CS540 Introduction to Artificial Intelligence

Deep Learning II: Convolutional Neural Networks

Yingyu Liang

University of Wisconsin-Madison

Nov 4, 2021

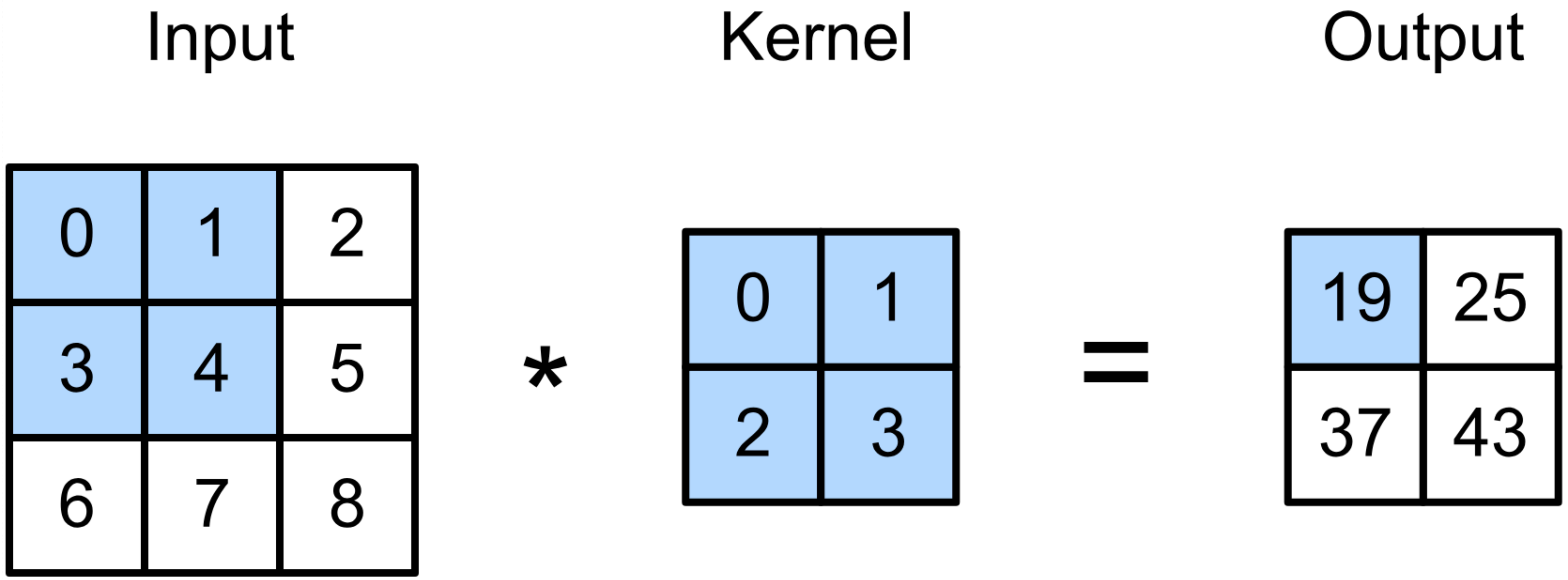
Slides created by Sharon Li [modified by Yingyu Liang]

Outline

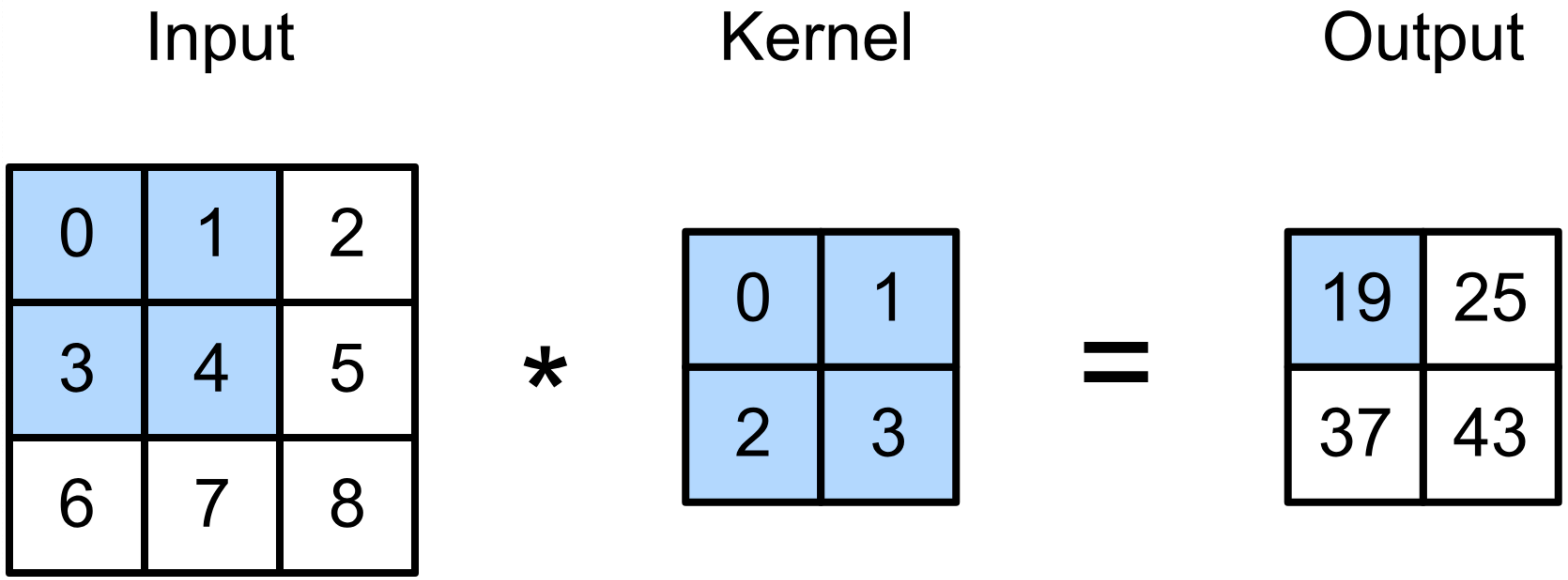
- Brief review of convolutional computations
- Convolutional Neural Networks
    - LeNet (first conv nets)
    - AlexNet
    - ResNet

Review: 2-D Convolution

Review: 2-D Convolution



Review: 2-D Convolution



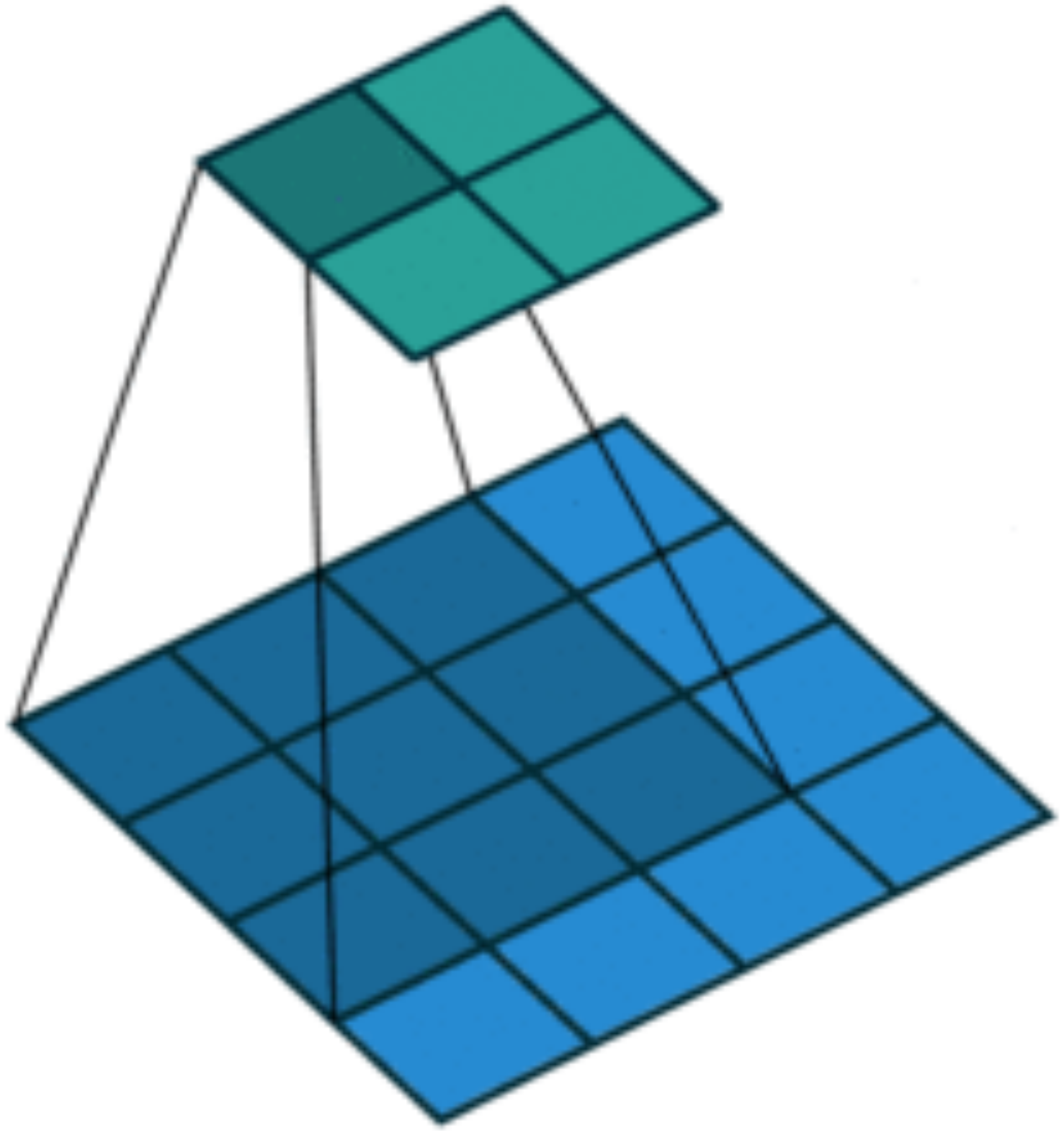
$$
\begin{aligned}
0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 &= 19, \\
1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 &= 25, \\
3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 &= 37, \\
4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 &= 43.
\end{aligned}
$$

Review: 2-D Convolution

Input * Kernel = Output

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

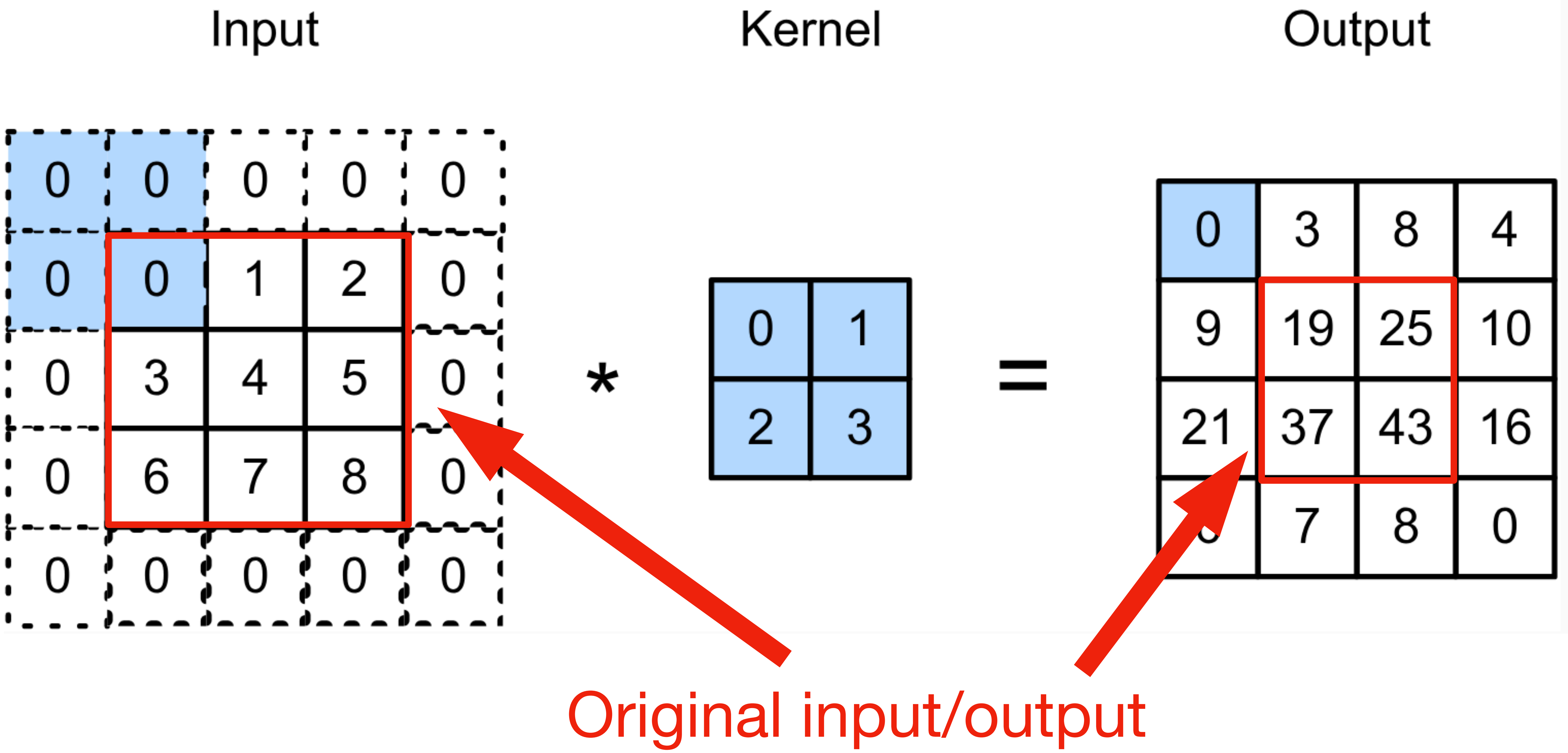
$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$



(vdumoulin@ Github)

Padding

Padding adds rows/columns around input

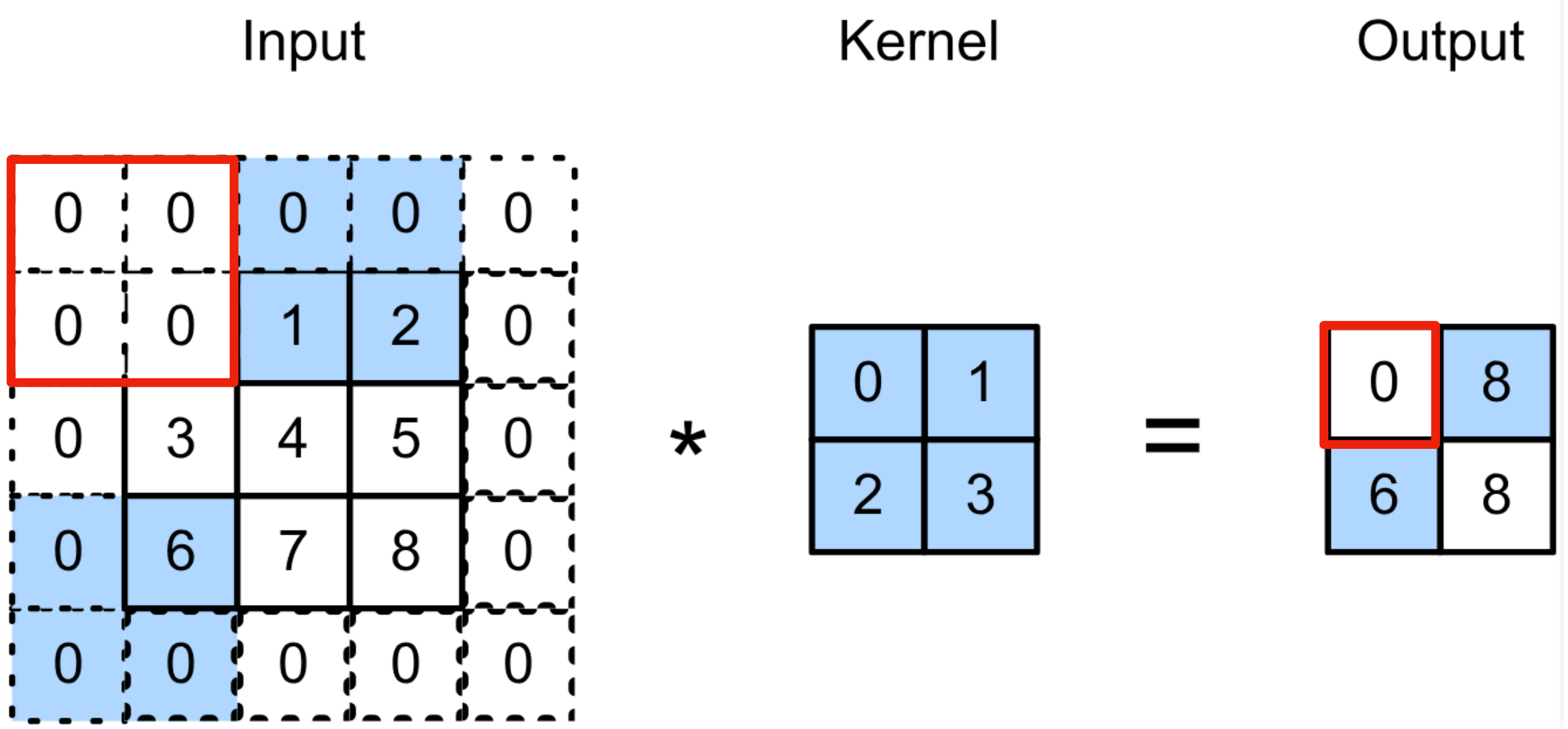


$$0 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times 3 = 0$$

Stride

- Stride is the #rows/#columns per slide

Strides of 3 and 2 for height and width



$$0 \times 0 + 0 \times 1 + 1 \times 2 + 2 \times 3 = 8$$
$$0 \times 0 + 6 \times 1 + 0 \times 2 + 0 \times 3 = 6$$

