CS 540 Introduction to Artificial Intelligence
Convolutional Neural Networks (II)

University of Wisconsin–Madison
Fall 2025, Section 3
October 24, 2025

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

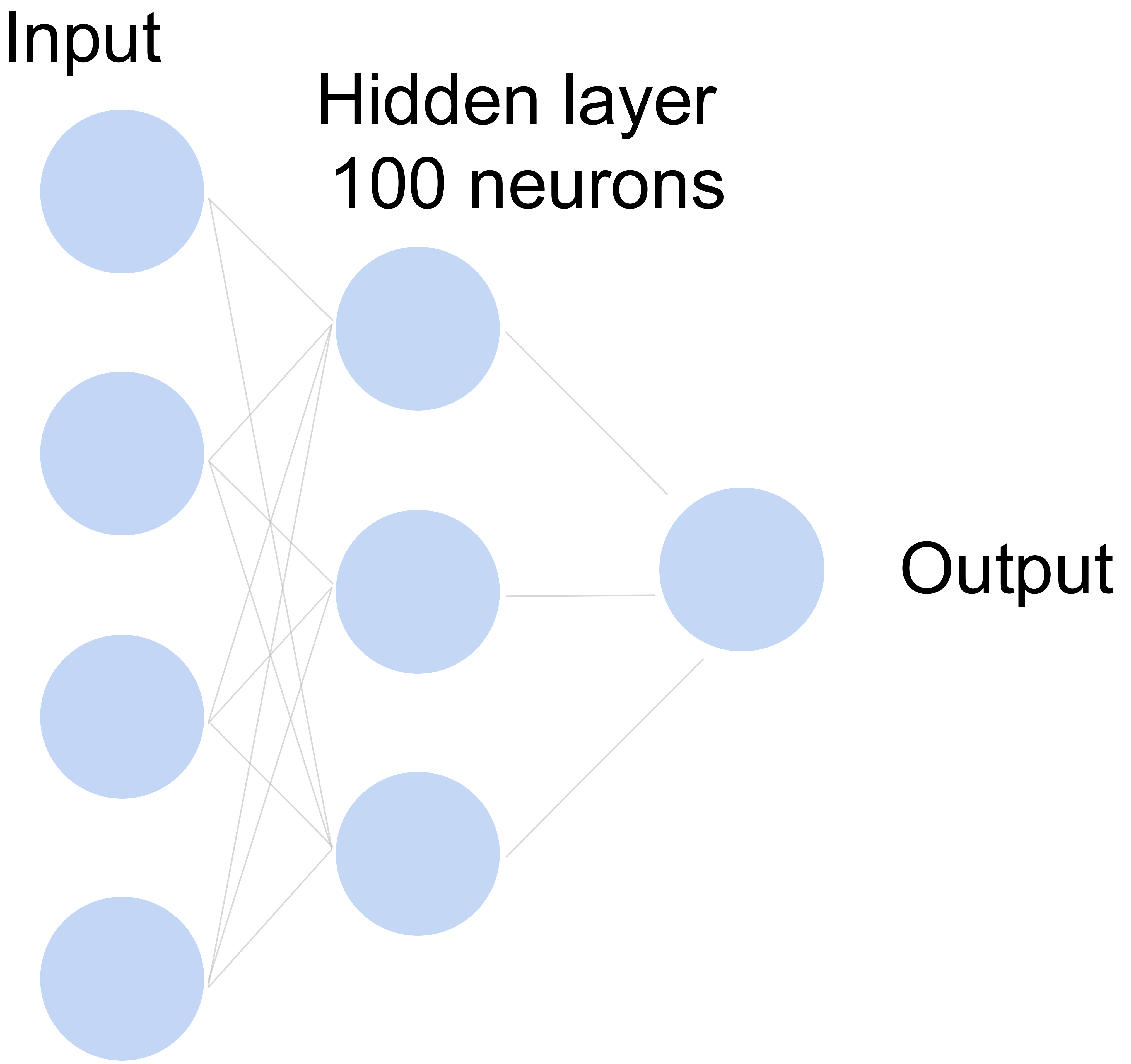
How to classify

Cats vs. dogs?

36M floats in a RGB image!

Fully Connected Networks

Cats vs. dogs?



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

Review: 2-D Convolution

Input

0	1	2
3	4	5
6	7	8

*

Kernel

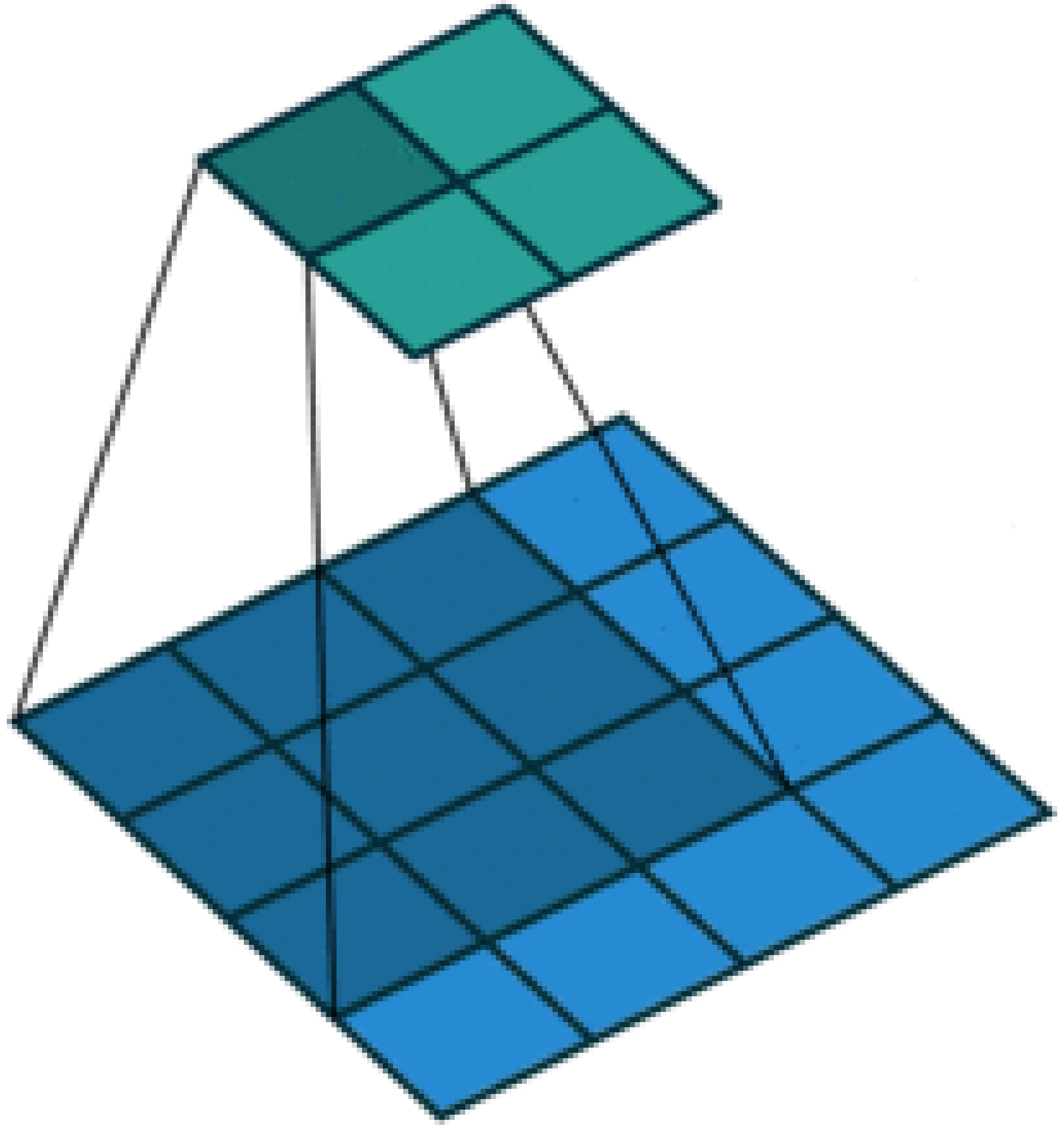
0	1
2	3

=

Output

19	25
37	43

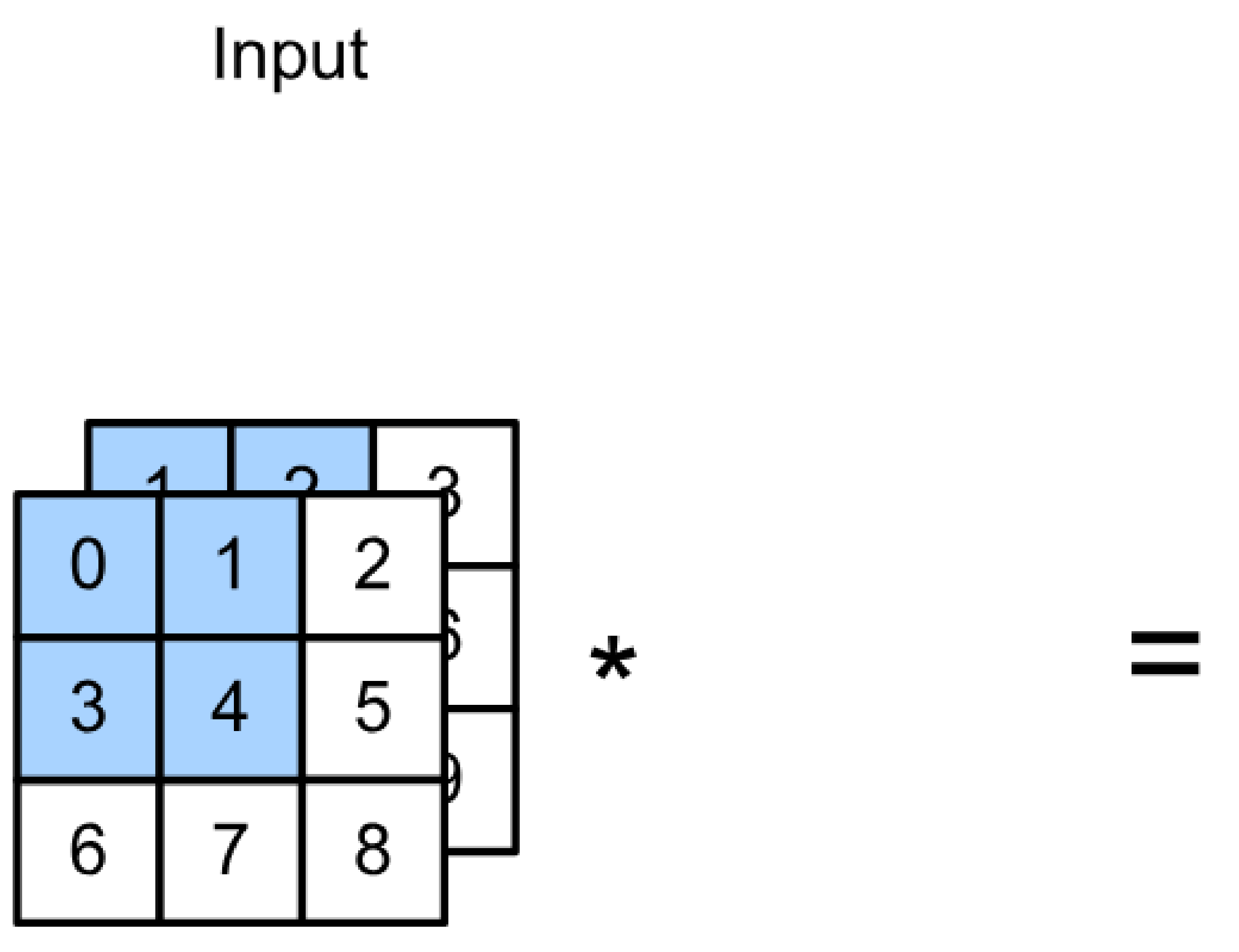
$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$



(vdumoulin@ Github)

