

# Convolutional Neural Networks

YOUNG & YANDEX

Радослав Нейчев  
Выпускник и преподаватель ШАД и МФТИ,  
руководитель группы ML-разработки в Яндексе,  
основатель  girafe



# Outline

**00** Neural networks recap

**01** Convolutional layer

**02** Pooling layers

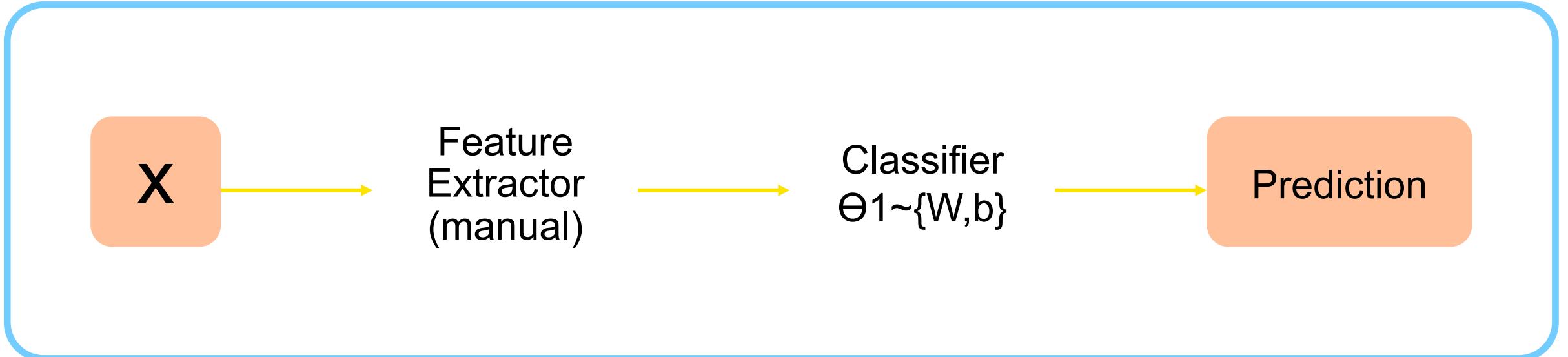
**03** Classification architectures  
(partial) overview

# Neural networks recap

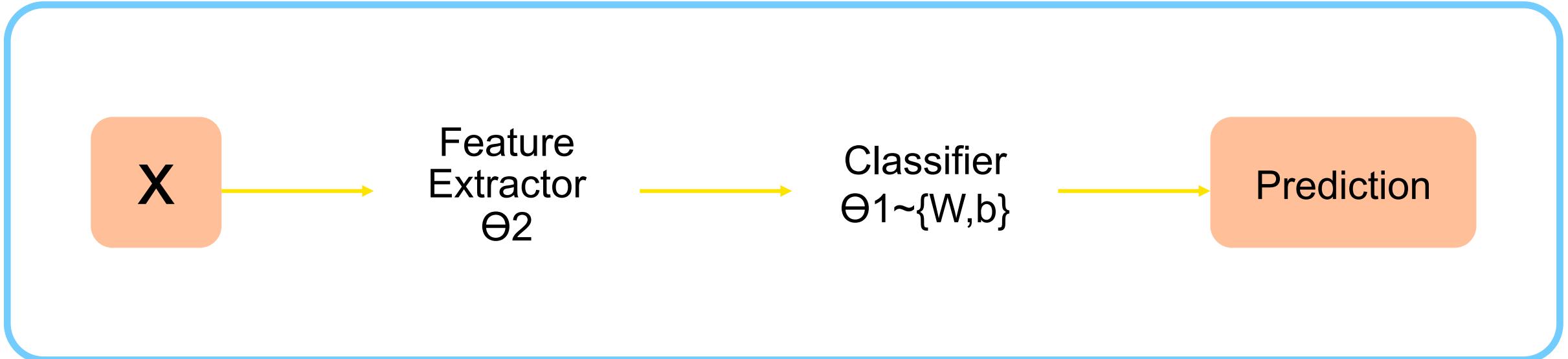
00



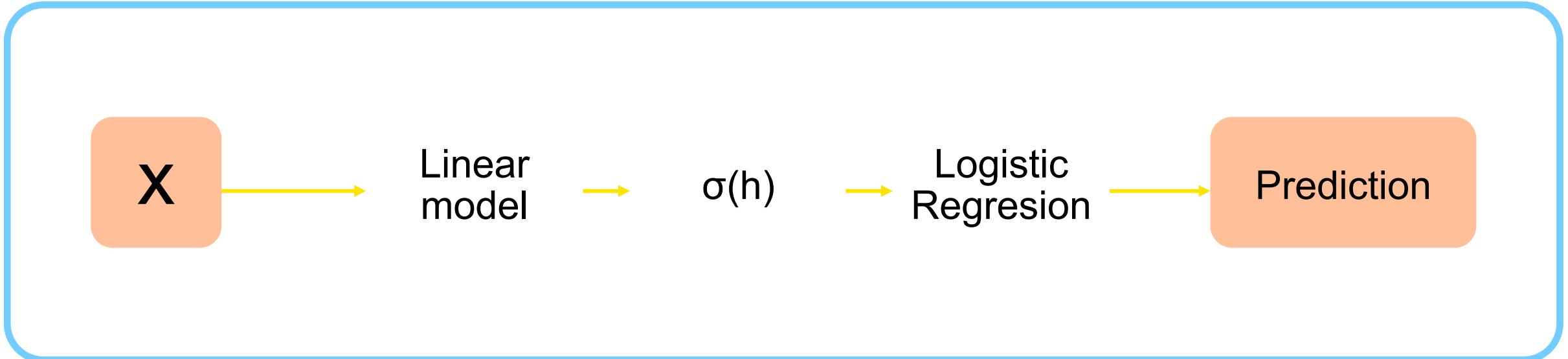
# Classic pipeline



# NN pipeline



# NN pipeline: example



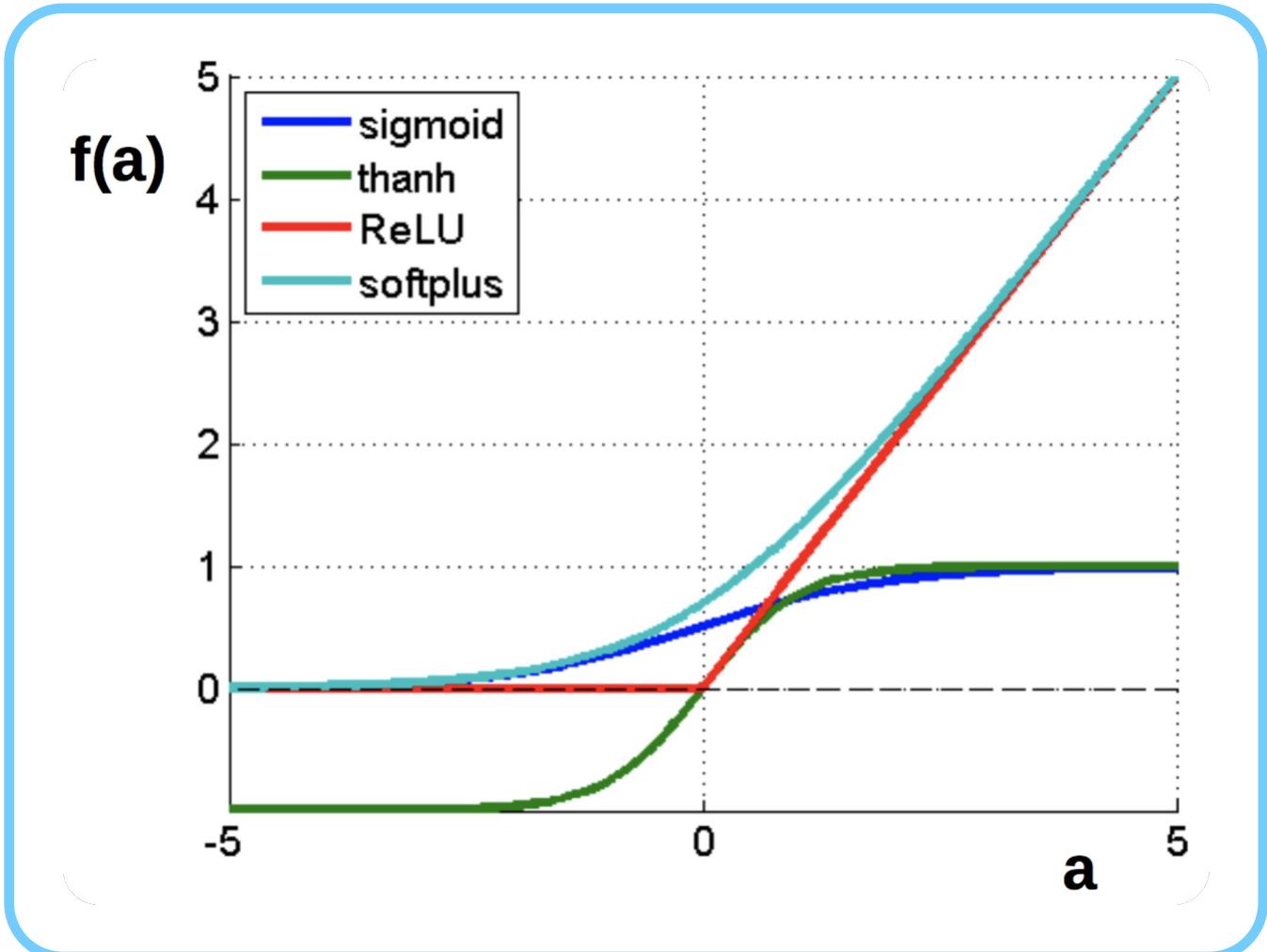
# Activation functions: nonlinearities

$$f(a) = \frac{1}{1 + e^{-a}}$$

$$f(a) = \tanh(a)$$

$$f(a) = \max(0, a)$$

$$f(a) = \log(1 + e^a)$$



# Some generally accepted terms

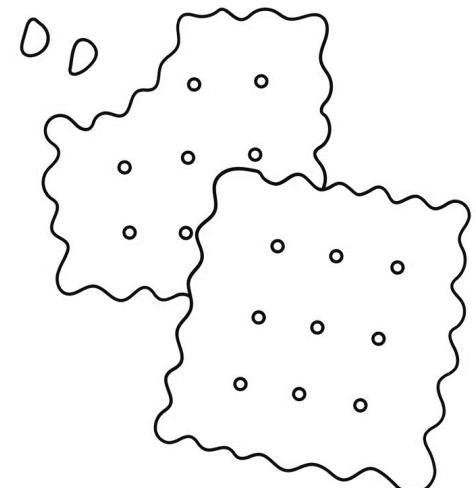
## 01 Layer — a building block for NNs :

- Dense/Linear/FC layer:  $f(x) = Wx+b$
- Nonlinearity layer:  $f(x) = \sigma(x)$
- Input layer, output layer
- A few more we will cover later

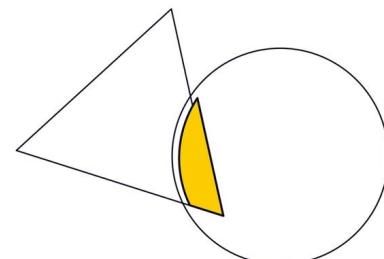
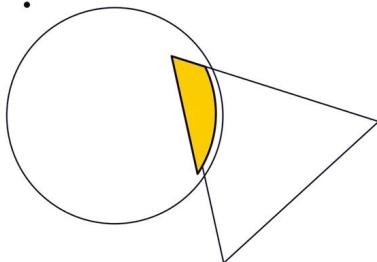
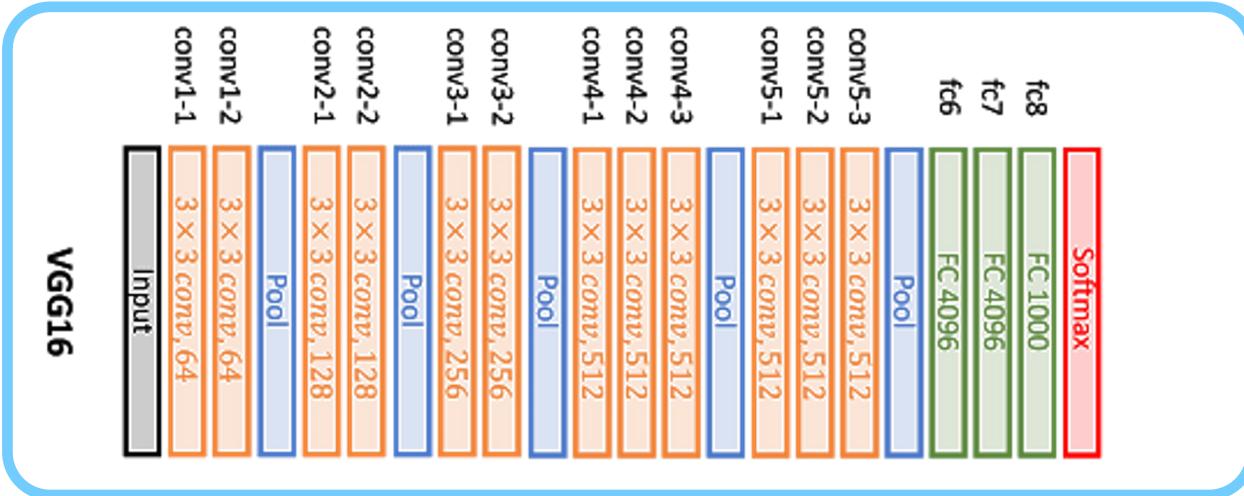
## 02 Activation function — function applied to layer output

- Sigmoid
- $\tanh$
- ReLU
- Any other function to get nonlinear intermediate signal in NN