Output shape

Kernel/filter size

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

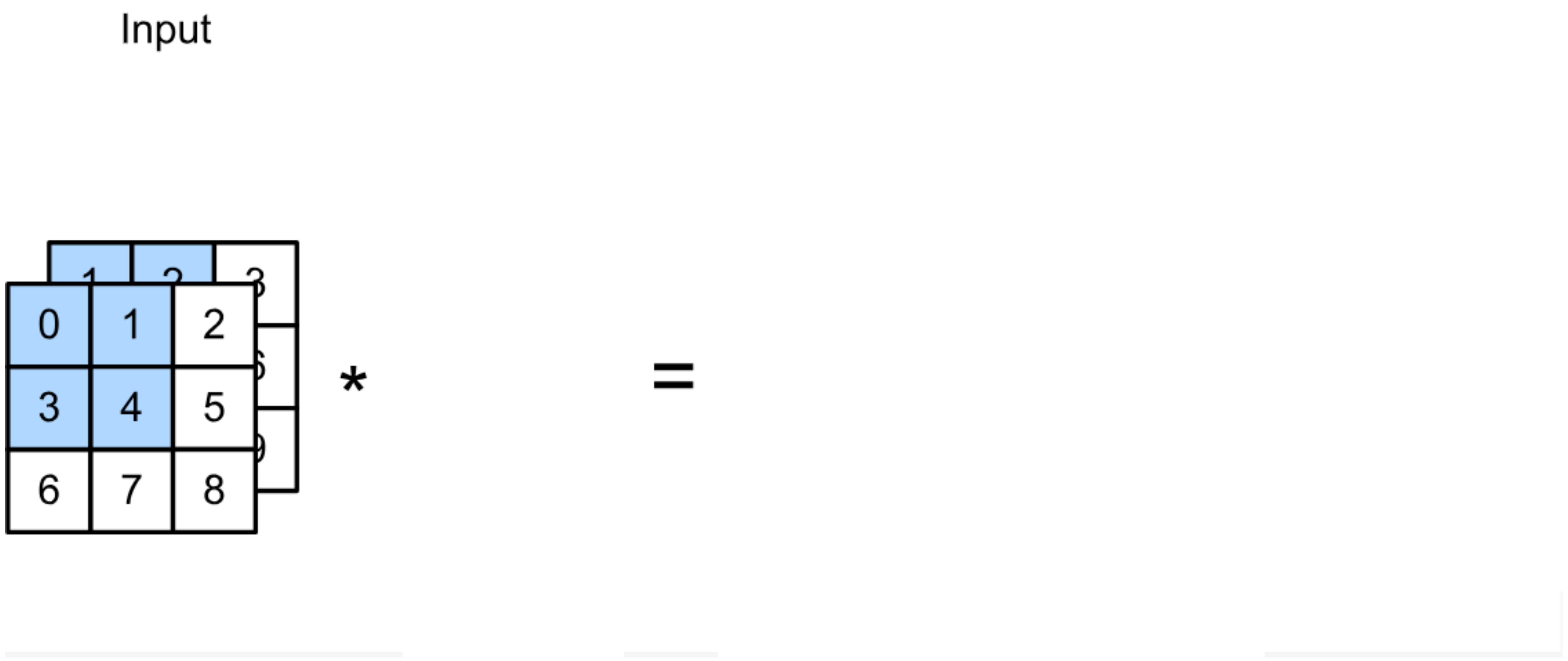
Input size

Pad

Stride

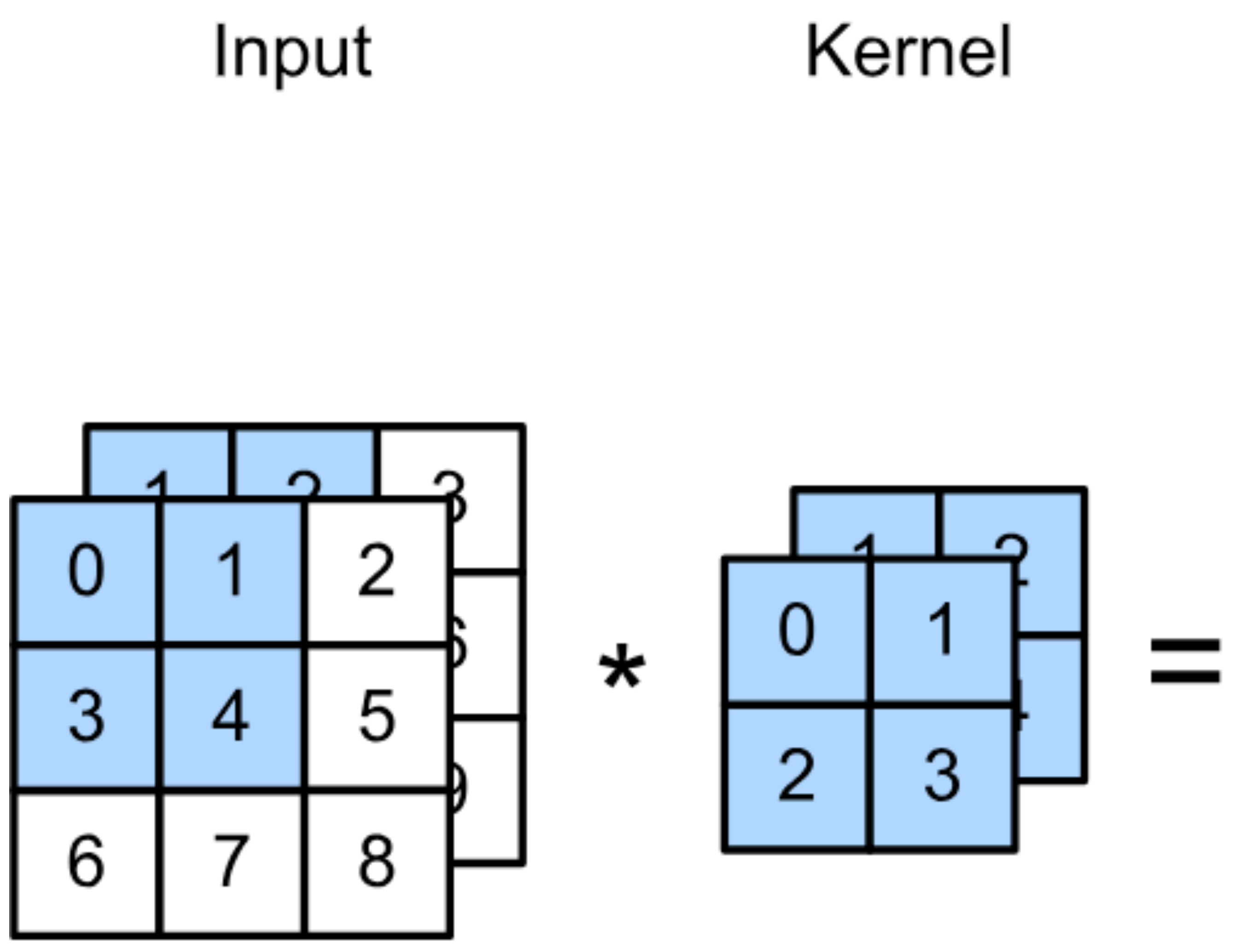
Review: Multiple Input Channels

- 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



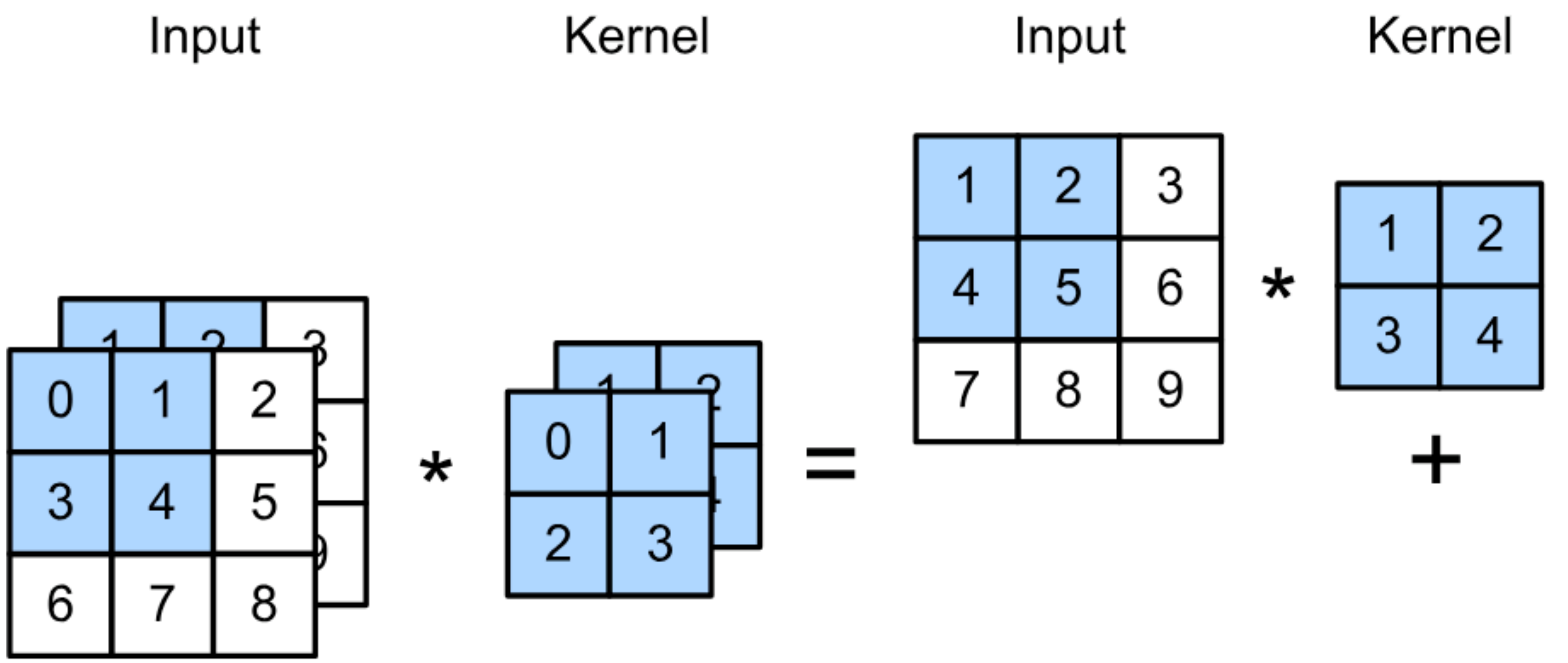
Review: Multiple Input Channels

- 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



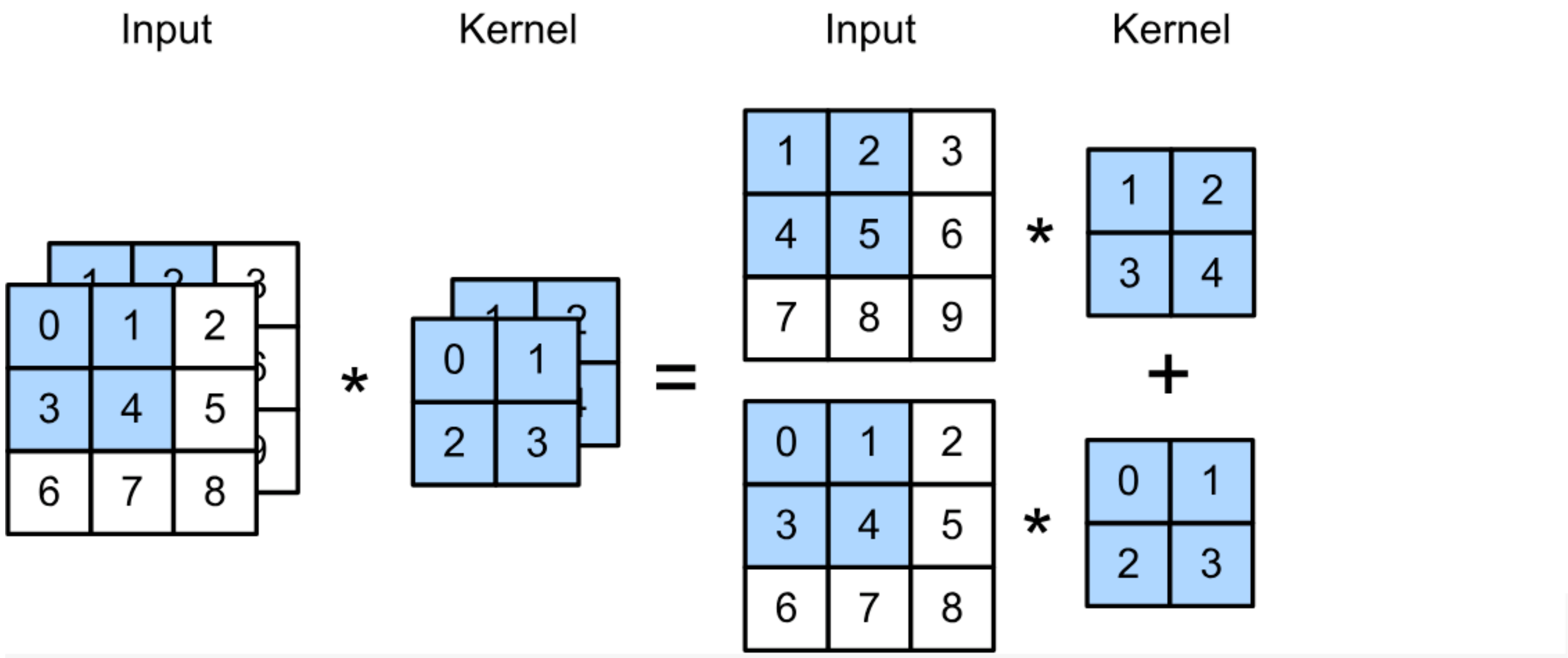
Review: Multiple Input Channels

- 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



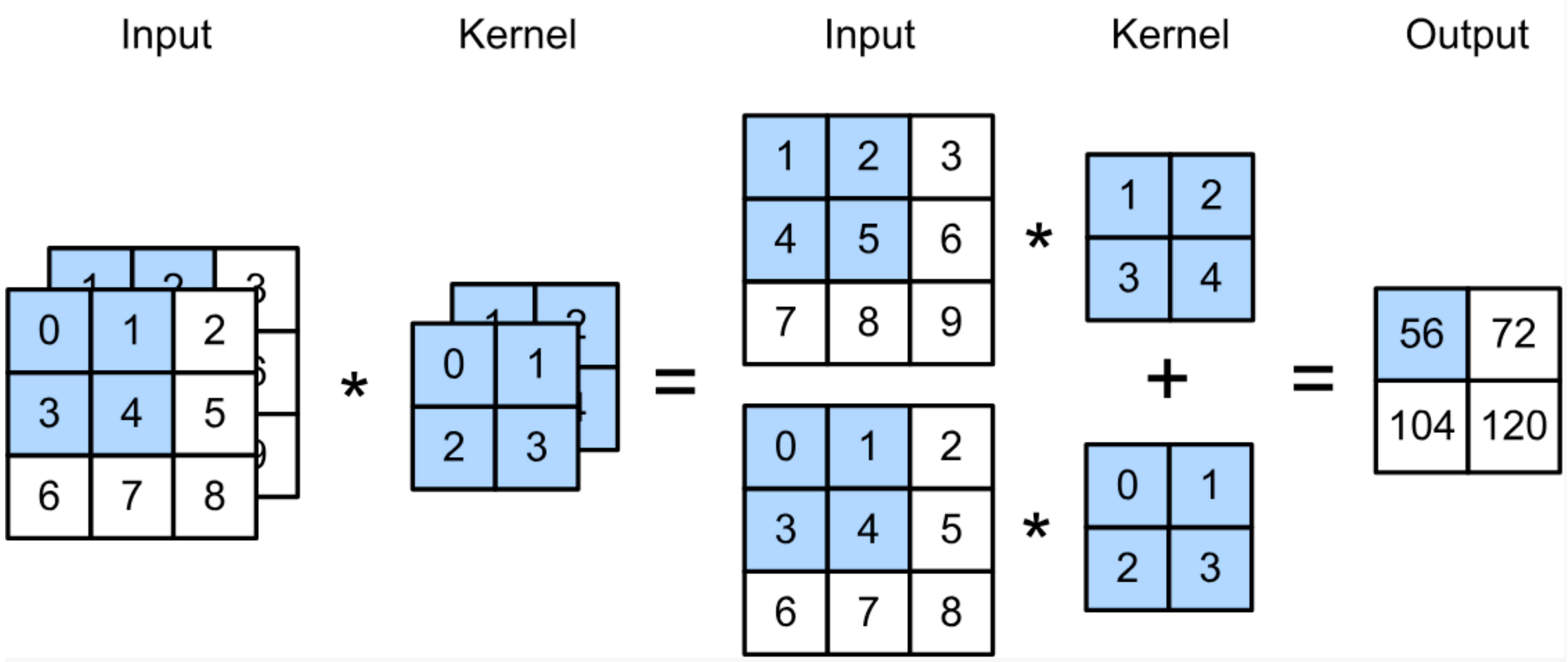
Review: Multiple Input Channels

- 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

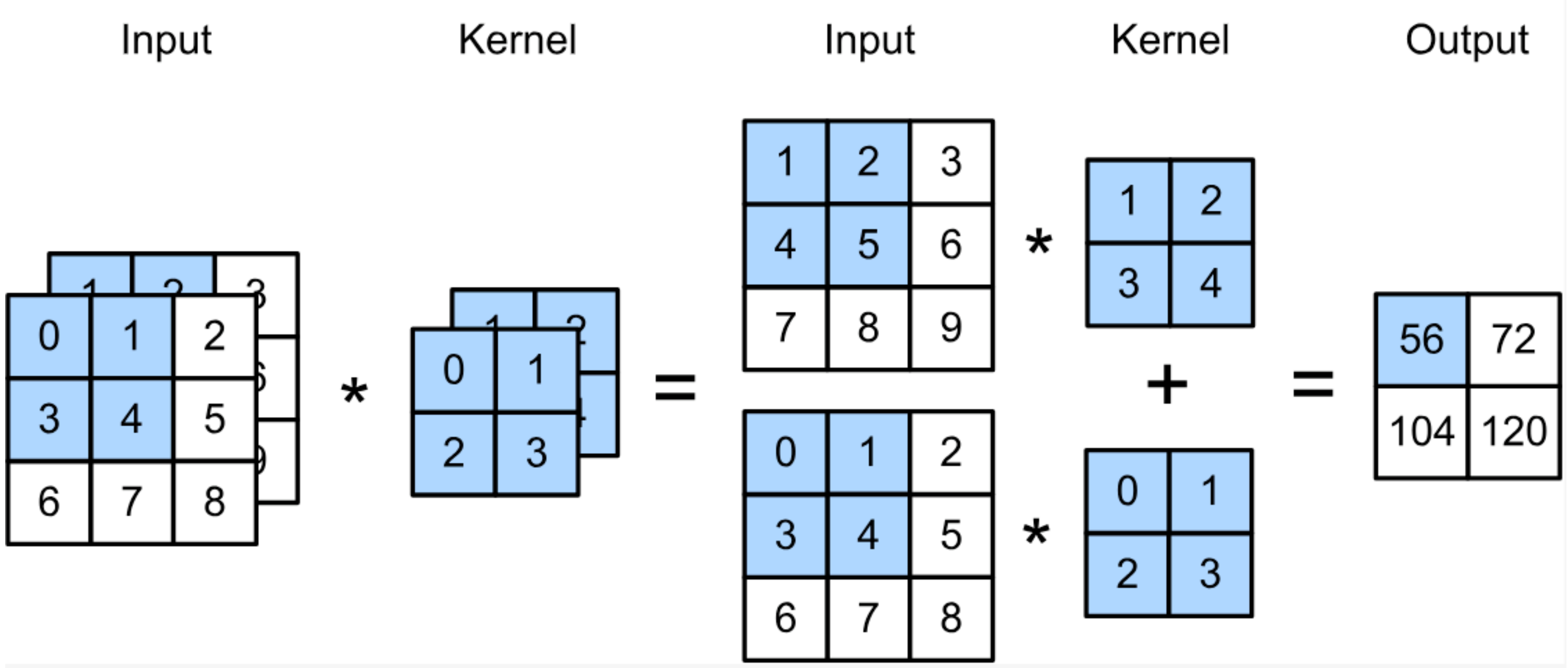


(

-

Review: Multiple Input Channels

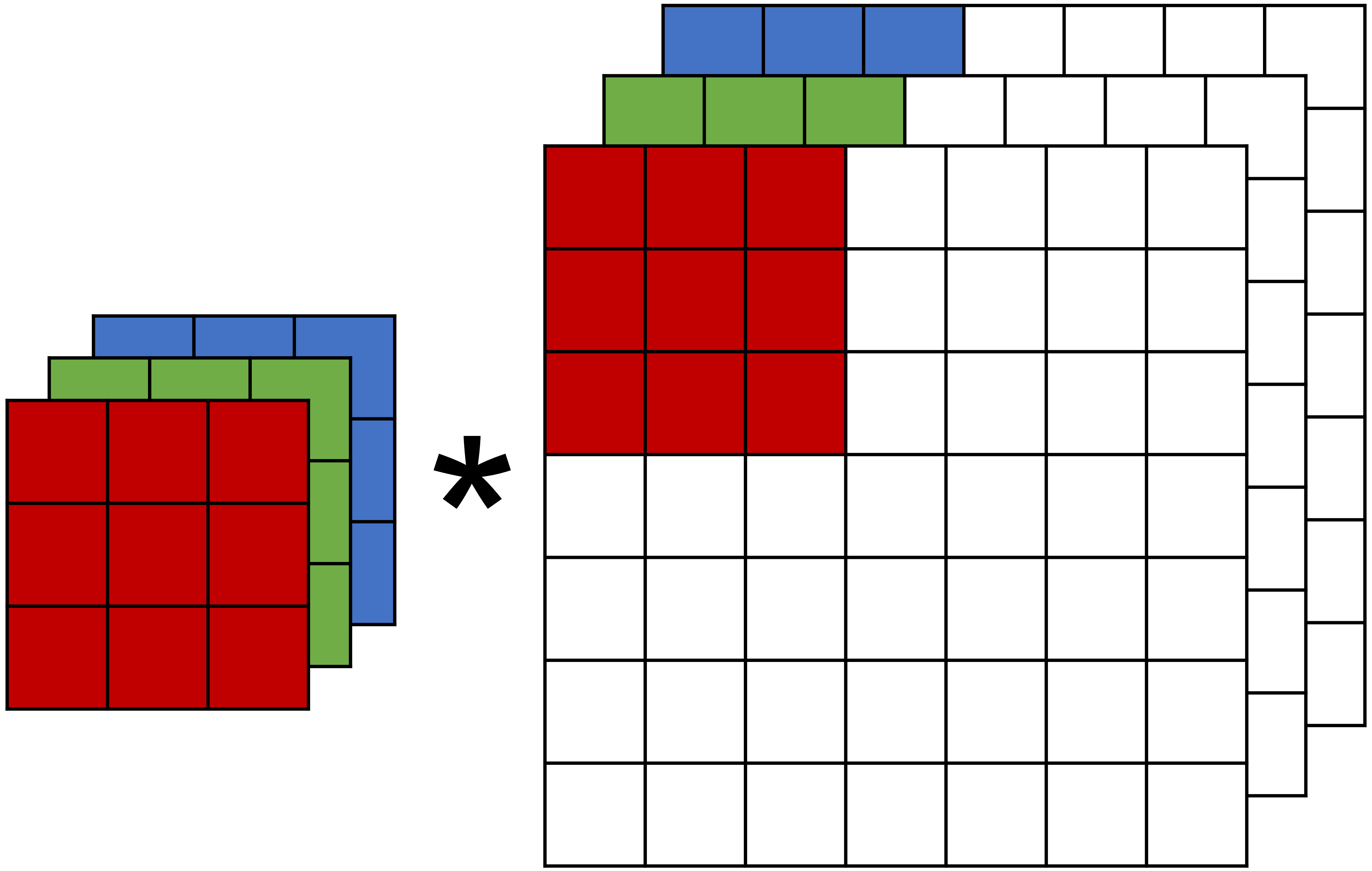
- Input and kernel can be 3D, e.g., an RGB image have 3 channels
- Have a kernel for each channel, and then sum results over channels