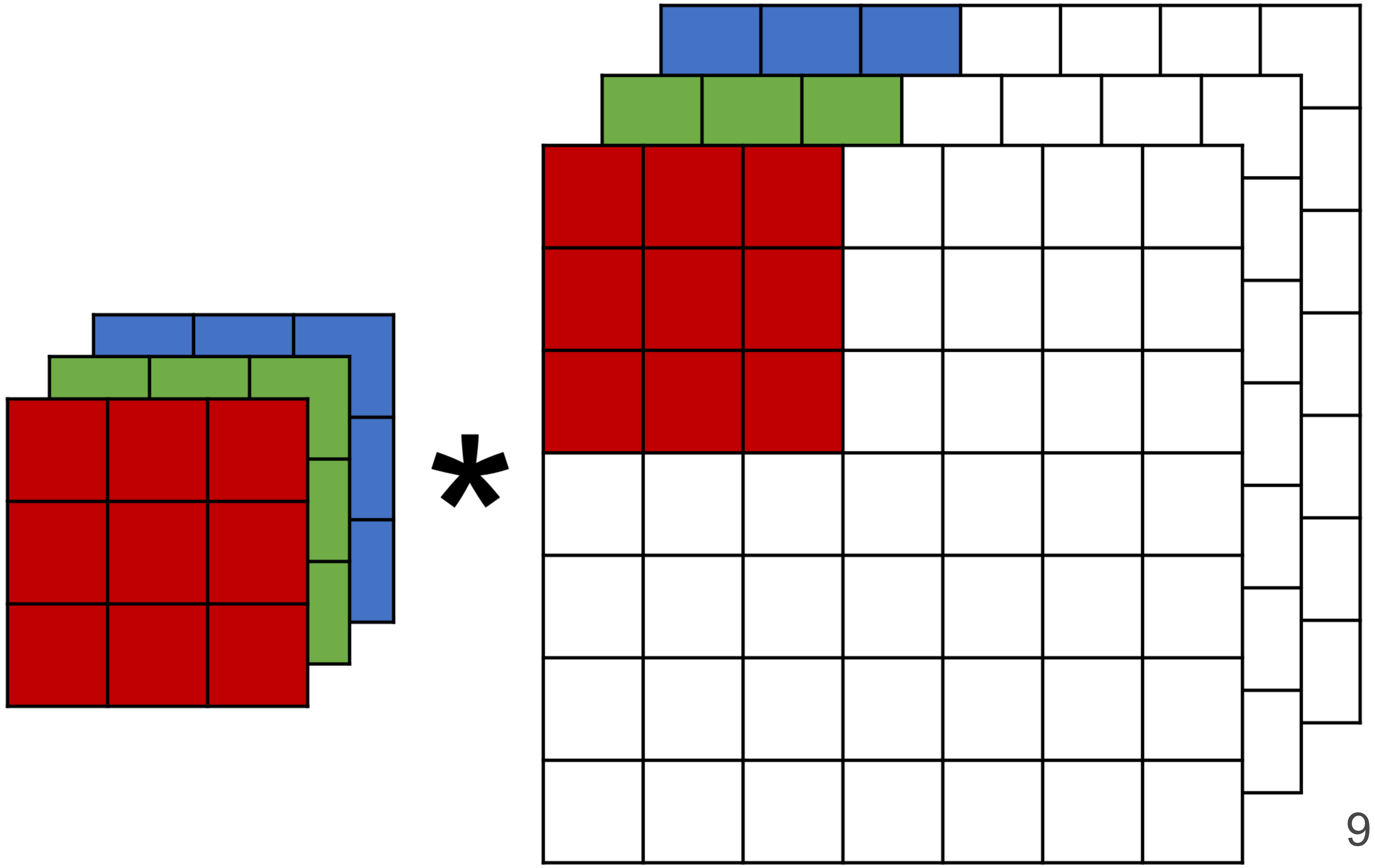
Review: Multiple Input Channels

- 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



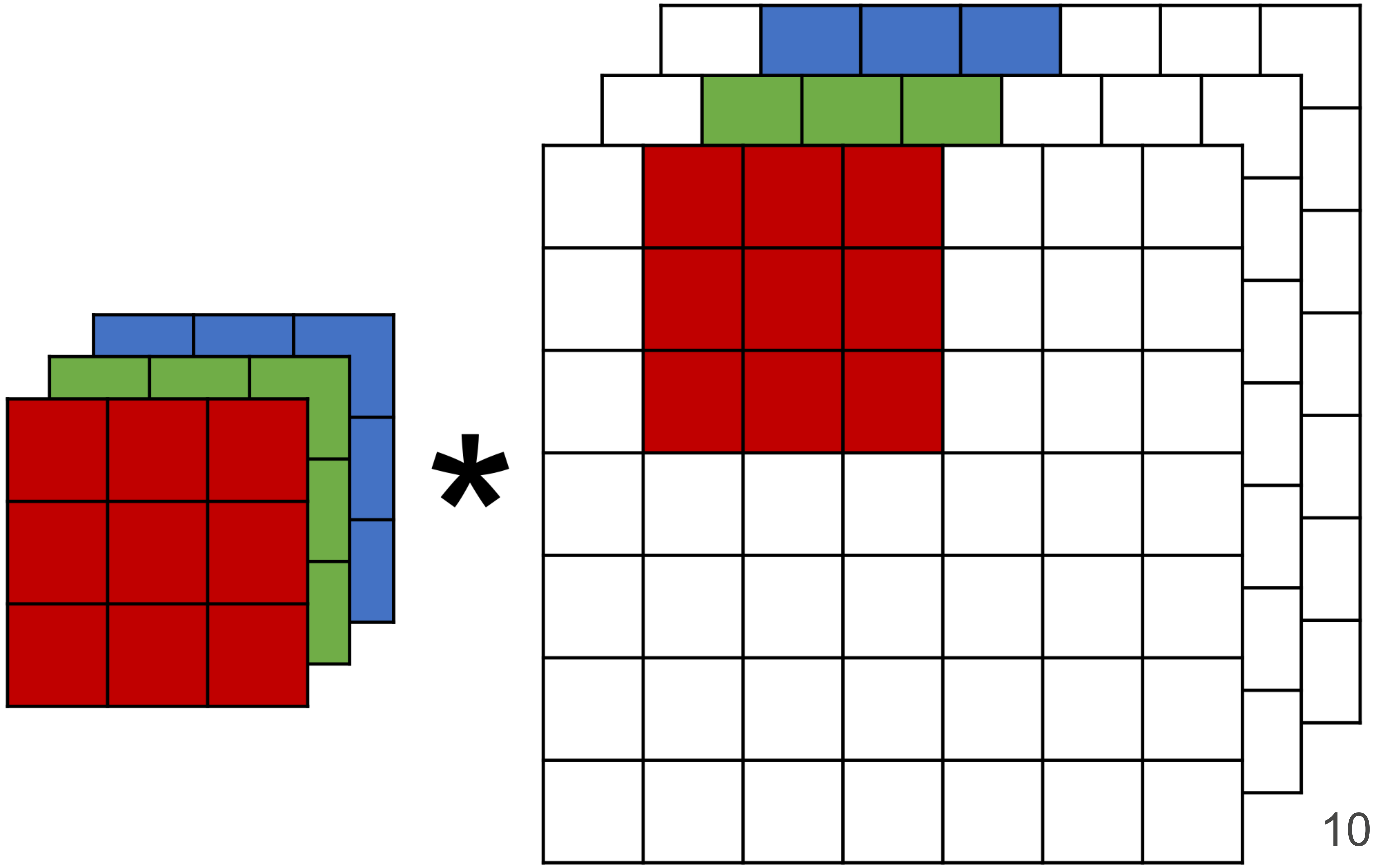
Review: Multiple Input Channels

- 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



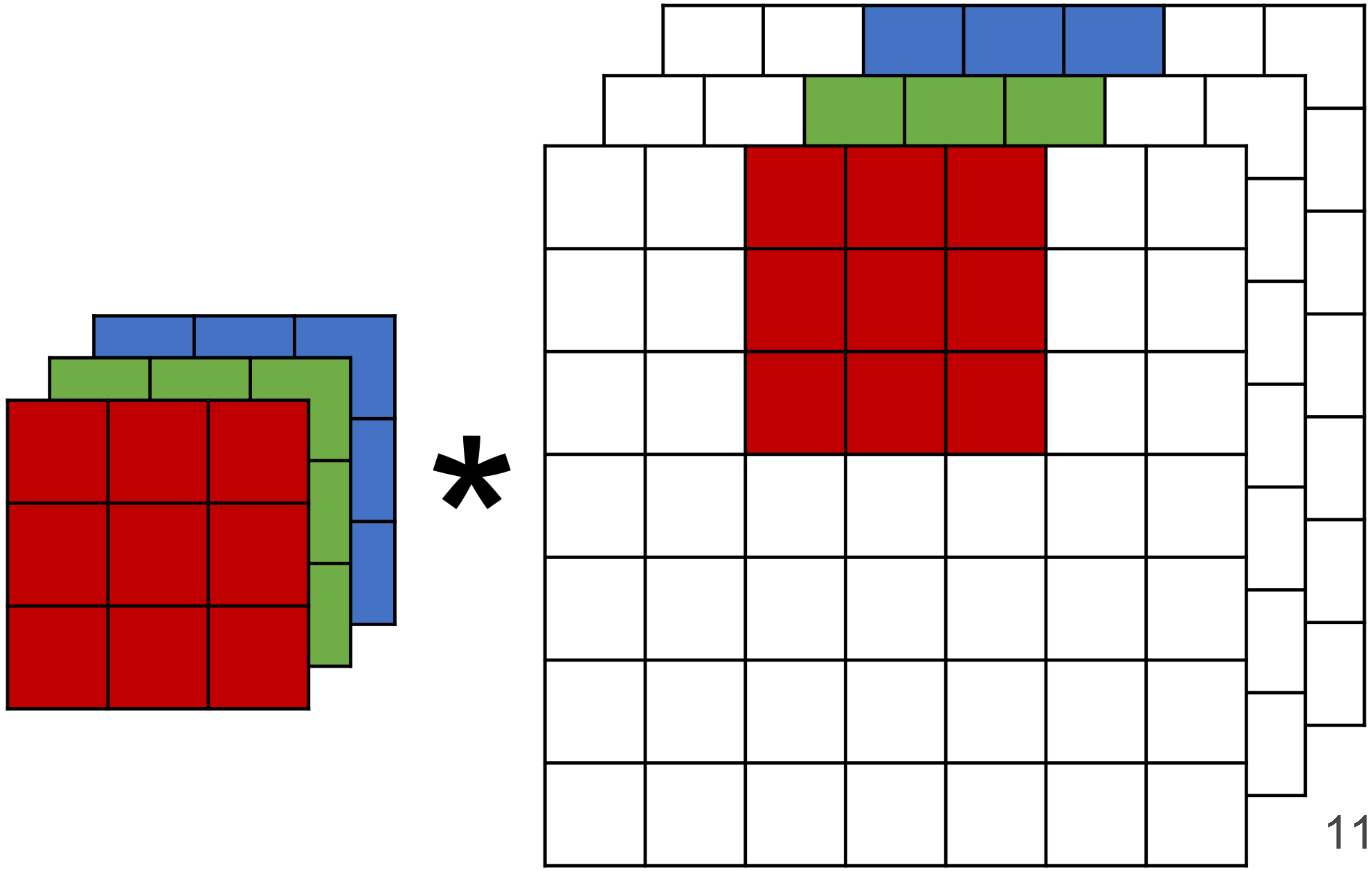
Review: Multiple Input Channels

- 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



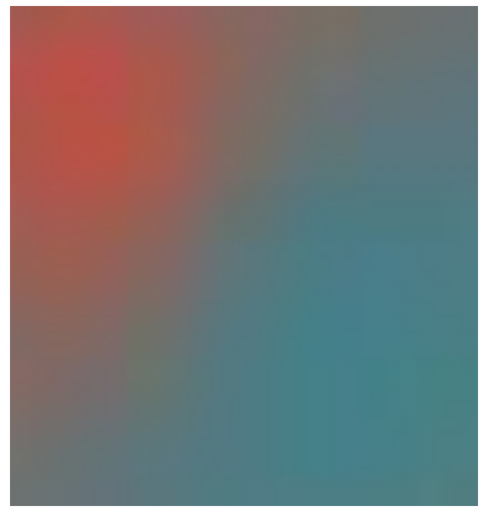
Review: Multiple Input Channels

- 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



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)



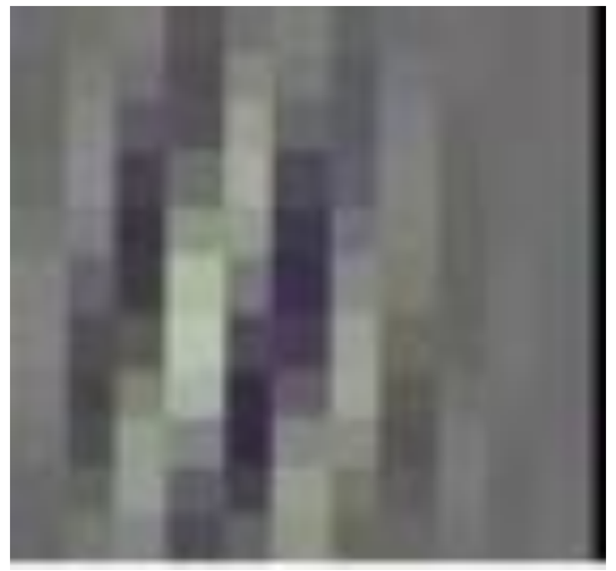
One filter
(3 channels)