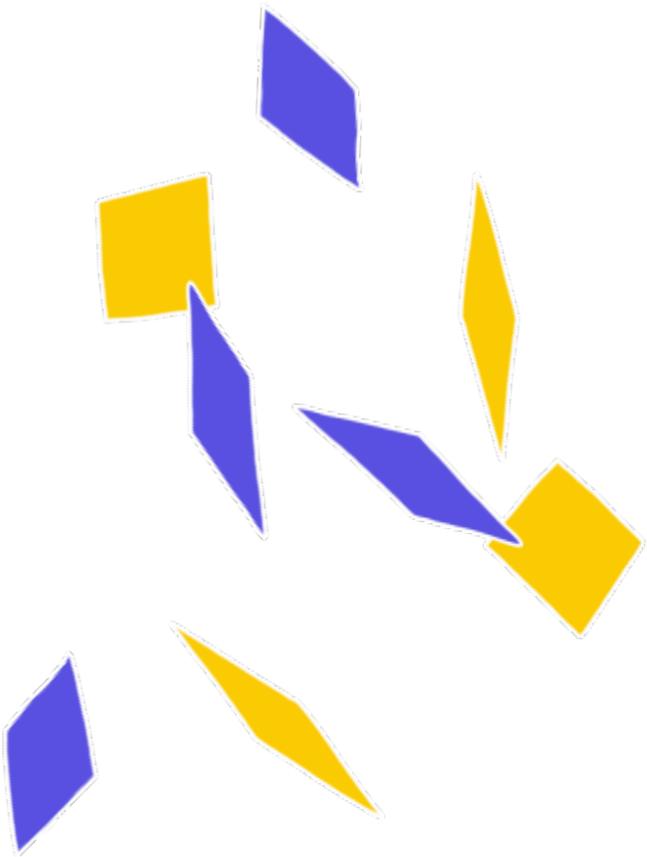
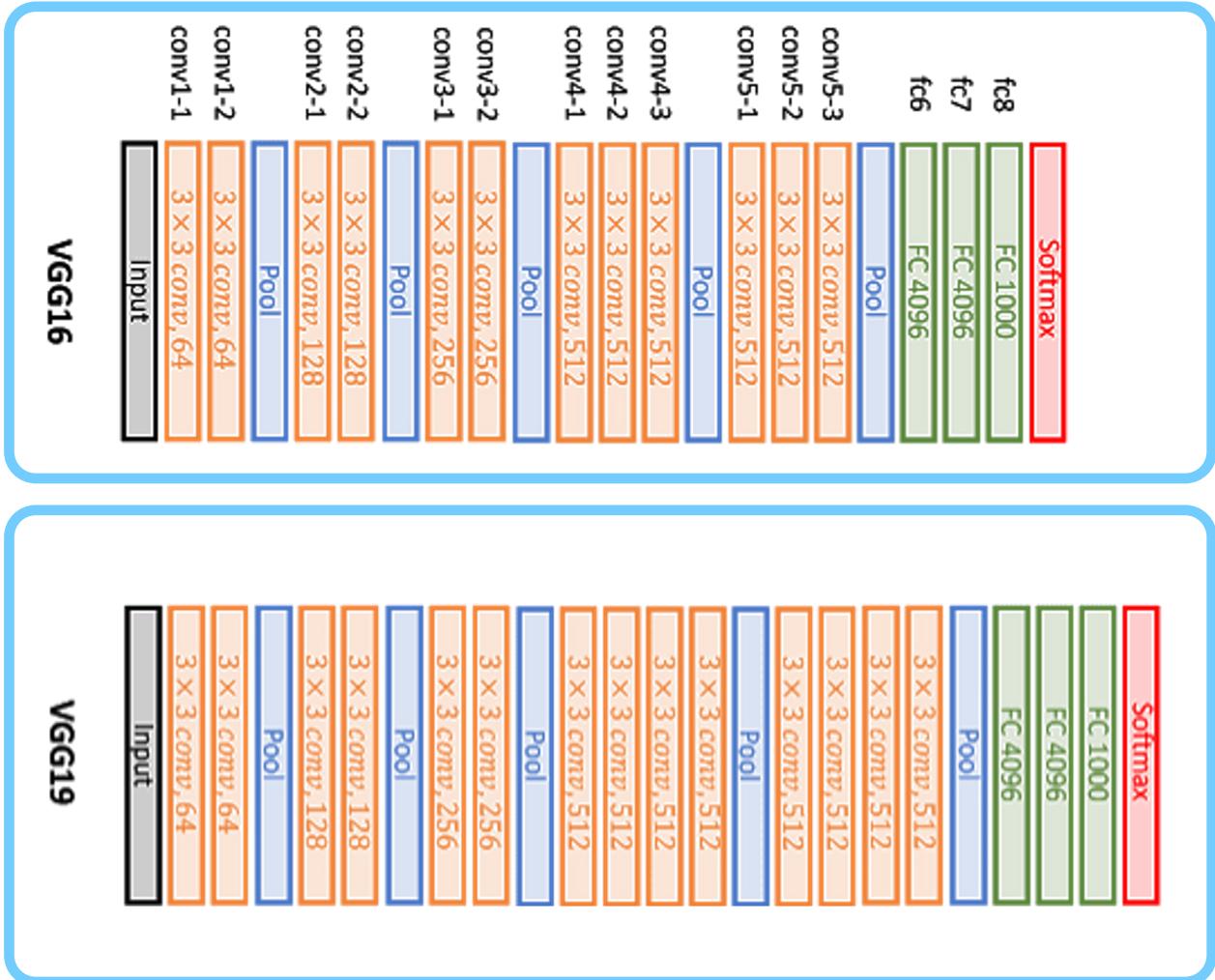
## 03 Backpropagation — a fancy word for “chain rule”



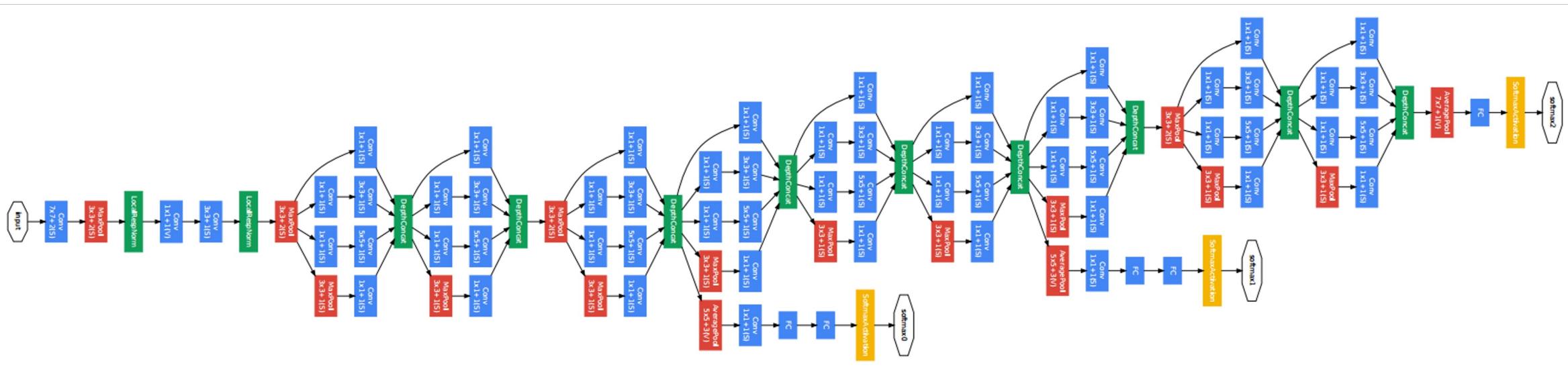
# Actually, networks can be deep



# And deeper...

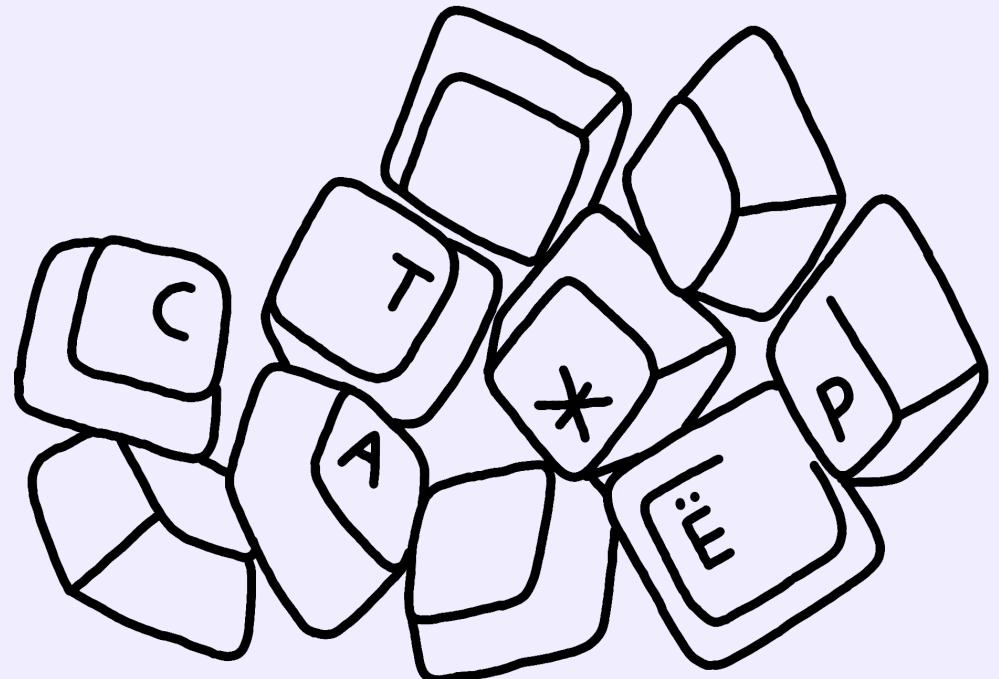


# Much deeper...

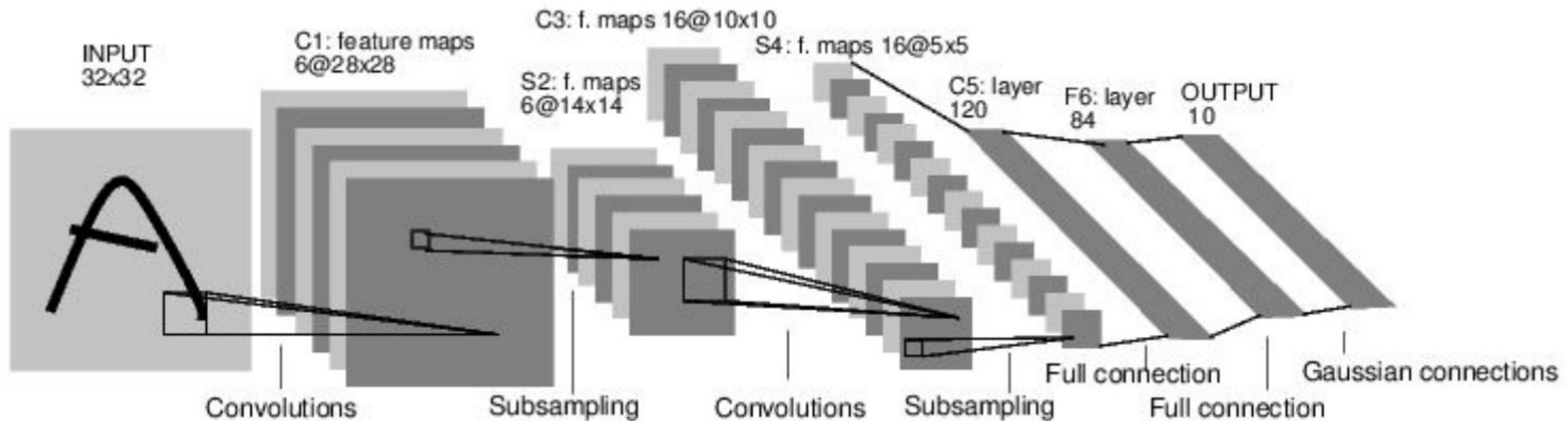


# Convolutional layer

01



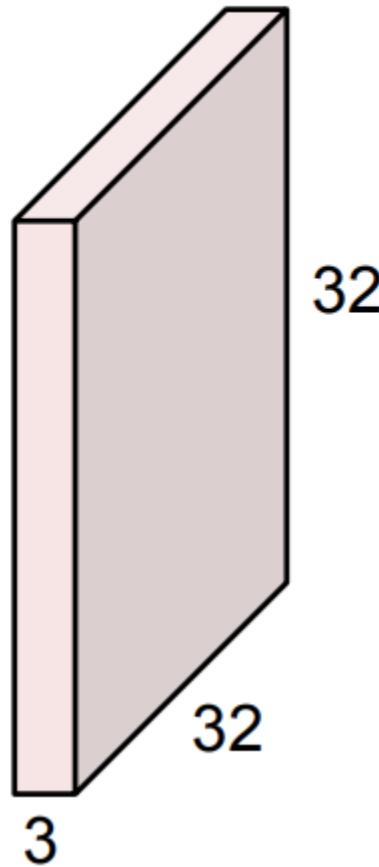
# Convolutions in Neural Networks



[LeNet-5, LeCun 1998]

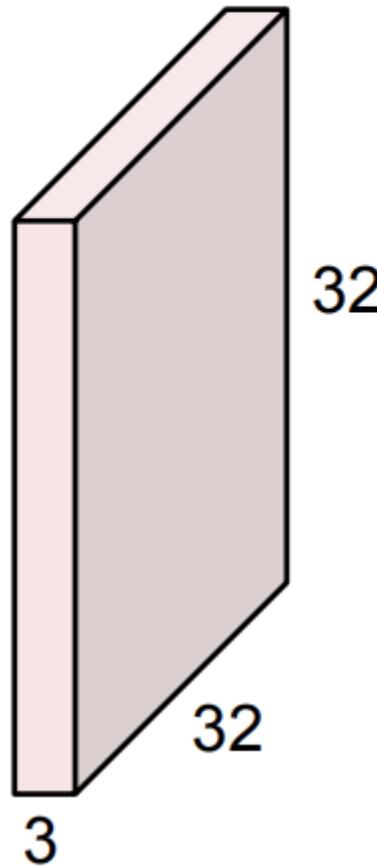
# Convolutional layer

32x32x3 image



# Convolutional layer

32x32x3 image



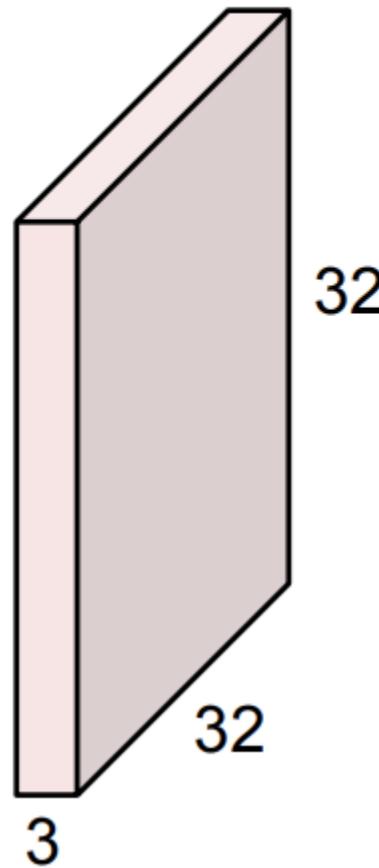
5x5x3 filter



**Convolve** the filter with the image  
i.e. “slide over the image spatially,  
computing dot products”

# Convolutional layer

32x32x3 image



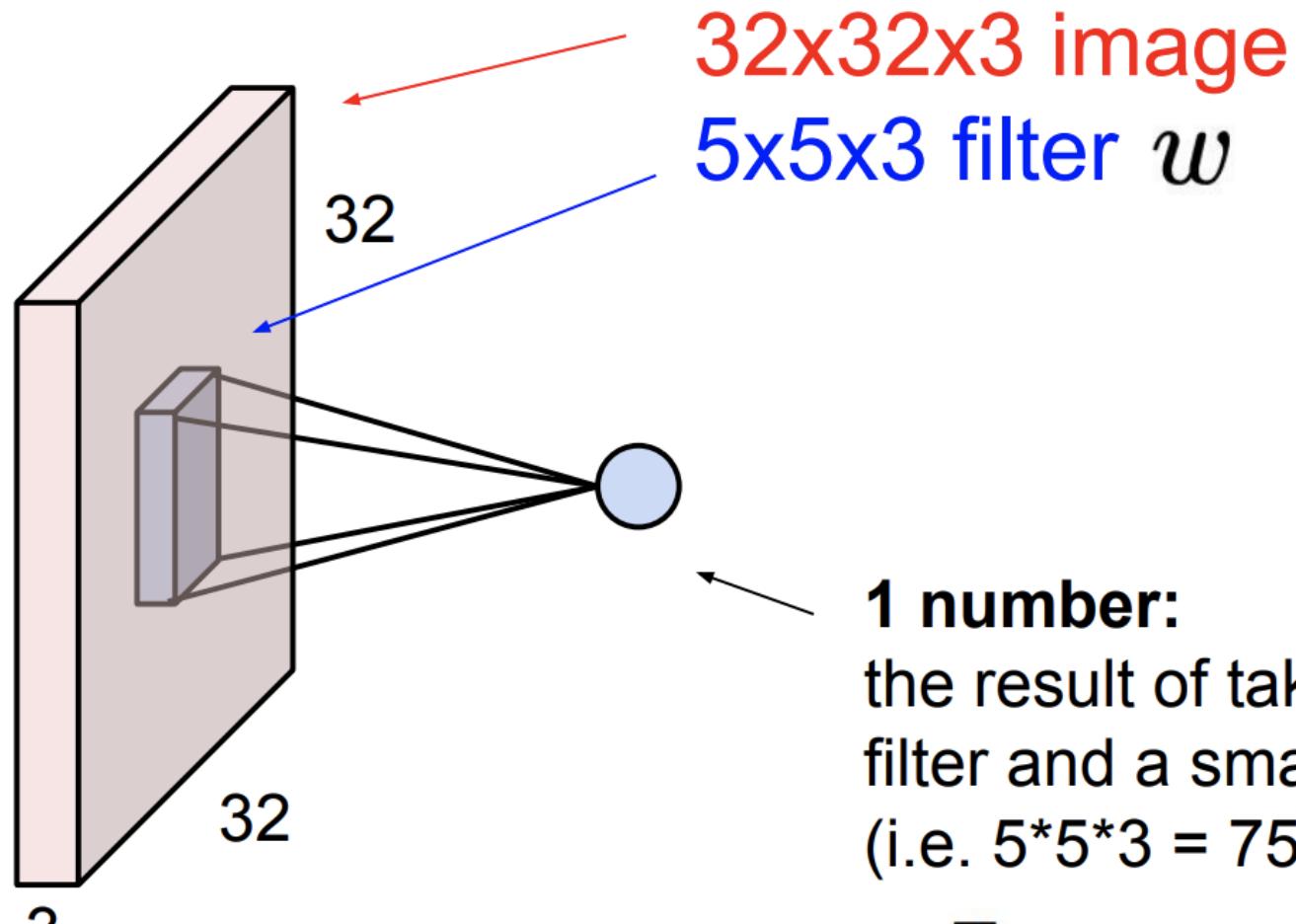
Filters extend the depth of the original image