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

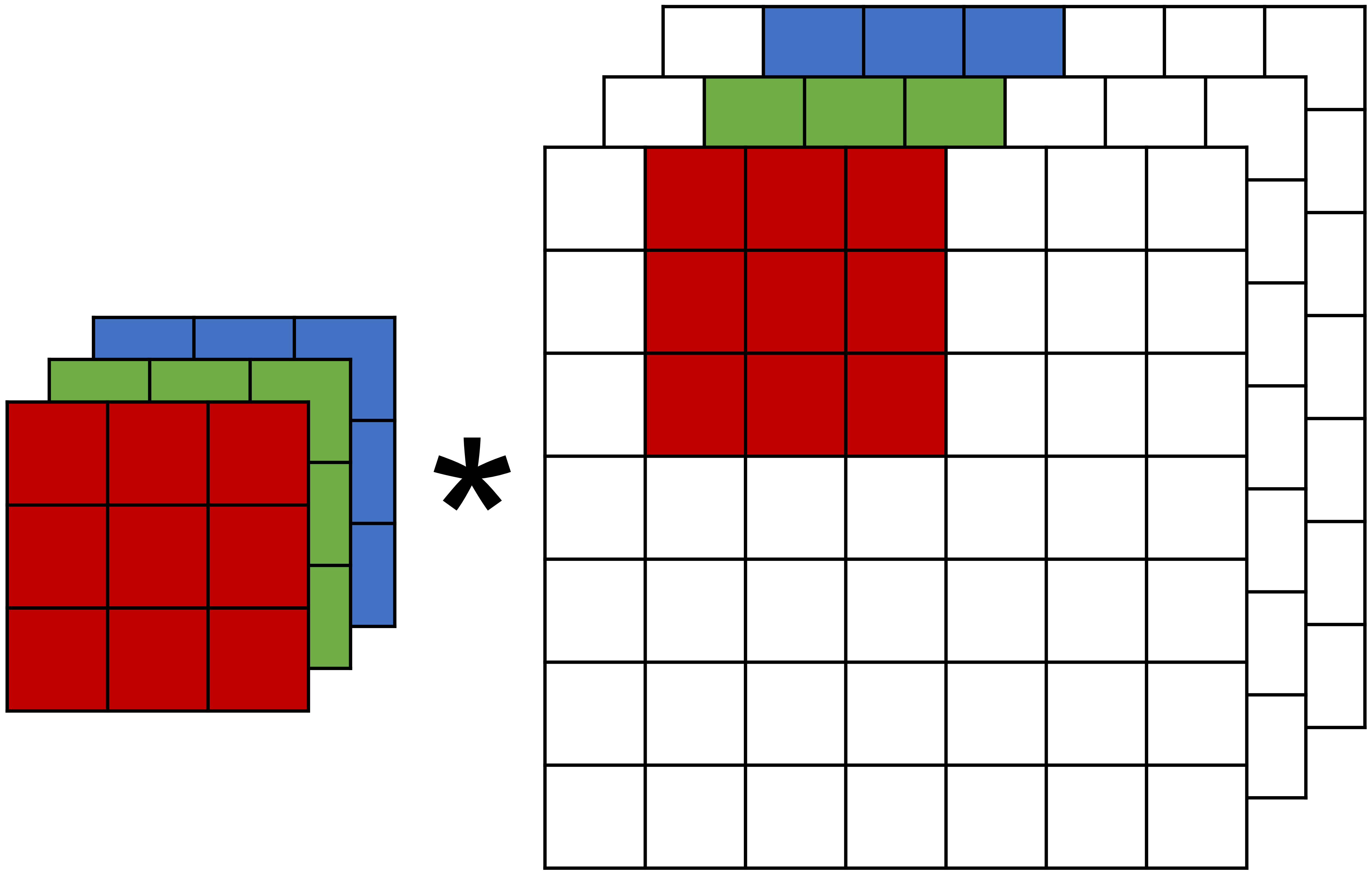
Review: Multiple Input Channels

- 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



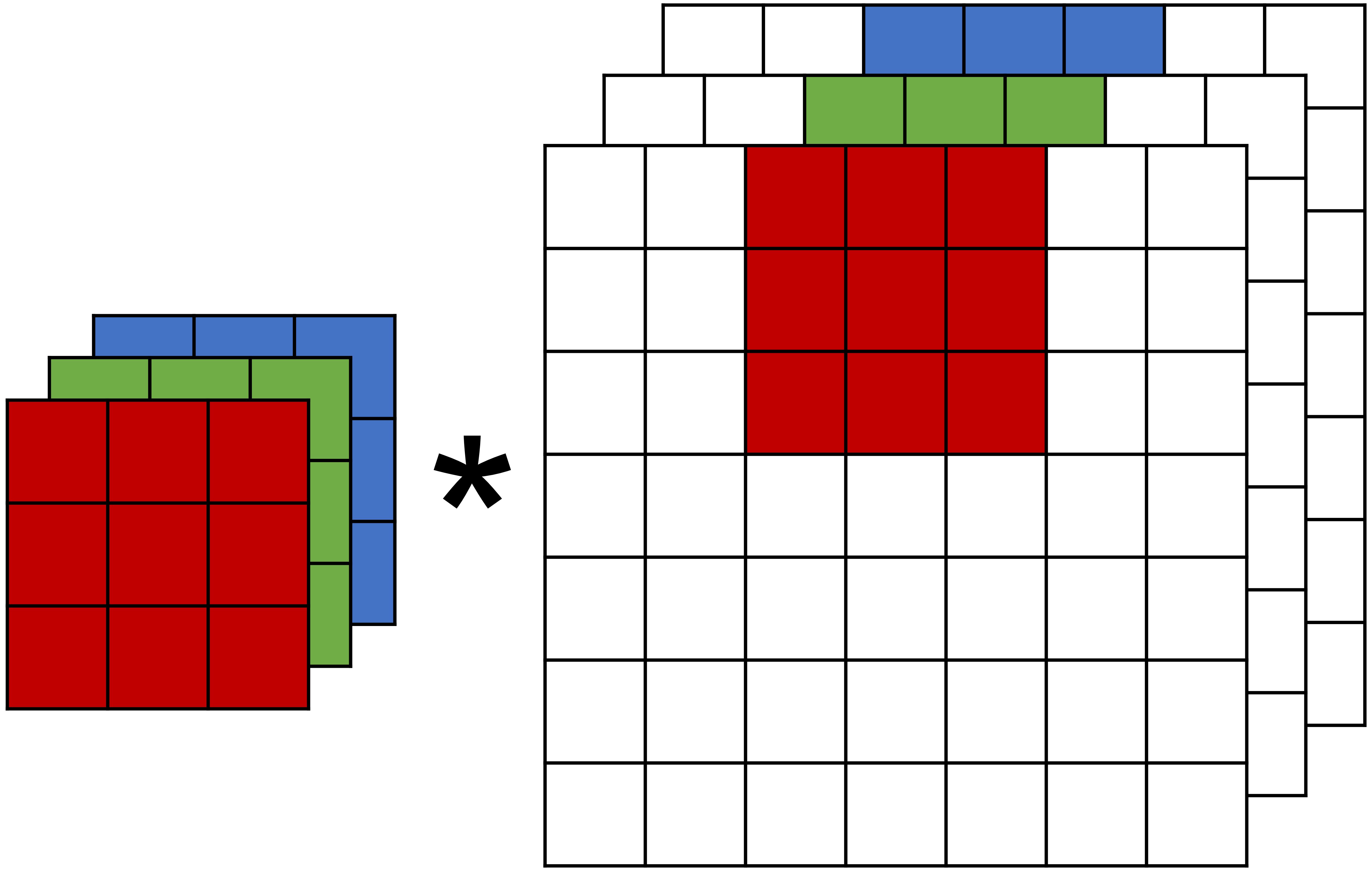
Review: Multiple Input Channels

- 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



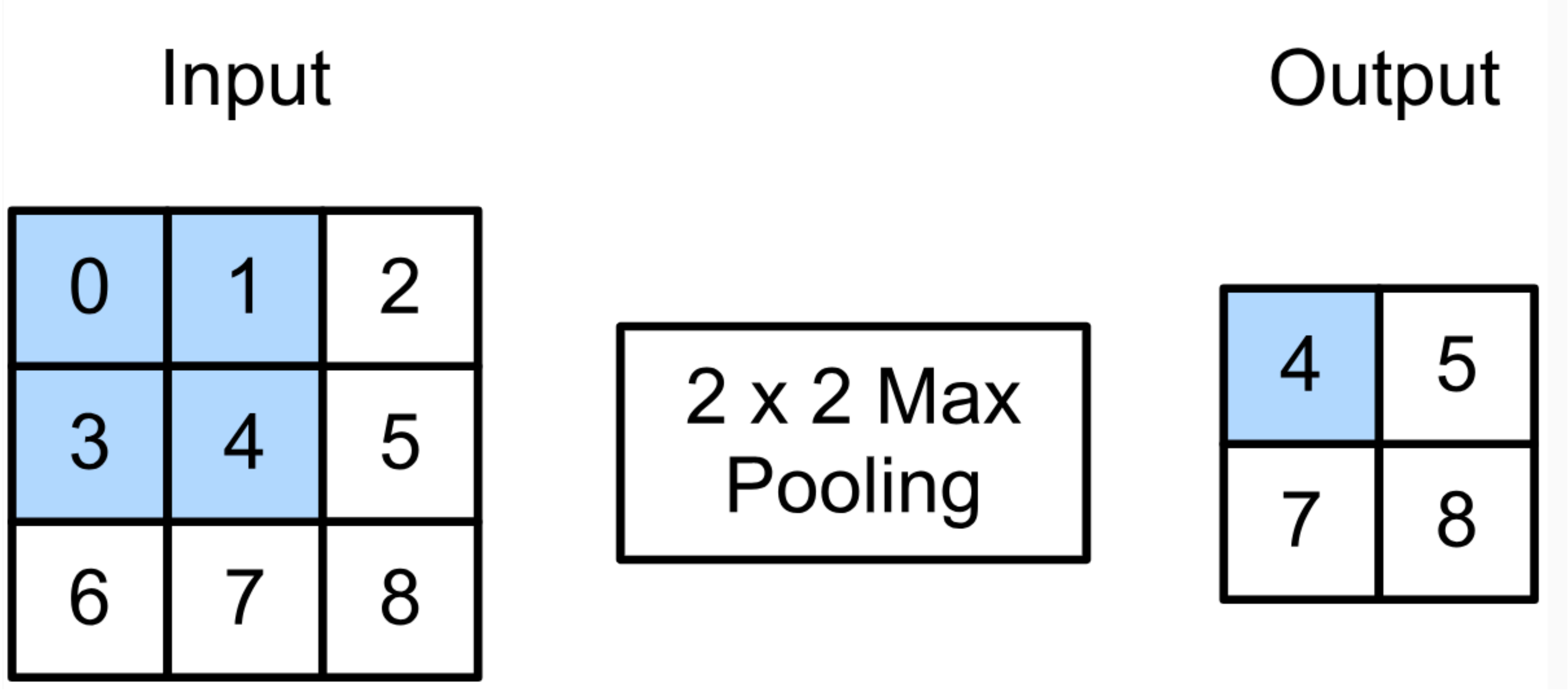
Review: Multiple Input Channels

- 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

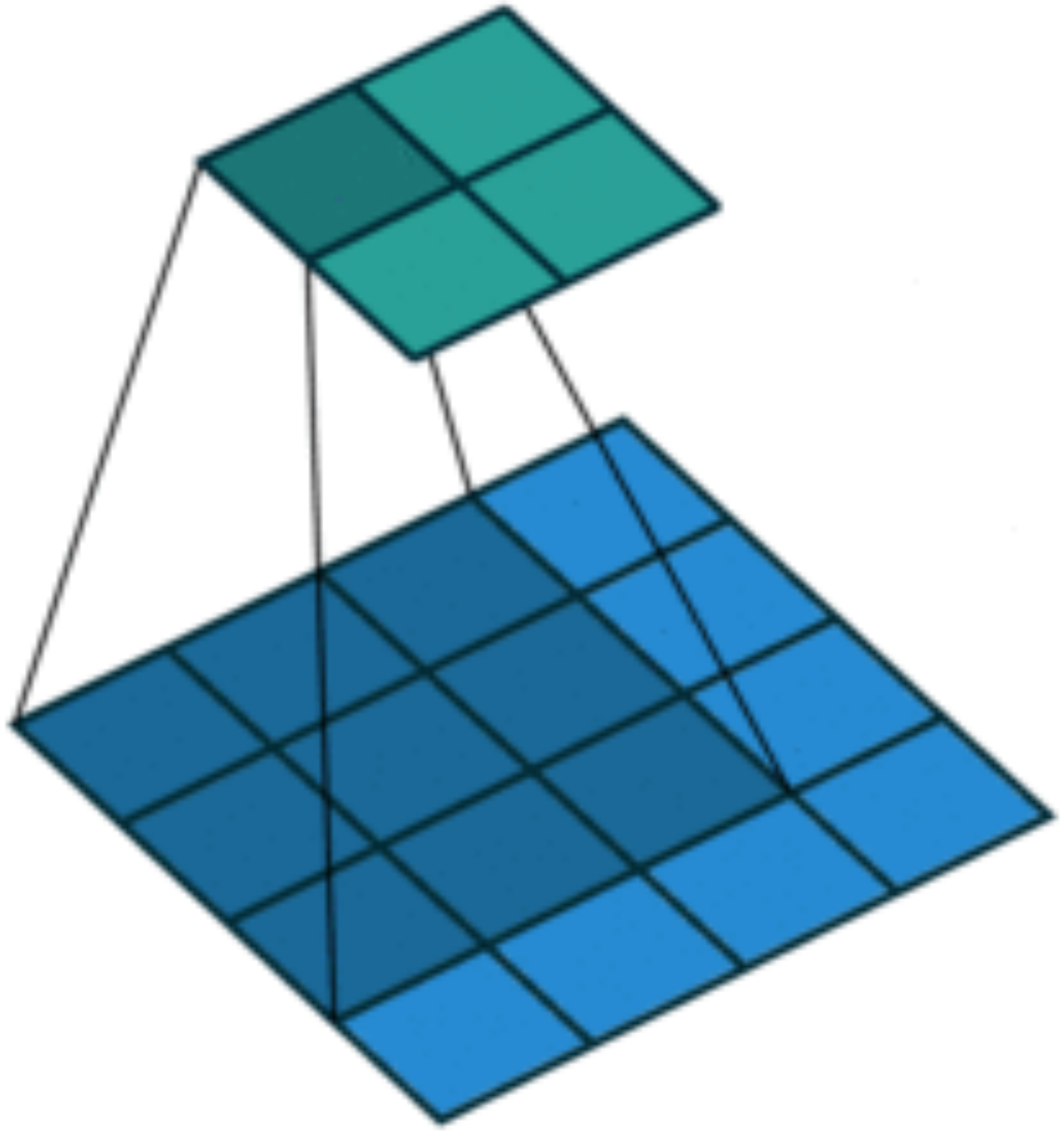


Review: 2-D Max Pooling

- Returns the maximal value in the sliding window

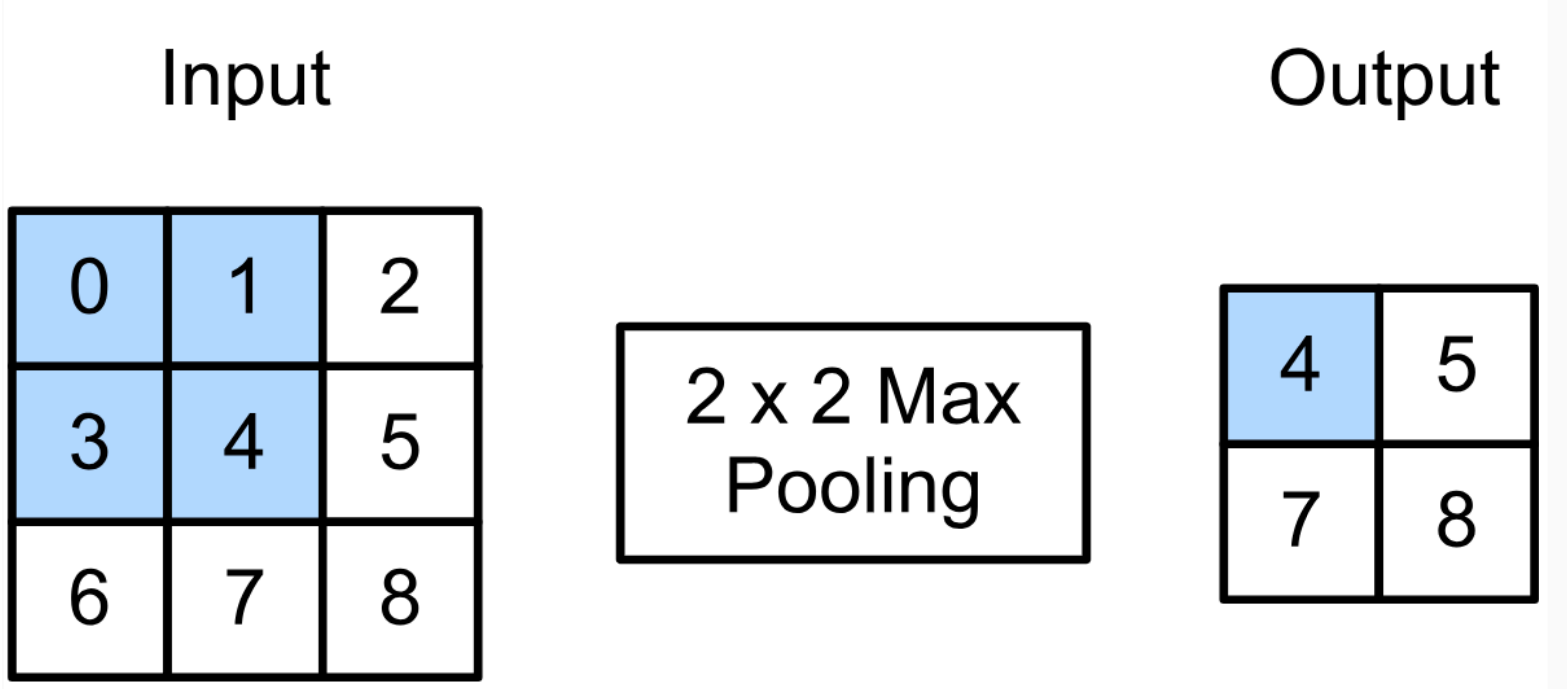


$$\max(0,1,3,4) = 4$$

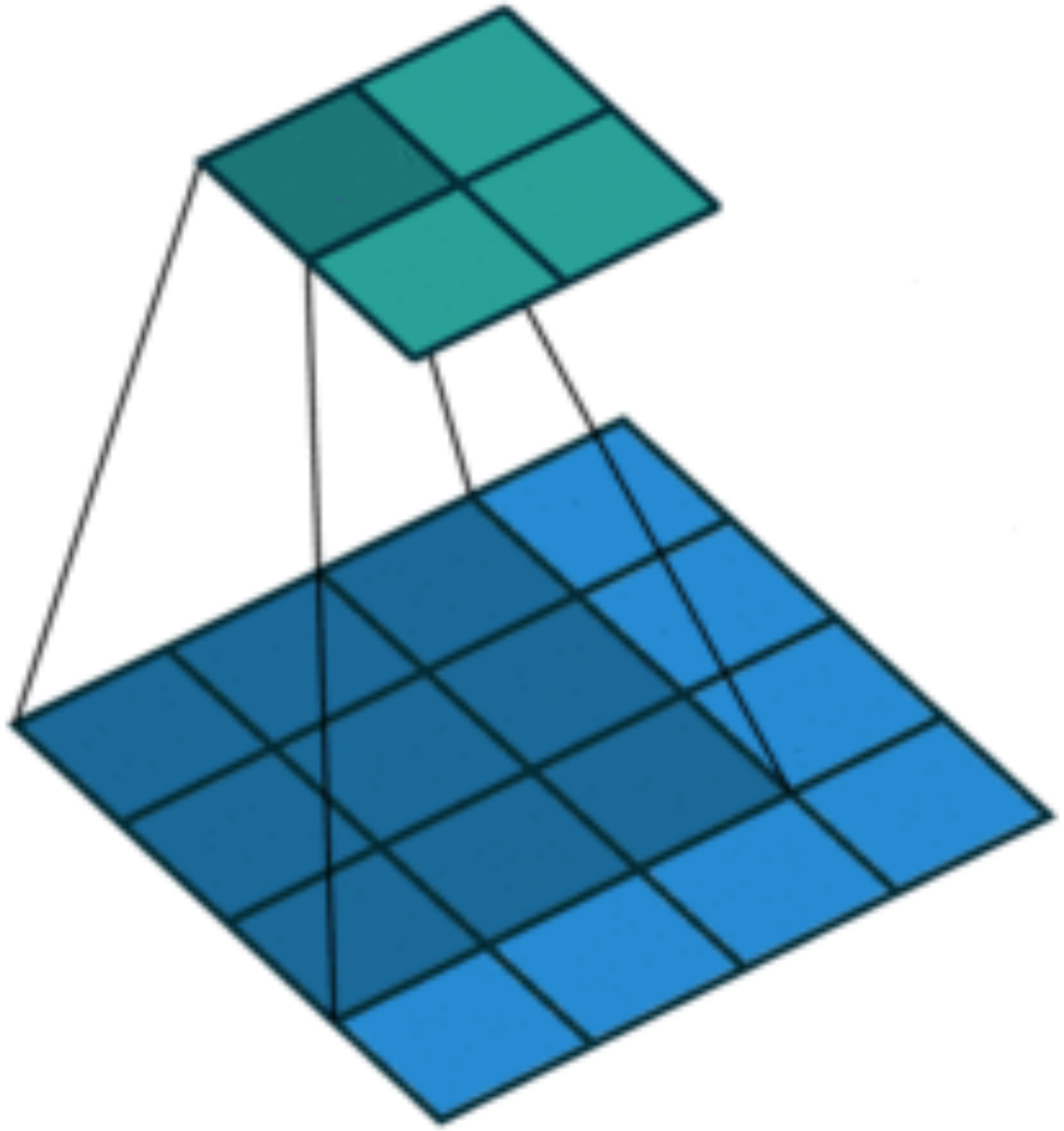


Review: 2-D Max Pooling

- Returns the maximal value in the sliding window



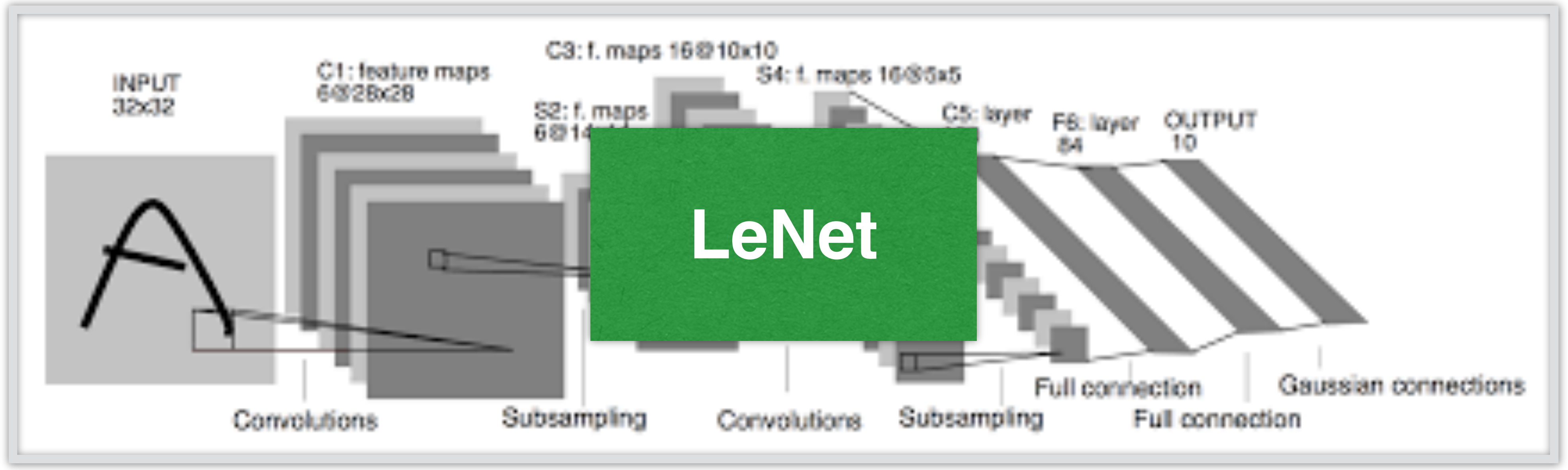
$$\max(0,1,3,4) = 4$$

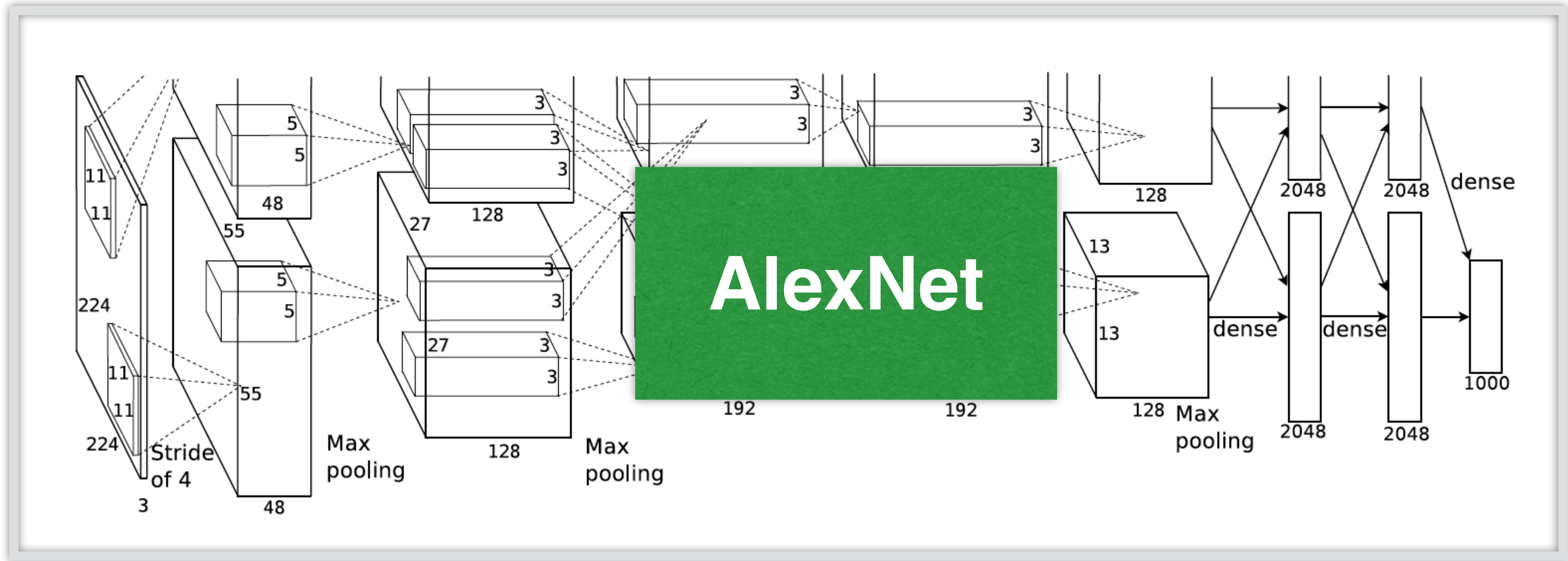


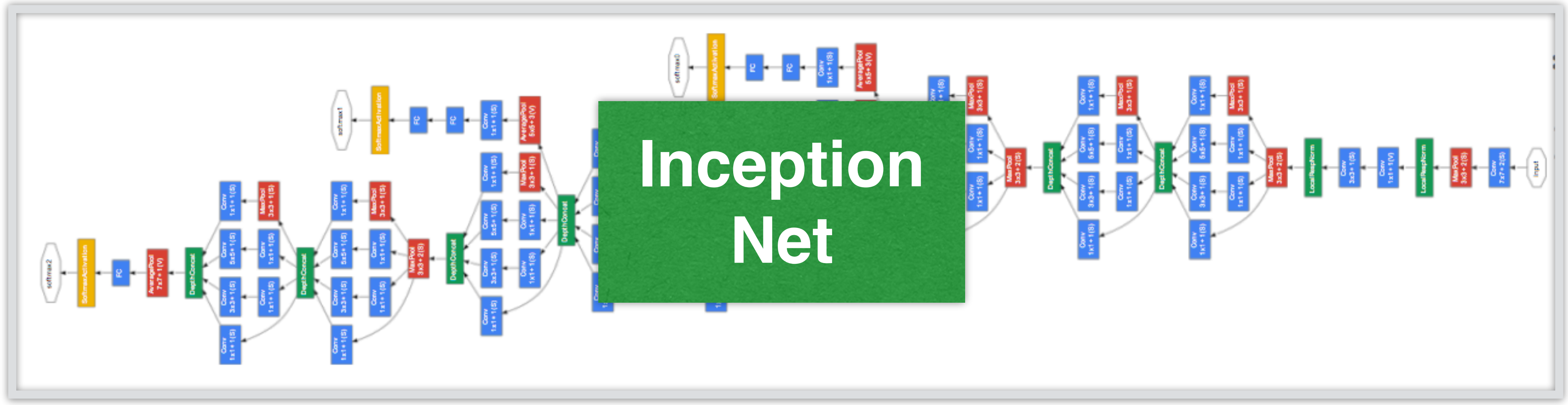
Convolutional Neural Networks

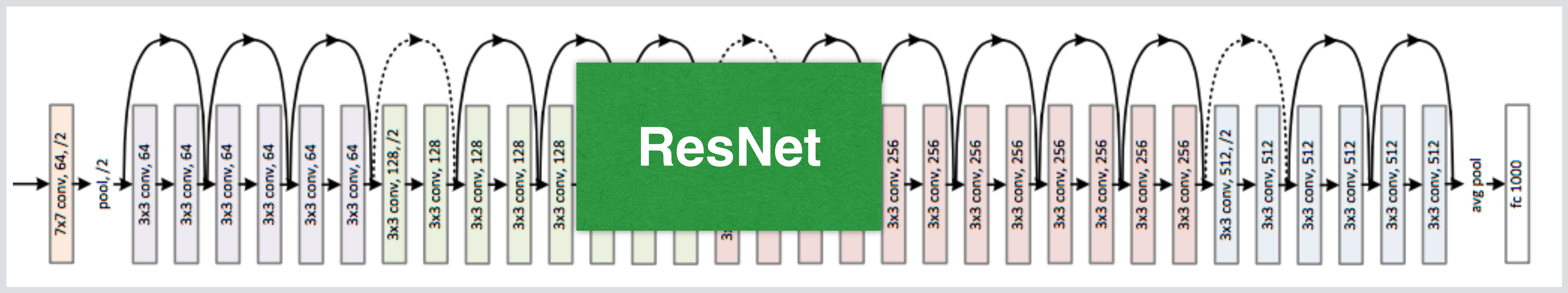
Evolution of neural net architectures

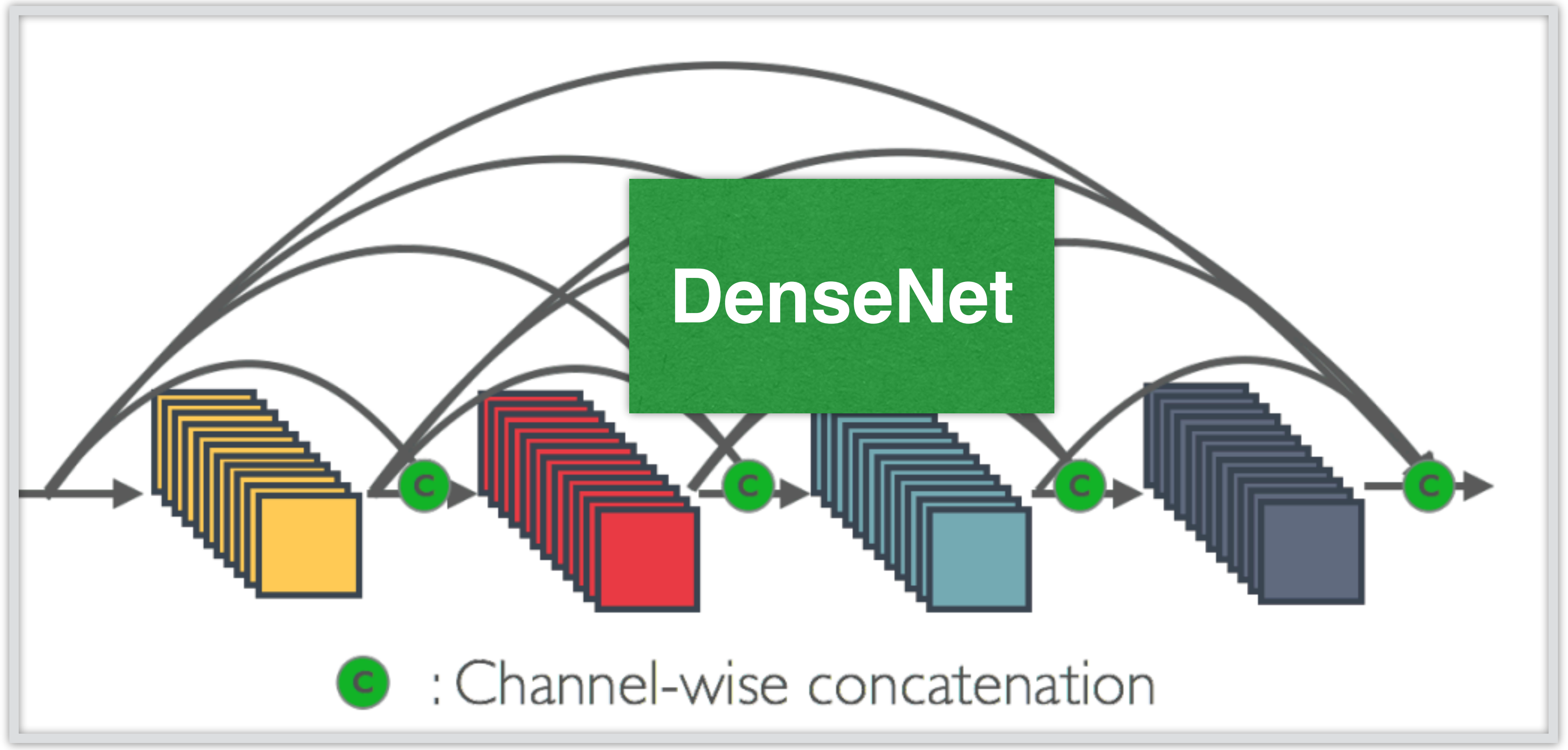
Evolution of neural net architectures



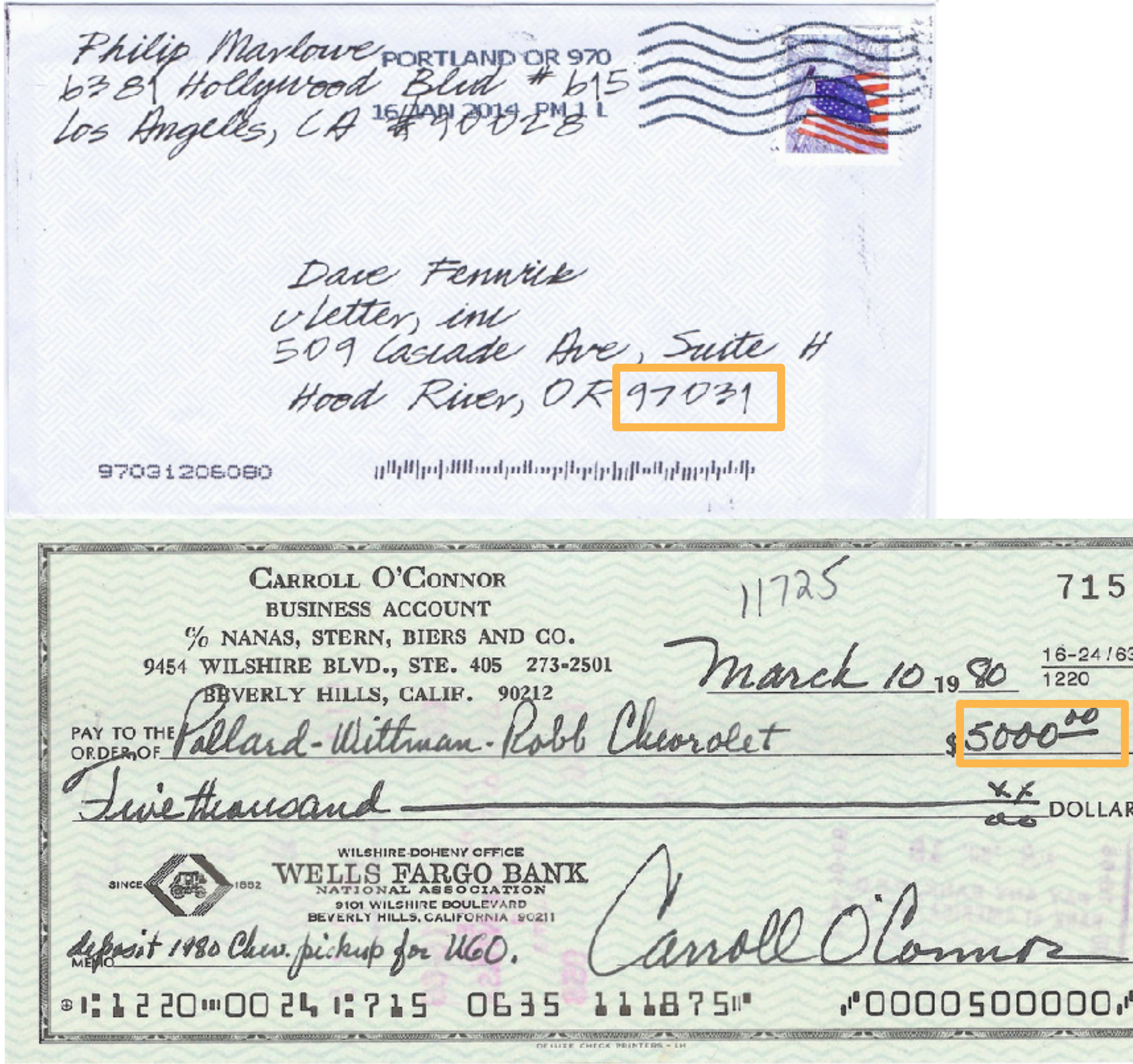






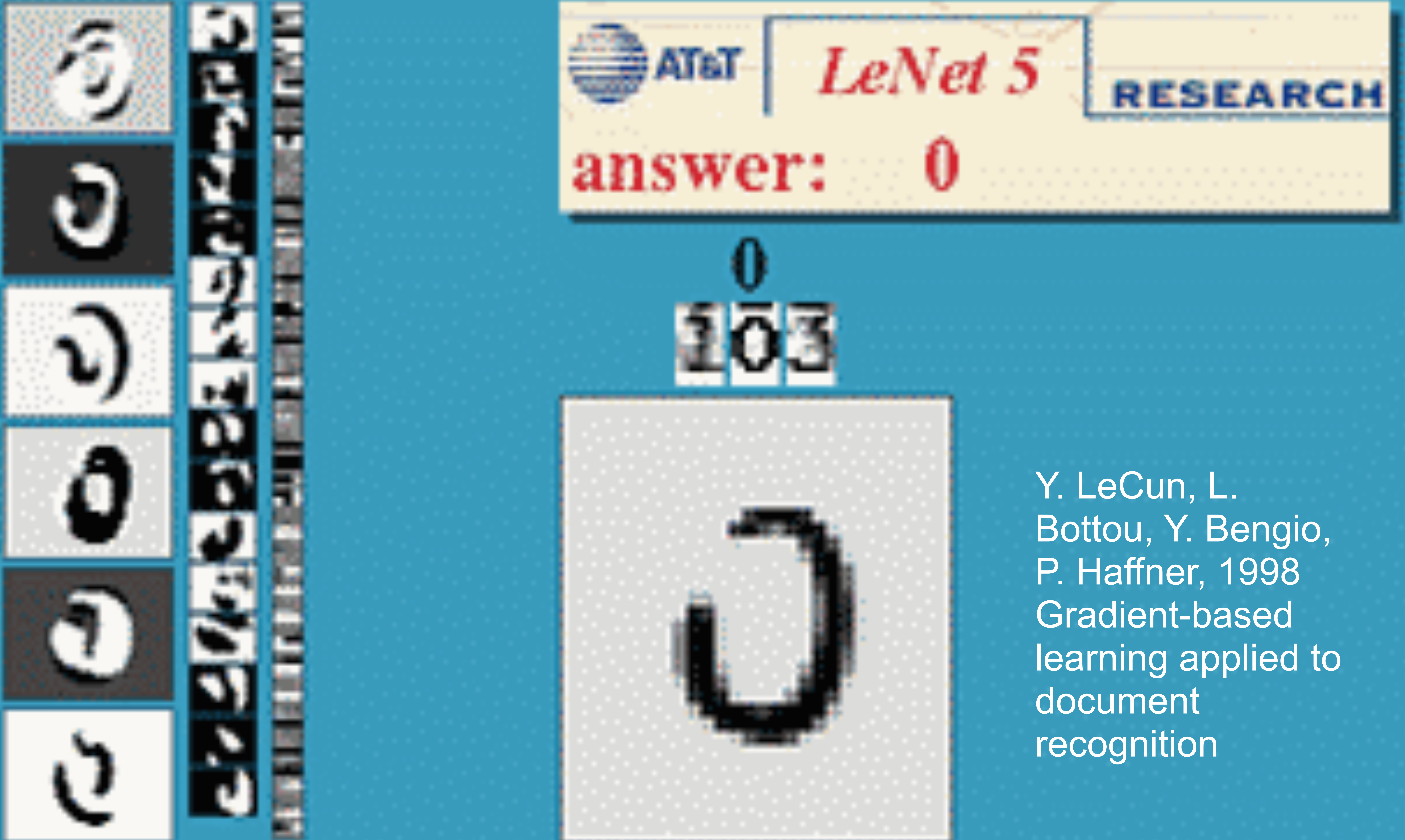


Handwritten Digit Recognition

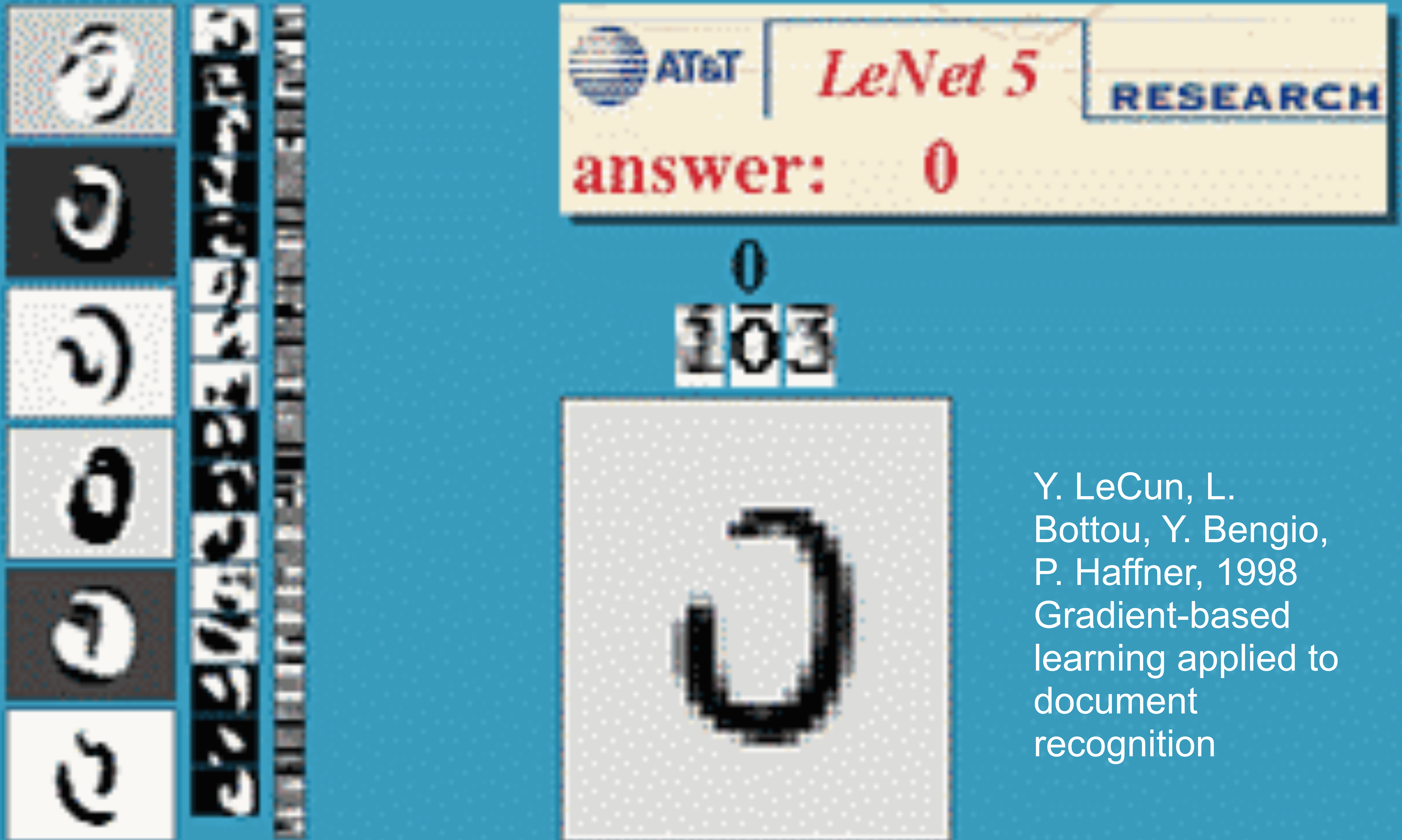


MNIST

- 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

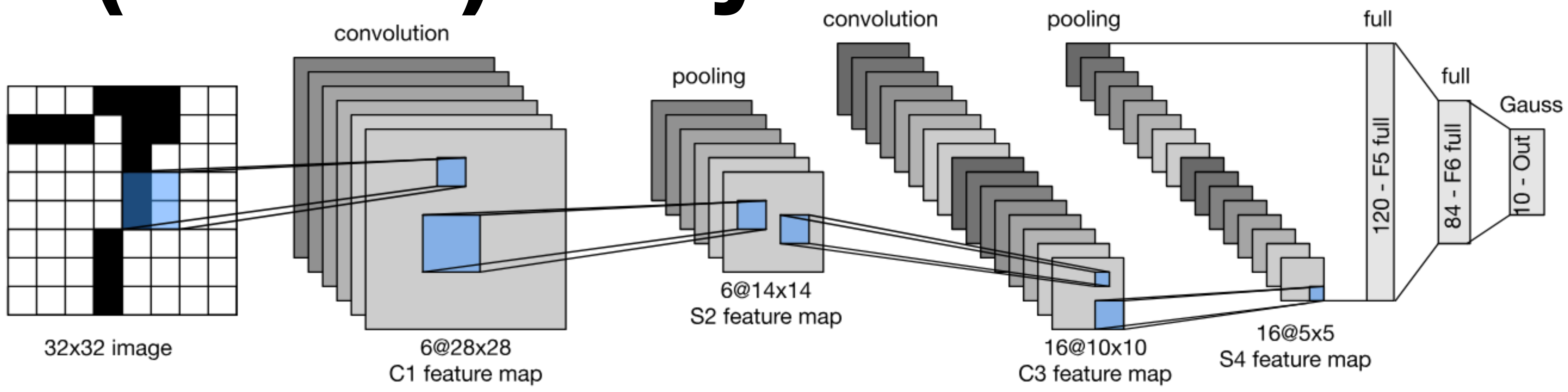


Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, 1998 Gradient-based learning applied to document recognition

LeNet Architecture (first conv nets)