*

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

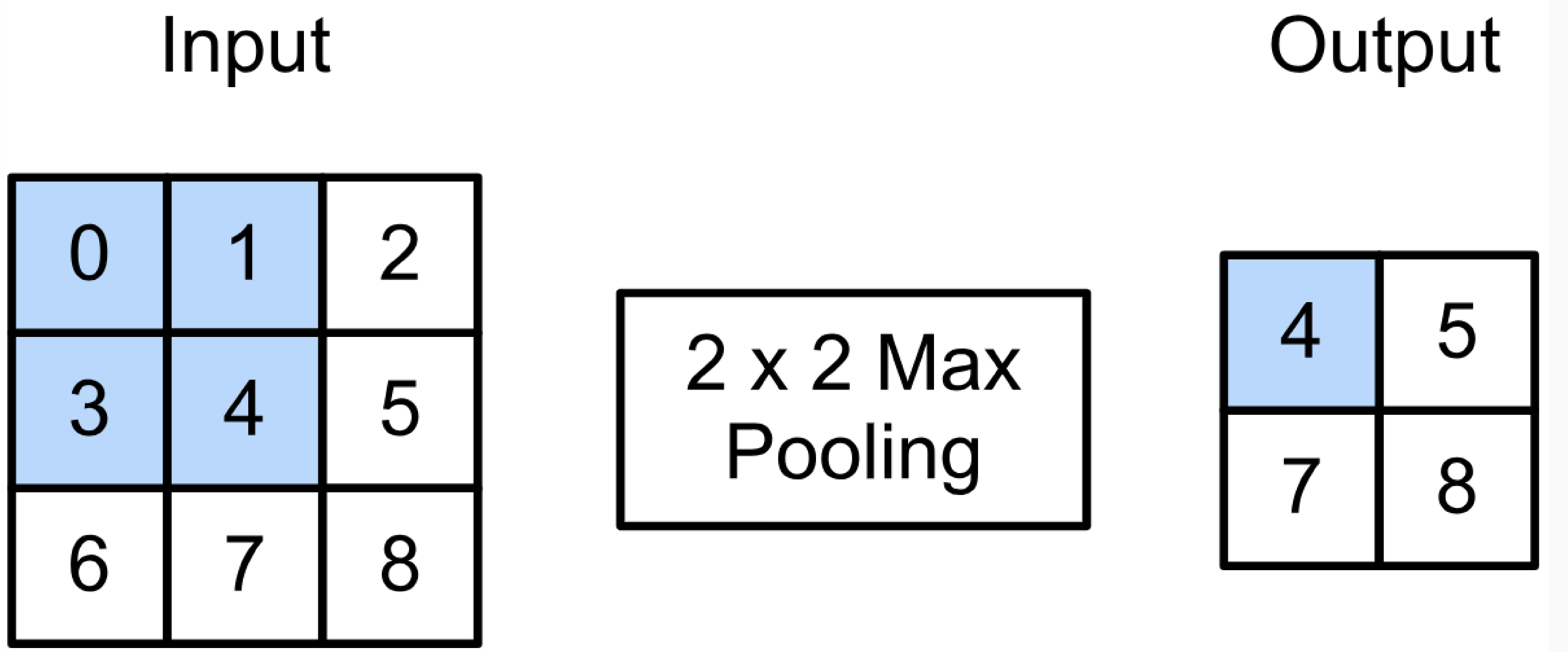
Pad

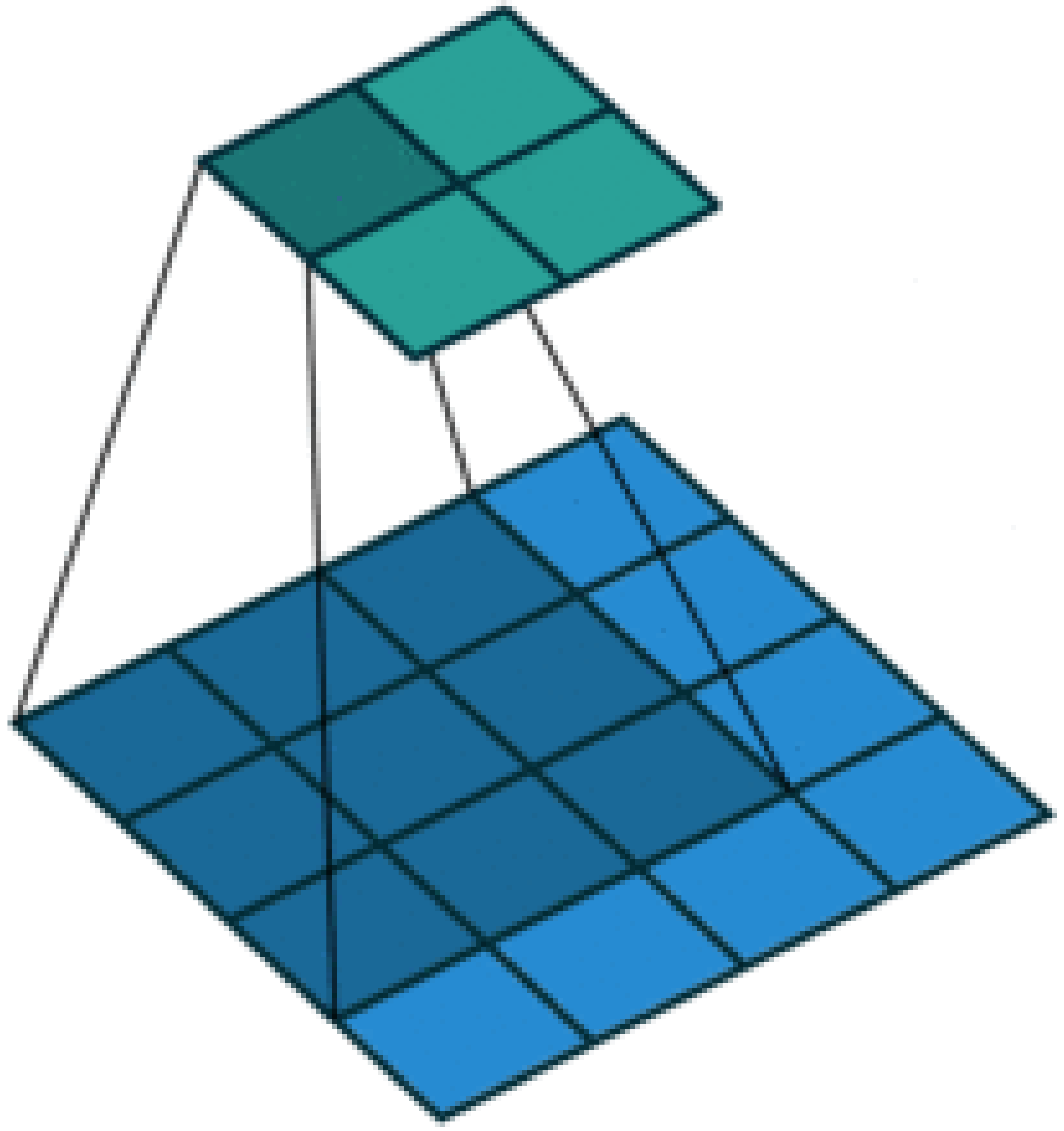
Stride

Pooling Layer

2-D Max Pooling

- Returns the maximal value in the sliding window



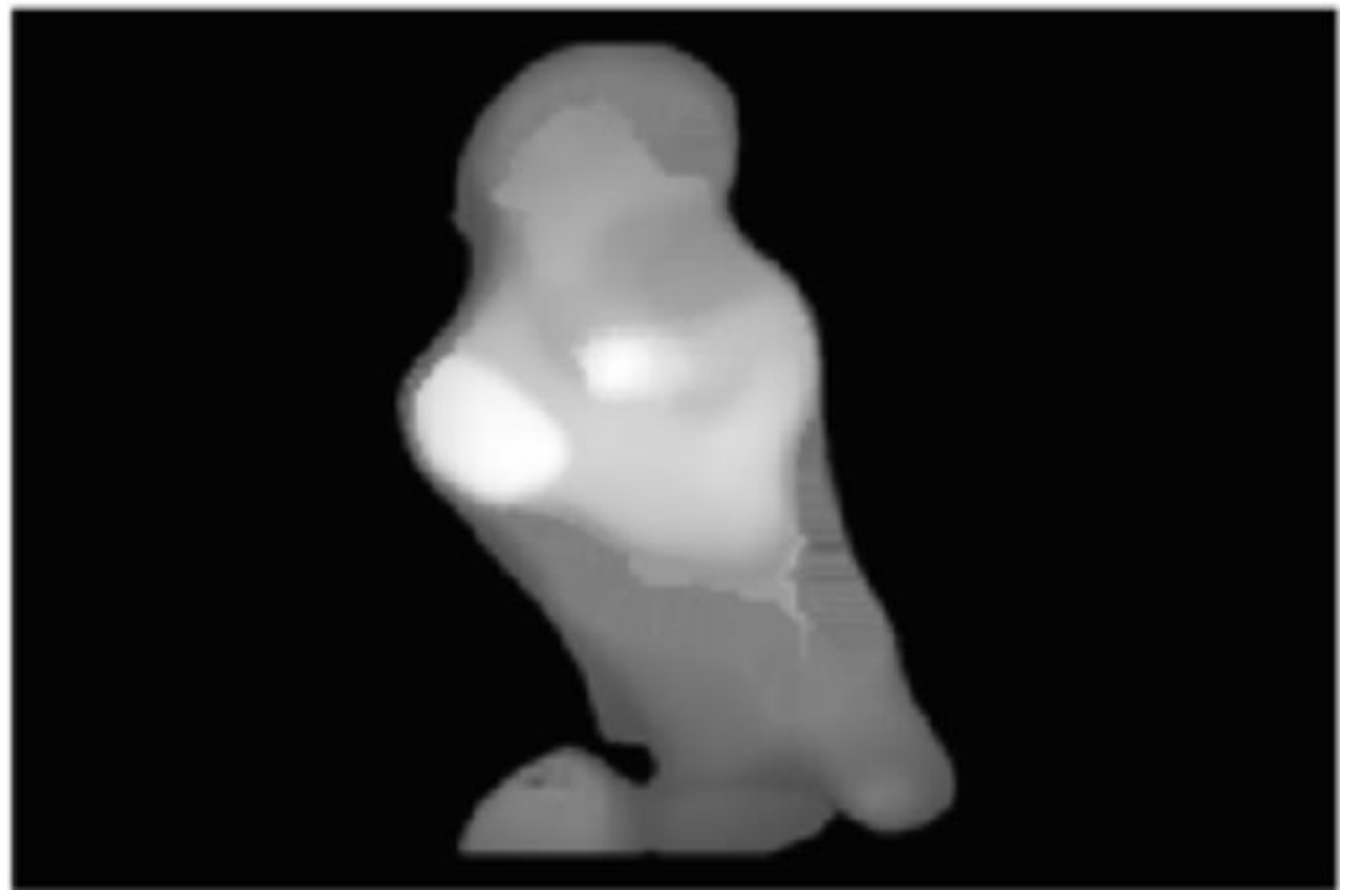


$$max(0,1,3,4) = 4$$

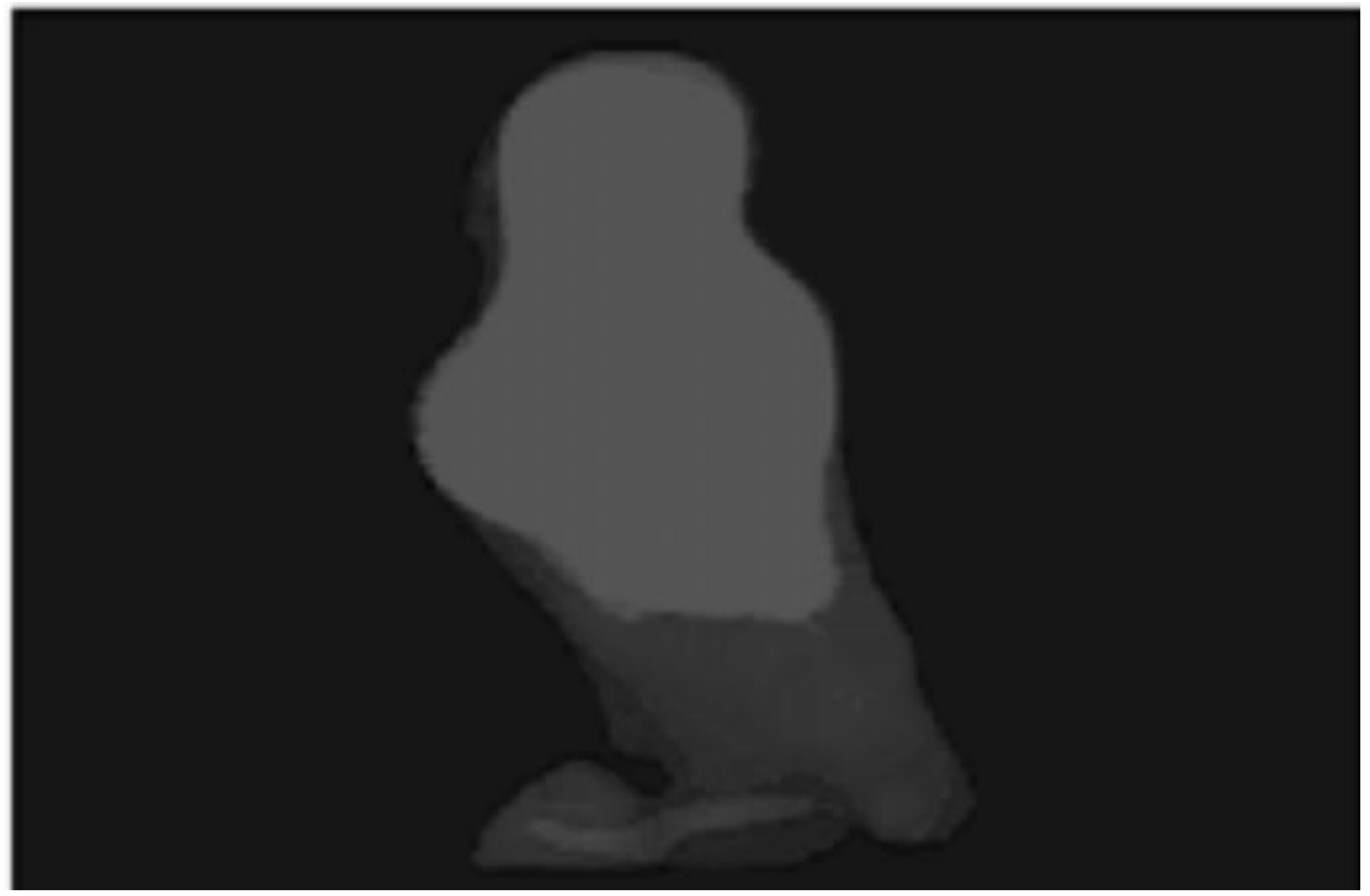
Average Pooling

- 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

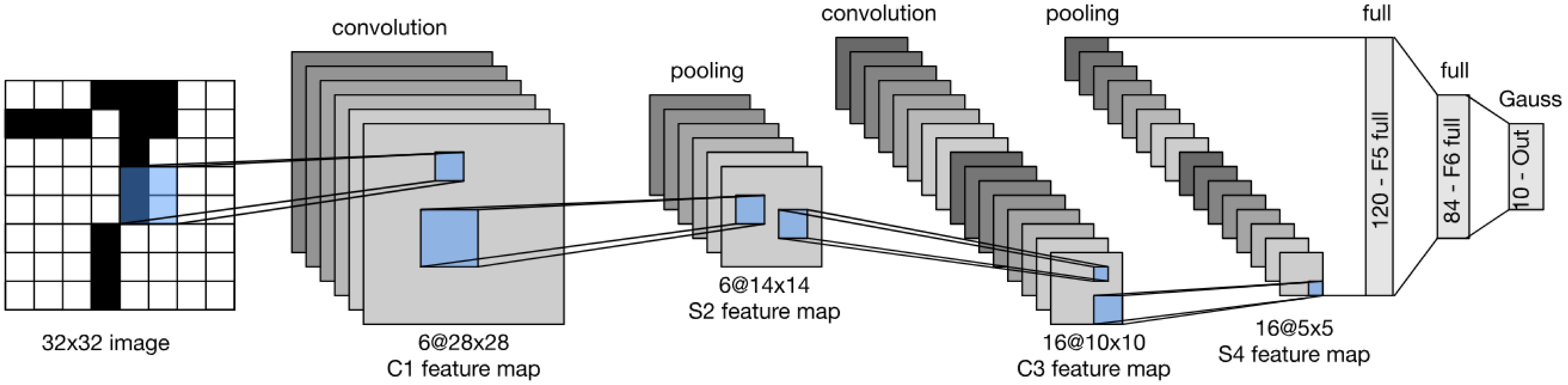
Max pooling



Average pooling



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?

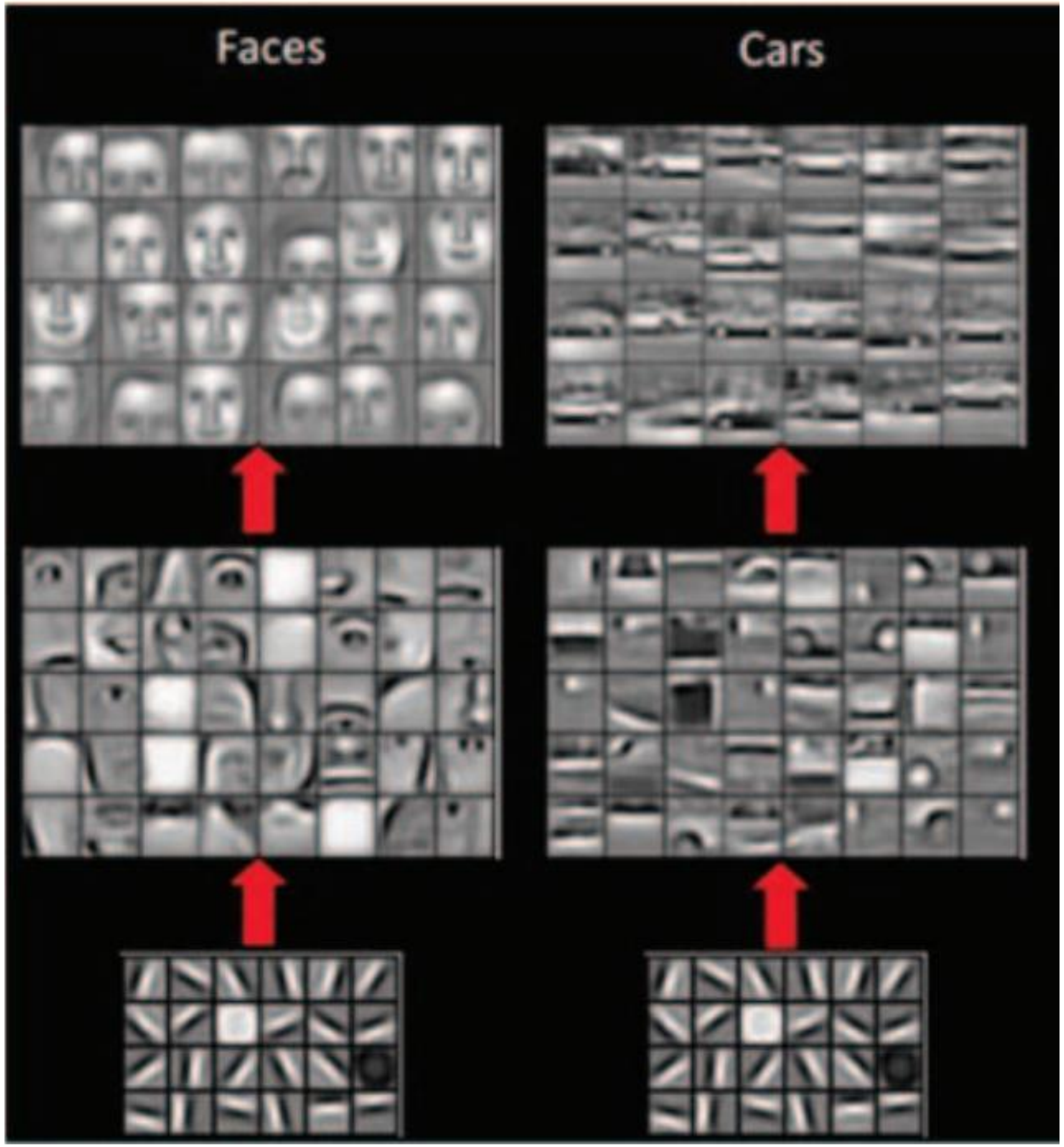
Look for features of cats or dogs in the image and use for decision.

