CS839 Special Topics in Deep Learning

Overview on Convolutional Neural Networks

Sharon Yixuan Li
University of Wisconsin-Madison

September 8, 2020

Reminder: Presentation Signup Sheet

Make sure you've signed up!

CS839 Special Topics in AI: Deep Learning -- Presentation Sign up

File Edit View Insert Format Data Tools Add-ons Help Last edit was made 41 minutes ago by ABHASH KUMAR SINGH

Course schedule: http://pages.cs.wisc.edu/~sharonli/courses/cs839_fall2020/schedule.html

Course schedule: http://pages.cs.wisc.edu/~sharonli/courses/cs839_fall2020/schedule.html

* Each presentation group can have a maximum 3 students. As a team, every student in the same group will receive the same score for the presentation.

* It's up to the presentation team to arrange the presentation & scribes in a way that maximize the outcome.

						(only fill this if C & E columns are full)	
Date	**Topic**	**Presenter 1 Name**	**Presenter 1 Email**	**Scribe person name**	**Scribe person email**	**PLZ MOVE TO OTHER SLOTS IF C & E COLUMNS ARE AVAILABLE**	
September 10	Neural Architecure Design (+10% bonus in final grade)	Shri Shruthi Shridhar	shridhar2@wisc.edu	Sacha Jungerman	sjungerman@wisc.edu	Diwanshu Jain	djain23@wisc.edu
September 15	Neural Architecure Design (+5% bonus in final grade)	Bhavya Goyal	bgoyal2@wisc.edu	Nils Palumbo	npalumbo@wisc.edu	Lichengxi Huang	lhuang236@wisc.edu
September 22	Trustworthy Deep Learning	Bastin Joseph	bjoseph5@wisc.edu	Yifei Ming	ming5@wisc.edu	Sean Chung	cchung49@wisc.edu
September 24	Trustworthy Deep Learning	Abhirav Gholba	gholba@wisc.edu	Niharika Tomar	Niharika Tomar		
September 29	Trustworthy Deep Learning	Maulik Shah	msshah4@wisc.edu	Yang Guo	yguo@cs.wisc.edu		
October 13	Interpretable Deep Learning	Grishma gupta	ggupta7@wisc.edu	Yunjia Zhang	yunjia@cs.wisc.edu	Lokit	lparas@wisc.edu
October 15	Interpretable Deep Learning	Ziqian Lin	zlin284@wisc.edu	Sreya Dutta Roy	duttaroy@wisc.edu		
October 22	Deep Learning Generalization and Theory	Tanmayee Joshi	tsjoshi@wisc.edu	Sean Chung	cchung49@wisc.edu		
October 27	Deep Learning Generalization and Theory	Peyman Morteza	morteza@wisc.edu				
November 3	Learning with less supervision			Abhash Kumar Singh	abhashkumar.singh@wisc.edu		
November 5	Learning with less supervision	Yien Xu	yien@cs.wisc.edu	Lichengxi Huang	lhuang236@wisc.edu		
November 10	Learning with less supervision	Liang Shang	lshang6@wisc.edu	Rui Huang		Siyang Chen	schen658@wisc.edu
November 17	Lifelong learning			Abhirav Gholba	gholba@wisc.edu		
November 19	Lifelong learning	Akshata	akshatabhat@cs.wisc.e	Zifan Liu	zifan@cs.wisc.edu		
December 1	Deep generative modeling	Aditya K A	aka@cs.wisc.edu				
December 3	Deep generative modeling	Roger	waleffe@wisc.edu	Jason			

Slide, Quiz, Notes Submission (Paper Presentation)

- Email to TA sunyiyou@cs.wisc.edu day before the presentation (by **6pm**)
  - Downloadable link to Google Drive slides
  - Keynote slides or Powerpoint slides
  - Quiz questions (3) and answers

Outline

- Brief review of convolutional computations
    - 2D convolution
    - Padding, stride etc
    - Multiple input and output channels
- Basic convolutional neural networks
    - **LeNet** (the first convolutional neural network)
    - **AlexNet**
    - **ResNet**
    - **DenseNet** (more in next class)

How to classify

Cats vs. dogs?

Dual

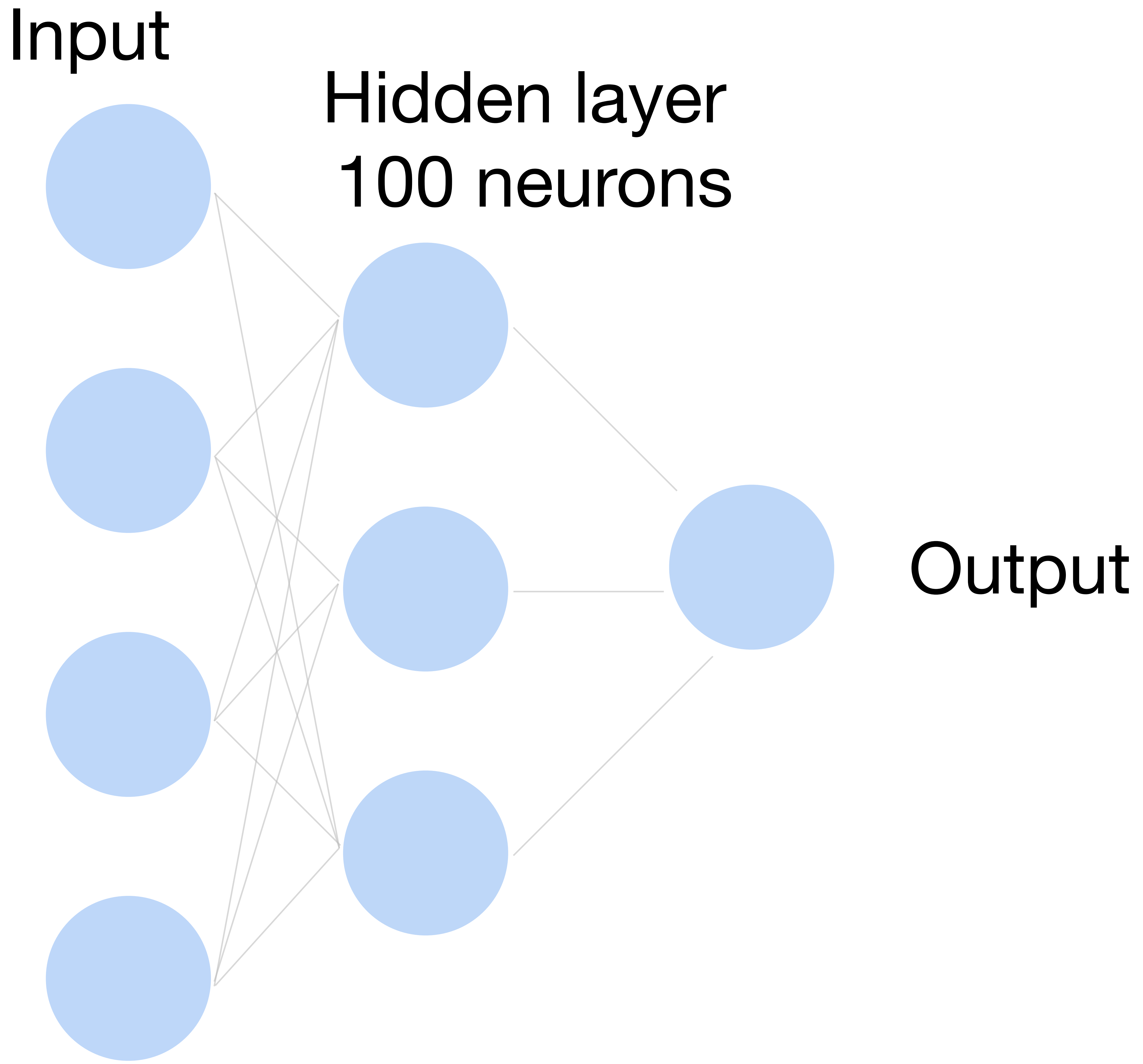
12MP

wide-angle and
telephoto cameras

36M floats in a RGB image!

Fully Connected Networks

Cats vs. dogs?



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

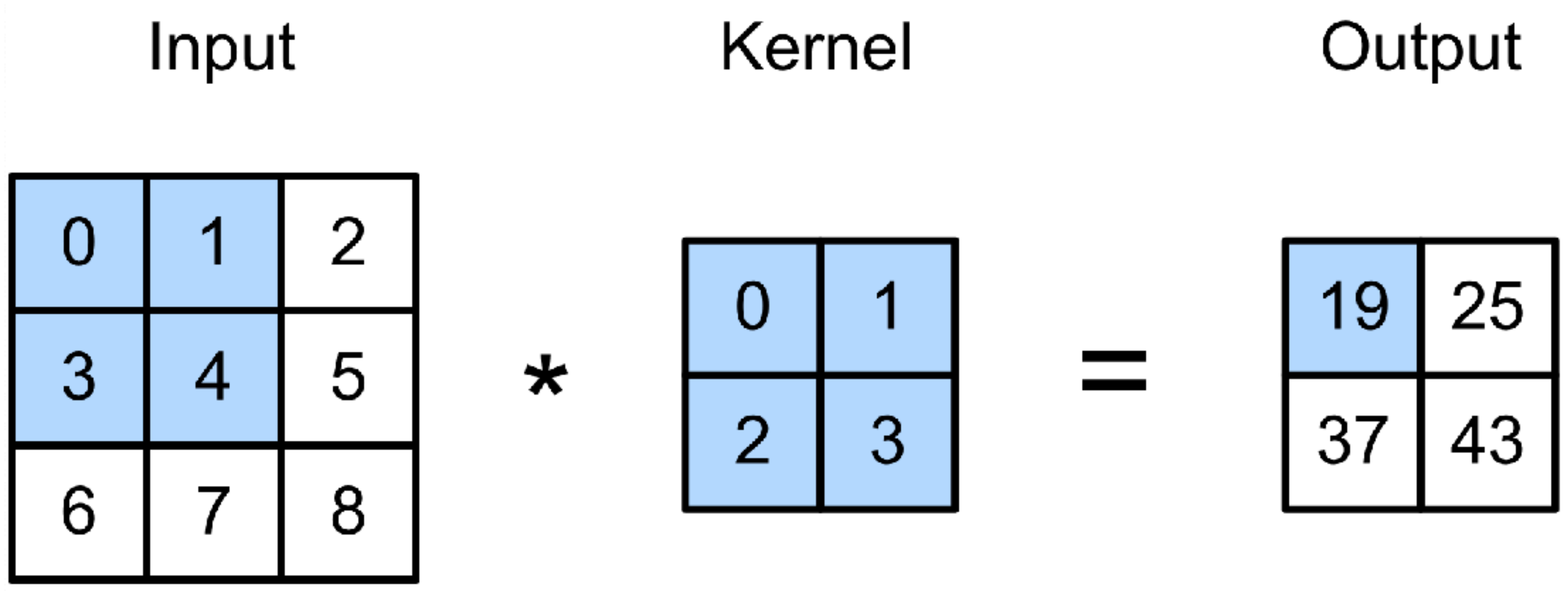
Convolutions come to rescue!

Where is Waldo?

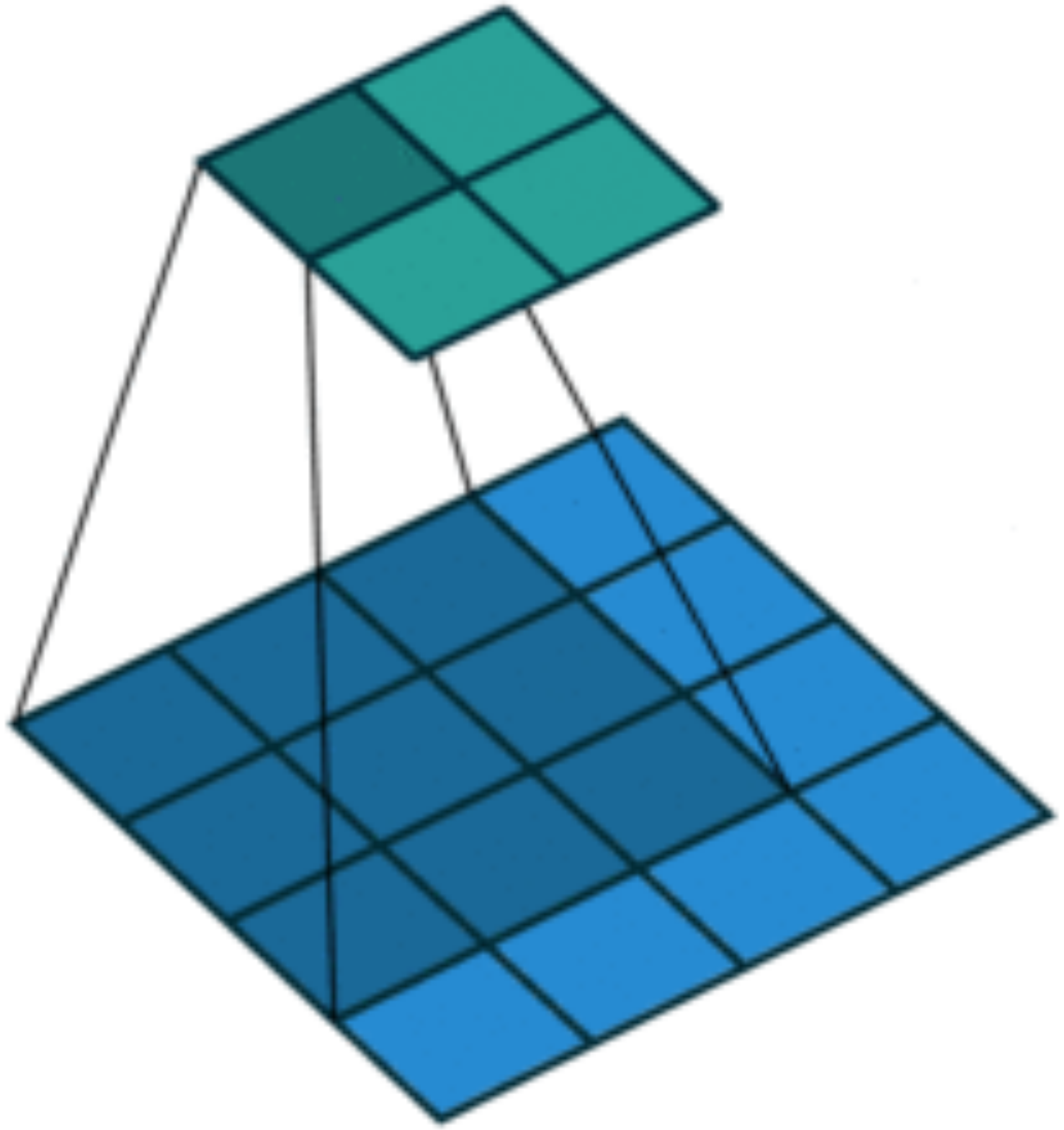
Why Convolution?

- Translation Invariance
- Locality

2-D Convolution

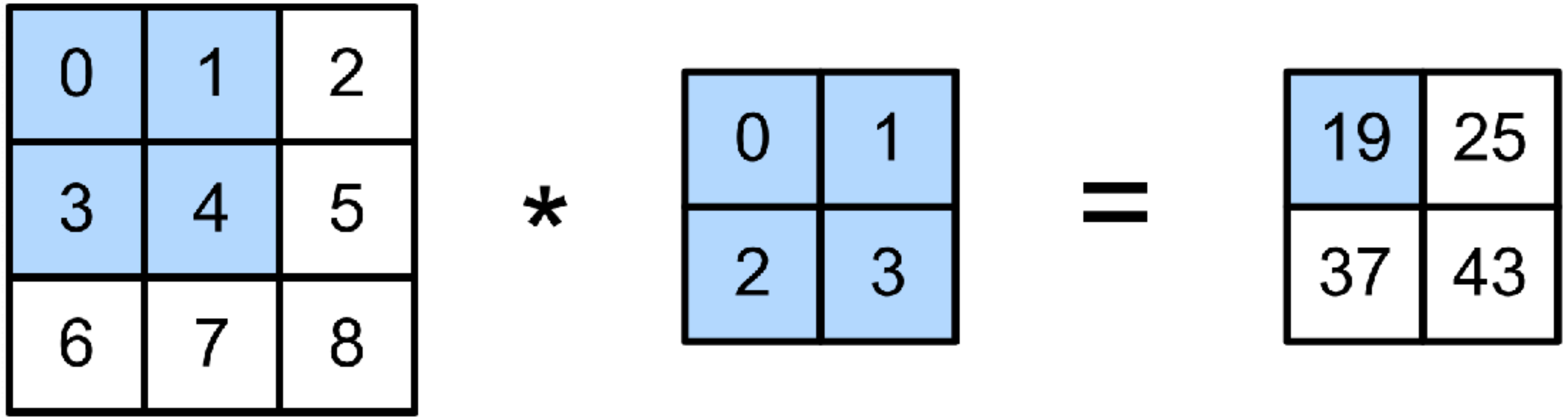


$$0 \times 0 + 1 \times 1 + 3 \times 2 + 4 \times 3 = 19,$$
$$1 \times 0 + 2 \times 1 + 4 \times 2 + 5 \times 3 = 25,$$
$$3 \times 0 + 4 \times 1 + 6 \times 2 + 7 \times 3 = 37,$$
$$4 \times 0 + 5 \times 1 + 7 \times 2 + 8 \times 3 = 43.$$



(vdumoulin@ Github)

2-D Convolution Layer

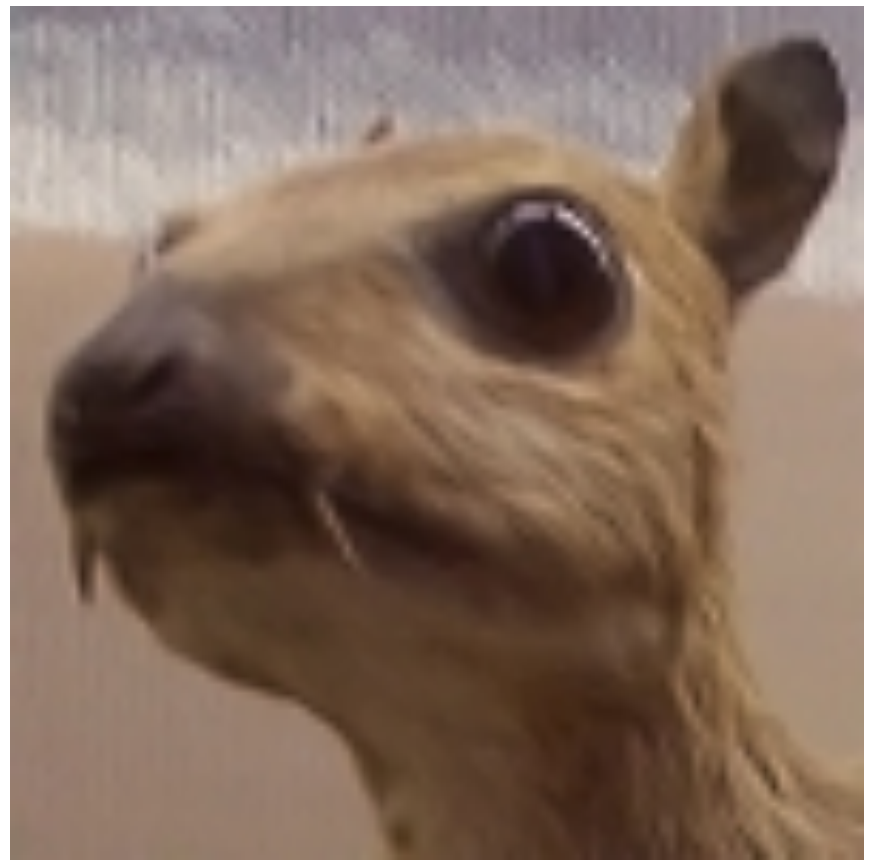


- $\mathbf{X} : n_h \times n_w$ input matrix
- $\mathbf{W} : k_h \times k_w$ kernel matrix
- b: scalar bias
- $\mathbf{Y} : (n_h - k_h + 1) \times (n_w - k_w + 1)$ output matrix