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

LeNet(variant) in Pytorch



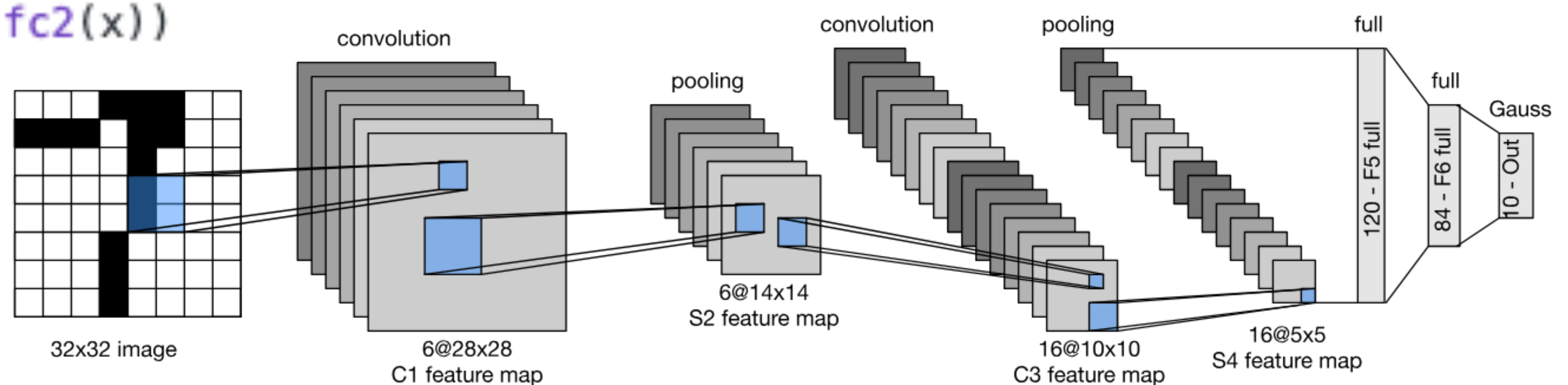

```python
def __init__(self):
    super(LeNet5, self).__init__()
    # Convolution (In LeNet-5, 32x32 images are given as input. Hence padding of 2 is done below)
    self.conv1 = torch.nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5, stride=1, padding=2, bias=True)
    # Max-pooling
    self.max_pool_1 = 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 (column
    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(variant) in Pytorch



Deng et al. 2009

AlexNet

AlexNet

- AlexNet won ImageNet competition in 2012

AlexNet

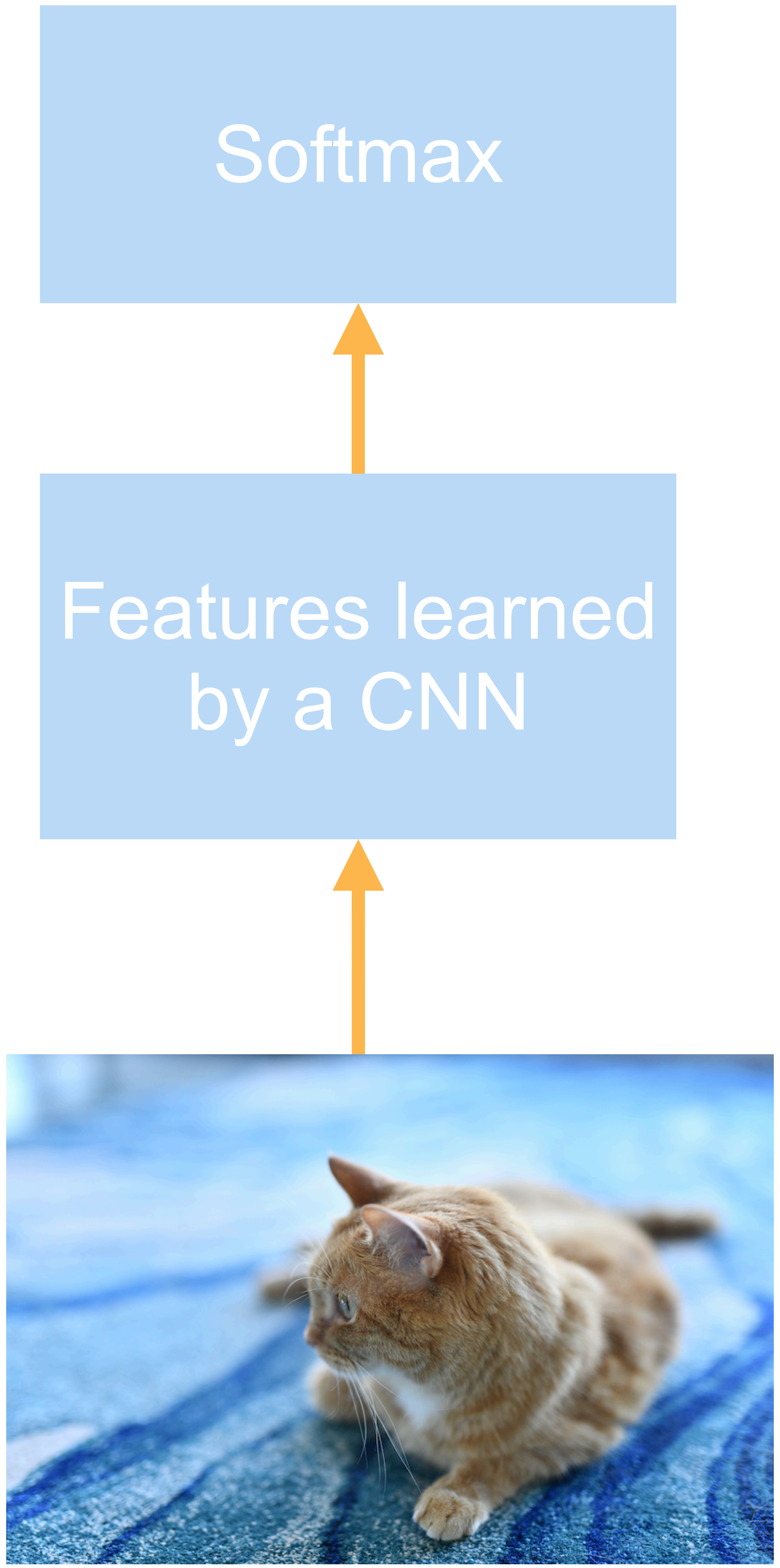
- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet

AlexNet

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

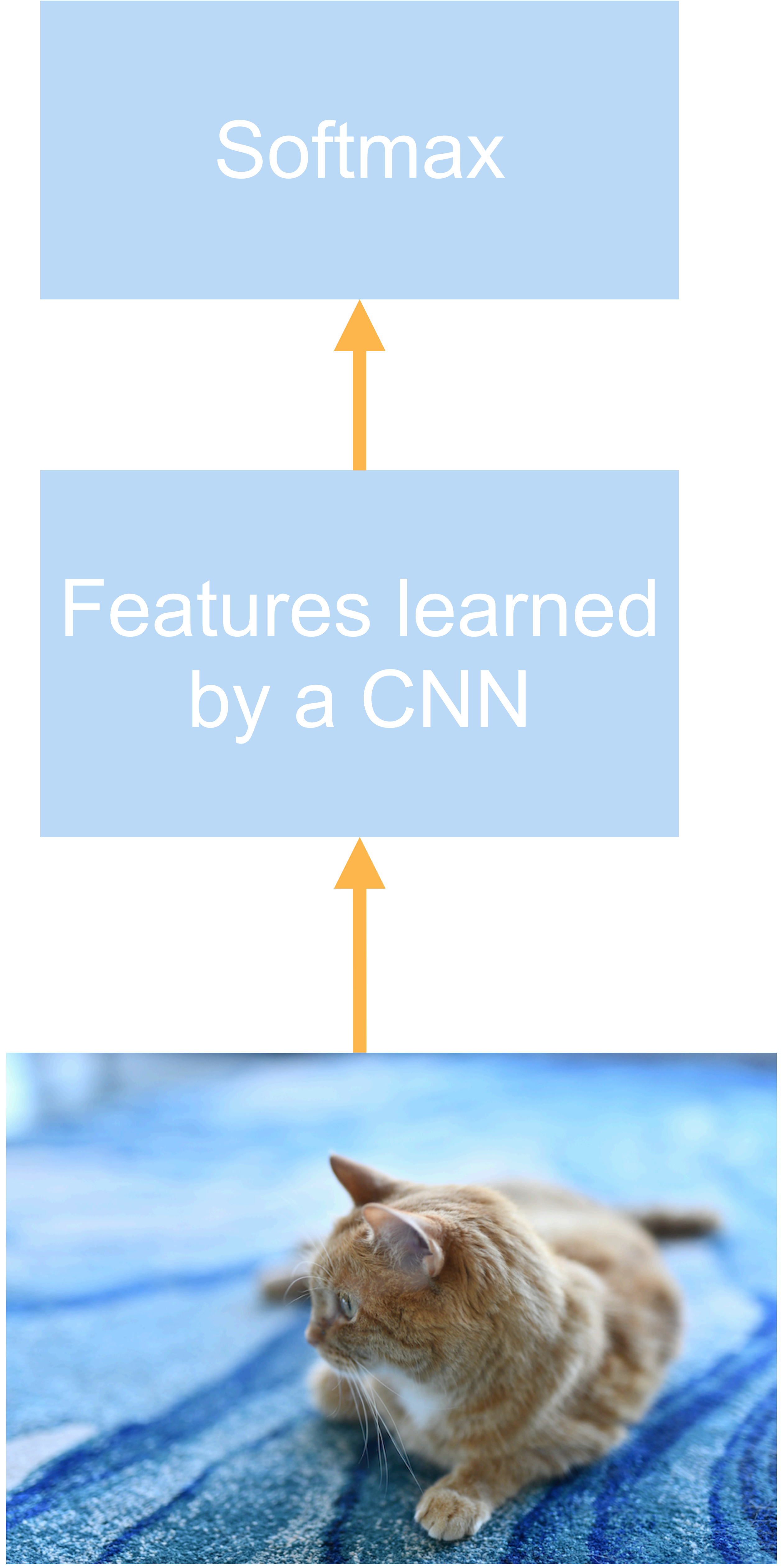
AlexNet

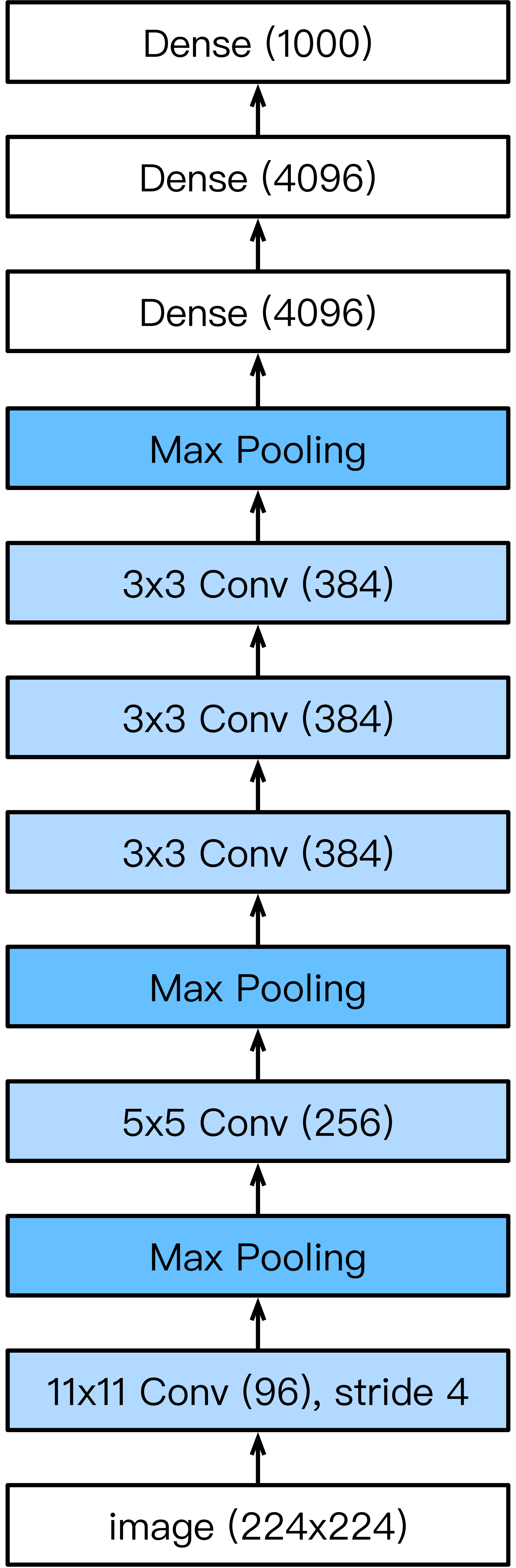
- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Paradigm shift for computer vision



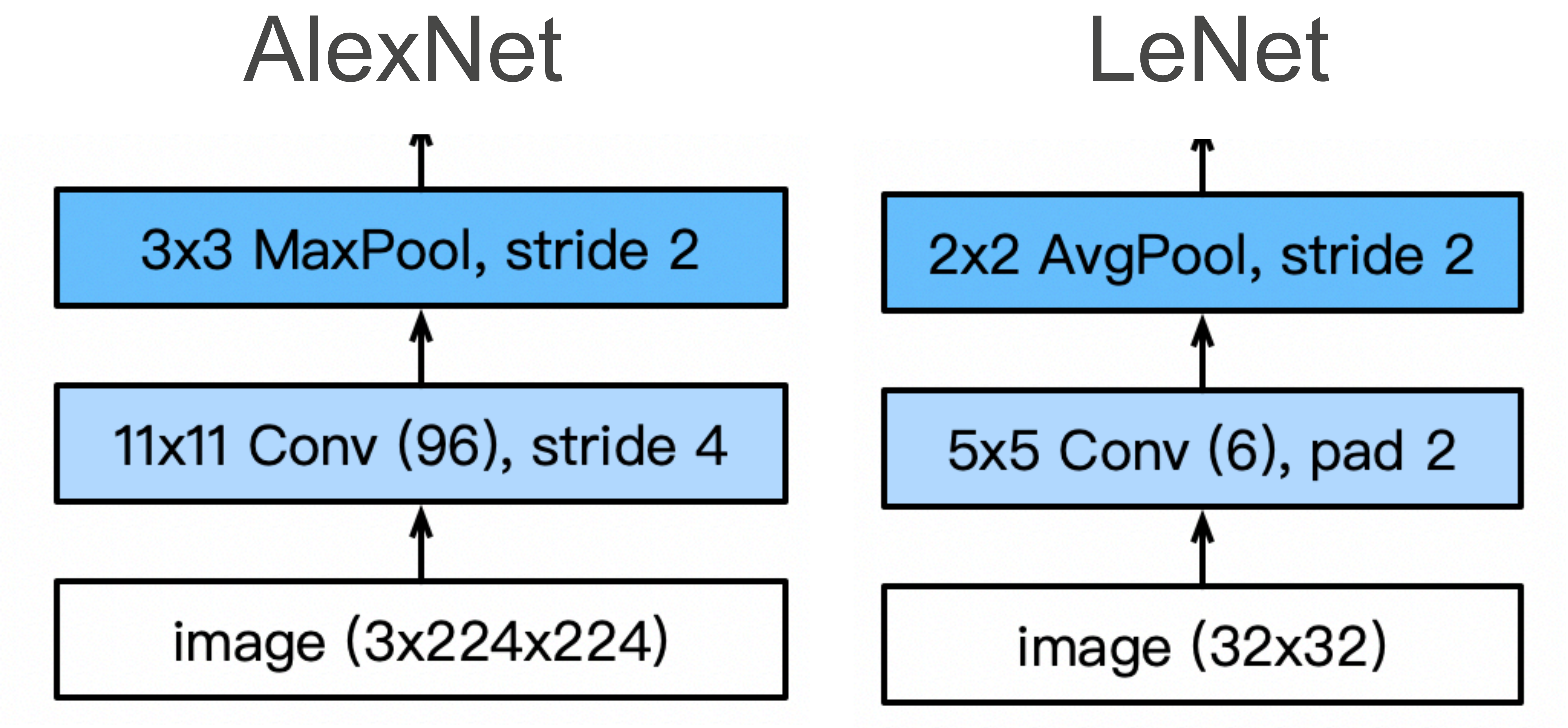
AlexNet

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



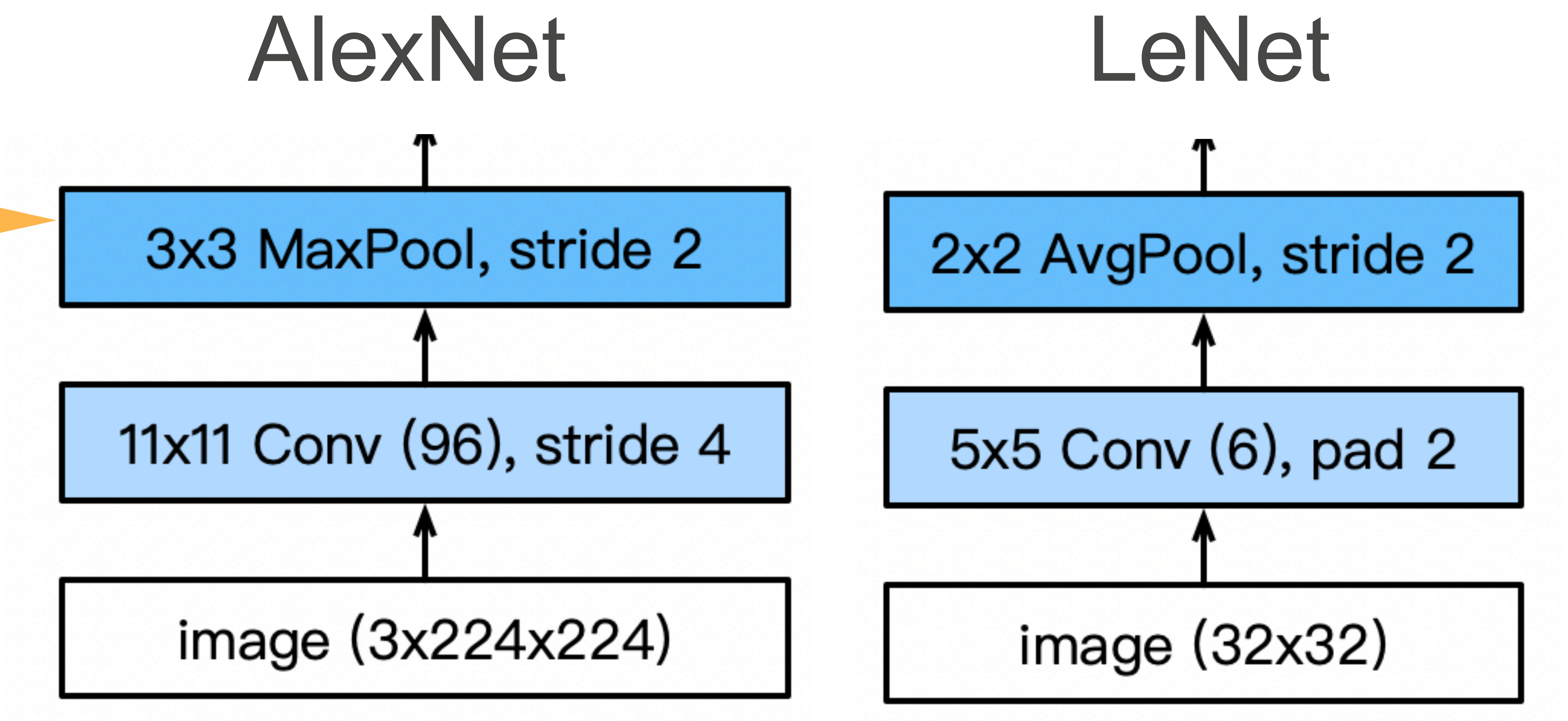


AlexNet Architecture



AlexNet Architecture

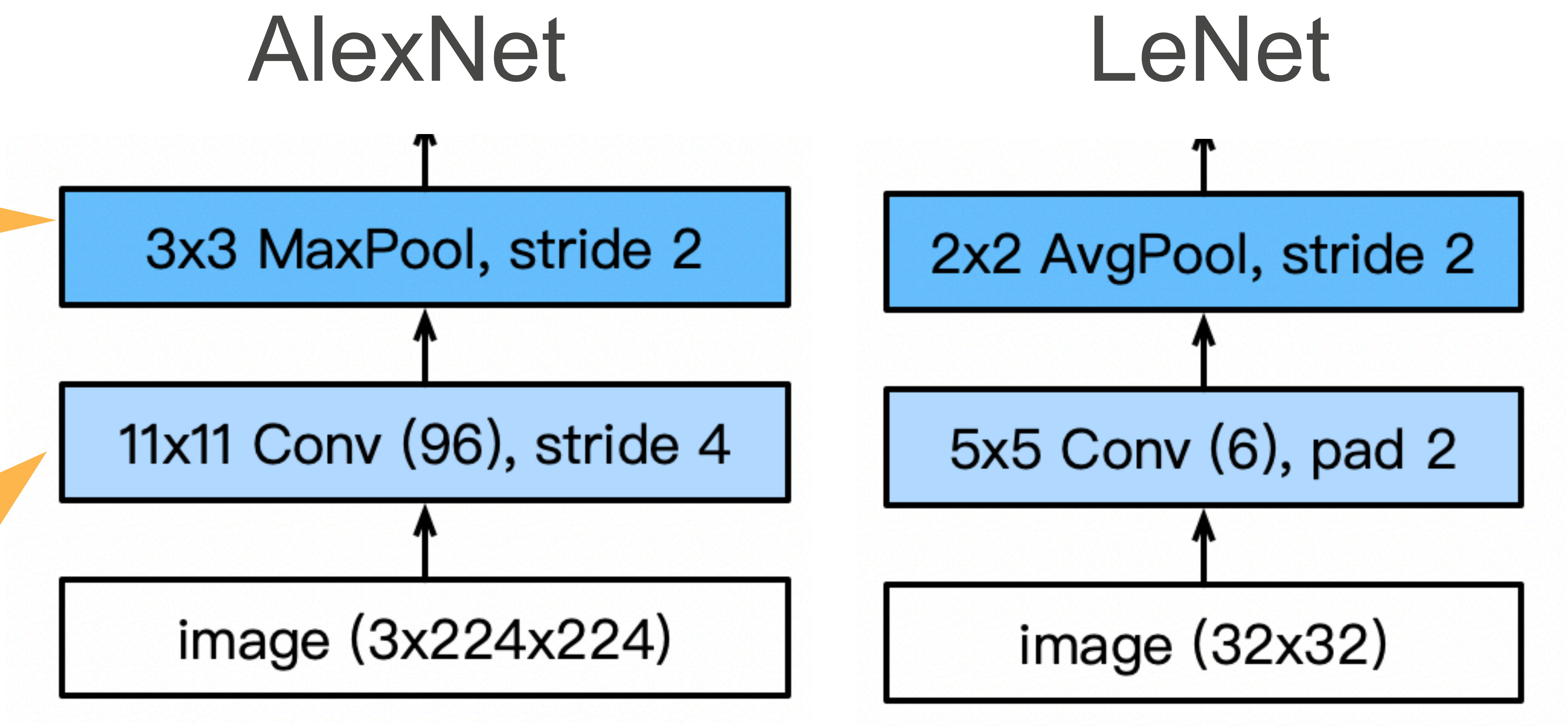
Larger pool size



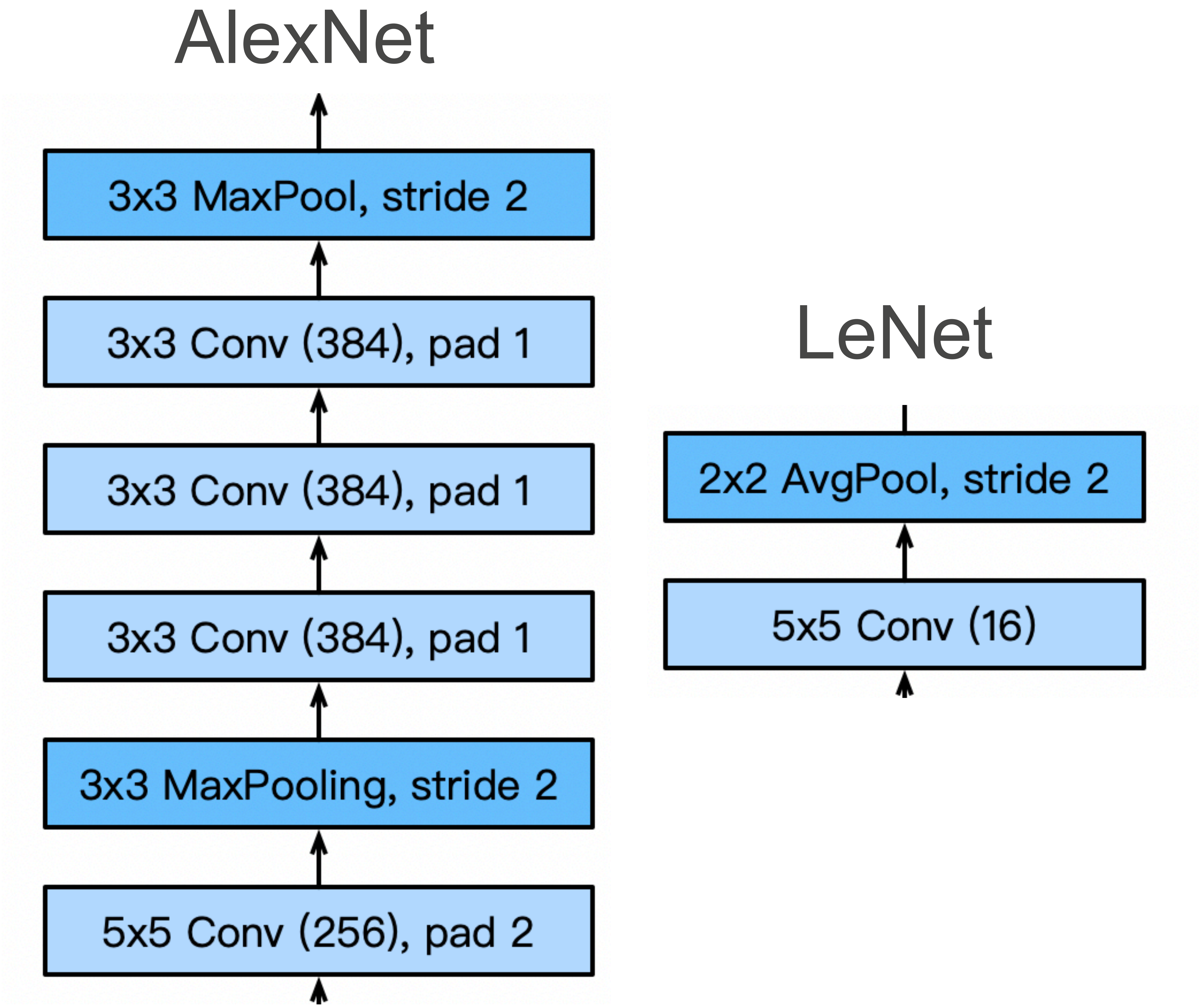
AlexNet Architecture

Larger pool size

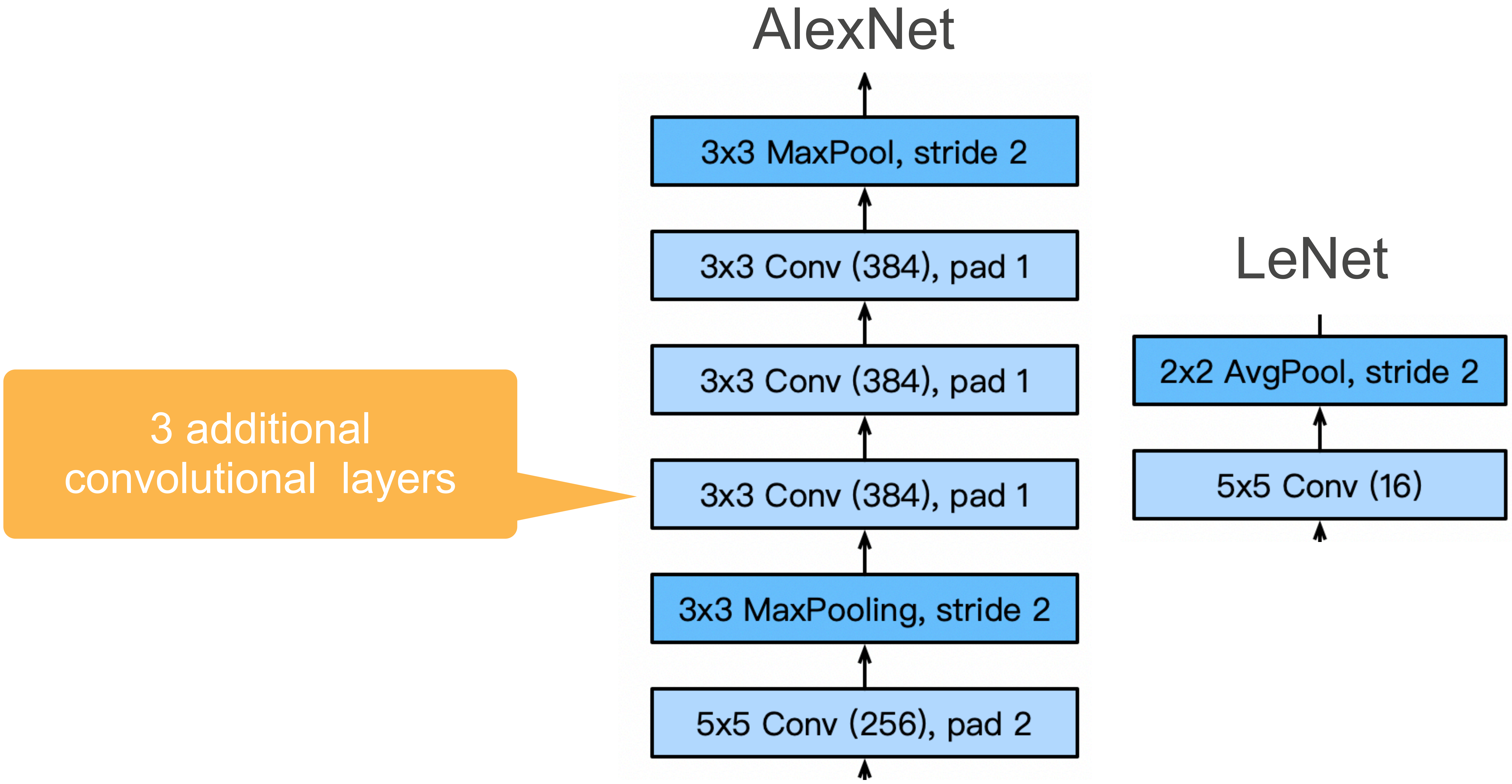
Larger kernel size, stride because of the increased image size, and more output channels.



AlexNet Architecture