5x5x3 filter



**Convolve** the filter with the image  
i.e. “slide over the image spatially,  
computing dot products”

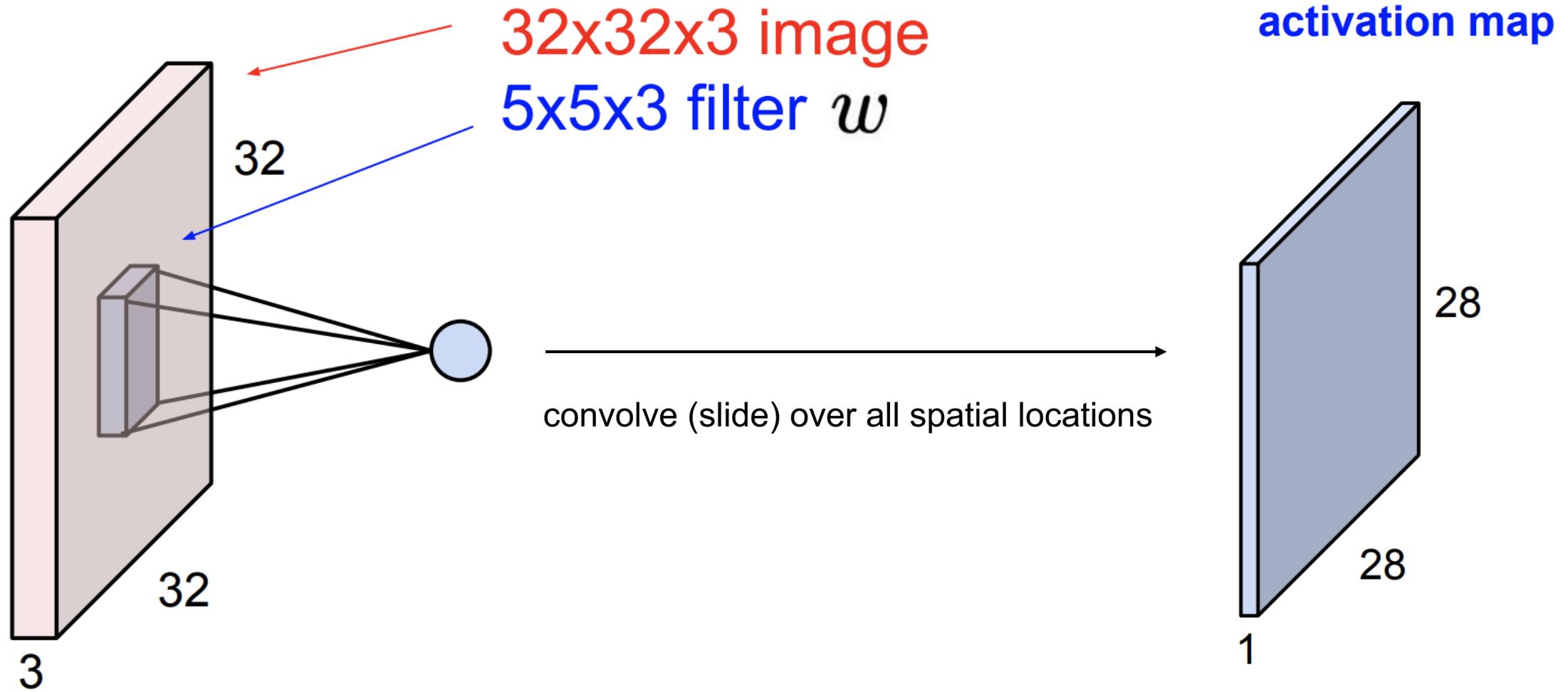
# Convolutional layer



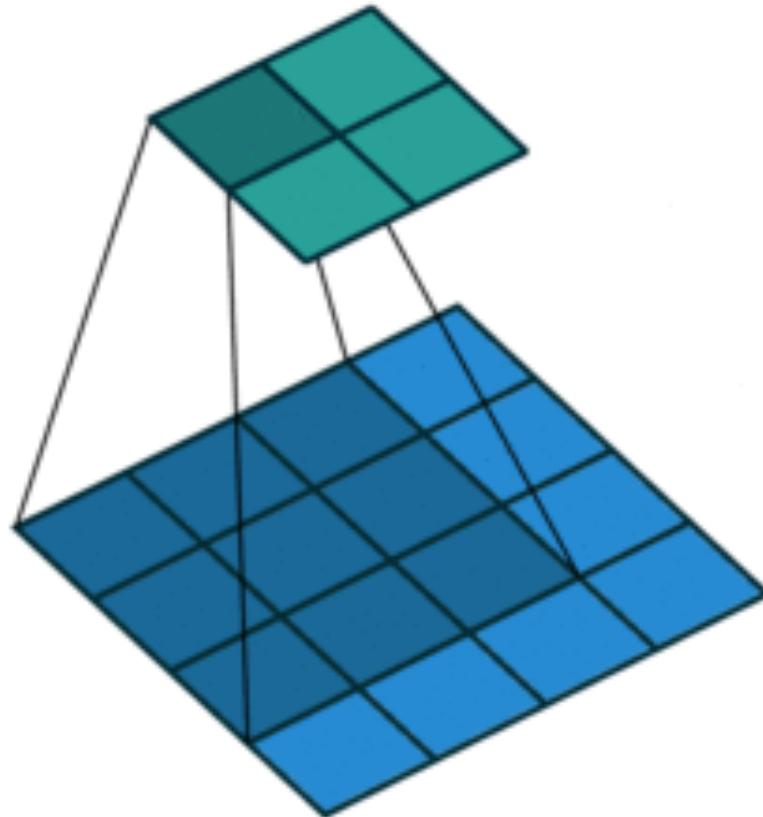
**1 number:**  
the result of taking a dot product between the  
filter and a small  $5 \times 5 \times 3$  chunk of the image  
(i.e.  $5 \times 5 \times 3 = 75$ -dimensional dot product + bias)

$$w^T x + b$$

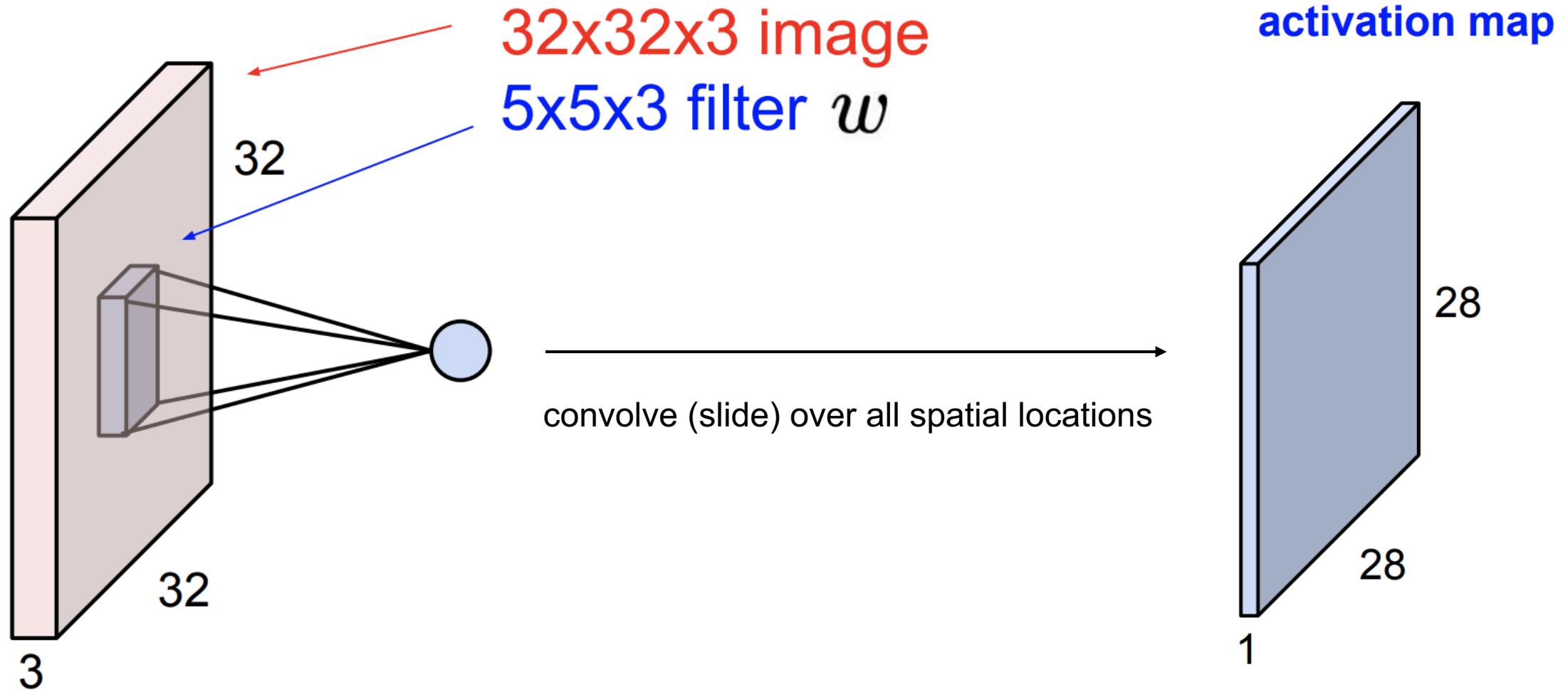
# Convolutional layer



# Convolutional layer

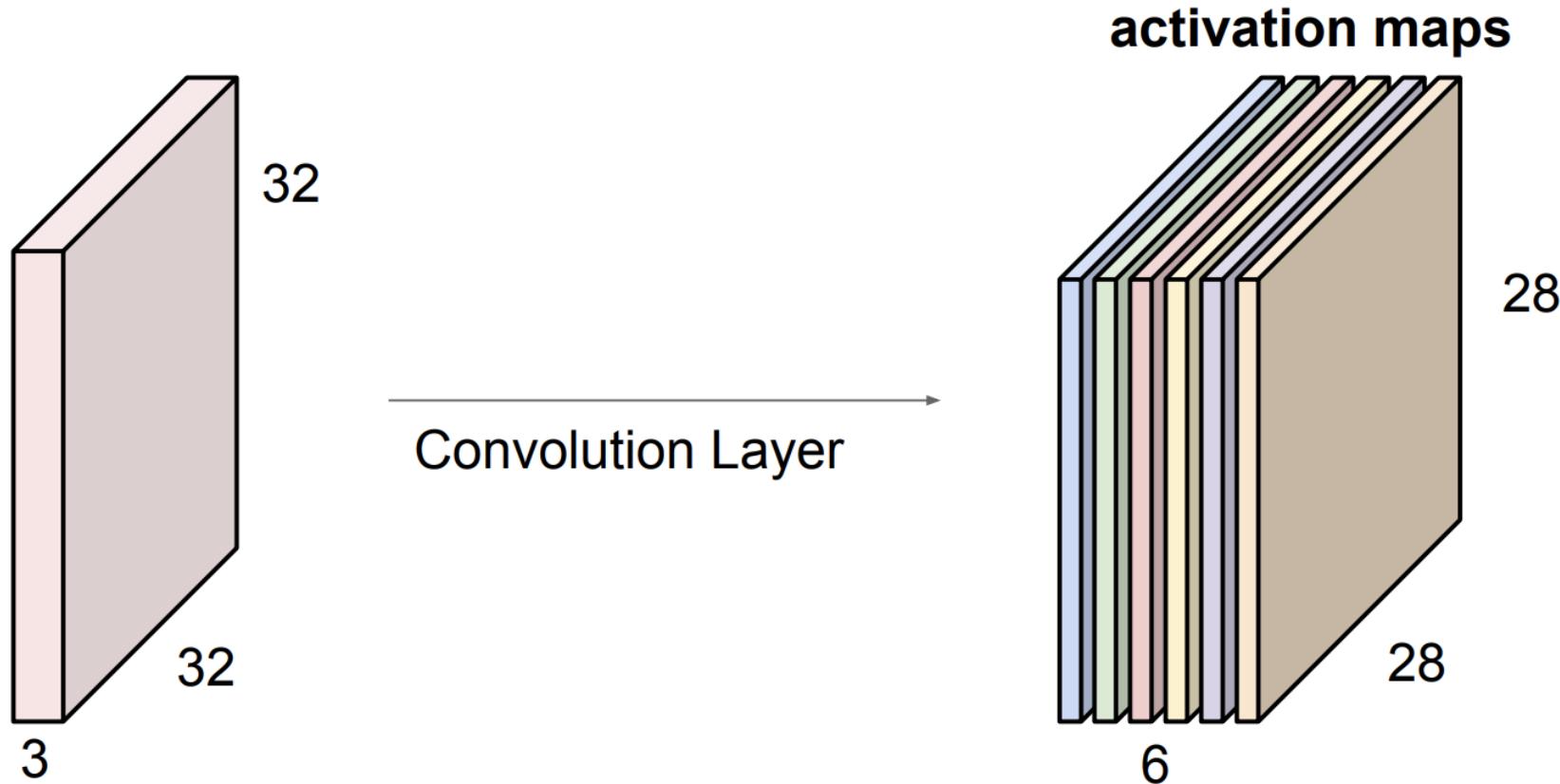


# Convolutional layer



# Convolutional layer

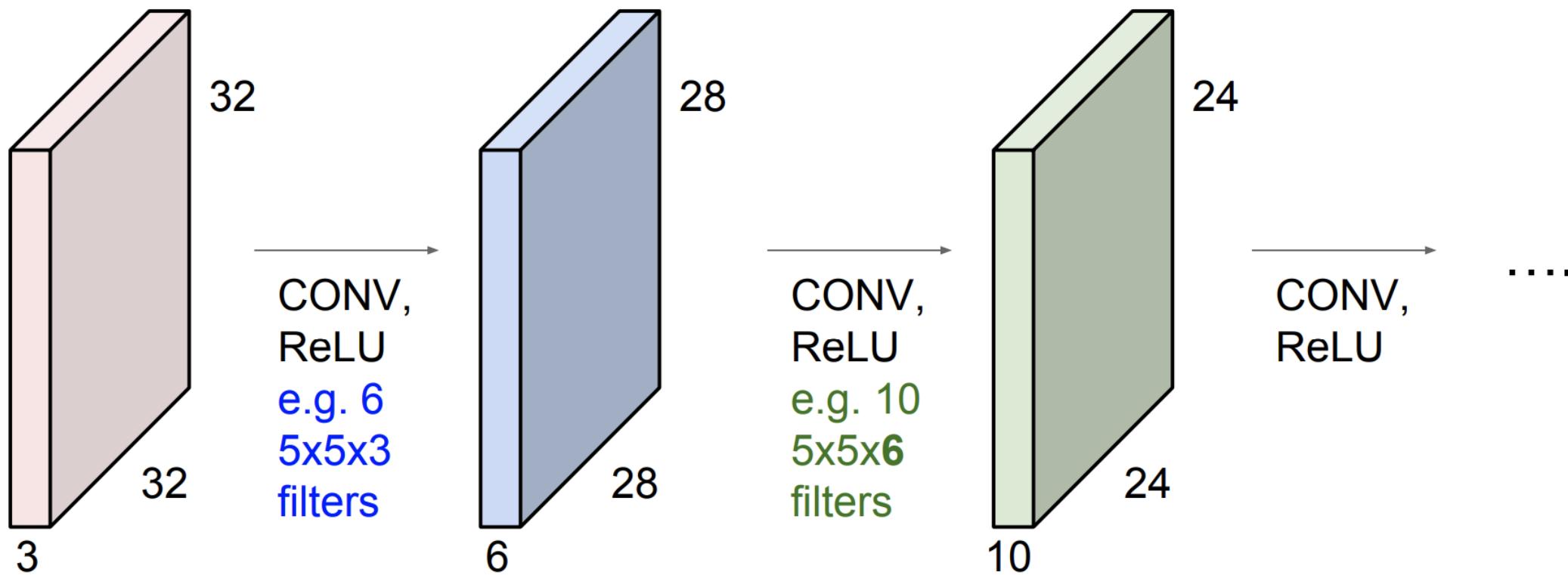
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