$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} + b$$

- **W** and *b* are learnable parameters

Examples

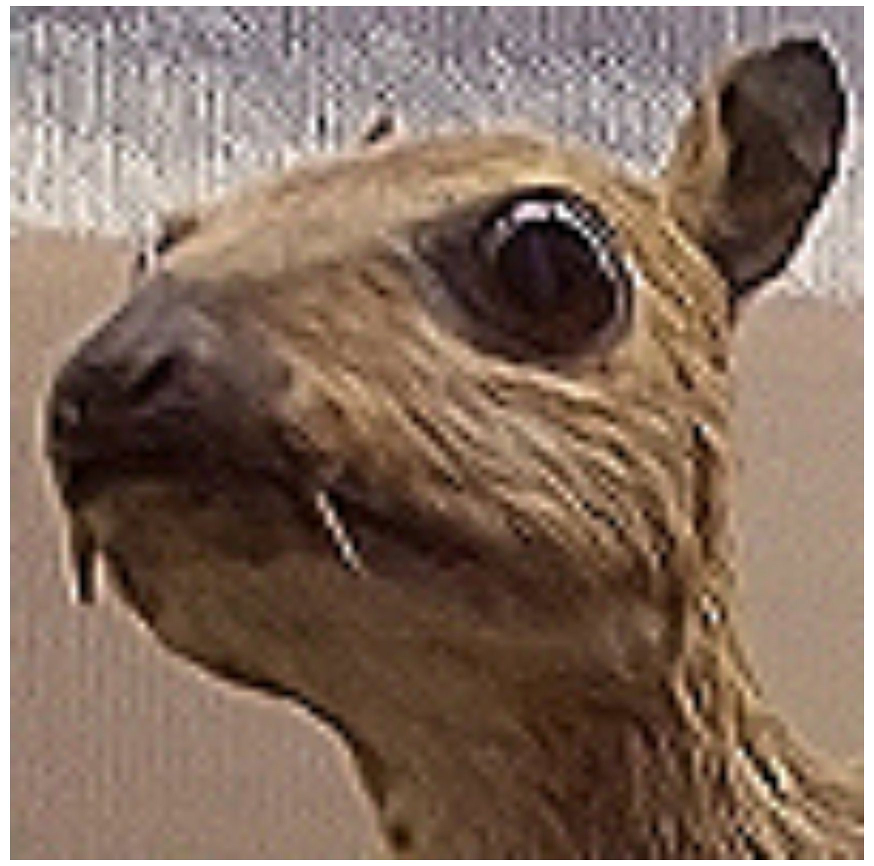


(wikipedia)

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Edge Detection

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$



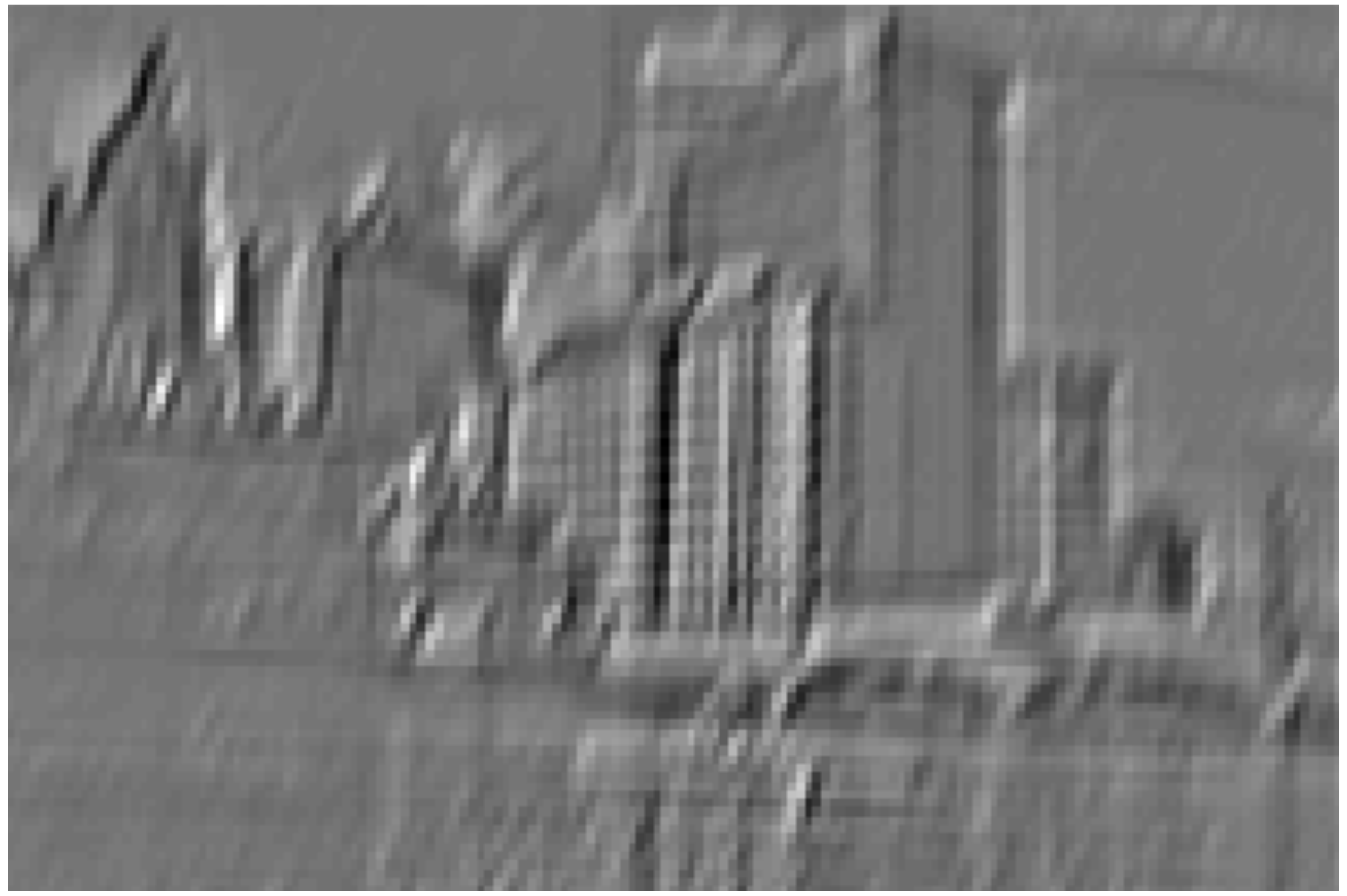
Sharpen

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gaussian Blur

Examples

(Rob Fergus)



Efficiency of Convolution

- Input size: 320 x 280
- Kernel Size: 2 x 1
- Output size: 319 x 280

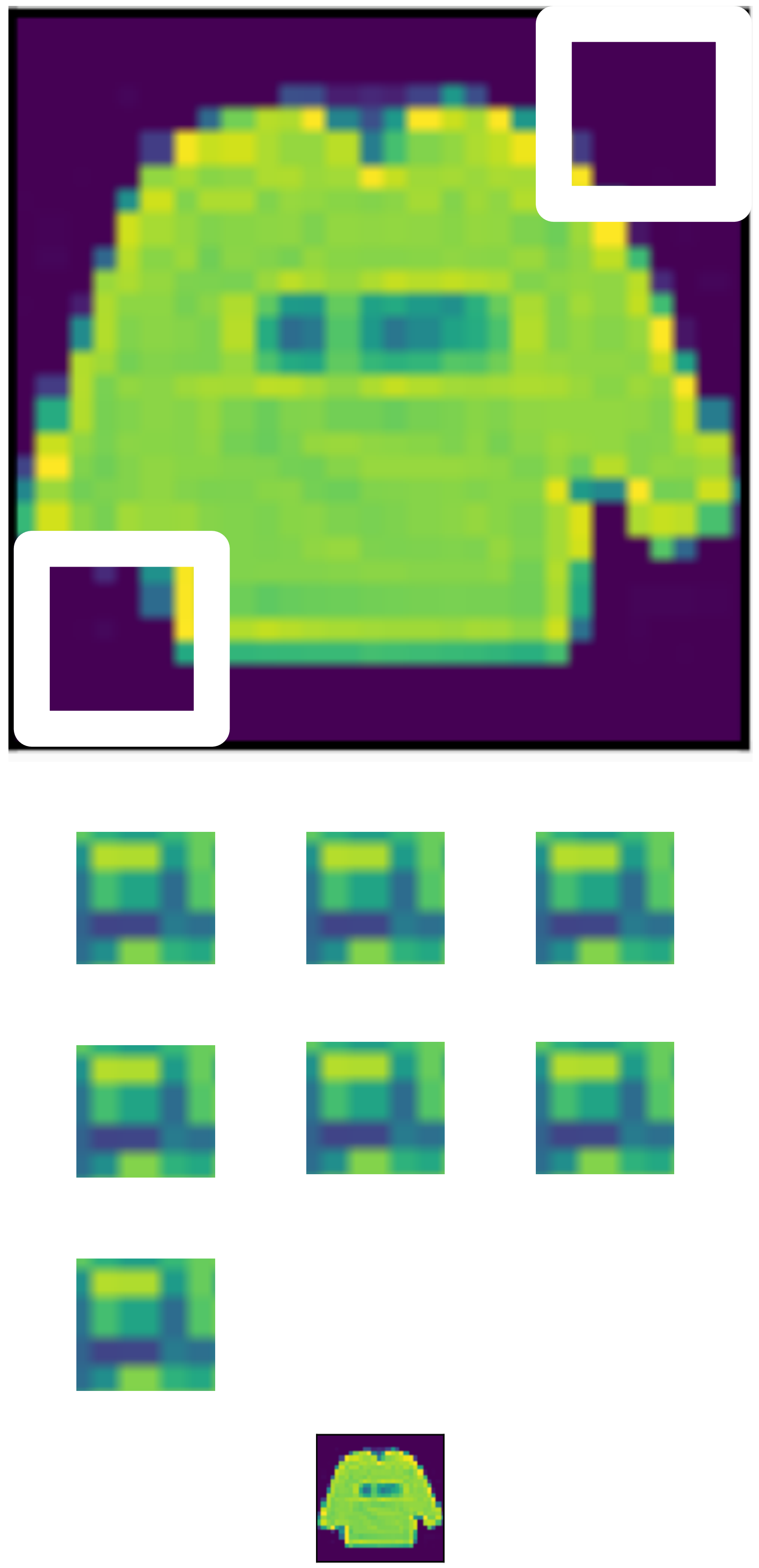
	Convolution	Dense matrix
Stored floats		
Float muls or adds		


Padding and Stride

Padding

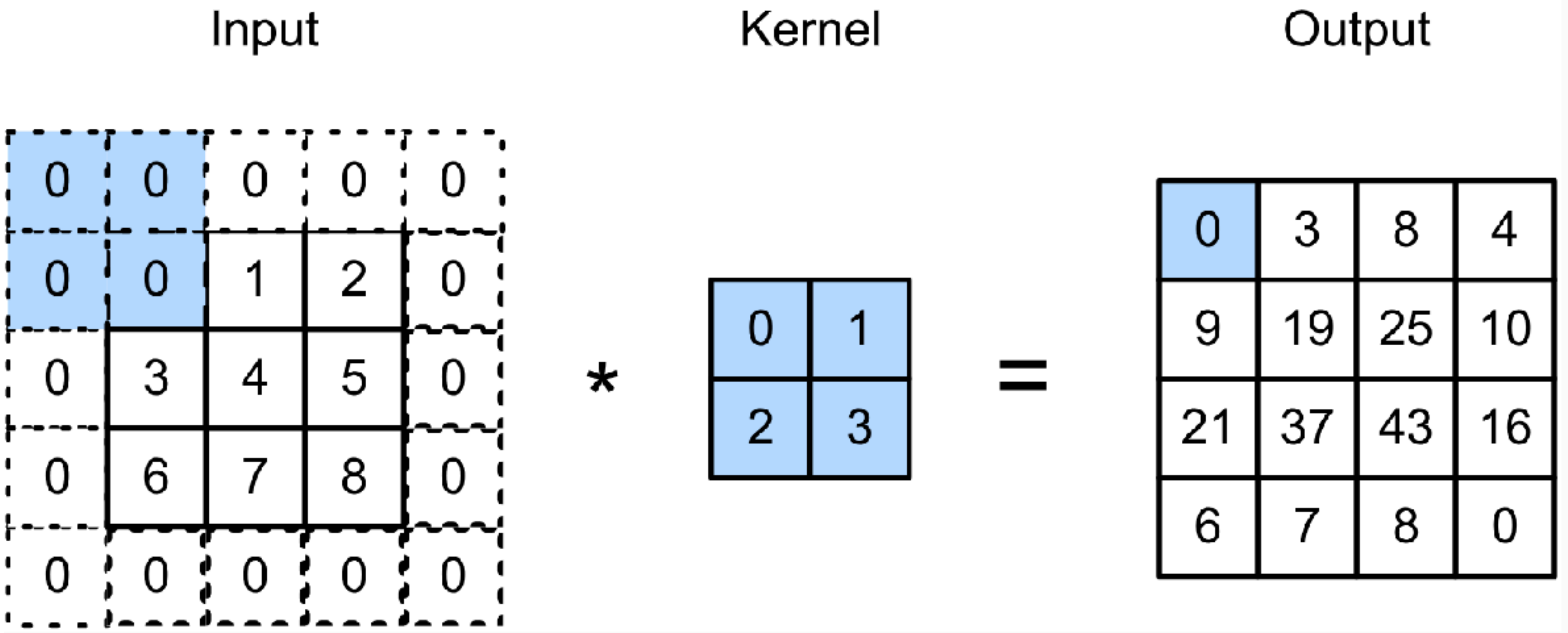
- Given a 32 x 32 input image
- Apply convolution with 5 x 5 kernel
  - 28 x 28 output with 1 layer
  - 4 x 4 output with 7 layers
- Shape decreases faster with larger kernels
  - Shape reduces from $n_h \times n_w$ to

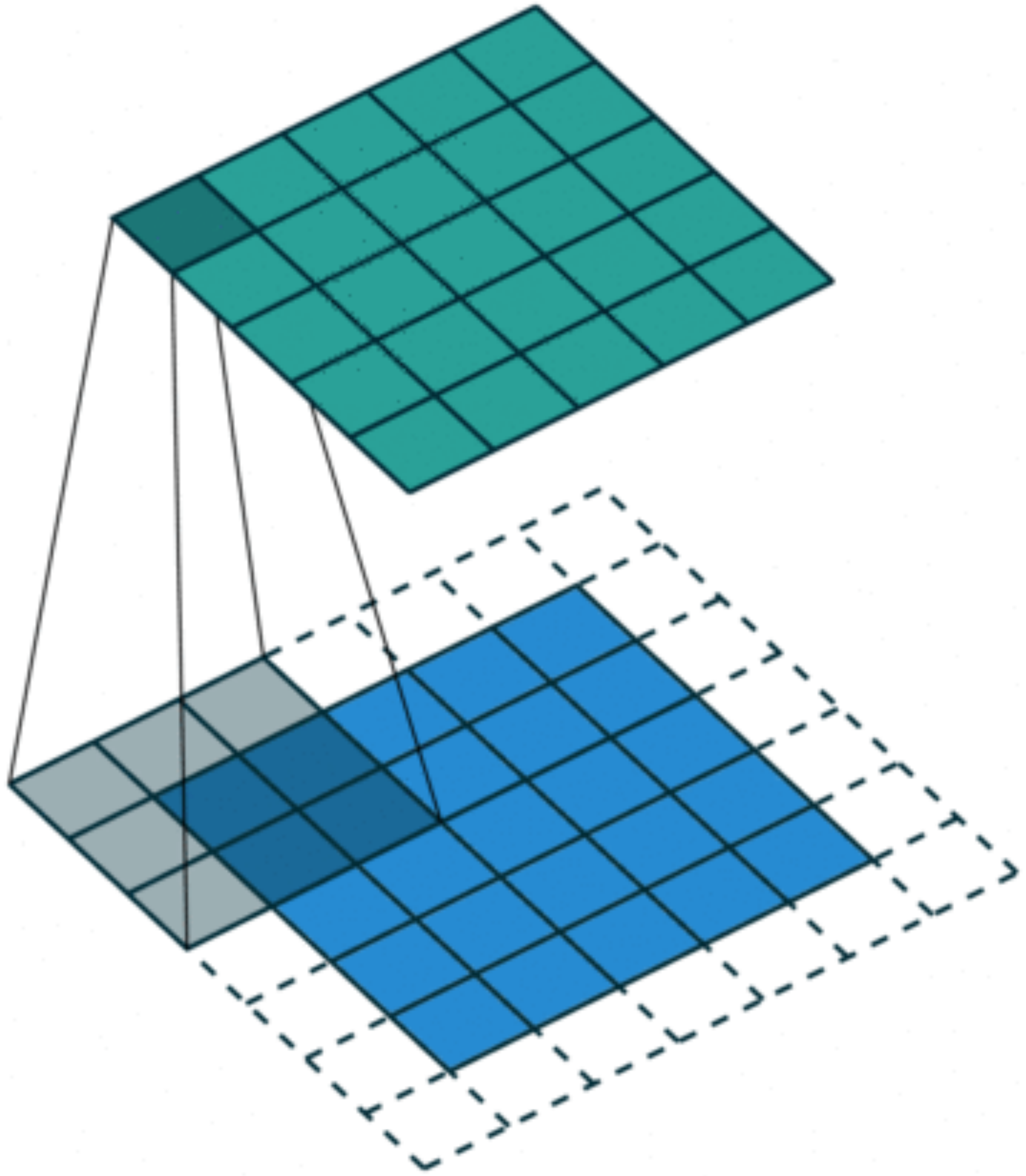
$$(n_h - k_h + 1) \times (n_w - k_w + 1)$$



