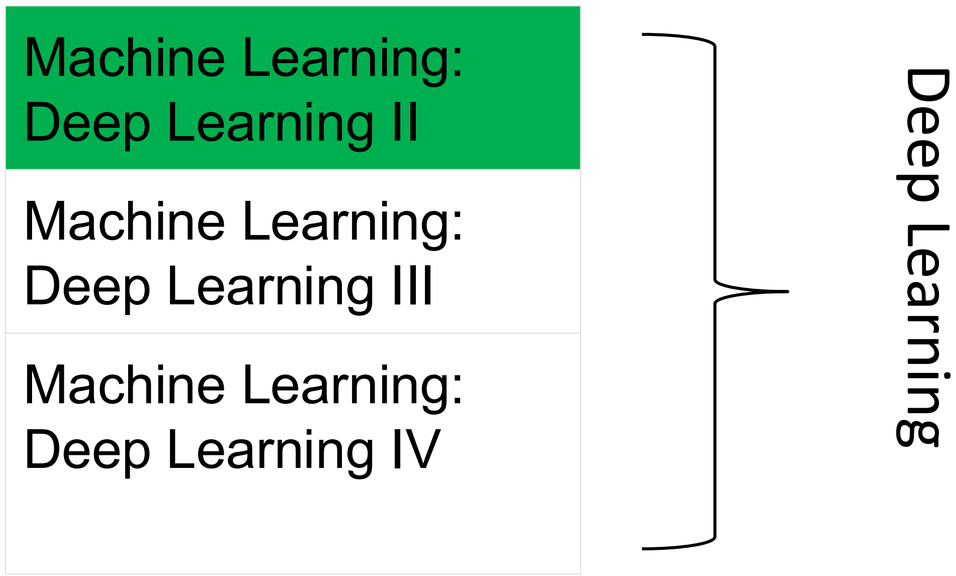


CS540 Introduction to Artificial Intelligence
Convolutional Neural Networks (II)

University of Wisconsin-Madison
Fall 2026 Sections 1 & 2

Announcements

Class roadmap and schedule:



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

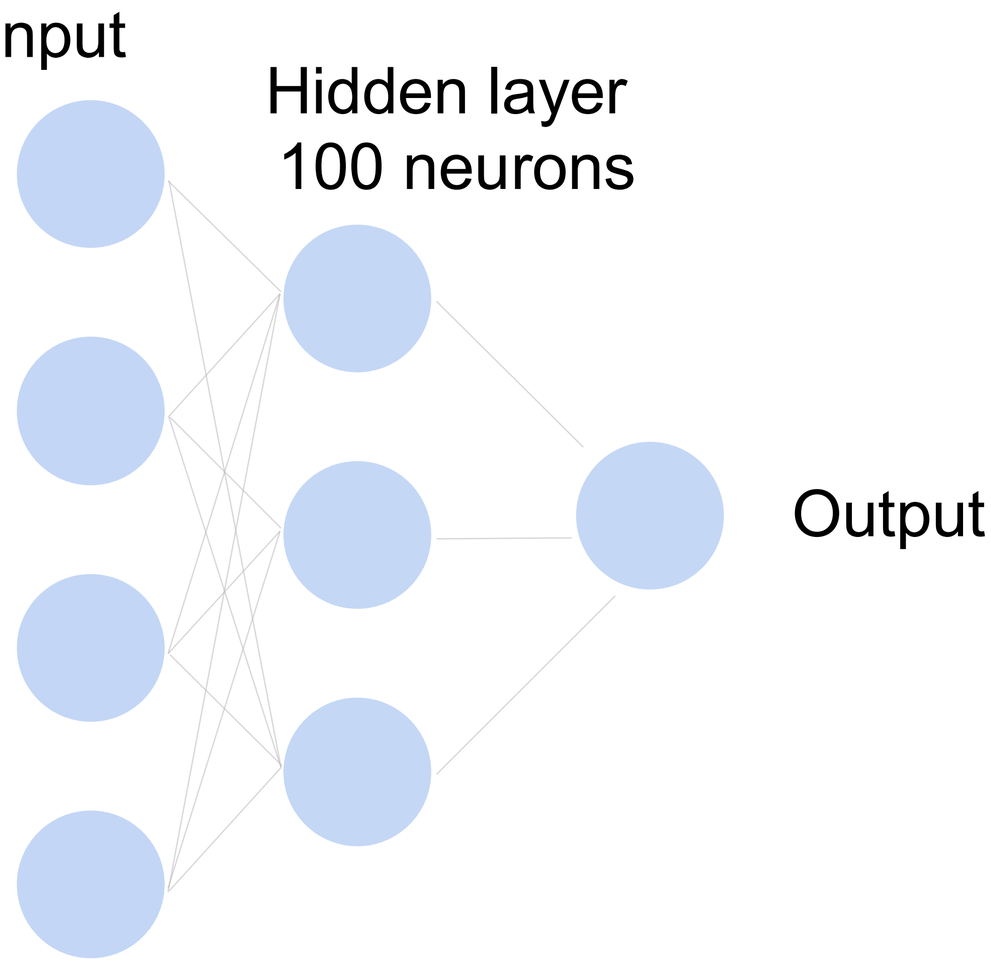
36M floats in a RGB image!

Fully Connected Networks

Cats vs. dogs?







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

Review: 2-D Convolution

Input

0	1	2
3	4	5
6	7	8

*

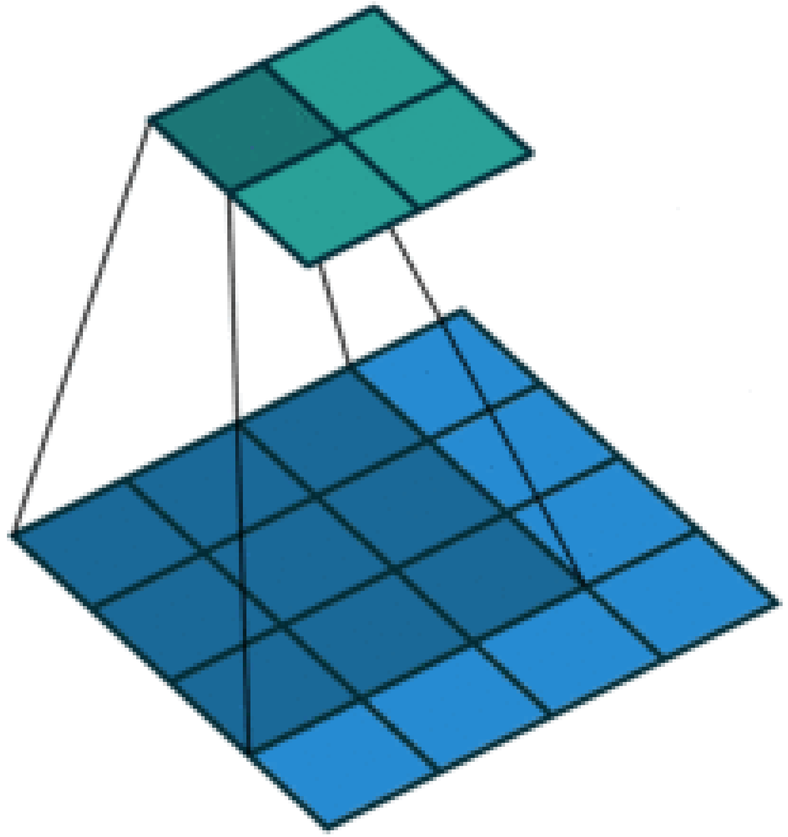
Kernel

0	1
2	3

=

Output

19	25
37	43



(vdumoulin@ Github)