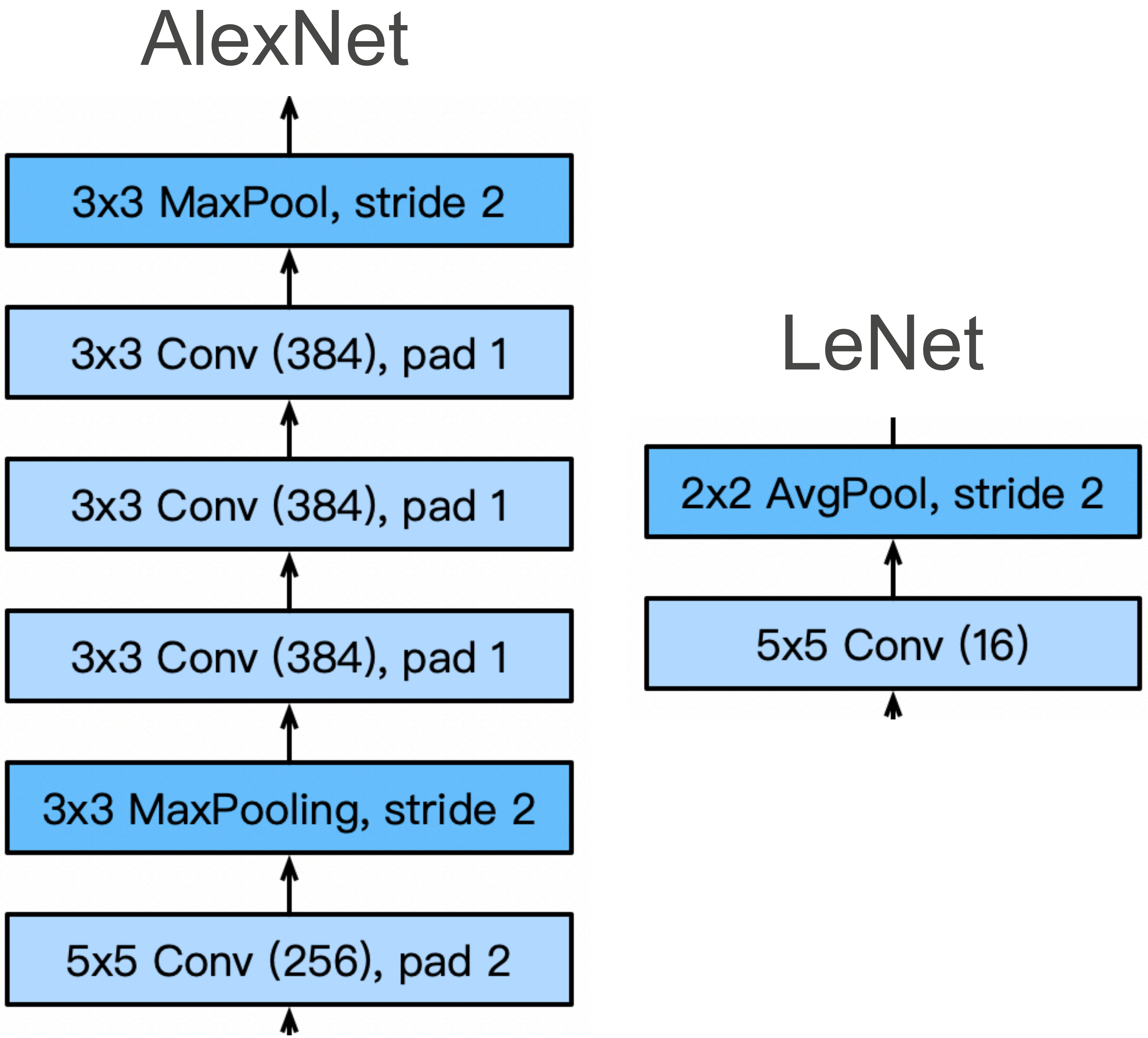
AlexNet Architecture



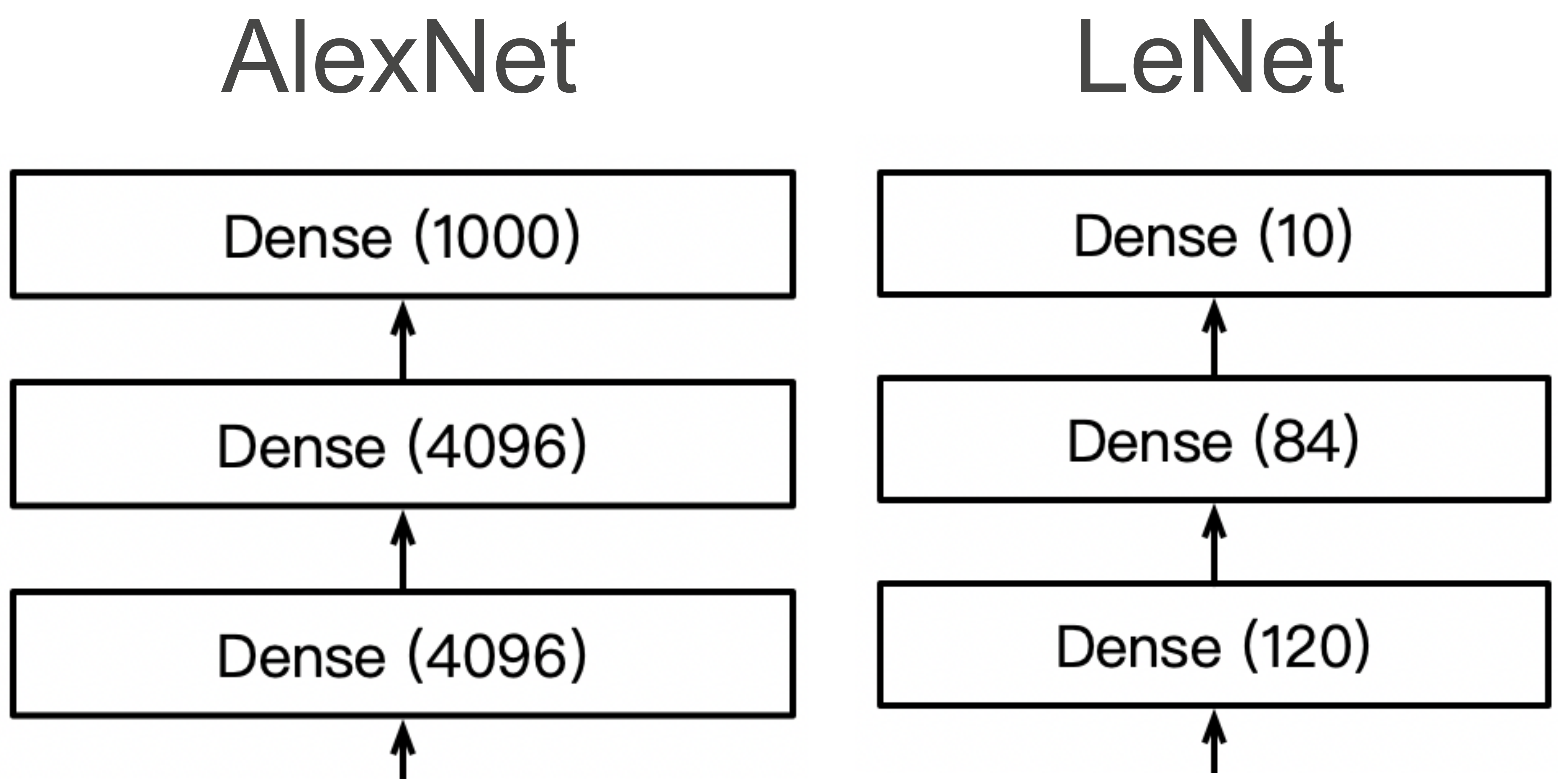
AlexNet Architecture

3 additional convolutional layers

More output channels.

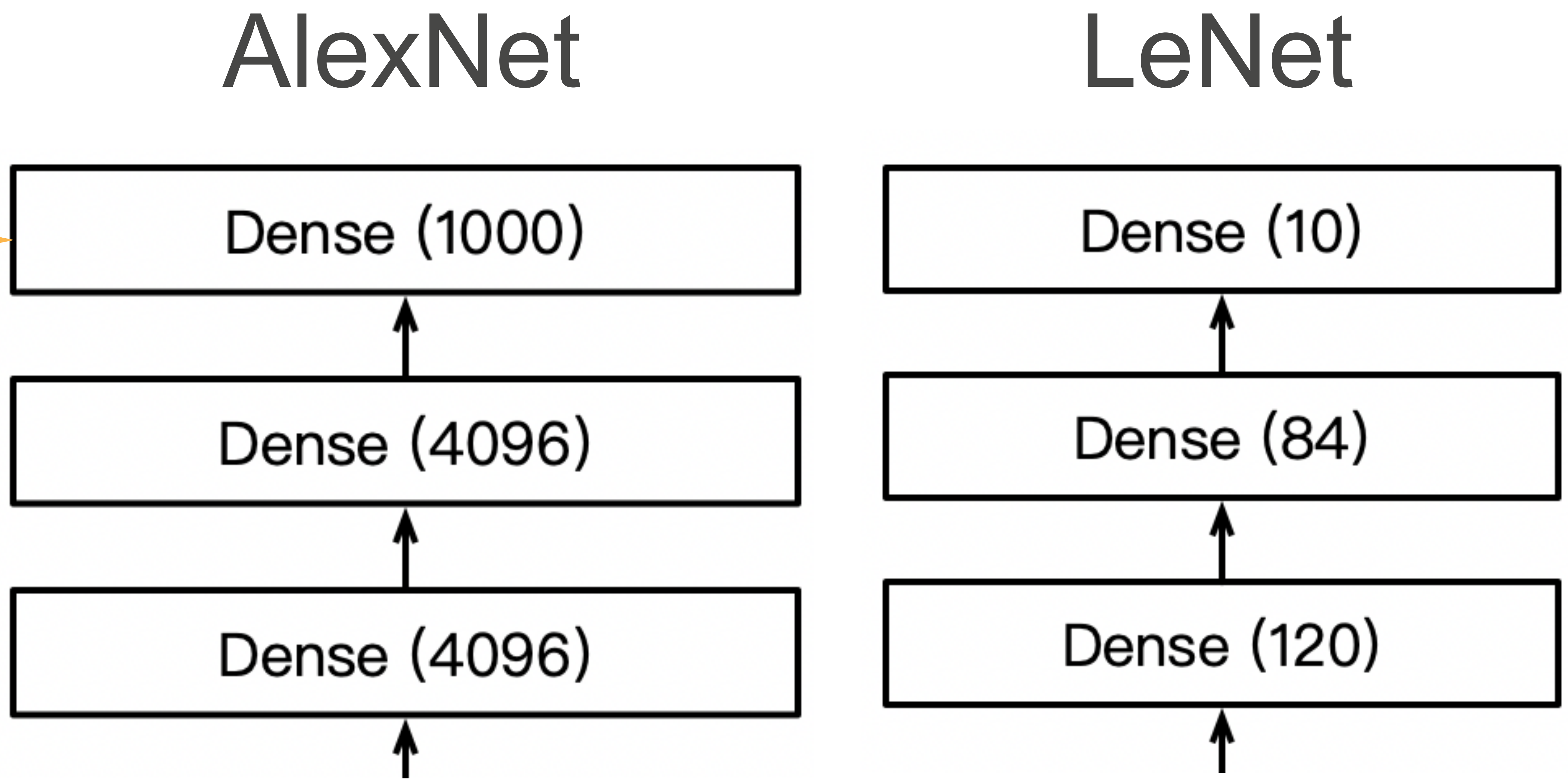


AlexNet Architecture

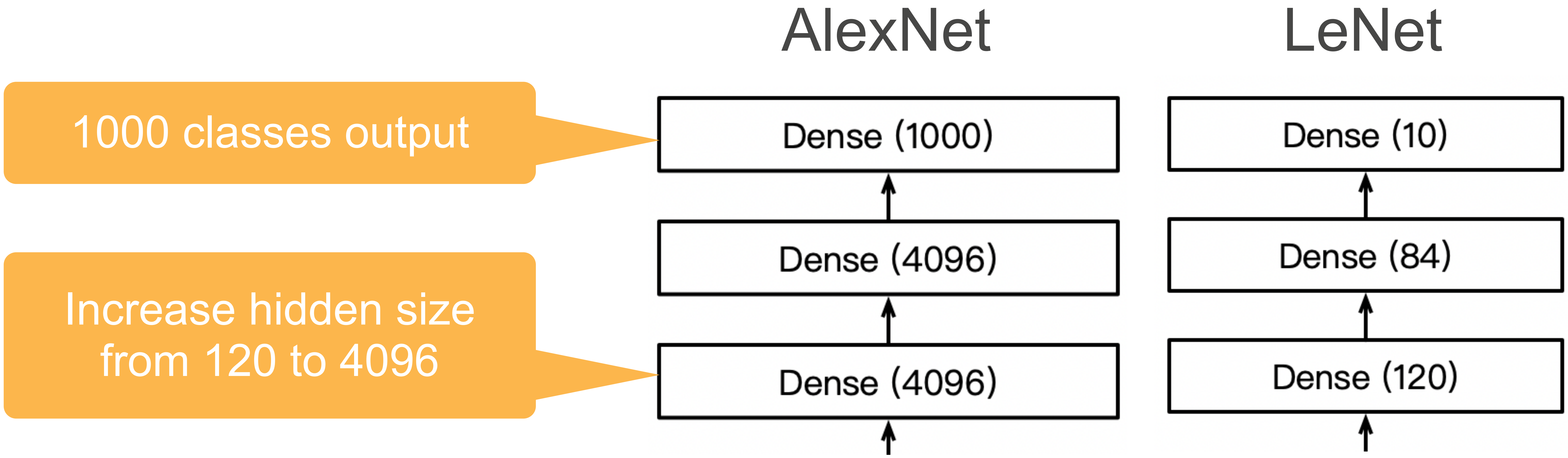


AlexNet Architecture

1000 classes output

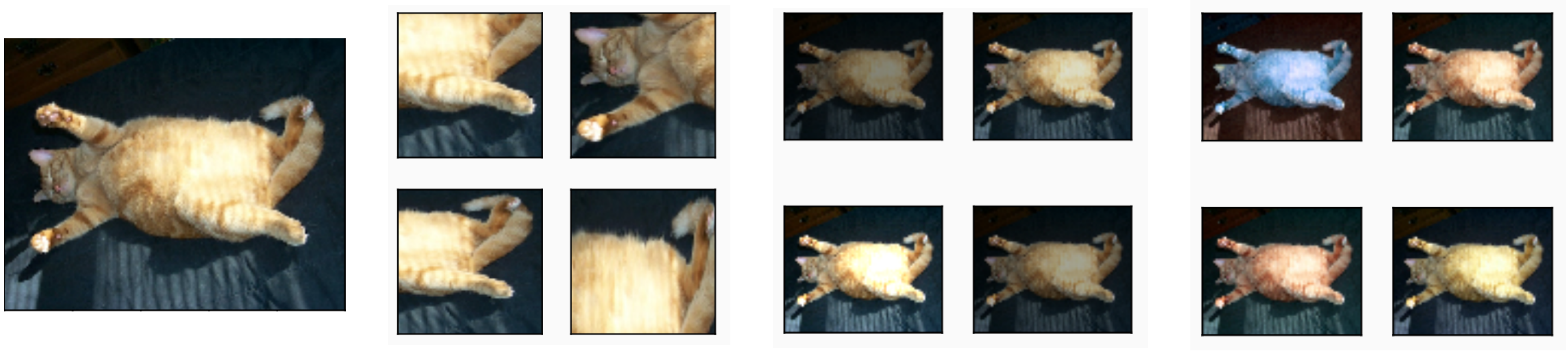


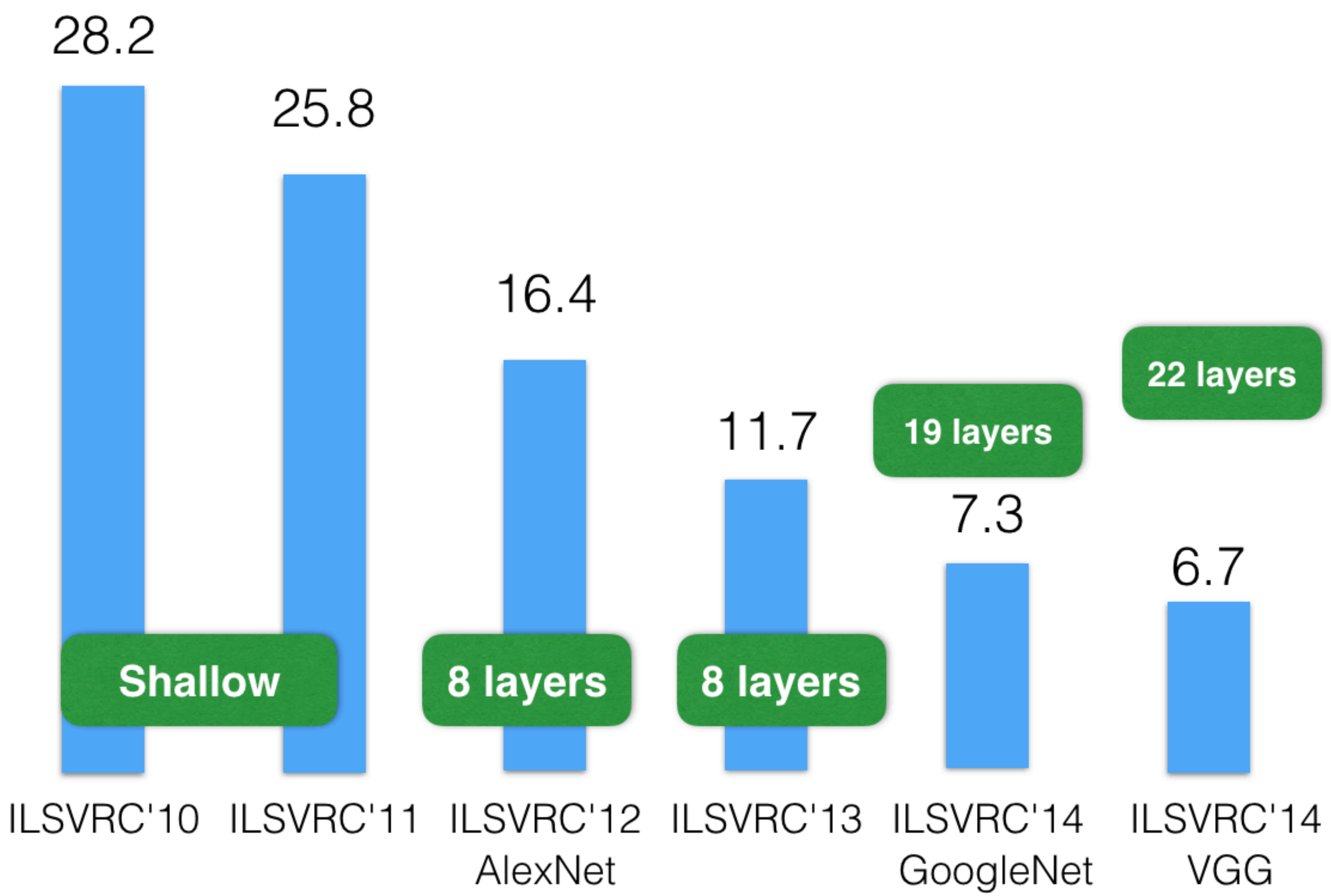
AlexNet Architecture



More Differences…

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





ImageNet Top-5 Classification Error (%)

Simple Idea: Add More Layers

- VGG: 19 layers. ResNet: 152 layers. Add more layers sufficient?

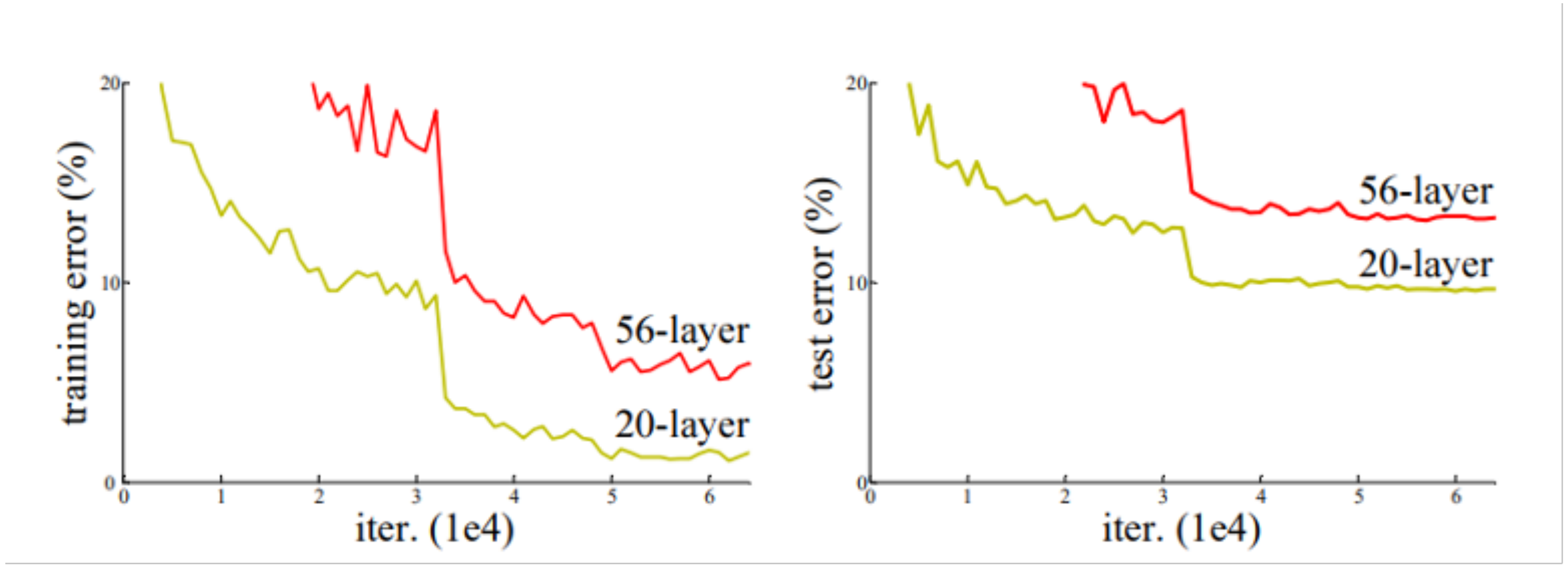
Simple Idea: Add More Layers

- VGG: 19 layers. ResNet: 152 layers. Add more layers sufficient?
- No! Some problems:
  - Vanishing gradients: more layers more likely
  - Instability: can’t guarantee we learn identity maps

Simple Idea: Add More Layers

- VGG: 19 layers. ResNet: 152 layers. Add more layers sufficient?
- No! Some problems:
  - Vanishing gradients: more layers more likely
  - Instability: can't guarantee we learn identity maps

Reflected in training error:



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

Depth Issues & Learning Identity

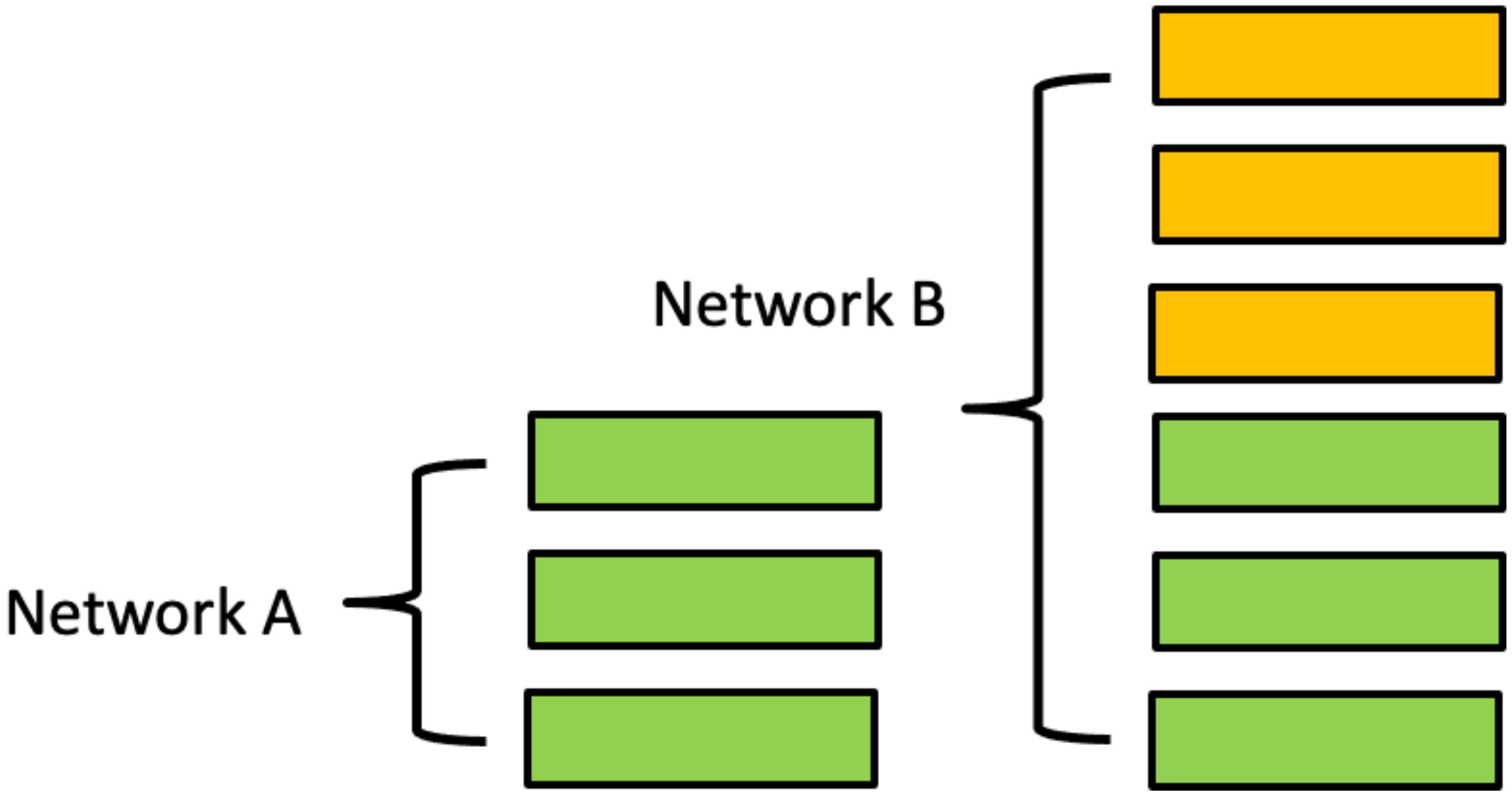
- Why would more layers result in worse performance

Depth Issues & Learning Identity

- Why would more layers result in worse performance
  - Same architecture, etc.