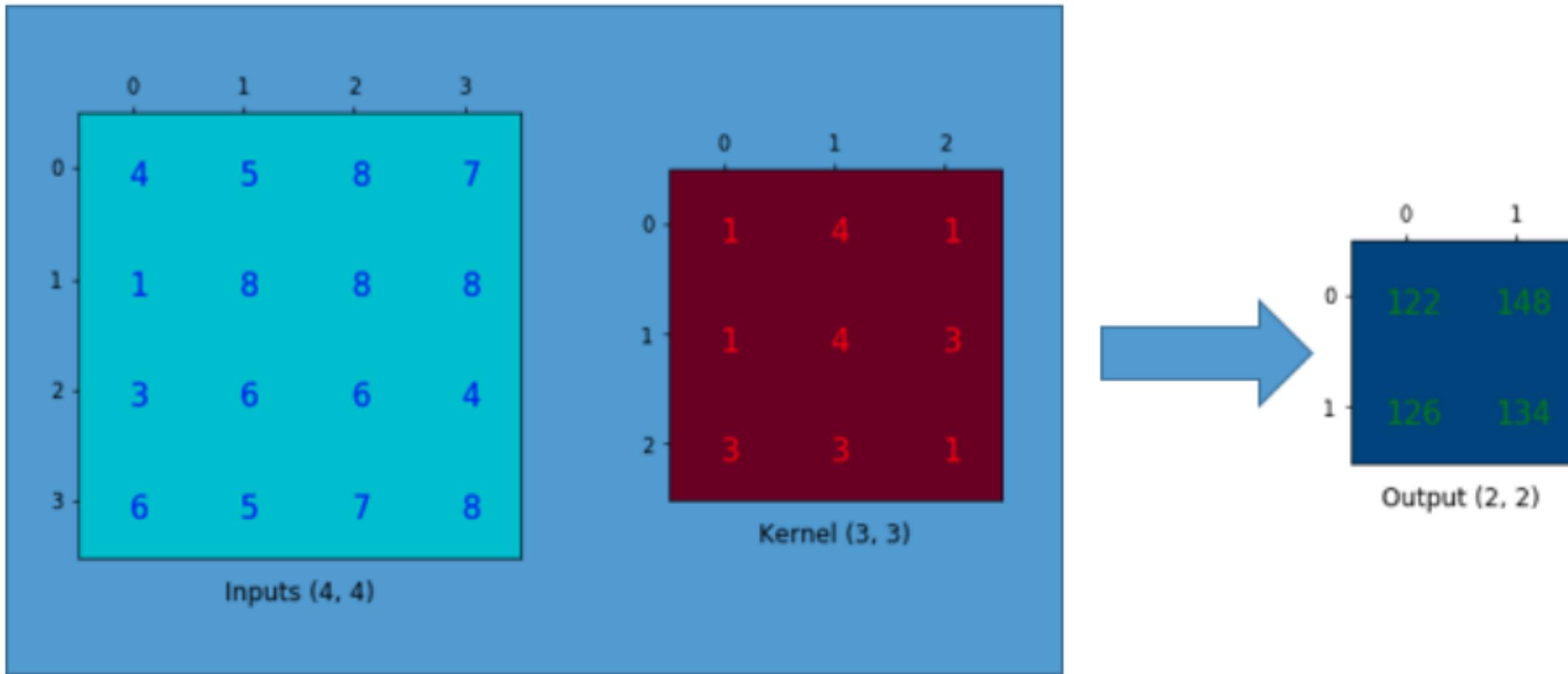
We stack these up to get a “new image” of size 28x28x6!

# Convolutional layer

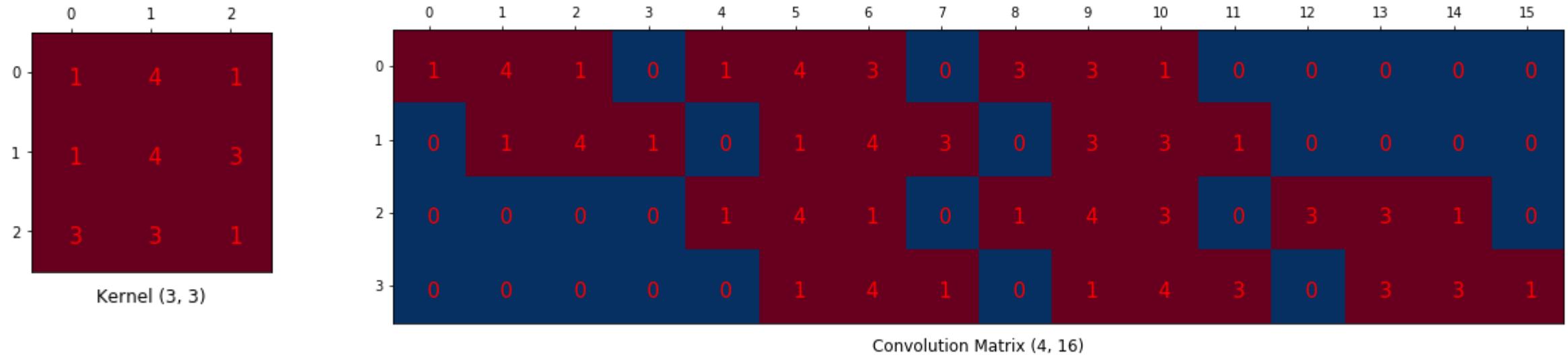
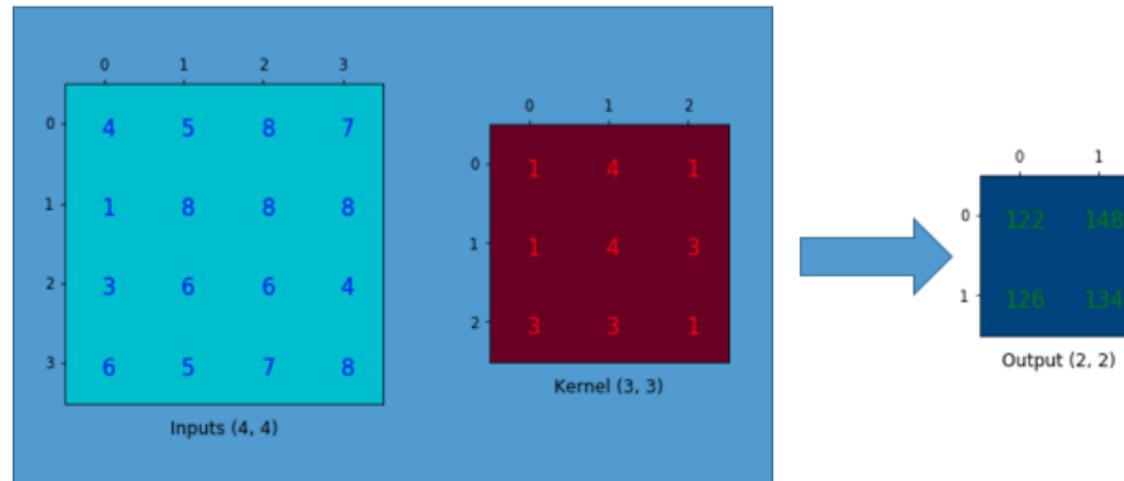
**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