Padding

Padding adds rows/columns around input





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

Padding

- Padding p_h rows and p_w columns, output shape will be

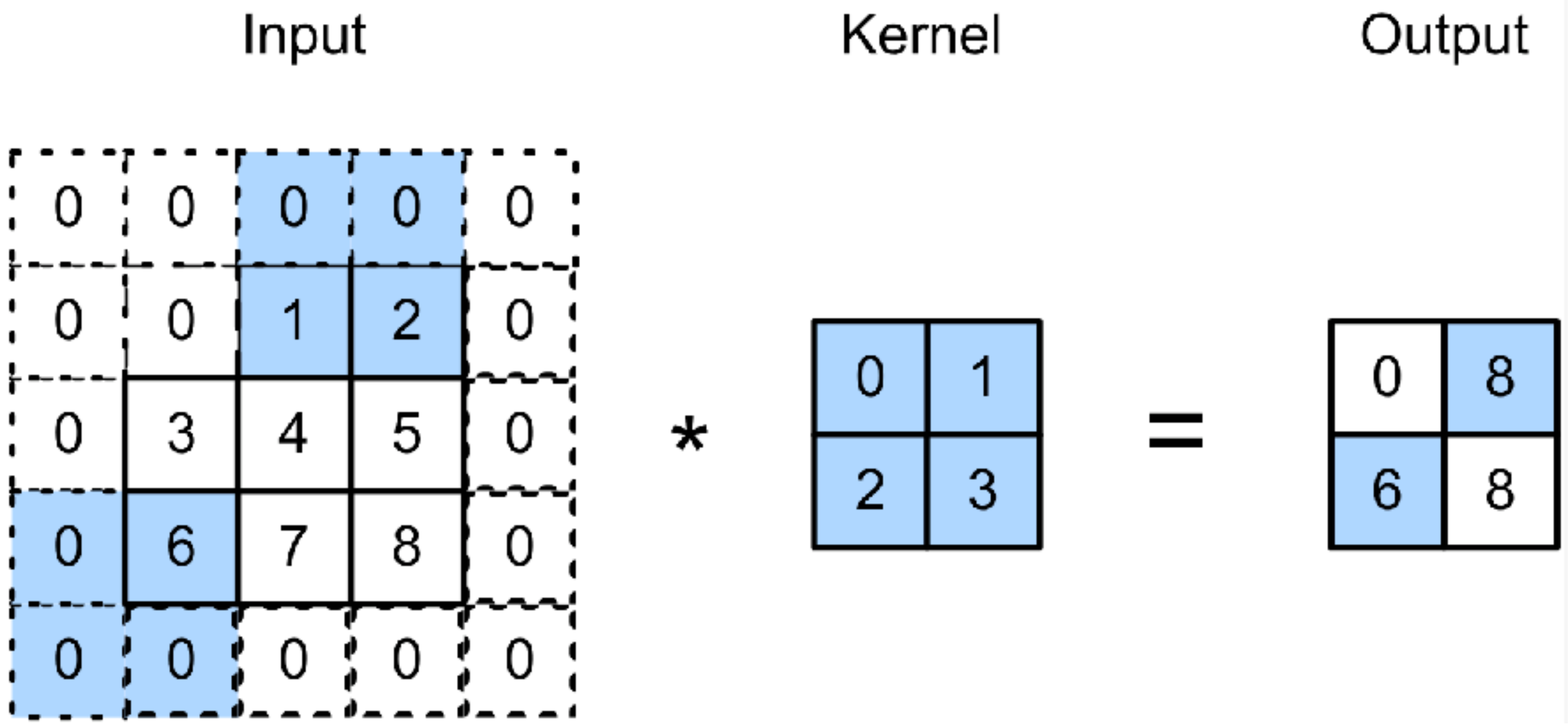
$$(n_h - k_h + p_h + 1) \times (n_w - k_w + p_w + 1)$$

- A common choice is $p_h = k_h - 1$ and $p_w = k_w - 1$
  - Odd k_h: pad $p_h/2$ on both sides
  - Even k_h: pad $\lceil p_h/2 \rceil$ on top, $\lfloor p_h/2 \rfloor$ on bottom

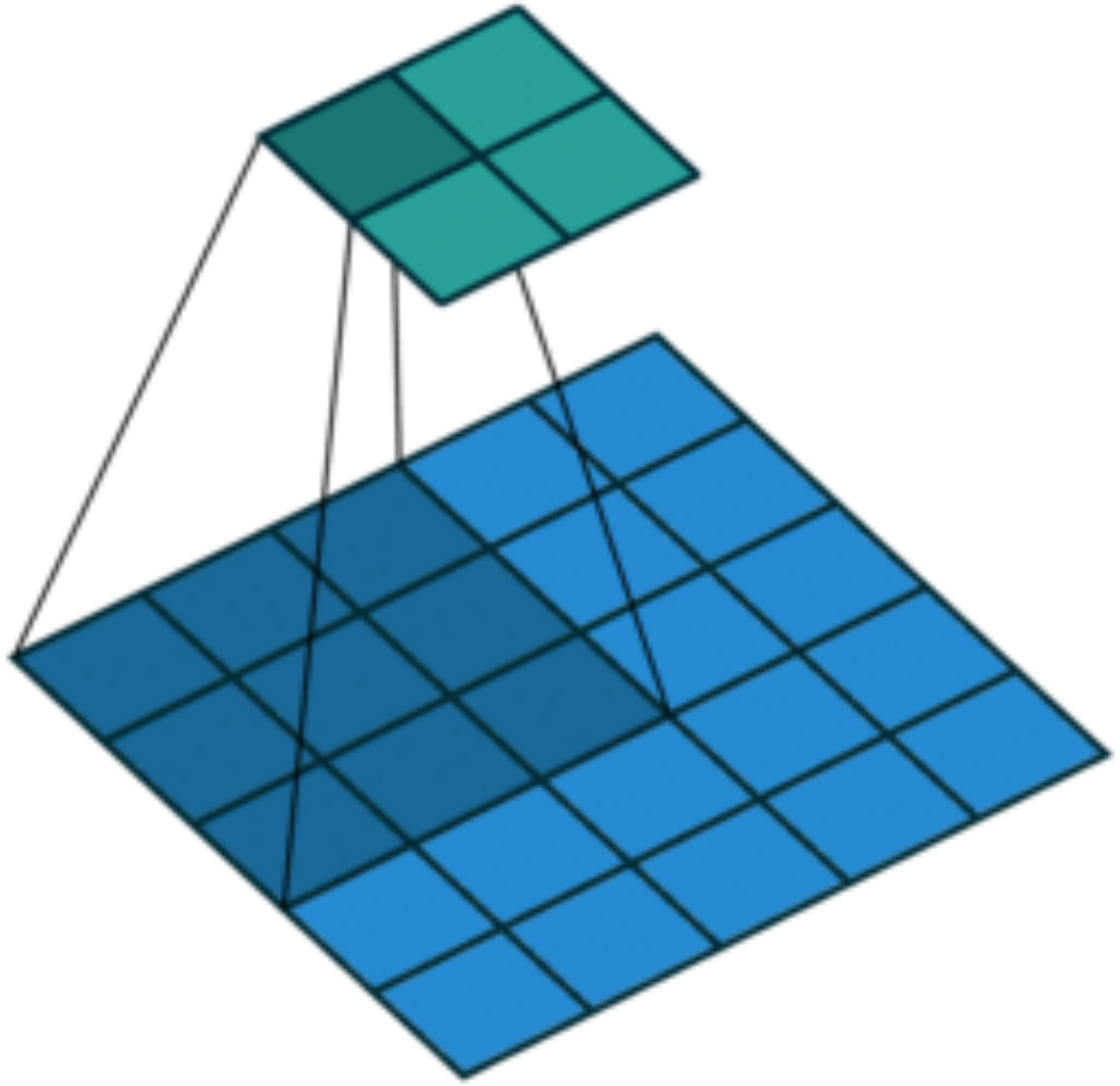
Stride

- Stride is the #rows/#columns per slide

Strides of 3 and 2 for height and width



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



Multiple Input and Output Channels

Multiple Input Channels

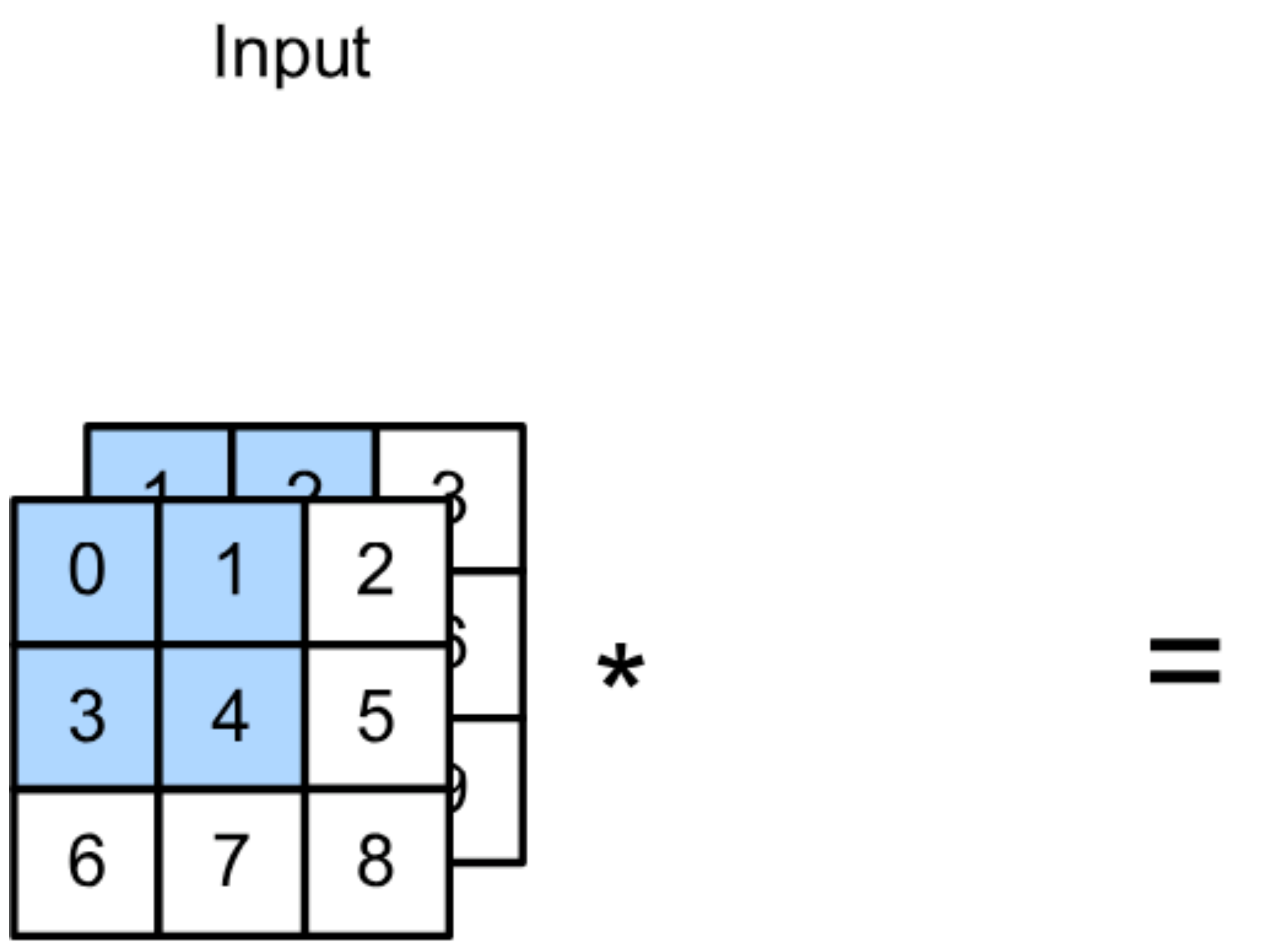
- Color image may have three RGB channels
- Converting to grayscale loses information

Multiple Input Channels

- Color image may have three RGB channels
- Converting to grayscale loses information

Multiple Input Channels

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



Multiple Input Channels

- $\mathbf{X} : c_i \times n_h \times n_w$ input
- $\mathbf{W} : c_i \times k_h \times k_w$ kernel
- $\mathbf{Y} : m_h \times m_w$ output

$$\mathbf{Y} = \sum_{i=0}^{c_i} \mathbf{X}_{i,:,:} \star \mathbf{W}_{i,:,:}$$

Multiple Output Channels

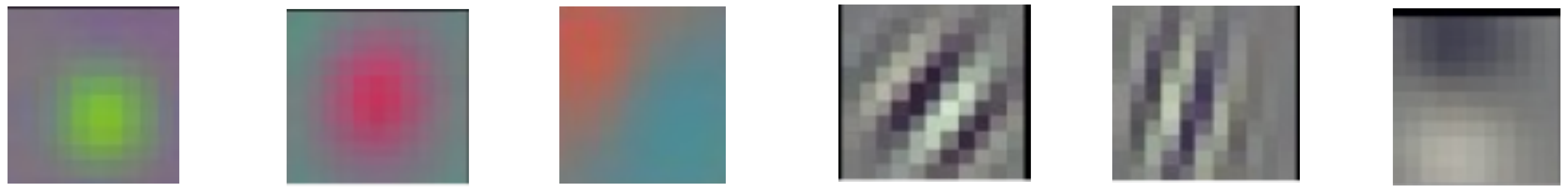
- No matter how many inputs channels, so far we always get single output channel
- We can have **multiple 3-D kernels**, each one generates a output channel
- Input $\mathbf{X} : c_i \times n_h \times n_w$
- Kernel $\mathbf{W} : c_o \times c_i \times k_h \times k_w$
- Output $\mathbf{Y} : c_o \times m_h \times m_w$

$$\mathbf{Y}_{i,:,:} = \mathbf{X} \star \mathbf{W}_{i,:,:,:}$$

$$\text{for } i = 1, \ldots, c_o$$

Multiple Input/Output Channels

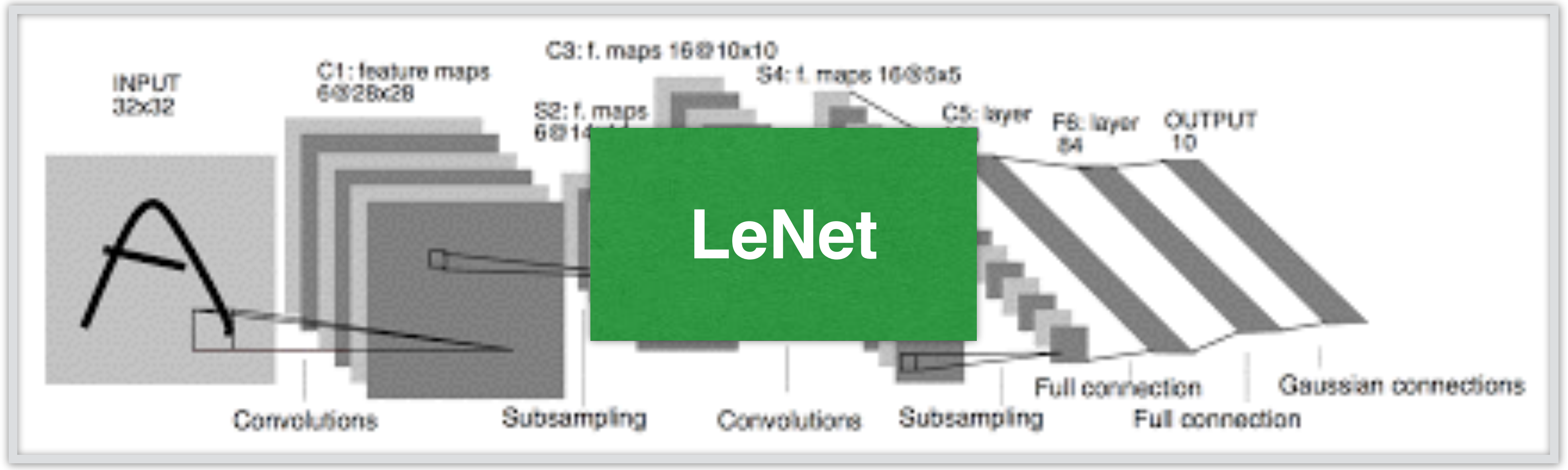
- Each output channel may recognize a particular pattern

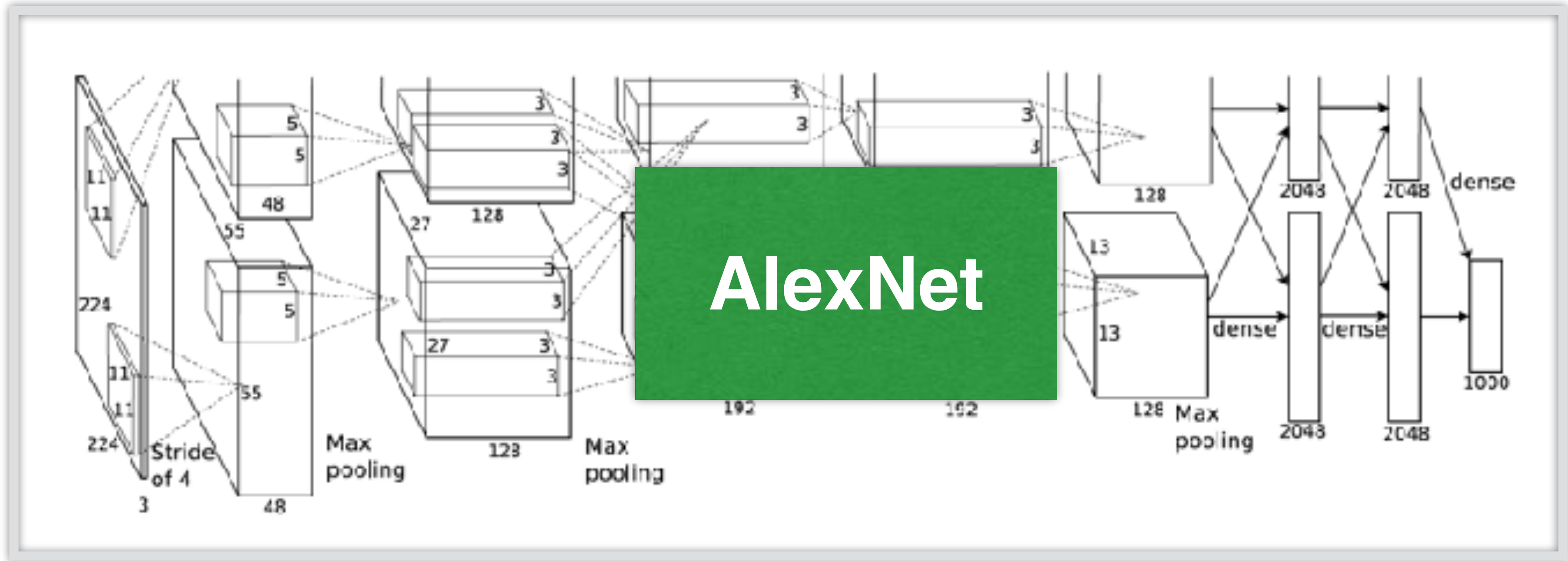


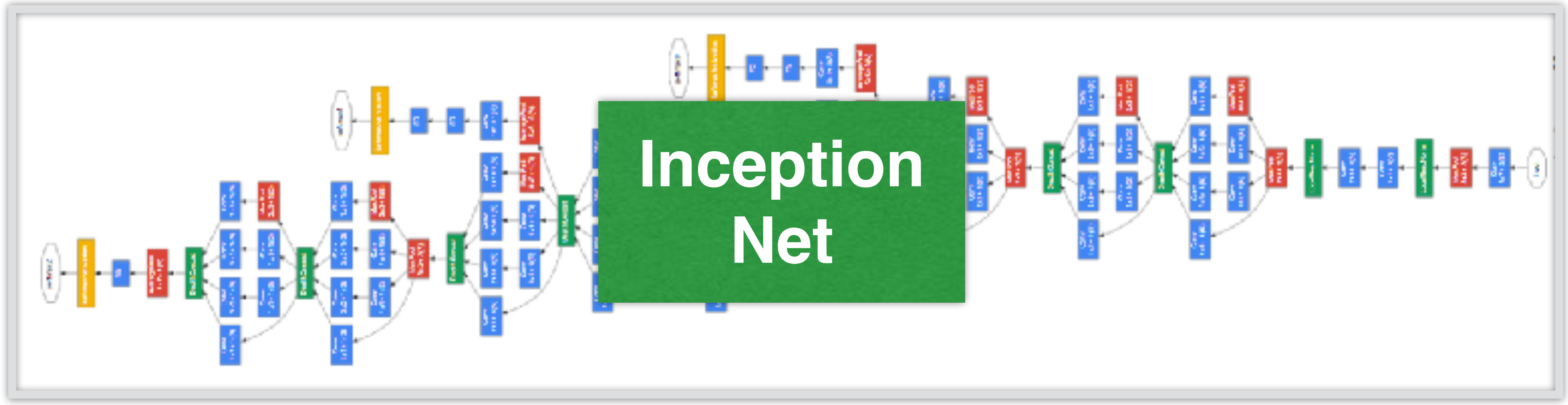
(Gabor filters)