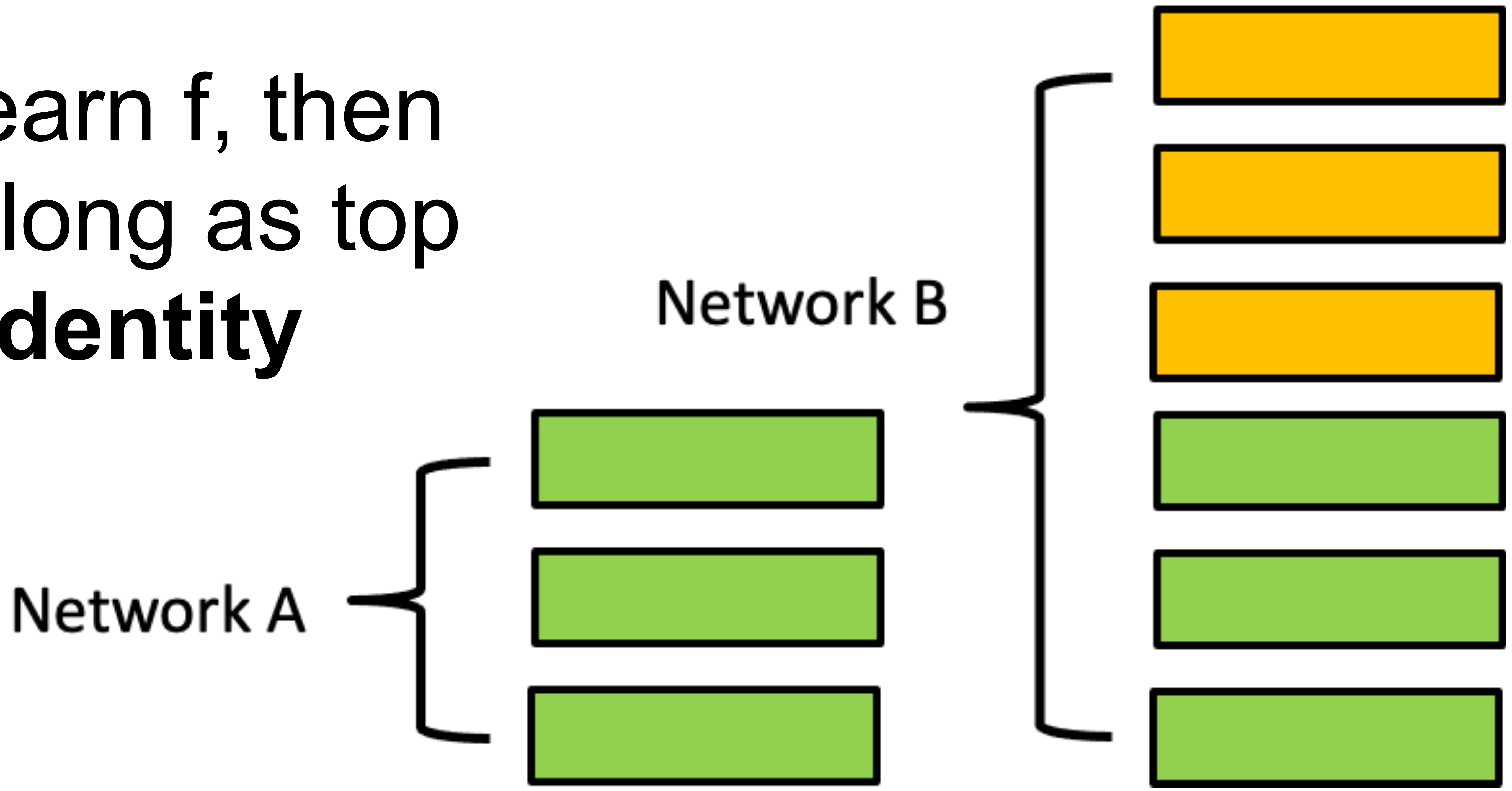
Depth Issues & Learning Identity

- Why would more layers result in worse performance
  - Same architecture, etc.



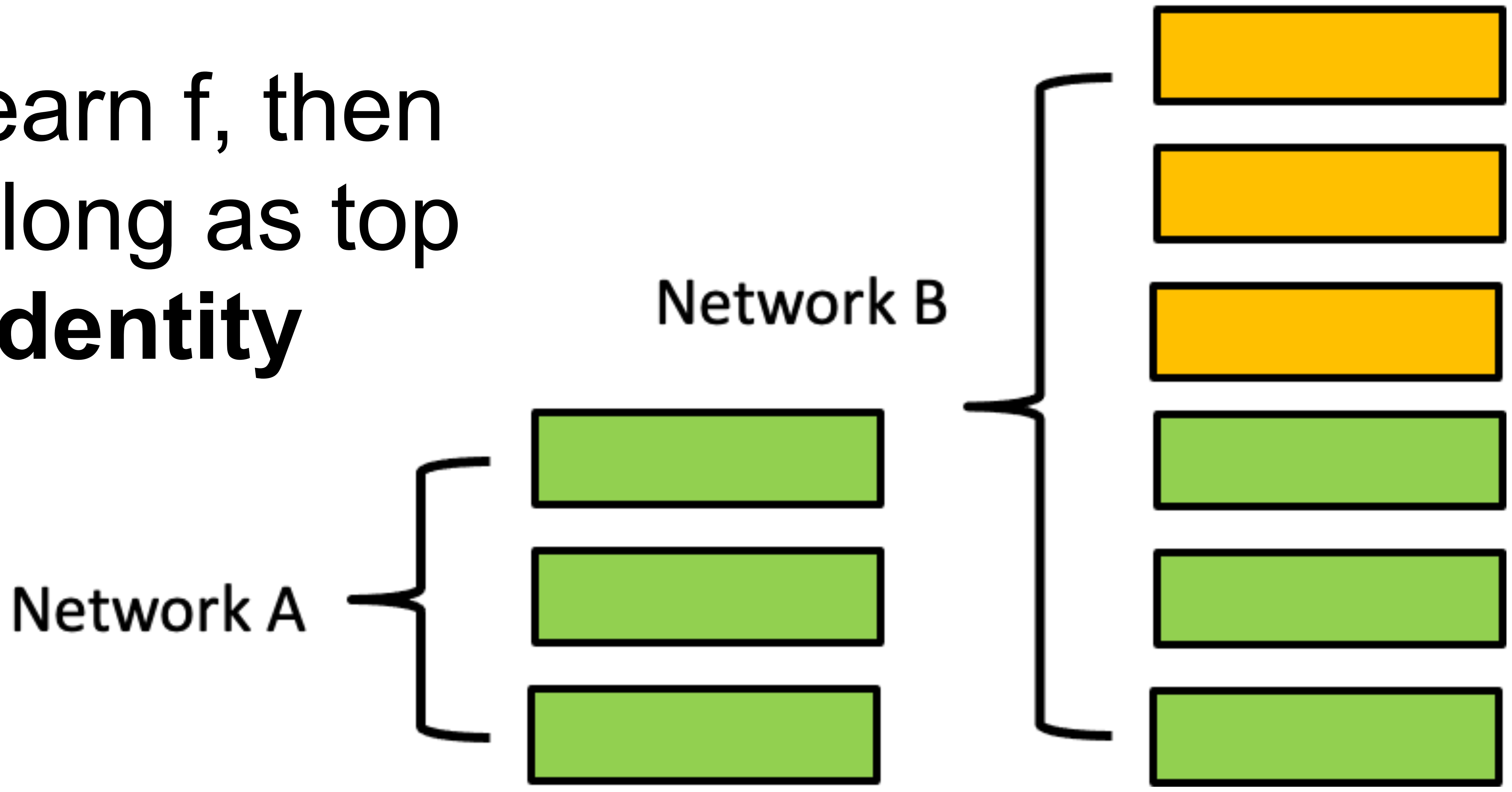
Depth Issues & Learning Identity

- 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**



Depth Issues & Learning Identity

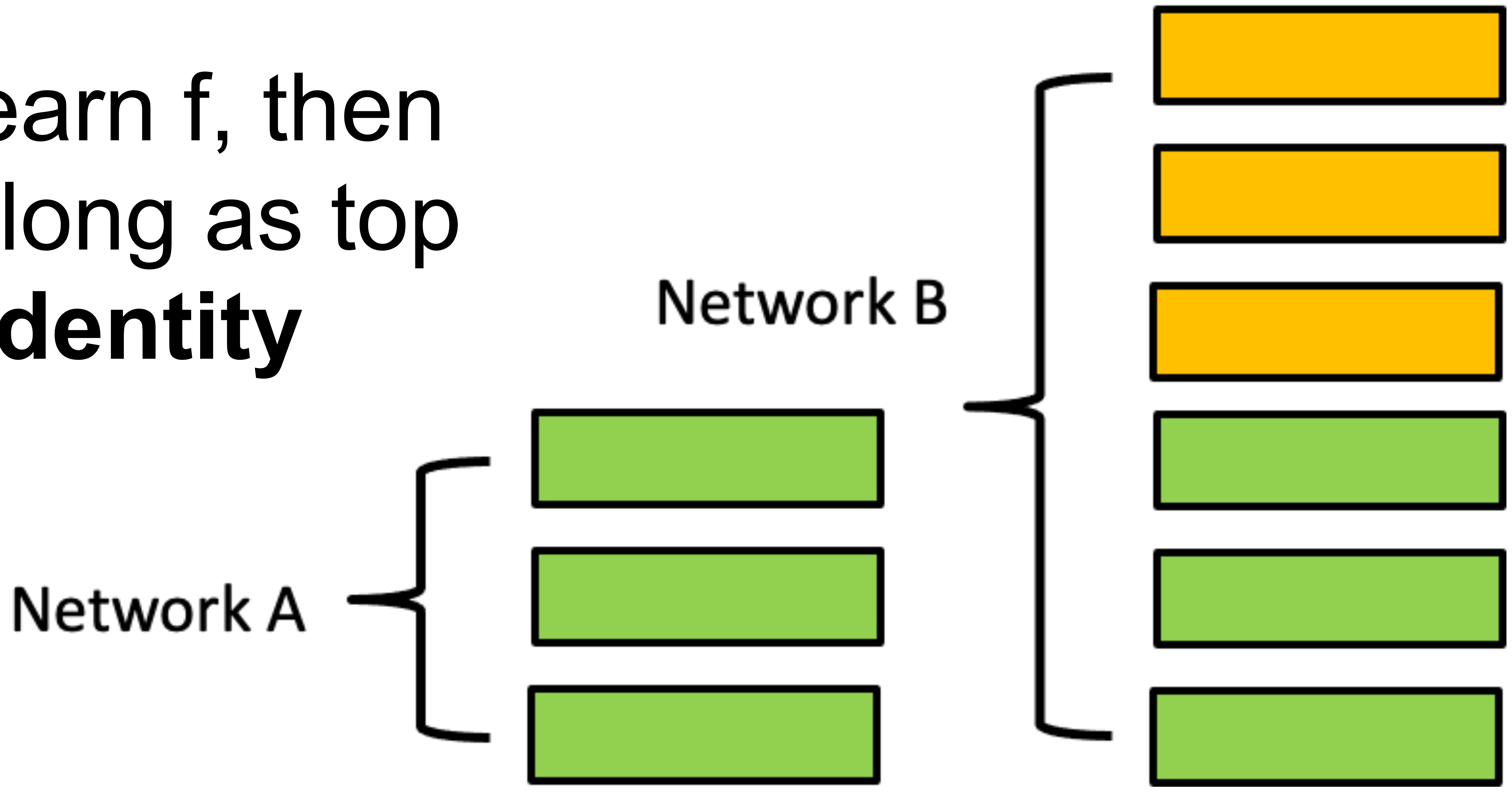
- Why would more layers result in worse performance
  - Same architecture, etc.
  - If the A can learn f, then so can B, as long as top layers learn **identity**



Q: can we learn identity here?

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**



Q: can we learn identity here?

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

Residual Connections

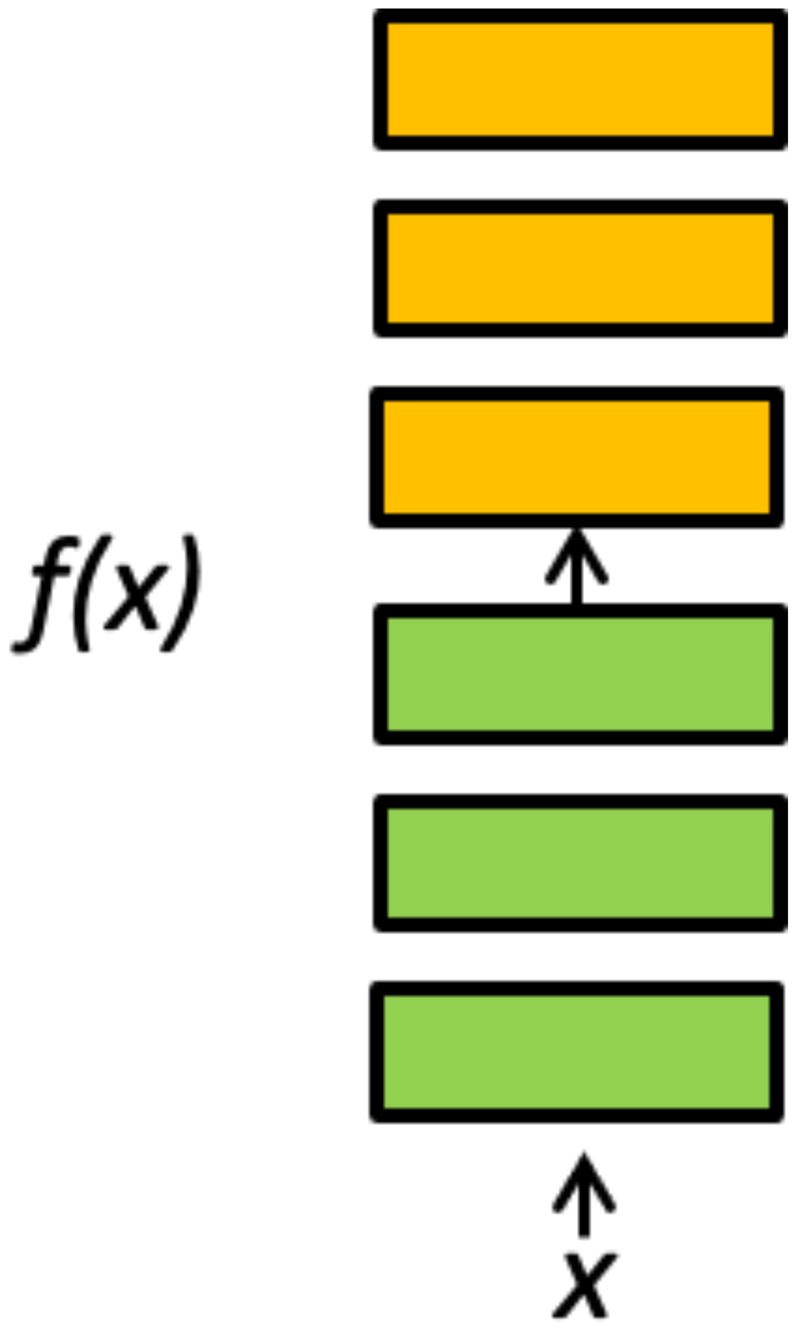
- **Idea:** identity might be hard to learn, but zero is easy!

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:

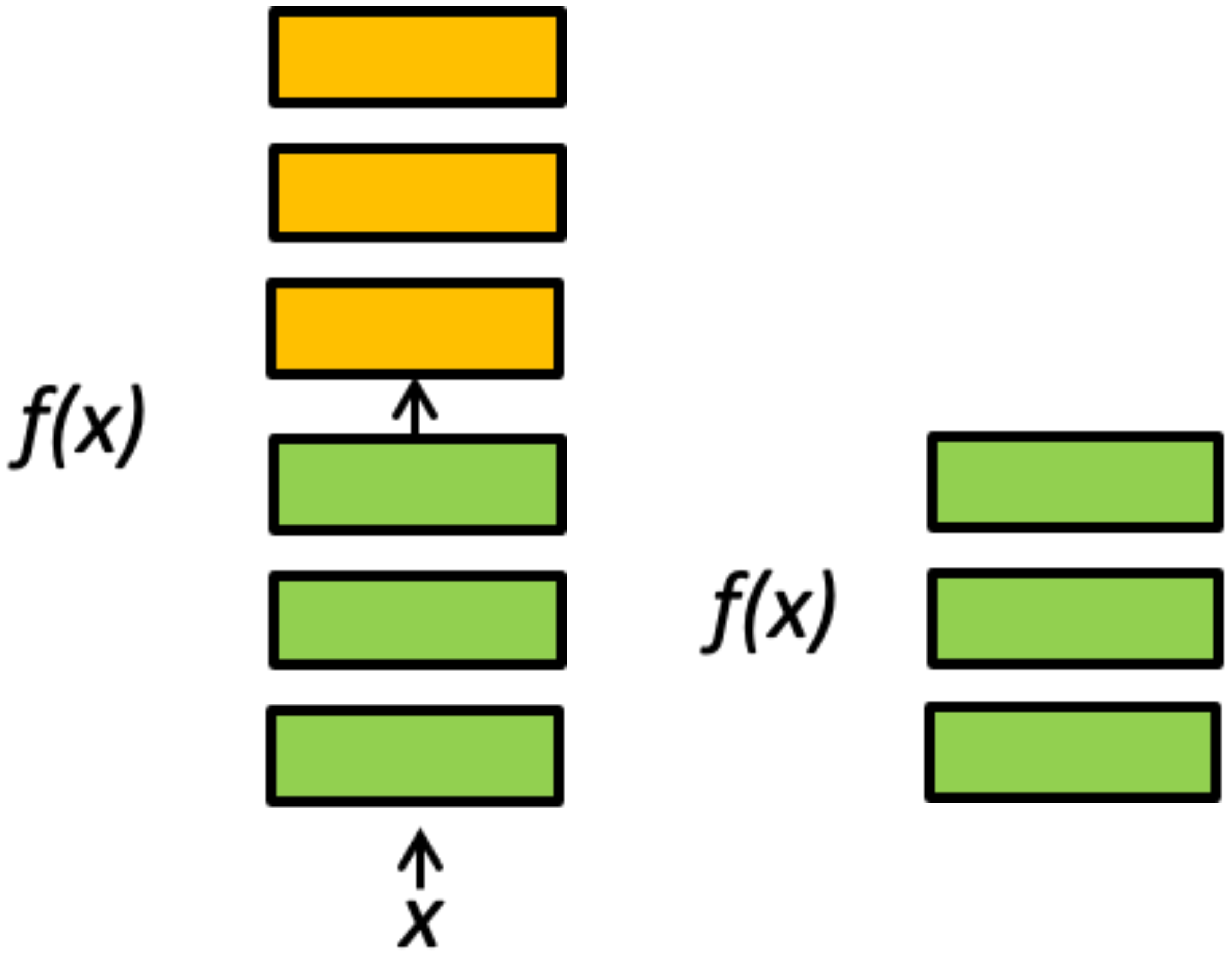
Residual Connections

- **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:



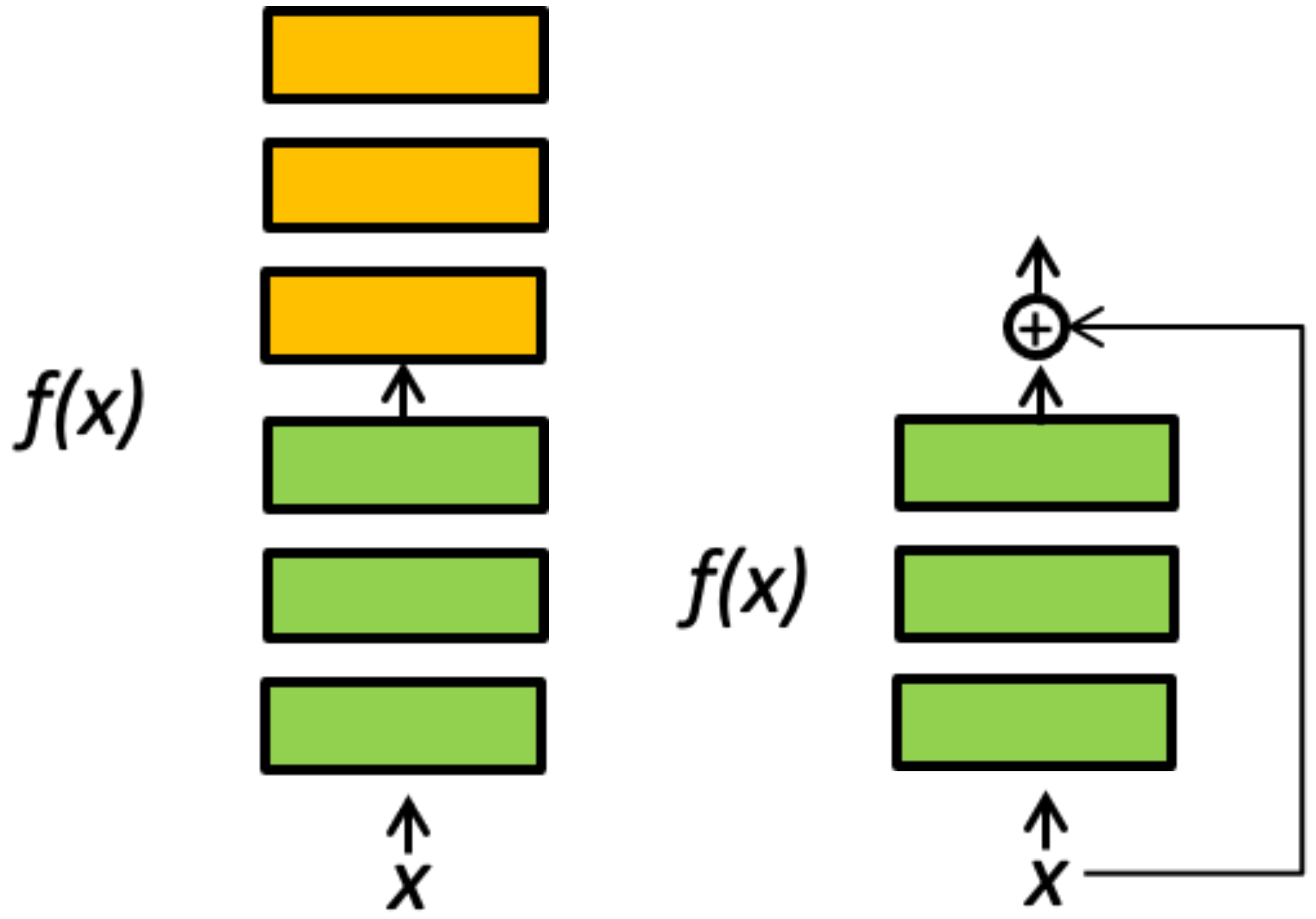
Residual Connections

- **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:



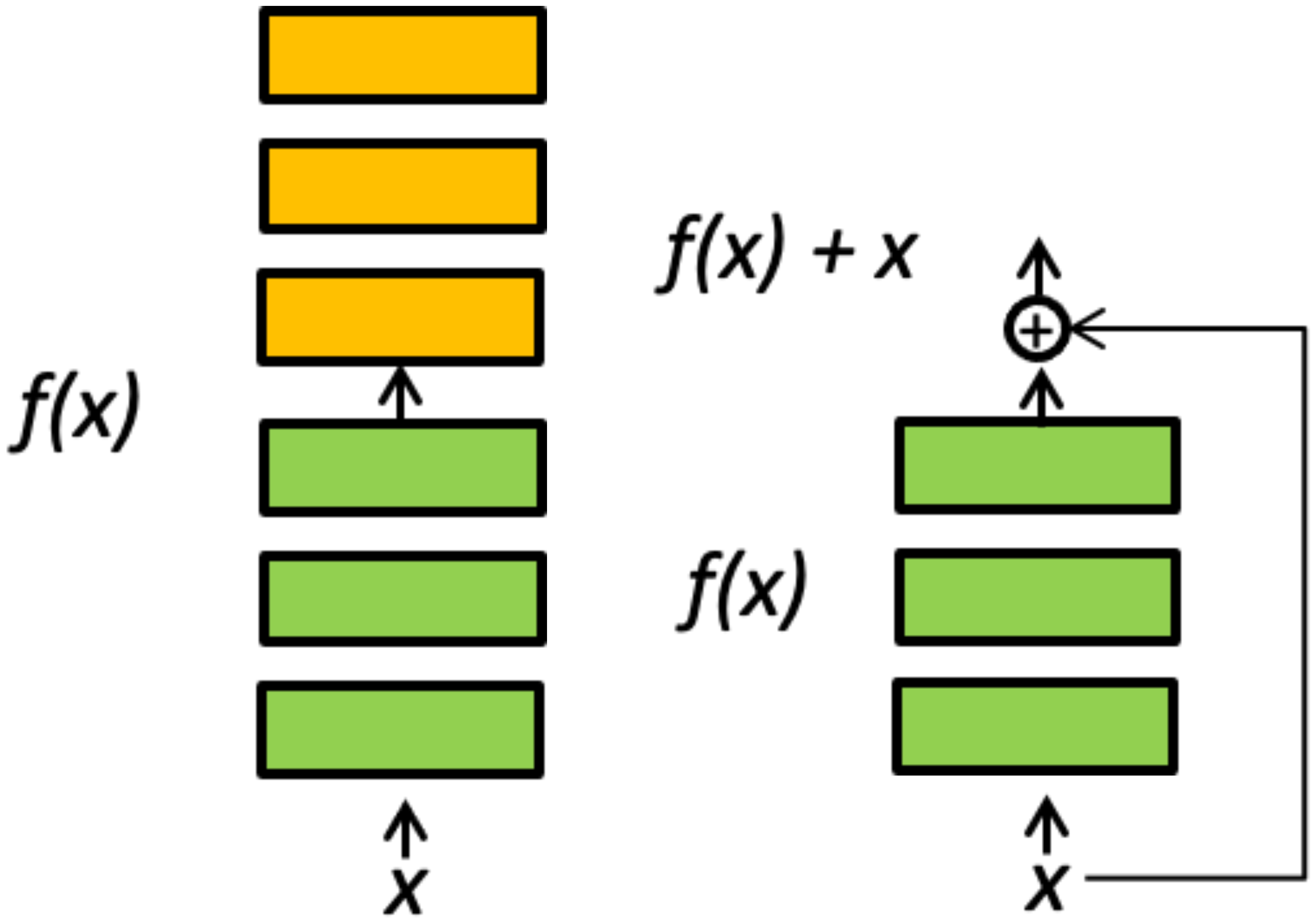
Residual Connections

- **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:



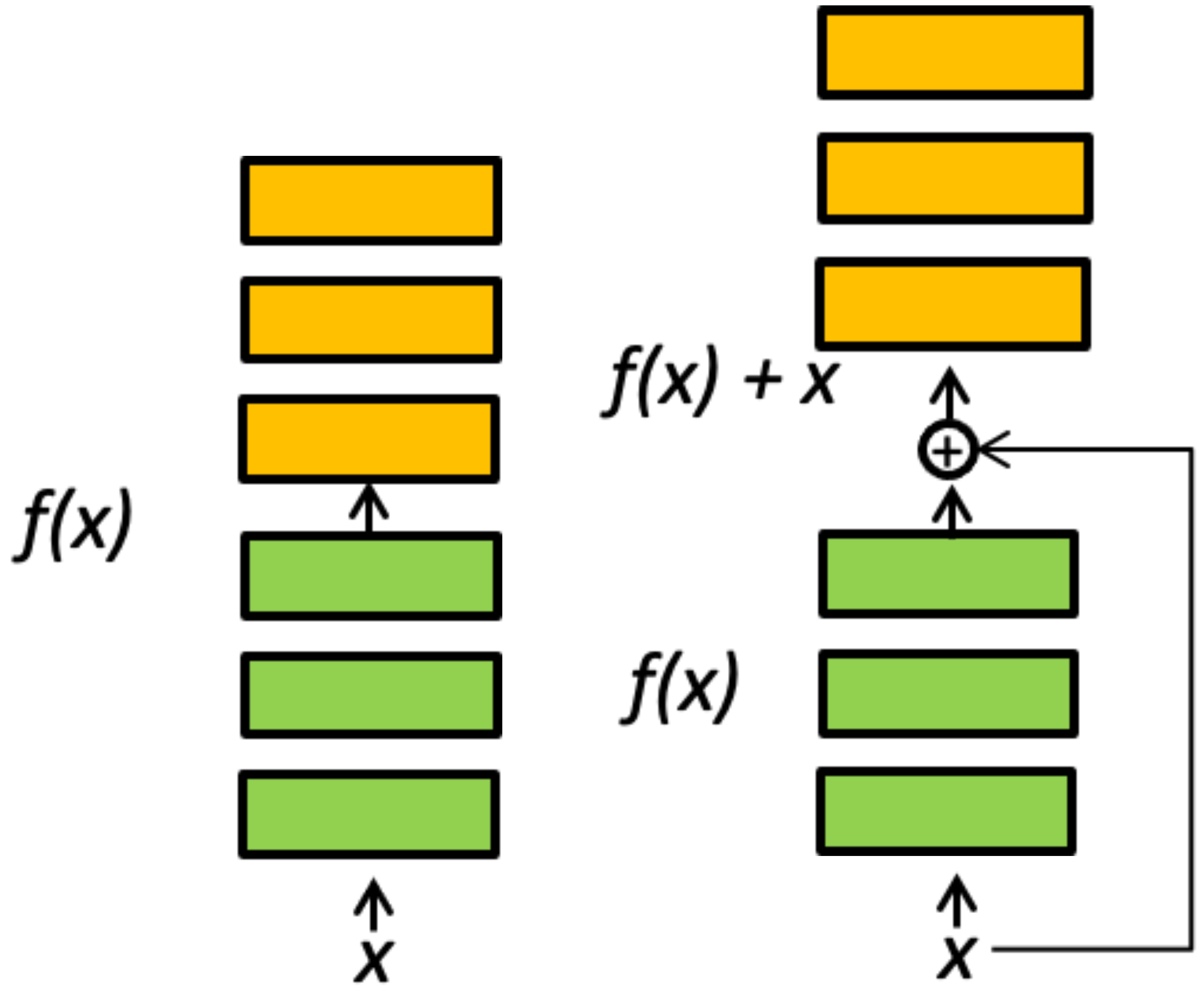
Residual Connections

- **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:



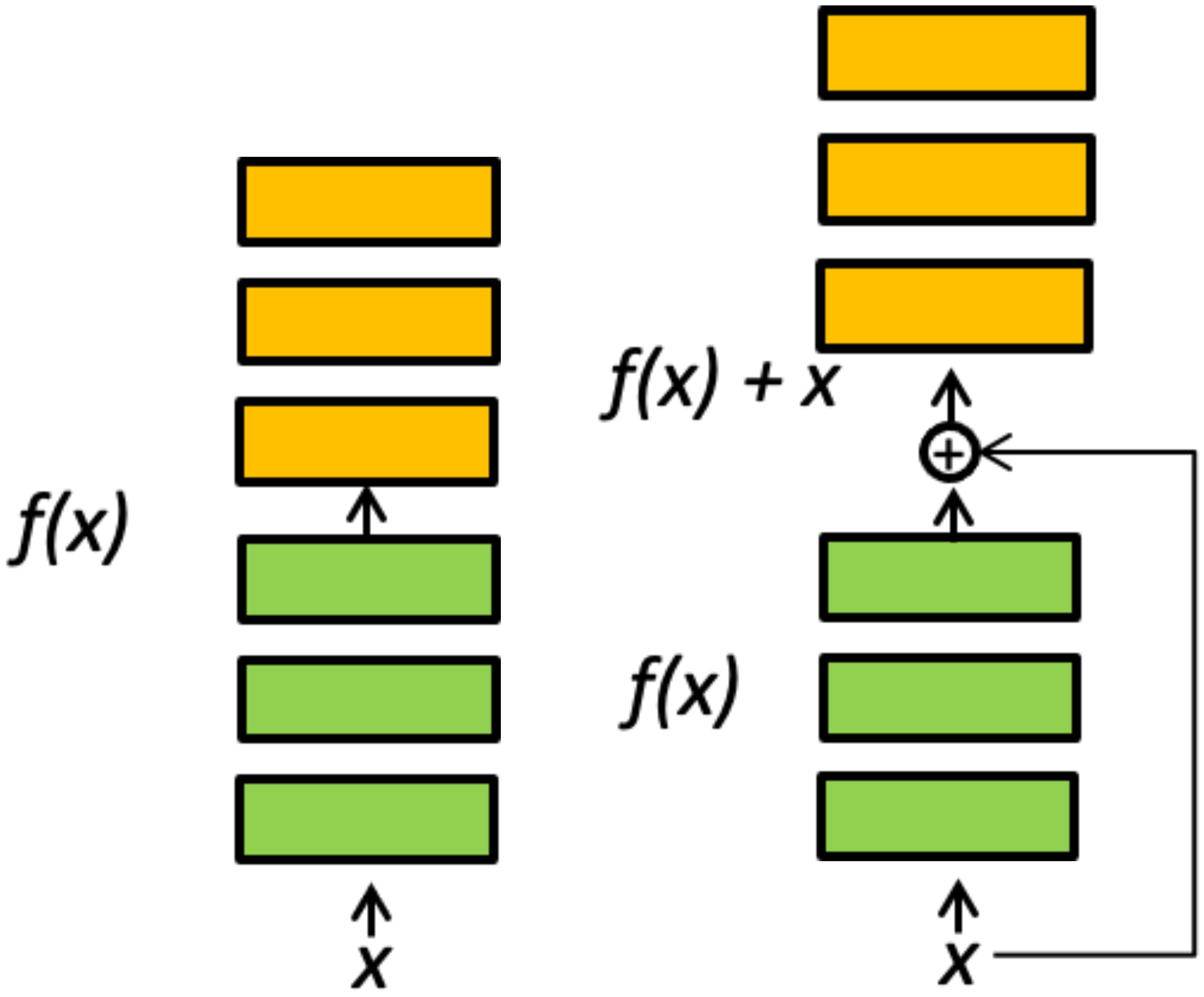
Residual Connections

- **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:



Residual Connections

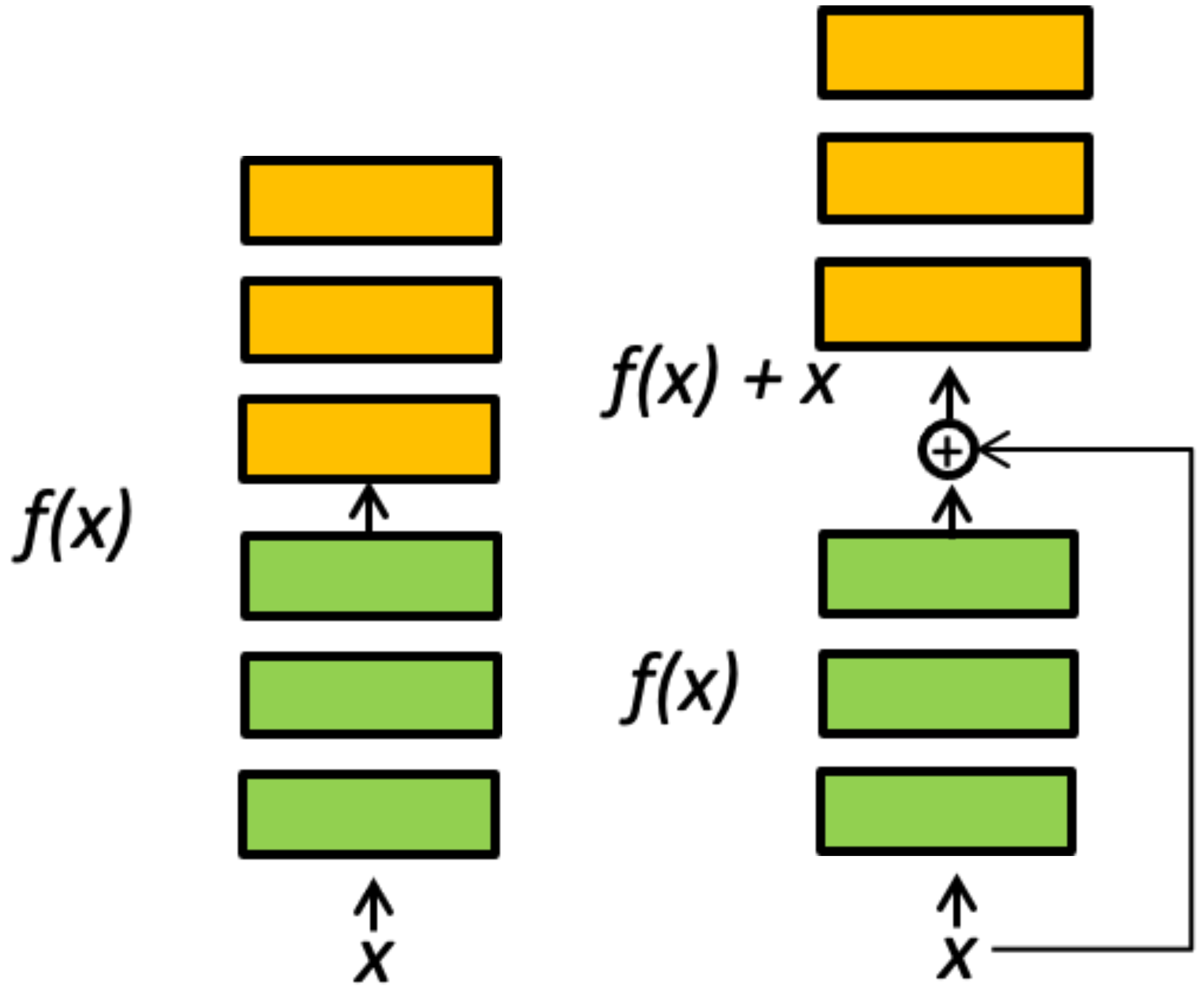
- **Idea:** identity might be hard to learn, but zero is easy!
  - Make all the weights tiny, produces zero for output
  - Can easily transform learning identity to learning zero:



Left: Conventional layer blocks

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:

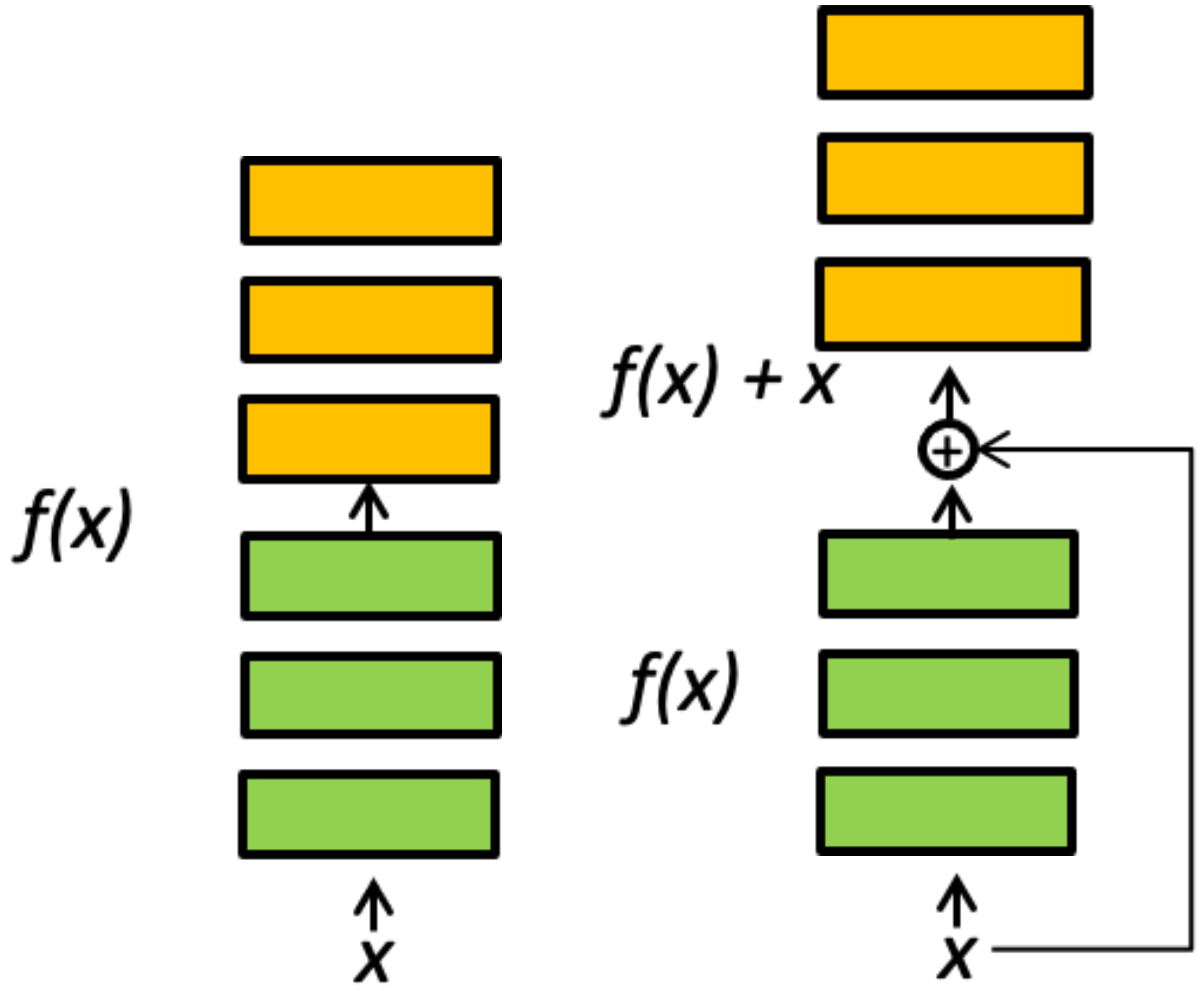


Left: Conventional layer blocks

Right: Residual layer blocks

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 layer blocks

Right: Residual layer blocks

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

ResNet Architecture

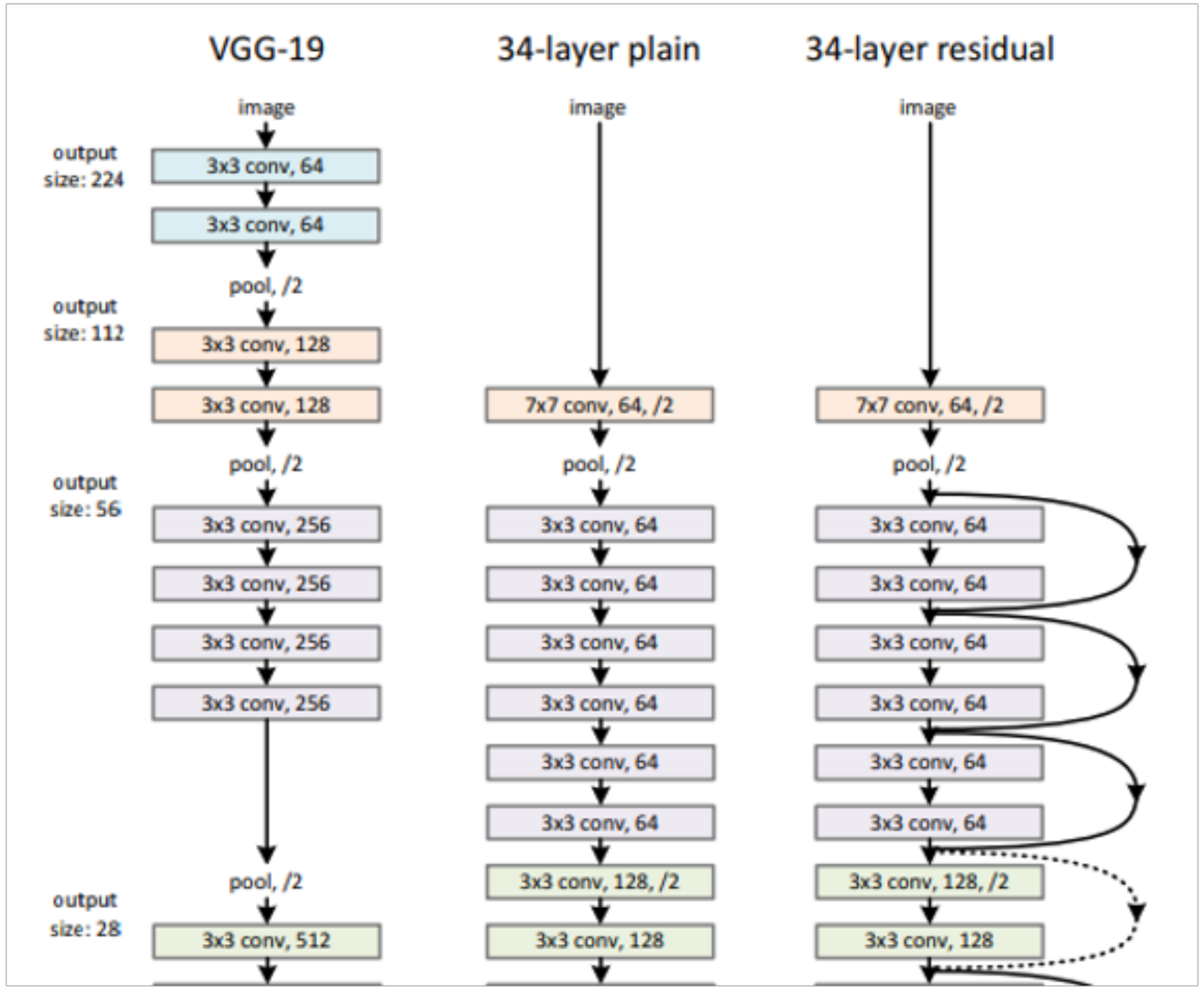
- **Idea:** Residual (skip) connections help make learning easier

ResNet Architecture

- **Idea:** Residual (skip) connections help make learning easier
  - Example architecture:

ResNet Architecture

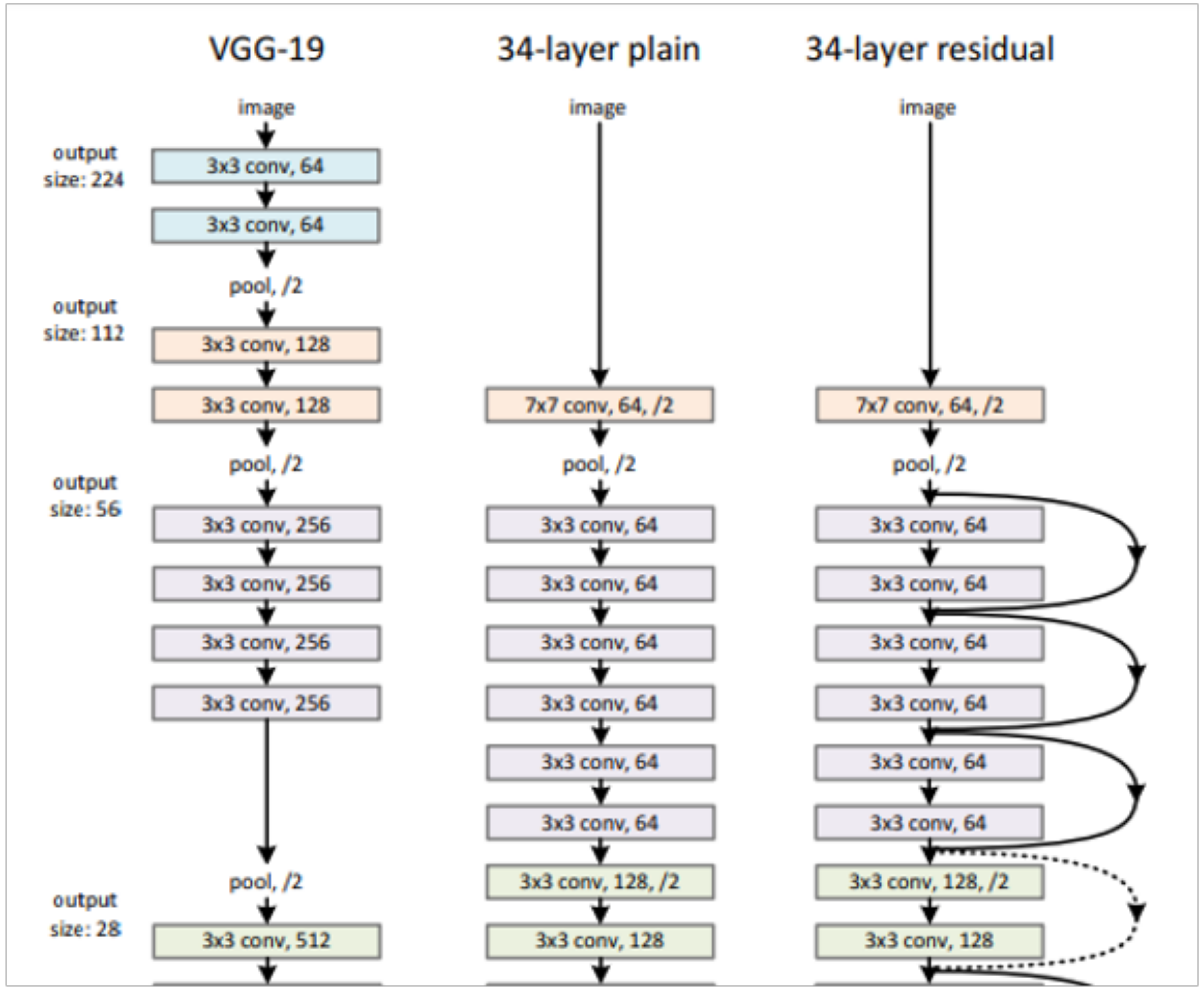
- **Idea:** Residual (skip) connections help make learning easier
  - Example architecture:



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

ResNet Architecture

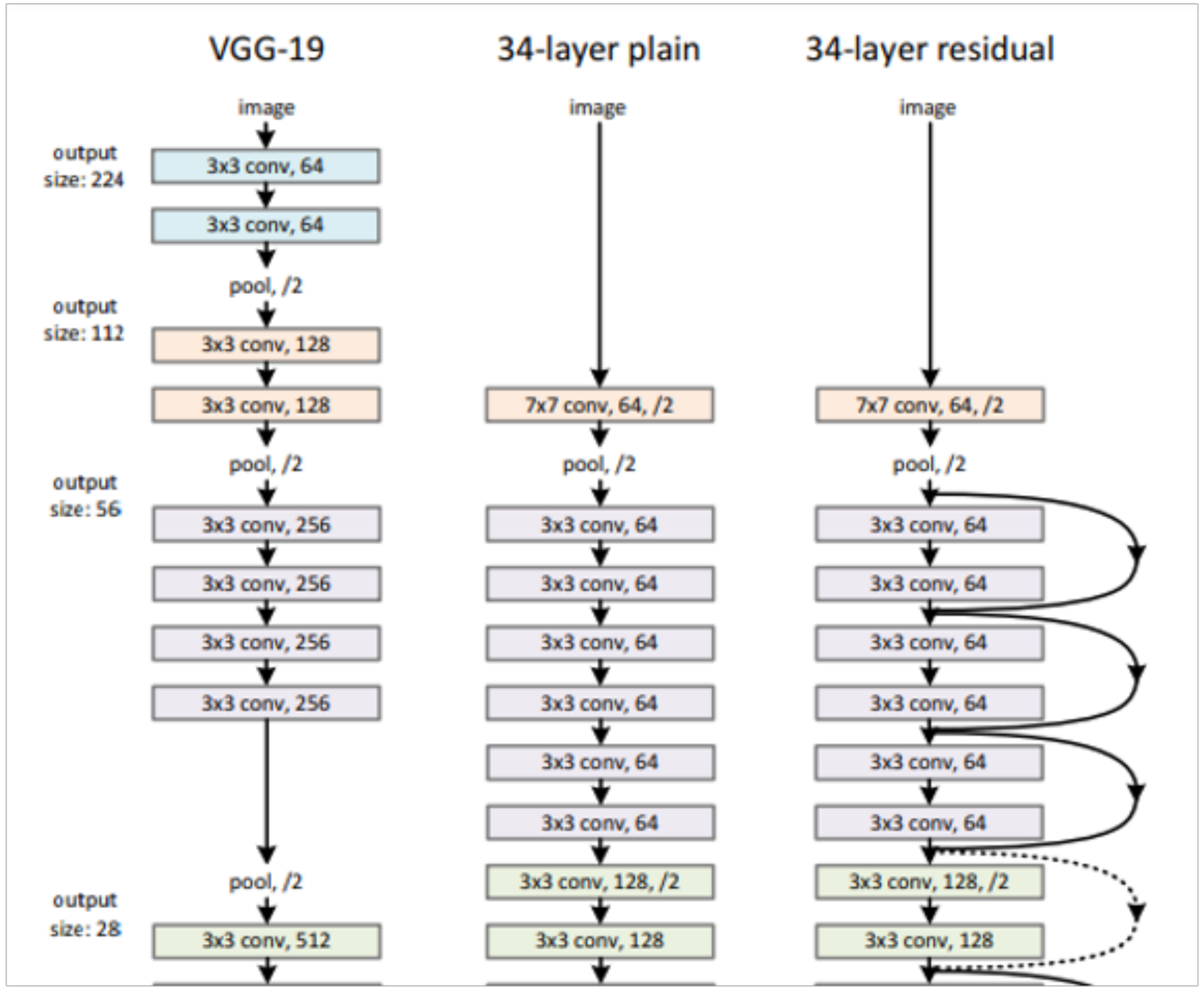
- **Idea:** Residual (skip) connections help make learning easier
  - Example architecture:
  - Note: residual connections
    - Every two layers for ResNet34



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

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”

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
    - ResNet

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