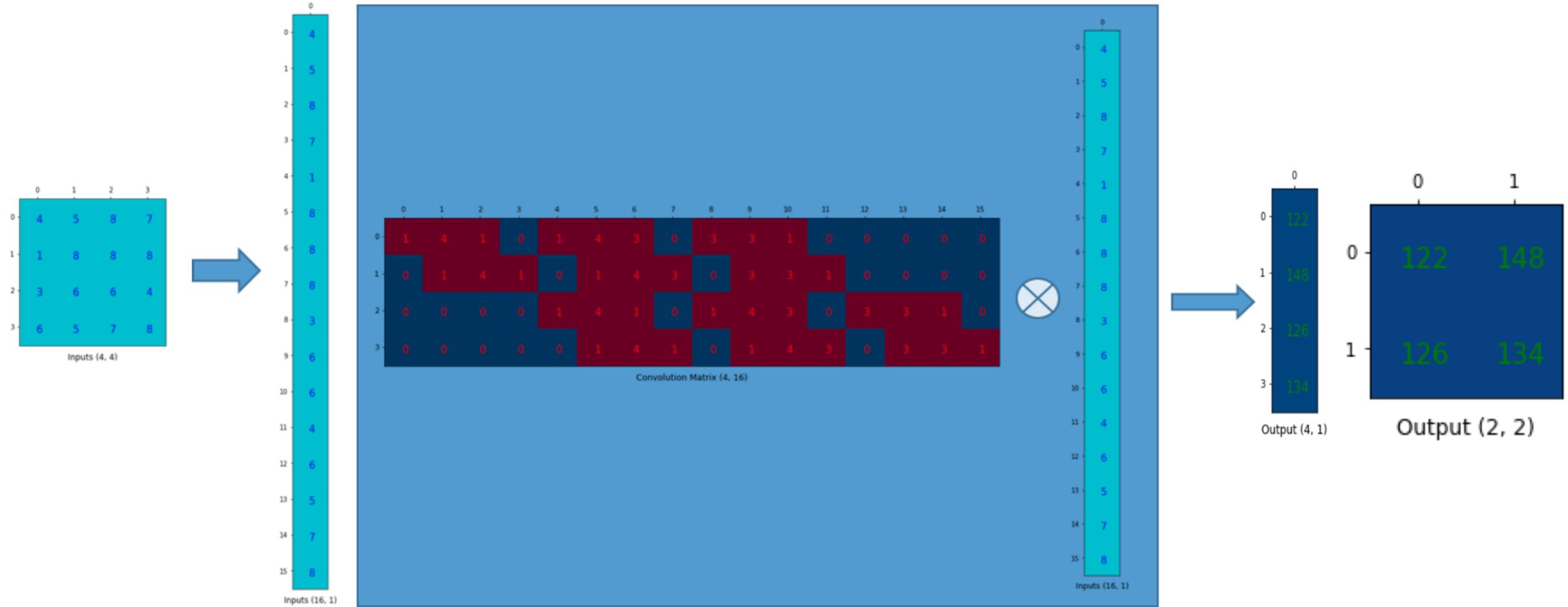
# Regular convolution



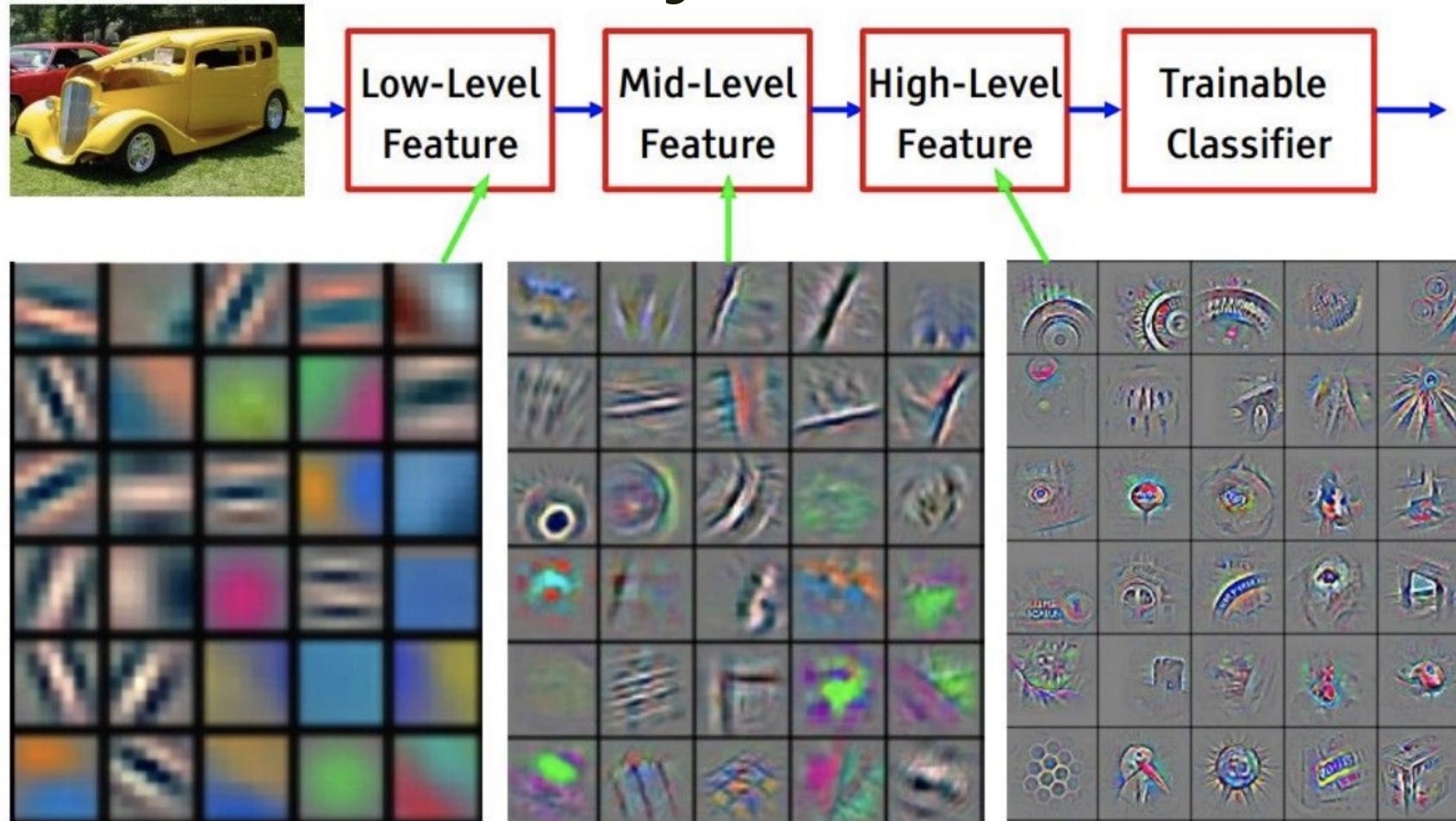
# Regular convolution: matrix form



# Regular convolution: matrix form



# Convolutional layer



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

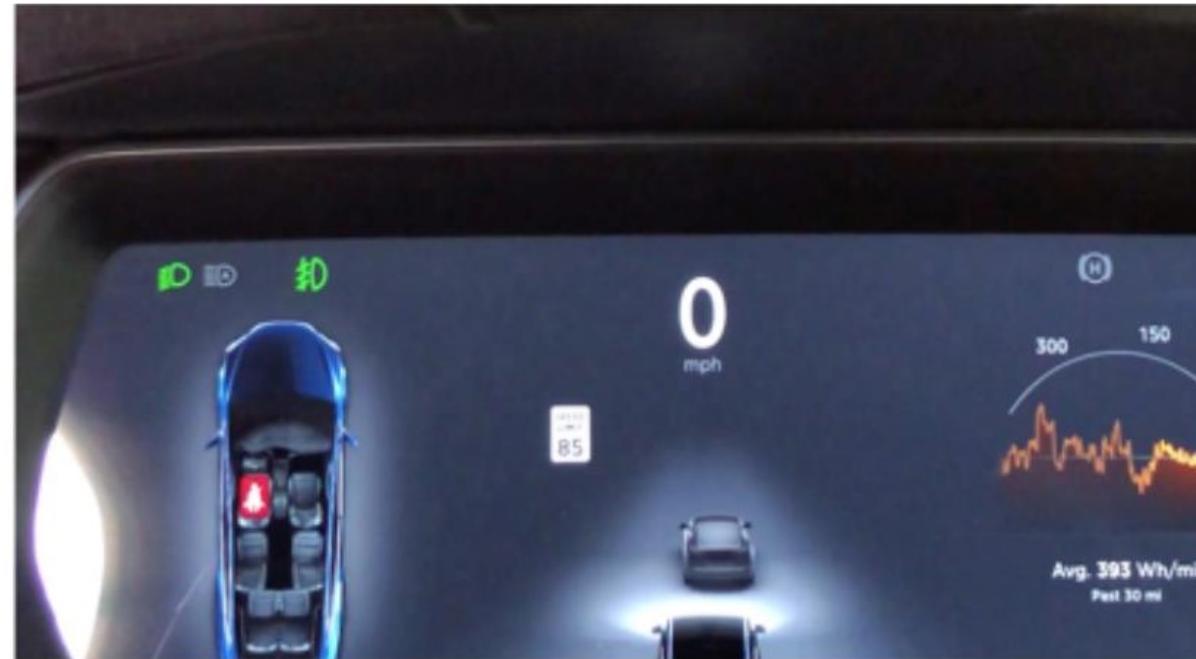
# Deep dream



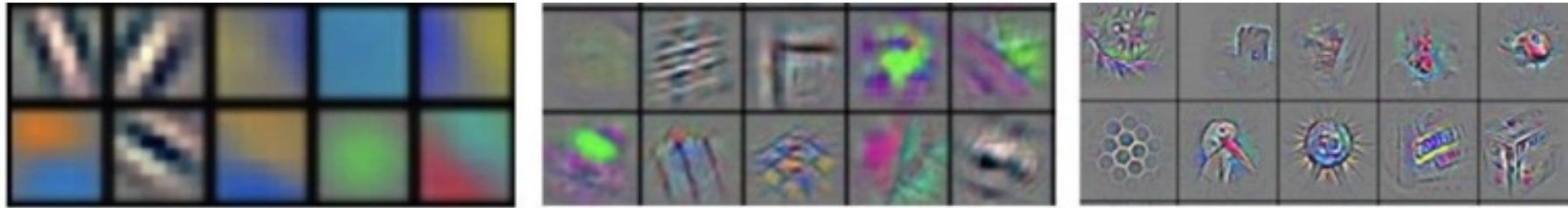
# Further analysis of CNN structure

Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

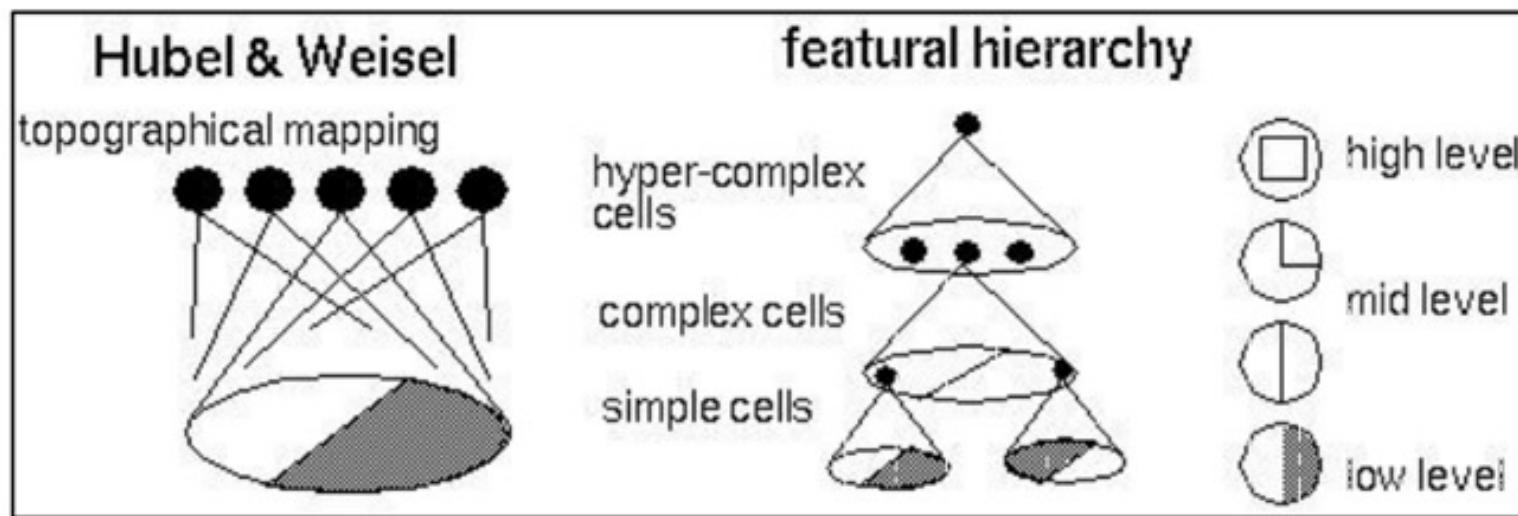
<https://arxiv.org/pdf/1610.02391>



# Convolutional layer and visual cortex



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]



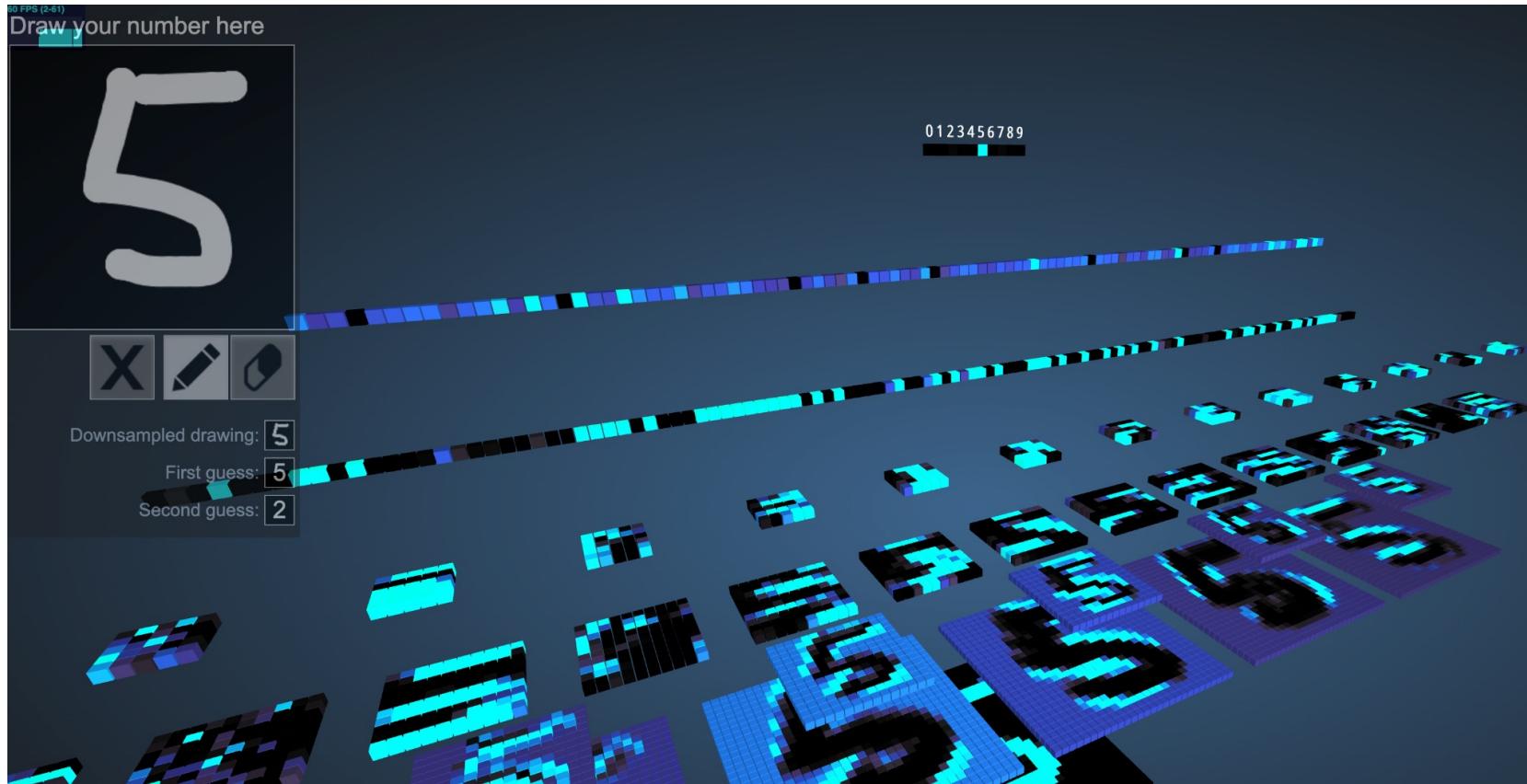
# Convolutional layer and visual cortex

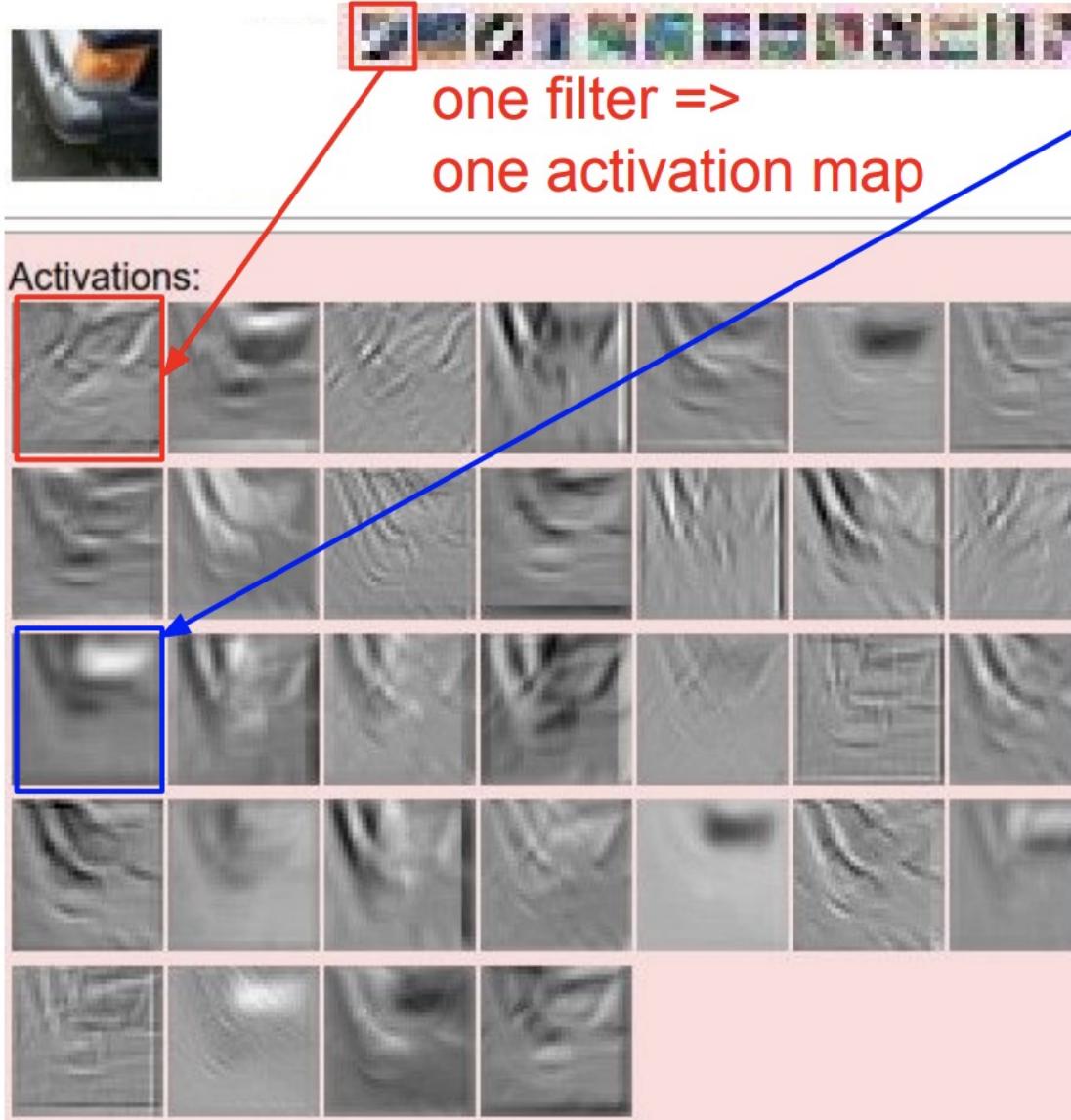


# MNIST demo

3D demo: [https://adamharley.com/nn\\_vis/cnn/3d.html](https://adamharley.com/nn_vis/cnn/3d.html)

2D demo (lightweight): [https://adamharley.com/nn\\_vis/cnn/2d.html](https://adamharley.com/nn_vis/cnn/2d.html)

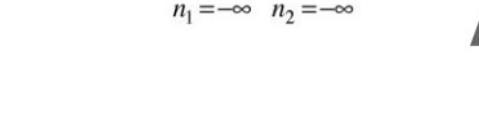




example 5x5 filters  
(32 total)

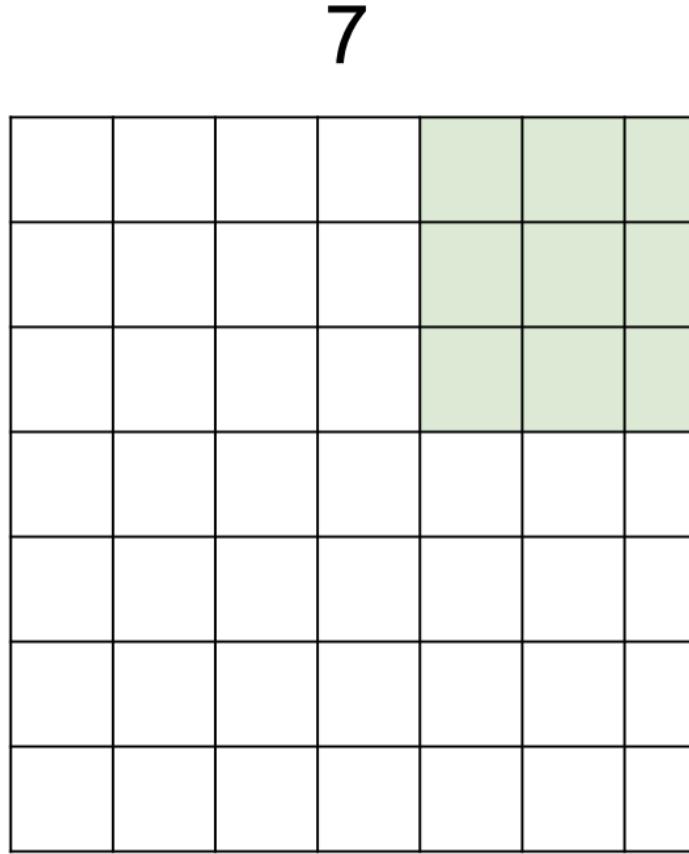
We call the layer convolutional  
because it is related to convolution  
of two signals:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x - n_1, y - n_2]$$



elementwise multiplication and sum of  
a filter and the signal (image)

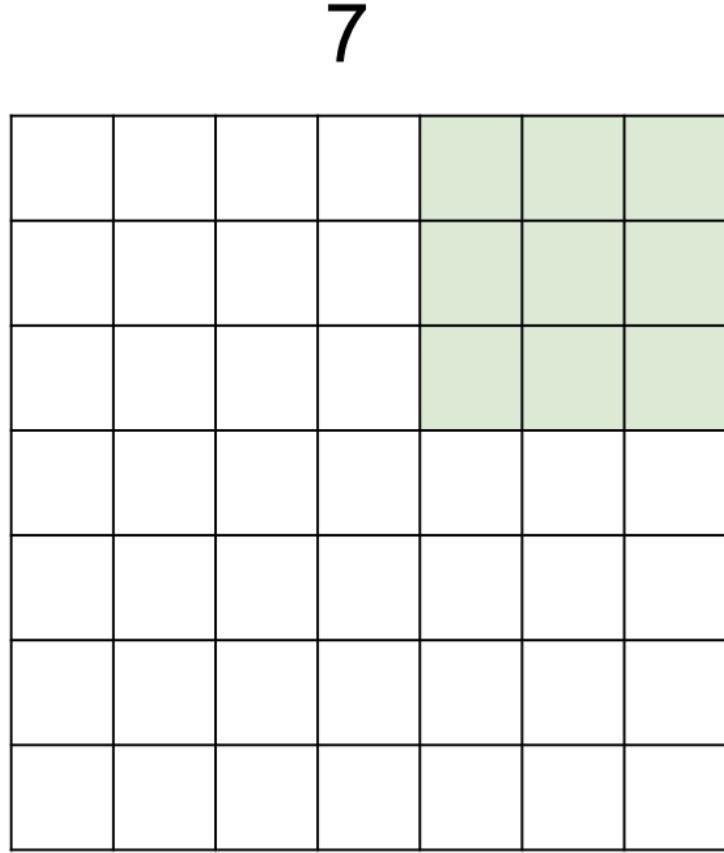
## A closer look at spatial dimensions:



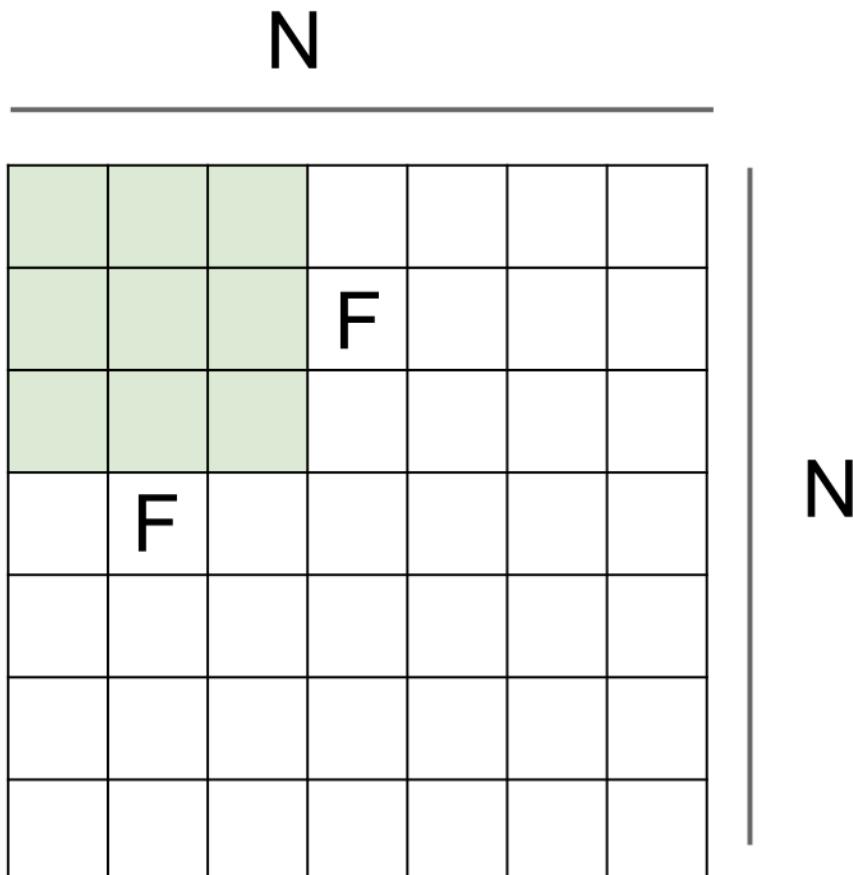
7x7 input (spatially)  
assume 3x3 filter

**=> 5x5 output**

## A closer look at spatial dimensions:



7x7 input (spatially)  
assume 3x3 filter  
applied **with stride 2**  
**=> 3x3 output!**



Output size:  
 **$(N - F) / \text{stride} + 1$**

e.g.  $N = 7$ ,  $F = 3$ :  
stride 1 =>  $(7 - 3)/1 + 1 = 5$   
stride 2 =>  $(7 - 3)/2 + 1 = 3$   
stride 3 =>  $(7 - 3)/3 + 1 = 2.33 : \backslash$

# In practice: Common to zero pad the border

0	0	0	0	0	0		
0							
0							
0							
0							

e.g. input 7x7

**3x3 filter, applied with stride 1**

**pad with 1 pixel border => what is the output?**

**7x7 output!**

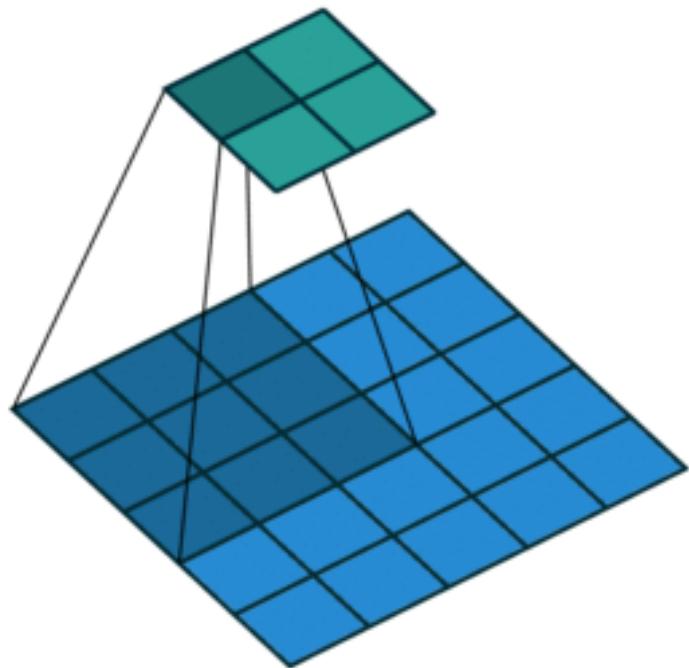
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with  $(F-1)/2$ . (will preserve size spatially)

e.g.  $F = 3 \Rightarrow$  zero pad with 1

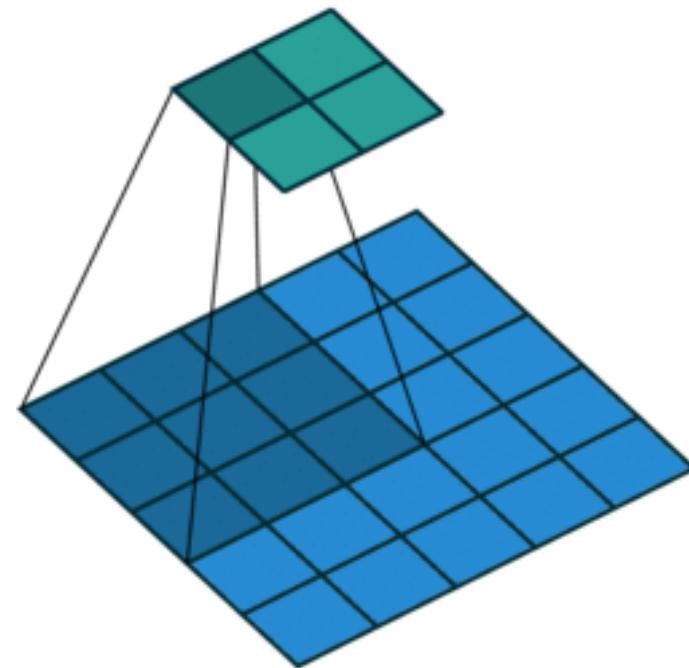
$F = 5 \Rightarrow$  zero pad with 2

$F = 7 \Rightarrow$  zero pad with 3

# Strides, padding

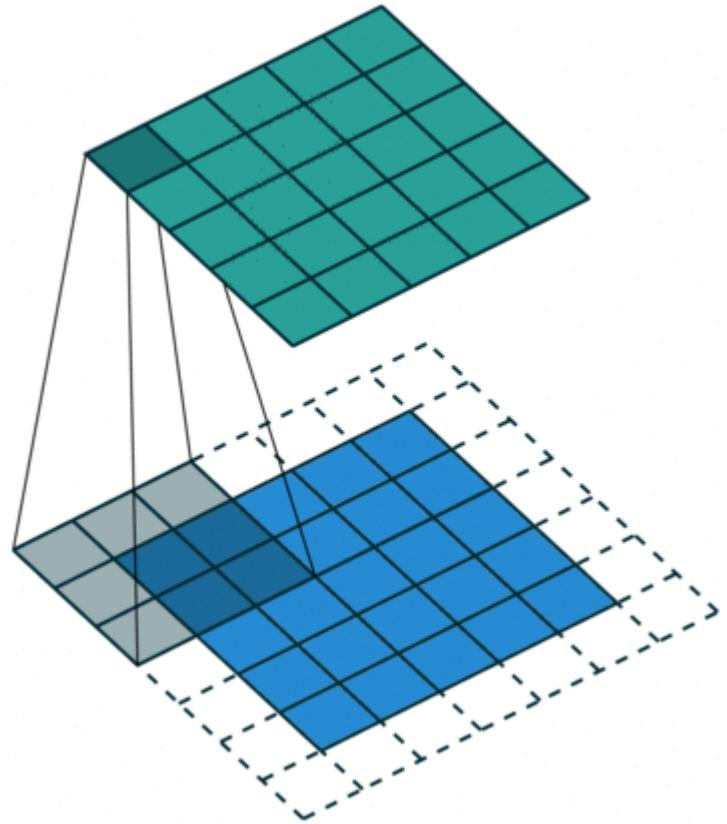


No padding, no strides

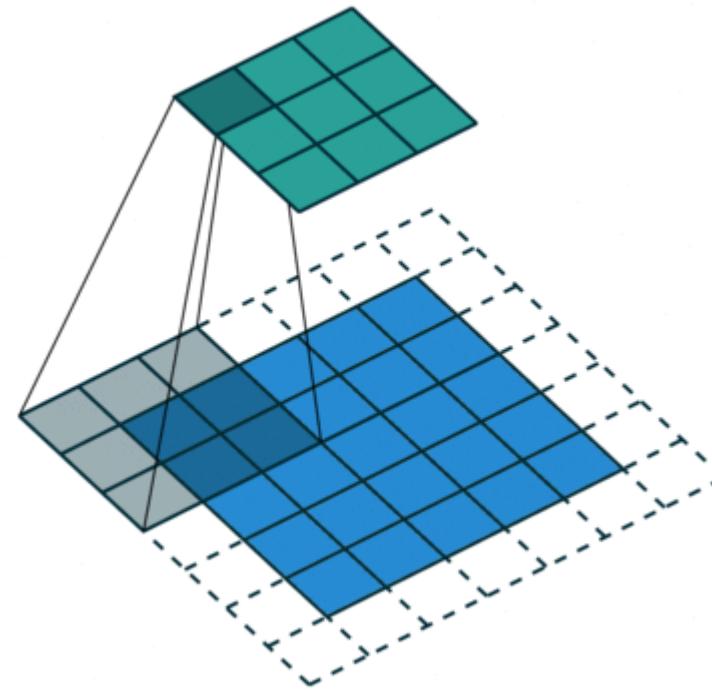


No padding, with strides

# Strides, padding



With padding, no strides



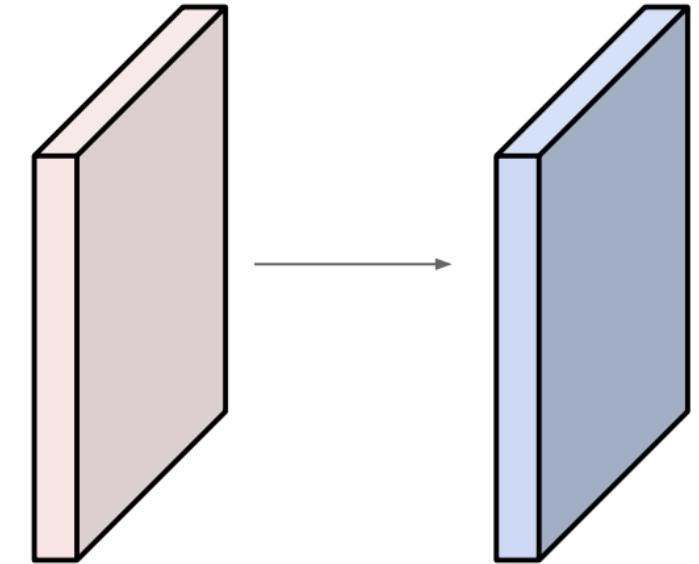
With padding, with strides

Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

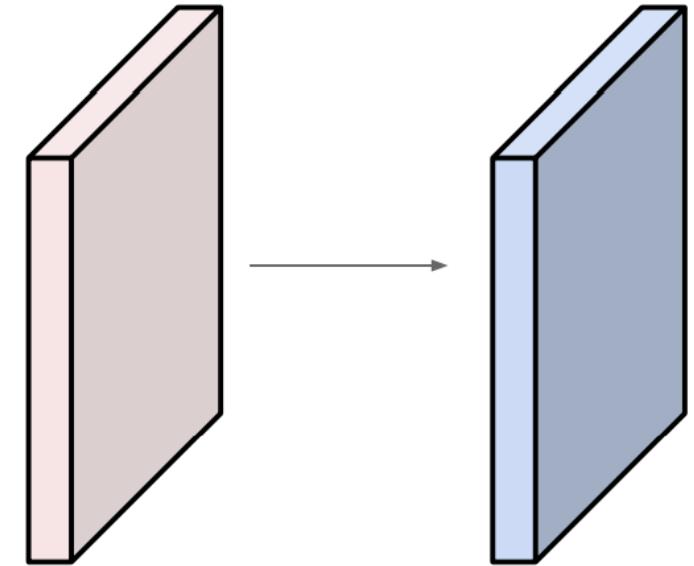
Output volume size: ?



Examples time:

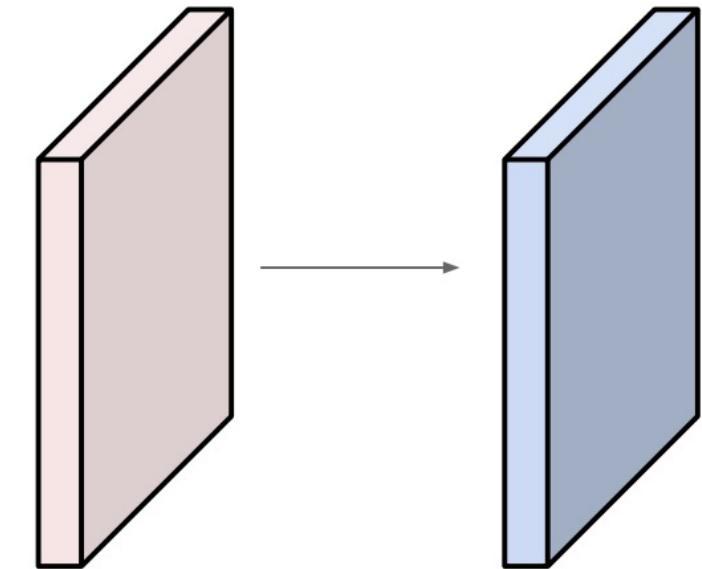
Input volume: **32x32x3**  
10 **5x5** filters with stride **1**, pad **2**

Output volume size:  
 $(32+2*2-5)/1+1 = 32$  spatially, so  
**32x32x10**



Examples time:

Input volume: **32x32x3**  
10 5x5 filters with stride 1, pad 2

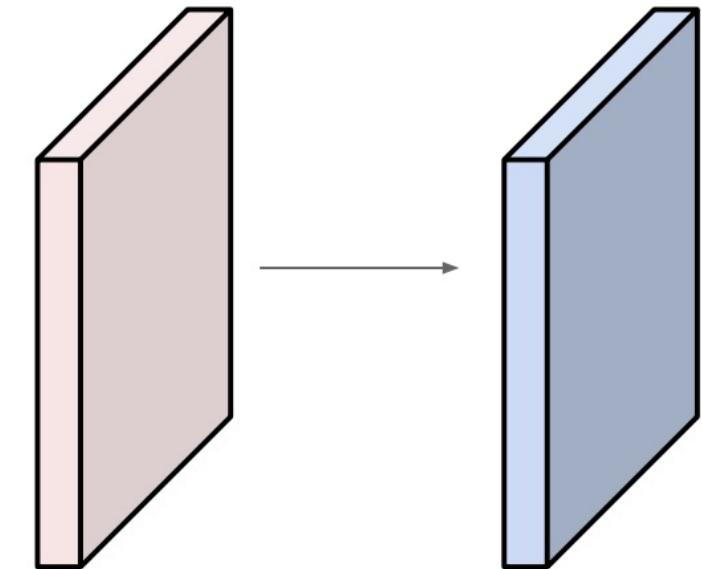


Number of parameters in this layer?

