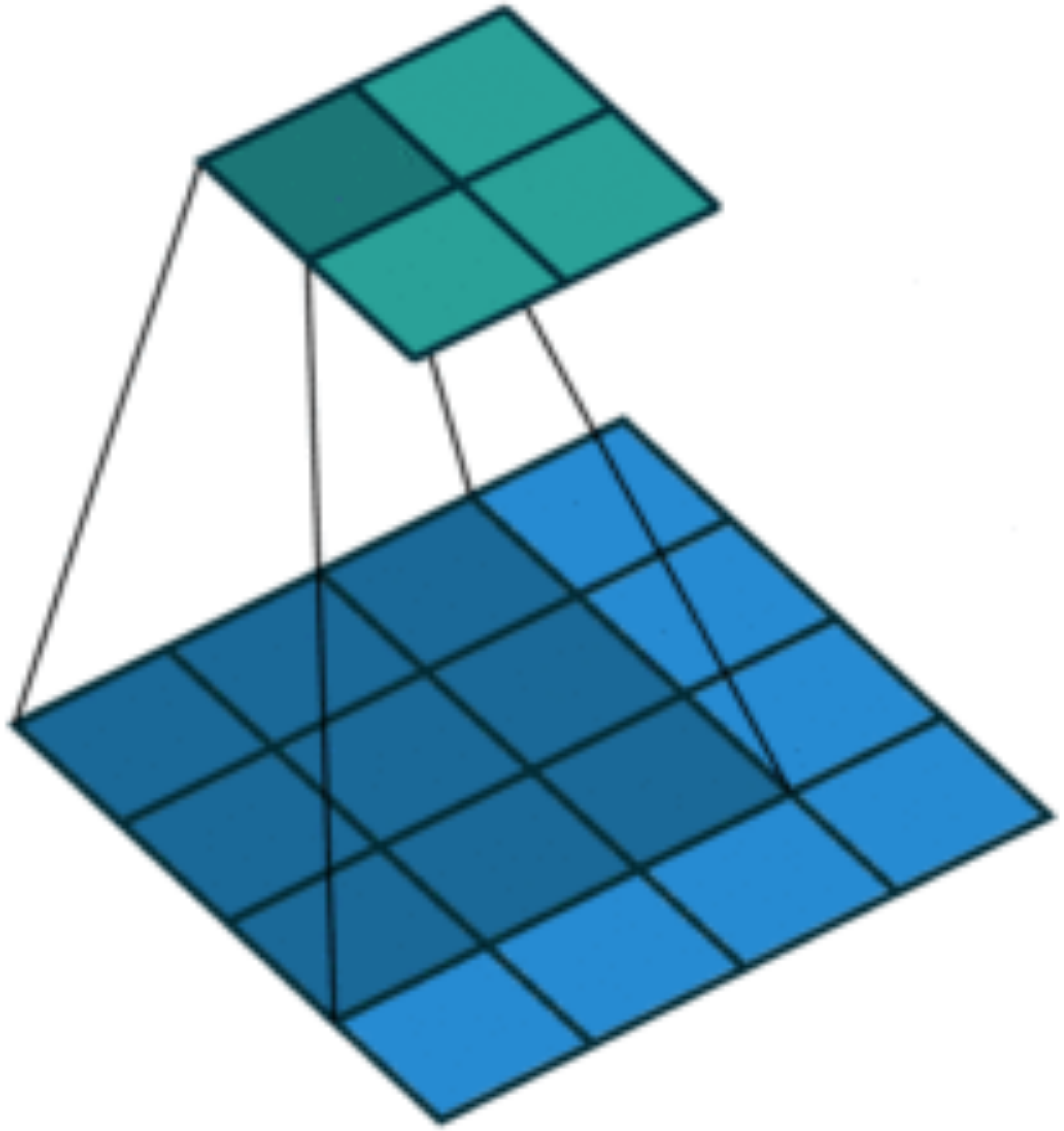
Slides Credit: Deep Learning Tutorial by Marc'Aurelio Ranzato