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

Review: Output shape of a 2D convolution

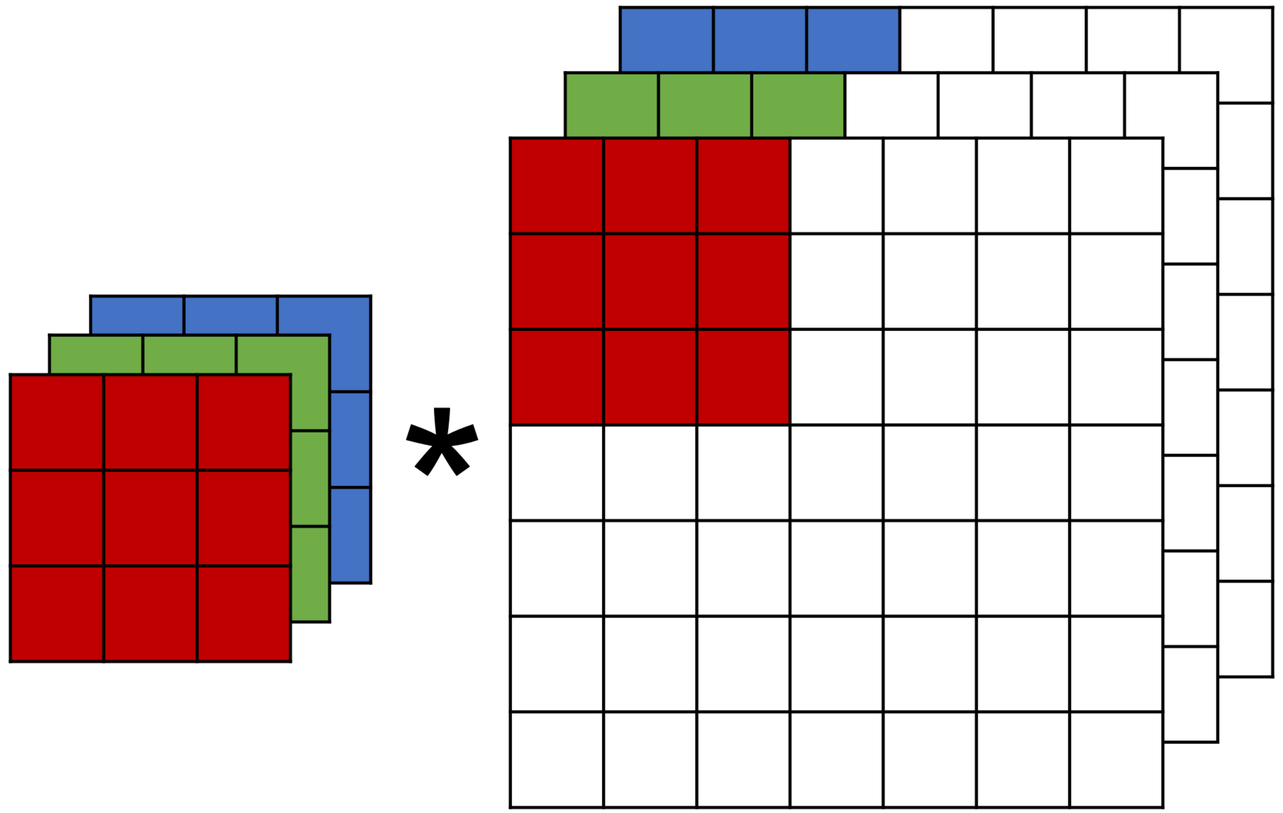
Kernel/filter size

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

Input size Pad Stride

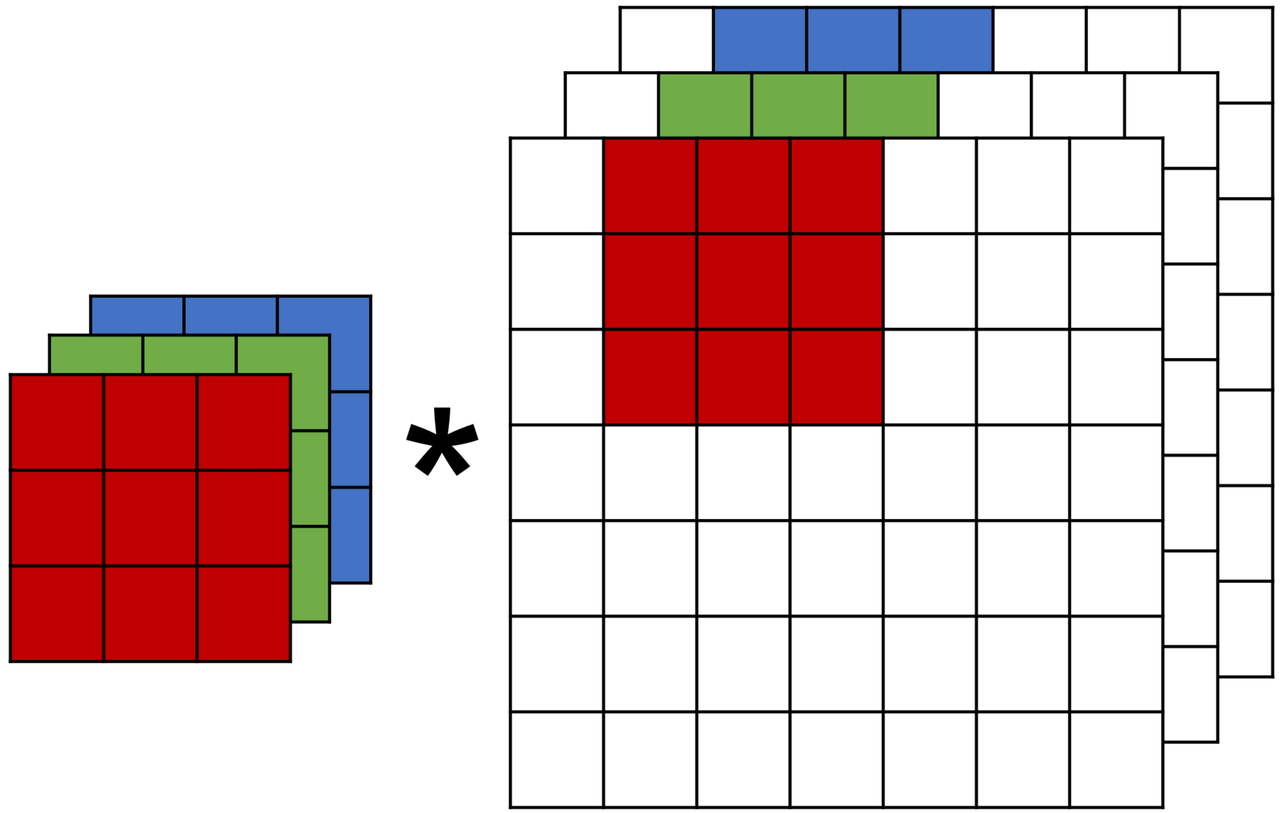
Review: Multiple Input Channels

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



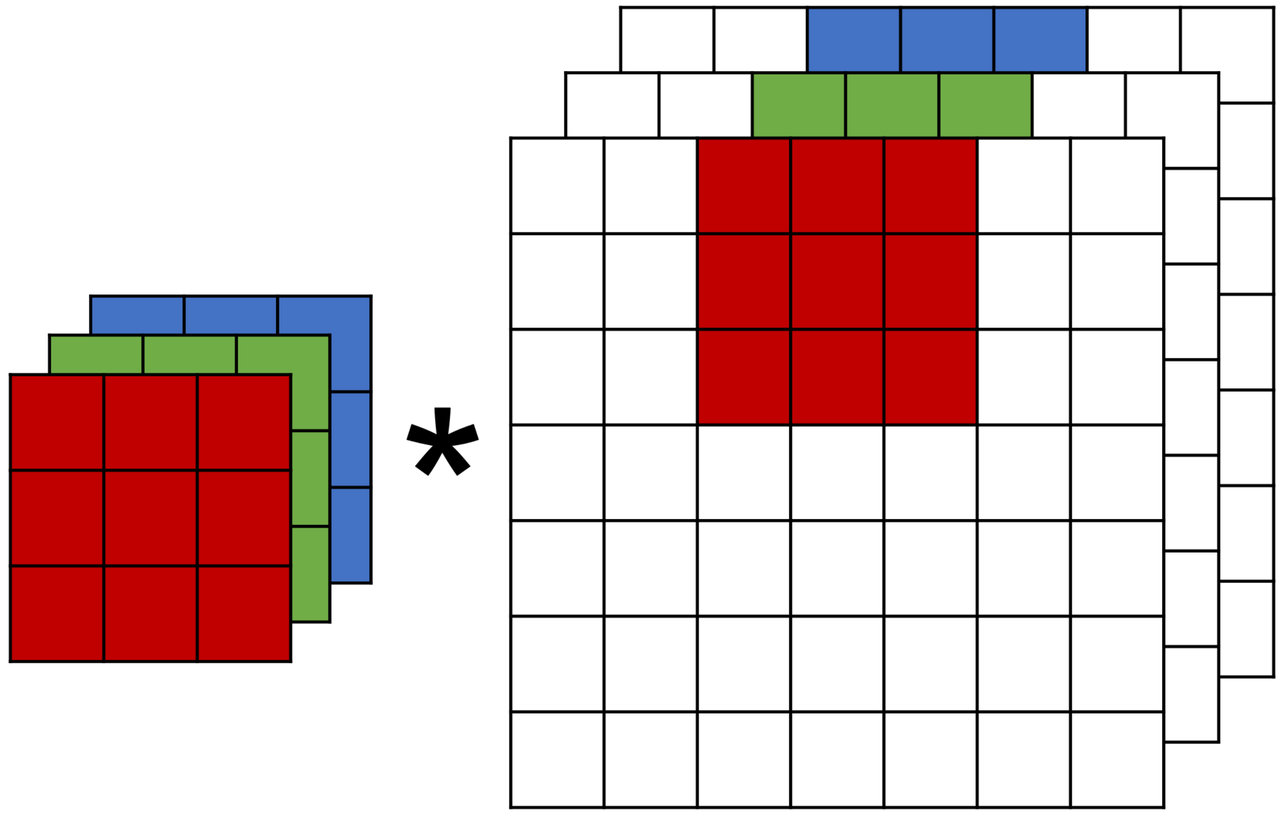
Review: Multiple Input Channels

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



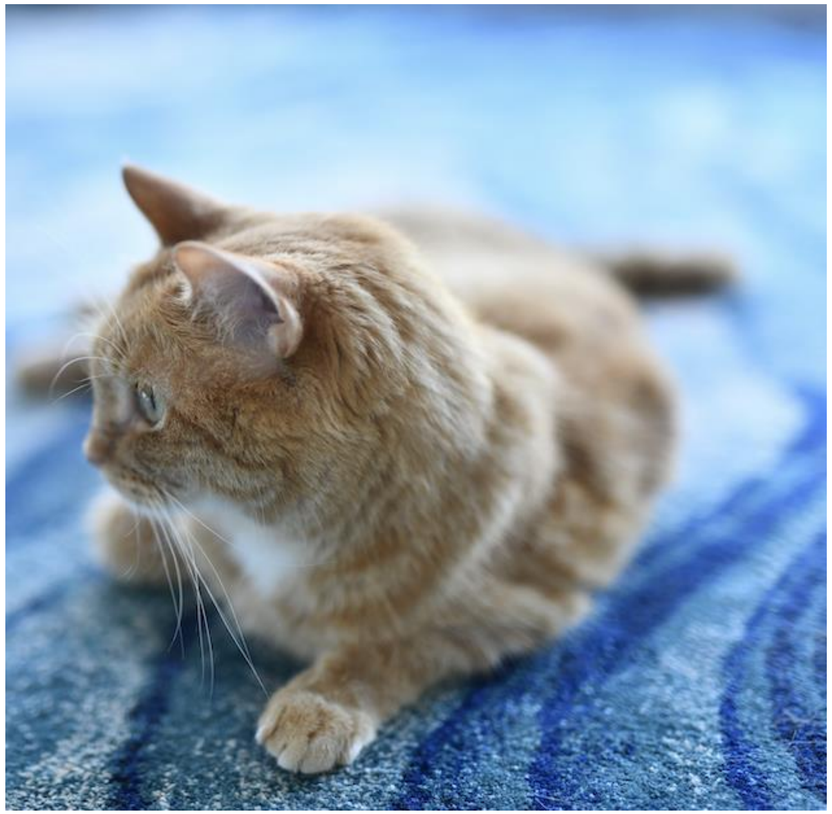
Review: Multiple Input Channels

- Also call each 3D kernel a "**filter**", which produces only **one** output channel (due to summation over channels)



one filter
(3 channels)

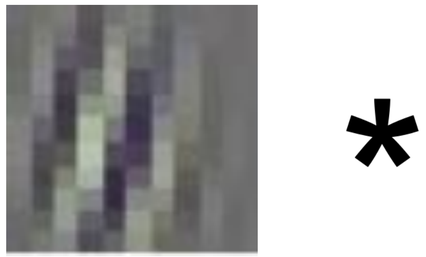
*



RGB (3 input channels)

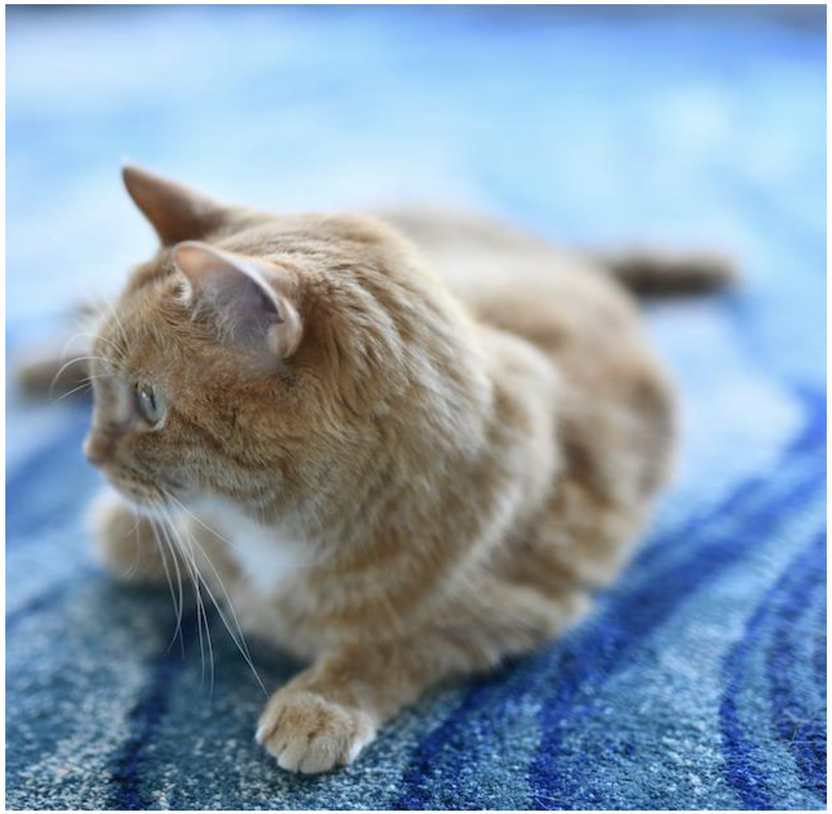
Multiple filters (in one layer)

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

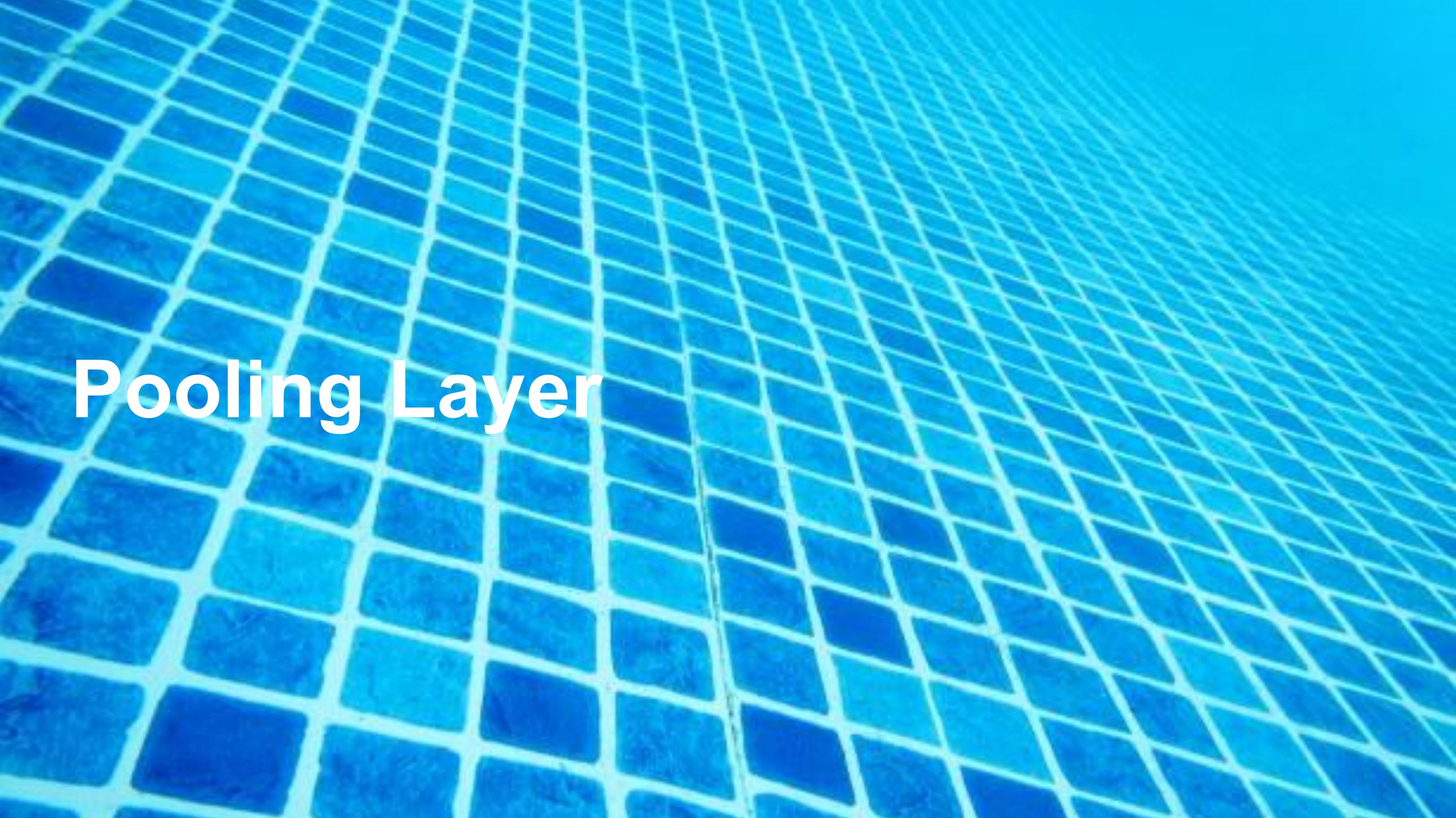


a different filter

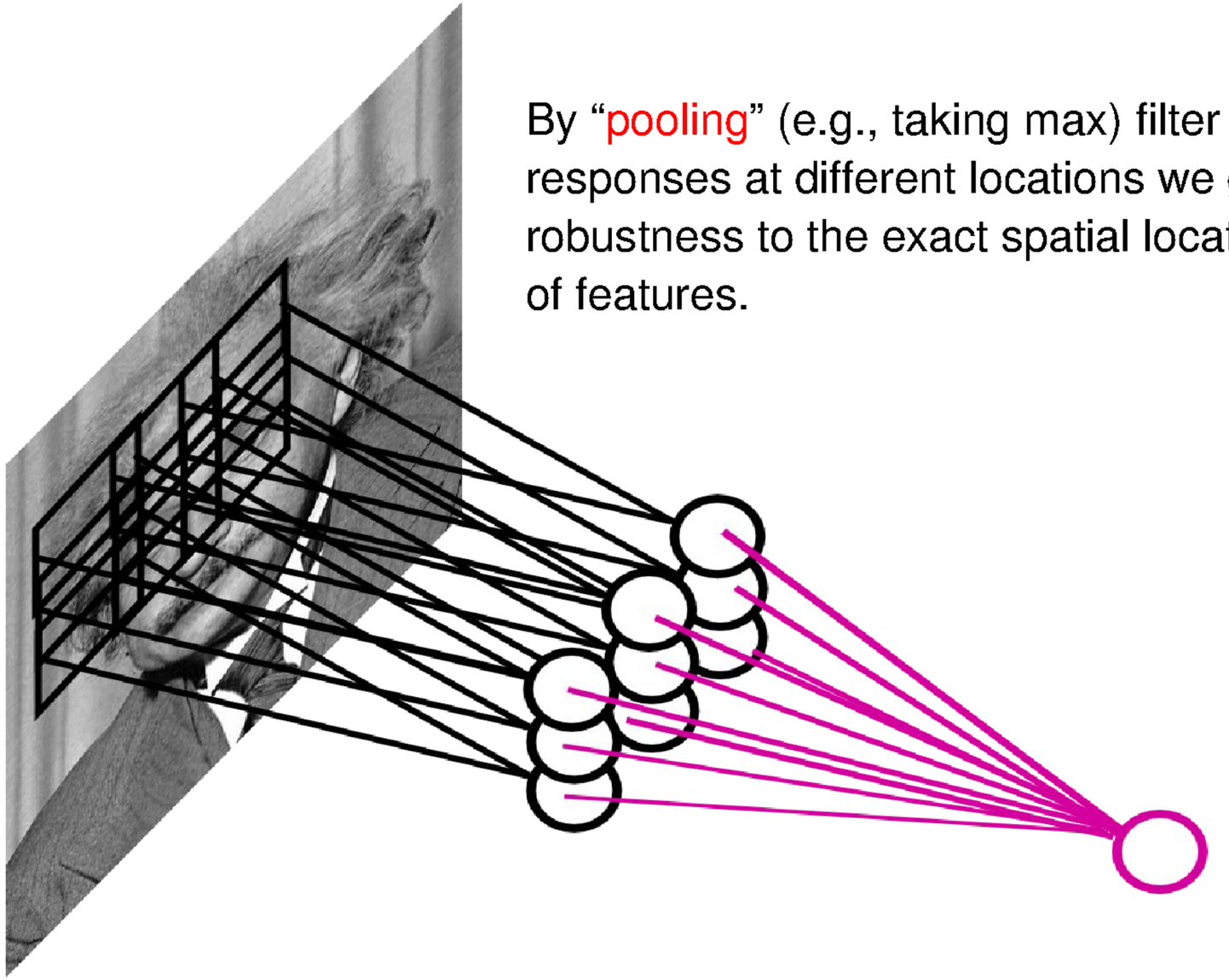
*



RGB (3 input channels)


Pooling Layer

Review: Pooling

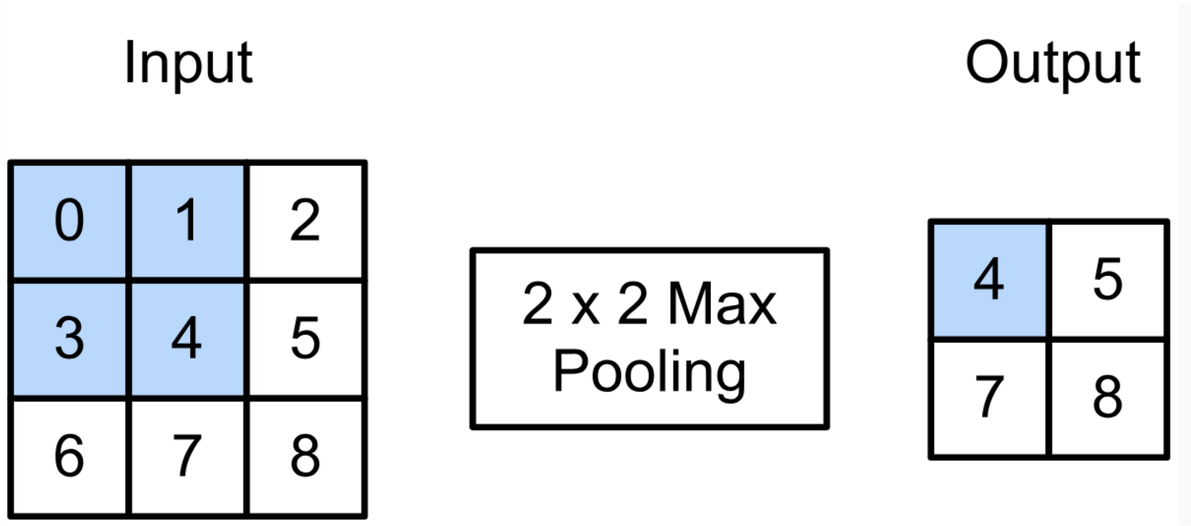


By “pooling” (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.