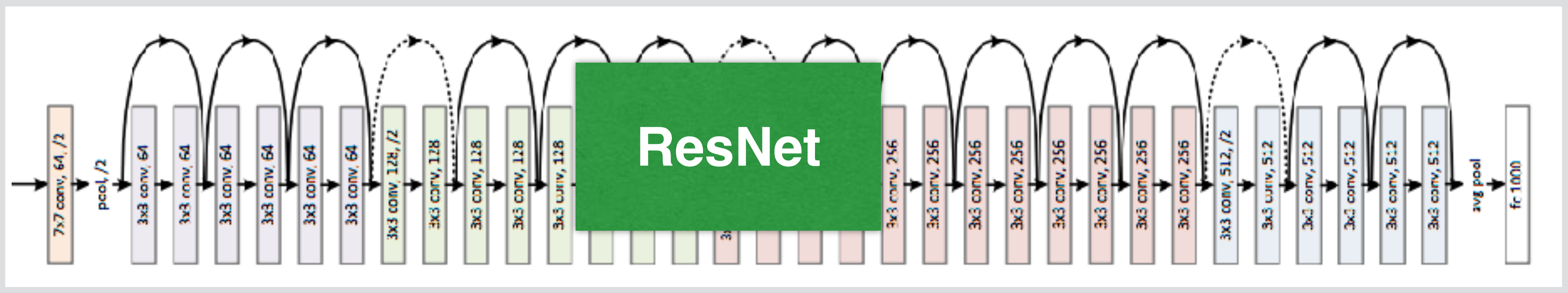
Convolutional Neural Networks

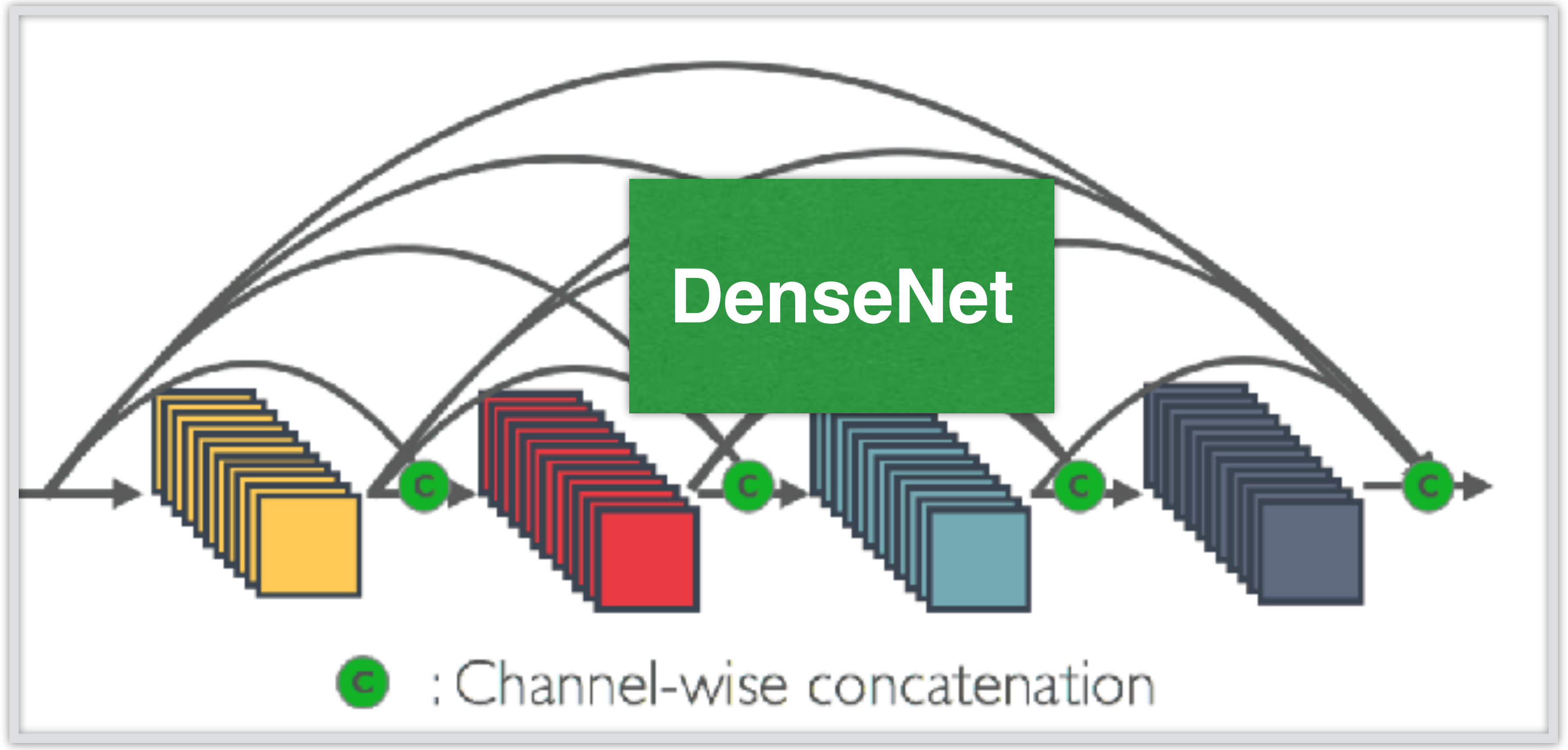
Evolution of neural net architectures







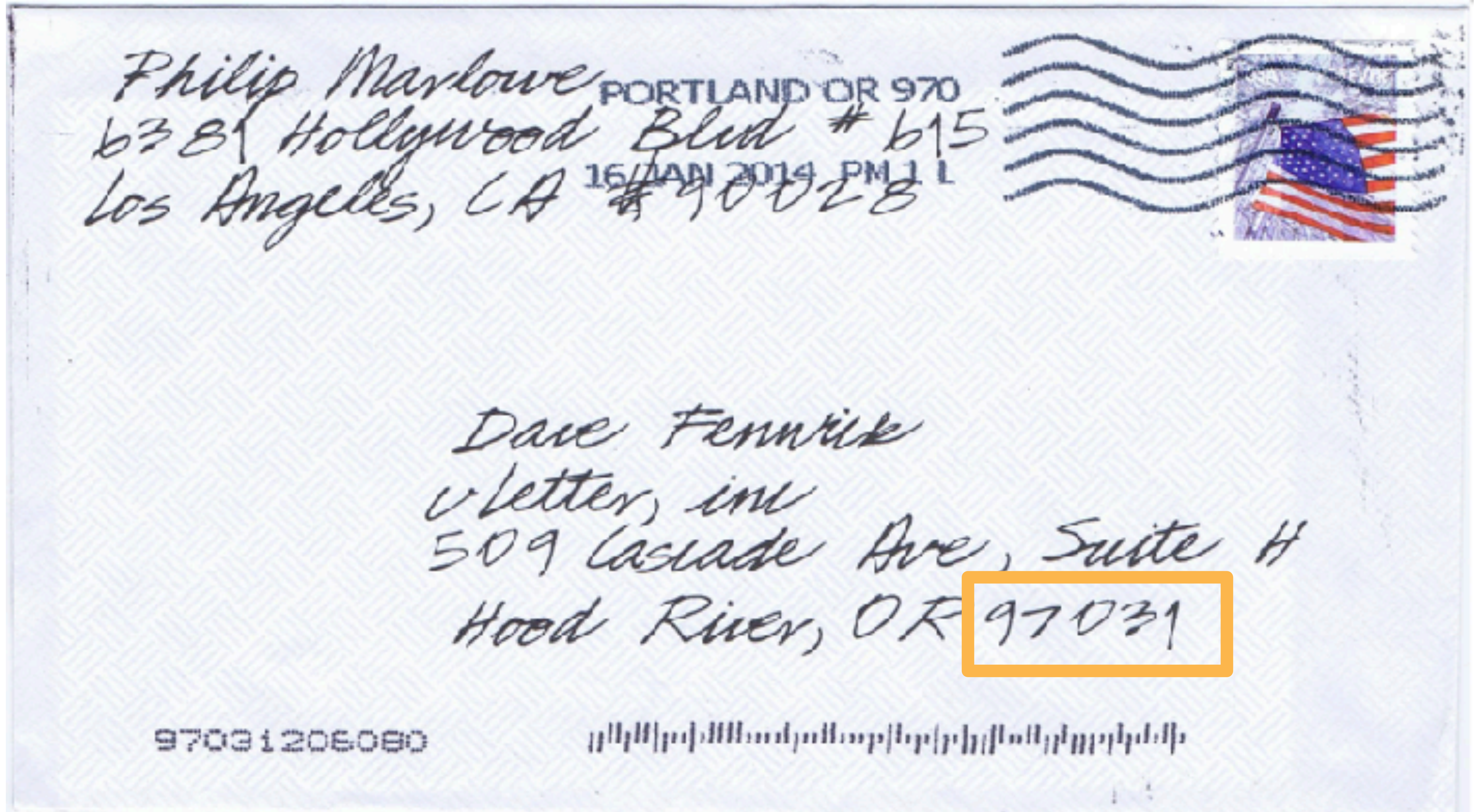


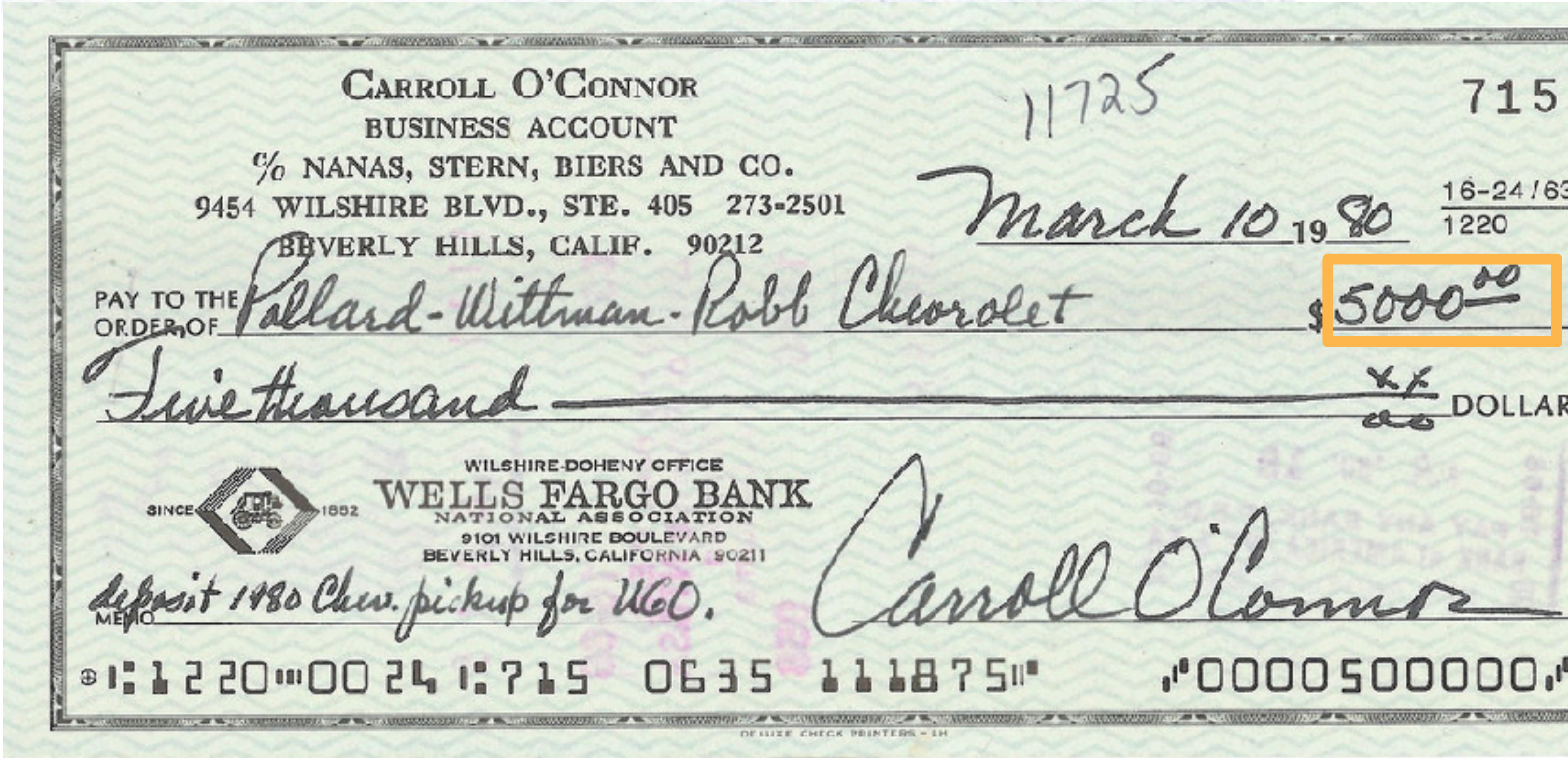


LeNet Architecture

gluon-cv.mxnet.io

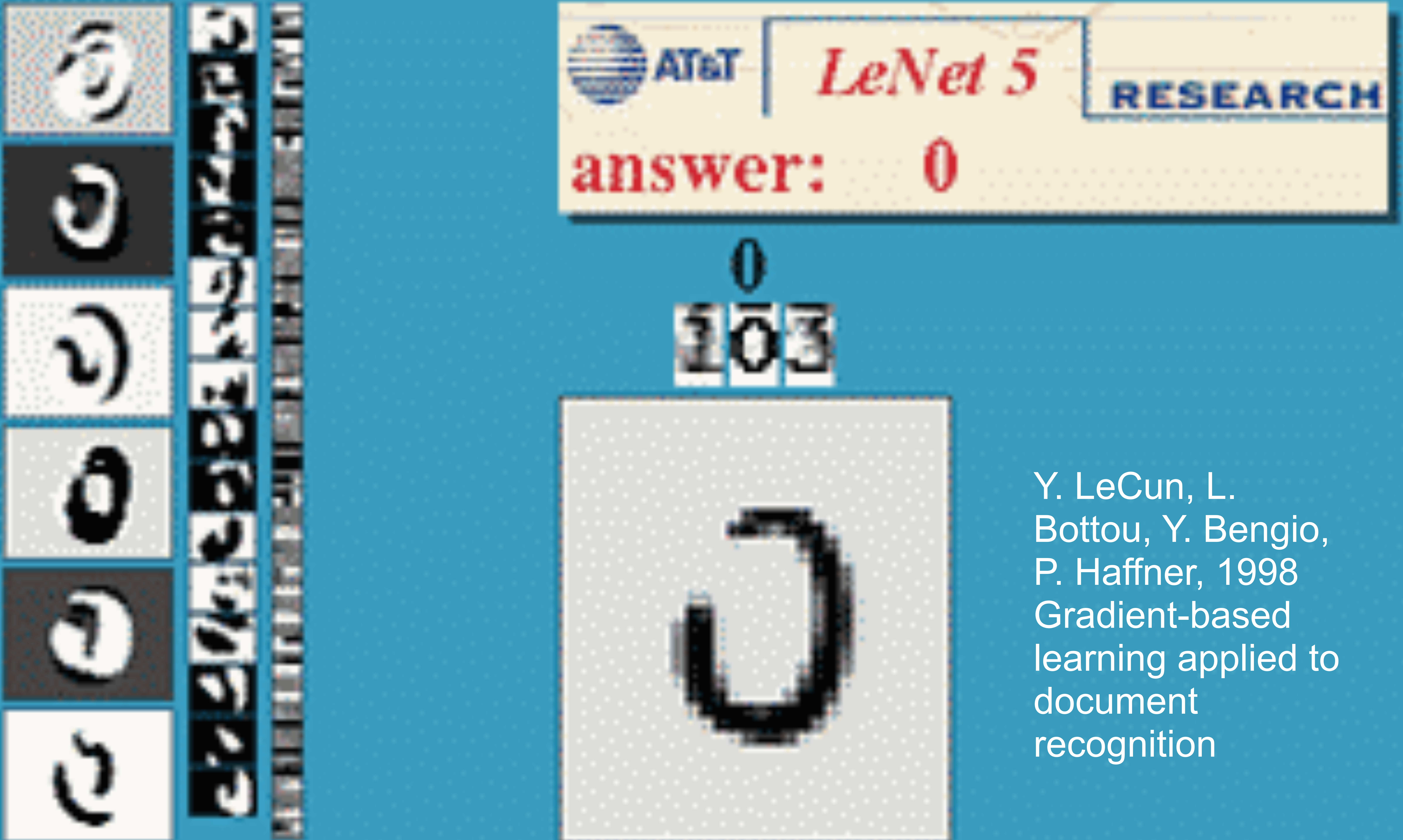
Handwritten Digit Recognition





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

LeNet in Pytorch

```python
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        self.conv1 = nn.Conv2d(1, 6, (5,5), padding=2)
        self.conv2 = nn.Conv2d(6, 16, (5,5))
        self.fc1   = nn.Linear(16*5*5, 120)
        self.fc2   = nn.Linear(120, 84)
        self.fc3   = nn.Linear(84, 10)
```