2-D Max Pooling

- Returns the maximal value in the sliding window

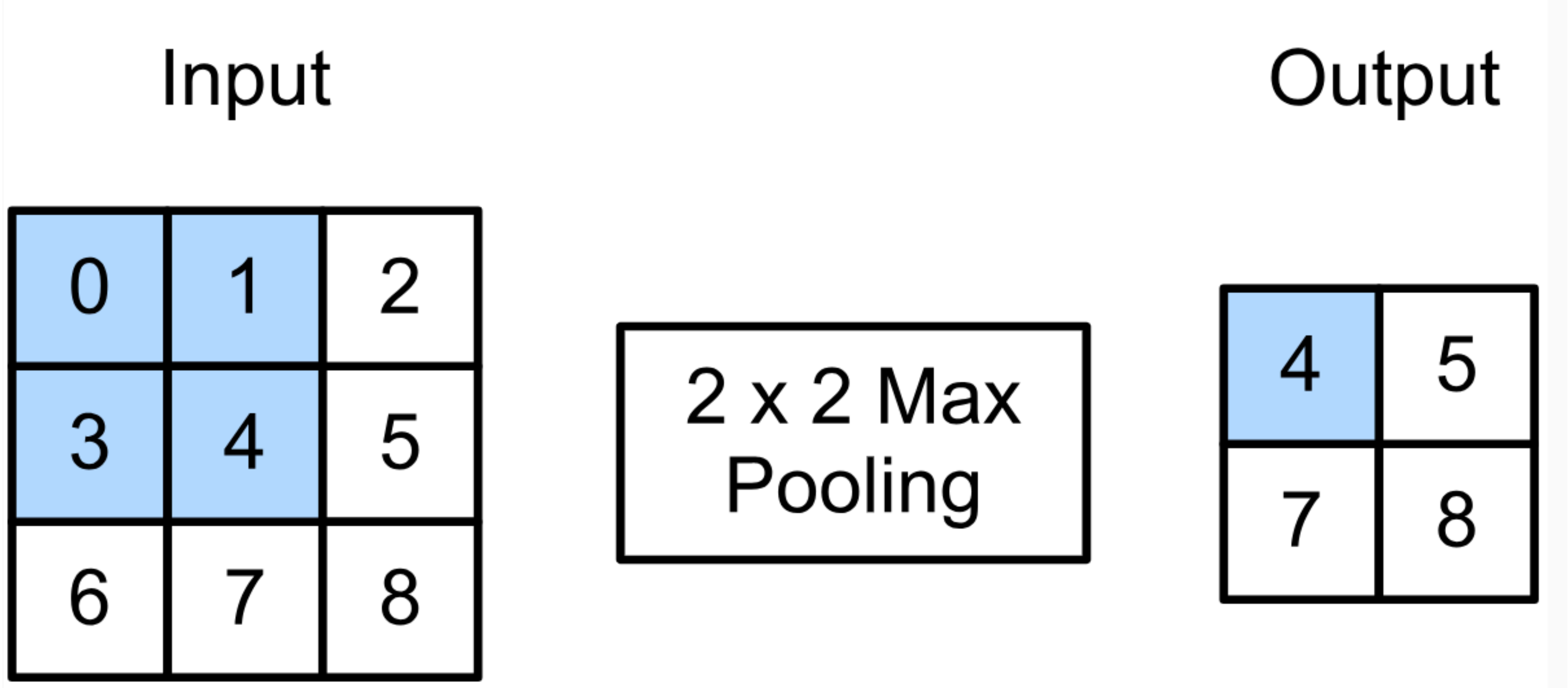


$$\max(0,1,3,4) = 4$$

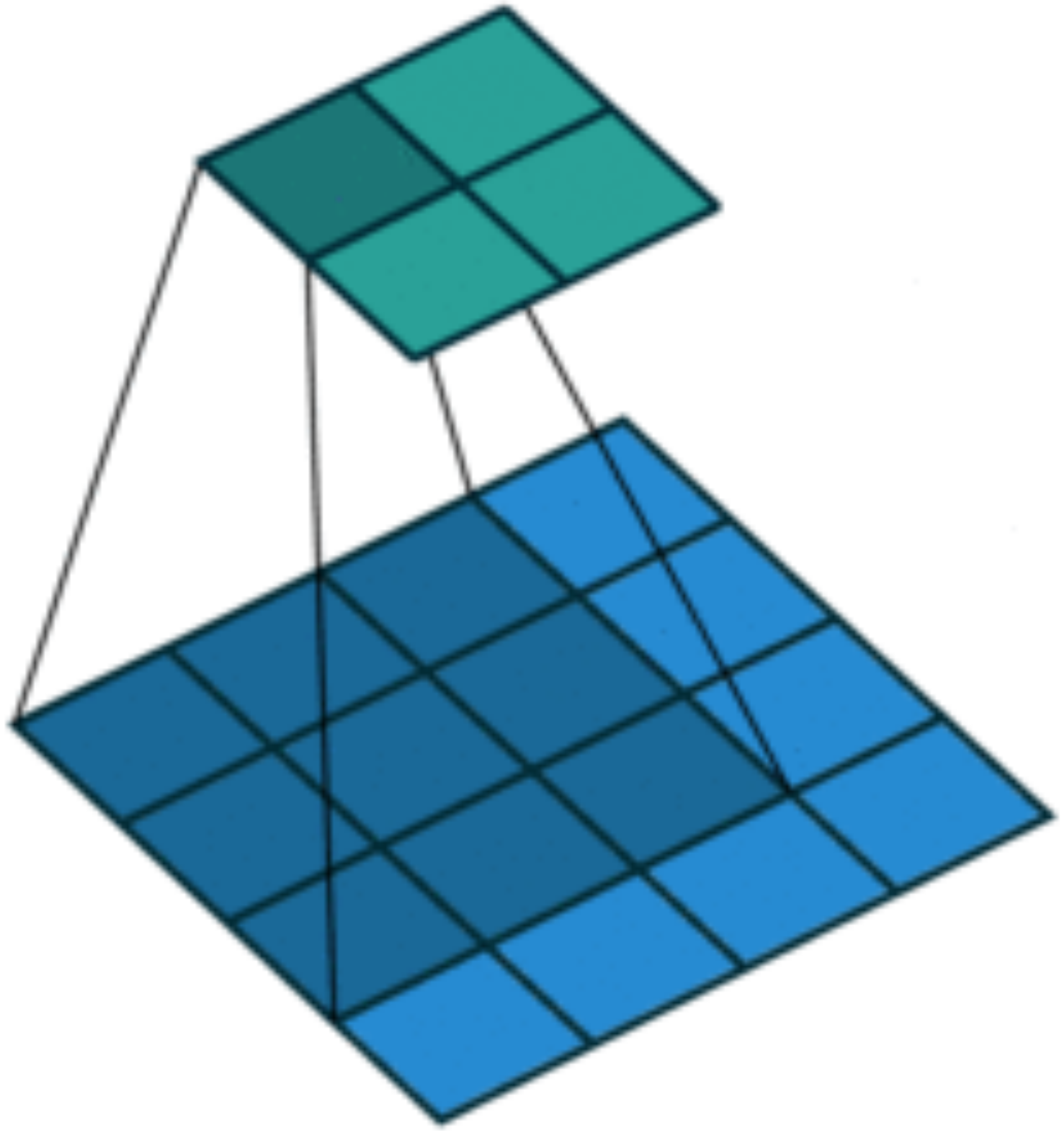


2-D Max Pooling

- Returns the maximal value in the sliding window



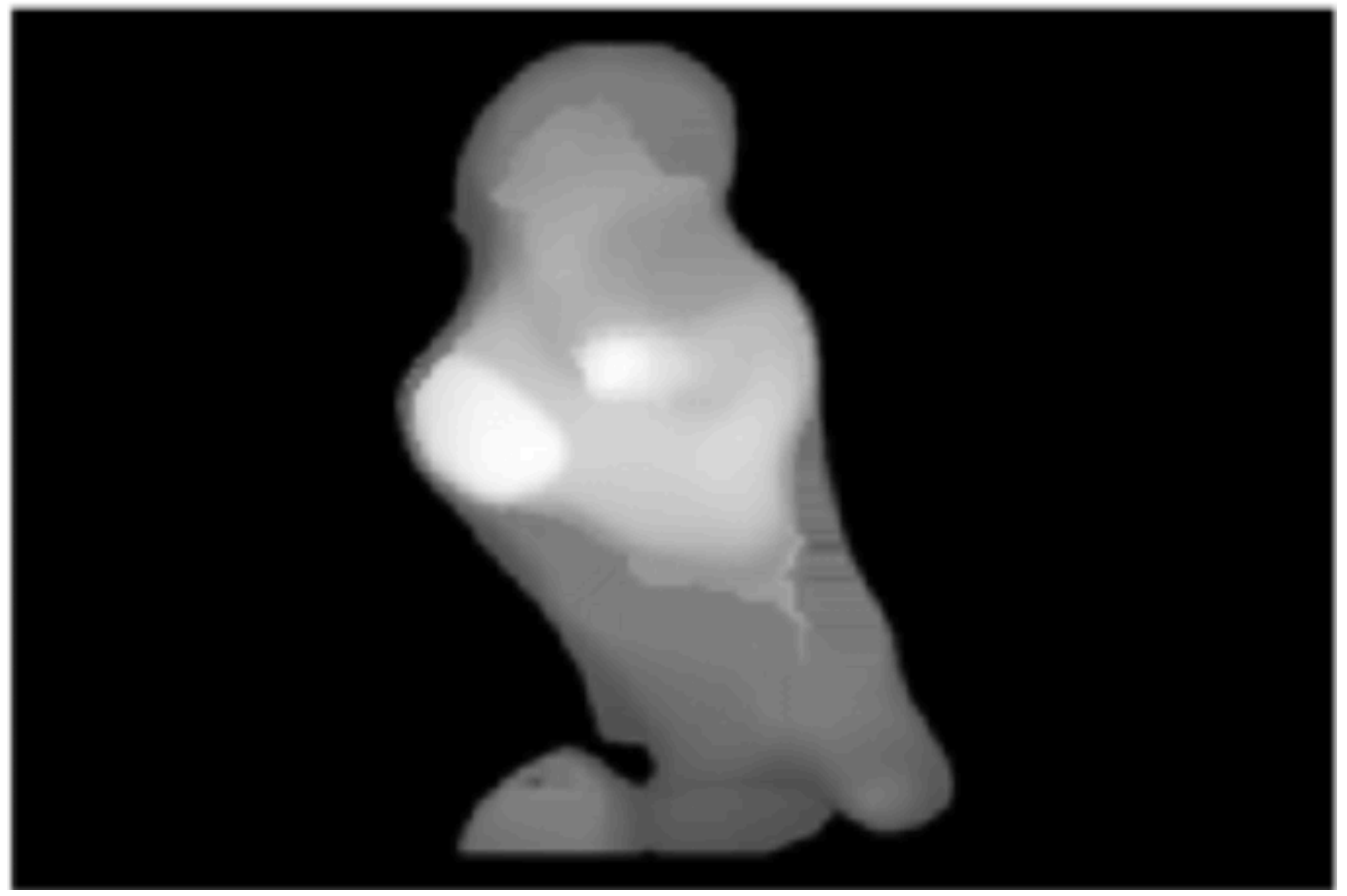
$$\max(0,1,3,4) = 4$$



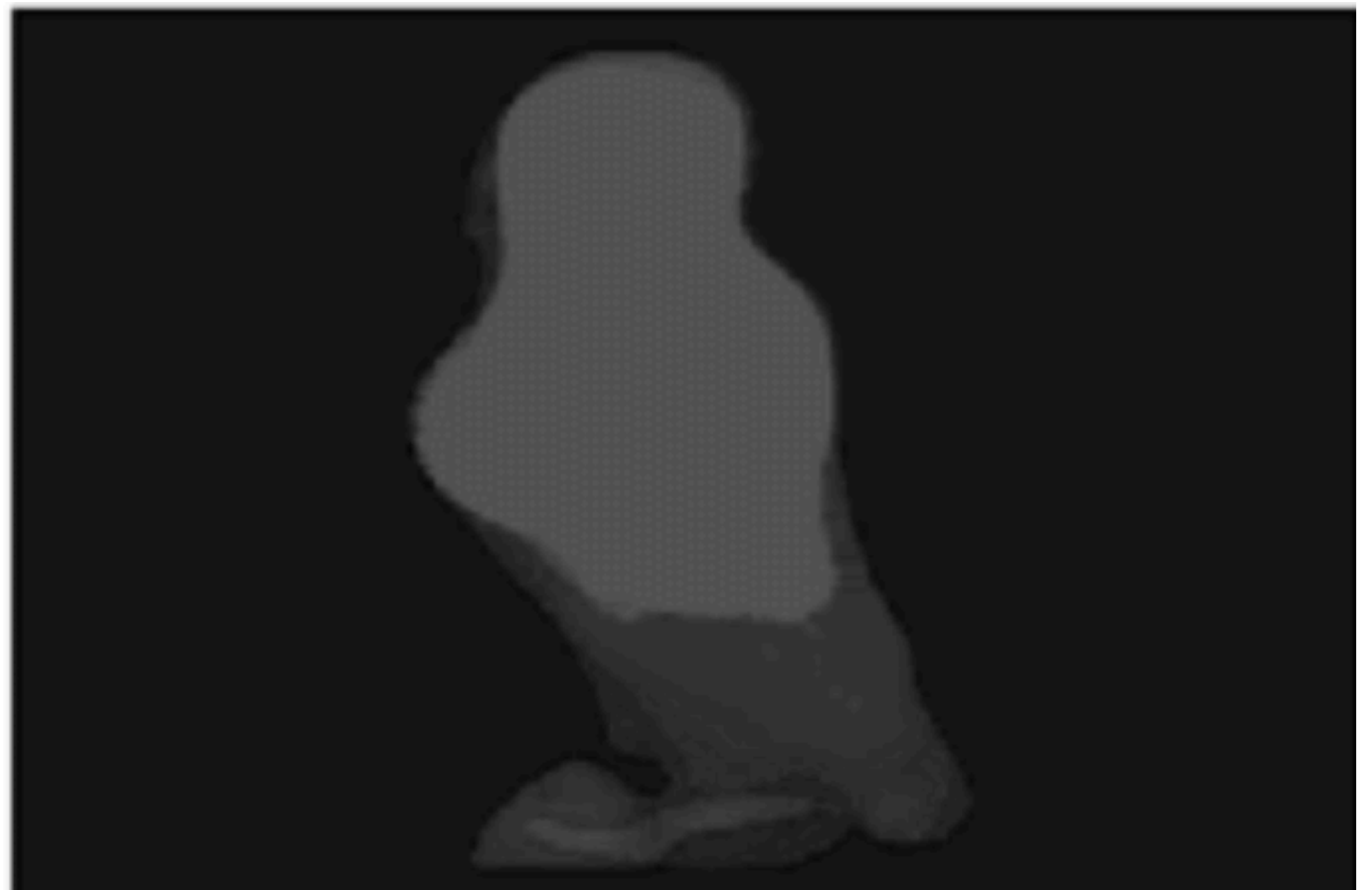
Average Pooling

- 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

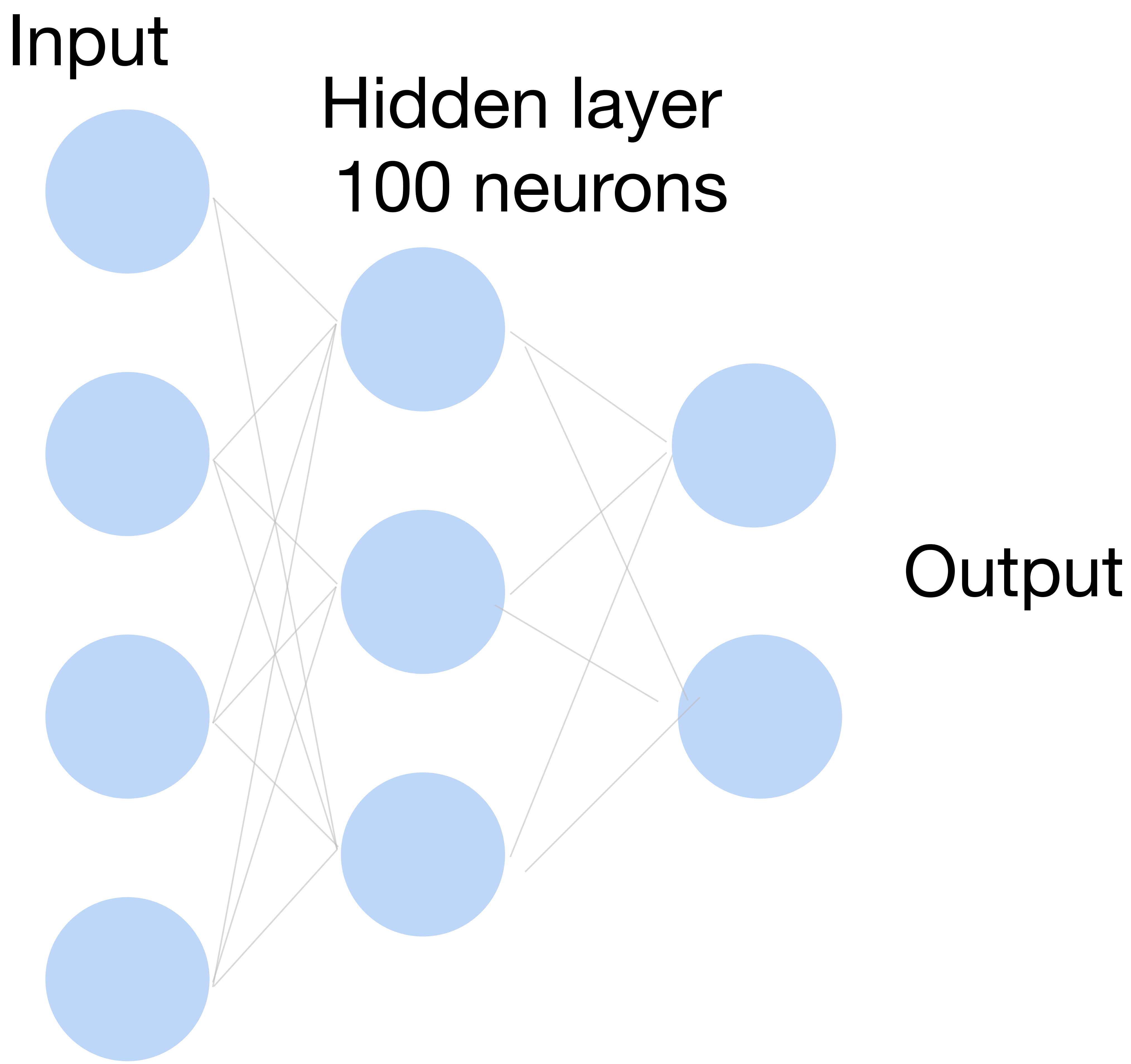


Average pooling



How to train a neural network?

Loss function: $\frac{1}{|D|} \sum_i \ell(\mathbf{x}_i, y_i)$

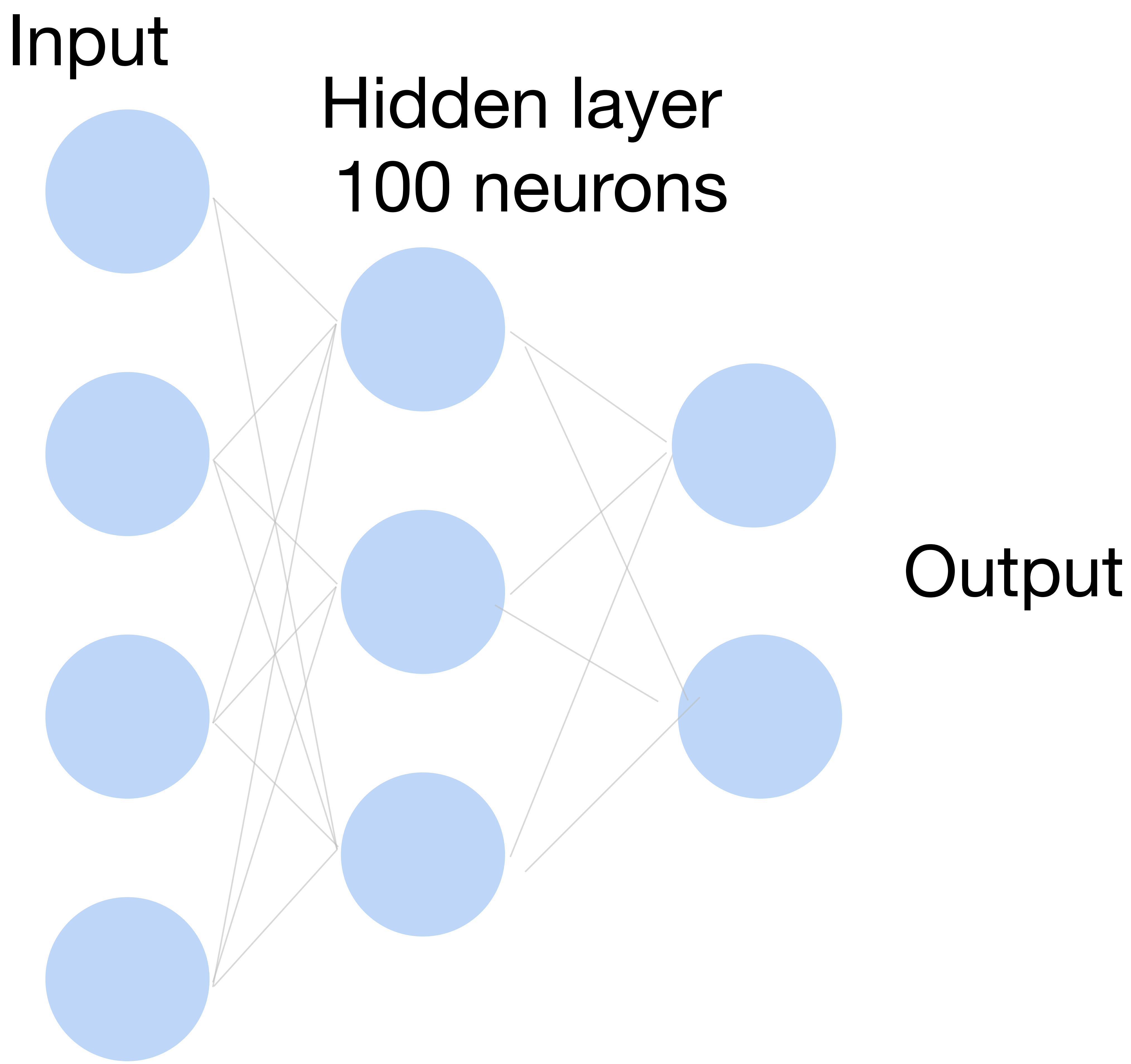


How to train a neural network?

Loss function: $\dfrac{1}{|D|}\sum_i \ell(\mathbf{x}_i, y_i)$

Per-sample loss:

$$\ell(\mathbf{x}, y) = \sum_{j=1}^{K} -y_j \log p_j$$



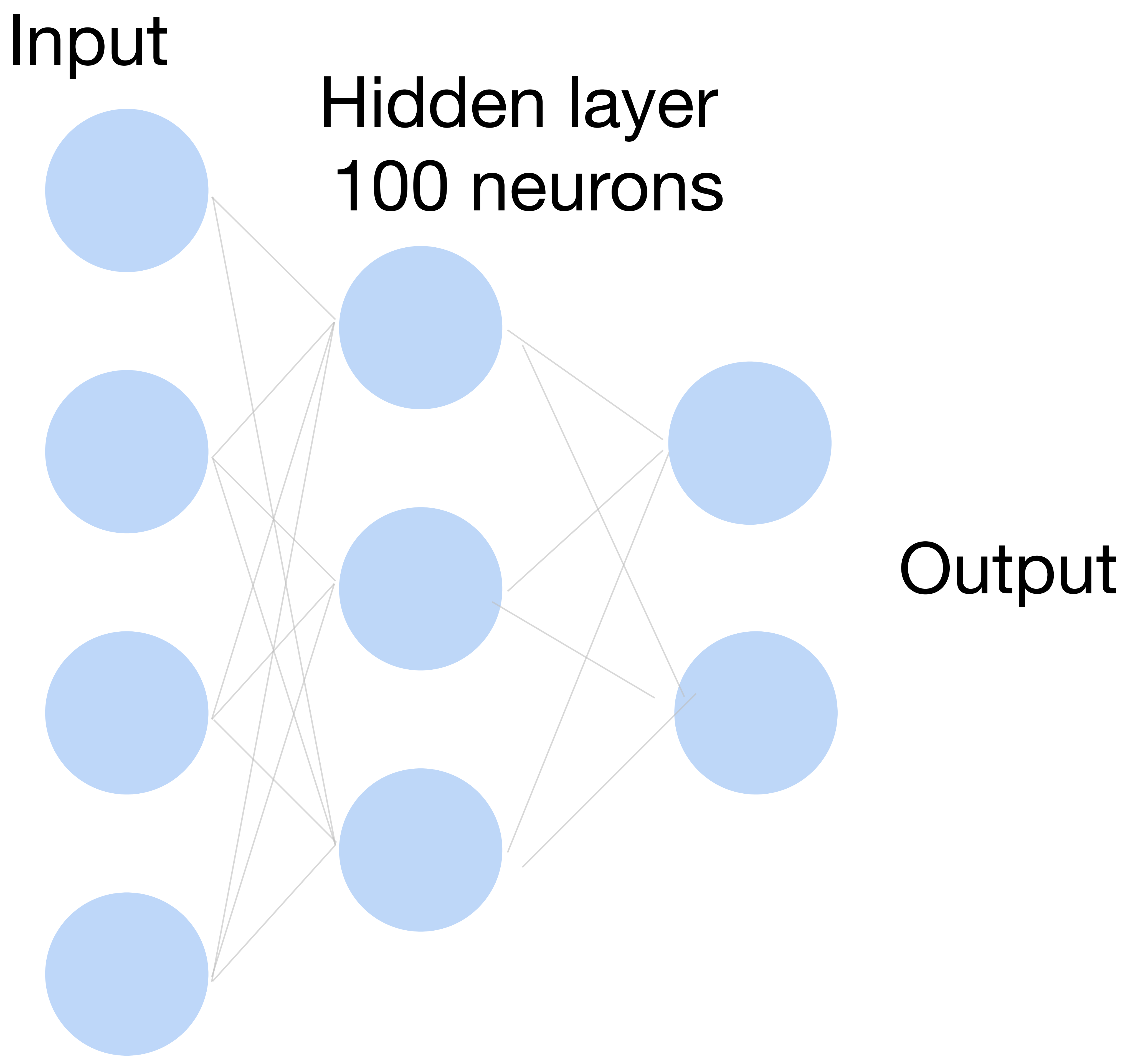
How to train a neural network?

Loss function: $$\frac{1}{|D|}\sum_i \ell(\mathbf{x}_i, y_i)$$

Per-sample loss:

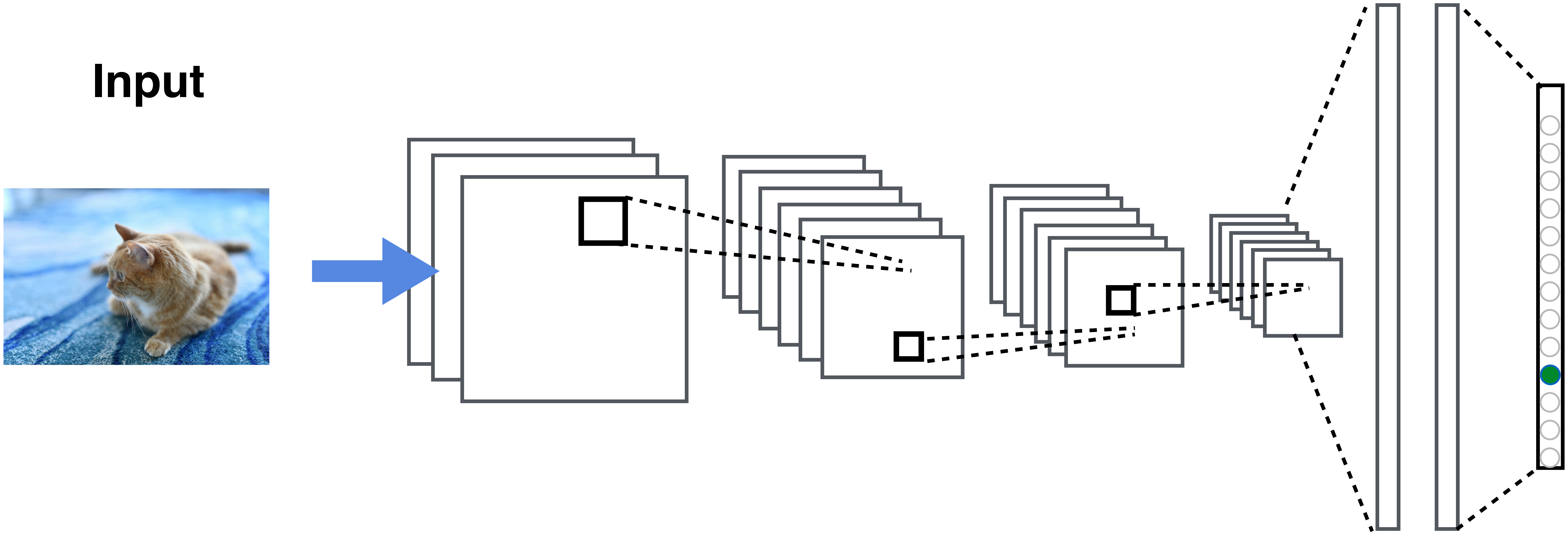
$$\ell(\mathbf{x}, y) = \sum_{j=1}^{K} -y_j \log p_j$$

Also known as cross-entropy loss or softmax loss

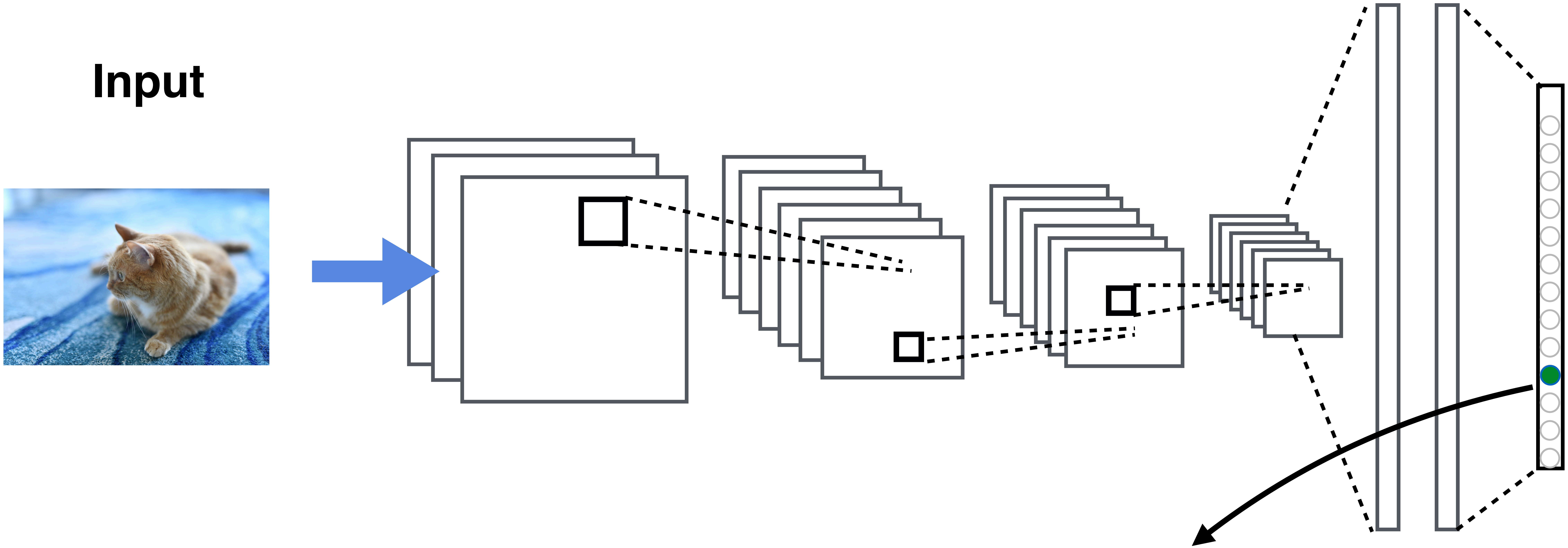


How to train a convolutional neural network?

Input



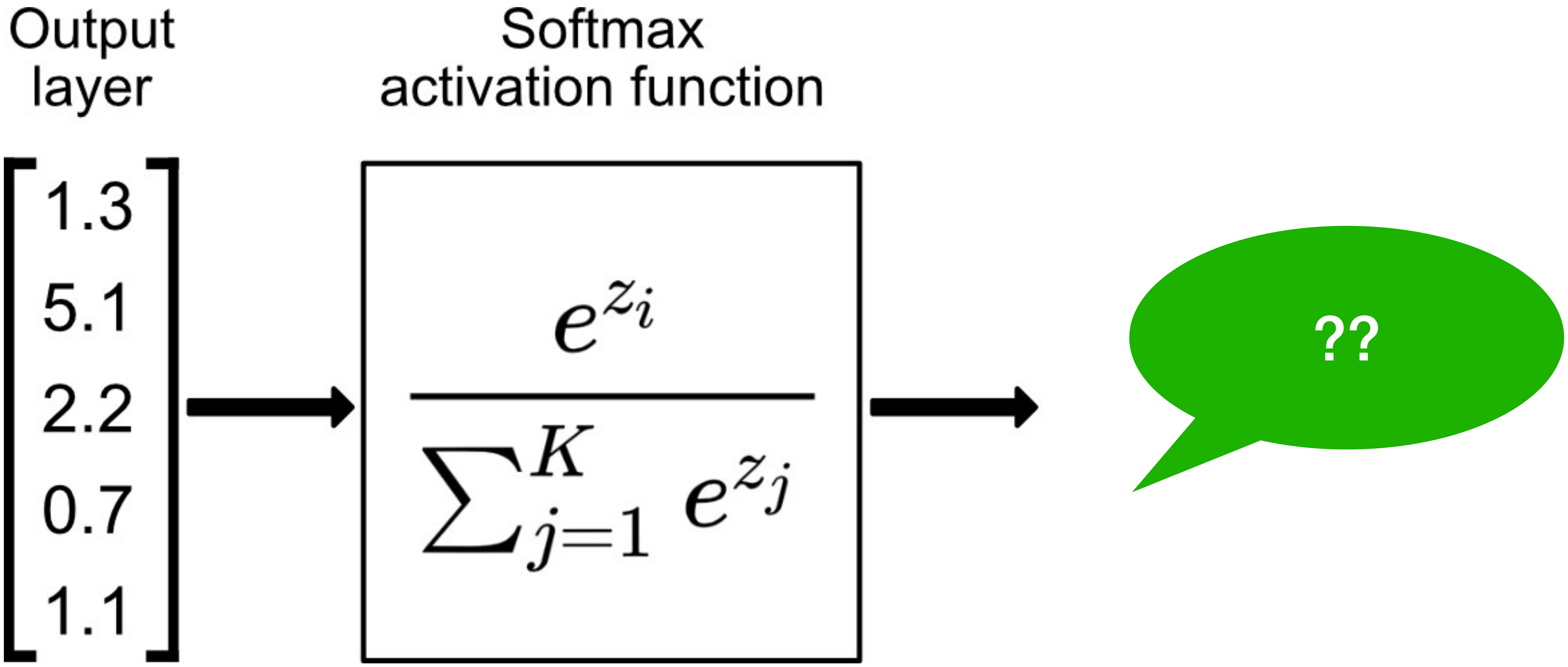
How to train a convolutional neural network?



$$p_i(\boldsymbol{x}) = \frac{\exp\left(f_i(\boldsymbol{x})\right)}{\sum_{j=1}^{N} \exp\left(f_j(\boldsymbol{x})\right)}, \quad \text{softmax}$$

Recall Softmax

Turns outputs f into probabilities (sum up to 1 across k classes)



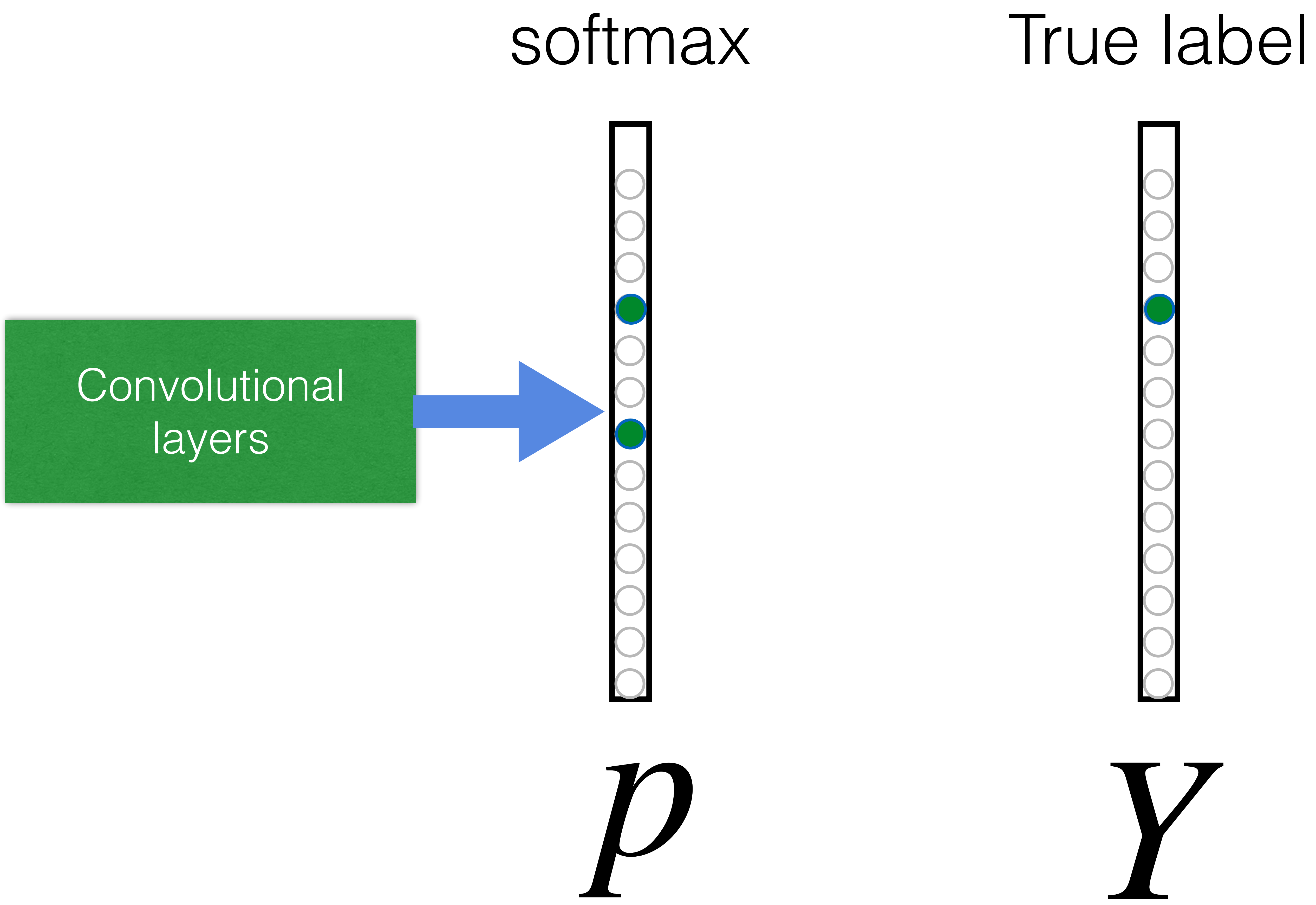
Recall Softmax

Turns outputs f into probabilities (sum up to 1 across k classes)

Output layer	Softmax activation function	Probabilities
$\begin{bmatrix} 1.3 \\ 5.1 \\ 2.2 \\ 0.7 \\ 1.1 \end{bmatrix}$	$\longrightarrow \dfrac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \longrightarrow$	$\begin{bmatrix} 0.02 \\ 0.90 \\ 0.05 \\ 0.01 \\ 0.02 \end{bmatrix}$

Normalized

Cross-Entropy Loss

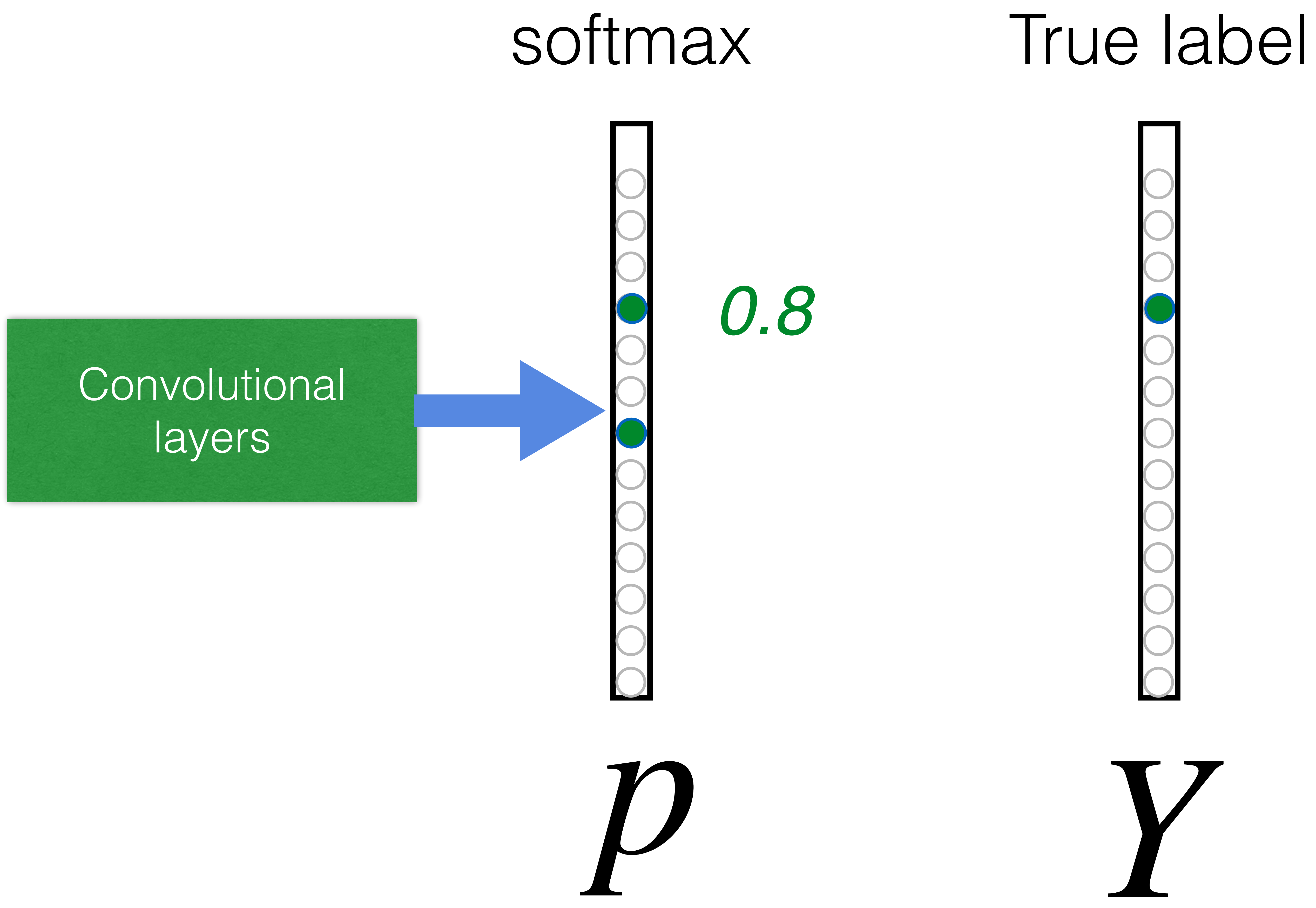


$$L_{CE} = \sum_i -Y_i \log(p_i)$$

$$= -\log(0.8)$$

Goal: push **p** and **Y** to be identical

Cross-Entropy Loss

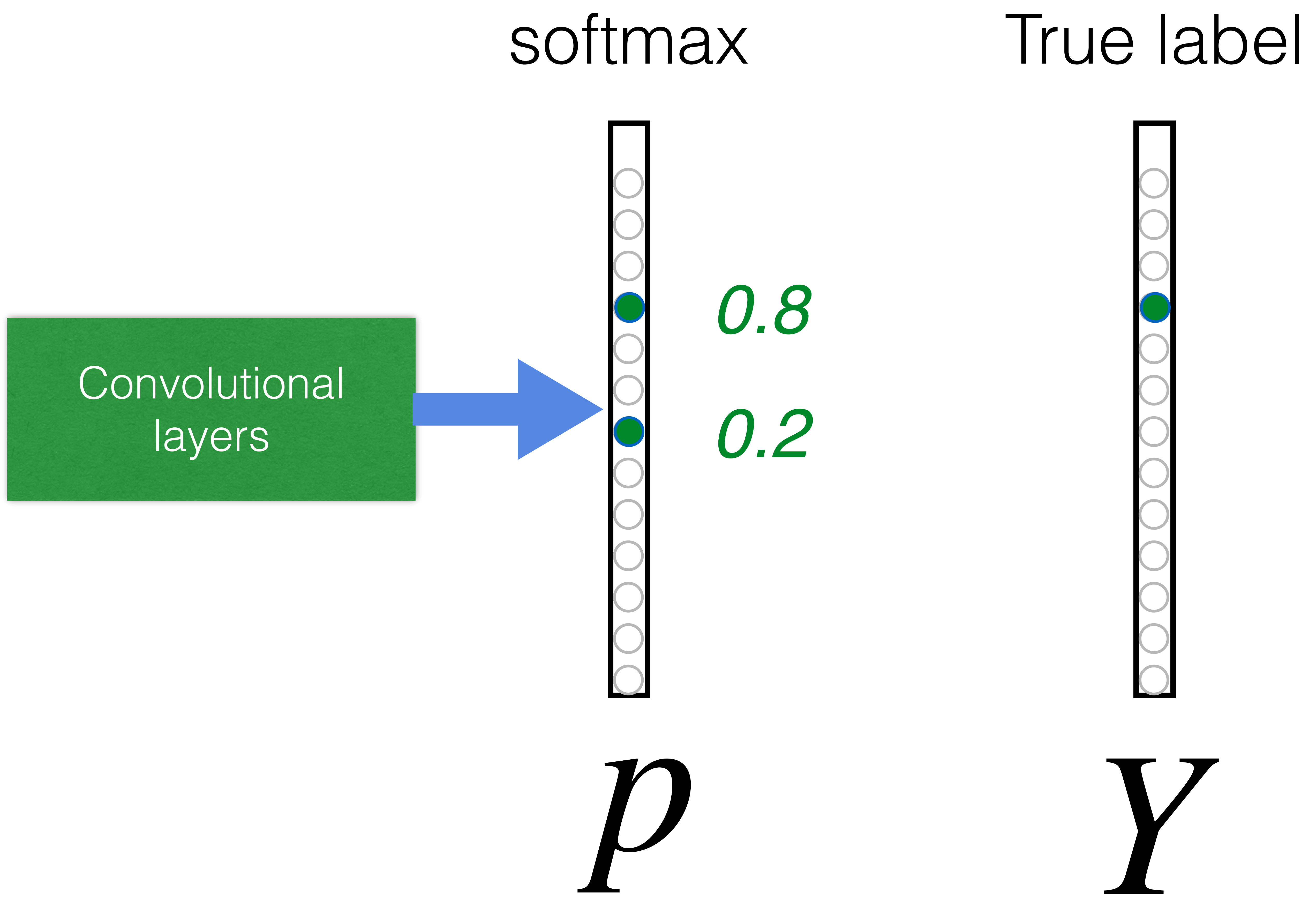


$$L_{CE} = \sum_i -Y_i \log(p_i)$$

$$= -\log(0.8)$$

Goal: push **p** and **Y** to be identical

Cross-Entropy Loss

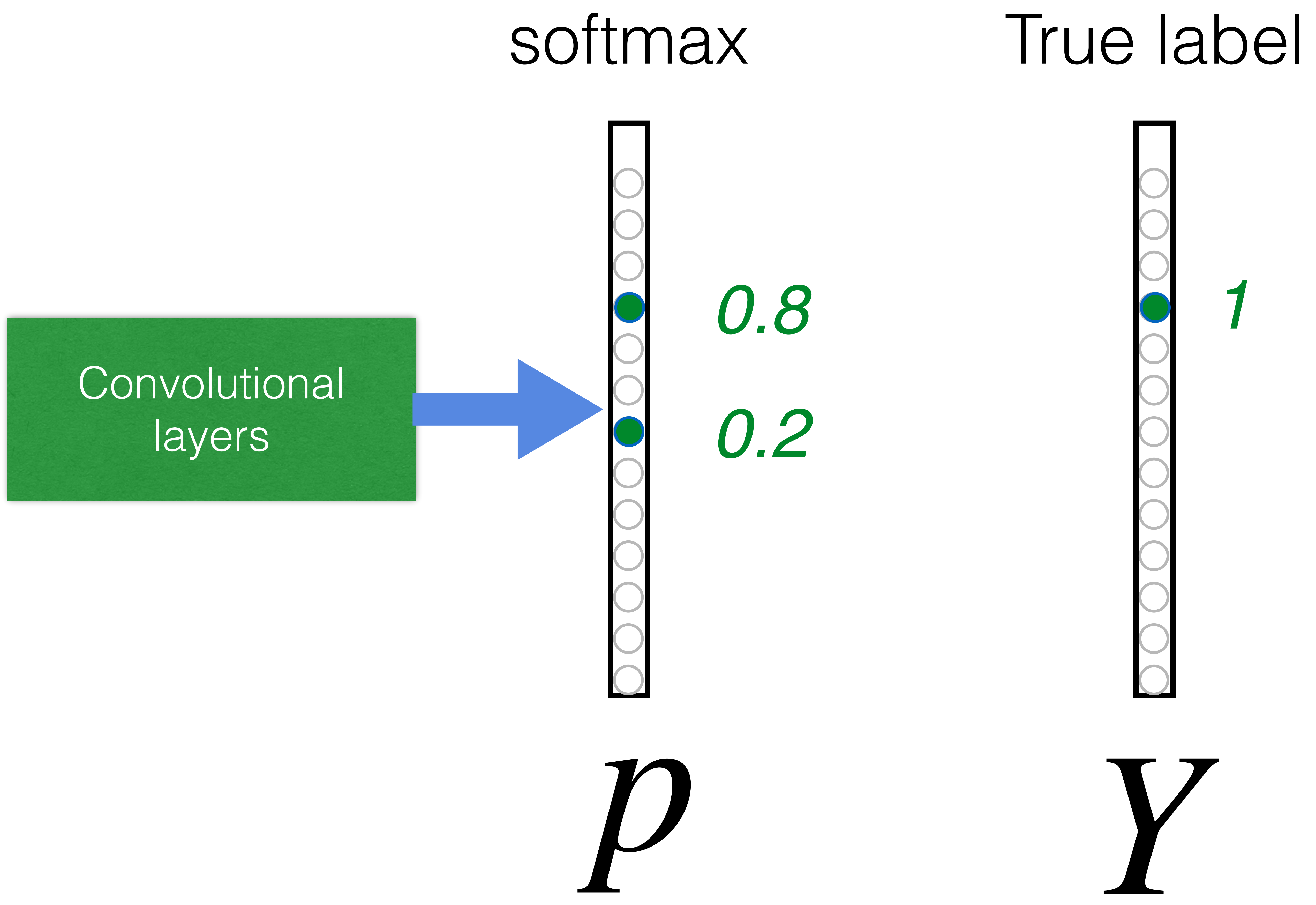


$$L_{CE} = \sum_i - Y_i \log(p_i)$$

$$= -\log(0.8)$$

Goal: push **p** and **Y** to be identical

Cross-Entropy Loss



$$L_{CE} = \sum_i - Y_i \log(p_i)$$

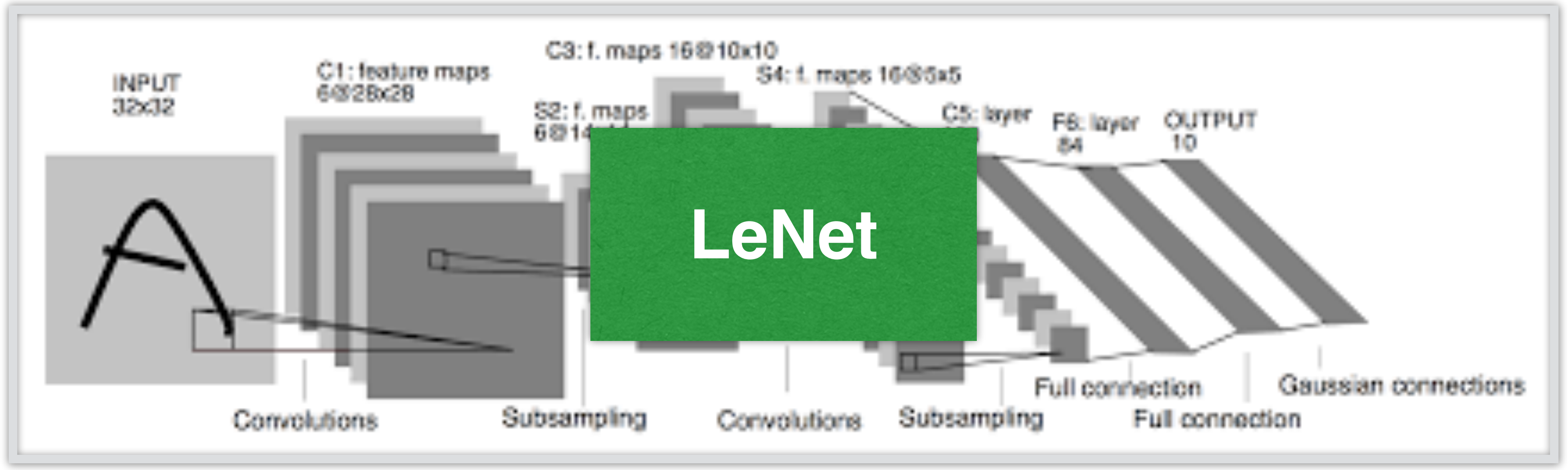
$$= -\log(0.8)$$

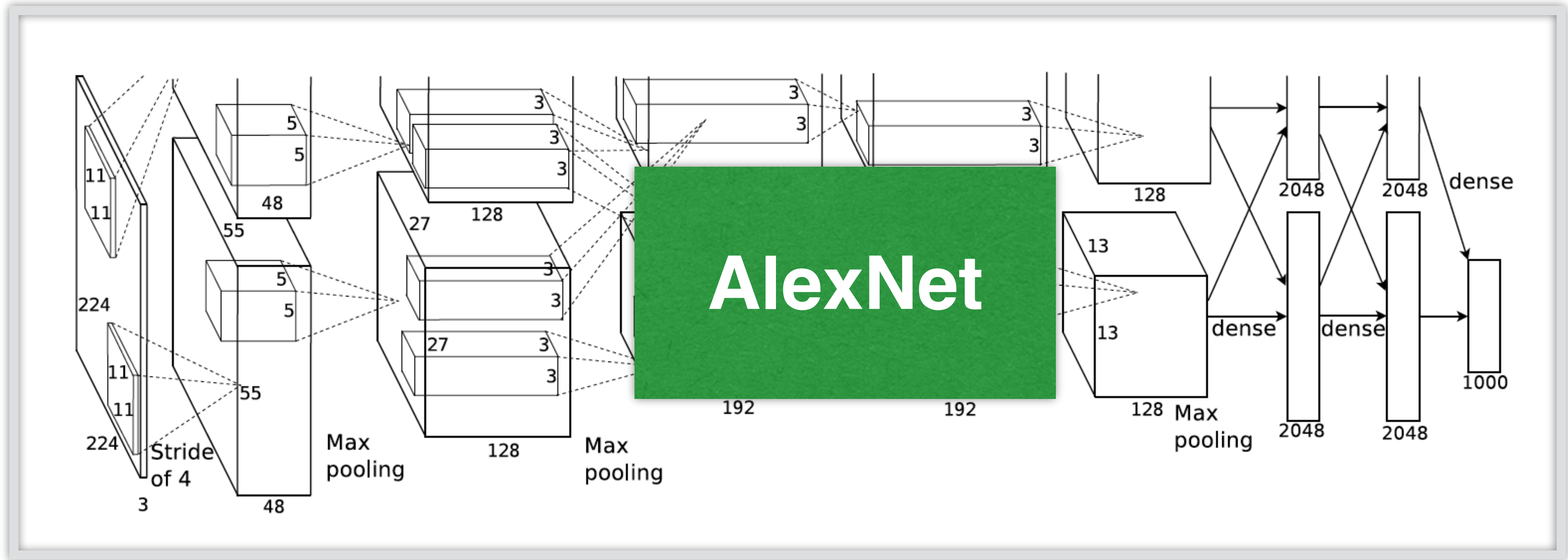
Goal: push **p** and **Y** to be identical

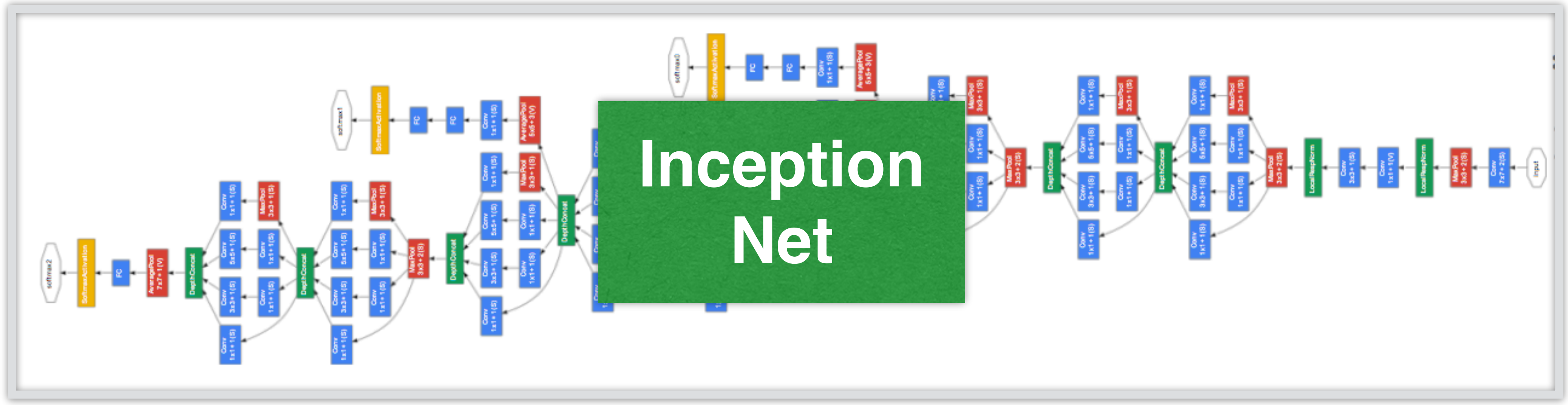
Convolutional Neural Networks

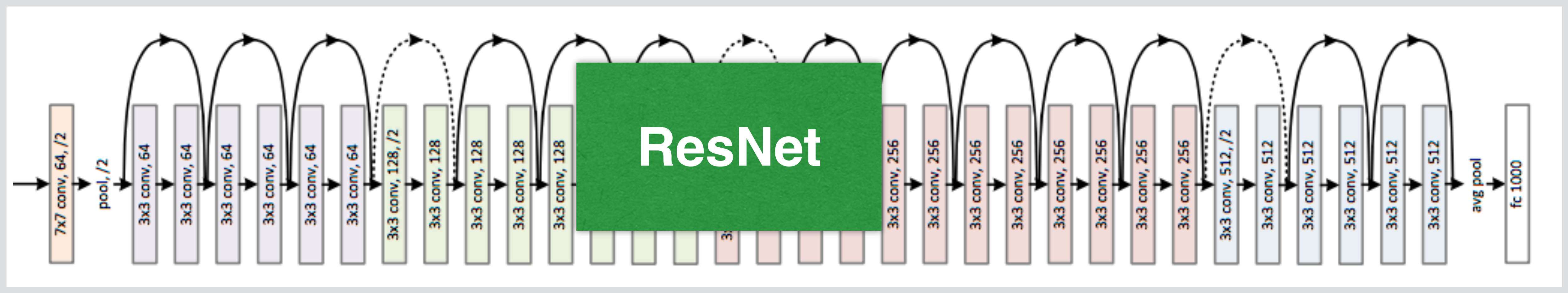
Evolution of neural net architectures

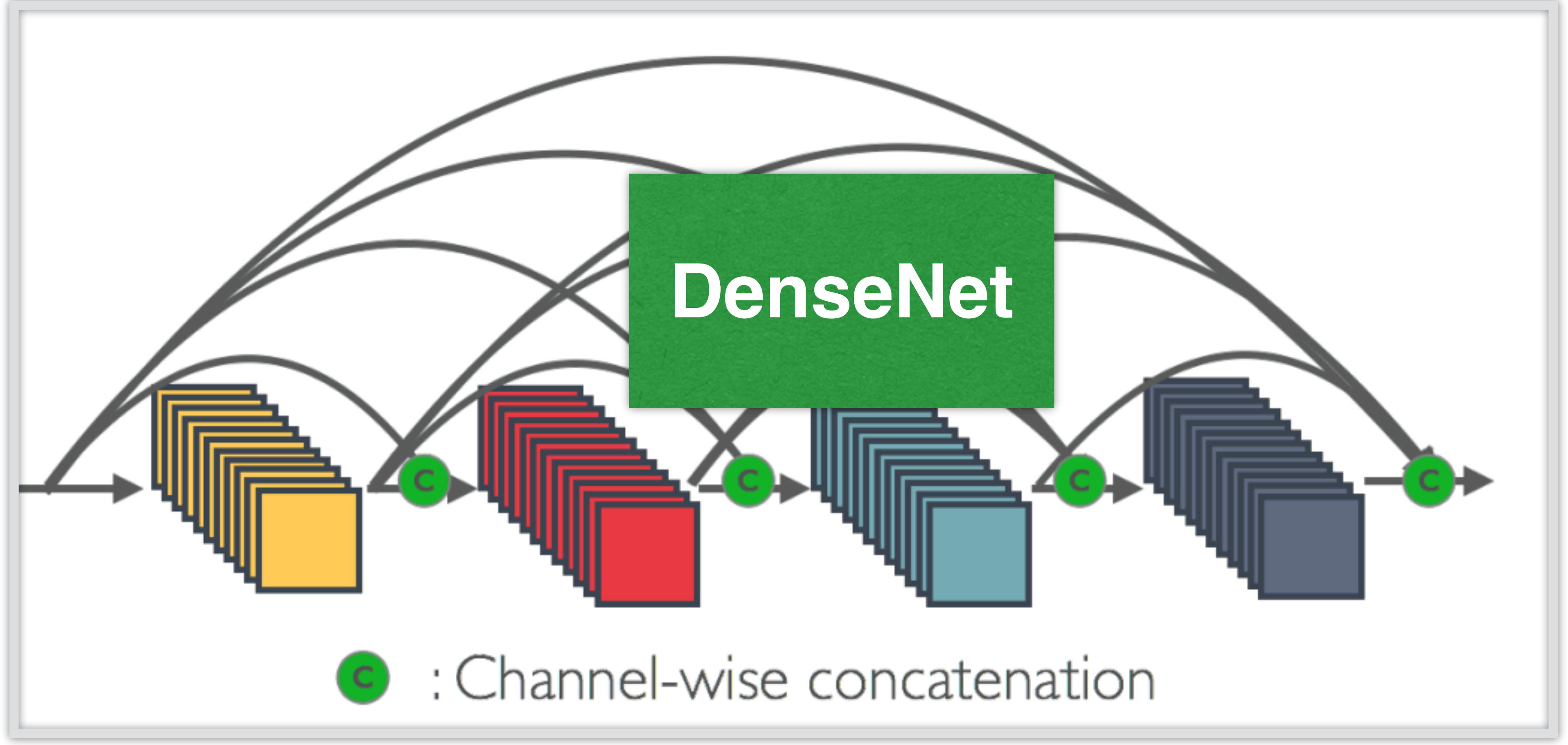
Evolution of neural net architectures







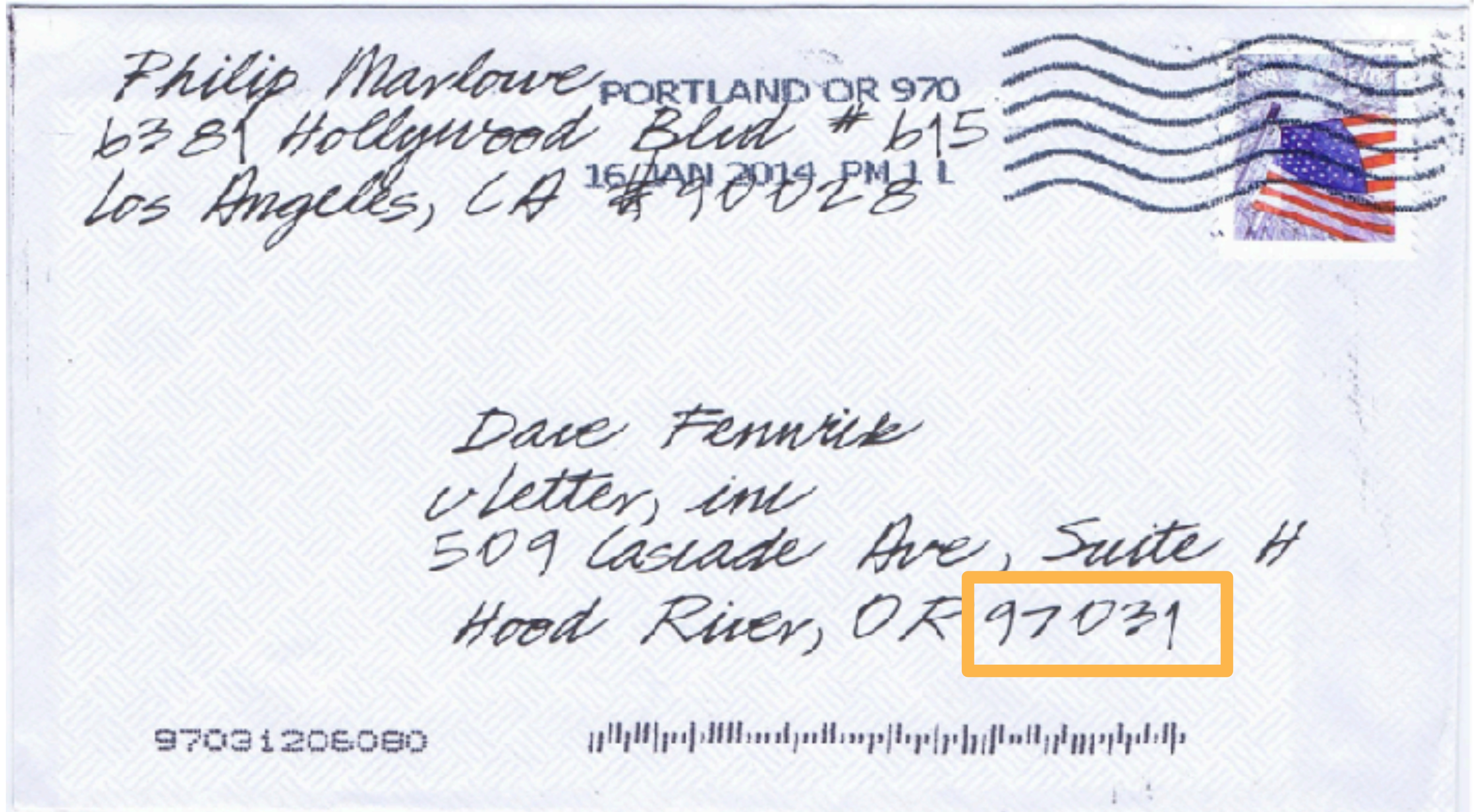


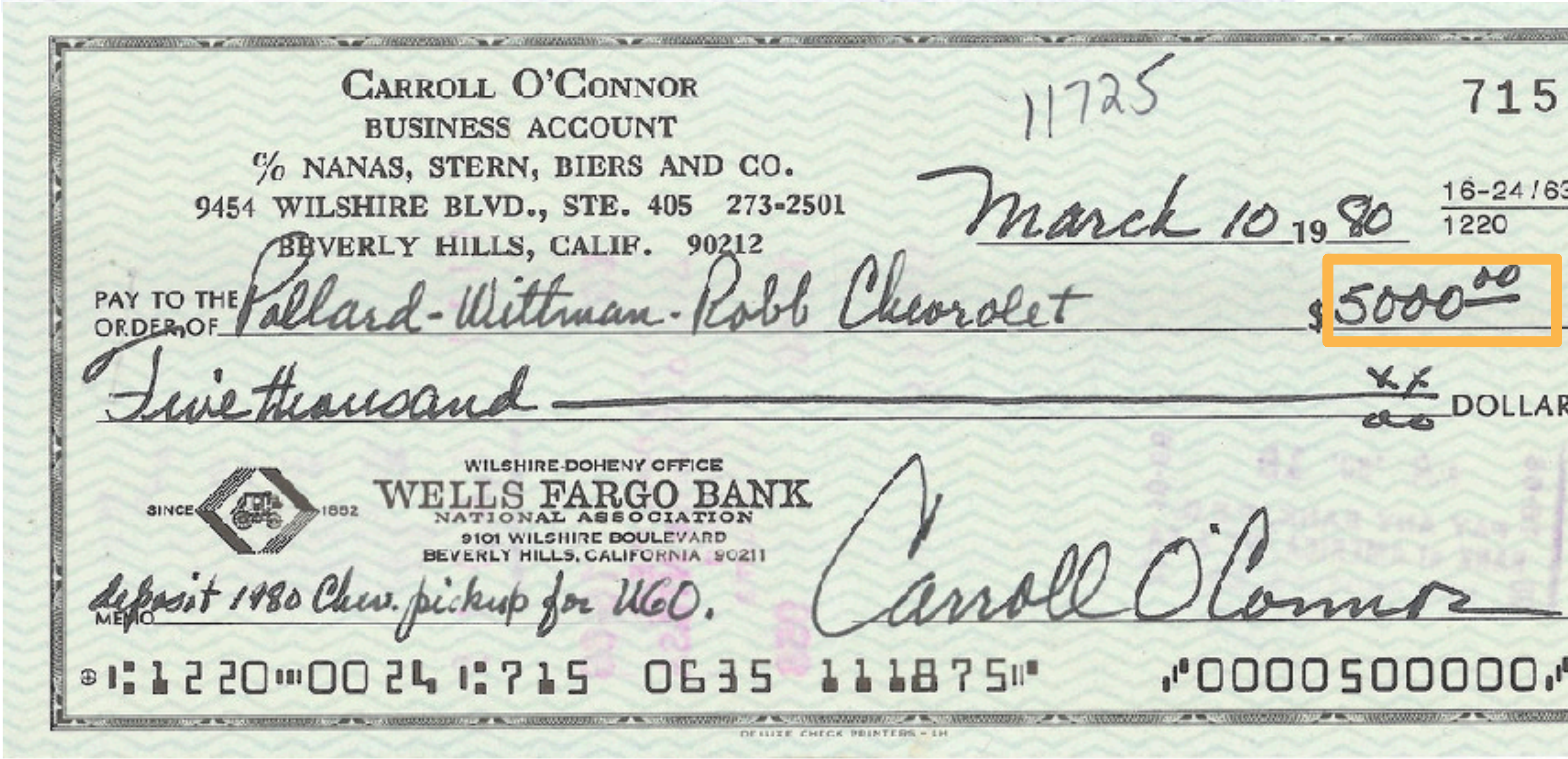


LeNet Architecture
(first conv nets)

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

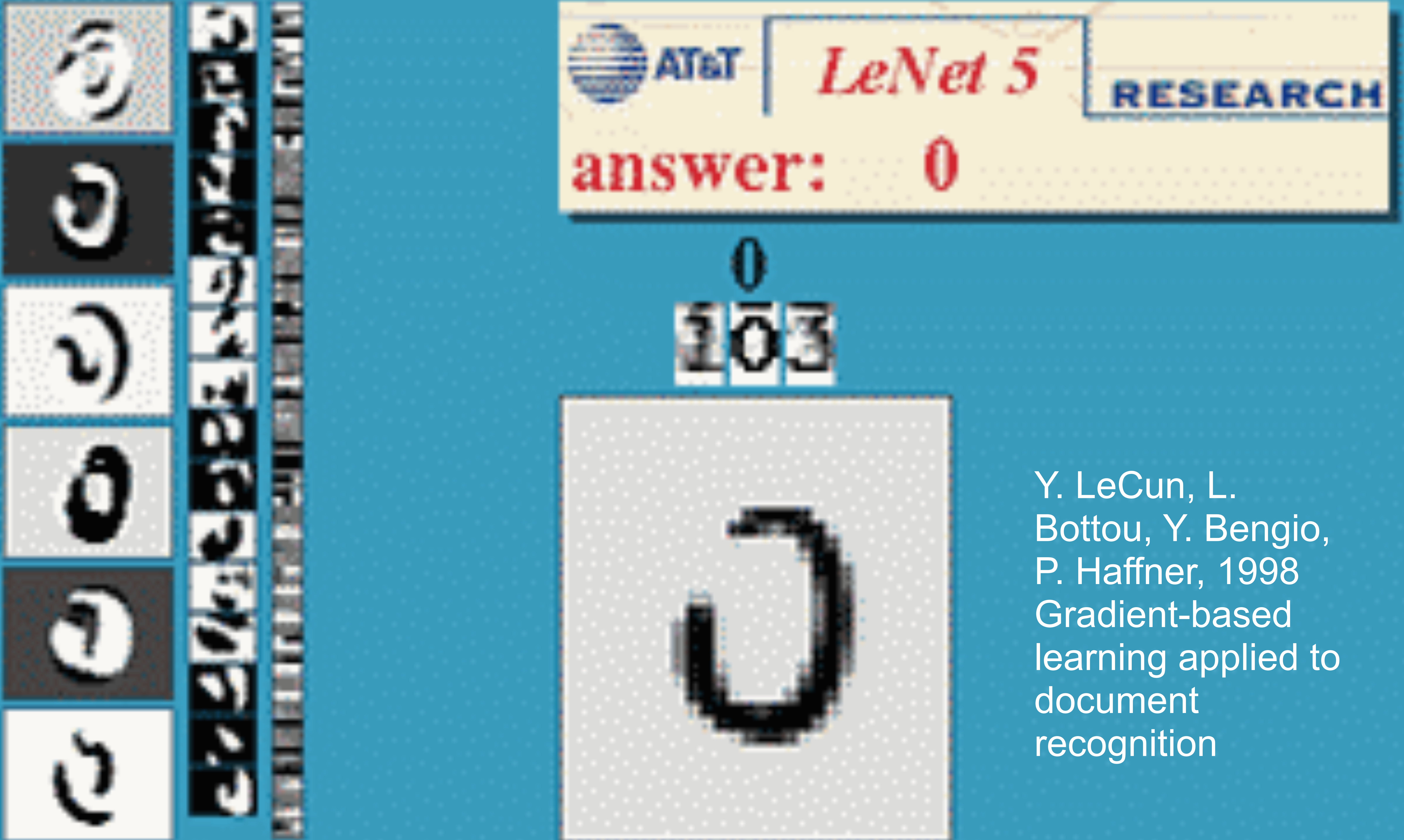
Handwritten Digit Recognition



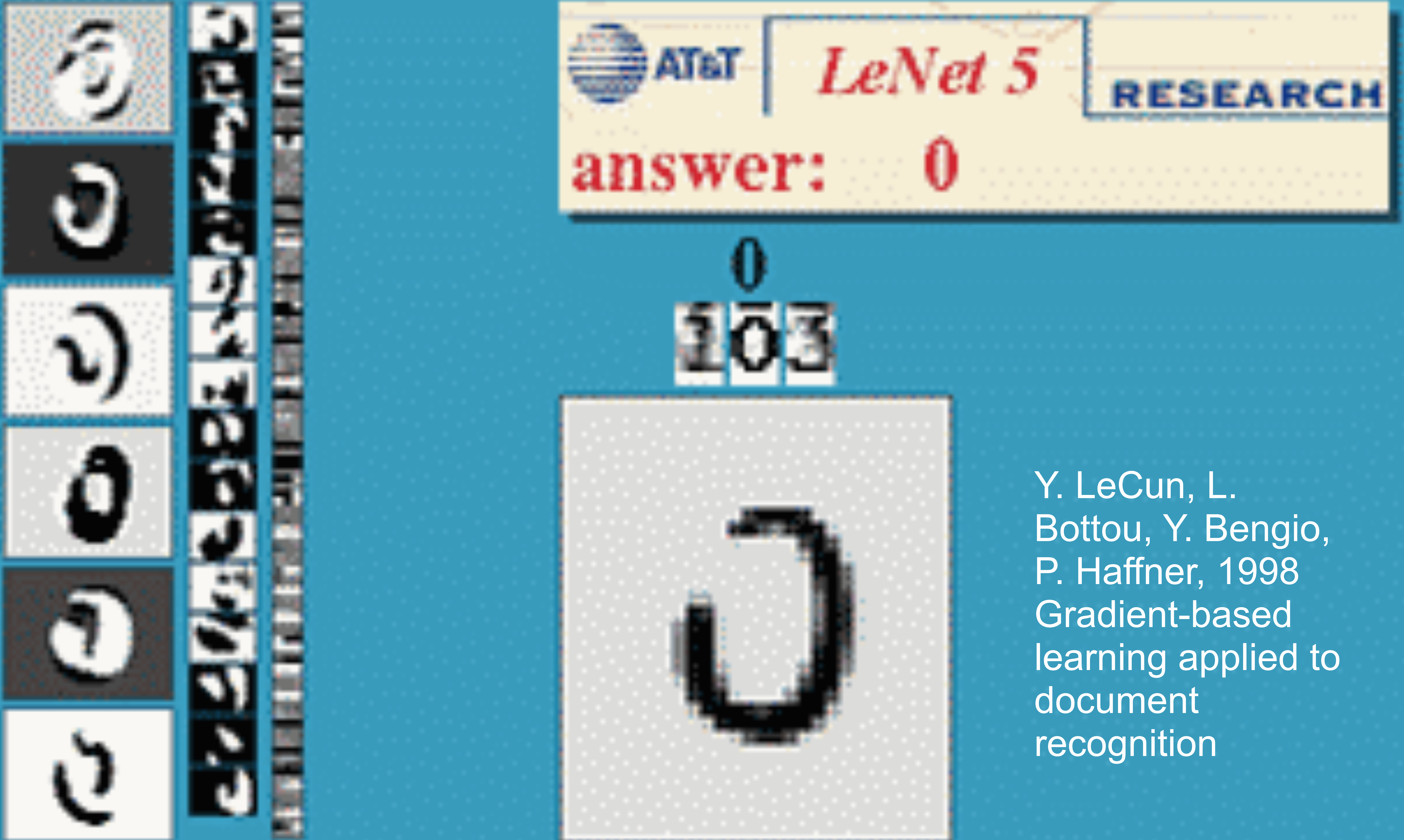


MNIST

- 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



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 (col
    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

```python
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
```

LeNet in Pytorch

Let’s walk through an example using PyTorch

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

AlexNet

224
11
11
224
11
11
3
Stride of 4
55
55
5
5
5
5
48
48
Max pooling
27
27
3
3
128
128
Max pooling
13
13
13
3
3
192
192
13
13
192
192
13
13
128
128
Max pooling
dense
dense
dense
2048
2048
2048
2048
1000

Deng et al. 2009

AlexNet

AlexNet

- AlexNet won ImageNet competition in 2012

AlexNet

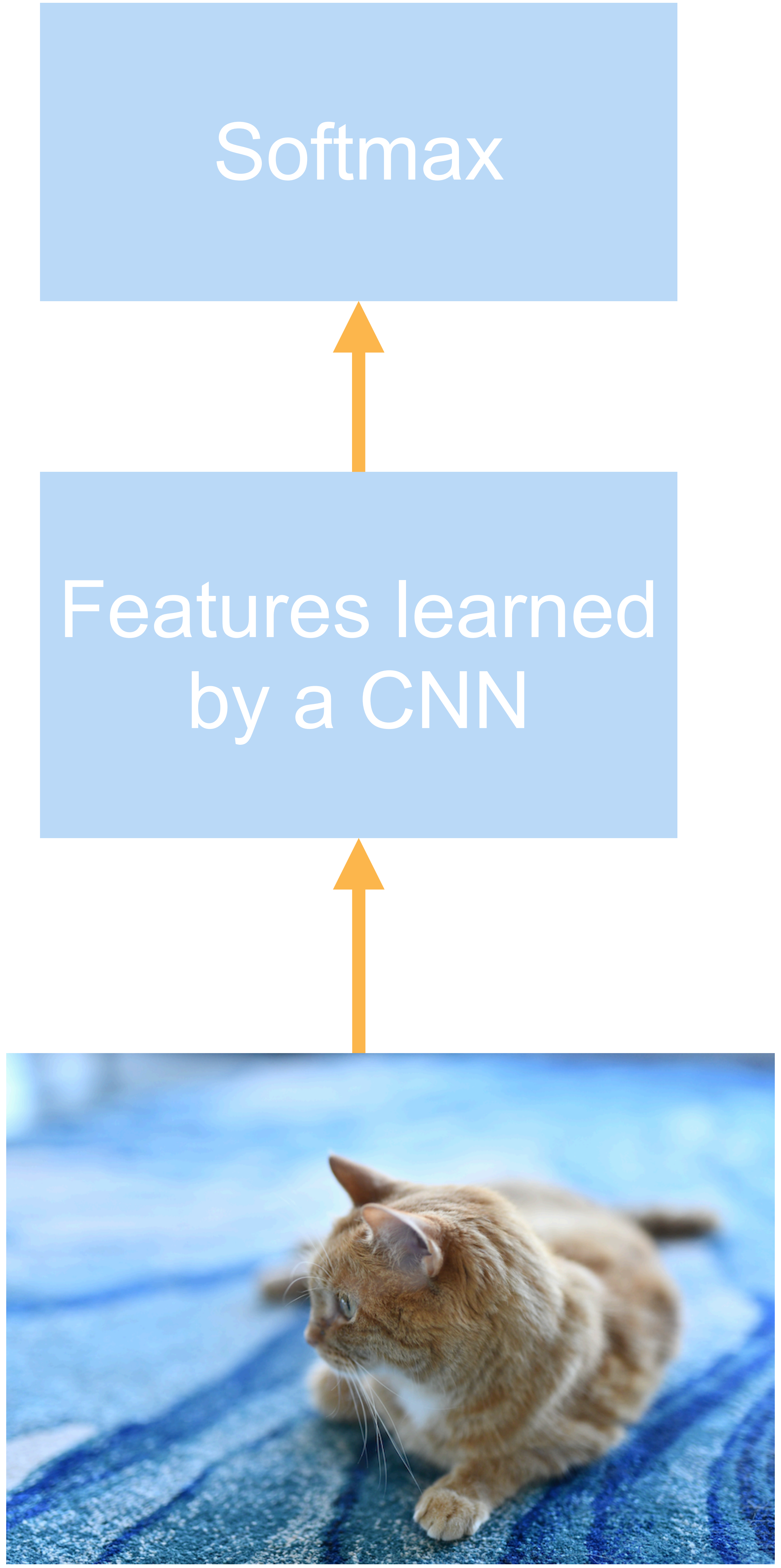
- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet

AlexNet

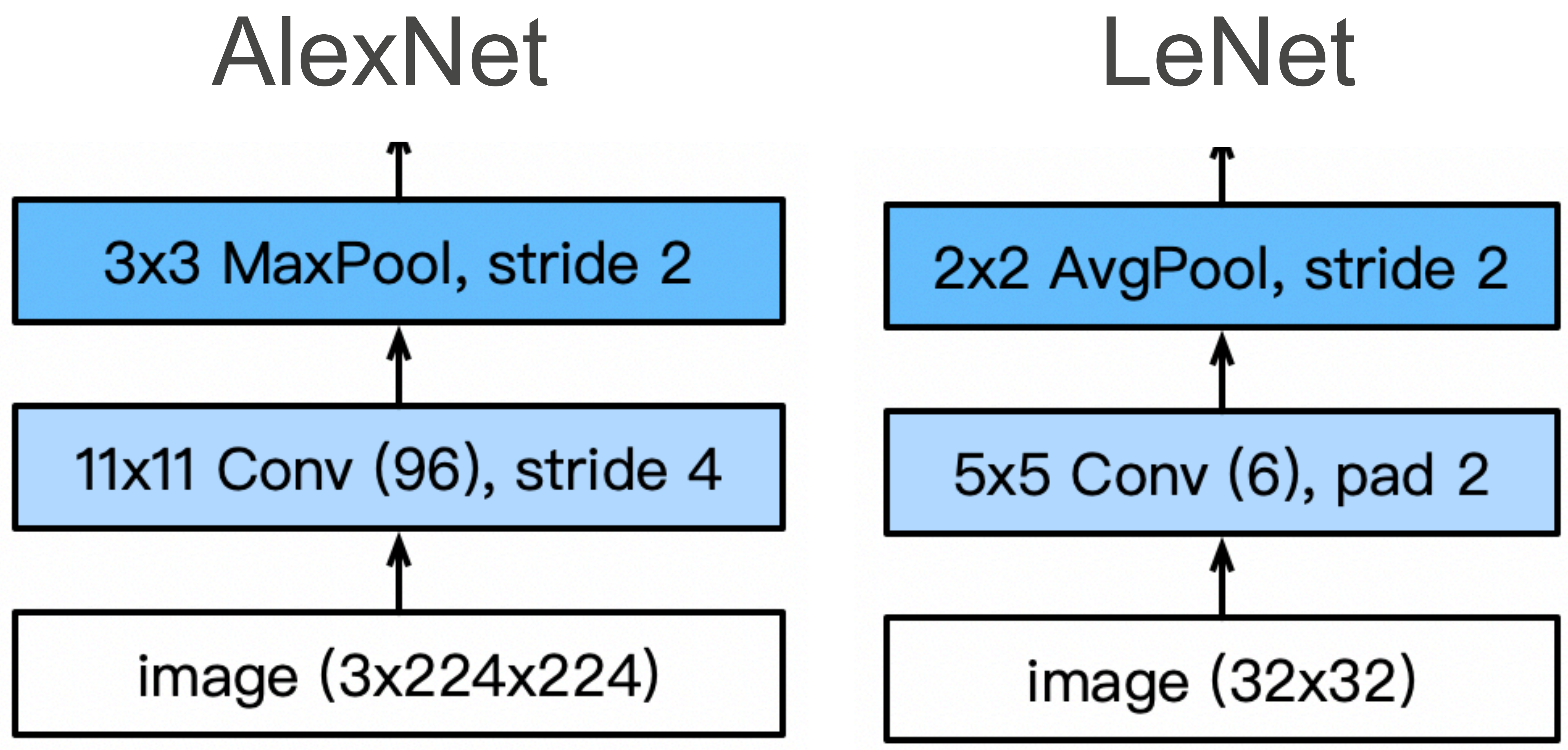
- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Paradigm shift for computer vision

AlexNet

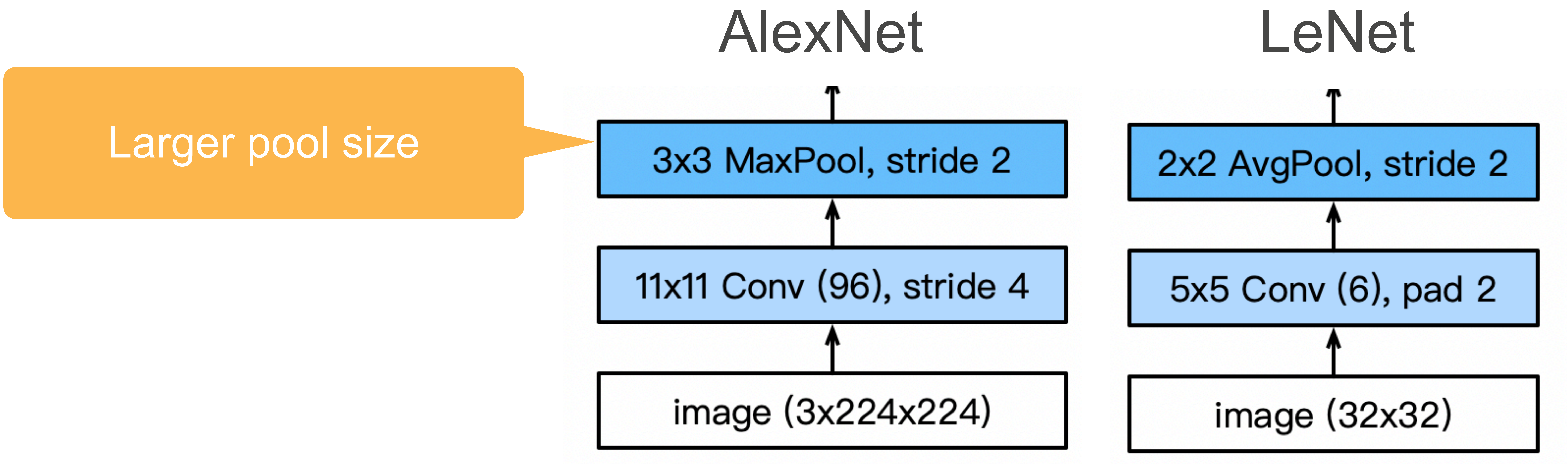
- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Paradigm shift for computer vision



AlexNet Architecture



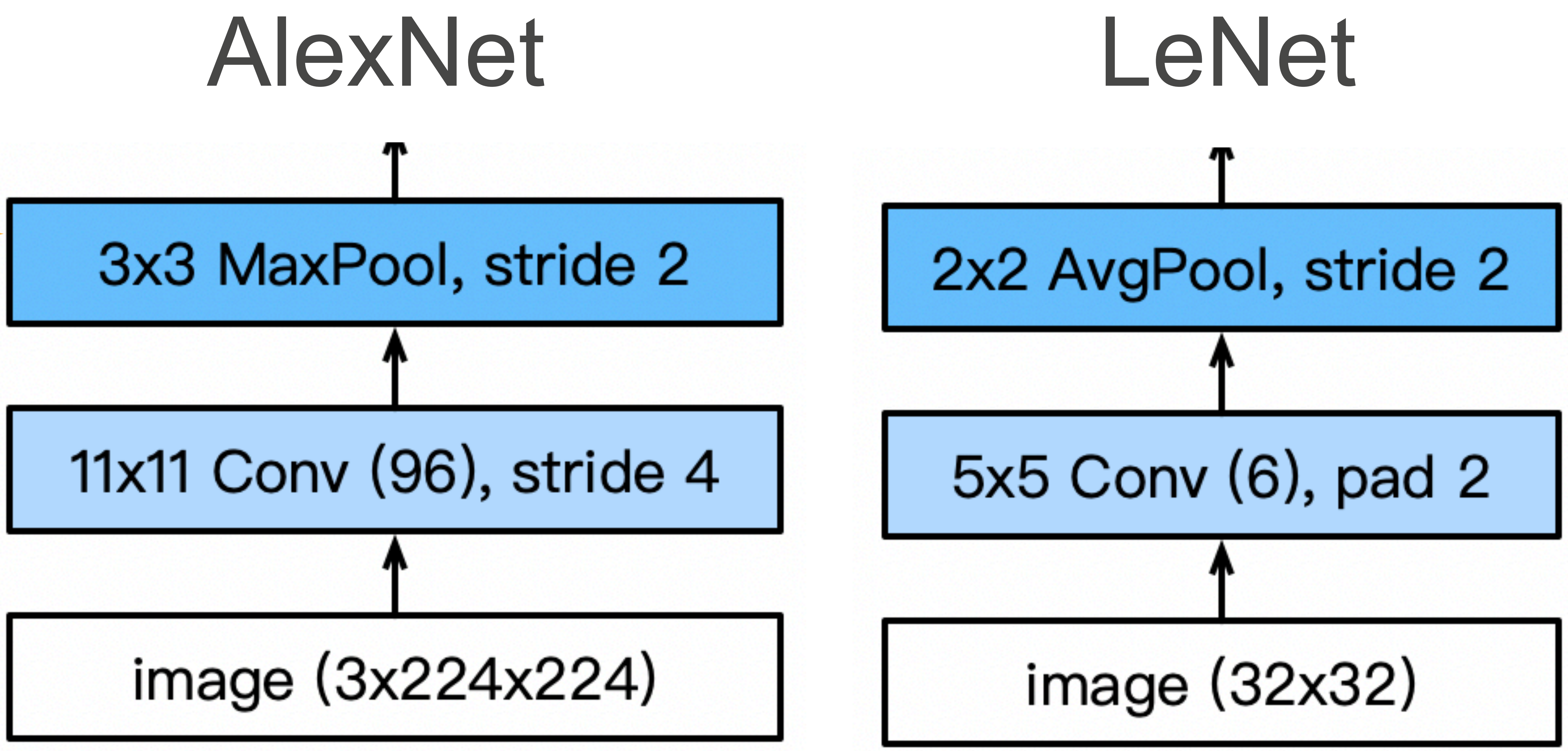
AlexNet Architecture



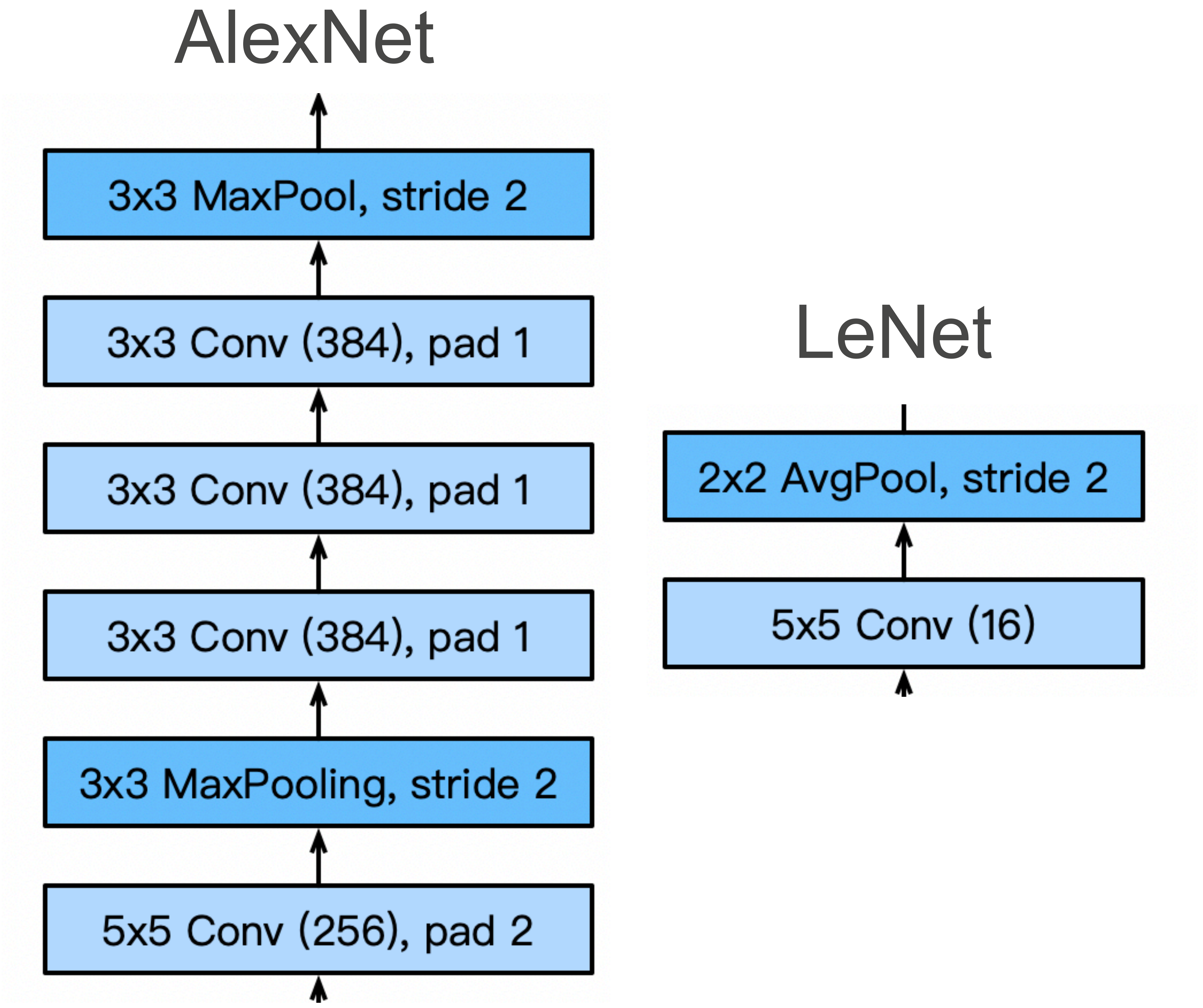
AlexNet Architecture

Larger pool size

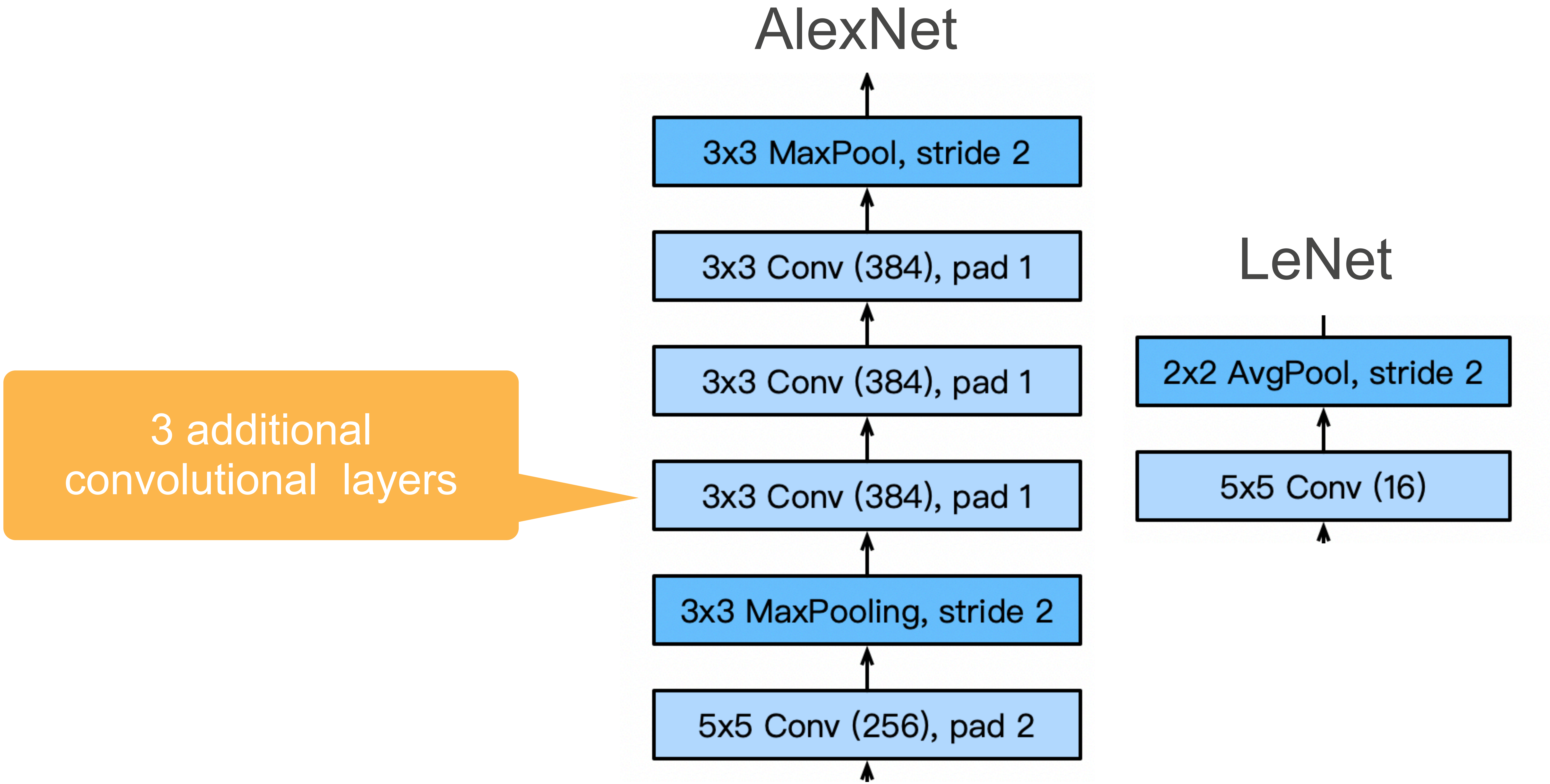
Larger kernel size, stride because of the increased image size, and more output channels.



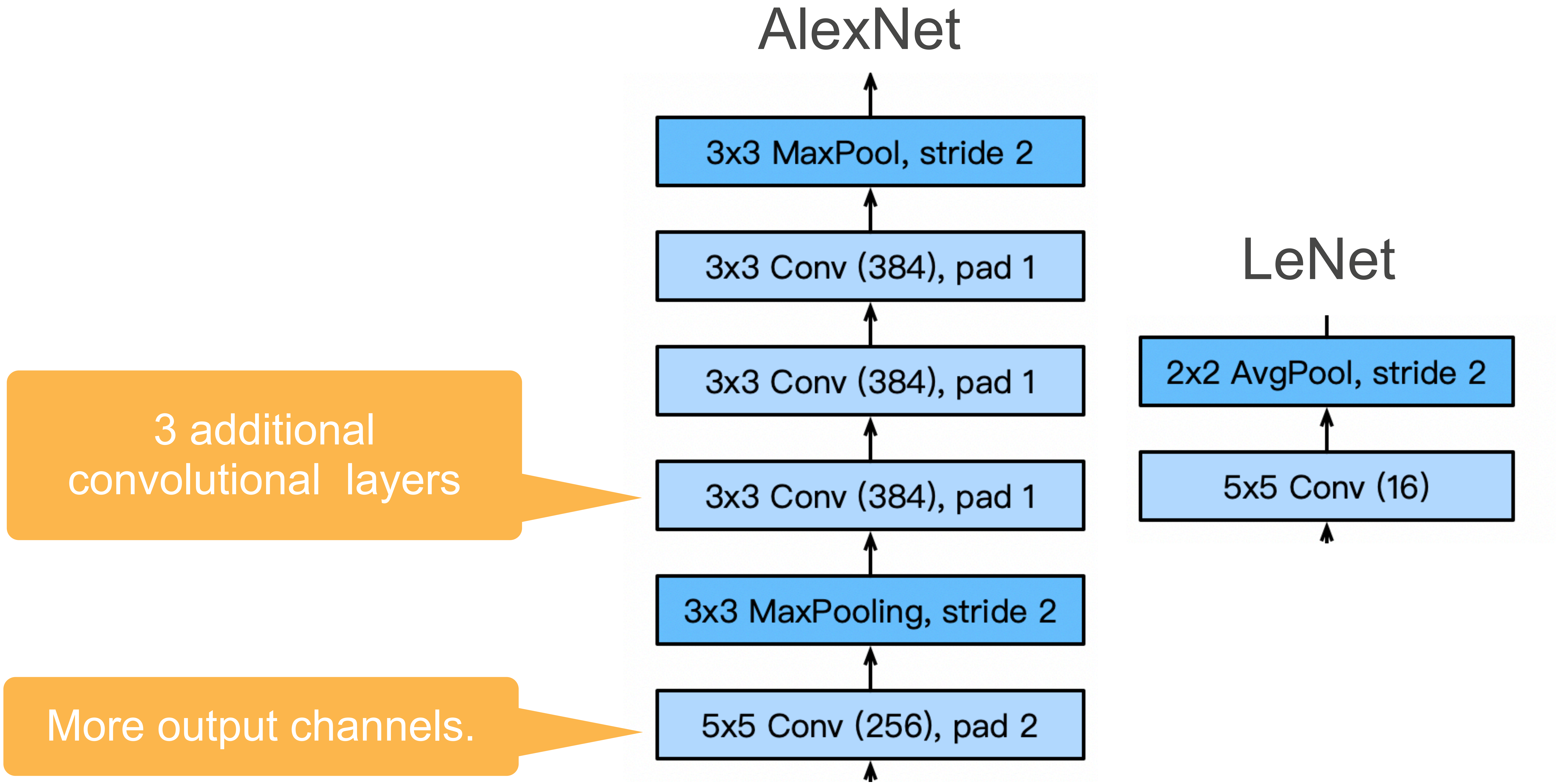
AlexNet Architecture



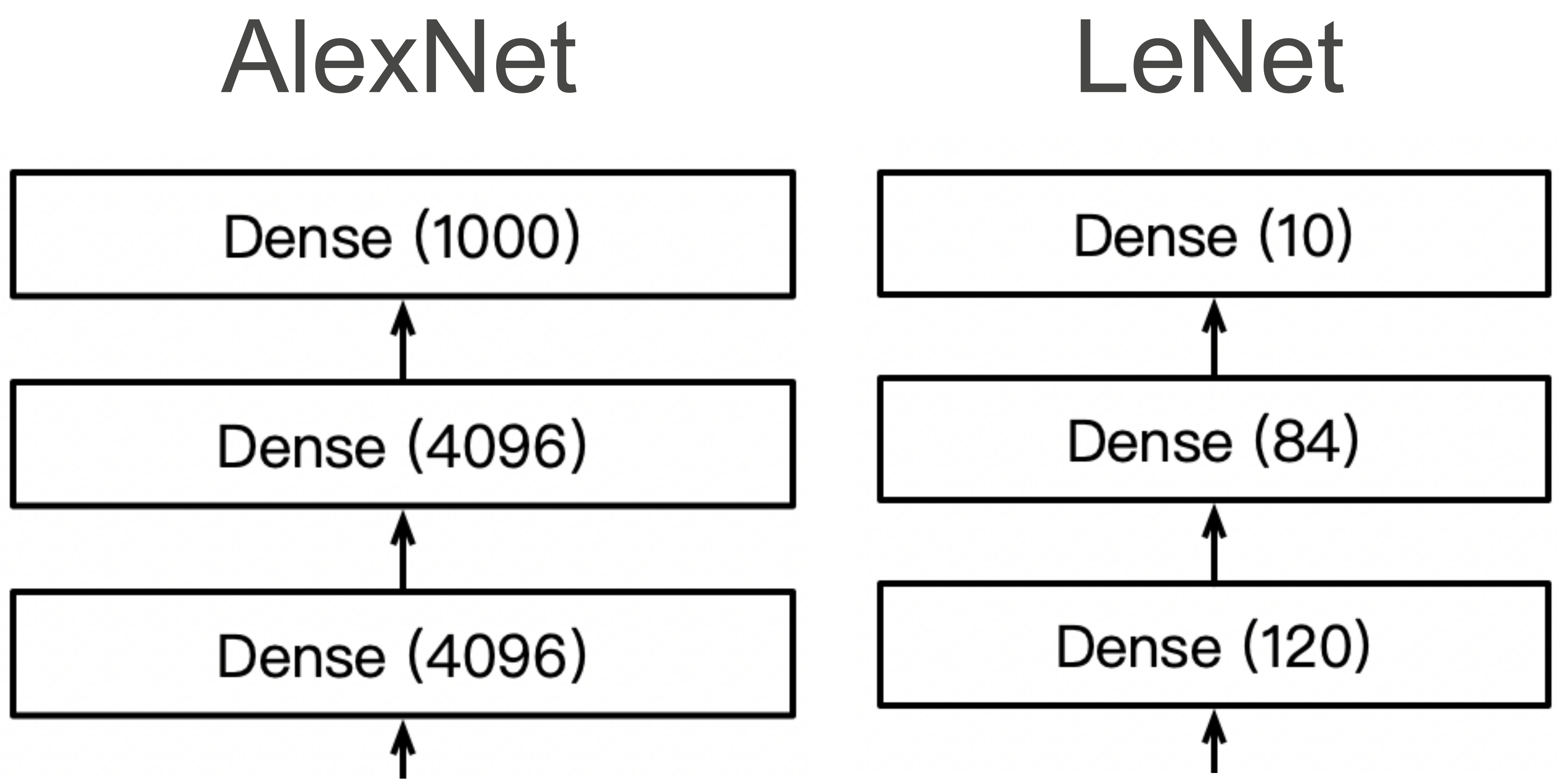
AlexNet Architecture



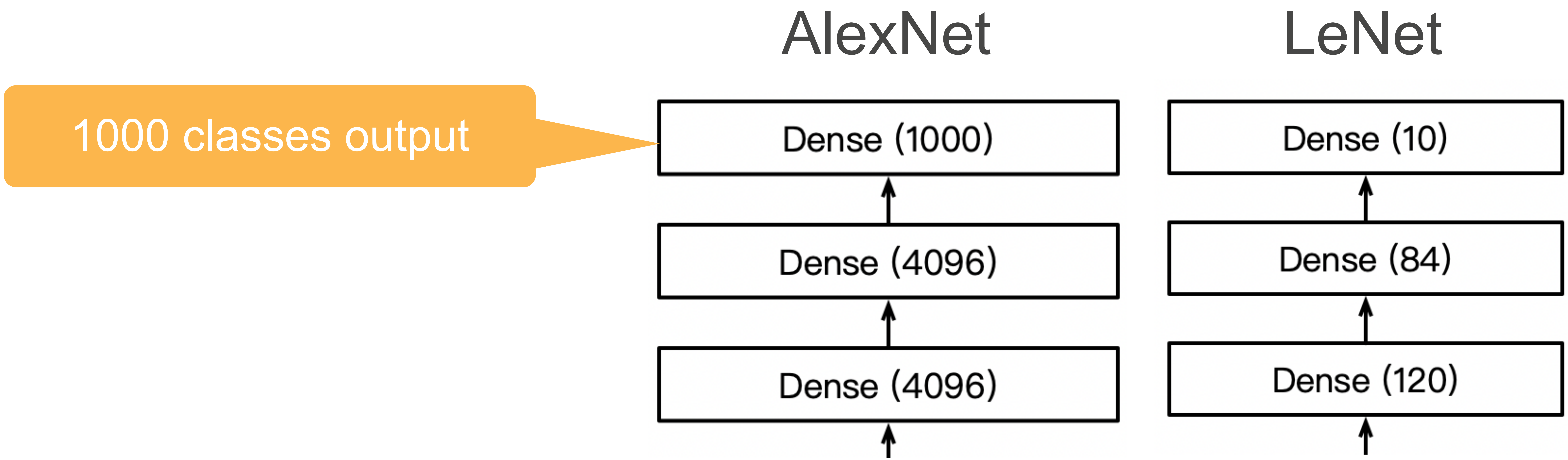
AlexNet Architecture



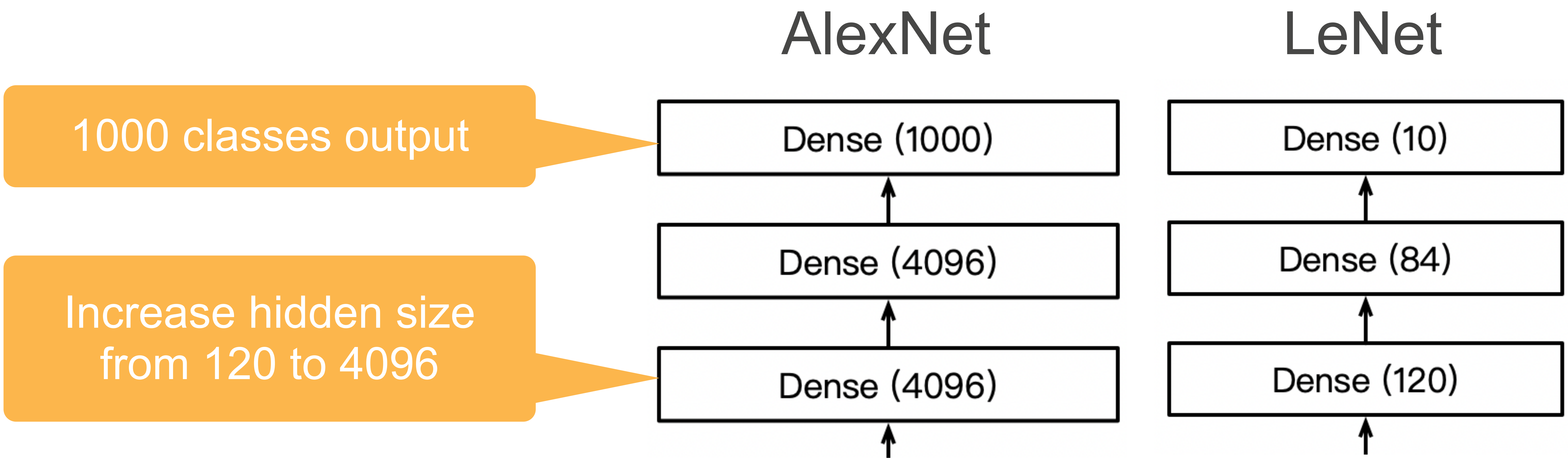
AlexNet Architecture



AlexNet Architecture

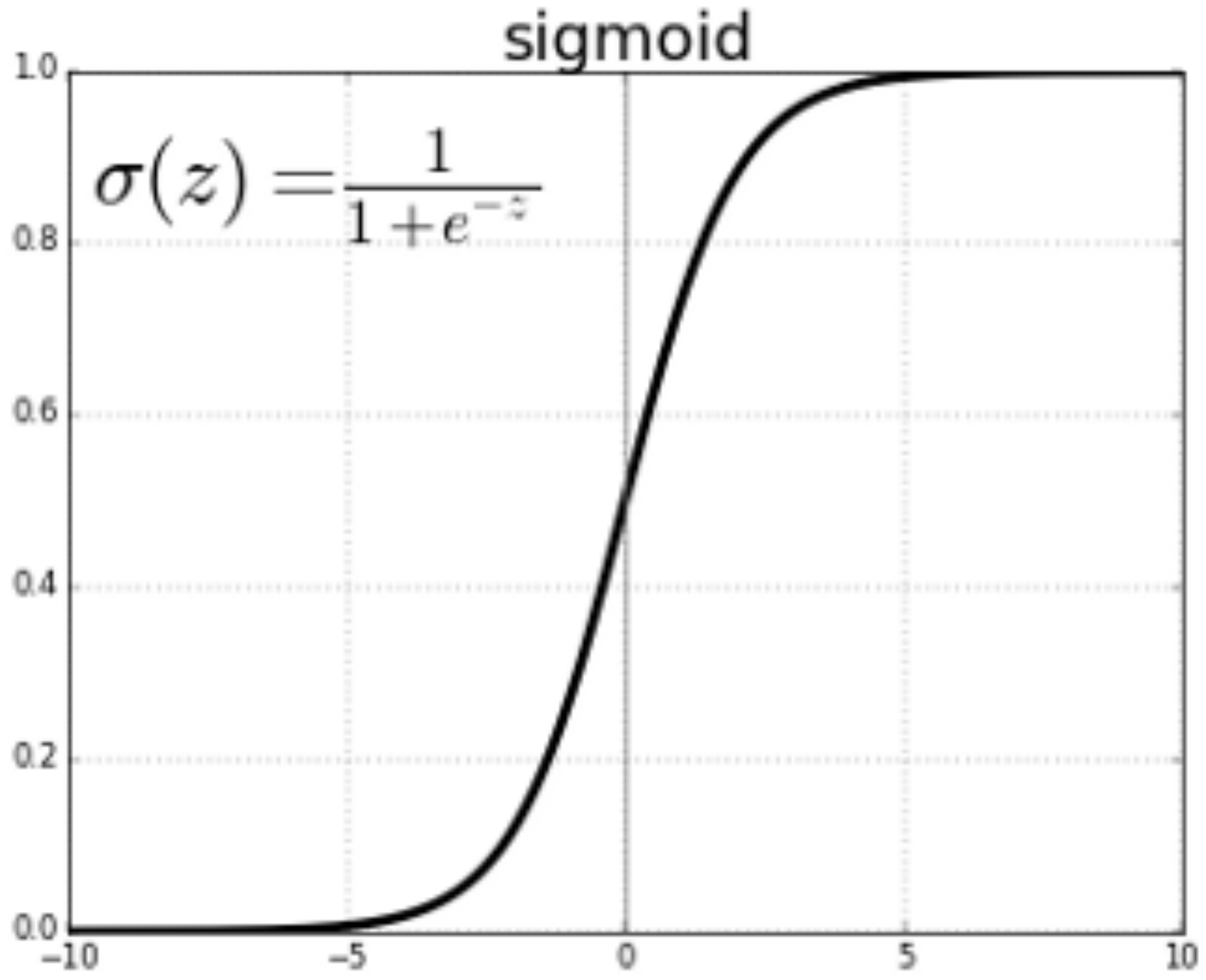


AlexNet Architecture



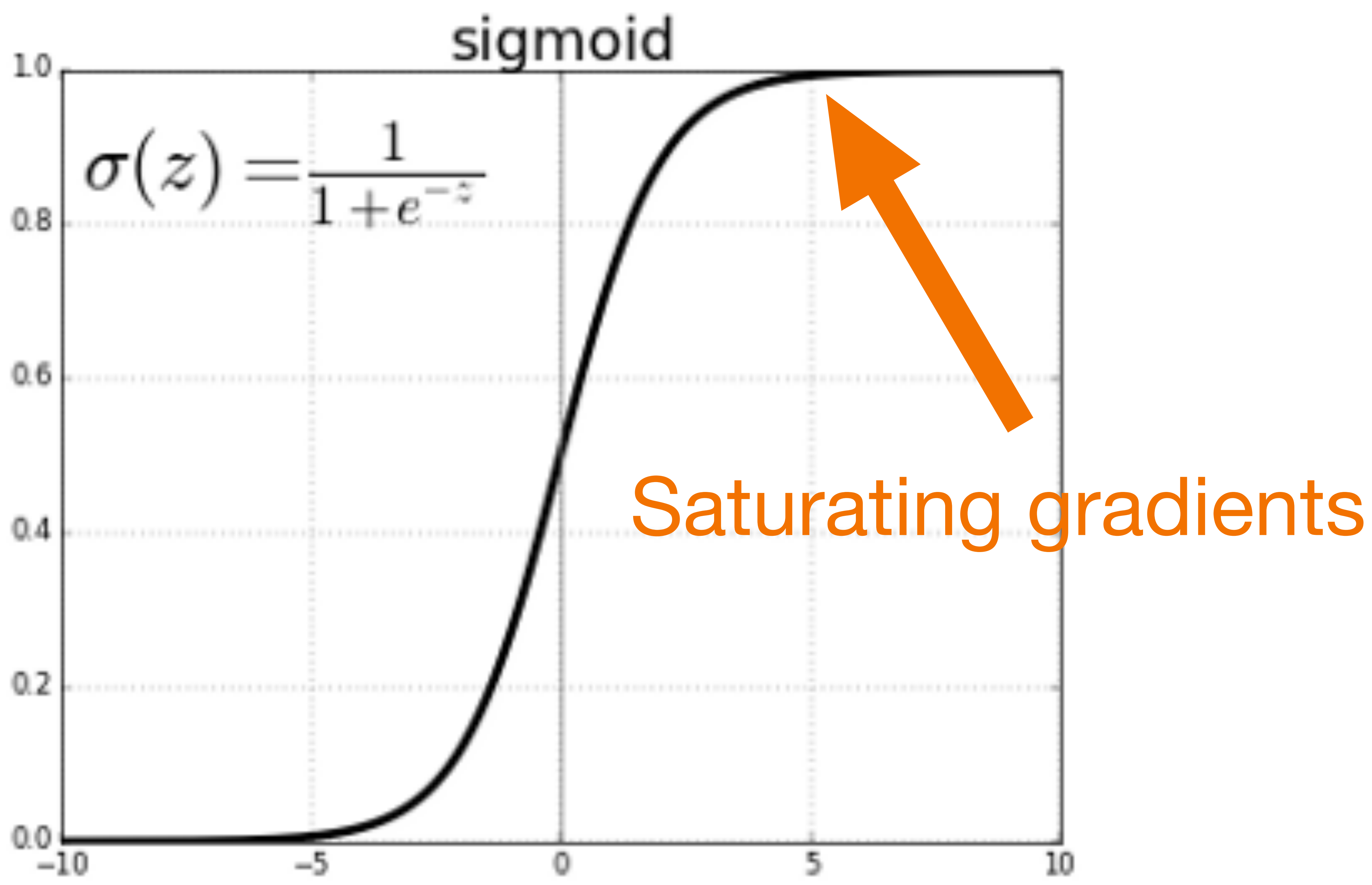
More Differences…

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



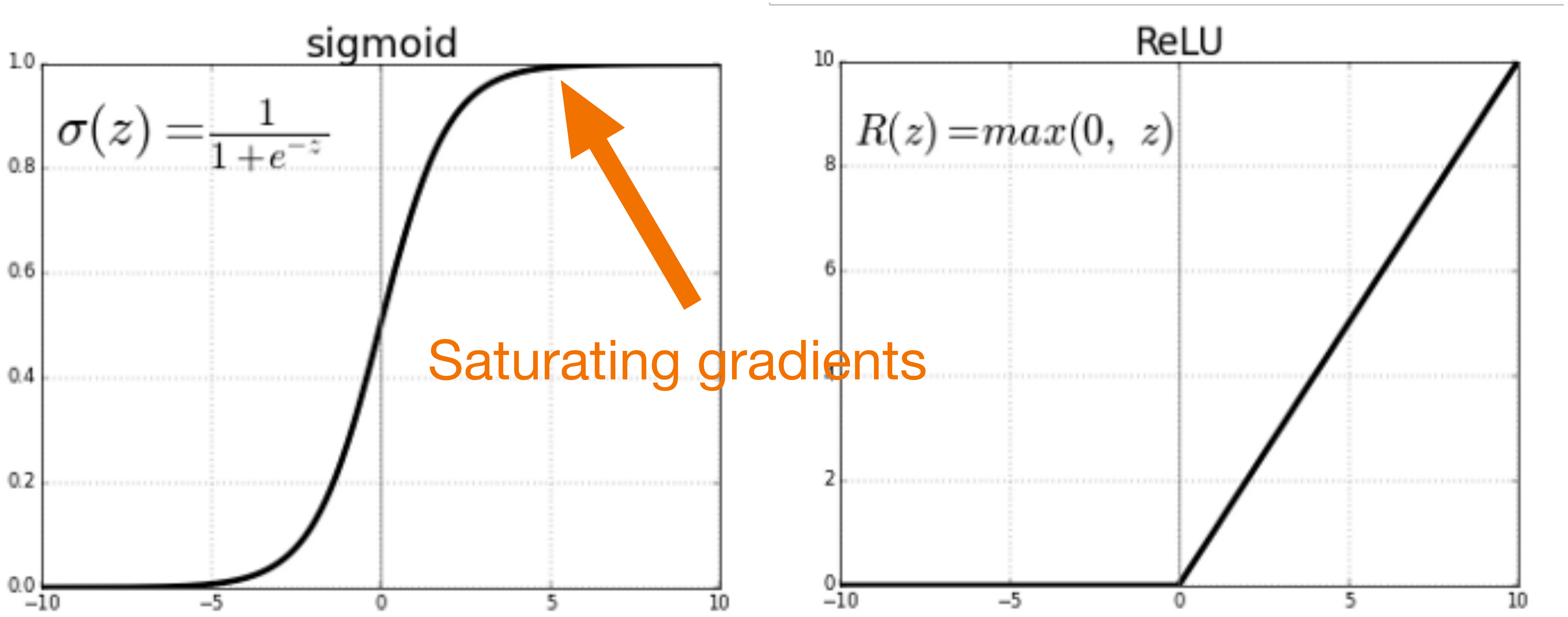
More Differences…

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



More Differences…

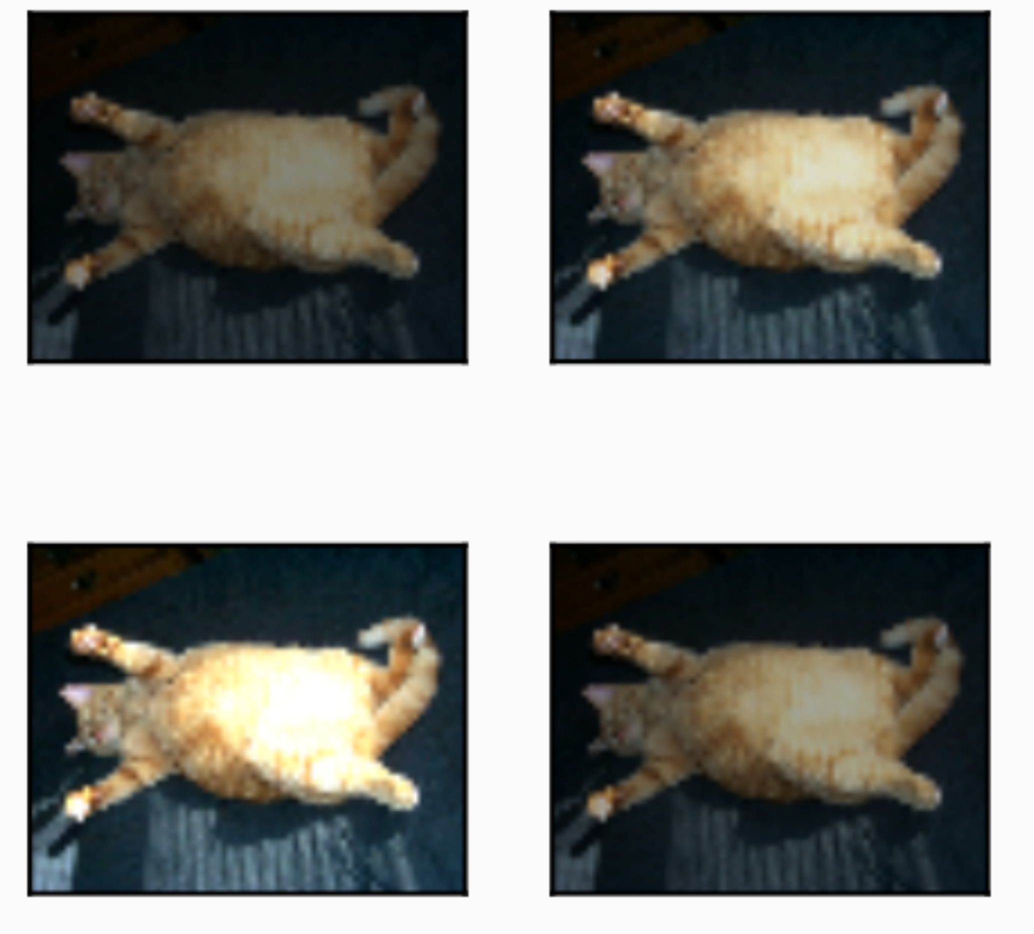
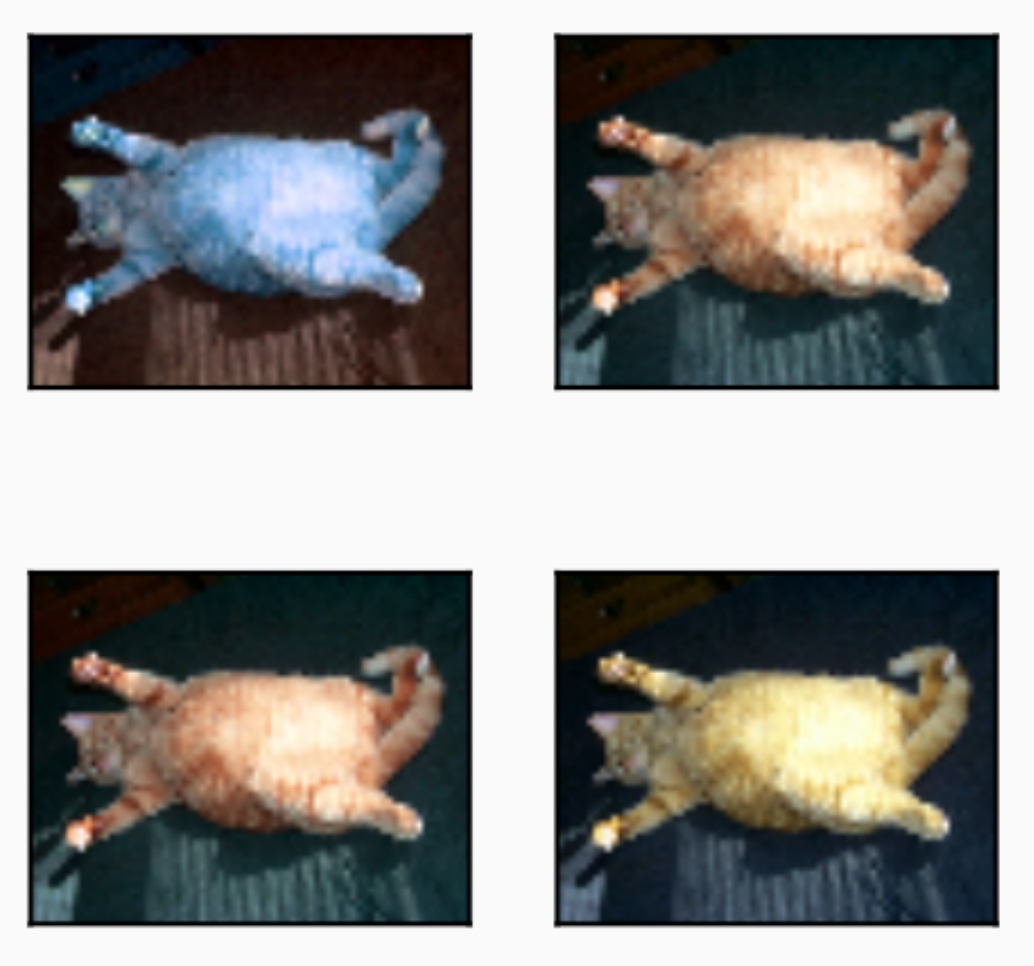
- Change activation function from sigmoid to ReLu (no more vanishing gradient)



More Differences…

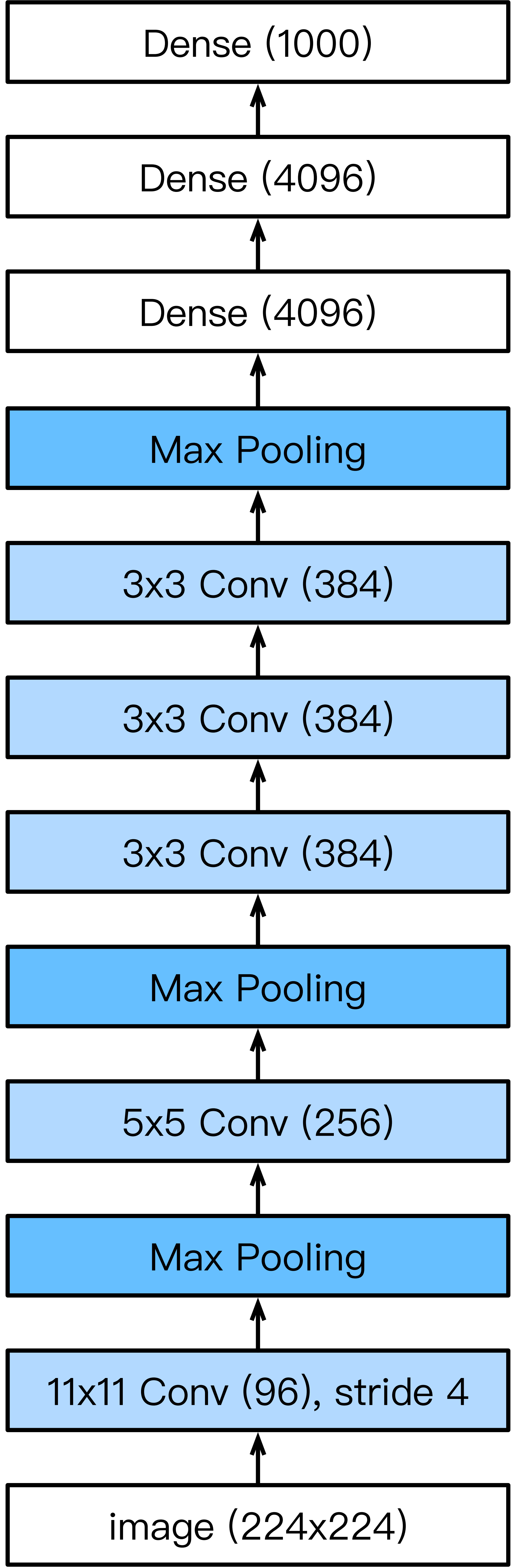
- Change activation function from sigmoid to ReLu (no more vanishing gradient)
- Data augmentation



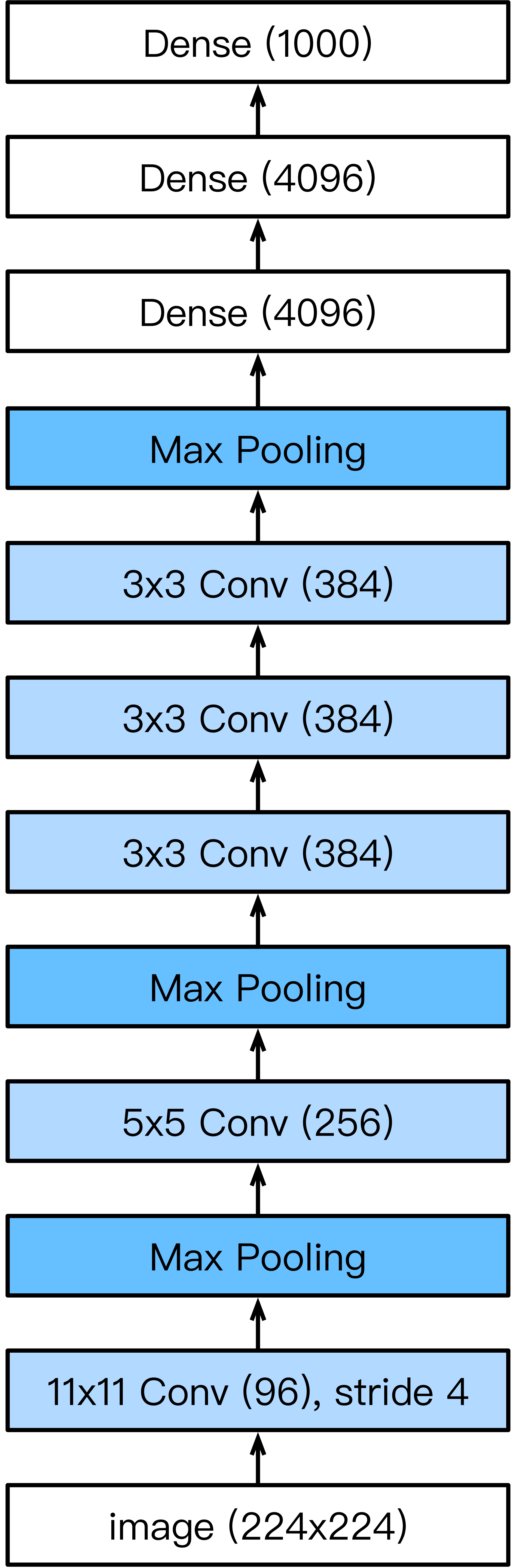
Complexity

	#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
Increase	11x	1x



Complexity

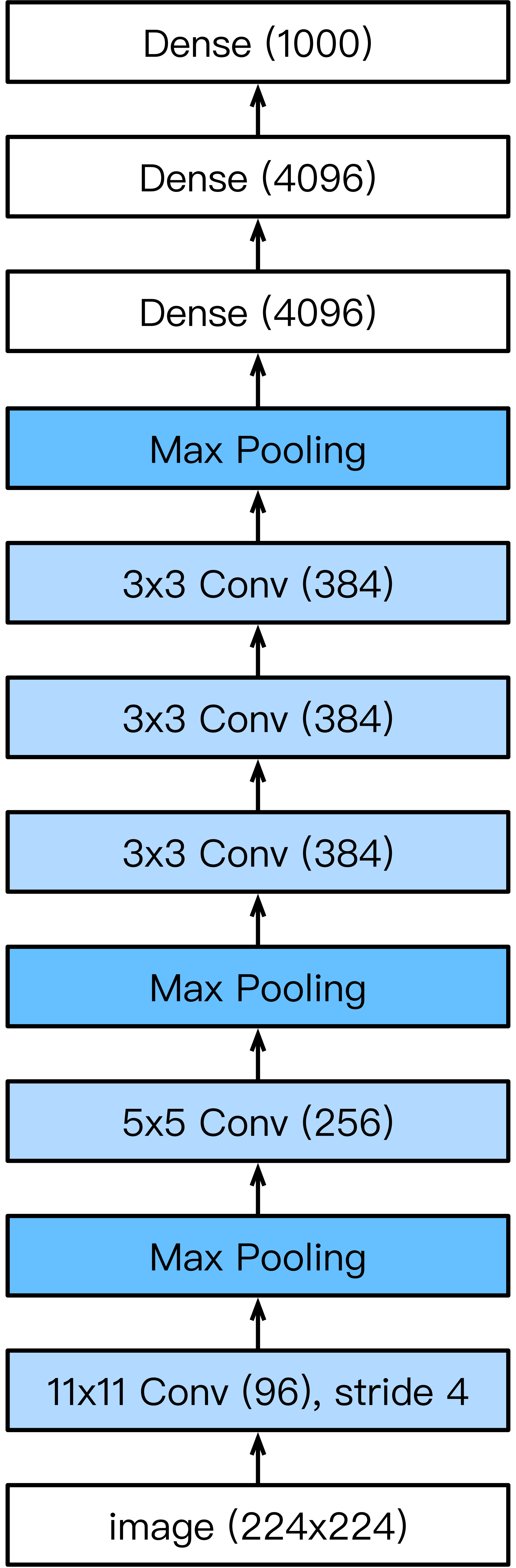
	#parameters	
	AlexNet	**LeNet**
Conv1	35K	150
Conv2	614K	2.4K
Conv3-5	3M	
Dense1	26M	0.048M
Dense2	16M	0.01M
Total	46M	0.06M
Increase	11x	1x

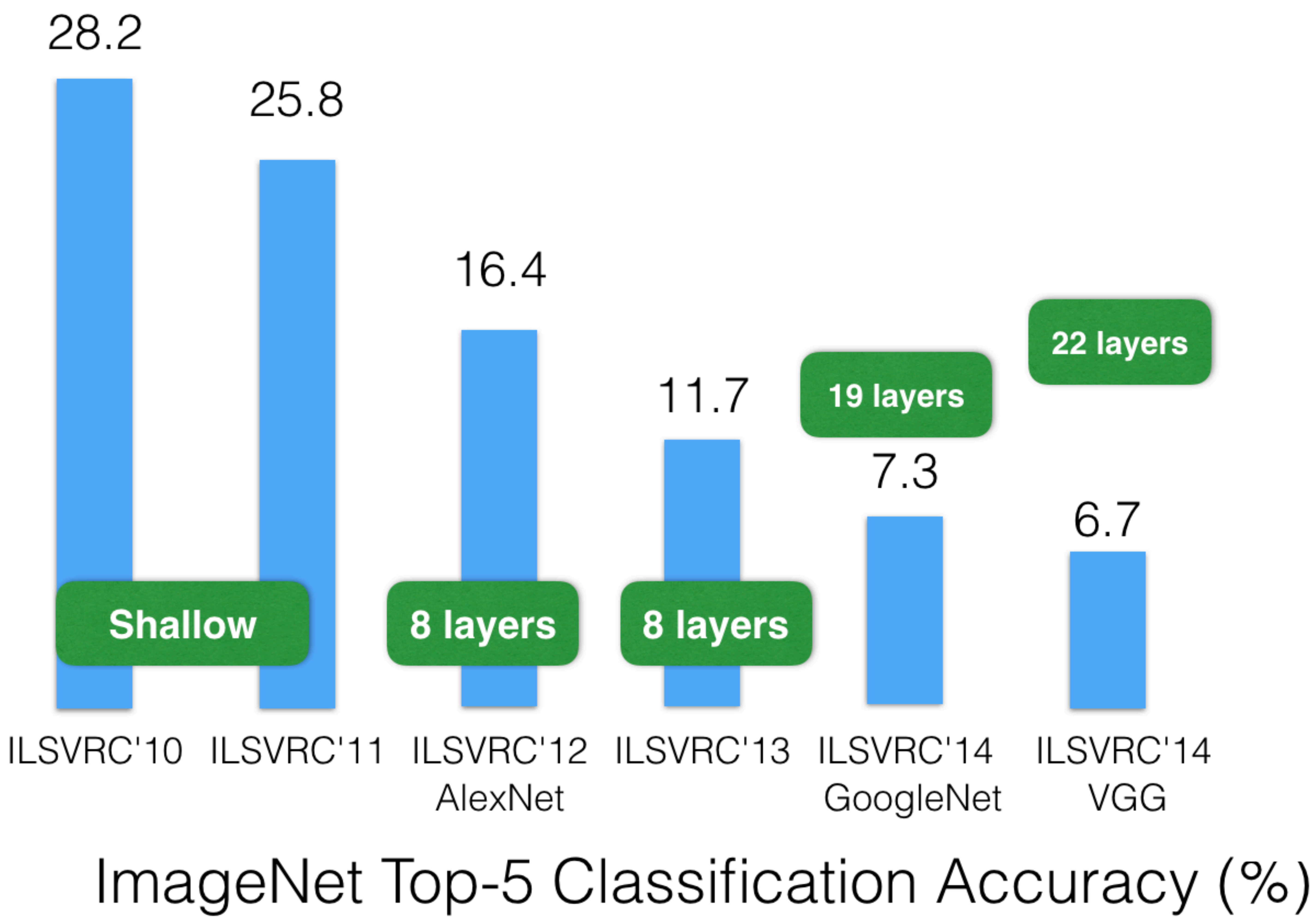


Complexity

	#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
Increase	11x	1x

11x11x3x96=35k




28.2
25.8
16.4
11.7
7.3
6.7
Shallow
8 layers
8 layers
19 layers
22 layers
ILSVRC'10
ILSVRC'11
ILSVRC'12
AlexNet
ILSVRC'13
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ImageNet Top-5 Classification Accuracy (%)

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

A. AlexNet contains 8 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 layers. The first five are convolutional layers.

B. The last three layers are fully connected layers.

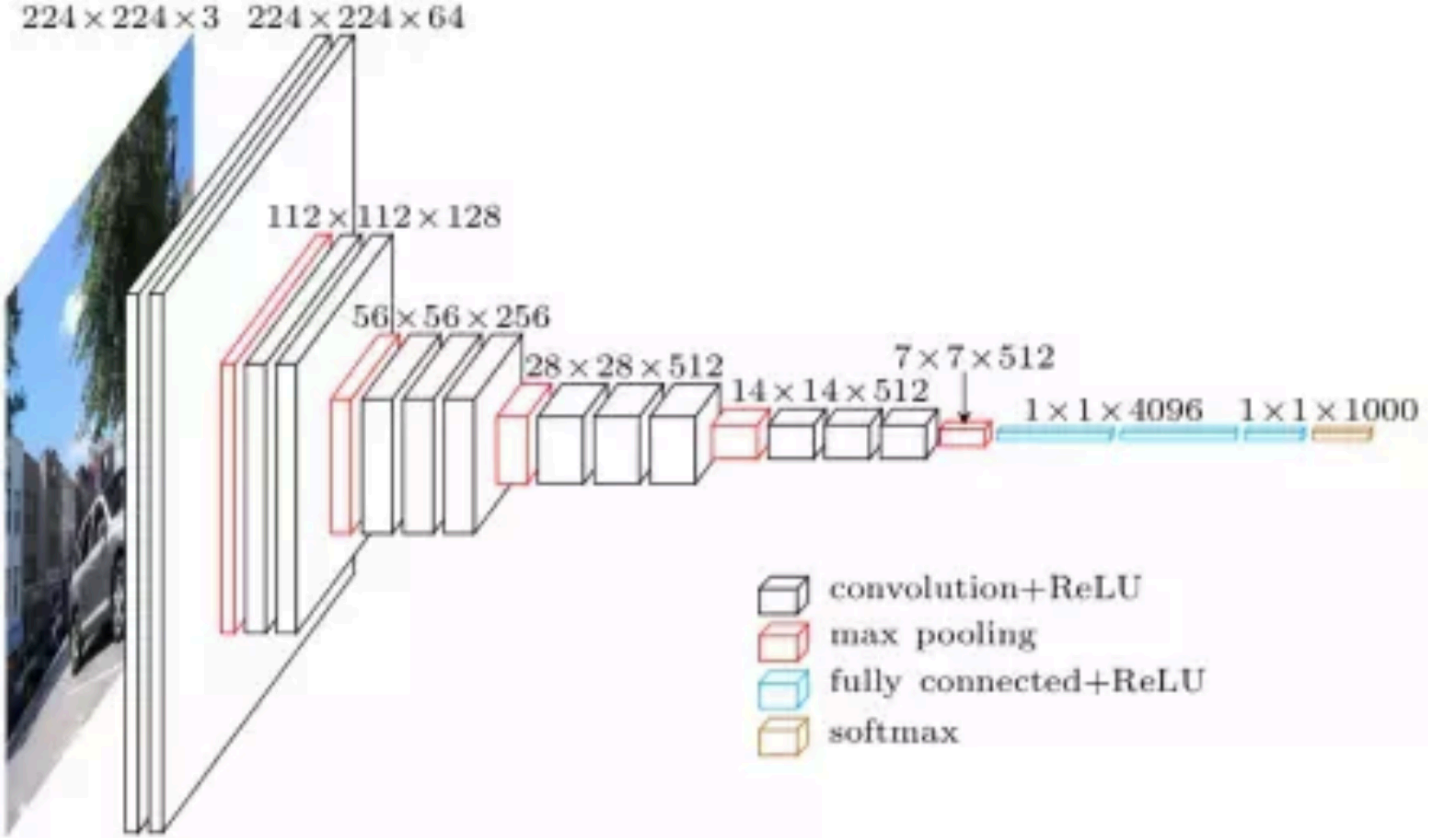
C. Some of the convolutional layers are followed by max-pooling (layers).

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



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

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)

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

What we’ve learned today

What we’ve learned today

- Brief review of convolutional computations

What we've learned today

- Brief review of convolutional computations
- Convolutional Neural Networks

What we've learned today

- Brief review of convolutional computations
- Convolutional Neural Networks
  - LeNet (first conv nets)

What we've learned today

- Brief review of convolutional computations
- Convolutional Neural Networks
    - LeNet (first conv nets)
    - AlexNet

What we've learned today

- Brief review of convolutional computations
- Convolutional Neural Networks
    - LeNet (first conv nets)
    - AlexNet
- PyTorch demo

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