- 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

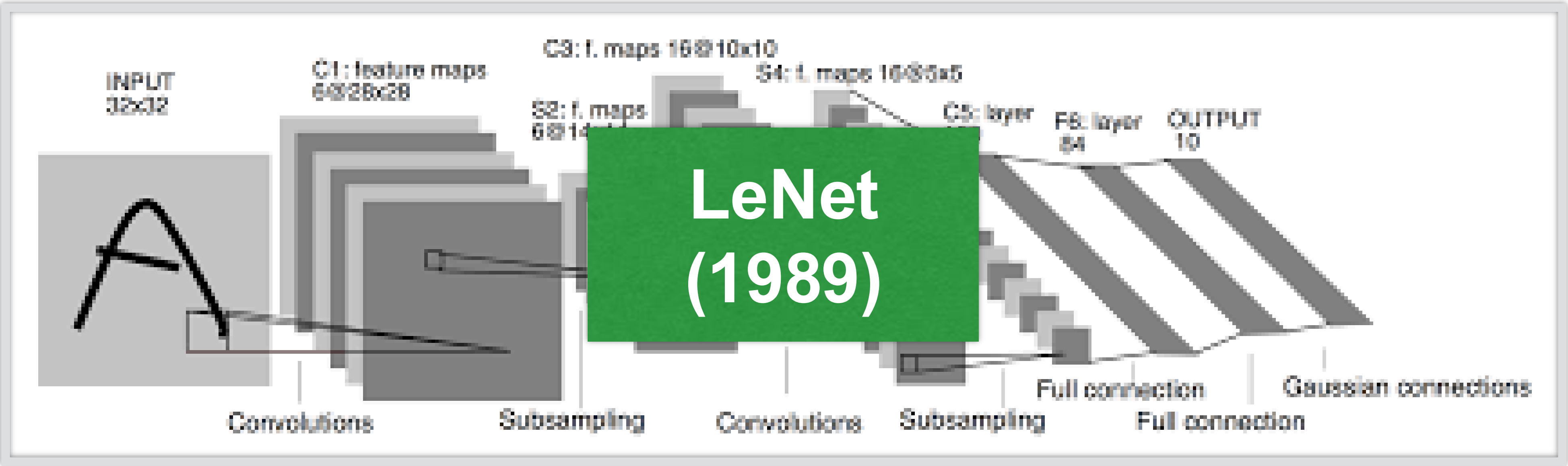
Middle layers recognize parts of objects

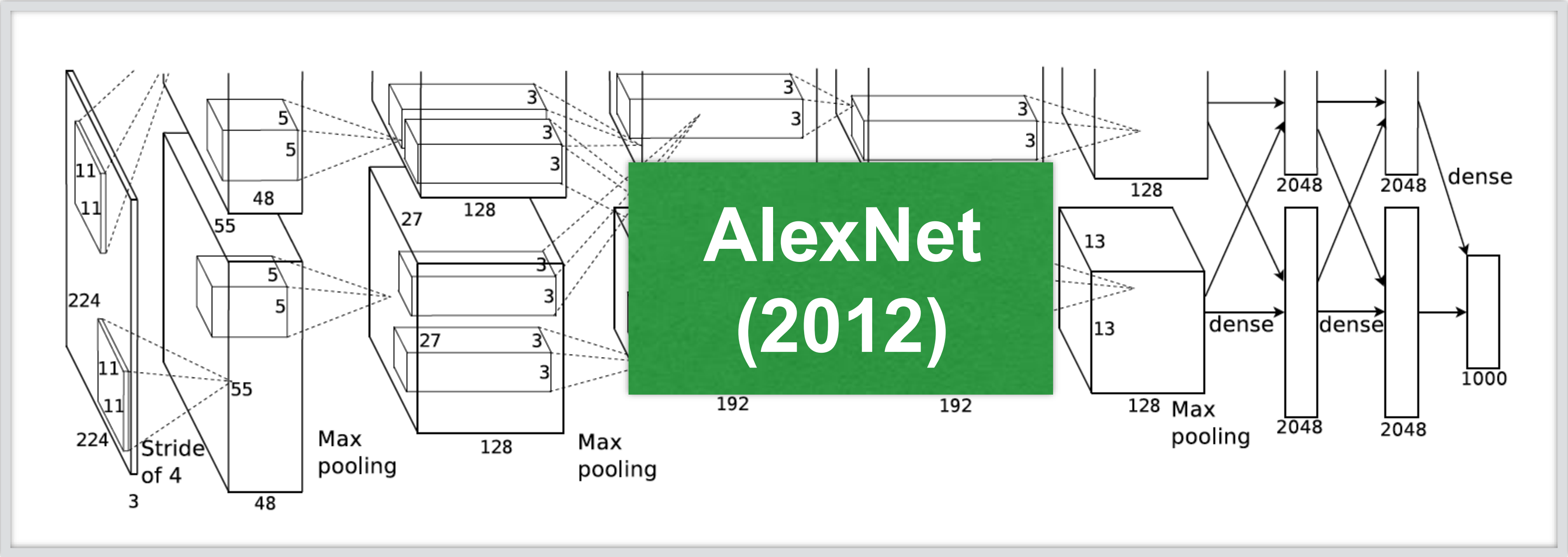
Early layers recognize simple patterns

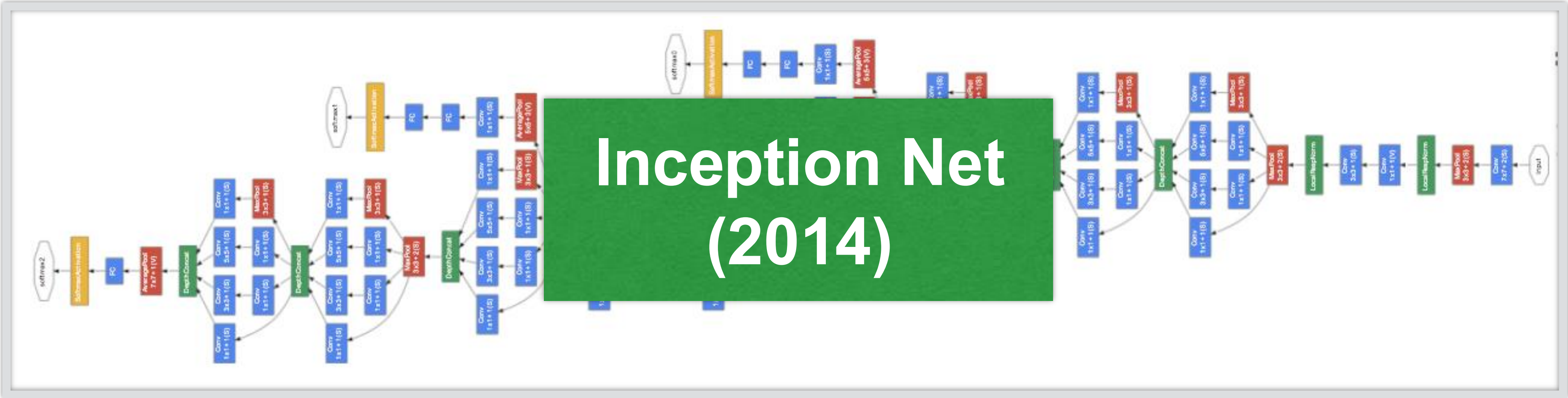
Adaptive Neuron Apoptosis for Accelerating Deep Learning on Large Scale Systems. Seigel et al. 2016.