https://www.kaggle.com/usingtc/lenet-with-pytorch

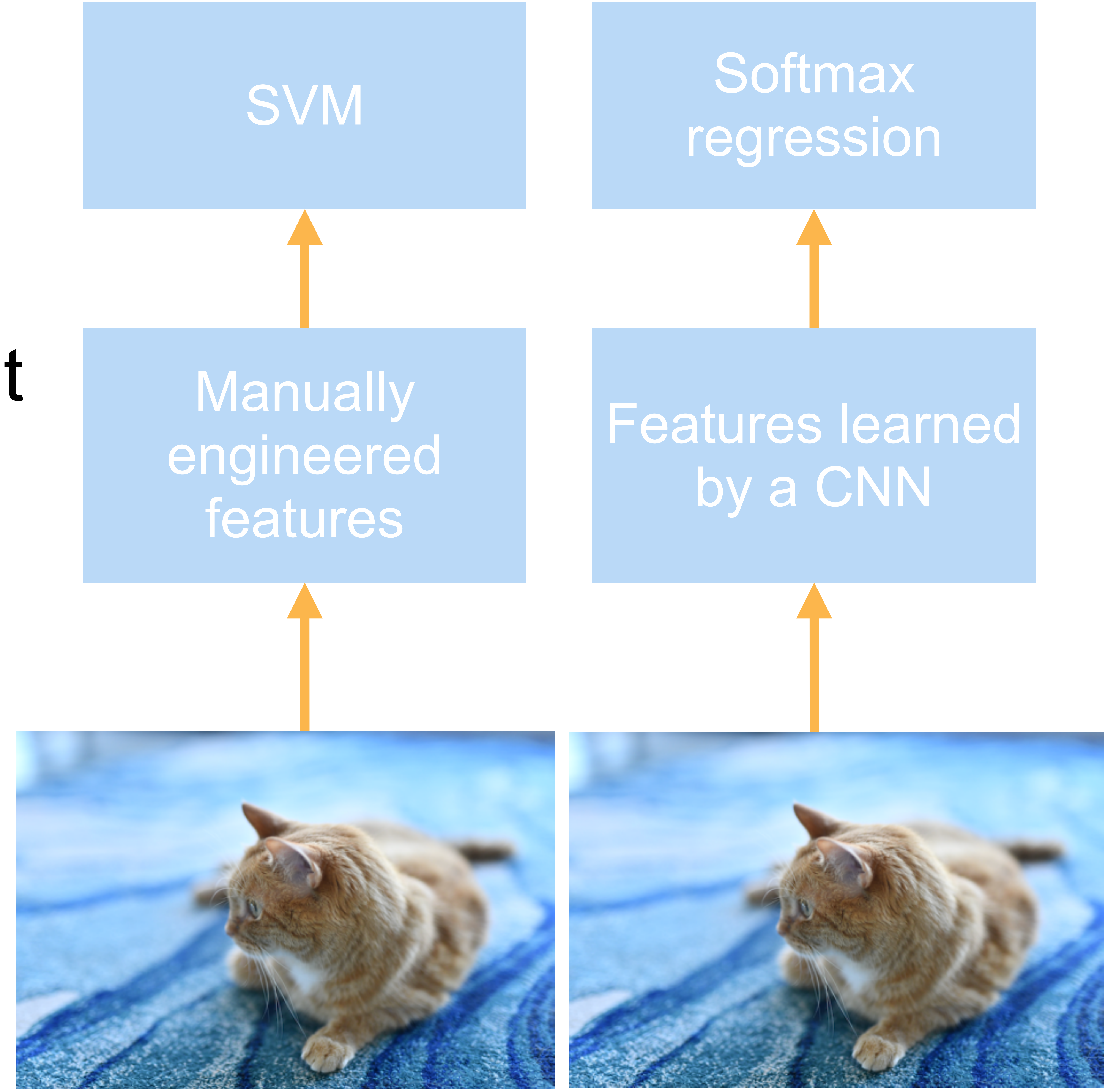
AlexNet

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

Deng et al. 2009

AlexNet

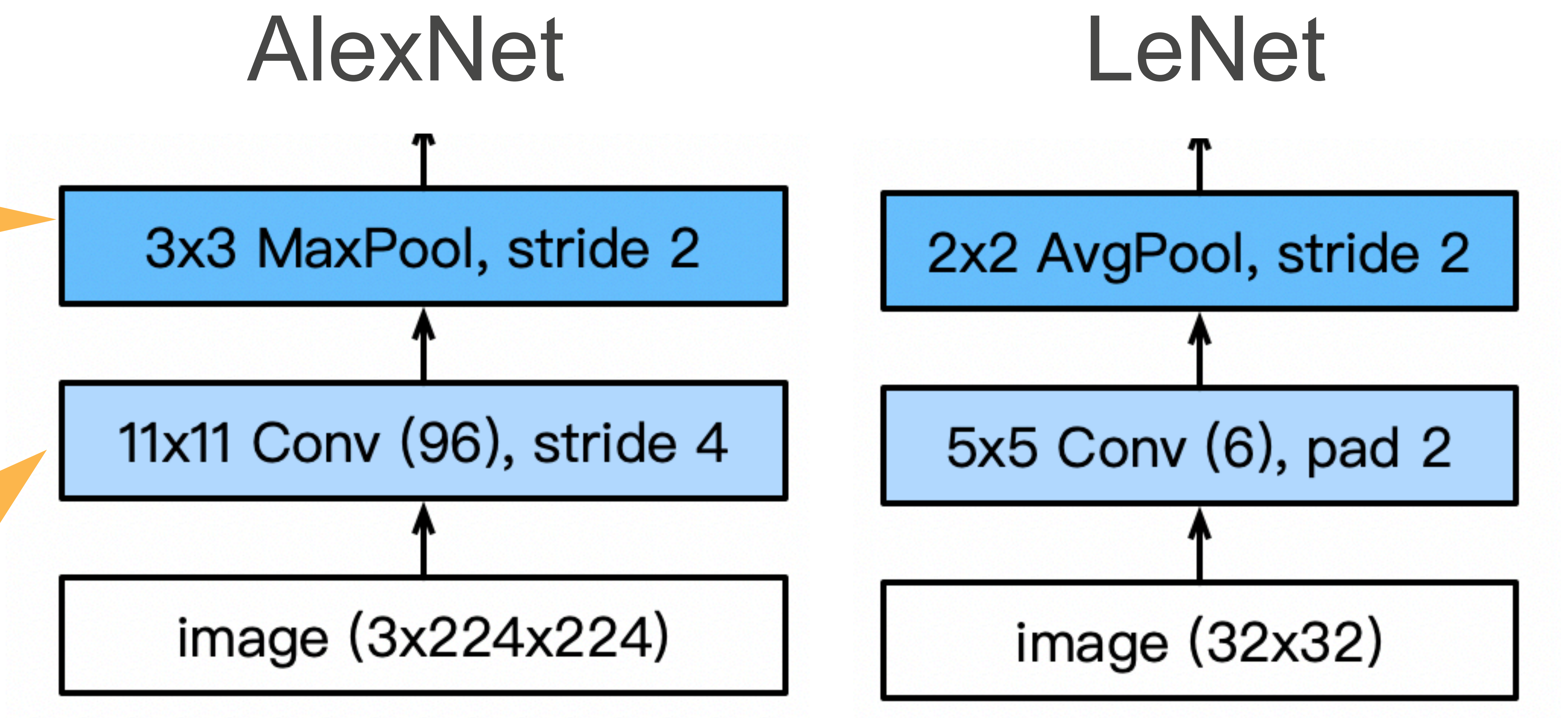
- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Paradigm shift for computer vision



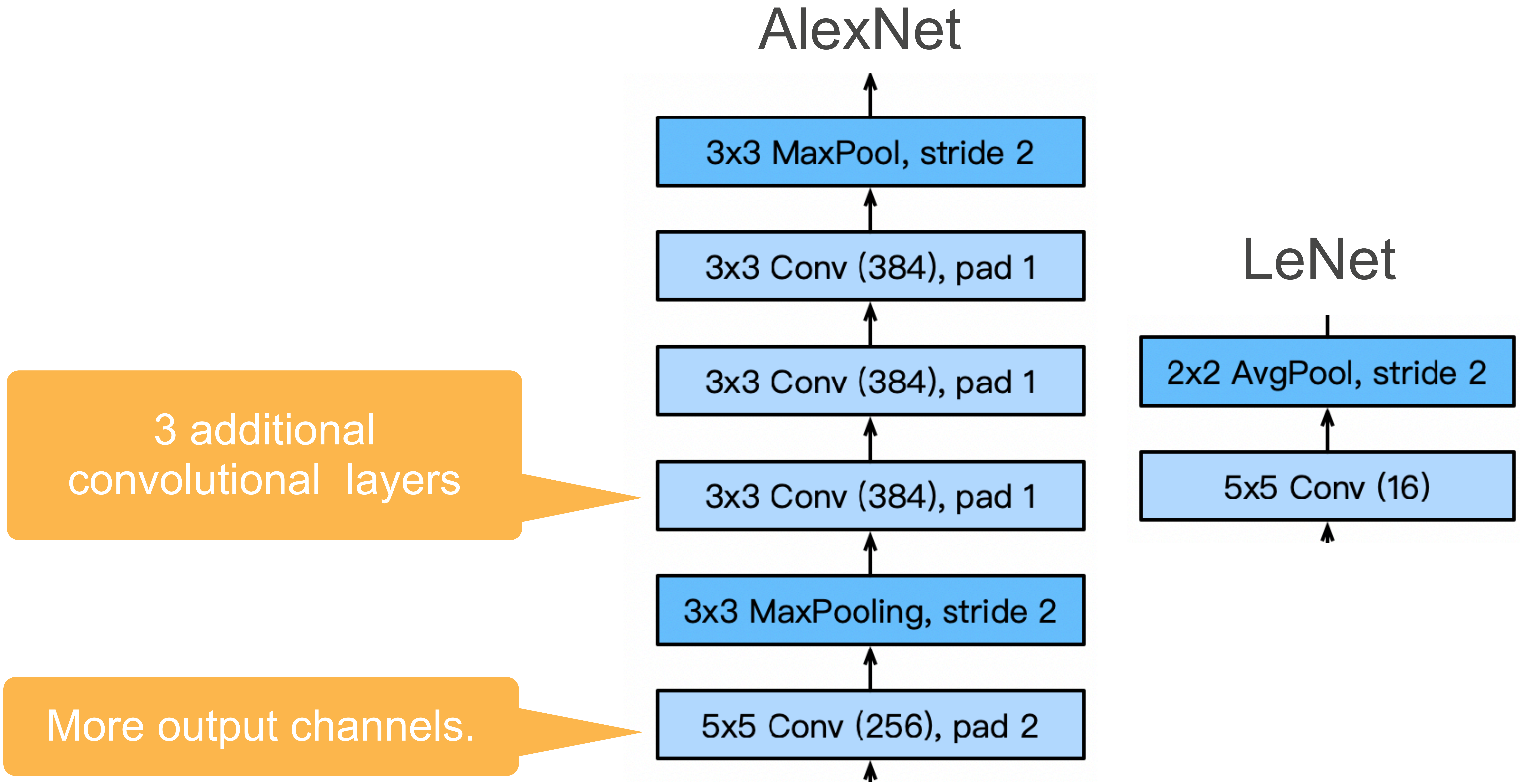
AlexNet Architecture

Larger pool size, change to max pooling

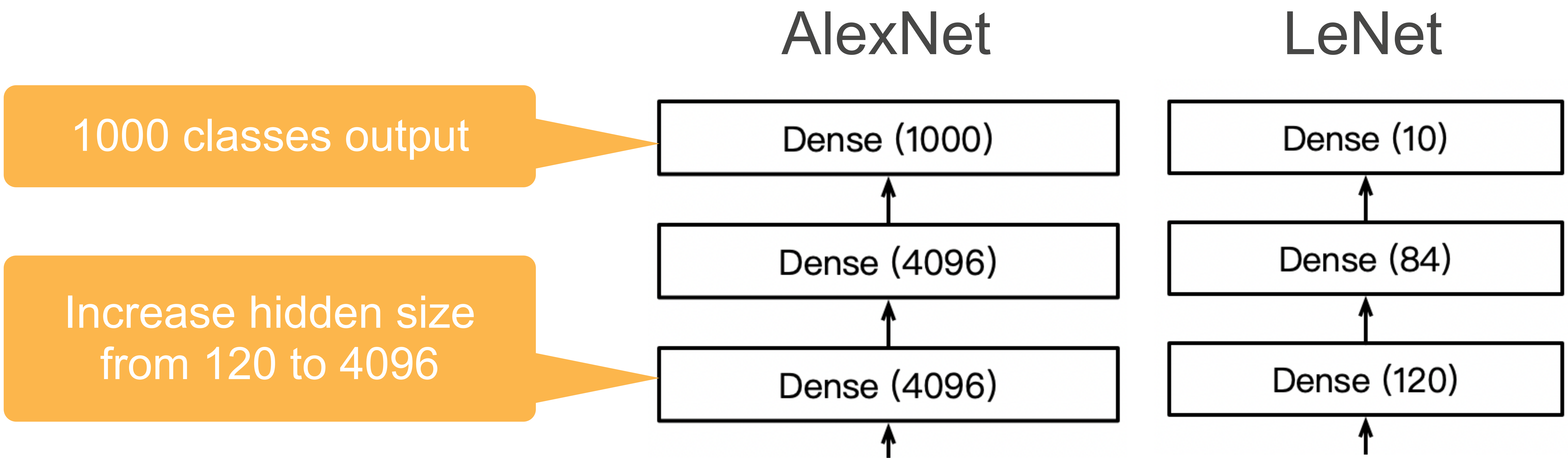
Larger kernel size, stride because of the increased image size, and more output channels.