Examples time:

Input volume: **32x32x3**

10 **5x5** filters with stride 1, pad 2



Number of parameters in this layer?

each filter has  $5*5*3 + 1 = 76$  params (+1 for bias)

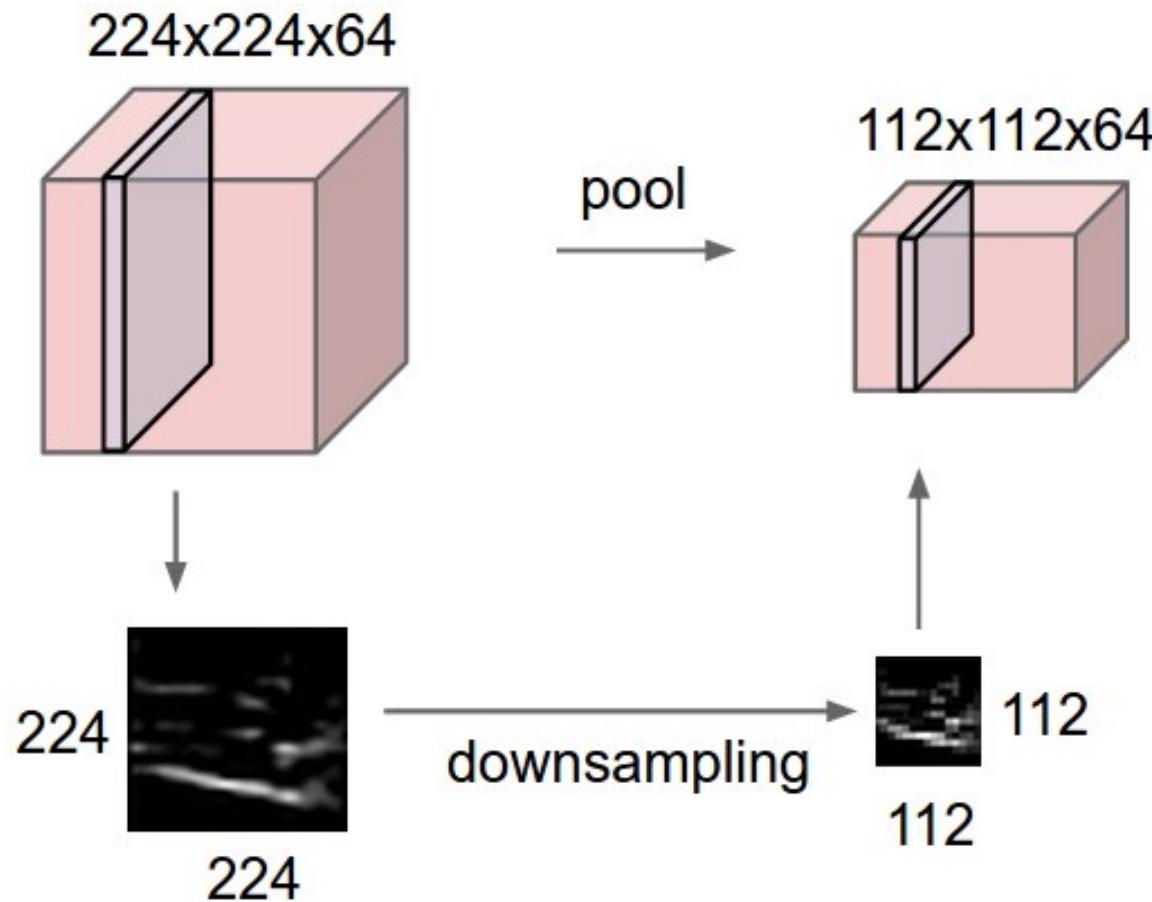
$$\Rightarrow 76 * 10 = 760$$

# Pooling layers

02

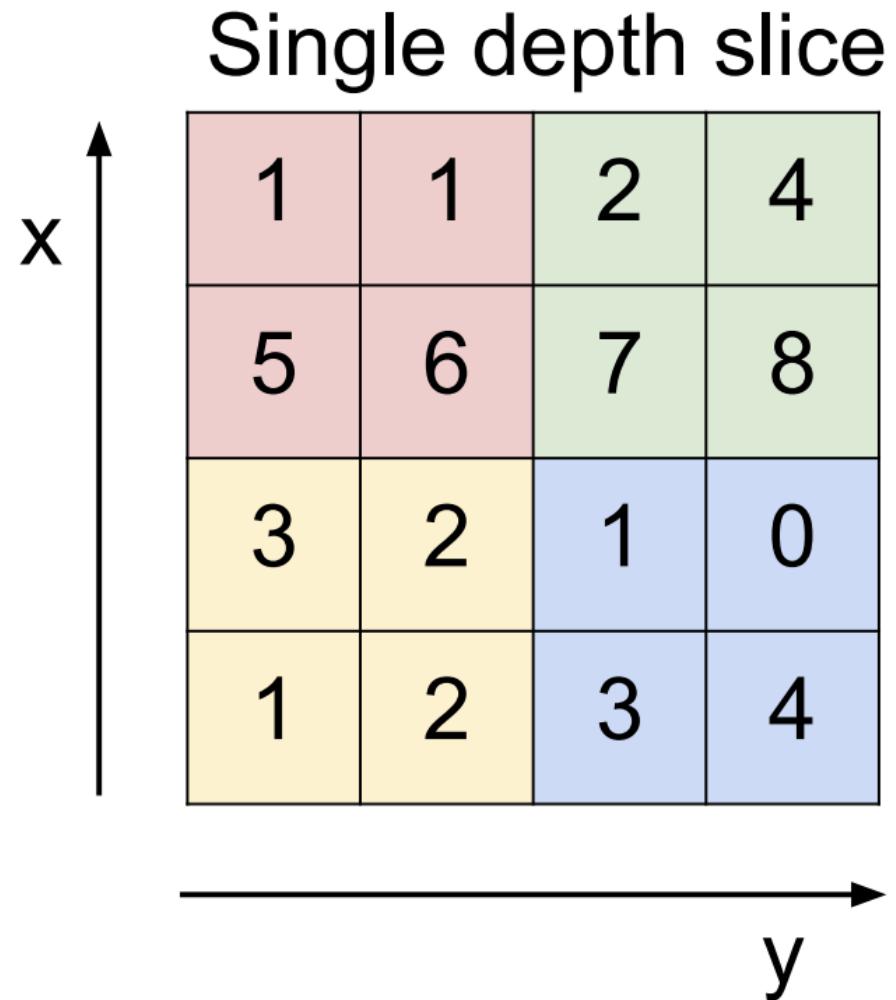


# Pooling layer



- Makes the representations smaller and more manageable
- Operates over each channel independently
- No trainable params, so doesn't overfit

# Max pooling



max pool with 2x2 filters  
and stride 2



6	8
3	4

# Average pooling

2	2	7	3
9	4	6	1
8	5	2	4
3	1	2	6

Average Pool  
→

Filter -  $(2 \times 2)$   
Stride -  $(2, 2)$

4.25	4.25
4.25	3.5

# Adaptive pooling

- Pools adaptive region size into fixed output size
- Fixed size is required for following linear layers
- Allows to put arbitrary image size into convolution network
- Both Max- and Average-

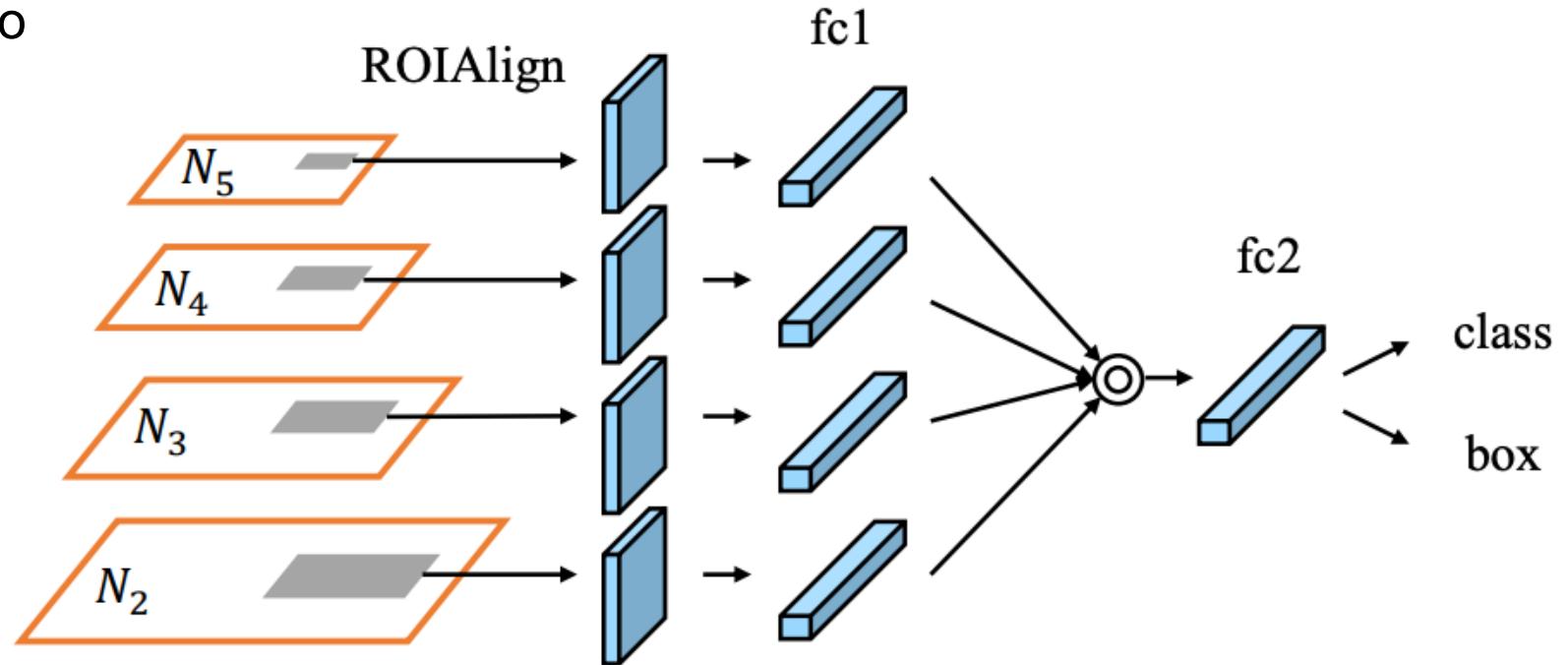
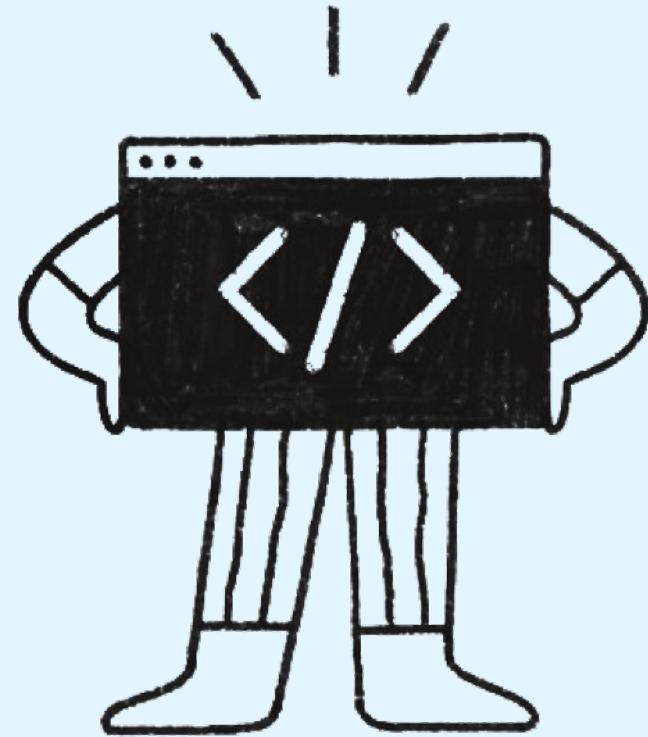


Figure 6. Illustration of adaptive feature pooling on box branch.

[torch.nn.AdaptiveAvgPool2d](https://pytorch.org/docs/stable/torch.nn.html#torch.nn.AdaptiveAvgPool2d)

# Classification architectures overview

03



# Image classification then

2007: Asirra

CAPTCHA that asks users to identify cats out of a set of 12 photographs of both cats and dogs.

Solved by **humans** in **99.6%** of cases.

Barring a major advance in machine vision, we expect **computers** will have no better than a **1/54,000 chance** of solving it

Please click on all the images that show cats:



[Elson et al](#)

# Image classification now

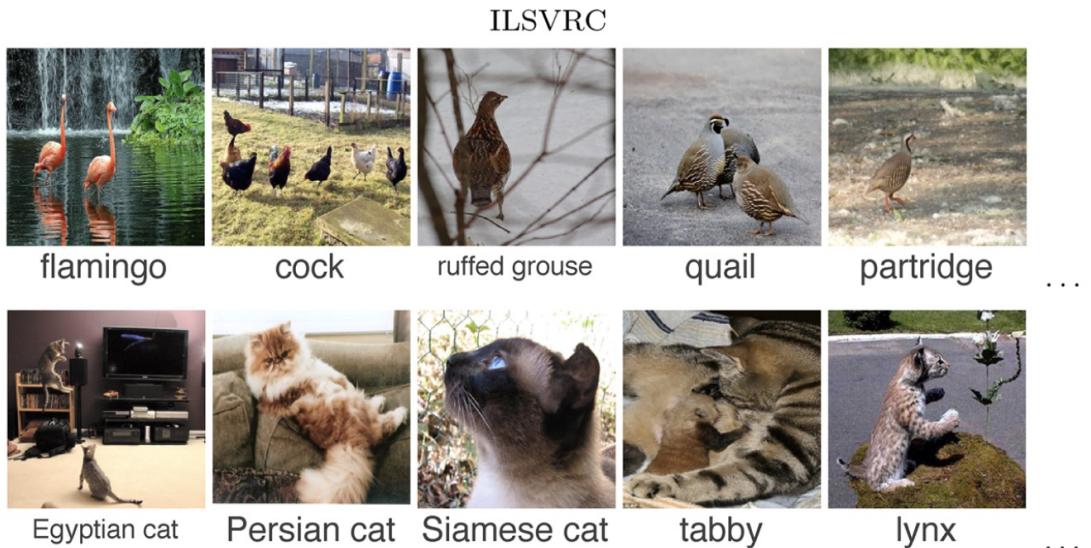
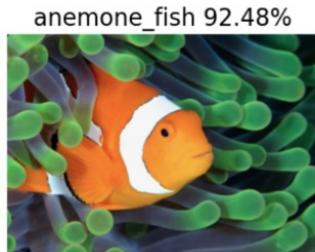
2014: Asirra on kaggle

[Competition page](#)

#	△	Team	Members	Score	Entries	Last	Solution
1	—	Pierre Sermanet		0.98914	5	10y	
2	▲ 4	orchid		0.98308	17	10y	Last Submission: 02/02/2014, 1:43:19 GMT+4
3	—	Owen		0.98171	15	10y	
4	—	Paul Covington		0.98171	3	10y	
5	▼ 3	Maxim Milakov		0.98137	24	10y	
6	▼ 1	we've been in KAIST	 	0.98102	8	10y	
7	▲ 1	Doug Koch		0.98057	6	10y	
8	▲ 2	fastml.com/cats-and-dogs		0.98000	6	10y	

# IMAGENET competition

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- since 2010
- 14.2m images
- 1000 classes
- <http://image-net.org/explore>