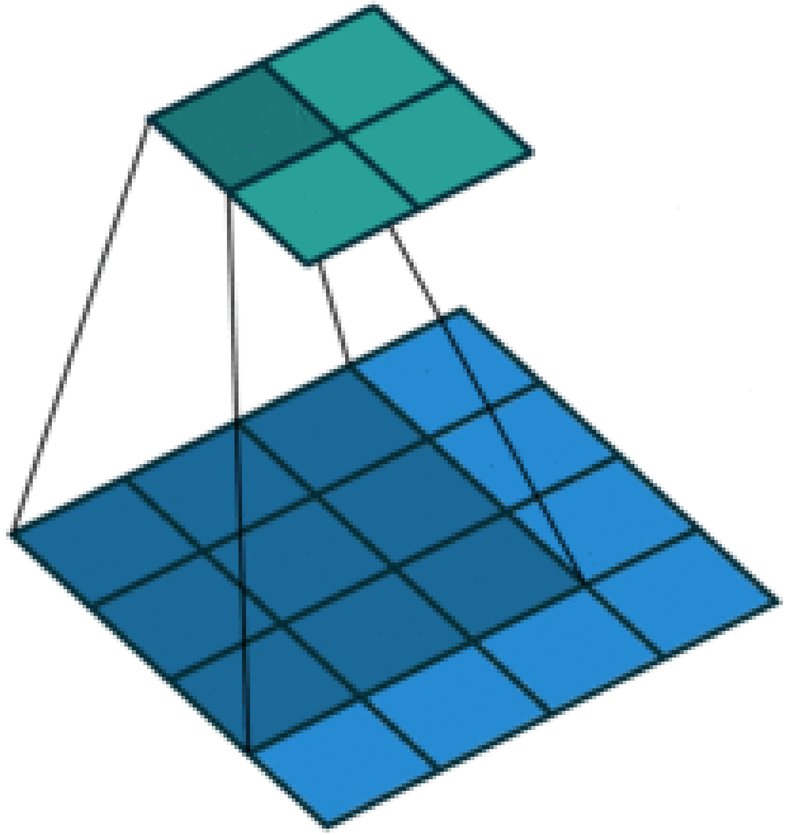
Slides Credit: Deep Learning Tutorial by Marc’Aurelio Ranzato

Review: 2-D Max Pooling

- Returns the maximal value in the sliding window



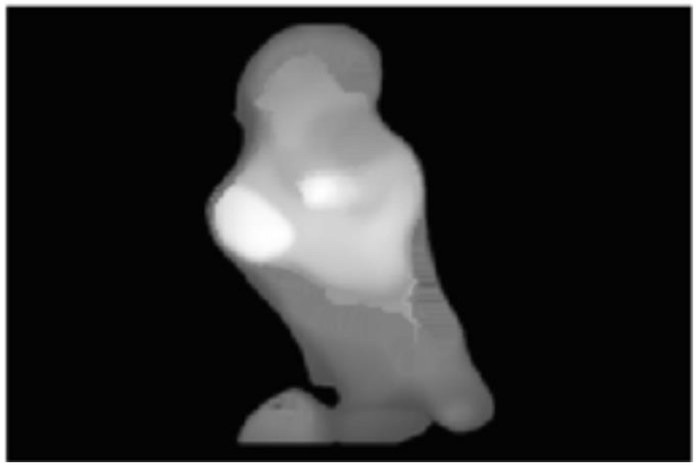
$$max(0,1,3,4) = 4$$



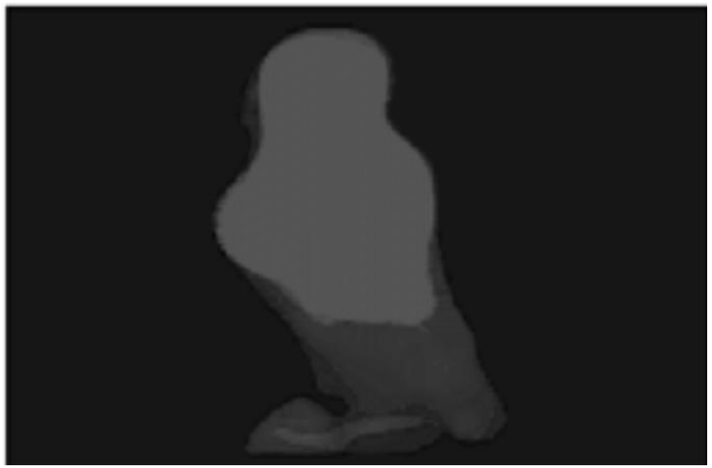
Review: Average Pooling

- 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

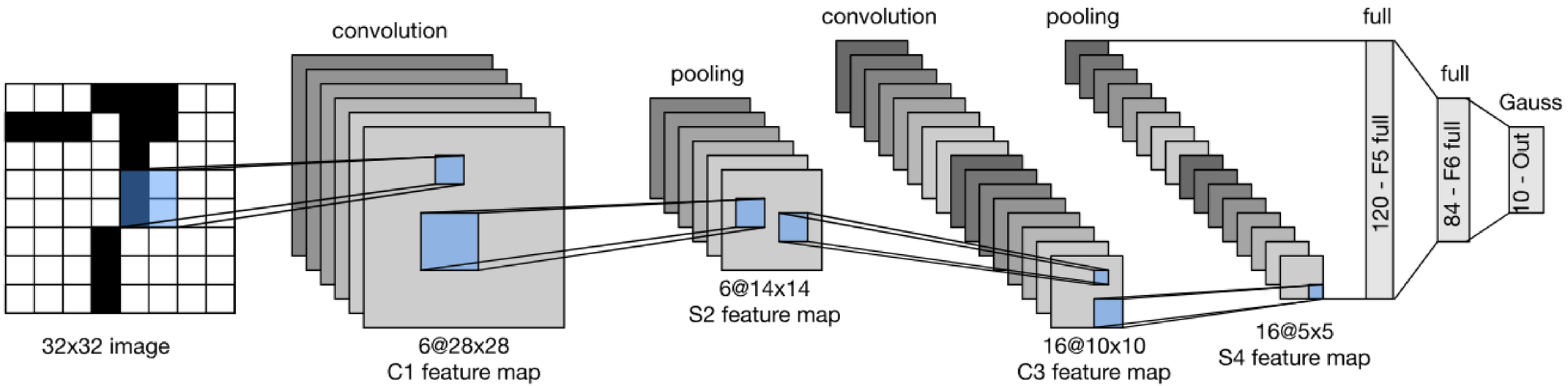
Max pooling



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.

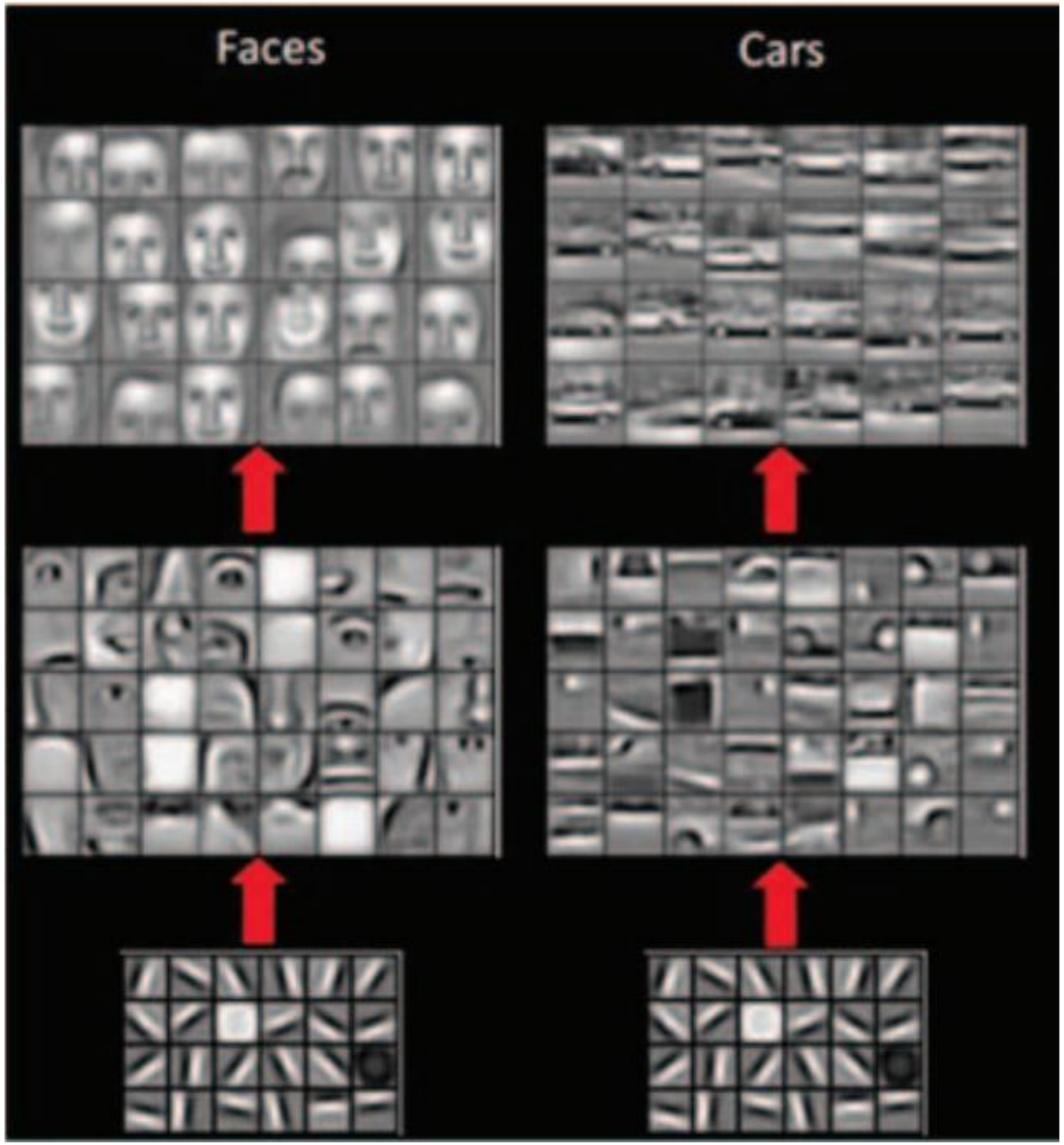
- Example: Dogs have longer snouts.
- How do you determine what is a long snout?

Feature Learning

Later layers recognize complete objects

Middle layers recognize parts of objects

Early layers recognize simple patterns

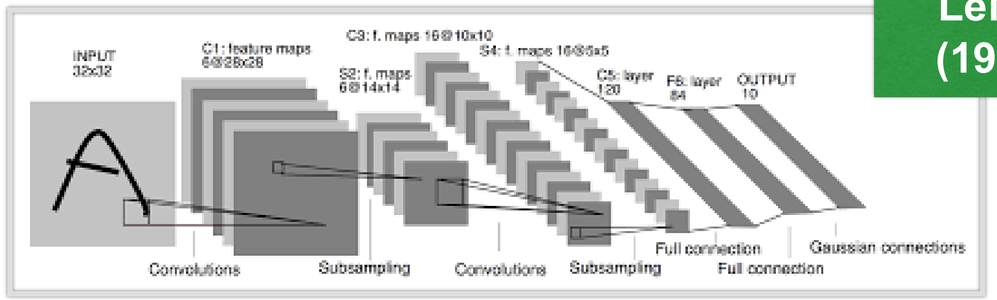


Adaptive Neuron Apoptosis for Accelerating Deep Learning on Large Scale Systems. Seigel et al. 2016.

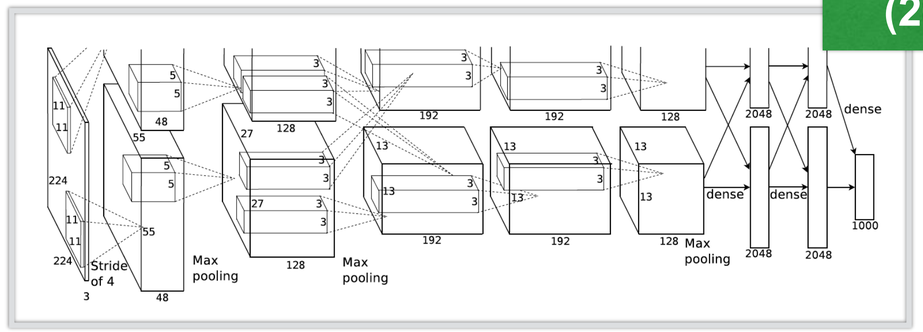
Convolutional Neural Networks Examples

Evolution of neural net architectures

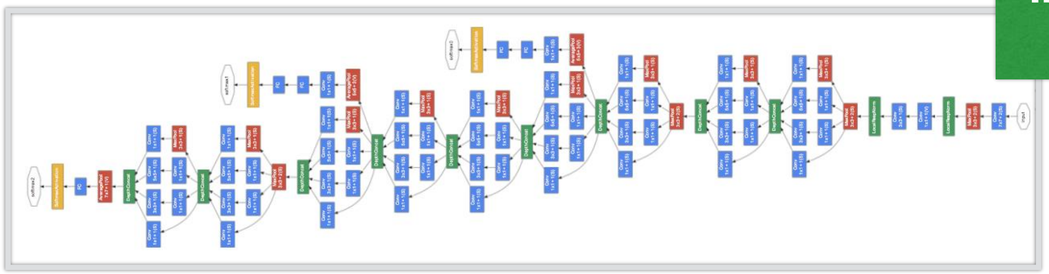
LeNet (1989)



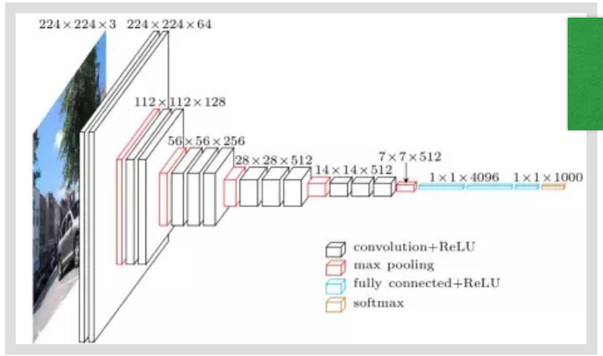
AlexNet (2012)



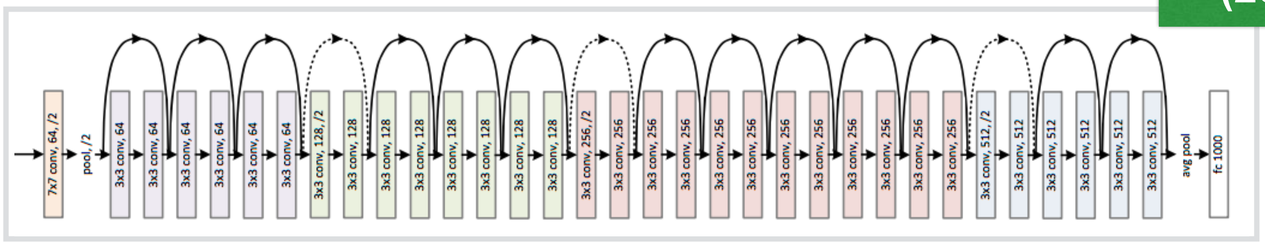
Inception Net (2014)



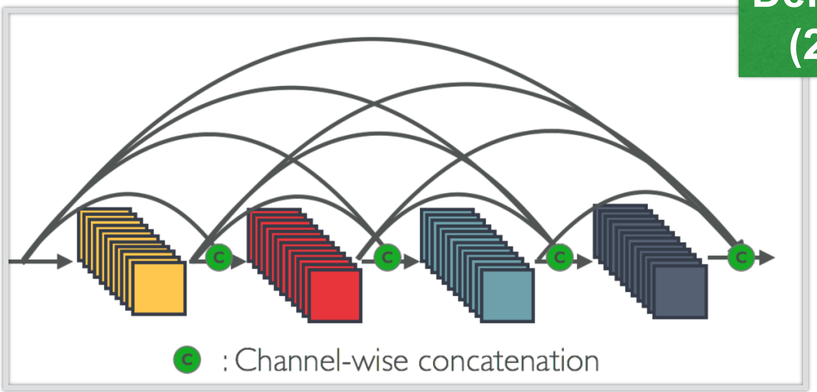
VGG (2014)



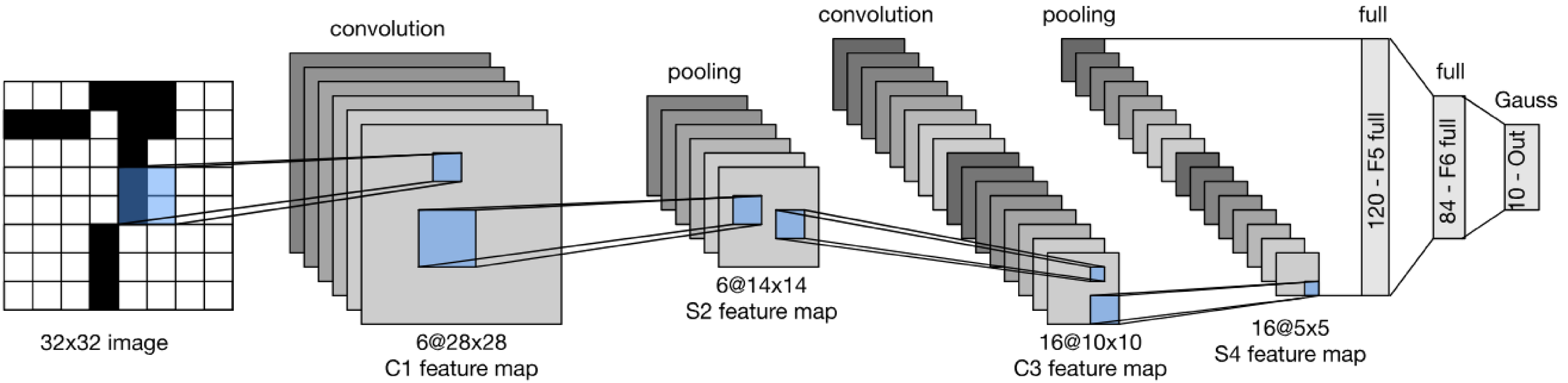
ResNet (2015)



DenseNet (2016)

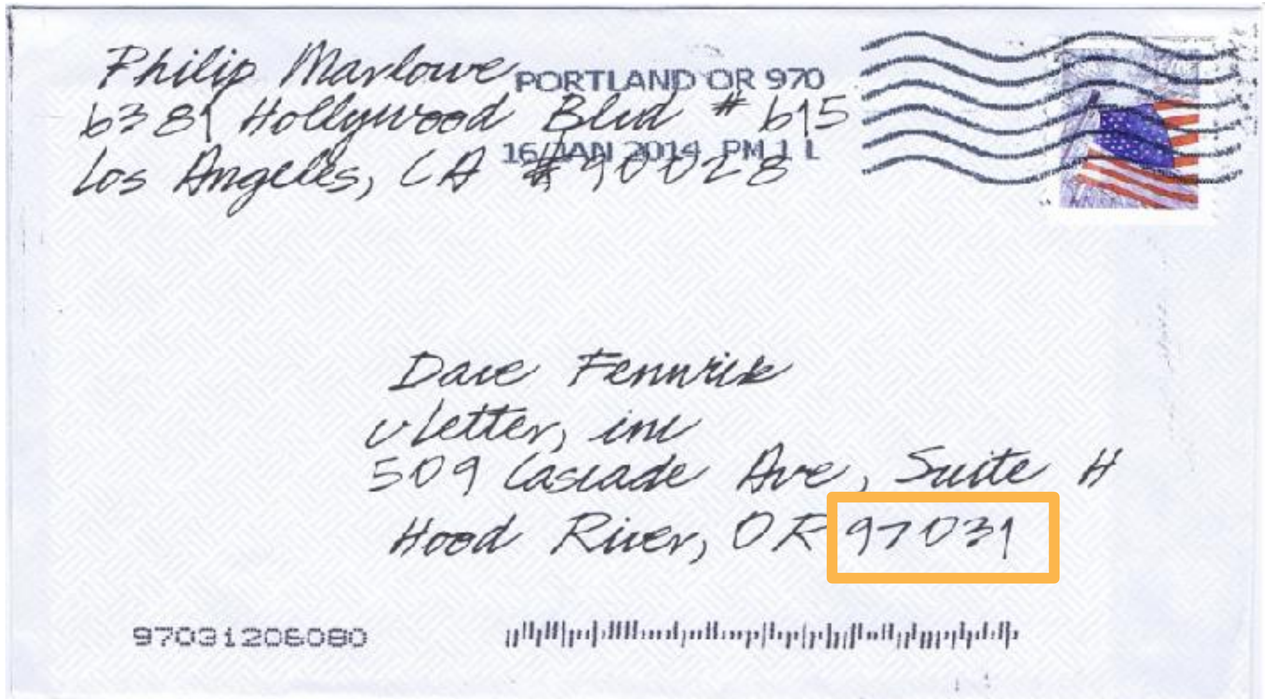


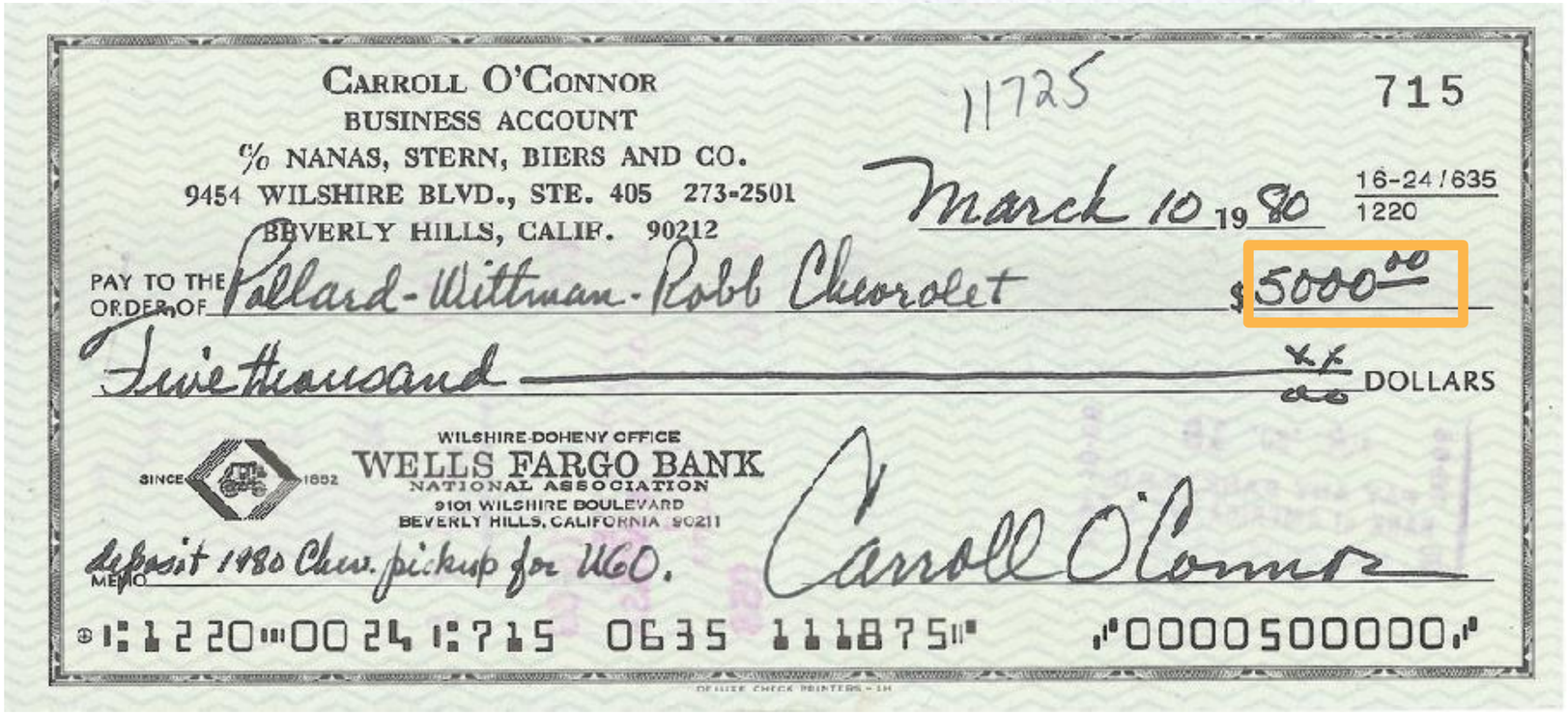
LeNet Architecture
(first convolutional neural net; 1989)



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

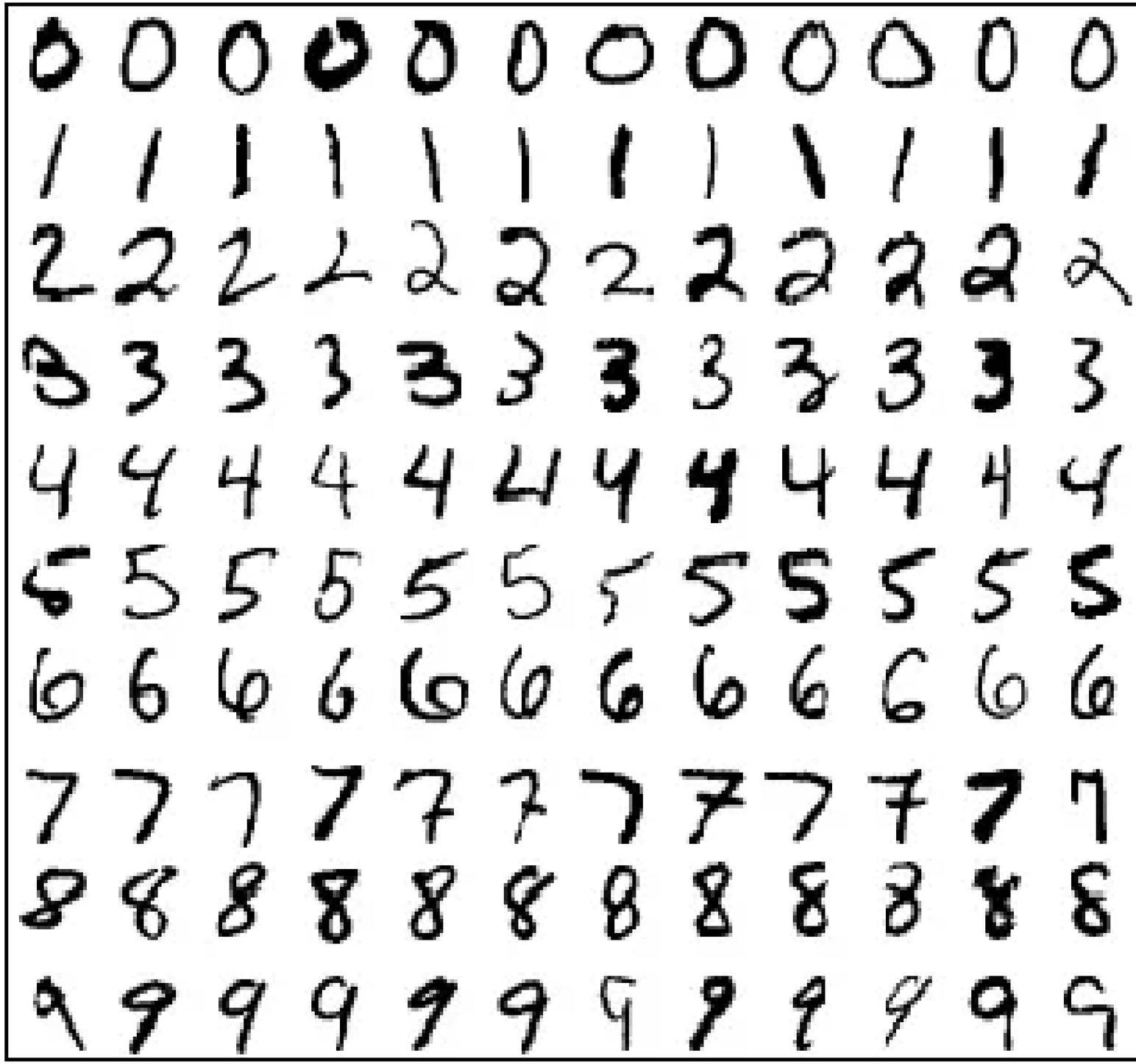
Handwritten Digit Recognition

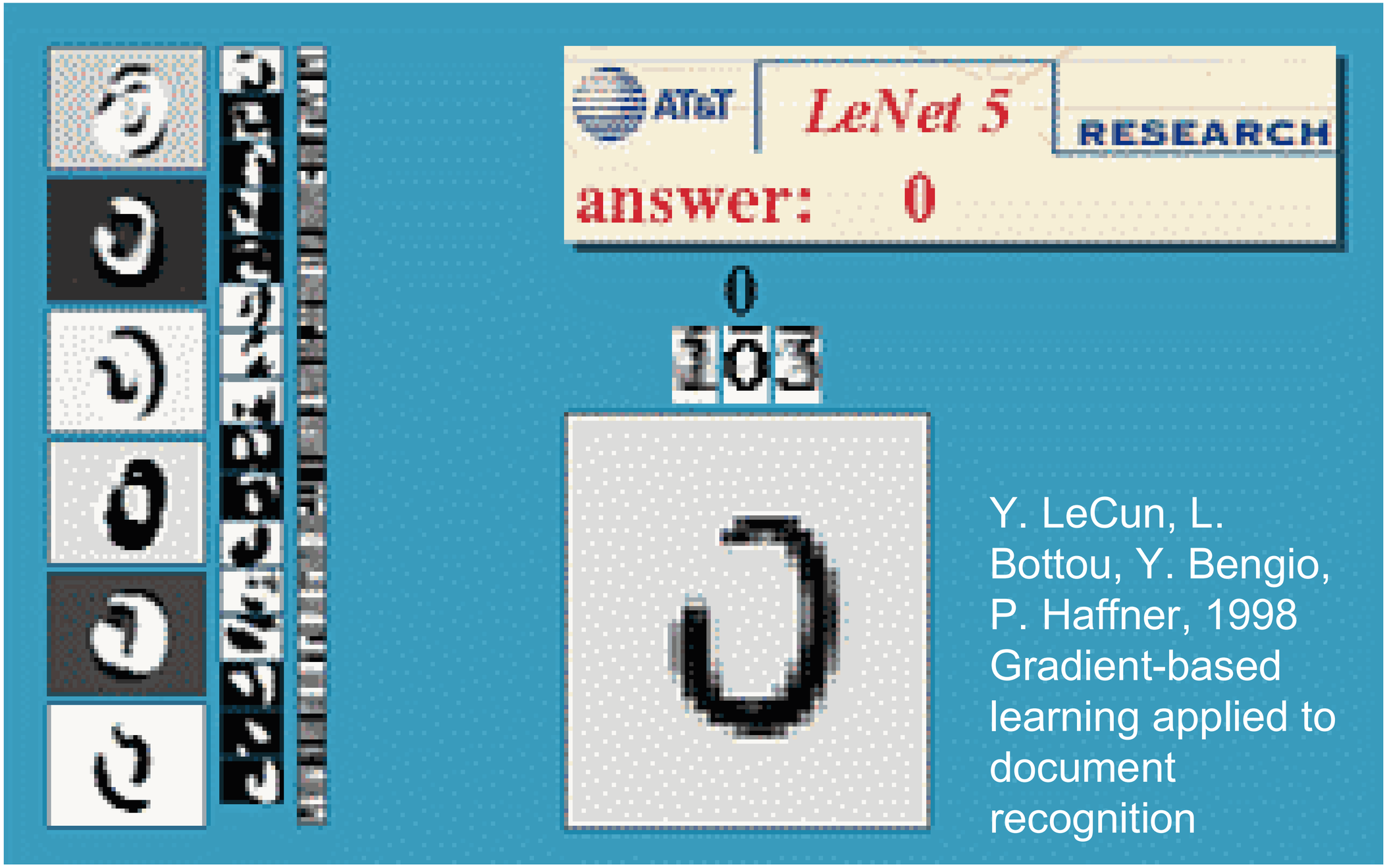




MNIST

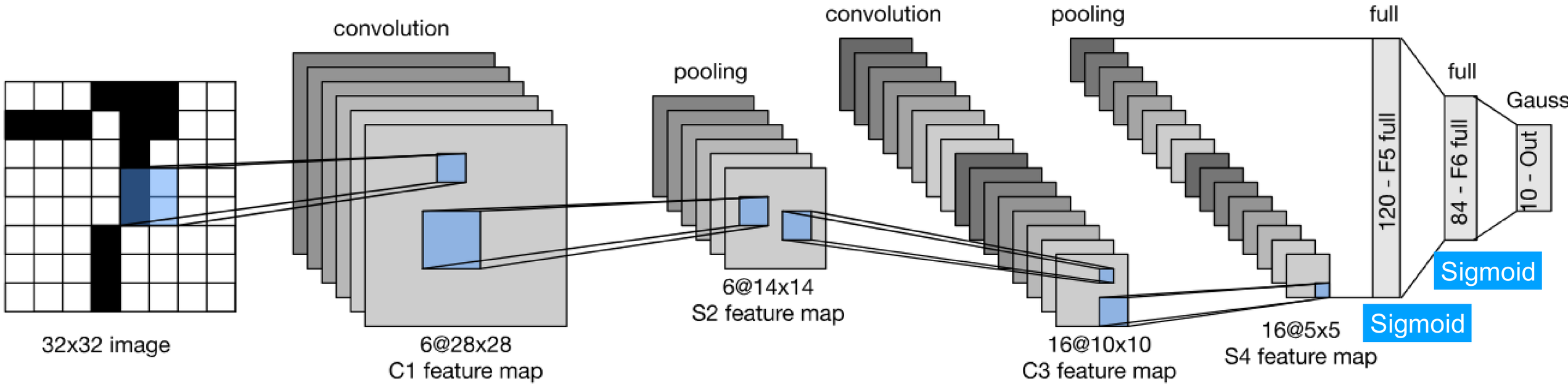
- Centered and scaled
- 50,000 training data
- 10,000 test data
- 28 x 28 images
- 10 classes