AlexNet Architecture

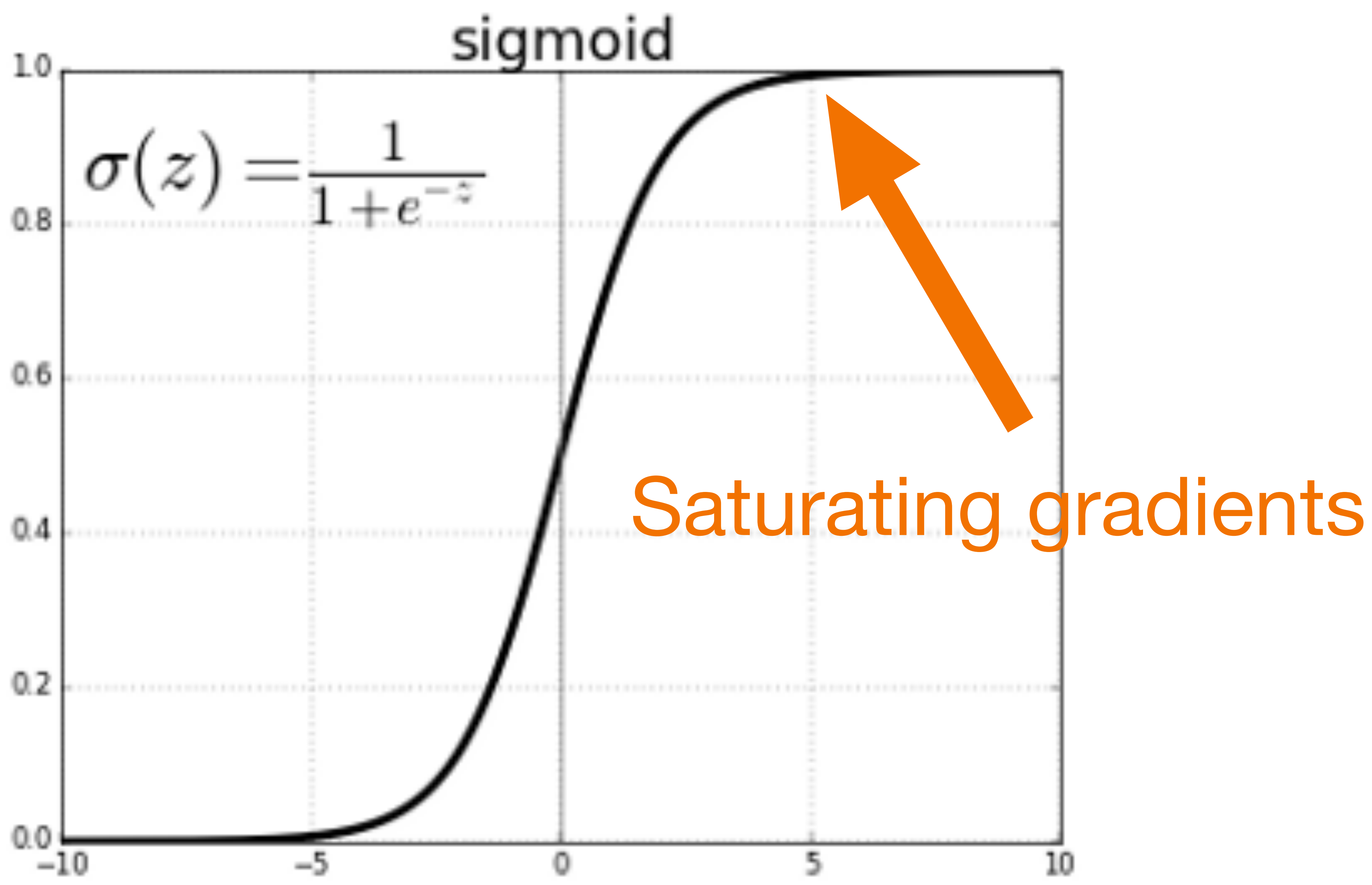


AlexNet Architecture



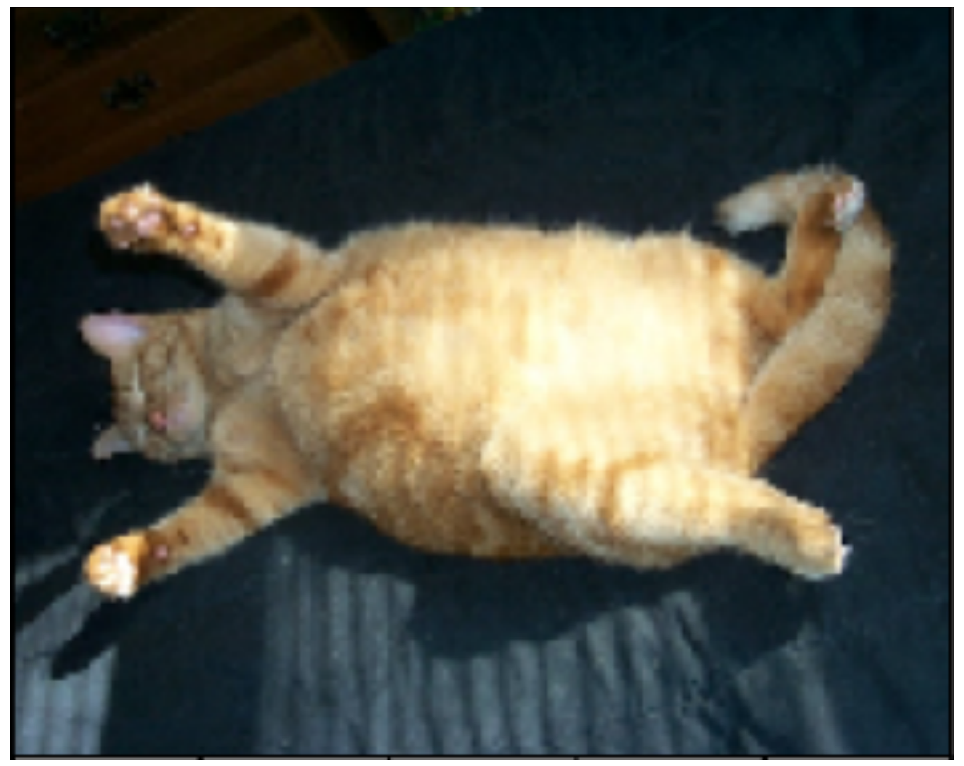
More Differences…

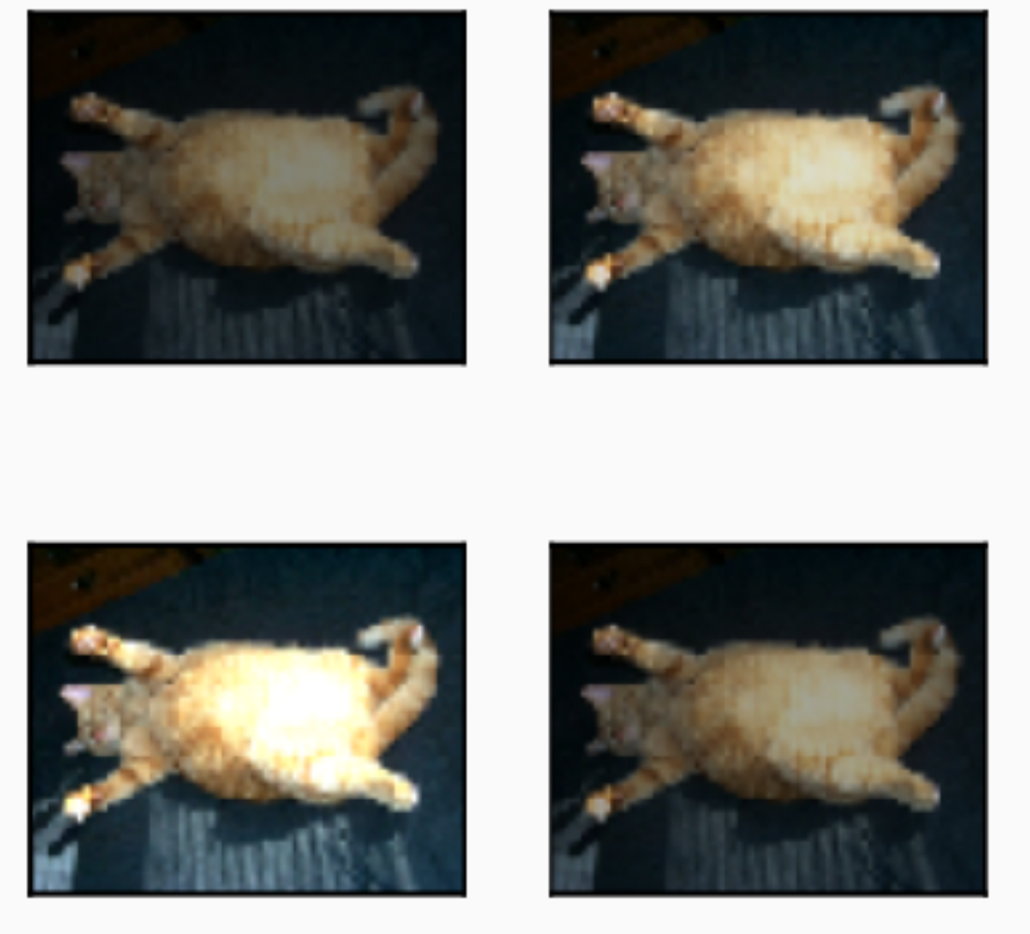
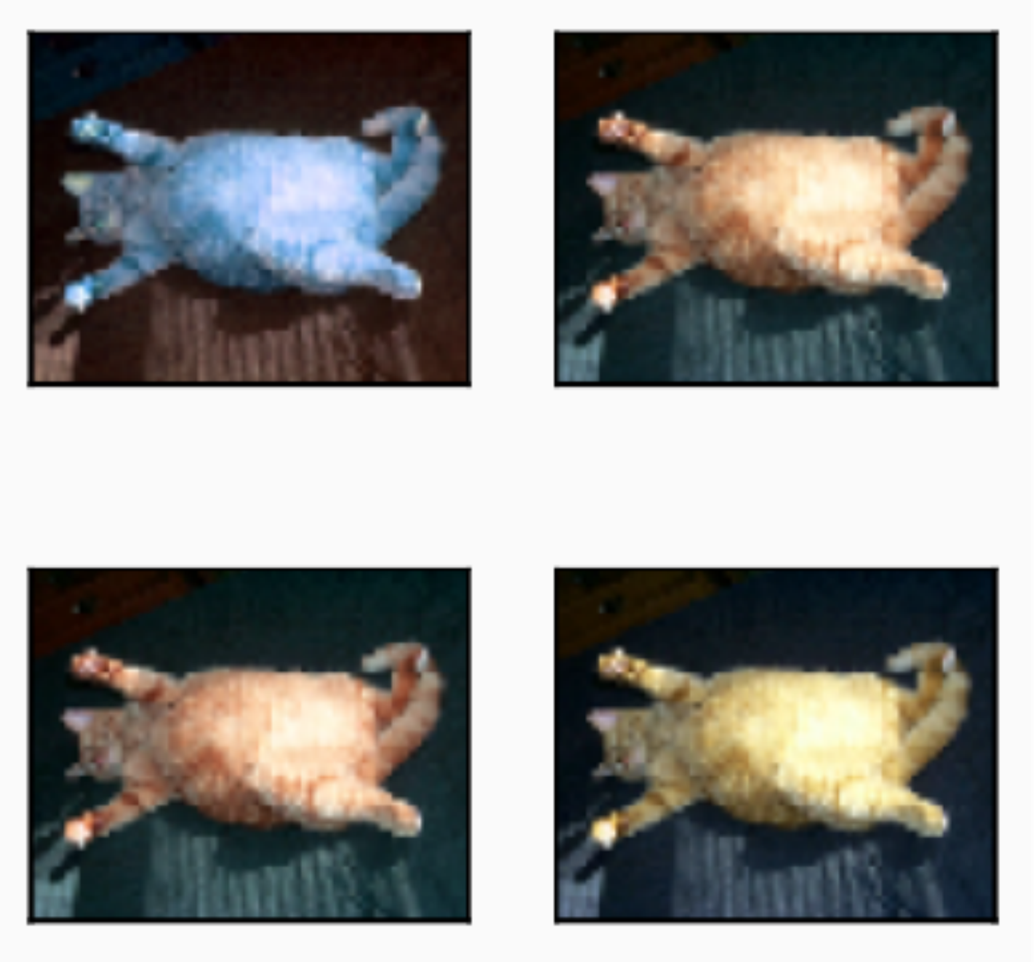
- Change activation function from sigmoid to ReLu (no more vanishing gradient)



More Differences…

- Change activation function from sigmoid to ReLu (no more vanishing gradient)
- Add a dropout layer after two hidden dense layers (better robustness / regularization)
- Data augmentation

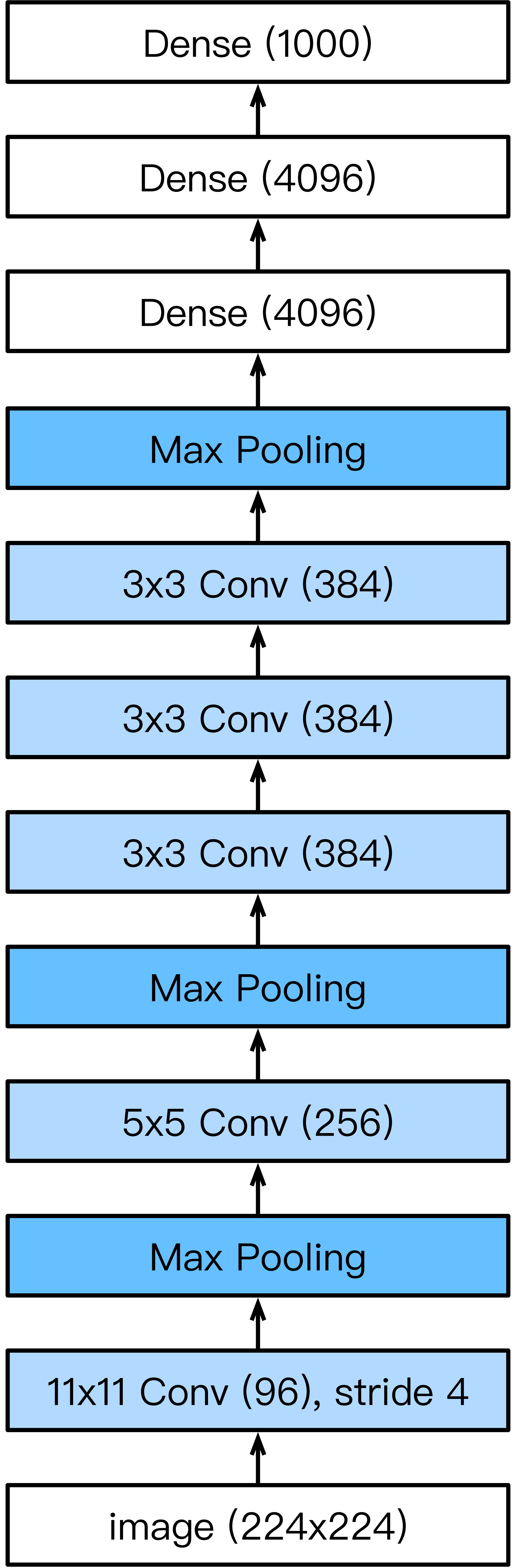


Complexity

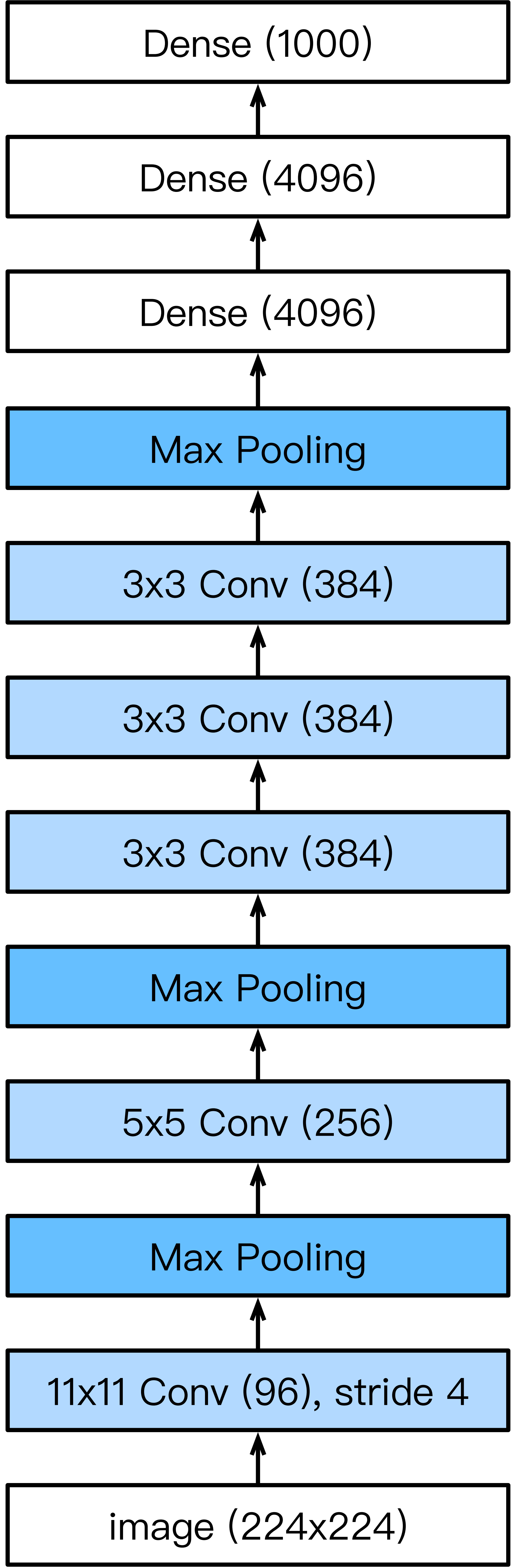
	#parameters	
	AlexNet	**LeNet**
Conv1	35K	150
Conv2	614K	2.4K
Conv3-5	3M	
Dense1	26M	0.48M
Dense2	16M	0.1M
Total	46M	0.6M
Increase	11x	1x

11x11x3x96=35k

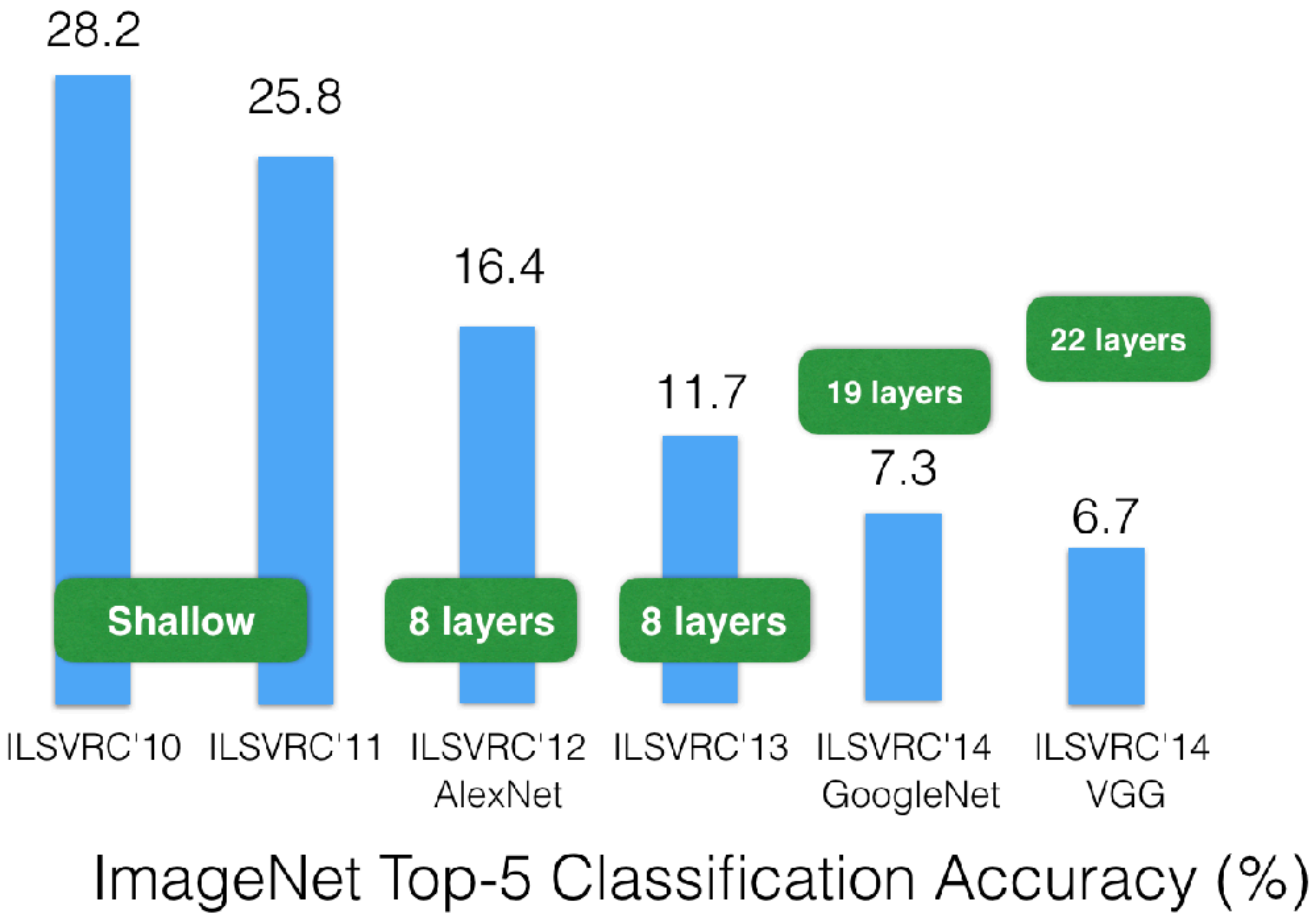


Complexity

	#parameters		FLOP	
	AlexNet	LeNet	AlexNet	LeNet
Conv1	35K	150	101M	1.2M
Conv2	614K	2.4K	415M	2.4M
Conv3-5	3M		445M	
Dense1	26M	0.48M	26M	0.48M
Dense2	16M	0.1M	16M	0.1M
Total	46M	0.6M	1G	4M
Increase	11x	1x	250x	1x

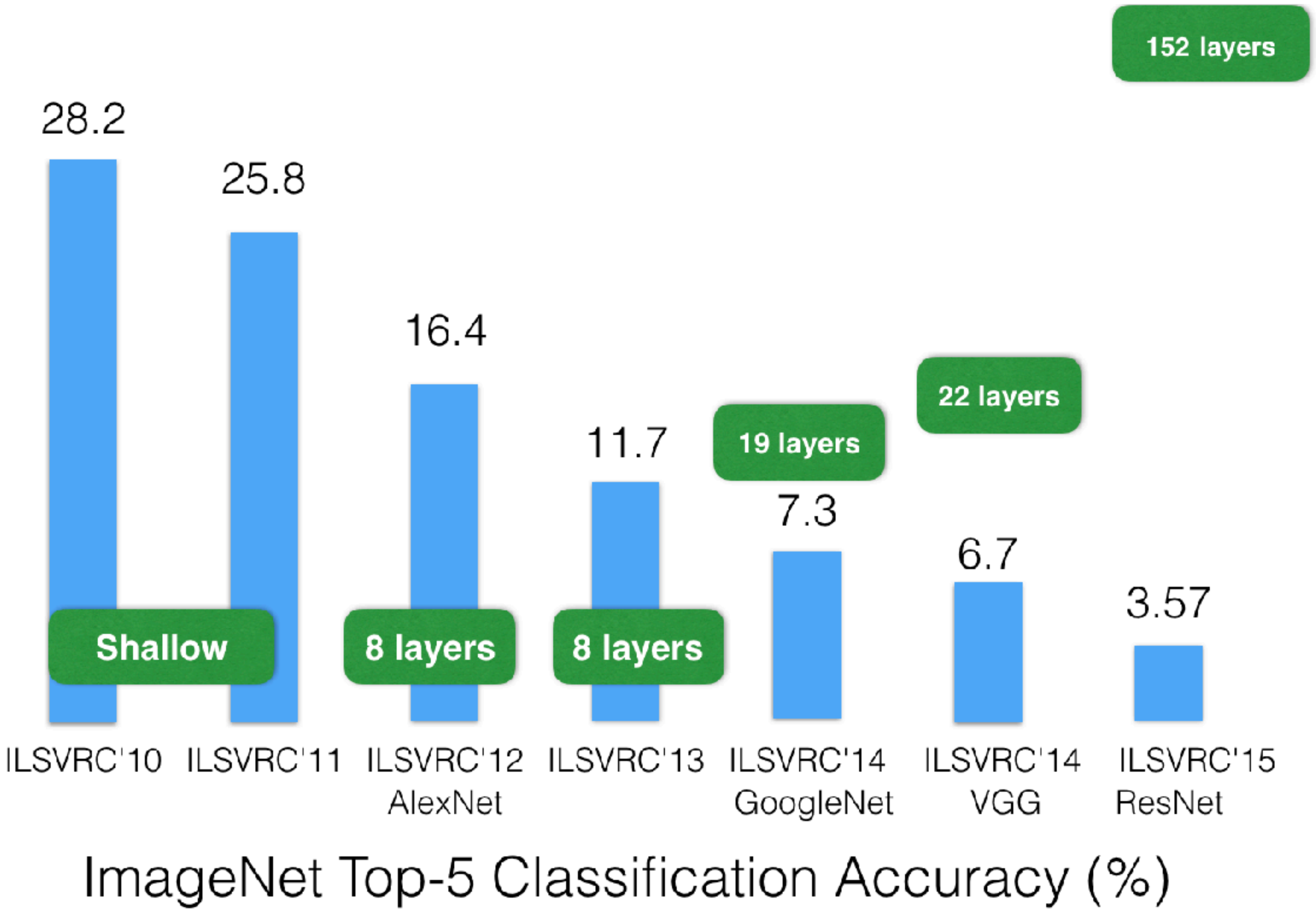


ResNet: Going deeper in depth



[He et al. 2015]

ResNet: Going deeper in depth



[He et al. 2015]

ResNet

[He et al., 2015]

- What happens when we simply stack more and more layers on a “plain” convolutional neural networks?

(Figure from CS231n course slides)

Deeper models are harder to optimize.