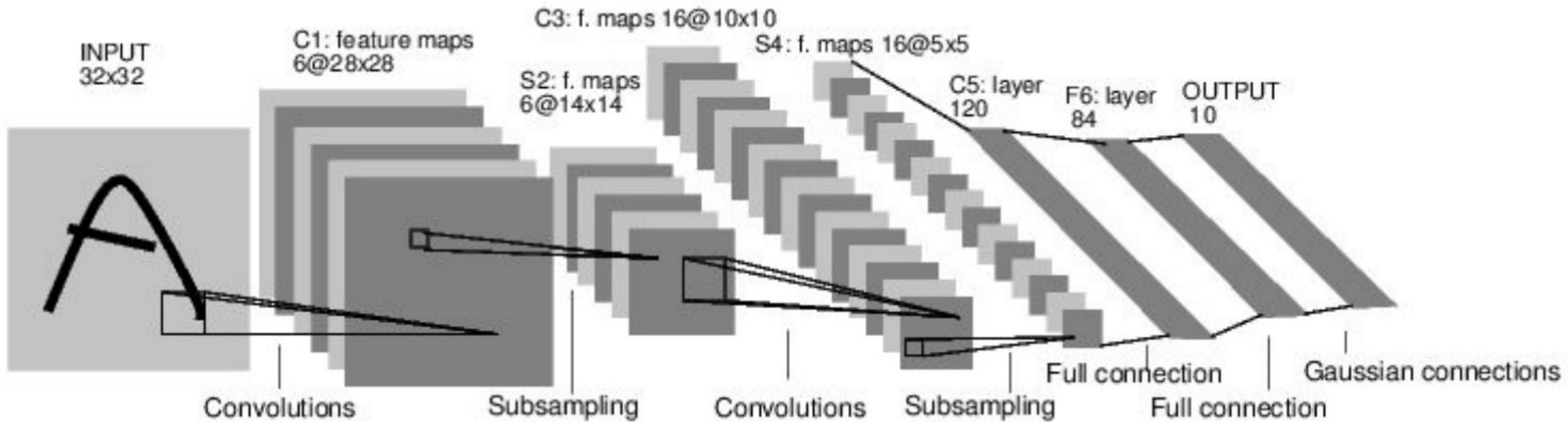
# IMAGENET dataset

- Most of classes are dogs
- Most correlated classes are chihuahua and cupcake

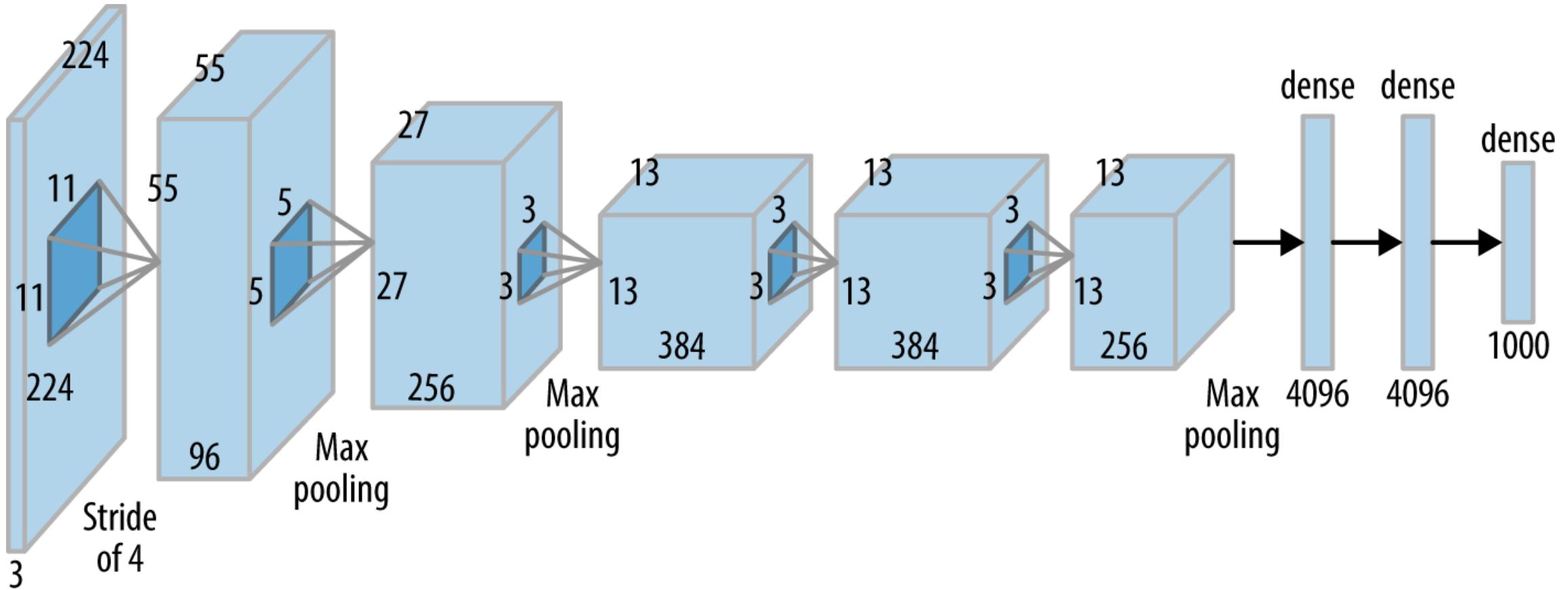
Чтобы подтвердить, что вы не робот, отметьте  
КЕКСЫ



# LeNet-5



# AlexNet

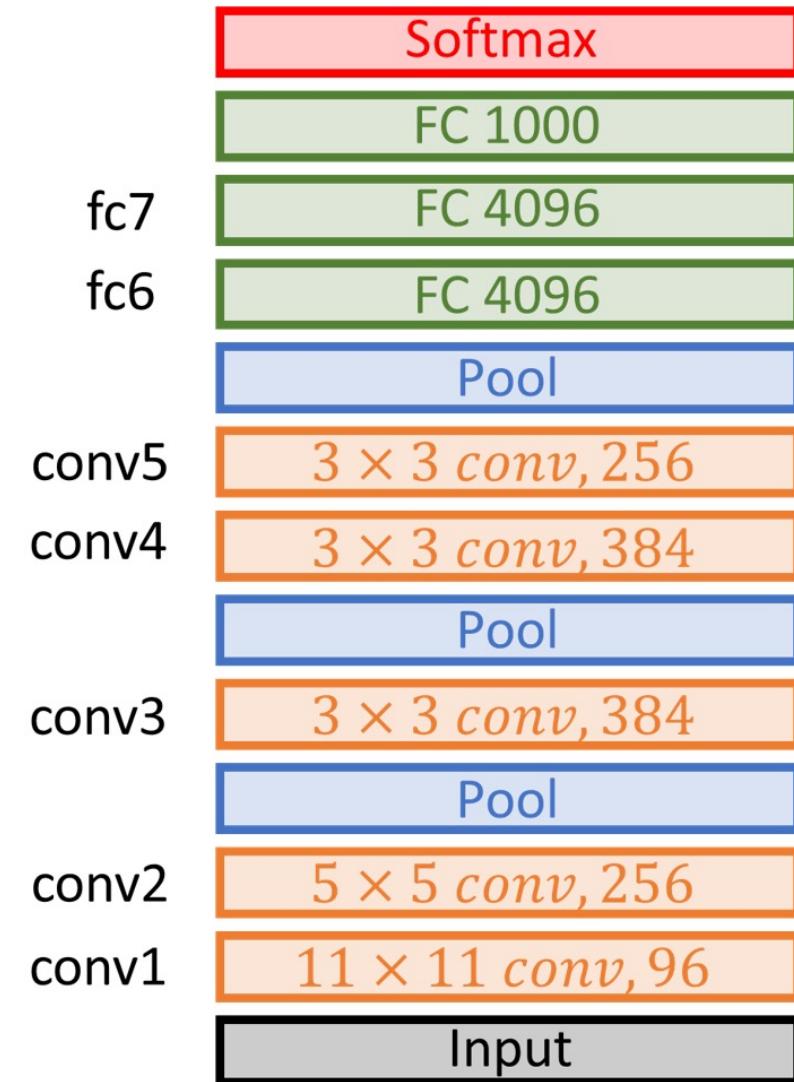


# AlexNet

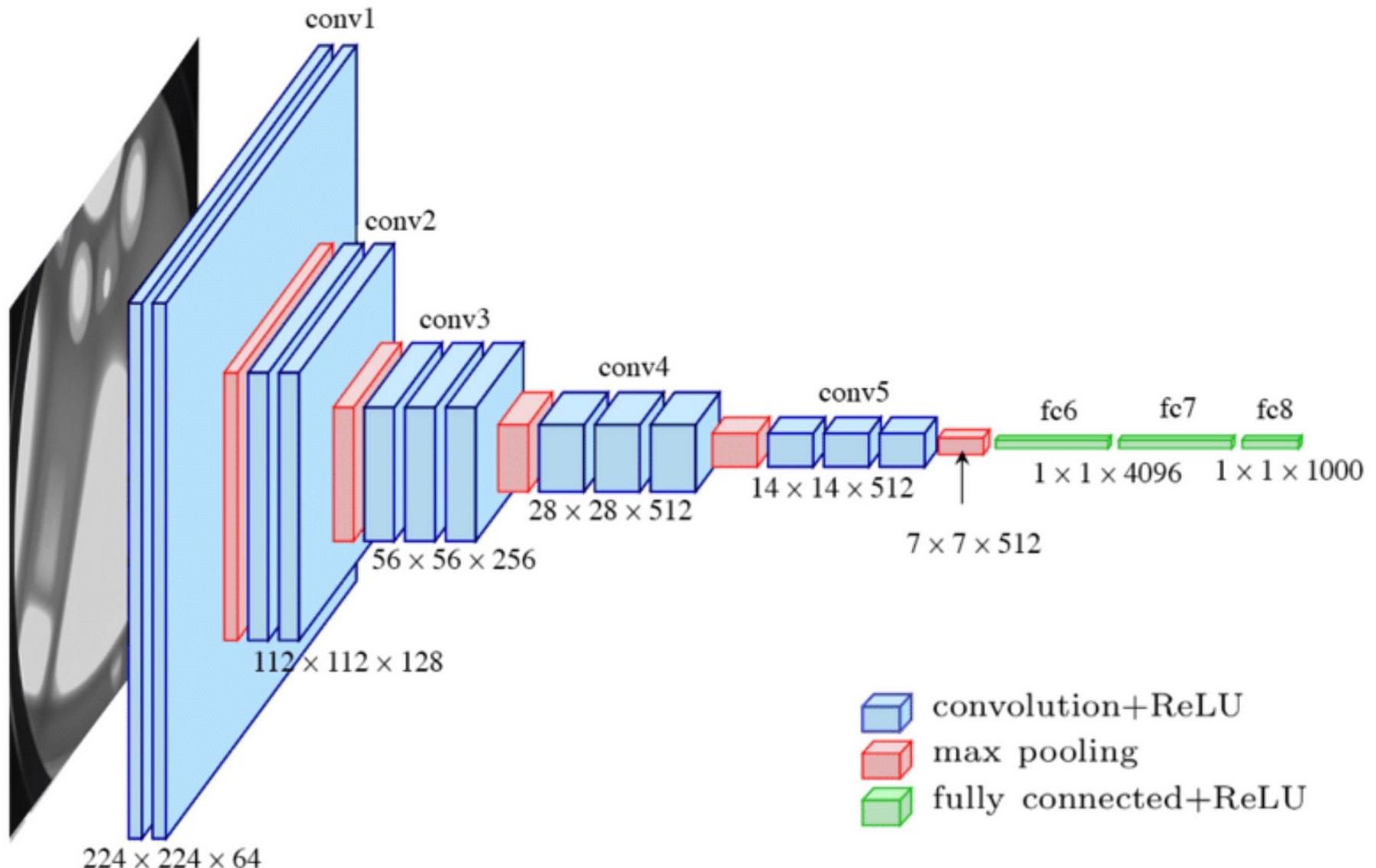
- Krizhevsky et al. 2012
- Used 2 GPUs for training
- First win of ImageNet competition with Deep Neural Networks
- Used big convolutions
- Fully connected at the end
- 7 layers

Takeouts:

- First use of ReLU
- Increasing number of channels
- 16.4%

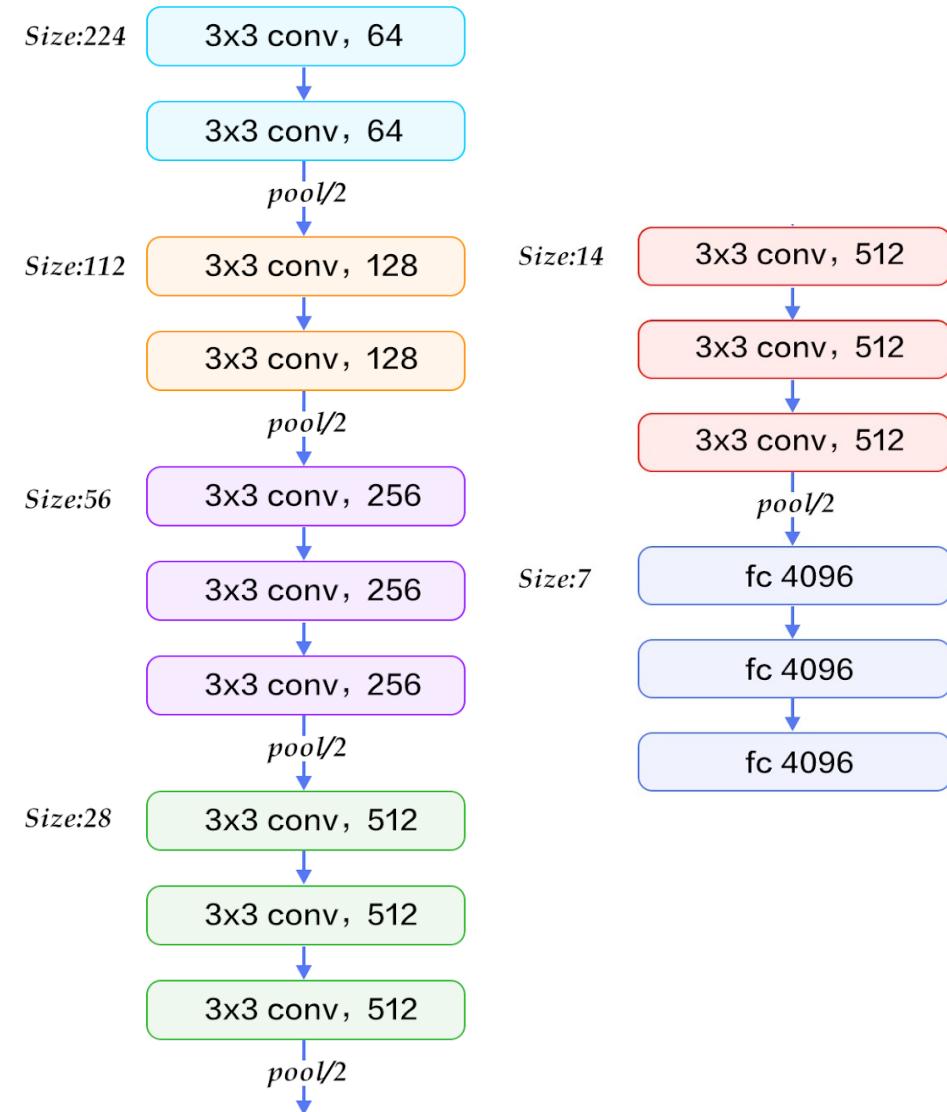


# VGGNet



# VGGNet

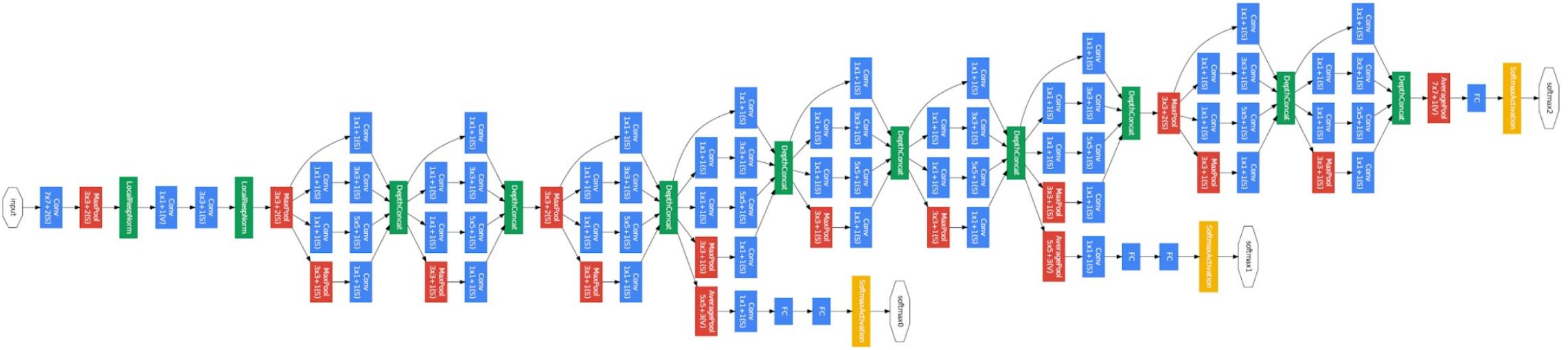
- Simonyan, Zisserman 2014
- All convolutions are 3x3
- Exponentially increasing channels
- 19 layers
- Requires a lot of memory
- Gradient vanishes =(



# GoogLeNet



## Inception module



[Szegedy et al., 2014]

Канал стажировок Яндекса

Спасибо за внимание



Доп. материалы



Y&D