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)
```

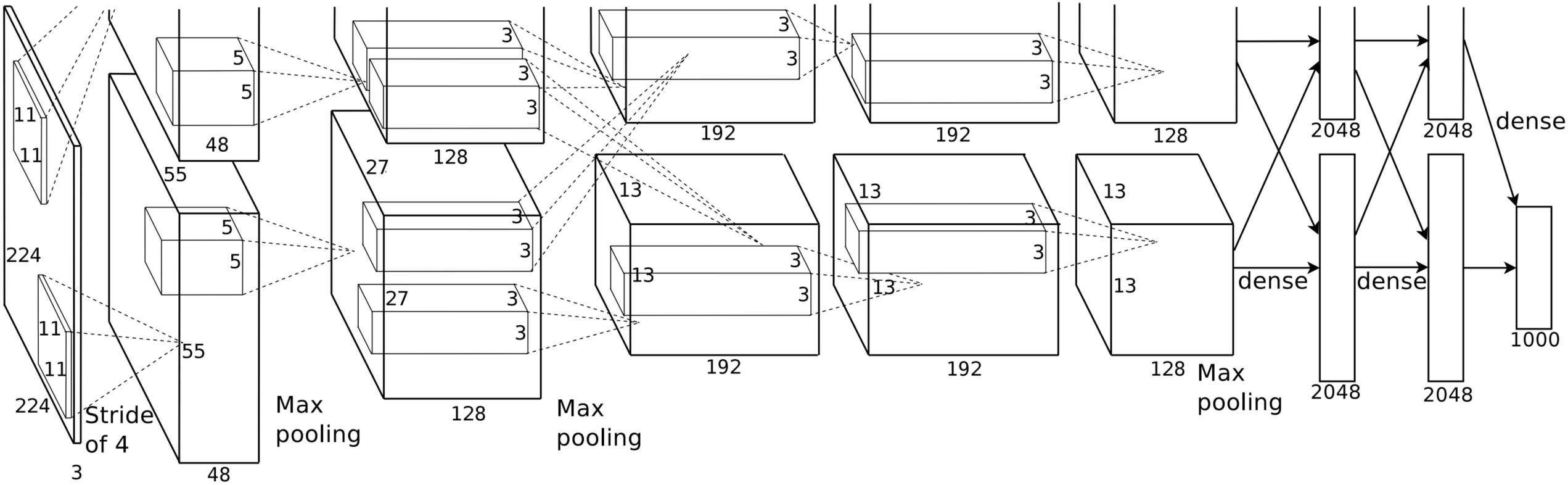
https://github.com/bollakarthikeya/LeNet-5-PyTorch/blob/master/lenet5_gpu.py

```
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)

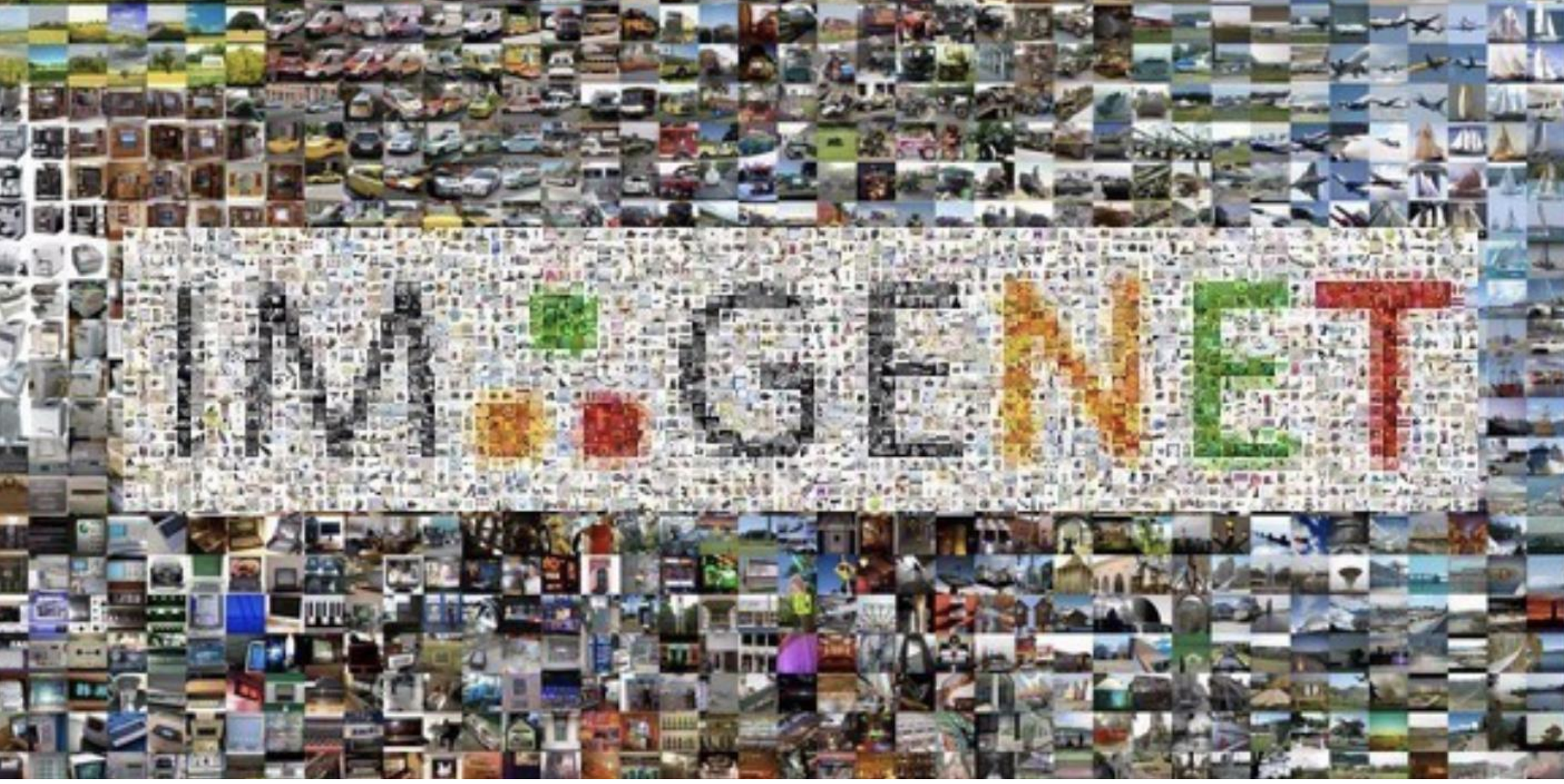
    return x
```

LeNet in Pytorch

AlexNet



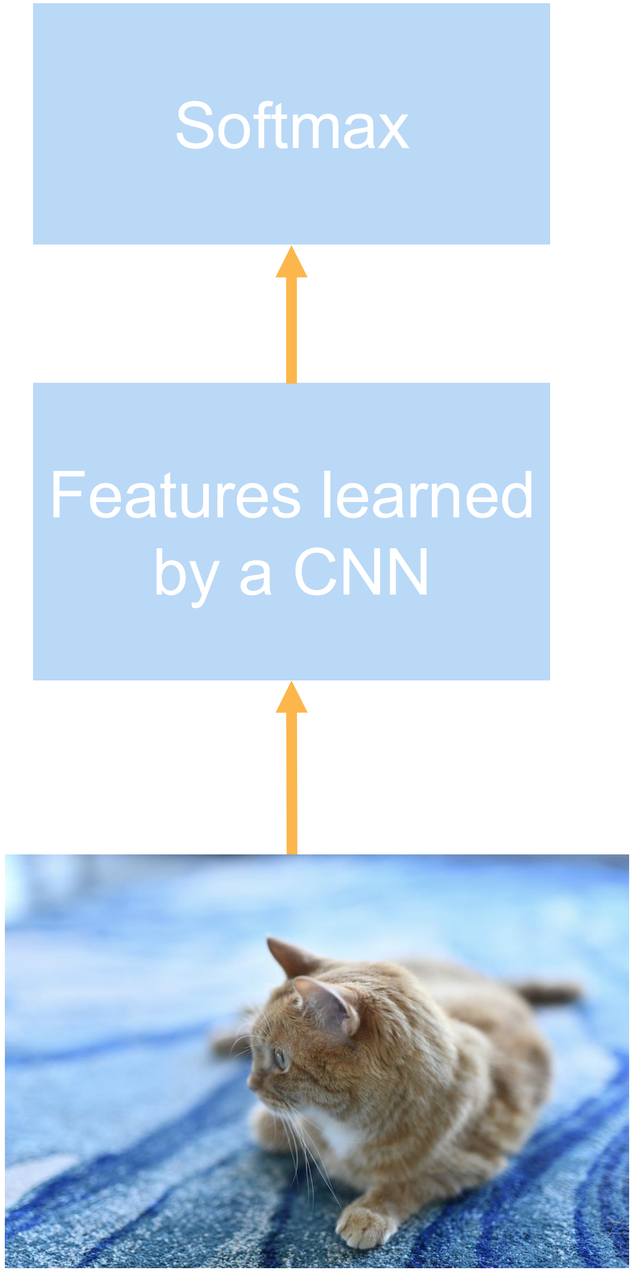
Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*.



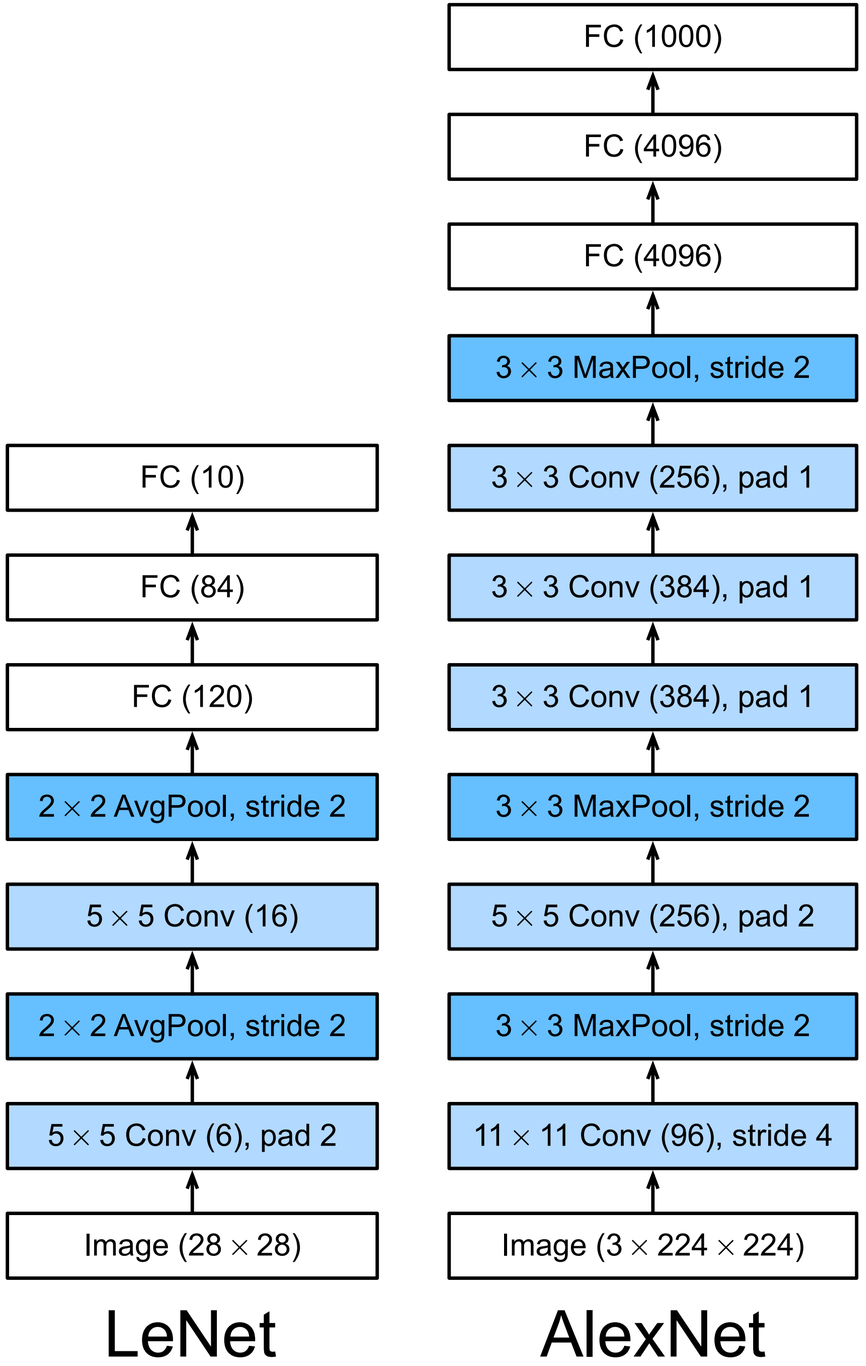
Deng et al. 2009

AlexNet

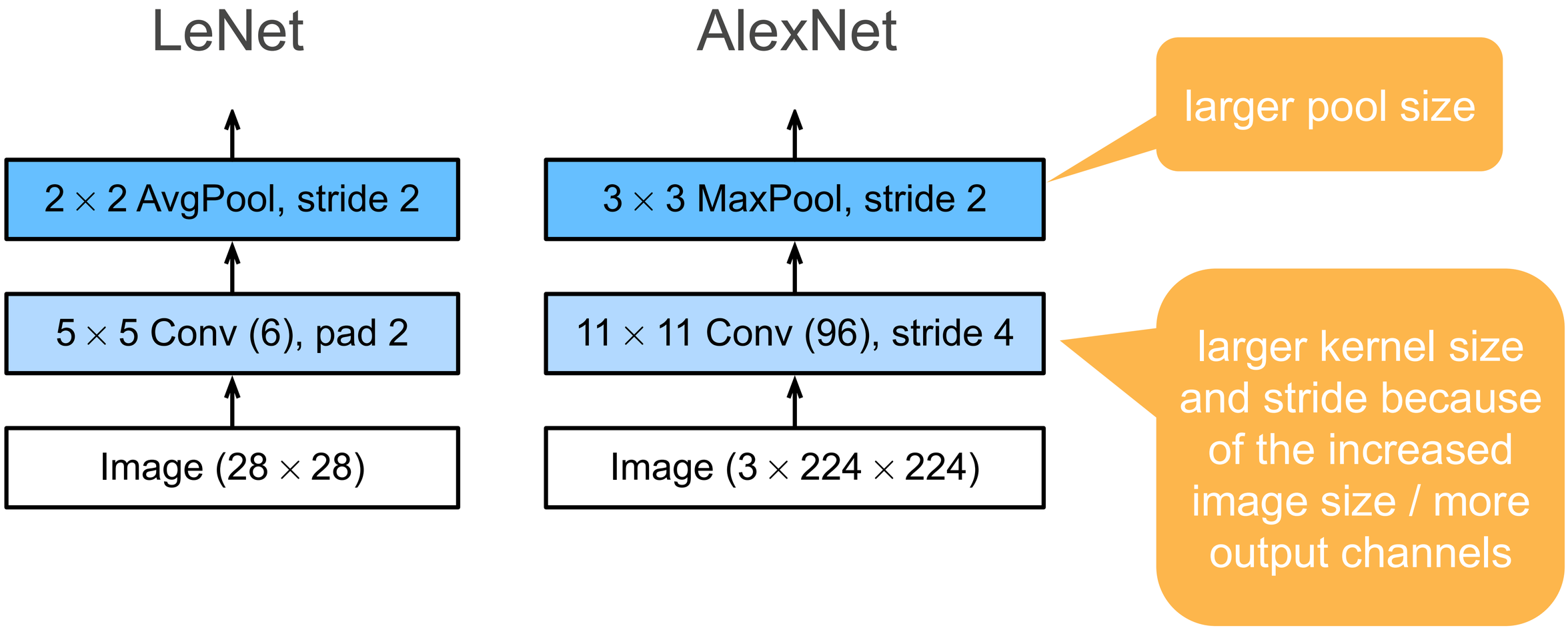
- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet, enabled by running on GPUs
- Paradigm shift for computer vision



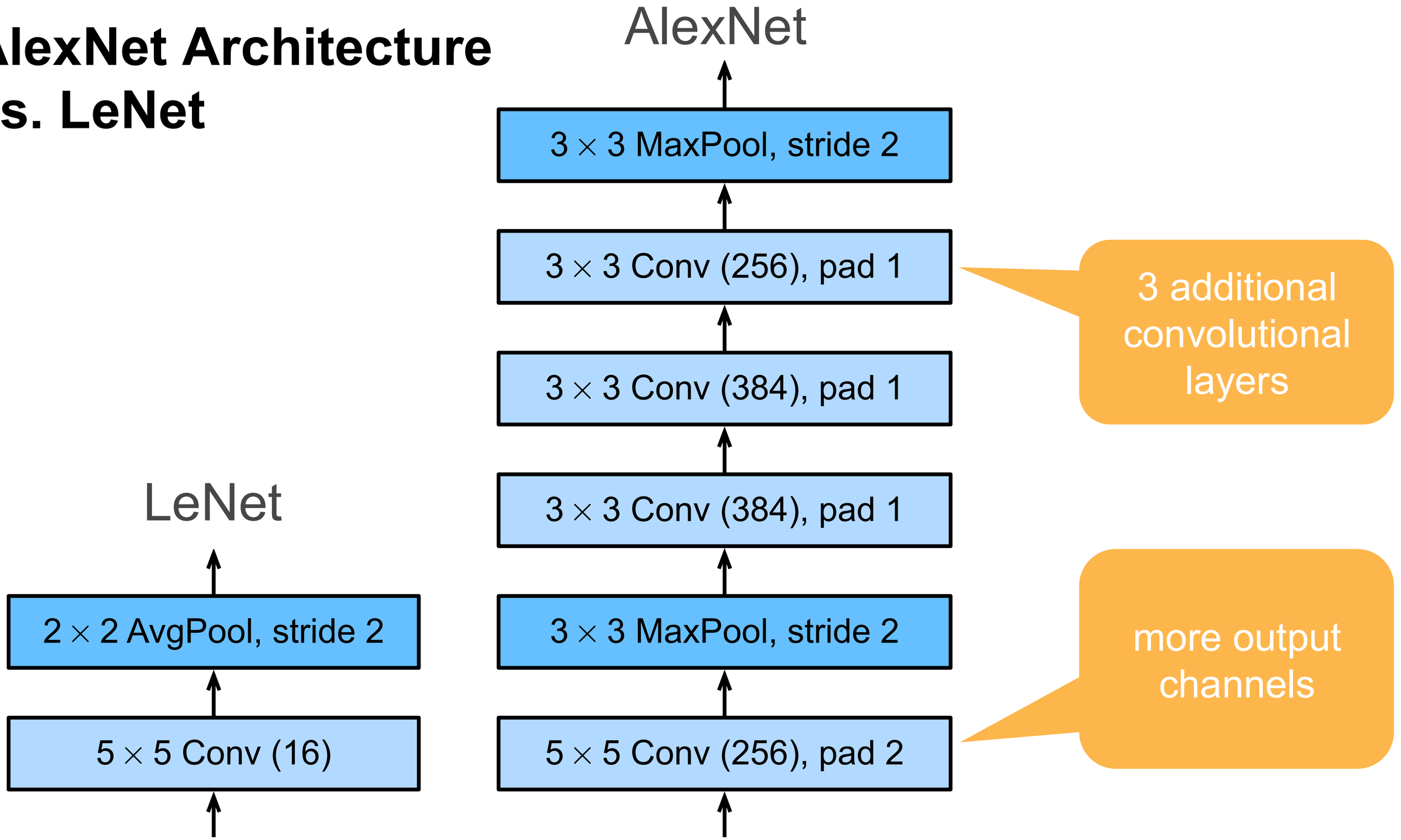
AlexNet Architecture vs. LeNet



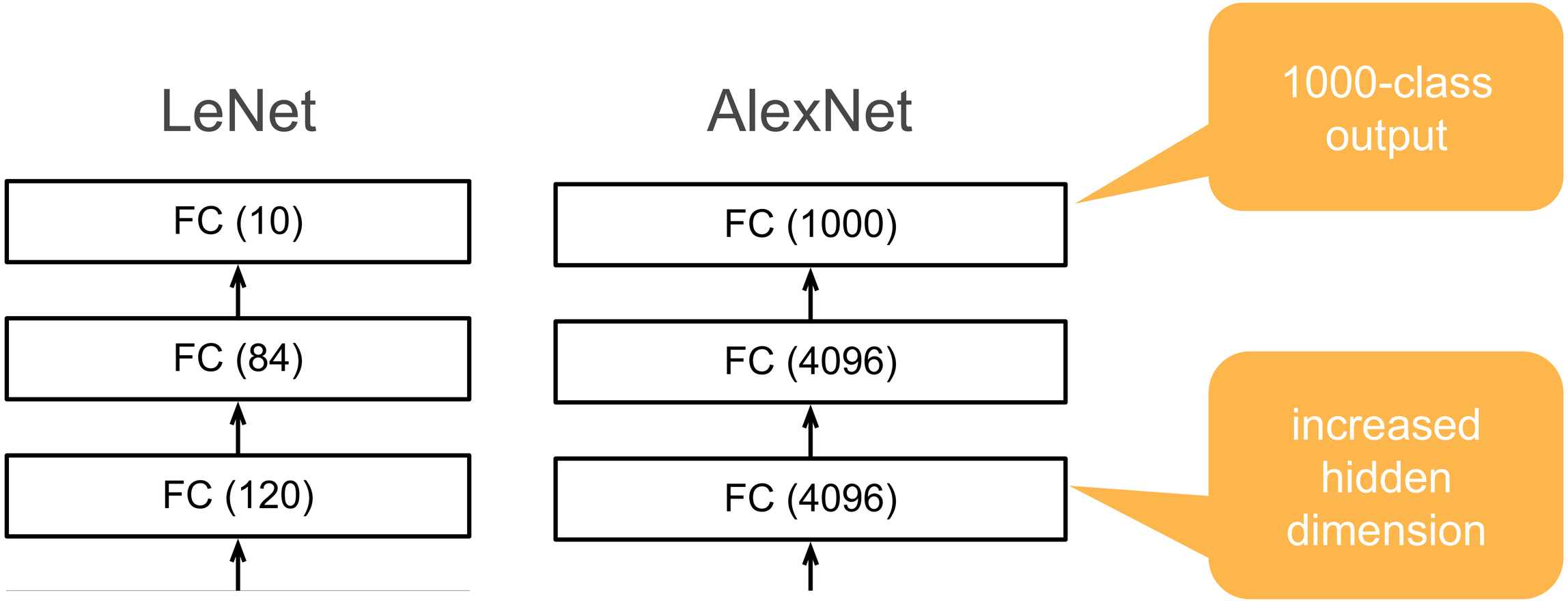
AlexNet Architecture vs. LeNet



AlexNet Architecture vs. LeNet

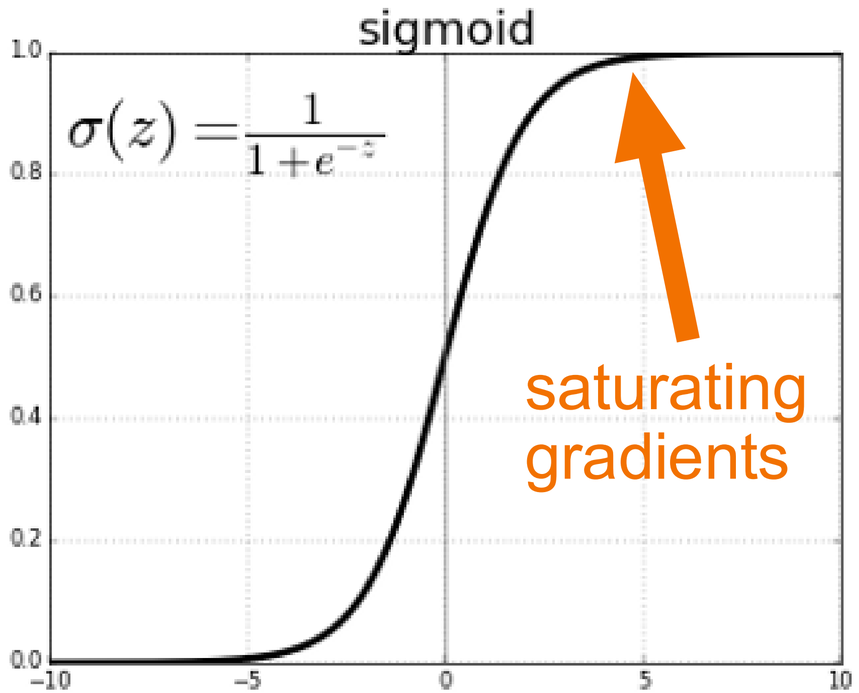


AlexNet Architecture vs. LeNet



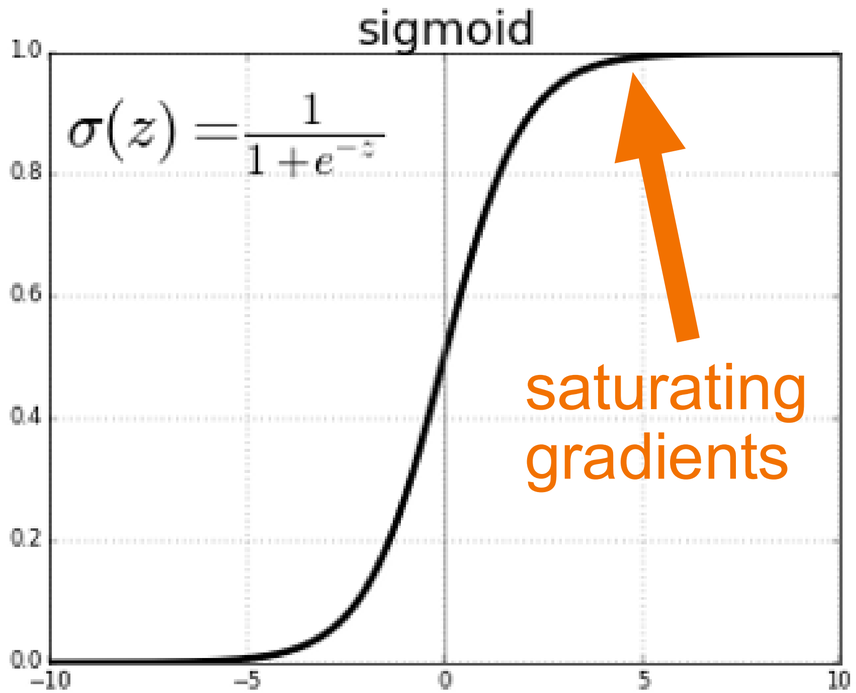
More Differences…

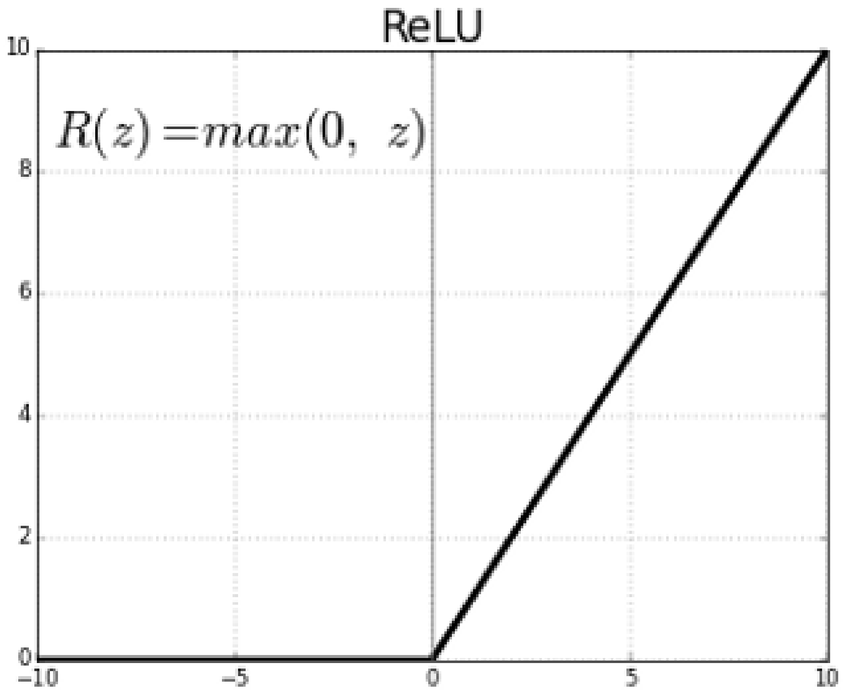
- Change activation function from sigmoid to ReLU (no more vanishing gradient)



More Differences…

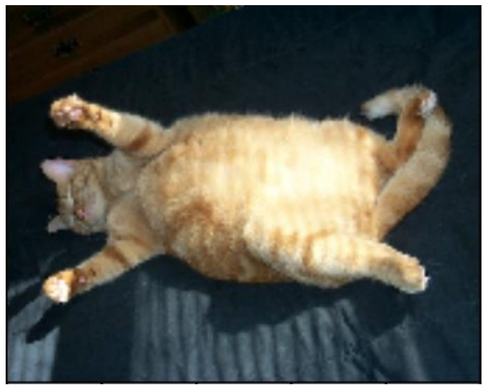
- Change activation function from sigmoid to ReLU (no more vanishing gradient)

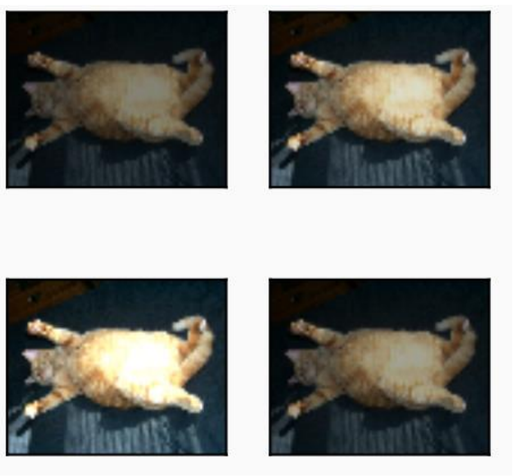
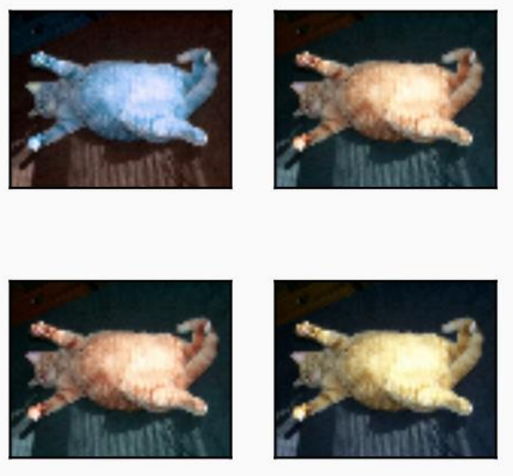




More Differences…

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

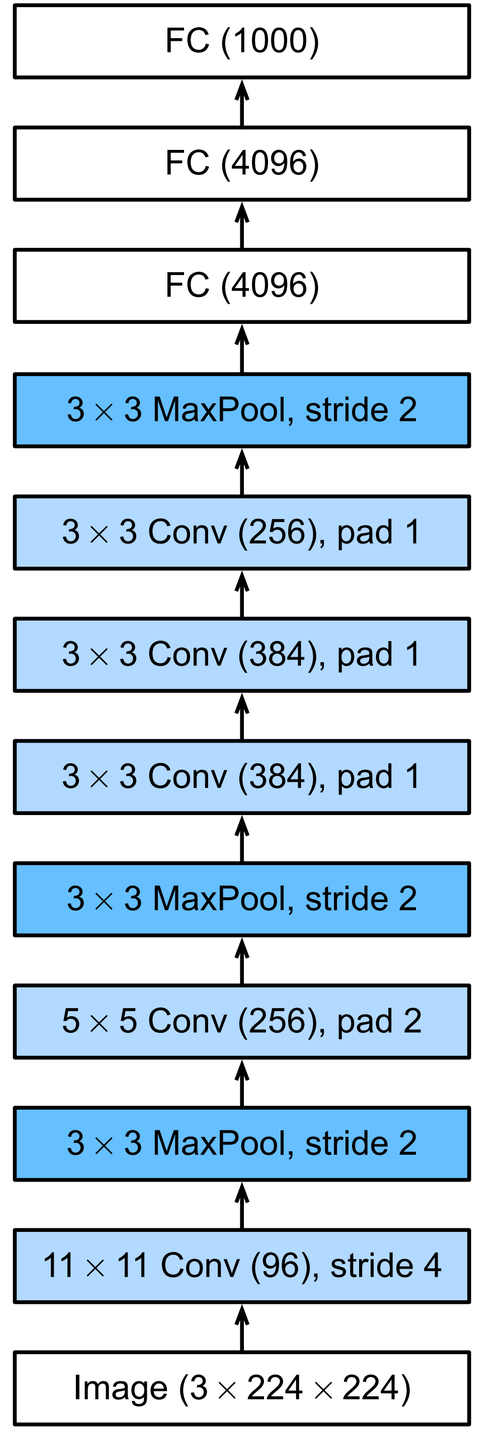


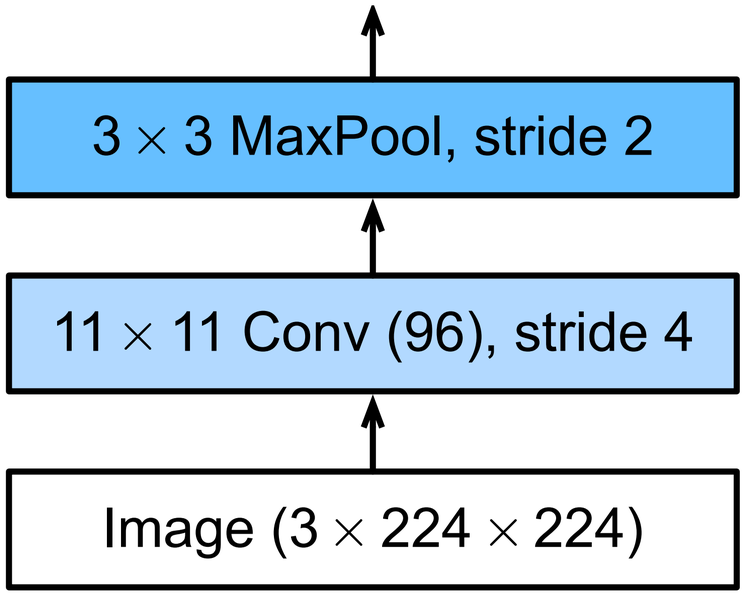
Complexity

	#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

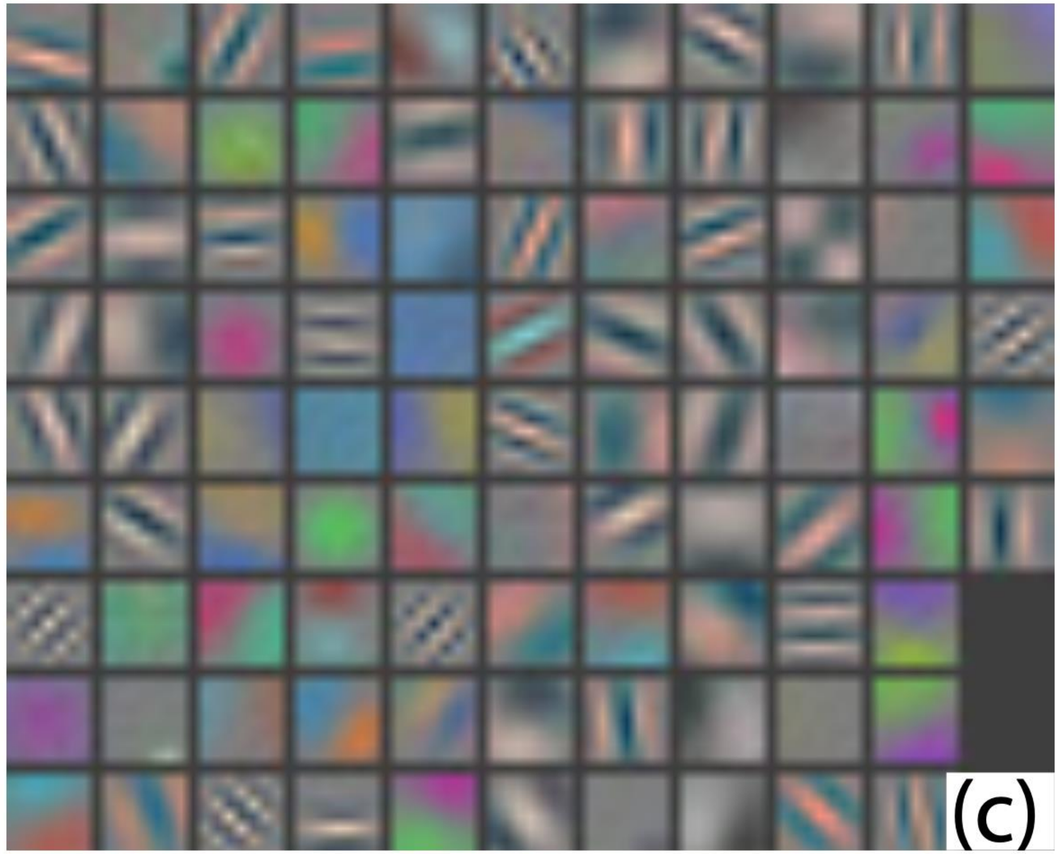
11x11x3x96=35k



AlexNet

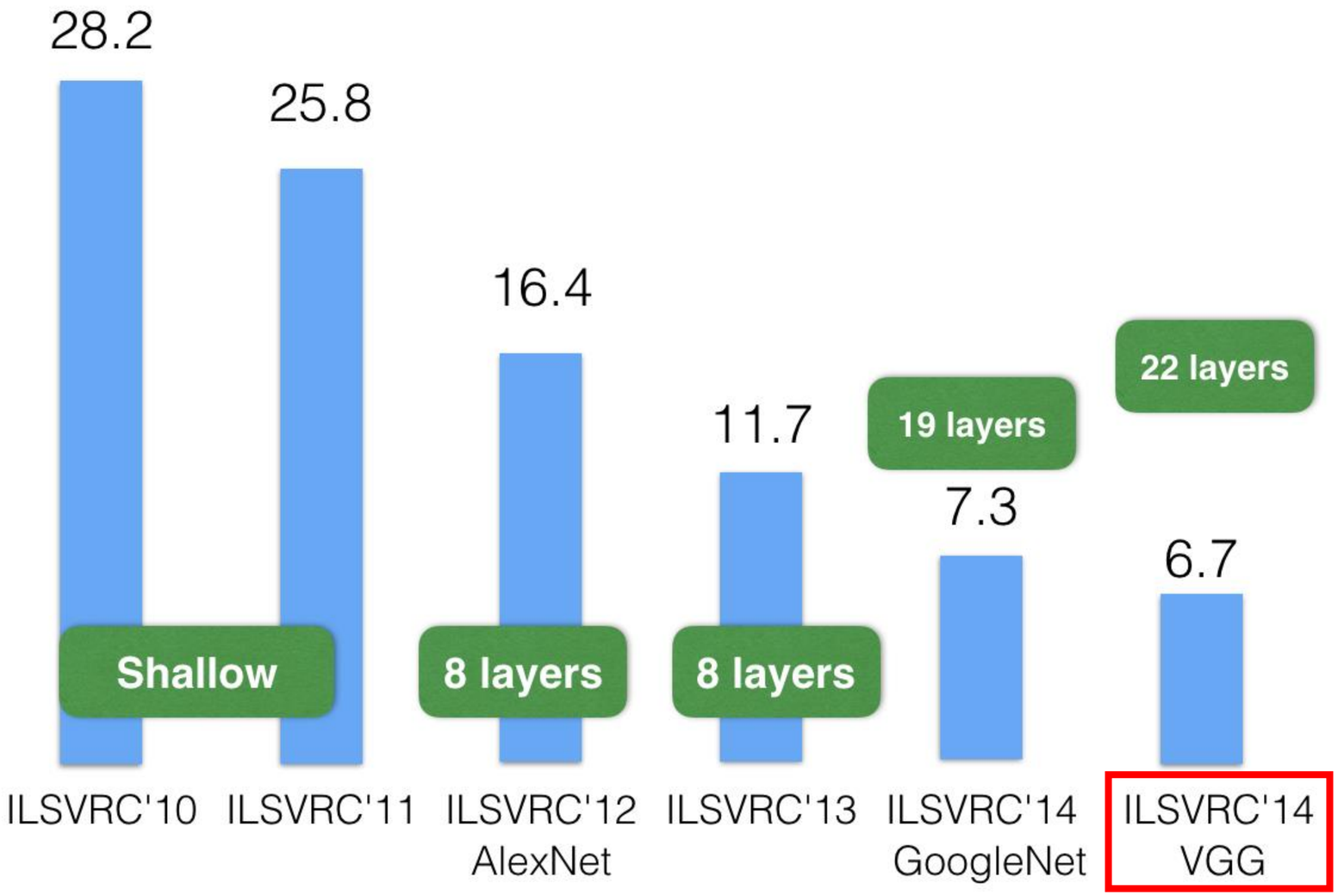


each Conv1 kernel is 3x11x11, can be visualized as an RGB patch



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

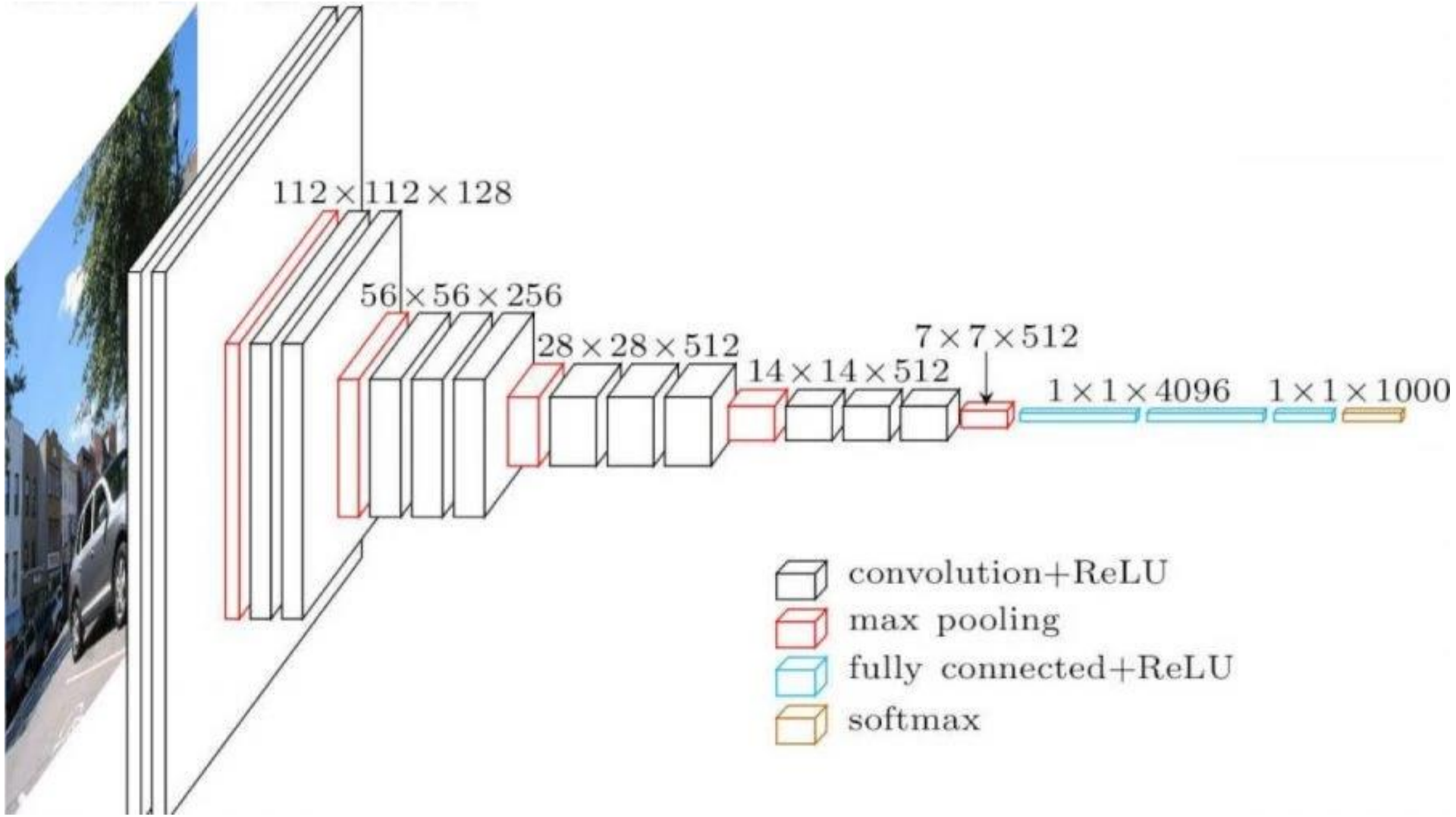

Adding Layers

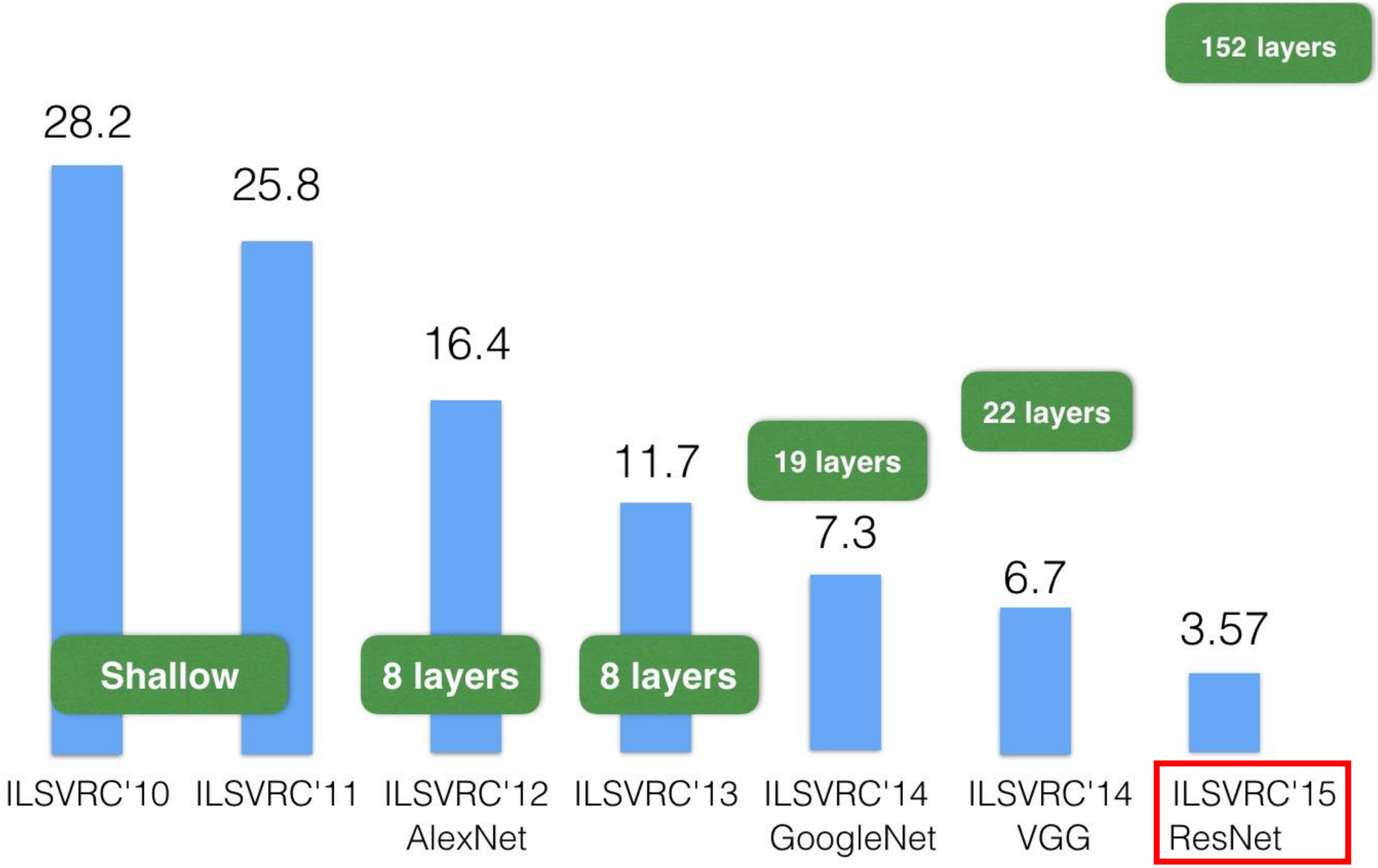


ImageNet Top-5 Classification Error (%)

VGG

VGG Block: Multiple convolution layers followed by pooling.





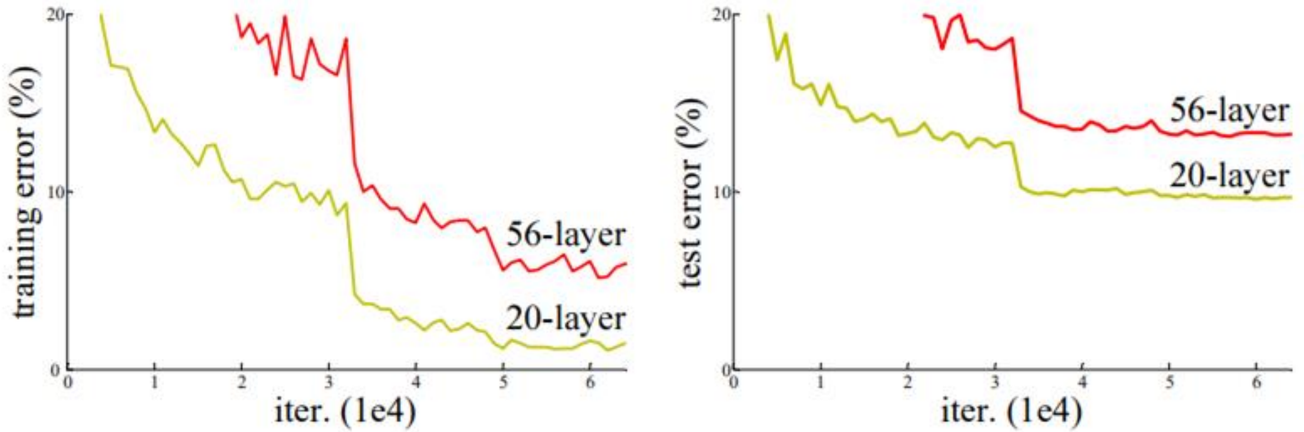
ImageNet Top-5 Classification Error (%)

Can we just keep adding layers to get from VGG (19 layers) to ResNet (152 layers)?

No! Some problems:

- i) Vanishing gradients: more layers ➔ more likely
- ii) Instability: deeper models are harder to optimize

Reflected in training error:

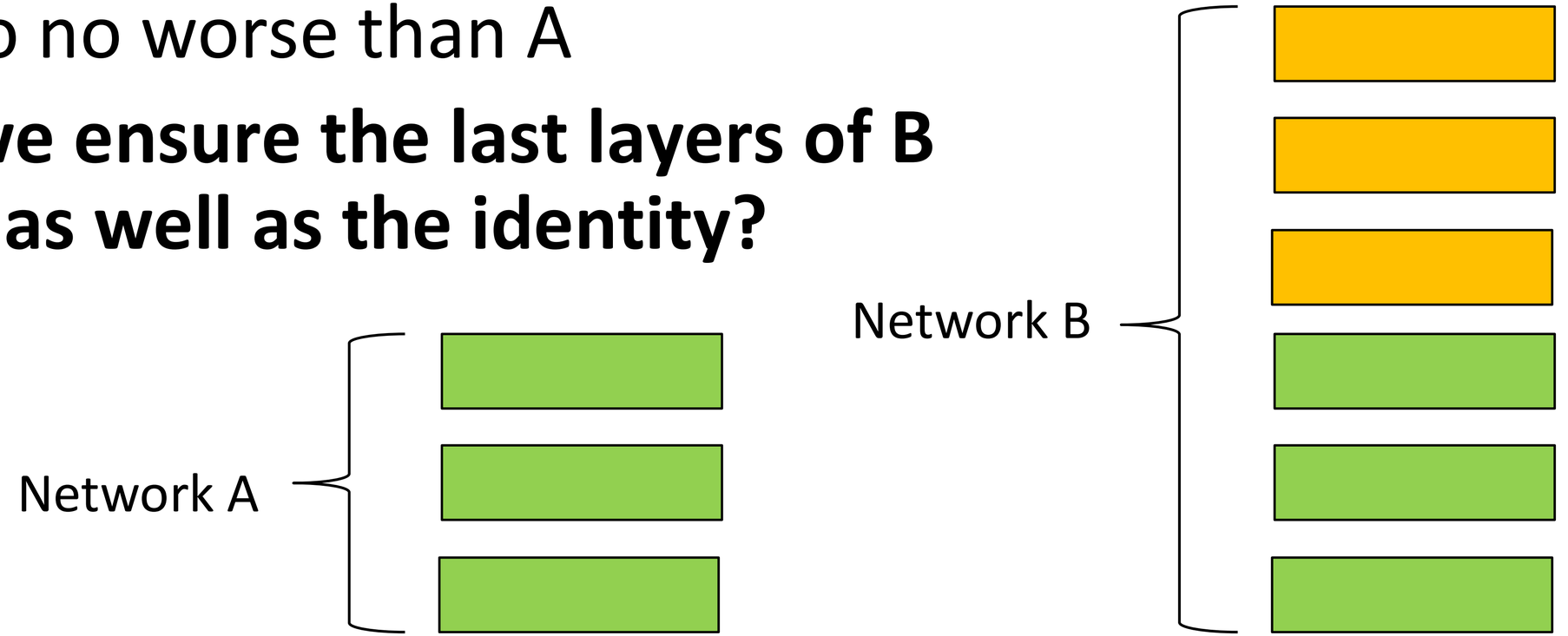


He et al: “Deep Residual Learning for Image Recognition”

Depth Issues & Learning Identity

Consider two architectures:

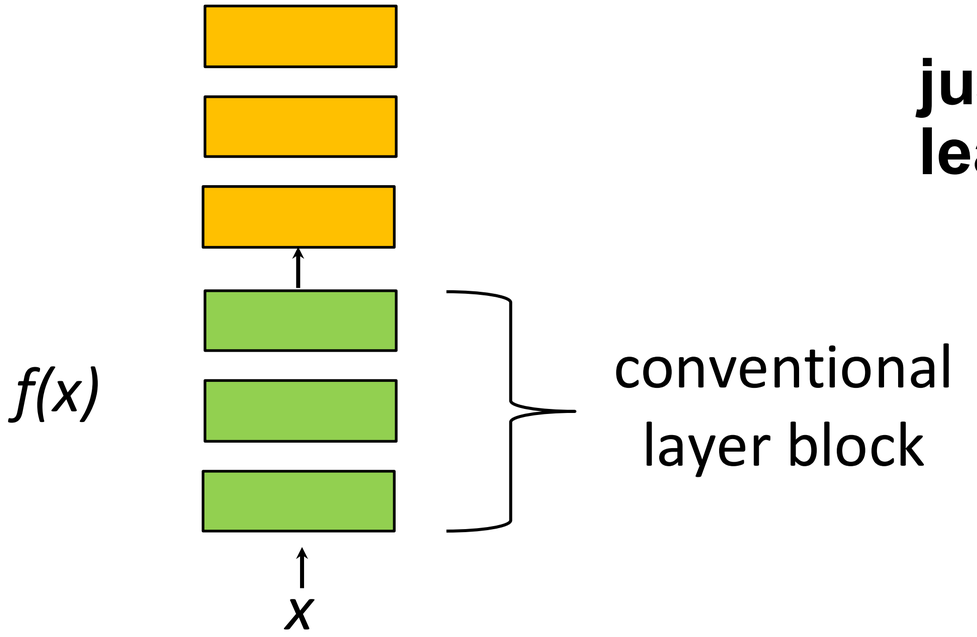
- Network B just adds layers on top of A
- If these new layers can learn the identity, B should do no worse than A
- **Idea: can we ensure the last layers of B do at least as well as the identity?**



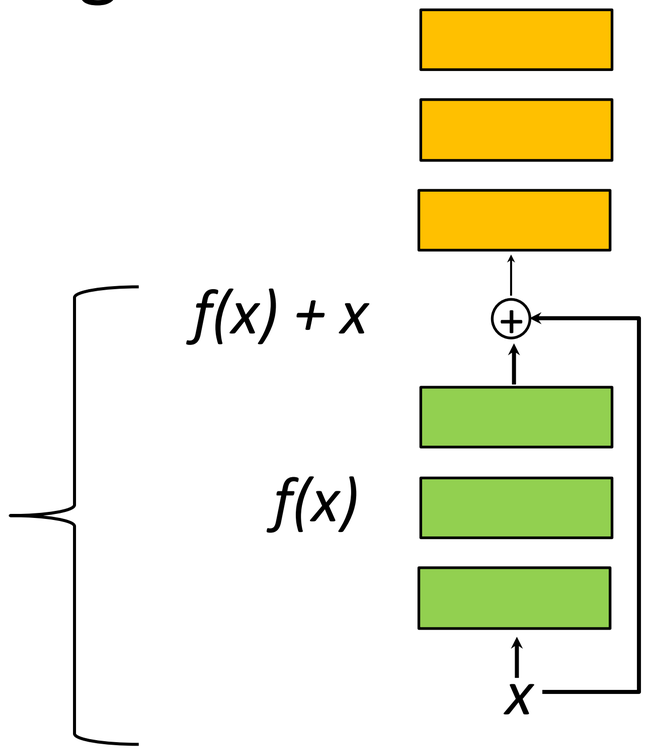
Residual Connections

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:



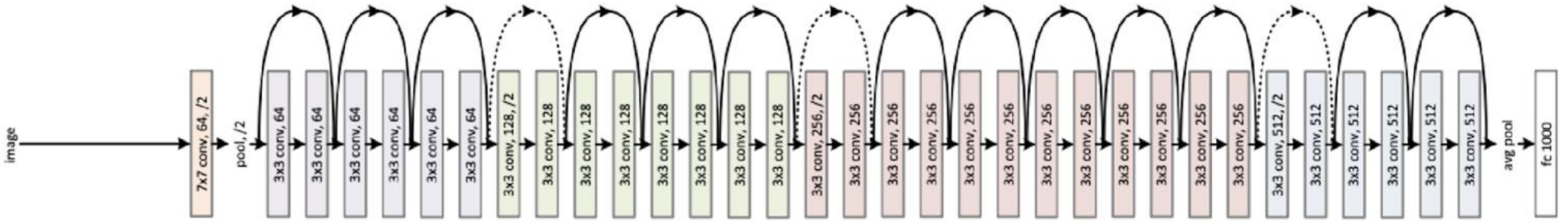
just need to learn f(x)=0!



ResNet Architecture

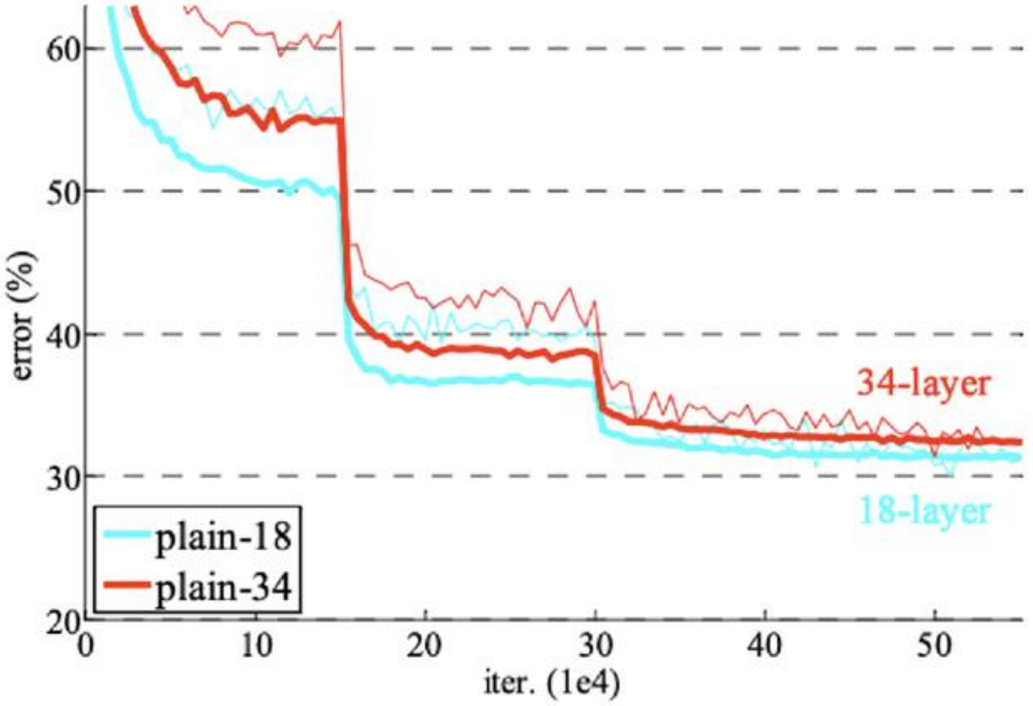