ResNet

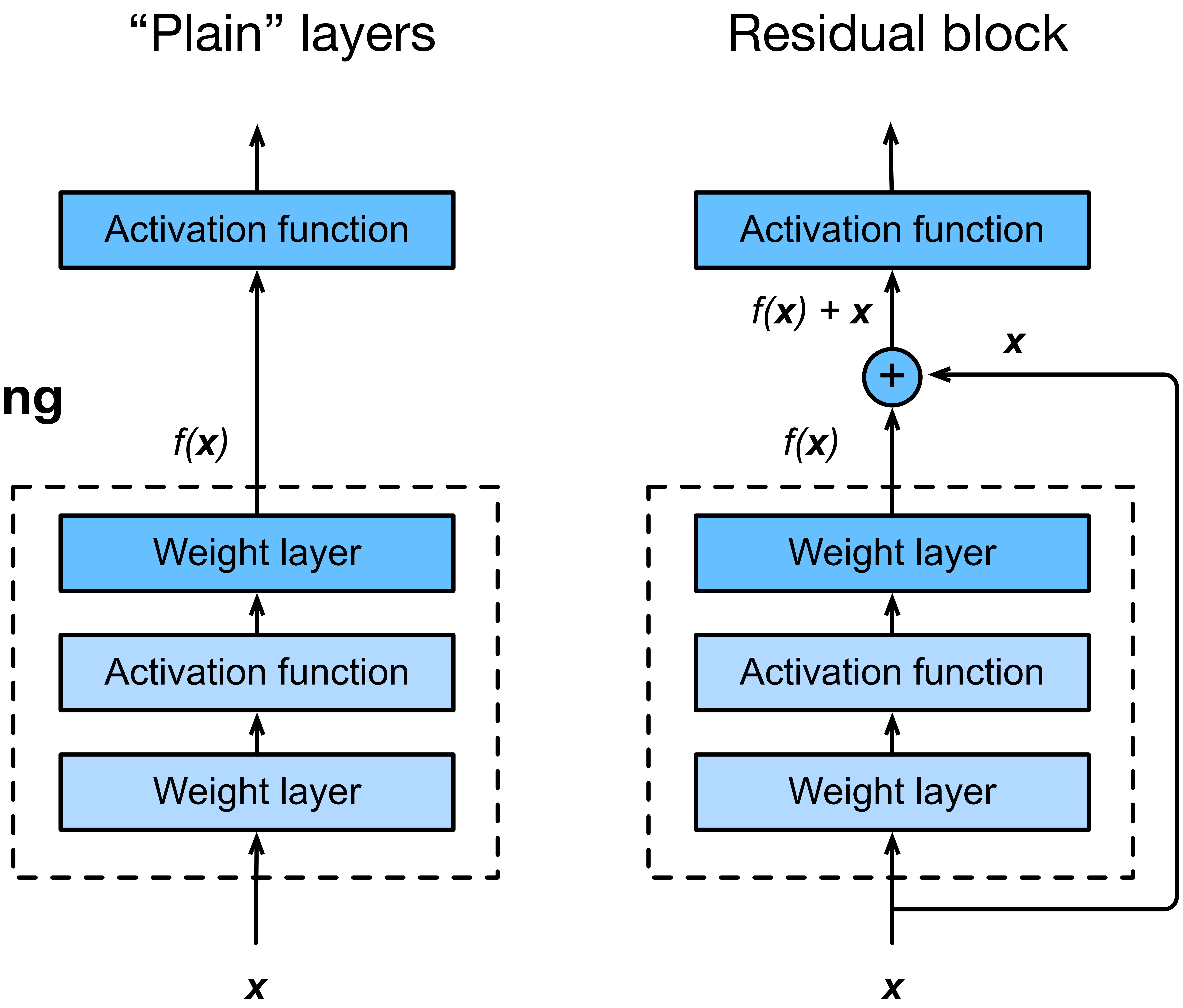
Solution:

Copy the learned layers from the shallower model and setting additional layers to **identity mapping**

$$H(x) = x + f(x)$$

Use layers to fit residual

$f(x) = H(x) - x$ Instead of $H(x)$

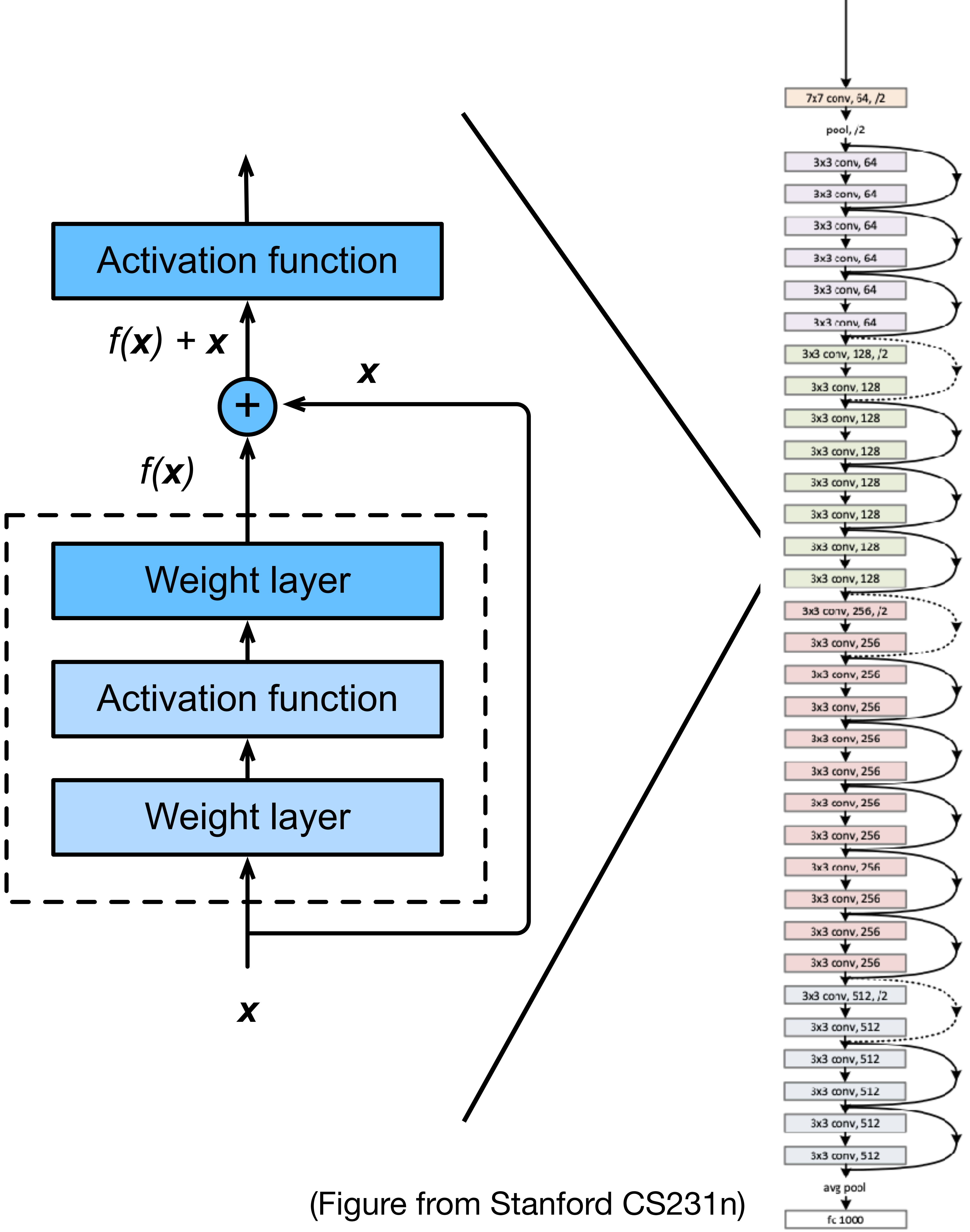


[He et al., 2015]

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				
		$\left[\begin{array}{c}3\times3, 64\\3\times3, 64\end{array}\right]\times2$	$\left[\begin{array}{c}3\times3, 64\\3\times3, 64\end{array}\right]\times3$	$\left[\begin{array}{c}1\times1, 64\\3\times3, 64\\1\times1, 256\end{array}\right]\times3$	$\left[\begin{array}{c}1\times1, 64\\3\times3, 64\\1\times1, 256\end{array}\right]\times3$	$\left[\begin{array}{c}1\times1, 64\\3\times3, 64\\1\times1, 256\end{array}\right]\times3$
conv3_x	28×28	$\left[\begin{array}{c}3\times3, 128\\3\times3, 128\end{array}\right]\times2$	$\left[\begin{array}{c}3\times3, 128\\3\times3, 128\end{array}\right]\times4$	$\left[\begin{array}{c}1\times1, 128\\3\times3, 128\\1\times1, 512\end{array}\right]\times4$	$\left[\begin{array}{c}1\times1, 128\\3\times3, 128\\1\times1, 512\end{array}\right]\times4$	$\left[\begin{array}{c}1\times1, 128\\3\times3, 128\\1\times1, 512\end{array}\right]\times8$
conv4_x	14×14	$\left[\begin{array}{c}3\times3, 256\\3\times3, 256\end{array}\right]\times2$	$\left[\begin{array}{c}3\times3, 256\\3\times3, 256\end{array}\right]\times6$	$\left[\begin{array}{c}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{array}\right]\times6$	$\left[\begin{array}{c}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{array}\right]\times23$	$\left[\begin{array}{c}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{array}\right]\times36$
conv5_x	7×7	$\left[\begin{array}{c}3\times3, 512\\3\times3, 512\end{array}\right]\times2$	$\left[\begin{array}{c}3\times3, 512\\3\times3, 512\end{array}\right]\times3$	$\left[\begin{array}{c}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{array}\right]\times3$	$\left[\begin{array}{c}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{array}\right]\times3$	$\left[\begin{array}{c}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{array}\right]\times3$
	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. Down-sampling 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

Repeat x4 times

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

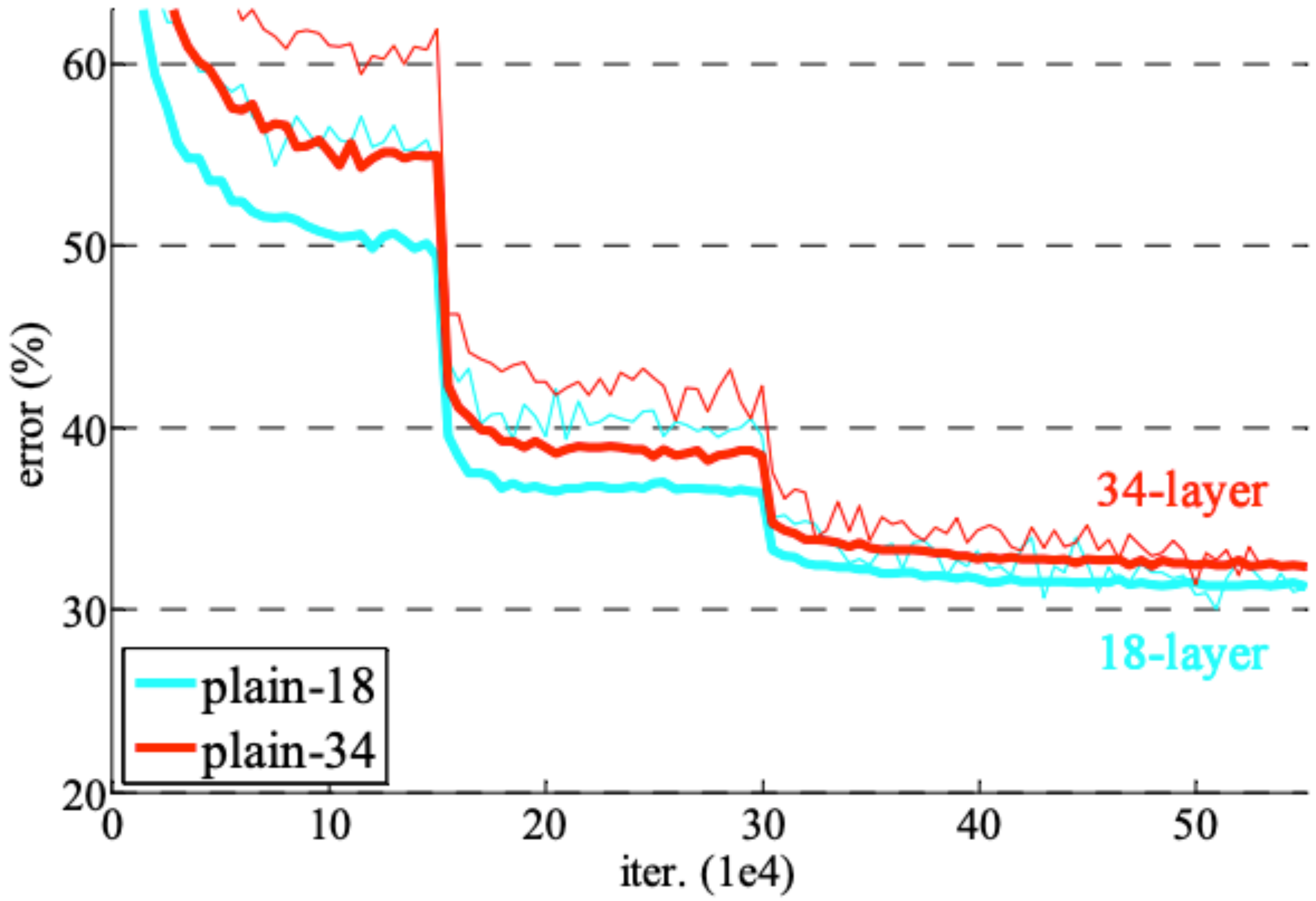
1+ 2x3 + 2x4 + 2x6 + 2x3 +1 = 34

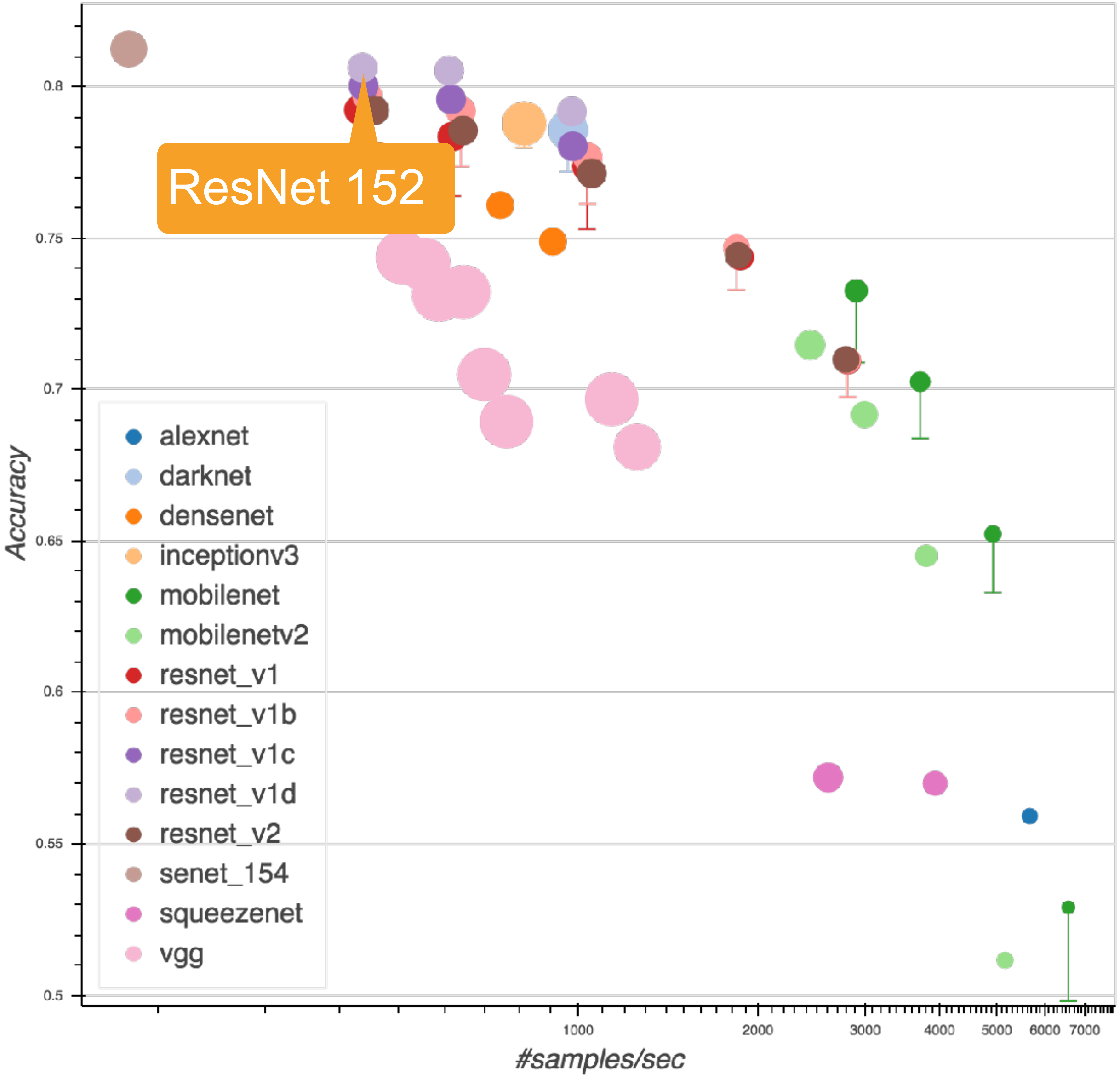
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
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. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

ResNet Training Curves on ImageNet

[He et al., 2015]