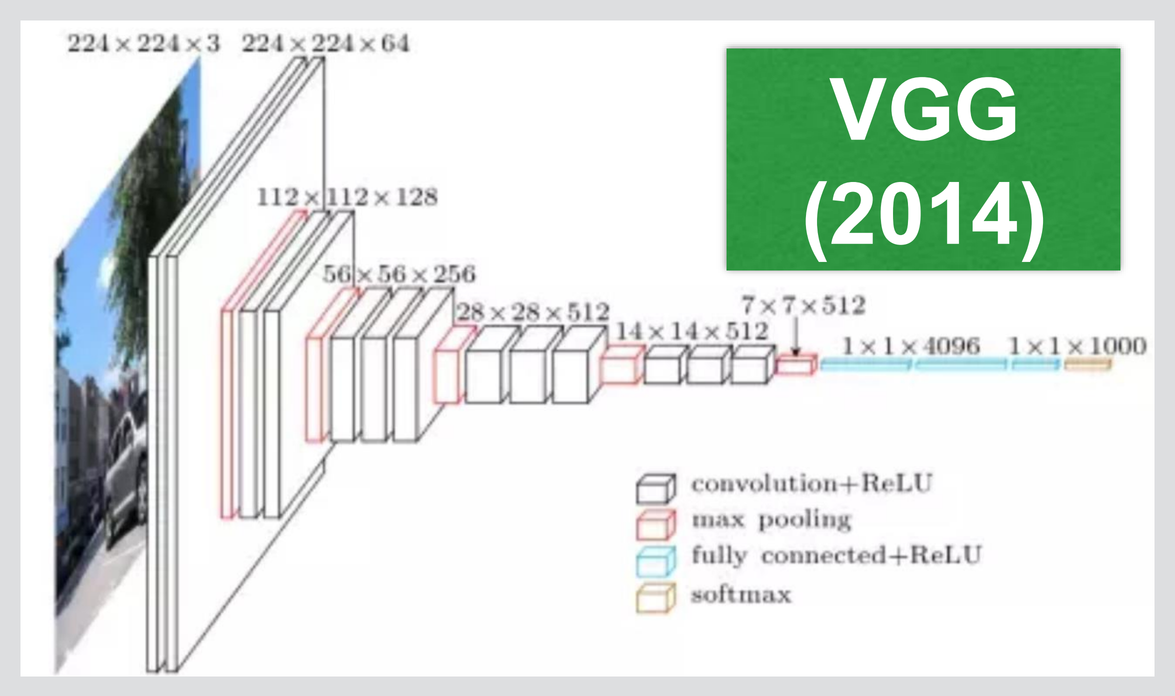
Convolutional Neural Networks Examples

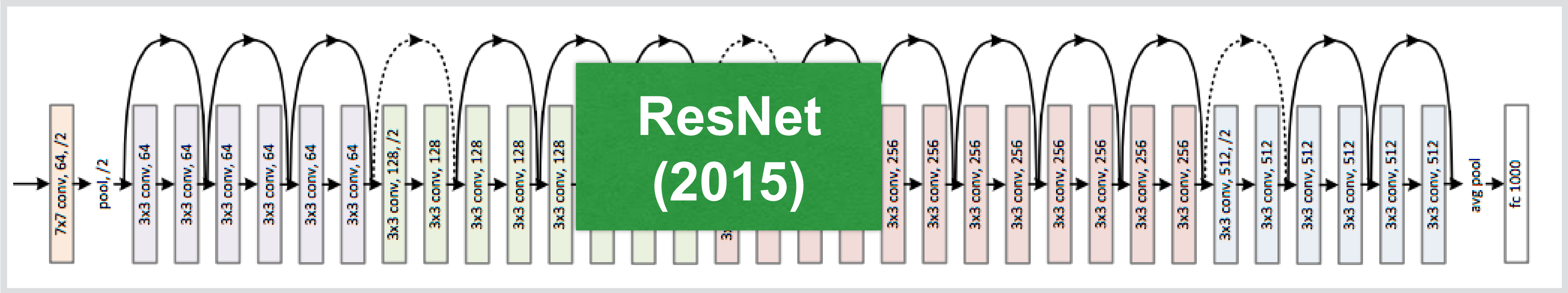
Evolution of neural net architectures

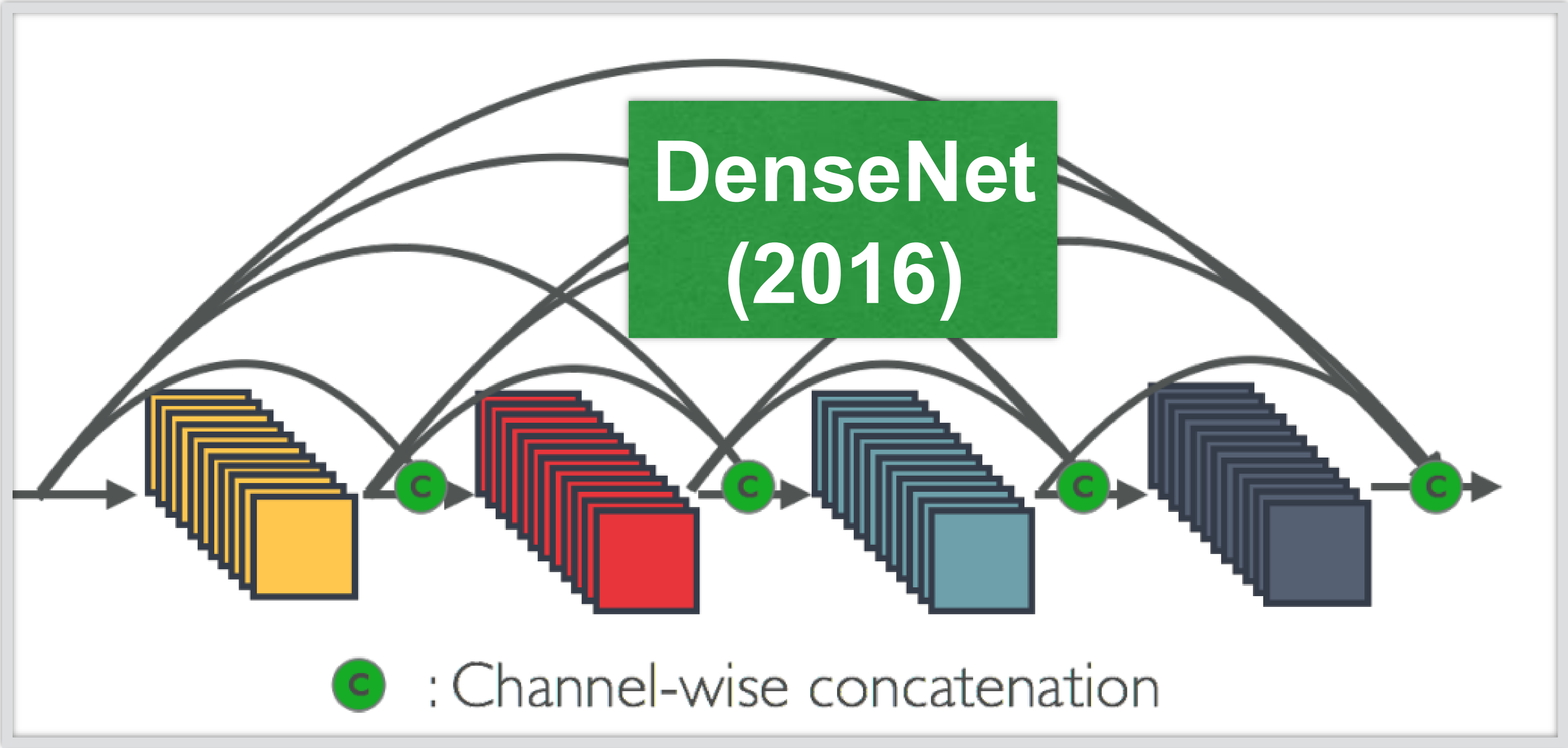








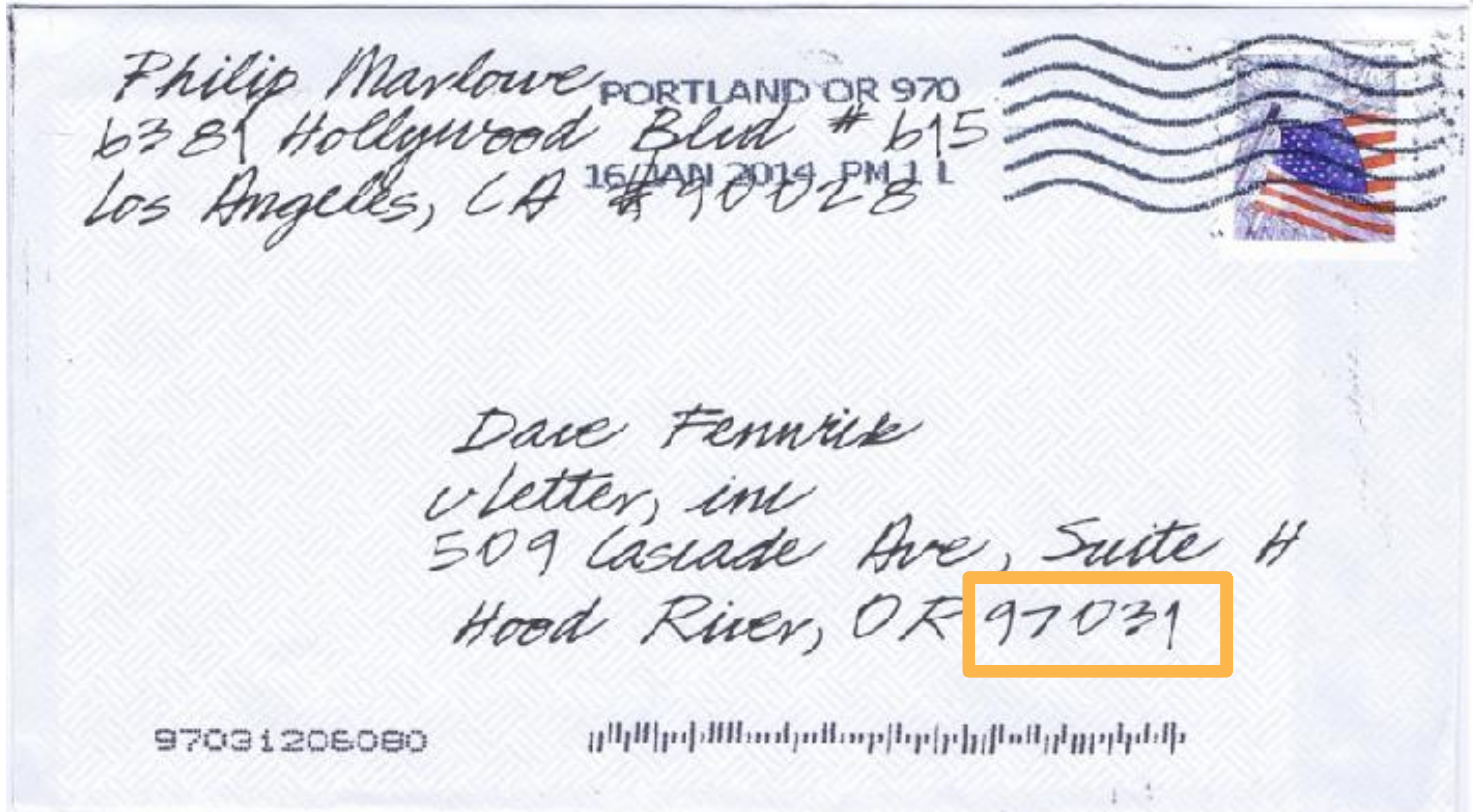


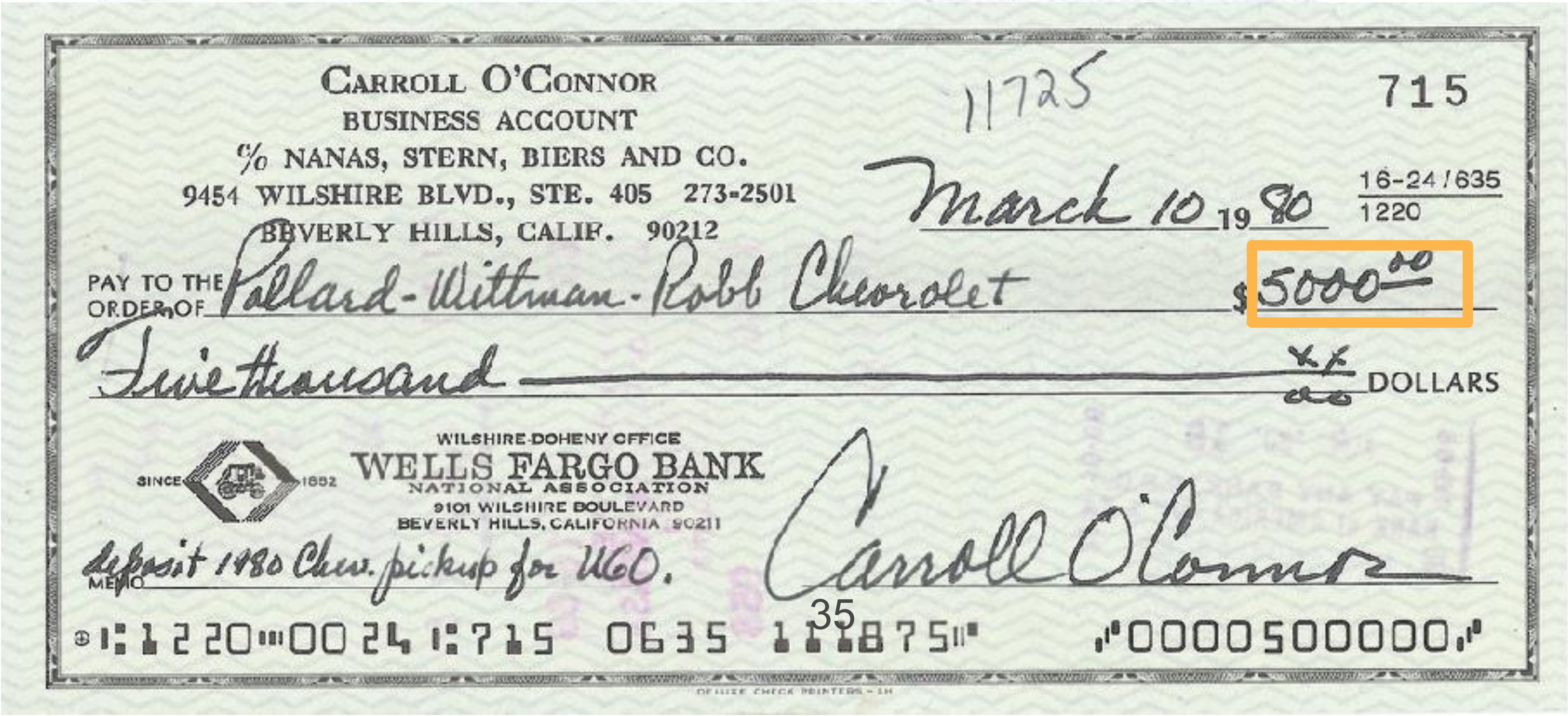


LeNet Architecture
(first convolutional neural net; 1989)

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

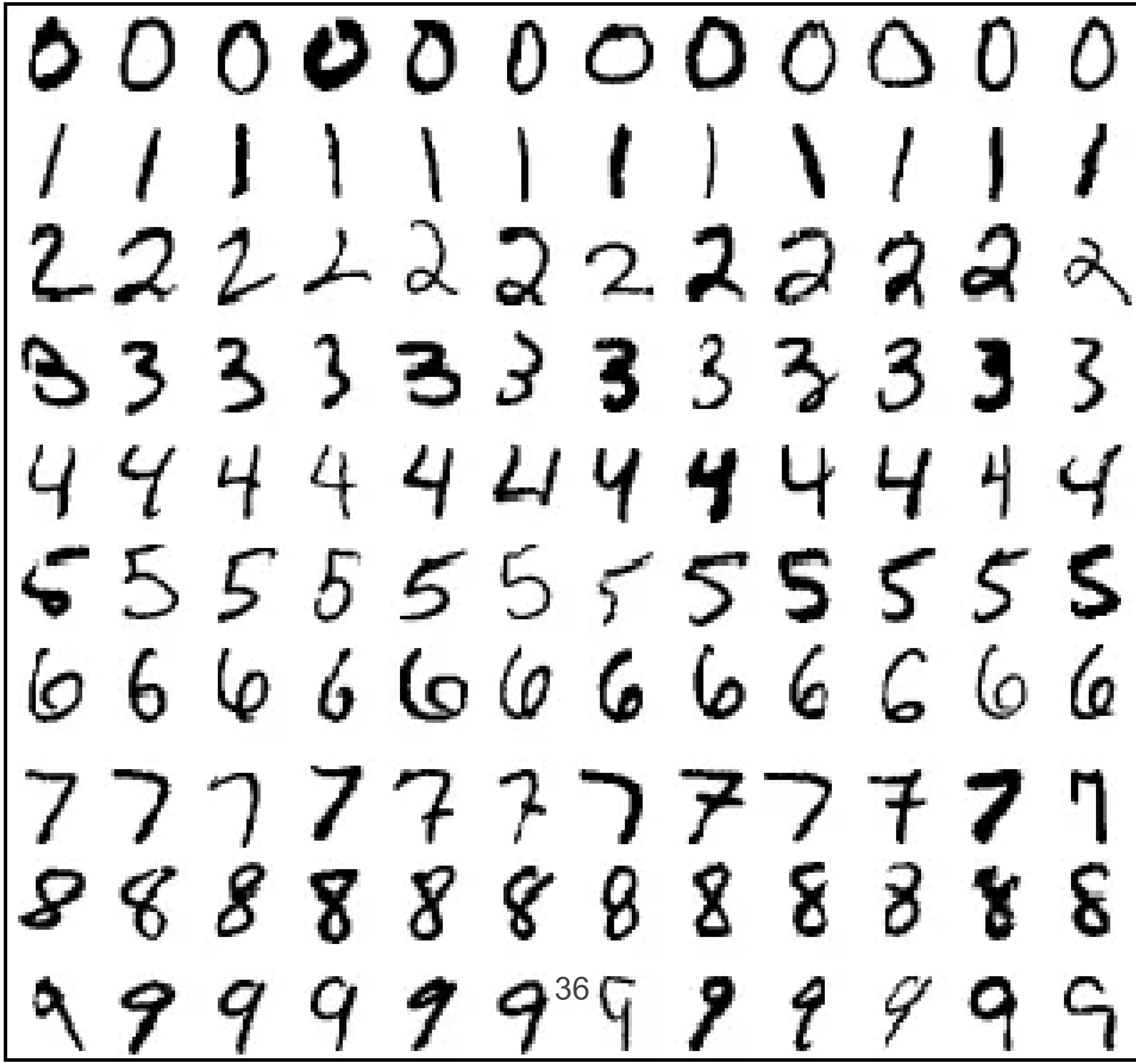
Handwritten Digit Recognition

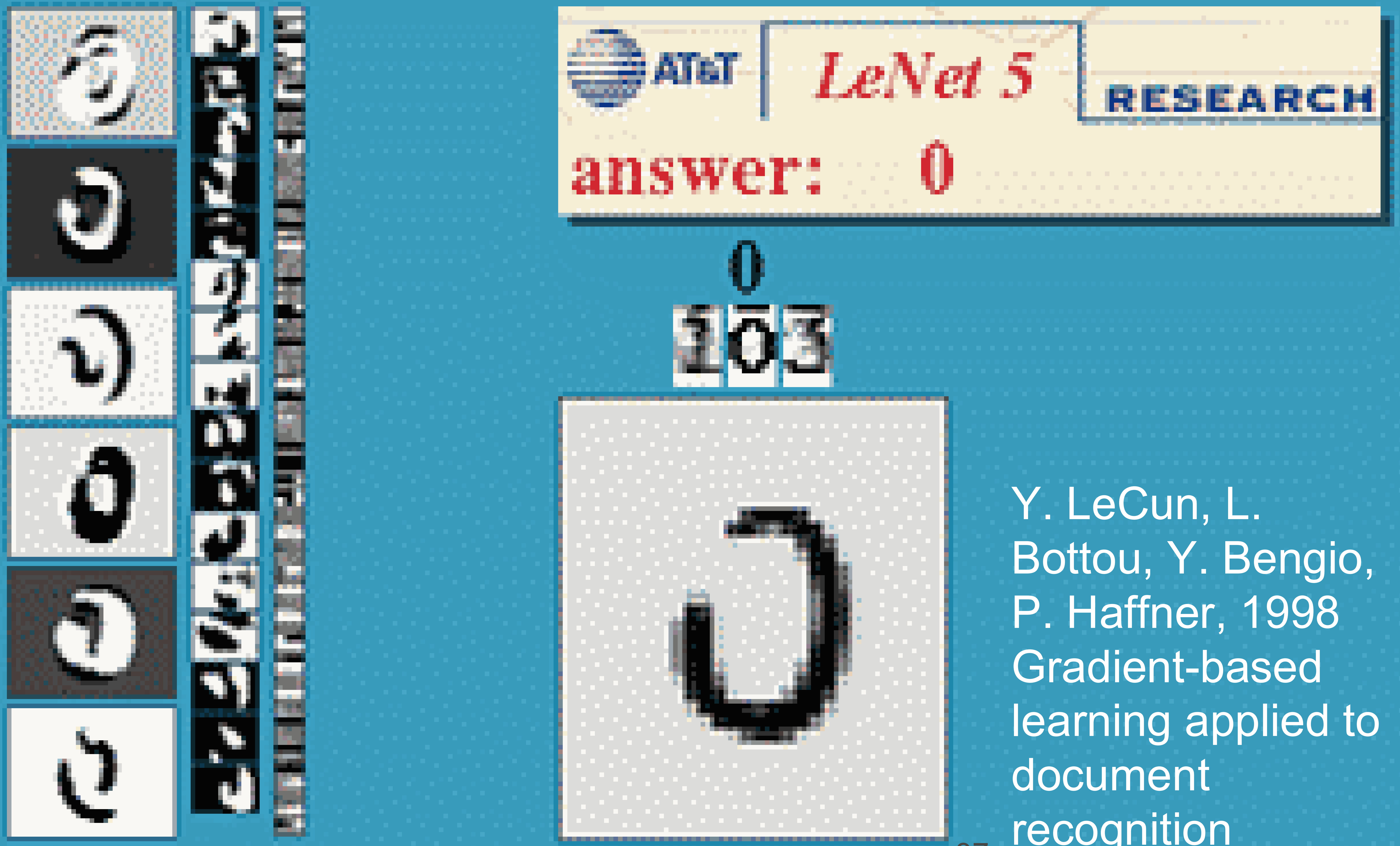




MNIST

- 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

```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

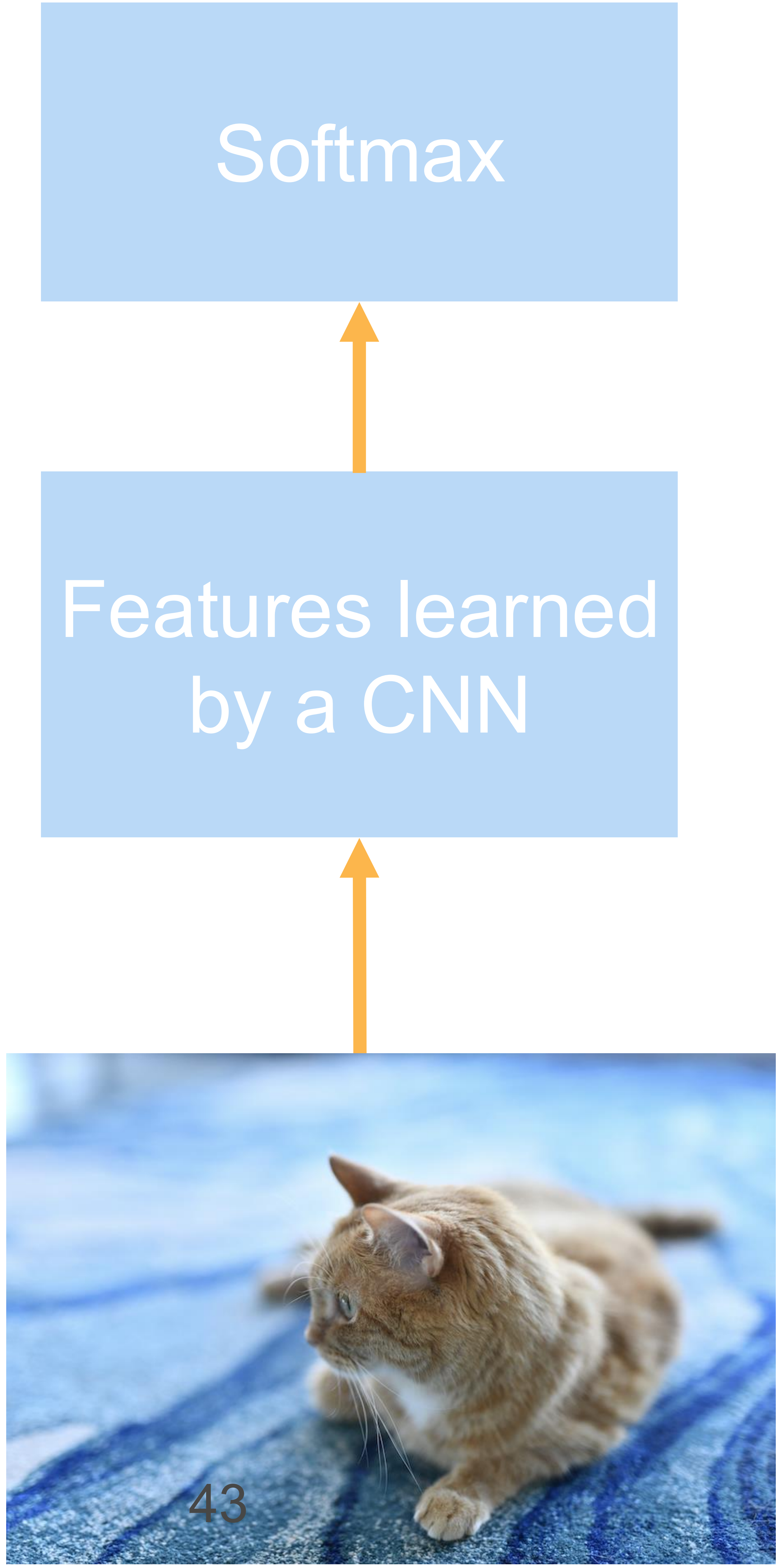
AlexNet

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

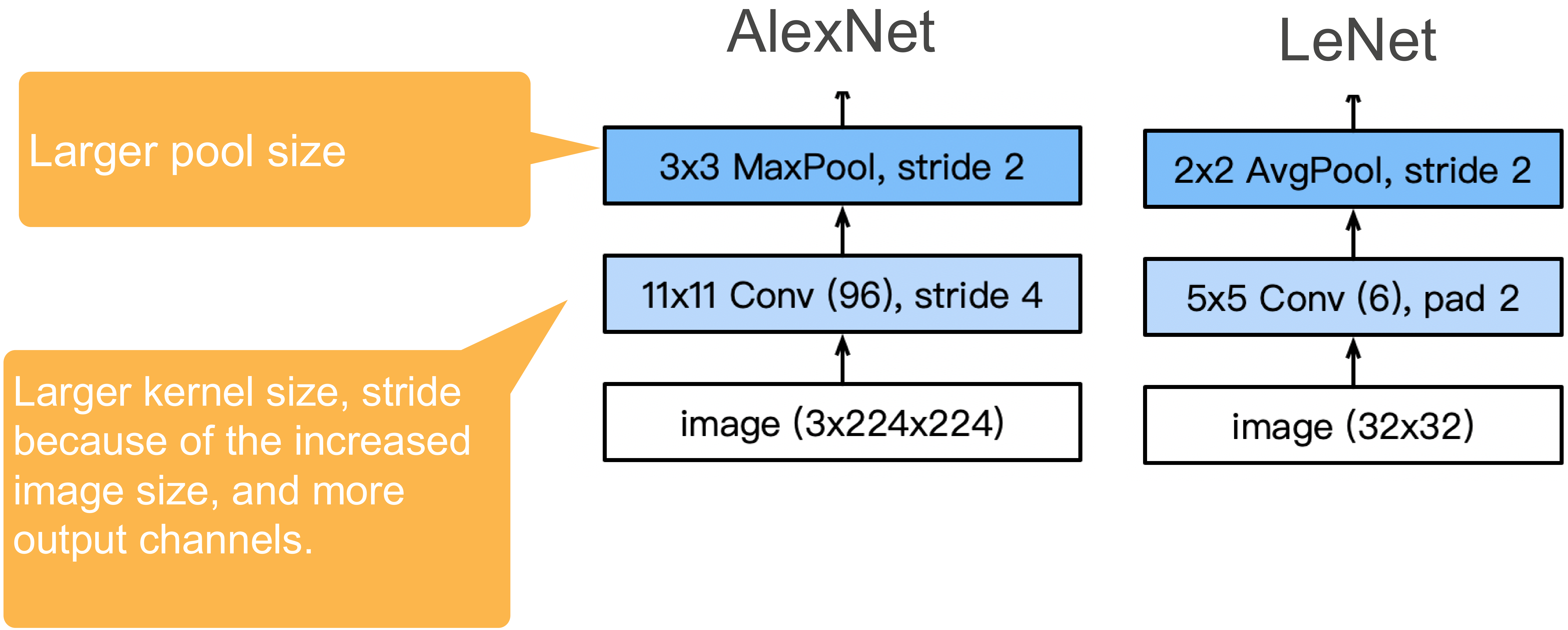
Deng et al. 2009

AlexNet

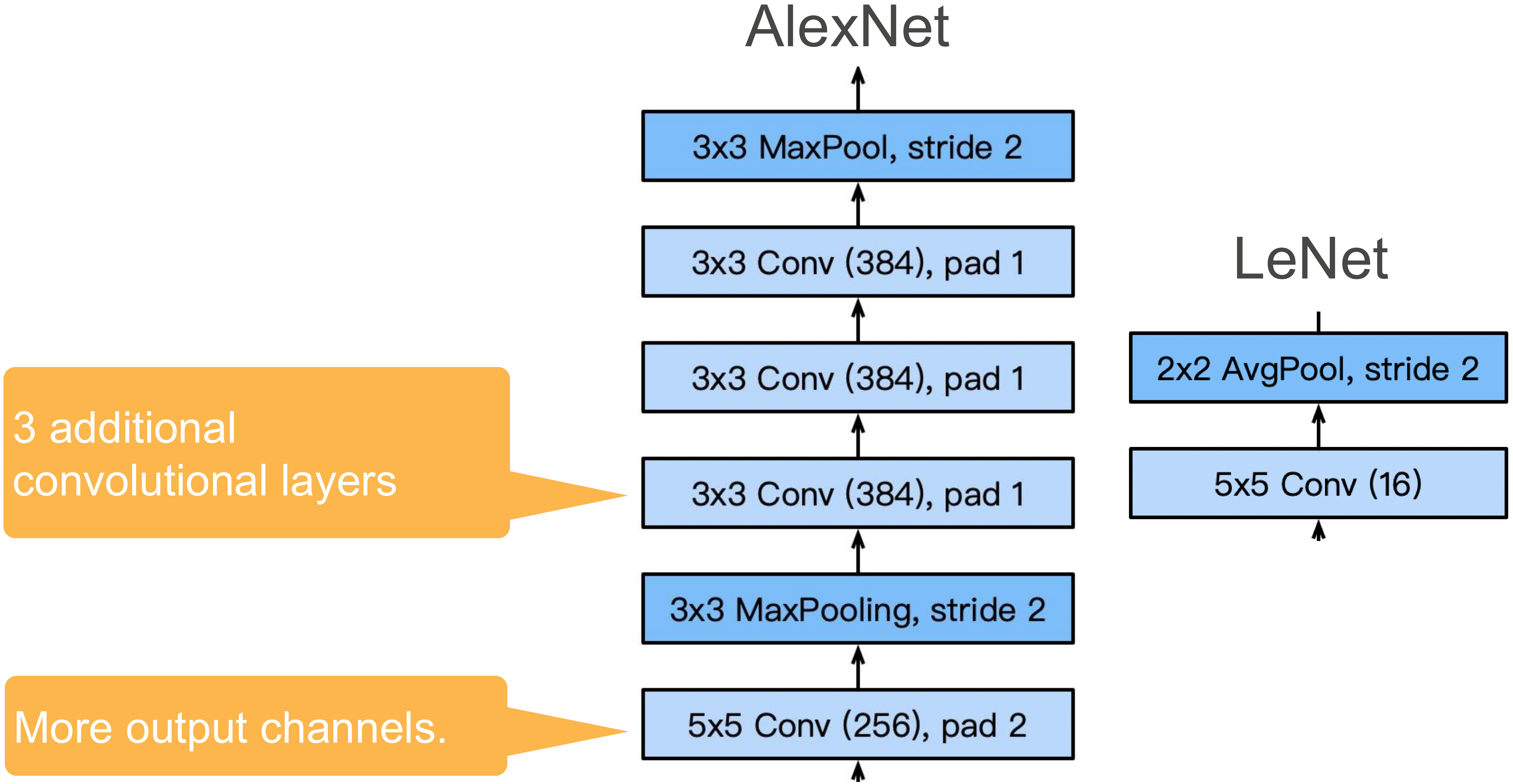
- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Paradigm shift for computer vision



AlexNet Architecture



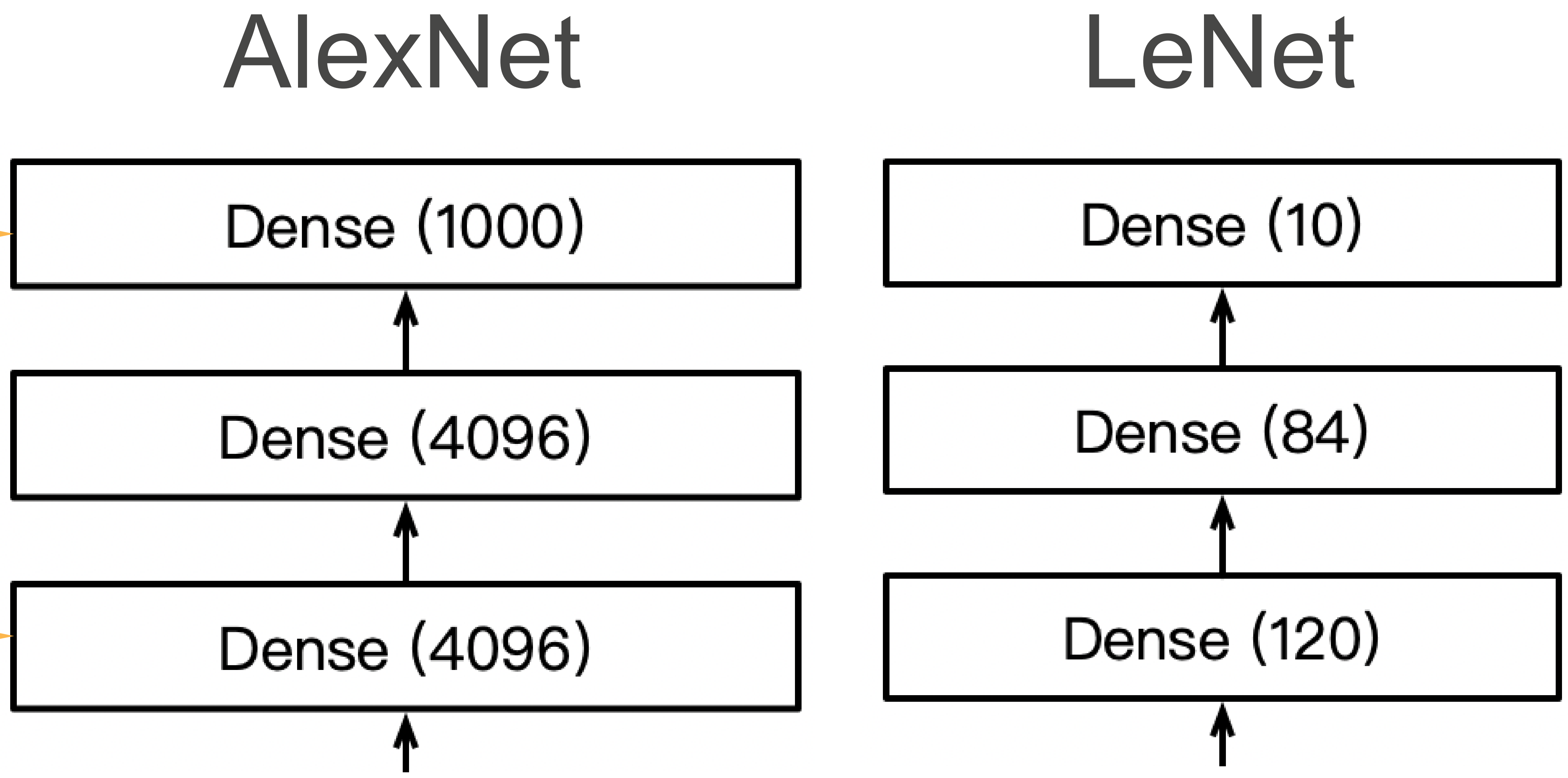
AlexNet Architecture



AlexNet Architecture

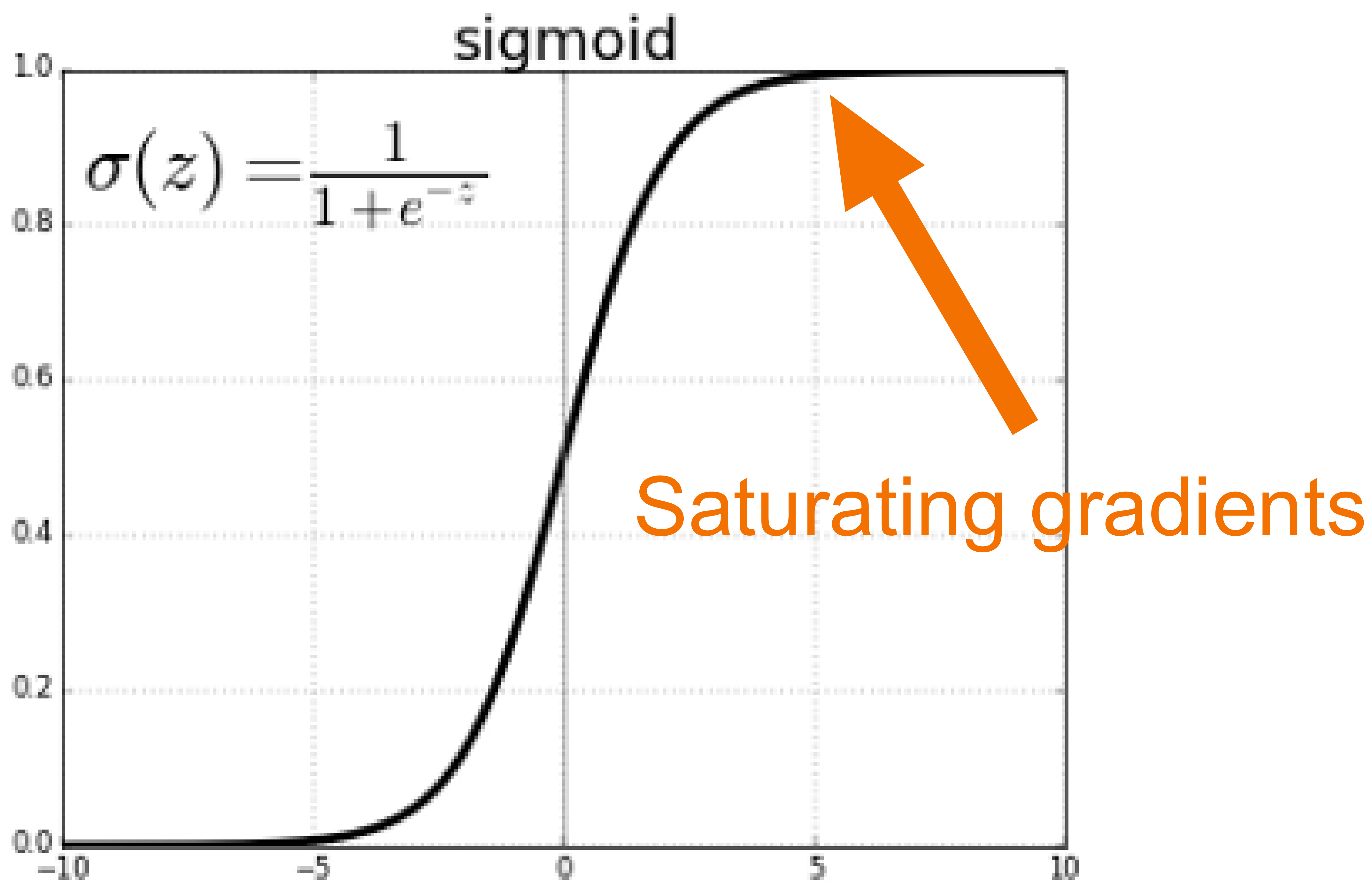
1000 classes output

Increase hidden size
from 120 to 4096



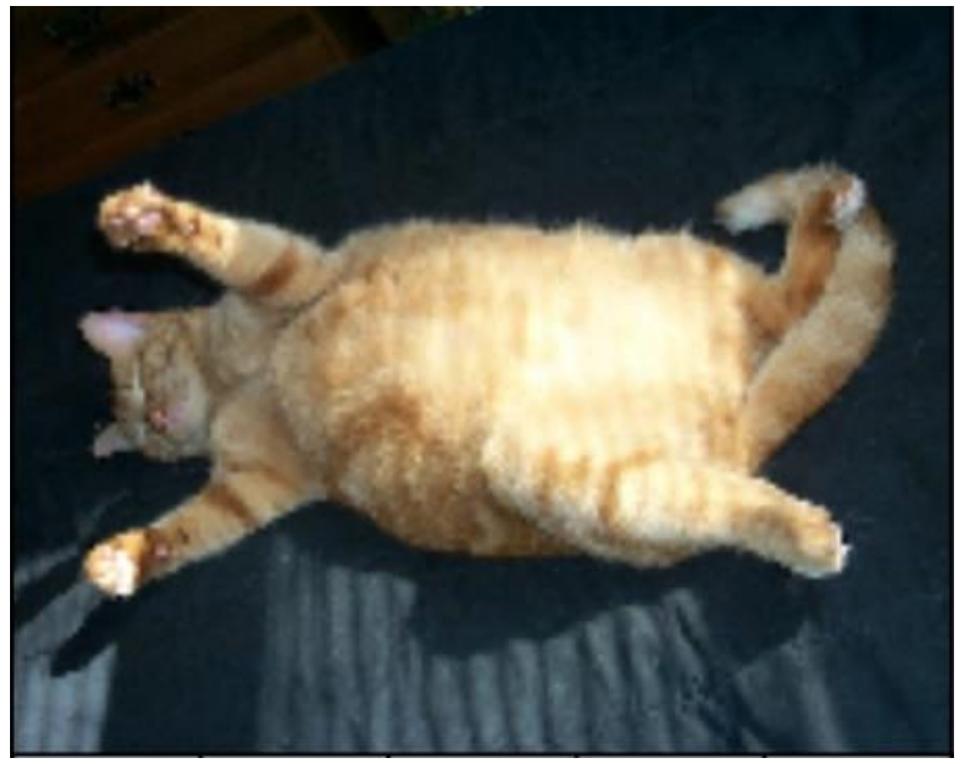
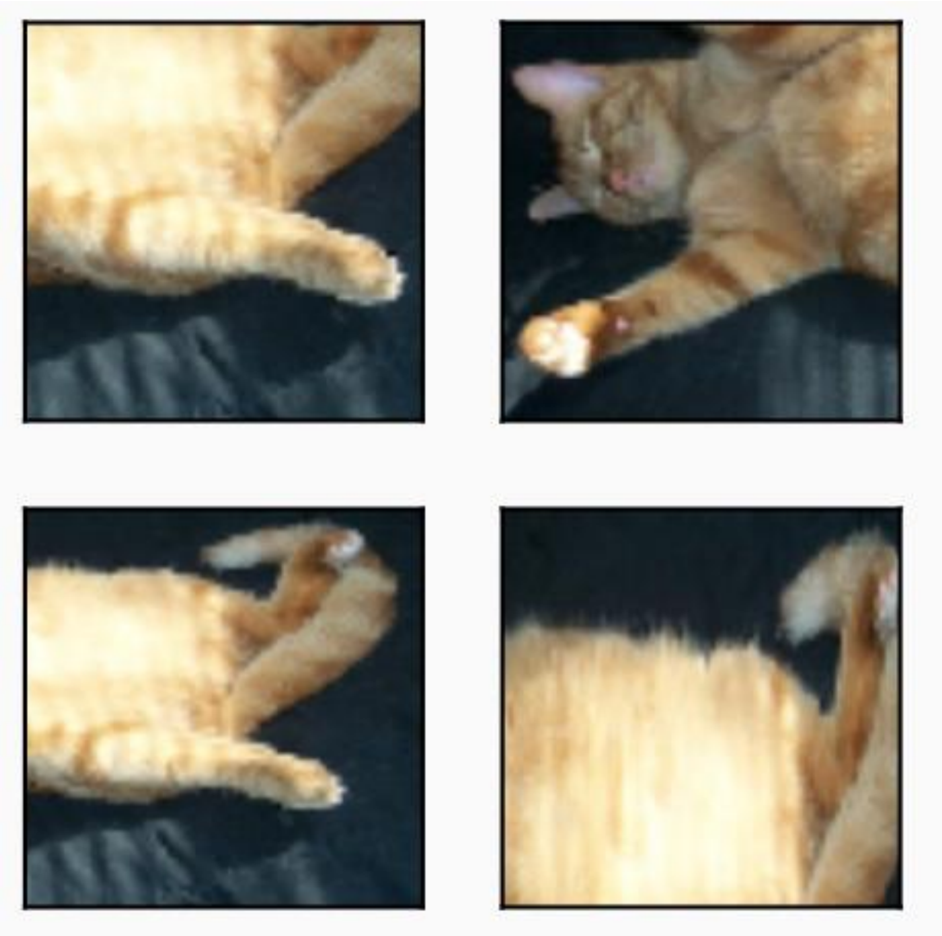
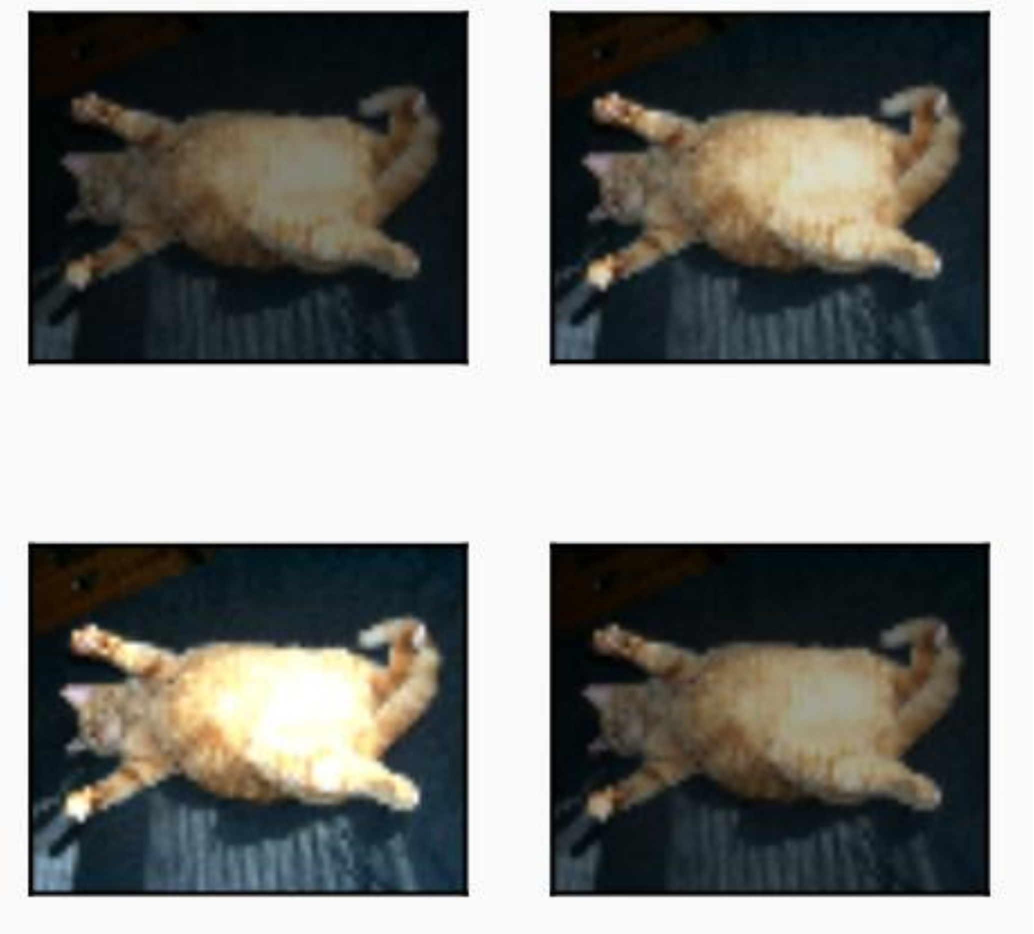
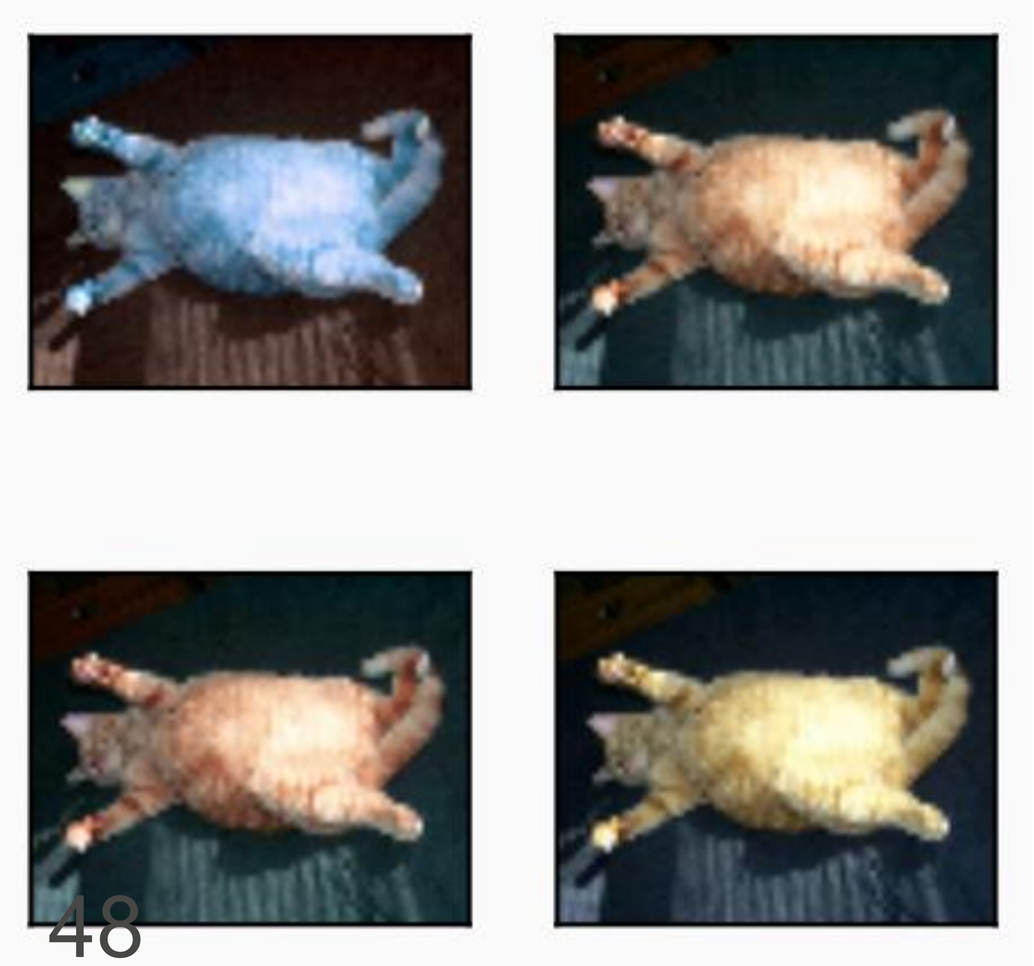
More Differences…

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



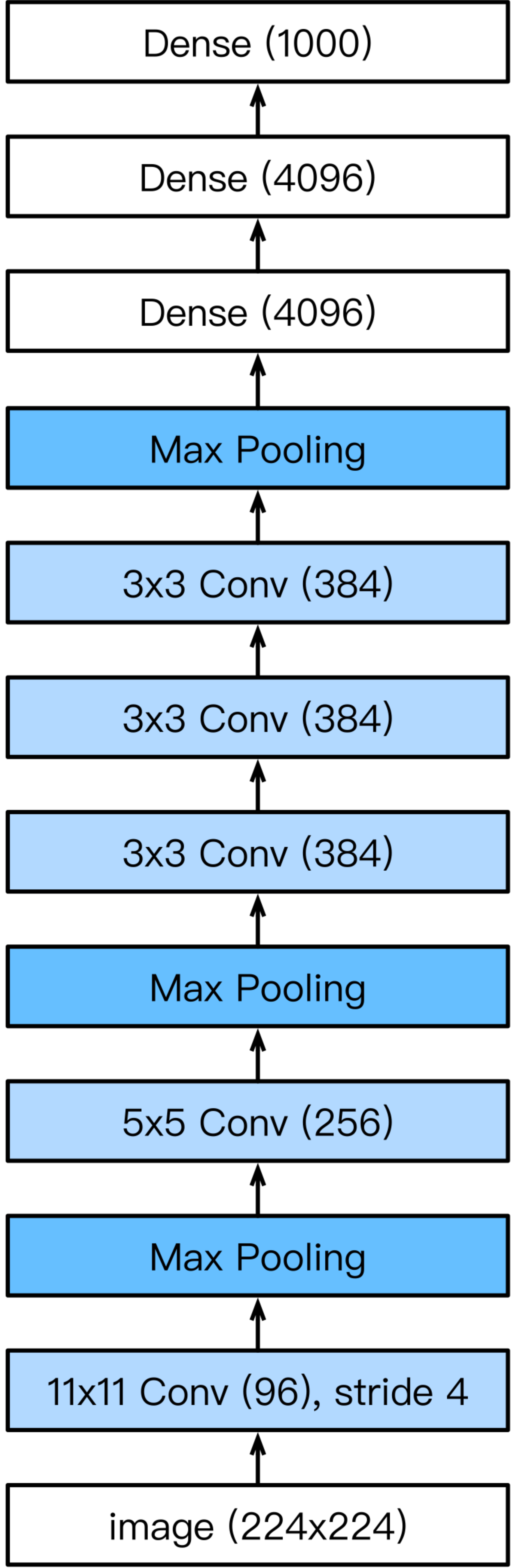
More Differences…

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

Complexity

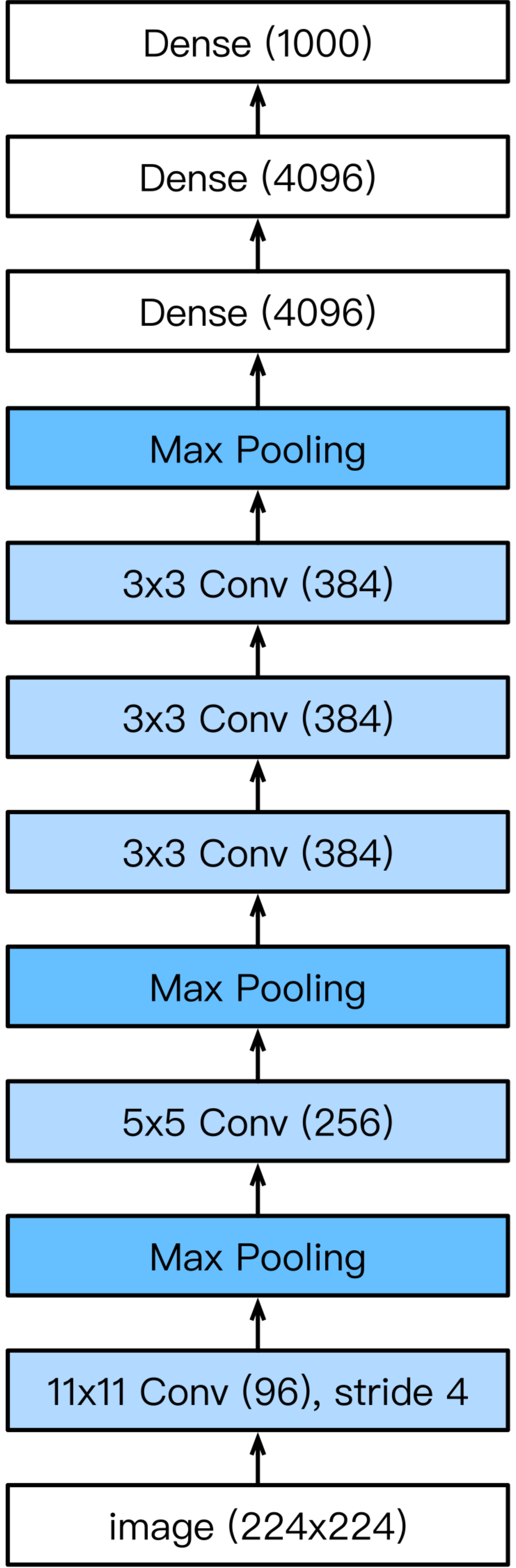
	#parameters	
	AlexNet	LeNet
Conv1	35K	150
Conv2	614K	2.4K
Conv3-5	3M	
Dense1	26M	0.048M
Dense2	16M	0.01M
Total	46M	0.06M

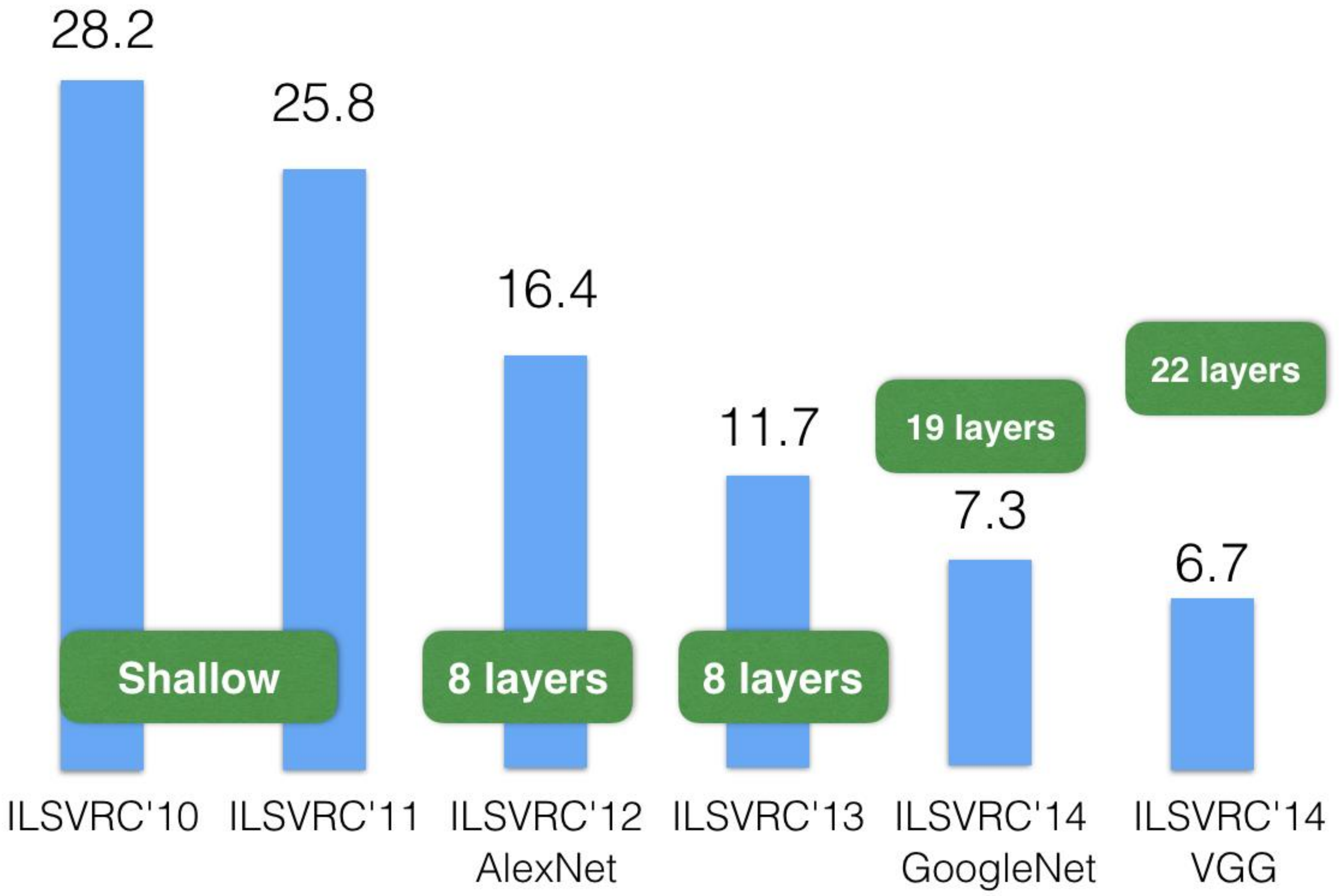


Complexity

	#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

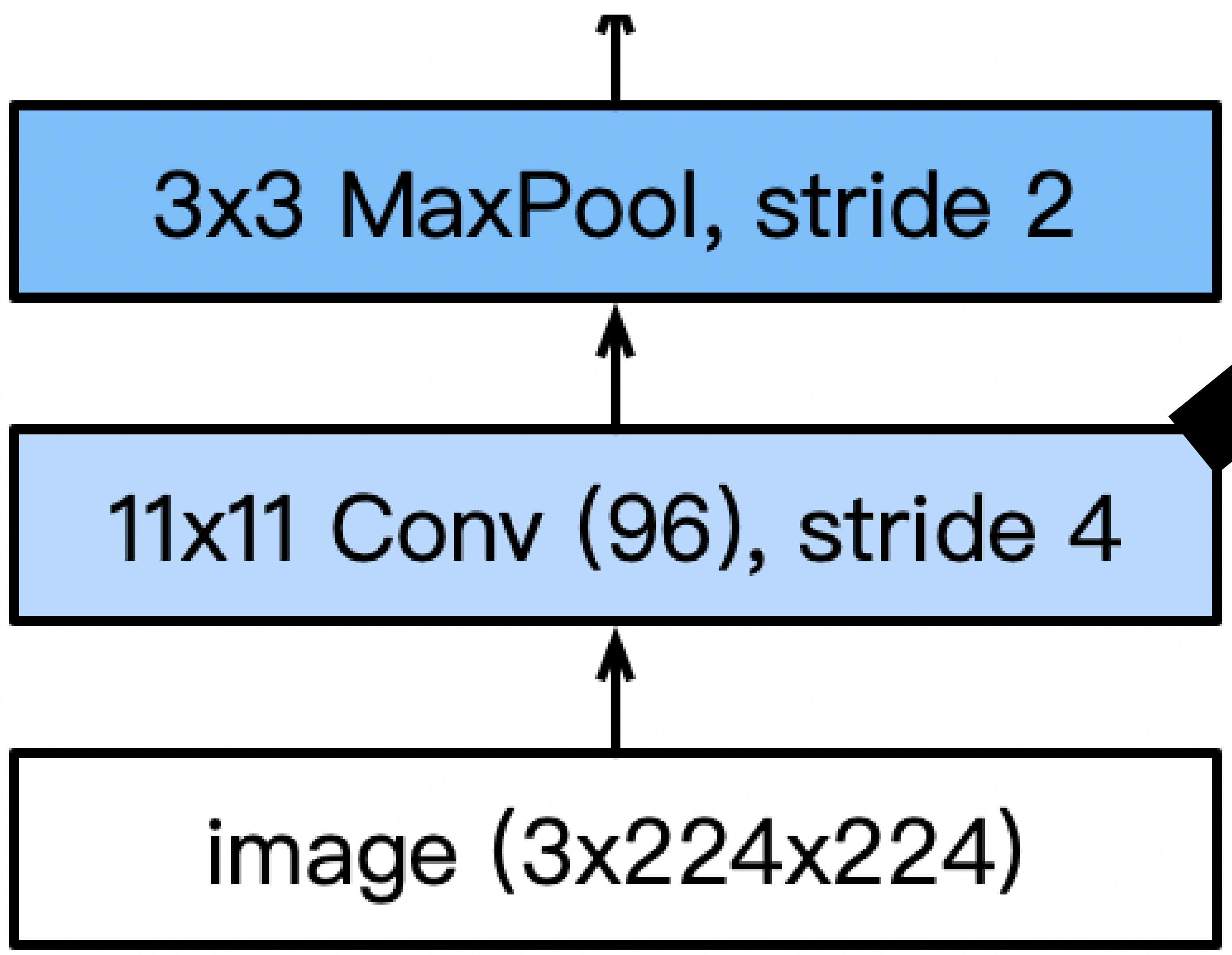
11x11x3x96=35k



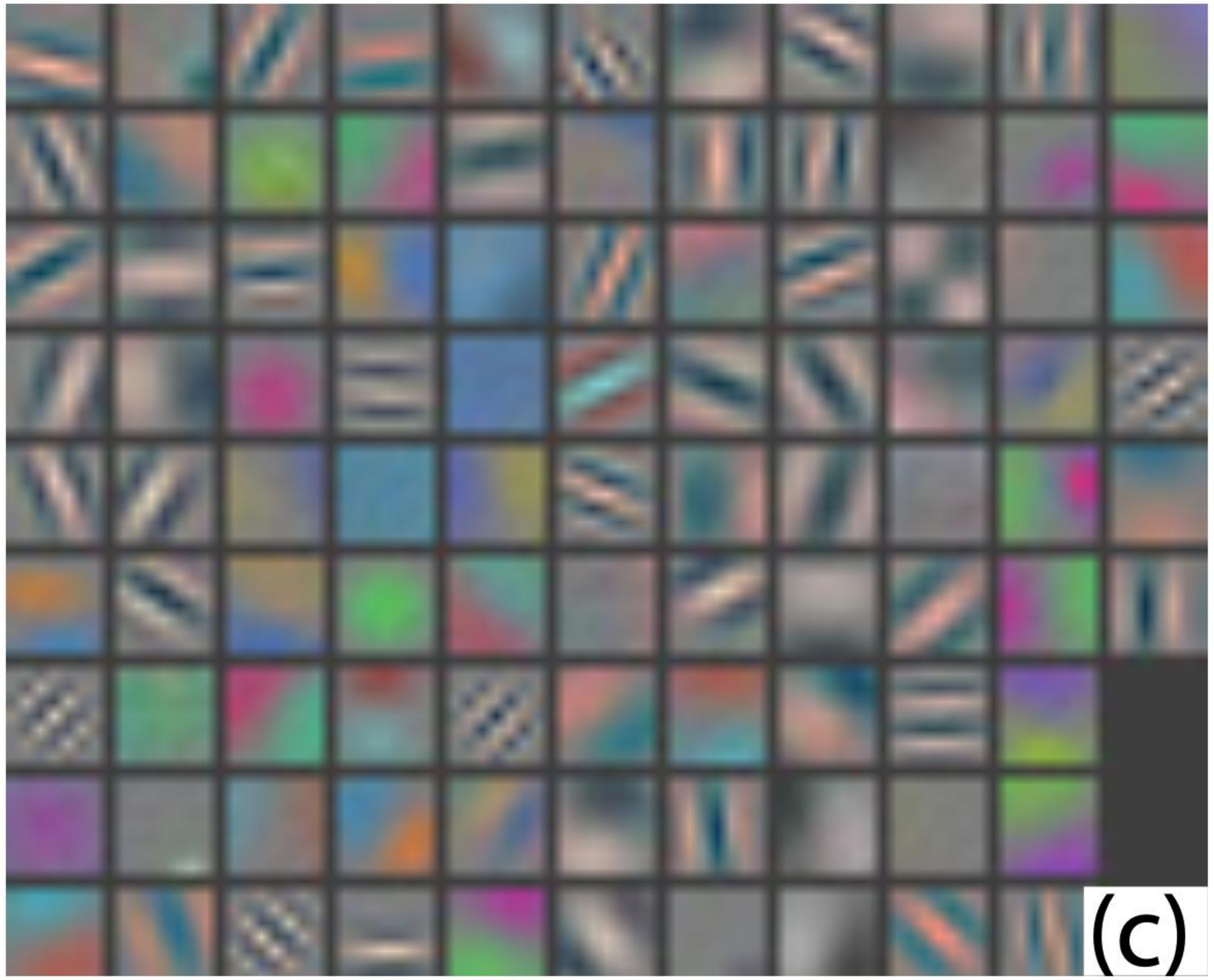


ImageNet Top-5 Classification Error (%)

AlexNet

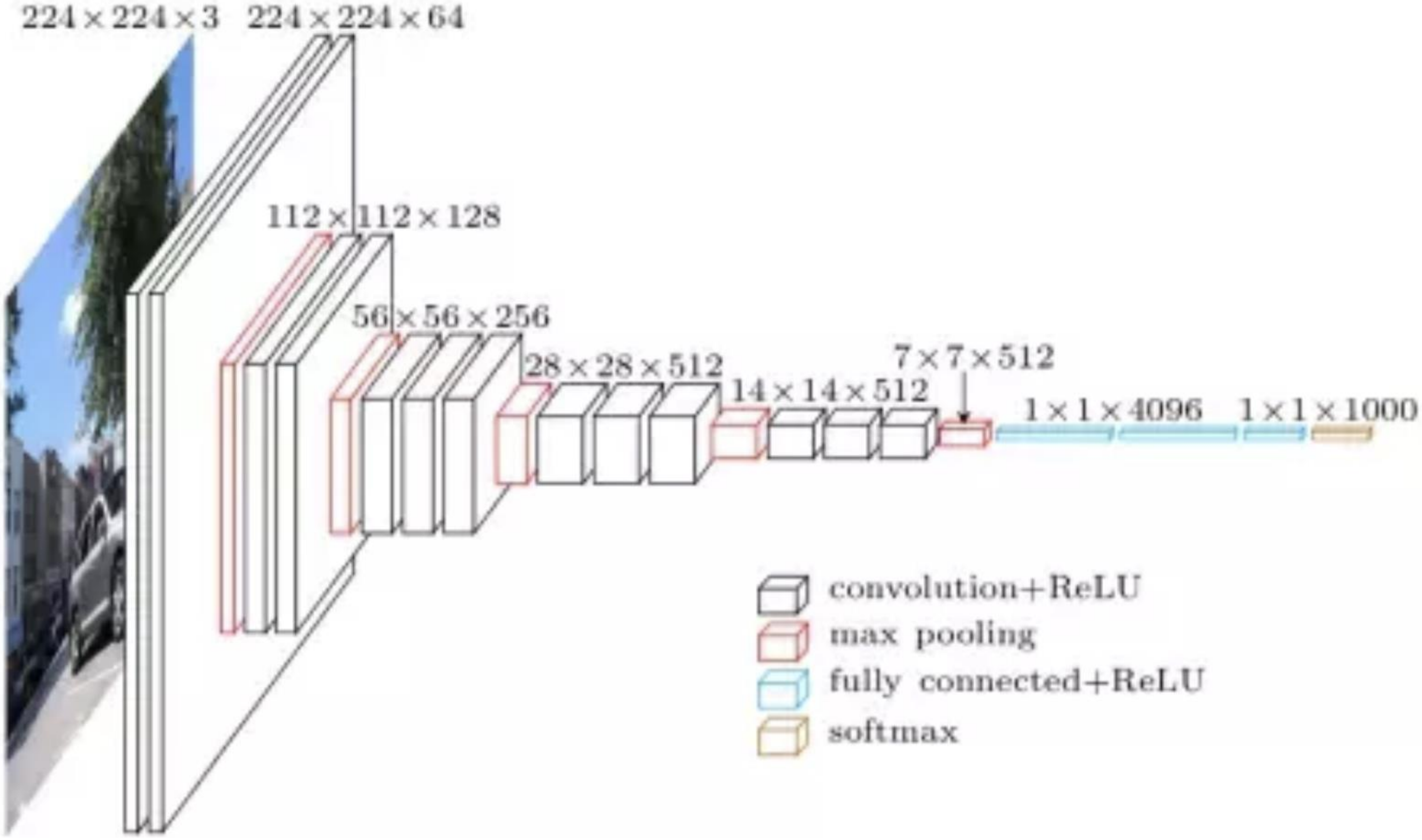


Each Conv1 kernel is 3x11x11, can be visualized as an RGB patch:



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

VGG



VGG Block: Multiple convolution layers followed by pooling.

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)

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.
- Overviewed the evolution of deeper convolutional networks

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