Idea: Residual (skip) connections help make learning easier

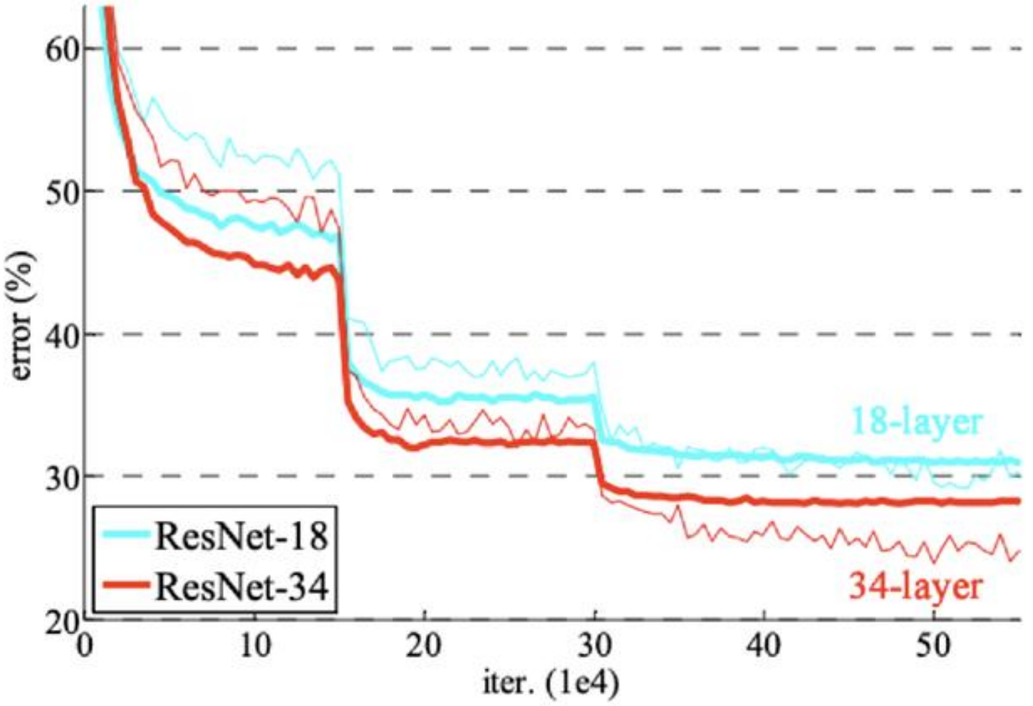
- Note: residual connections every two layers
- **Vastly better** performance
  - No additional parameters!
  - Set records on many benchmarks



ResNet Training Curves on ImageNet

[He et al., 2015]





Progress

- LeNet (1989)
  - 2 convolution + pooling layers
  - 2 hidden dense layers
- AlexNet (2012)
  - Bigger and deeper LeNet
  - ReLu, preprocessing
- VGG (2014)
  - Bigger and deeper AlexNet (repeated VGG blocks)
- ResNet (2015)
  - Residual (skip) connections

Summary of today

- Reviewed (some of) convolutional computations.
  - 2D convolutions, multiple input channels, pooling.
- Showed 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

Other Deep Architectures

- Convolutional neural networks are one of many special types of layers.
  - Main use is for processing images.
  - Also can be useful for handling time series.
- Other common architectures:
  - **Recurrent neural networks:** hidden activations are a function of input and activations from previous inputs. Designed for sequential data such as text.
  - **Graph neural networks:** take graph data as input.
  - **Transformers:** take sequences as input and learn what parts of input to pay attention to.



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