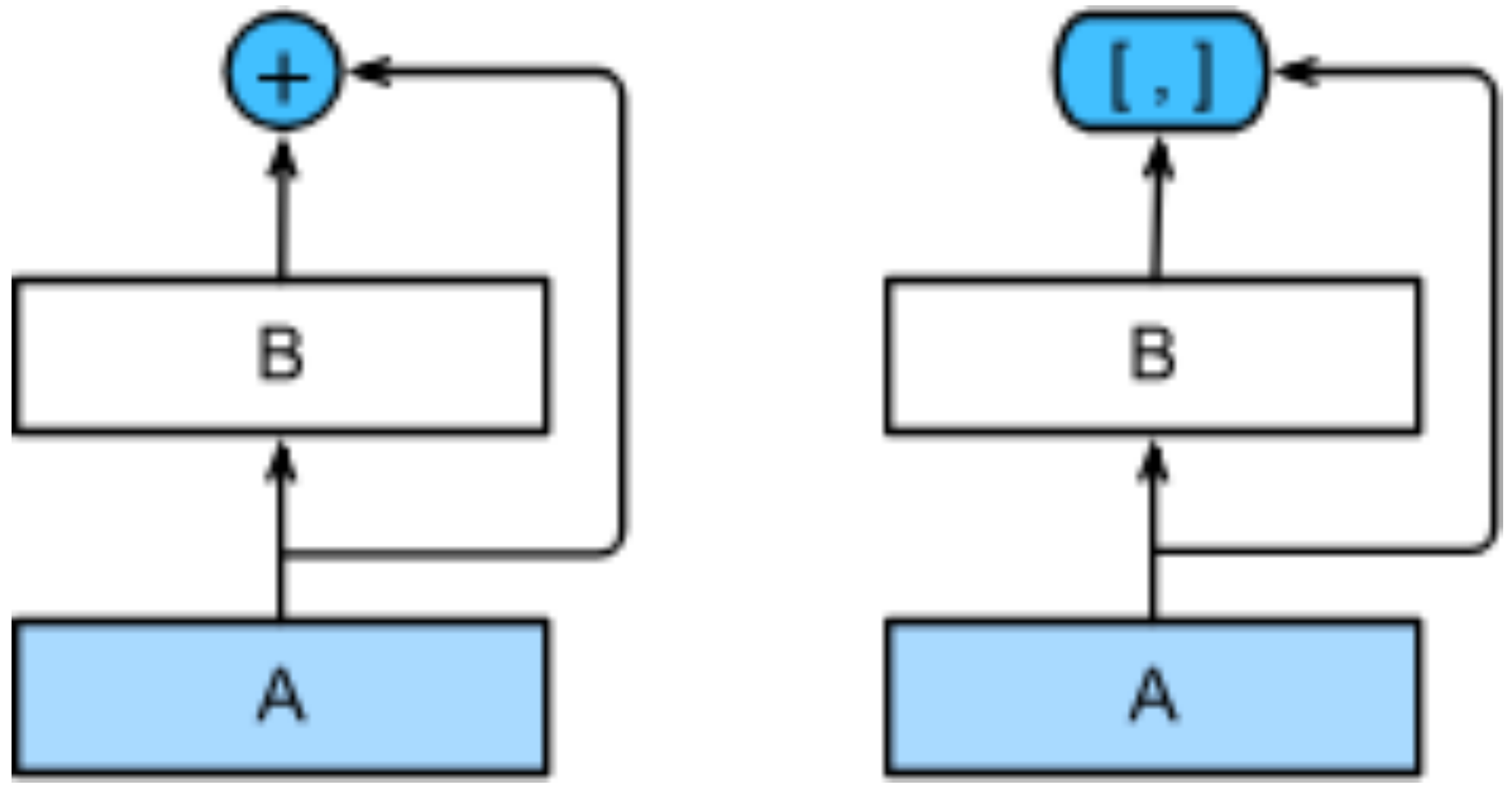
GluonCV Model Zoo

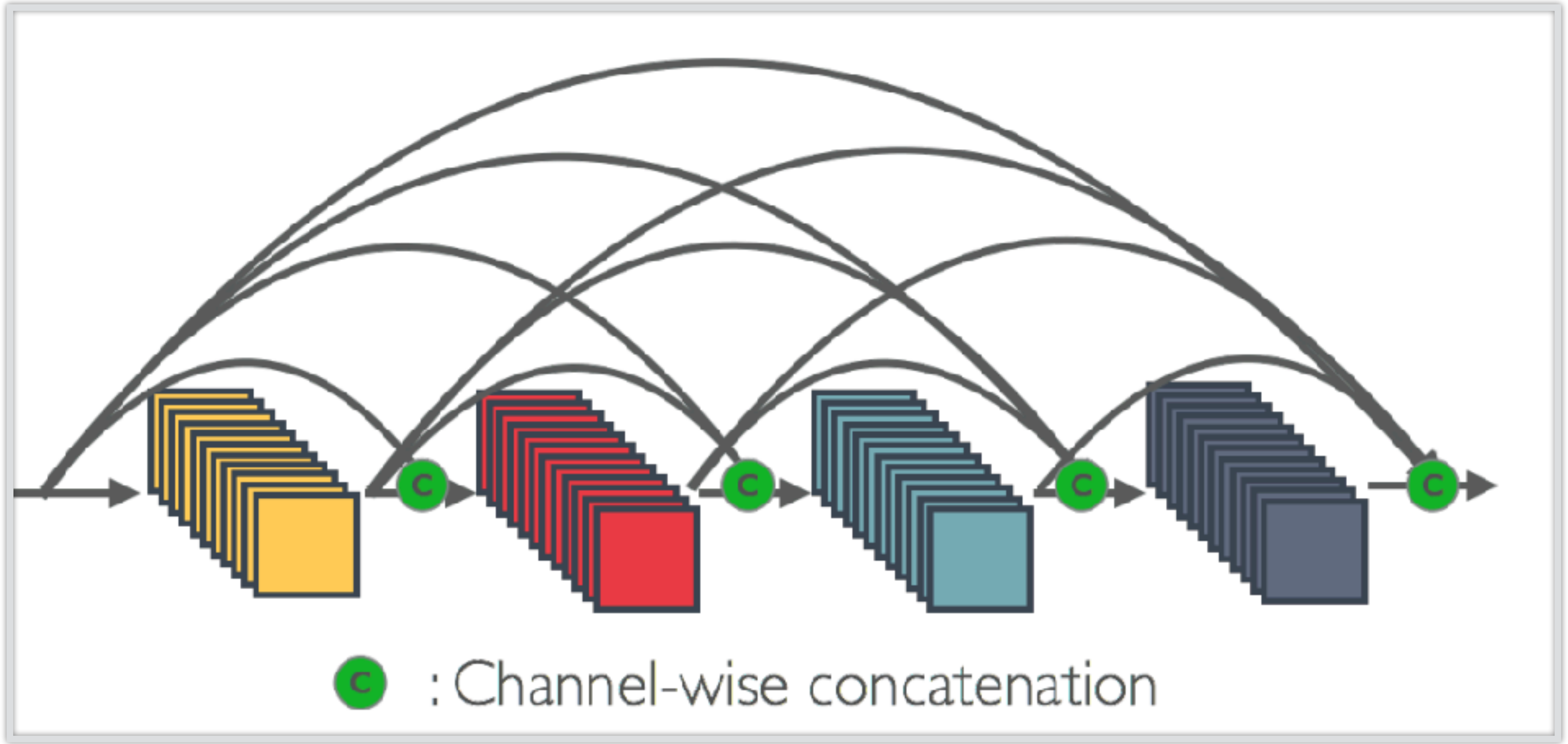
https://gluon-cv.mxnet.io/model_zoo/classification.html

More Ideas

DenseNet (Huang et al., 2016)

- ResNet **adds** x and f(x)
- DenseNet **concatenates** x and f(x)



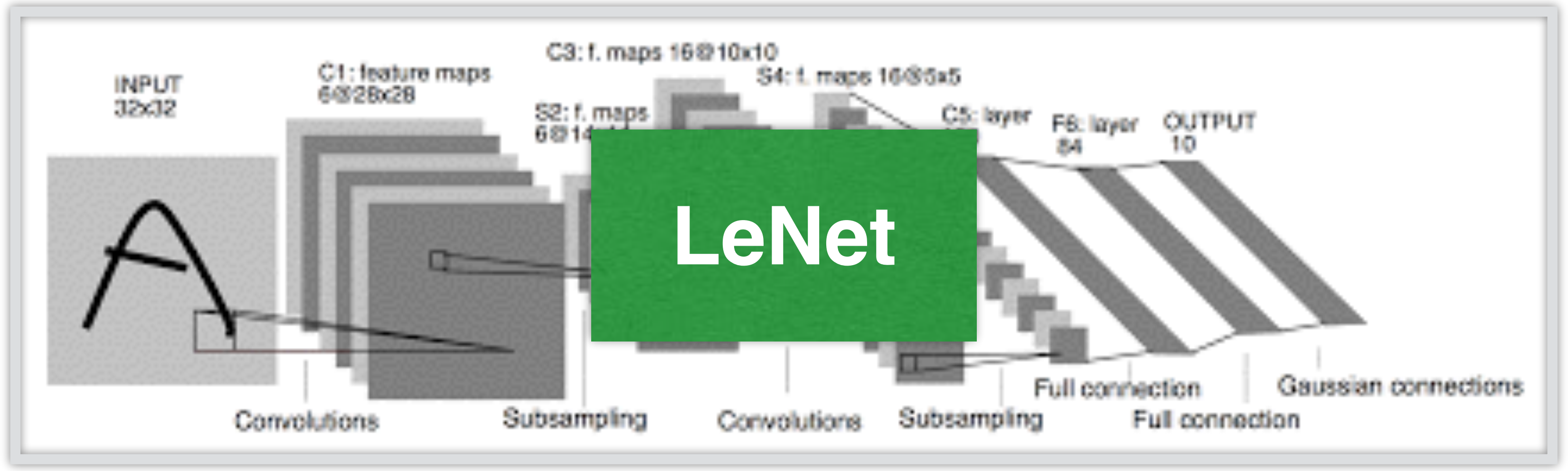


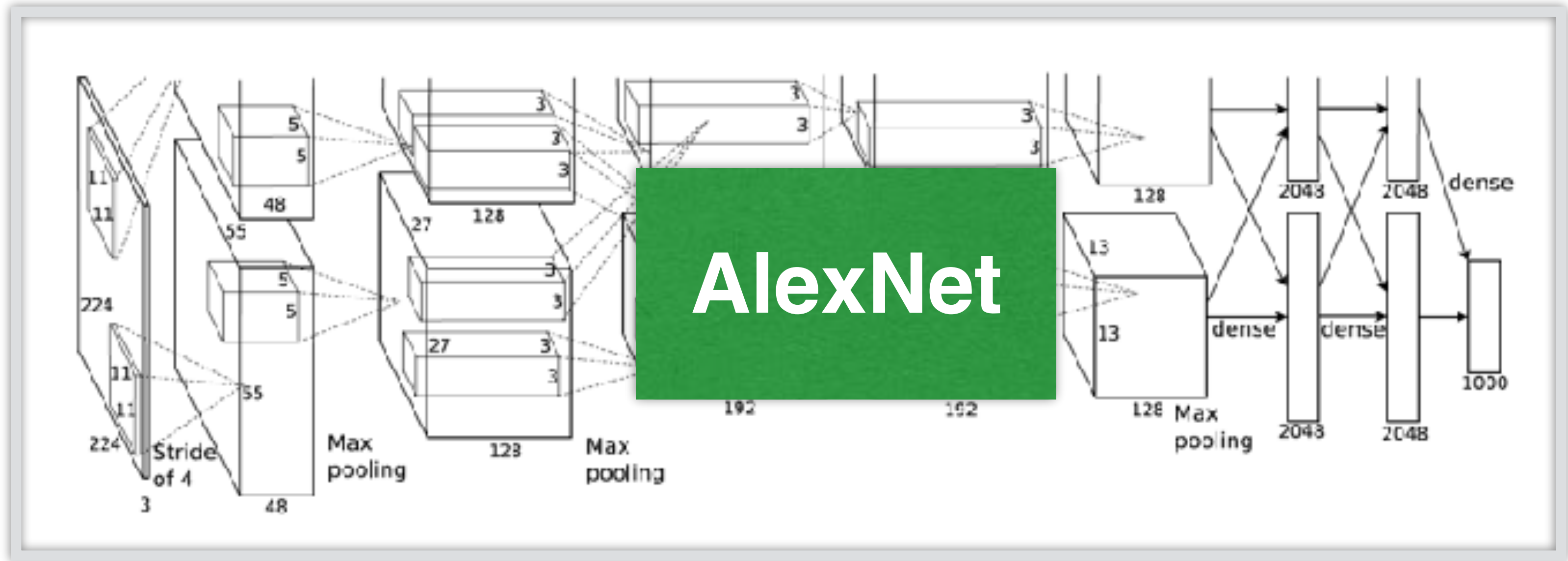
More to discuss next class!

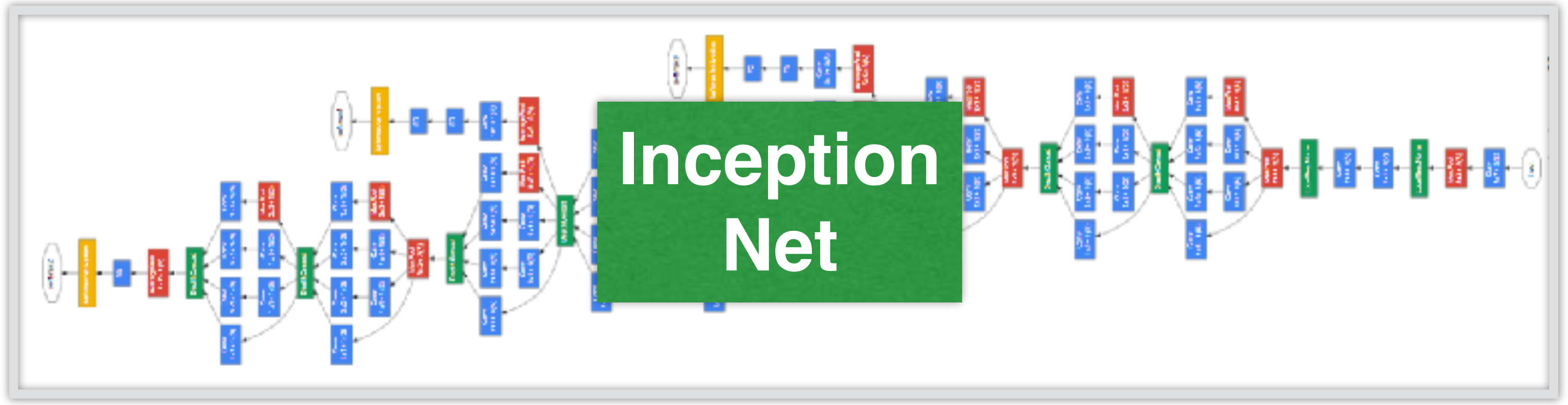
Other architectures to know…

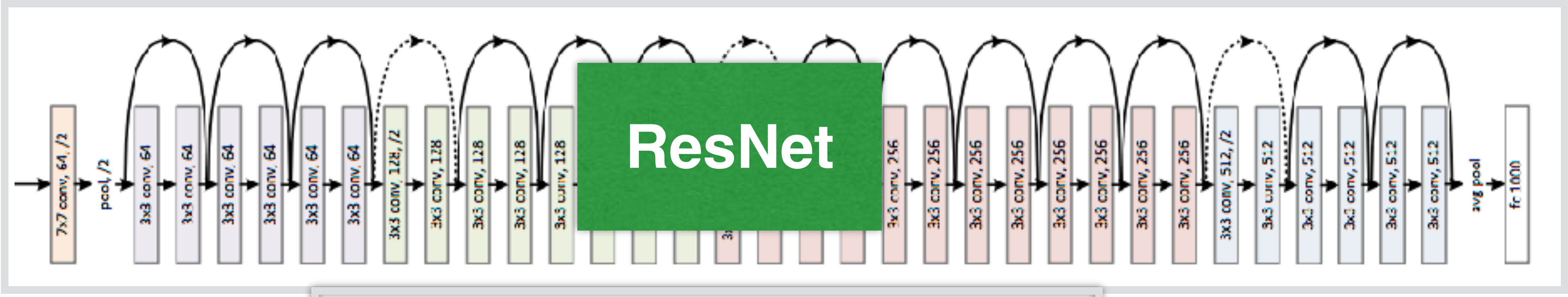
- **ResNeXt** [Xie et al., 2016]
- **Wide ResNet** [Zagoruyko et al. 2016]
- **Deep Networks with Stochastic Depth** [Huang et al. 2016]
- **FractalNet** [Larsson et al. 2017]
- **SqueezeNet** [Iandola et al. 2017]
- **ShuffleNet** [Zhang et al. 2018]

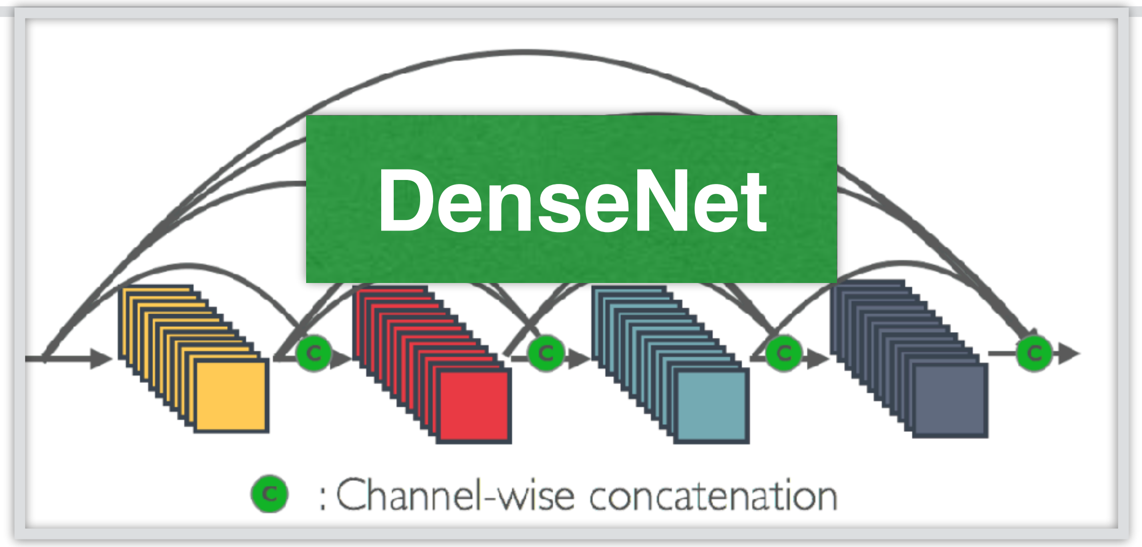
Neural Architecture Search



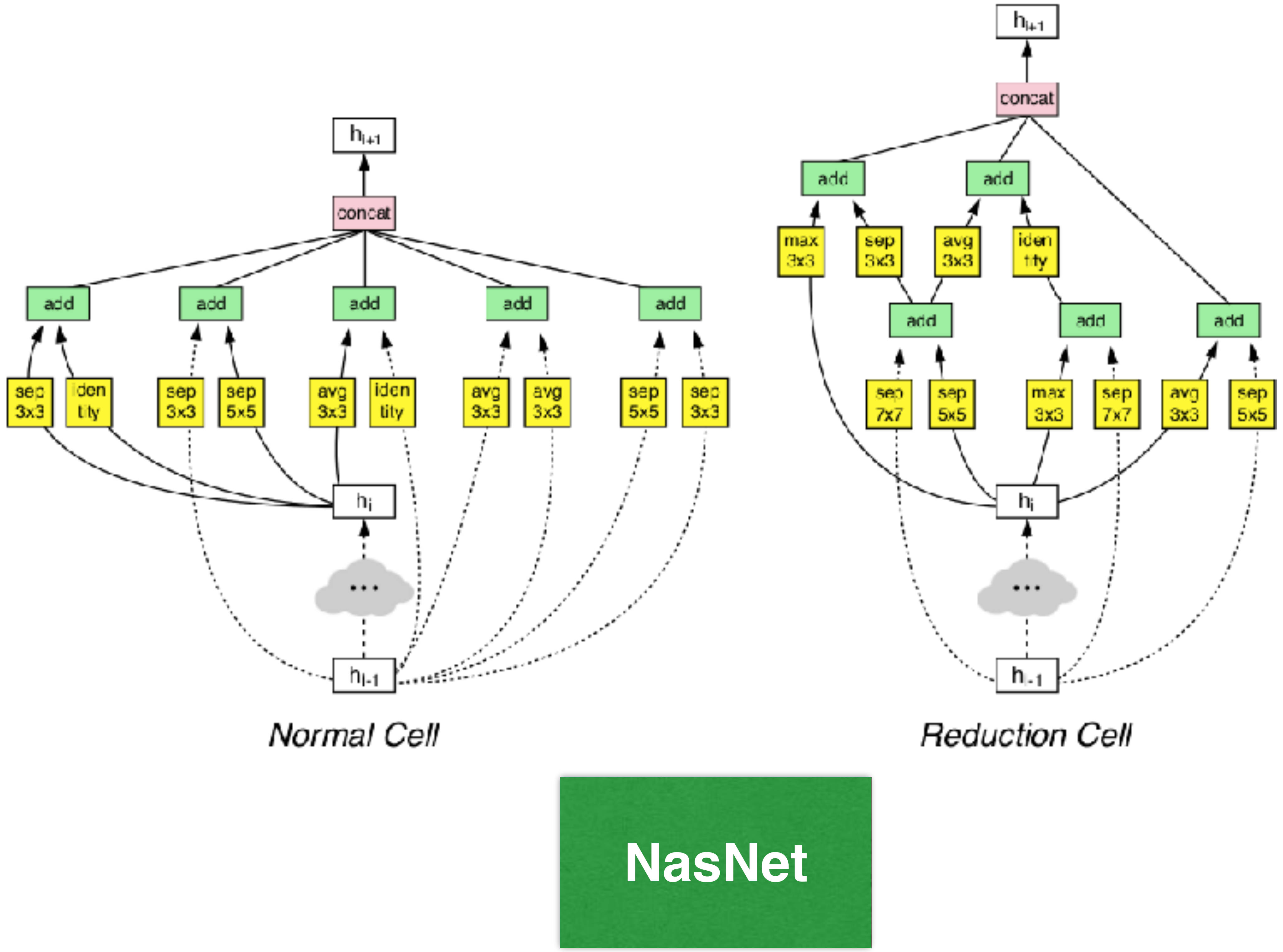








AutoML



[Zoph et al. 2017]

Summary

- Convolution computation
    - 2D conv
    - Padding, stride
    - Multiple input and output channels
- Case study of a few convolutional neural networks
    - **LeNet** (first conv nets)
    - **AlexNet**
    - **ResNet** (trend towards extremely deep networks)

Acknowledgement:

Some of the slides in these lectures have been adapted/borrowed from materials developed by Alex Smola and Mu Li:
https://courses.d2l.ai/berkeley-stat-157/index.html, and
by Fei-Fei Li, Justin Johnson and Serena Young http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf