Local descriptors Return of DL

YOUNG &78 YANDEX

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

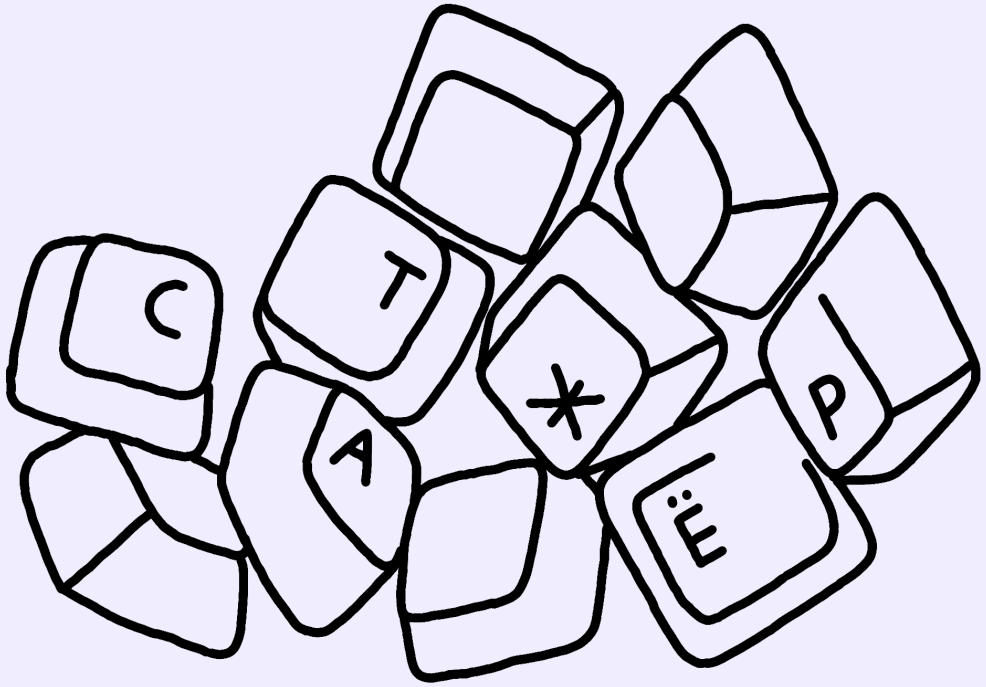
Outline

01 Local image descriptors

02 Backpropagation recap

03 Classification architectures overview

Local image descriptors



01

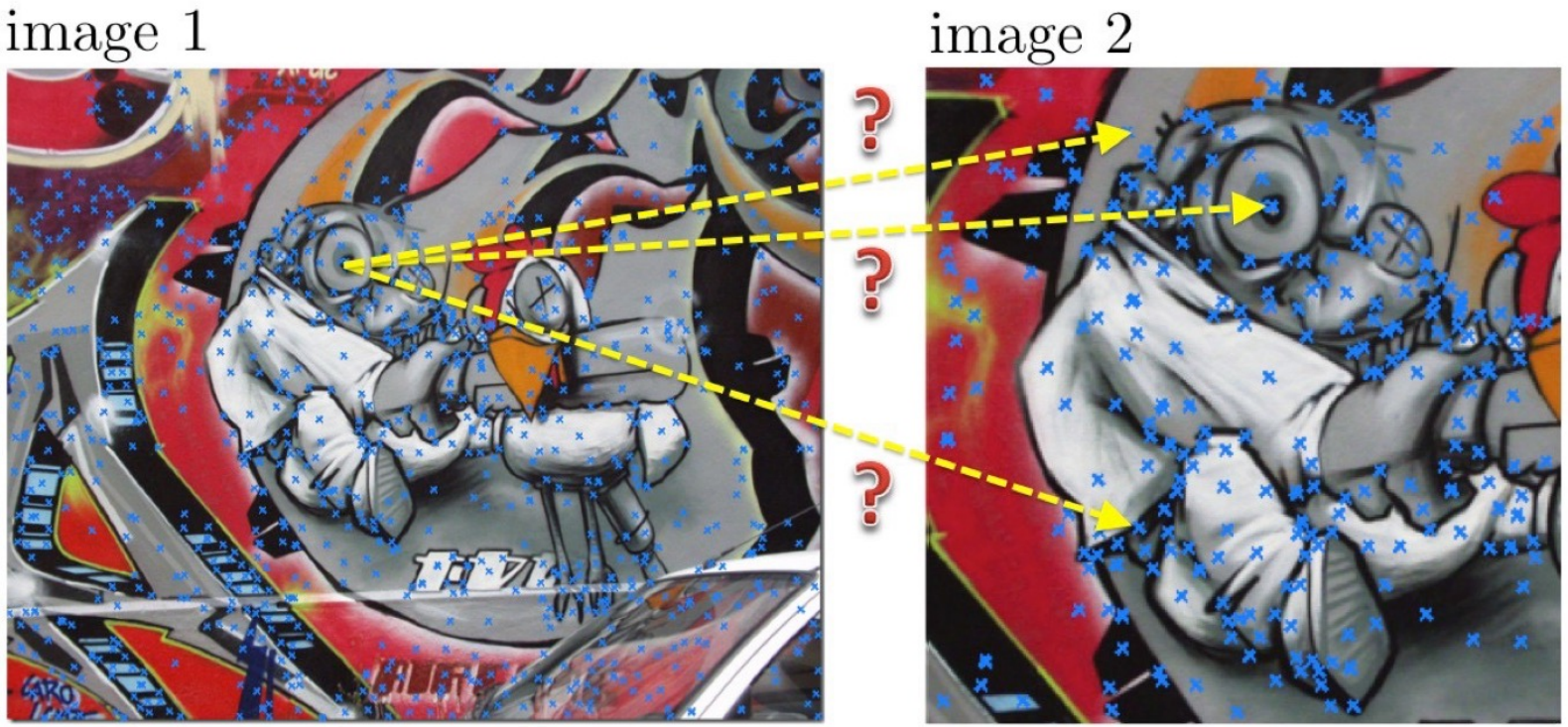
How to match images?

image 1

image 2

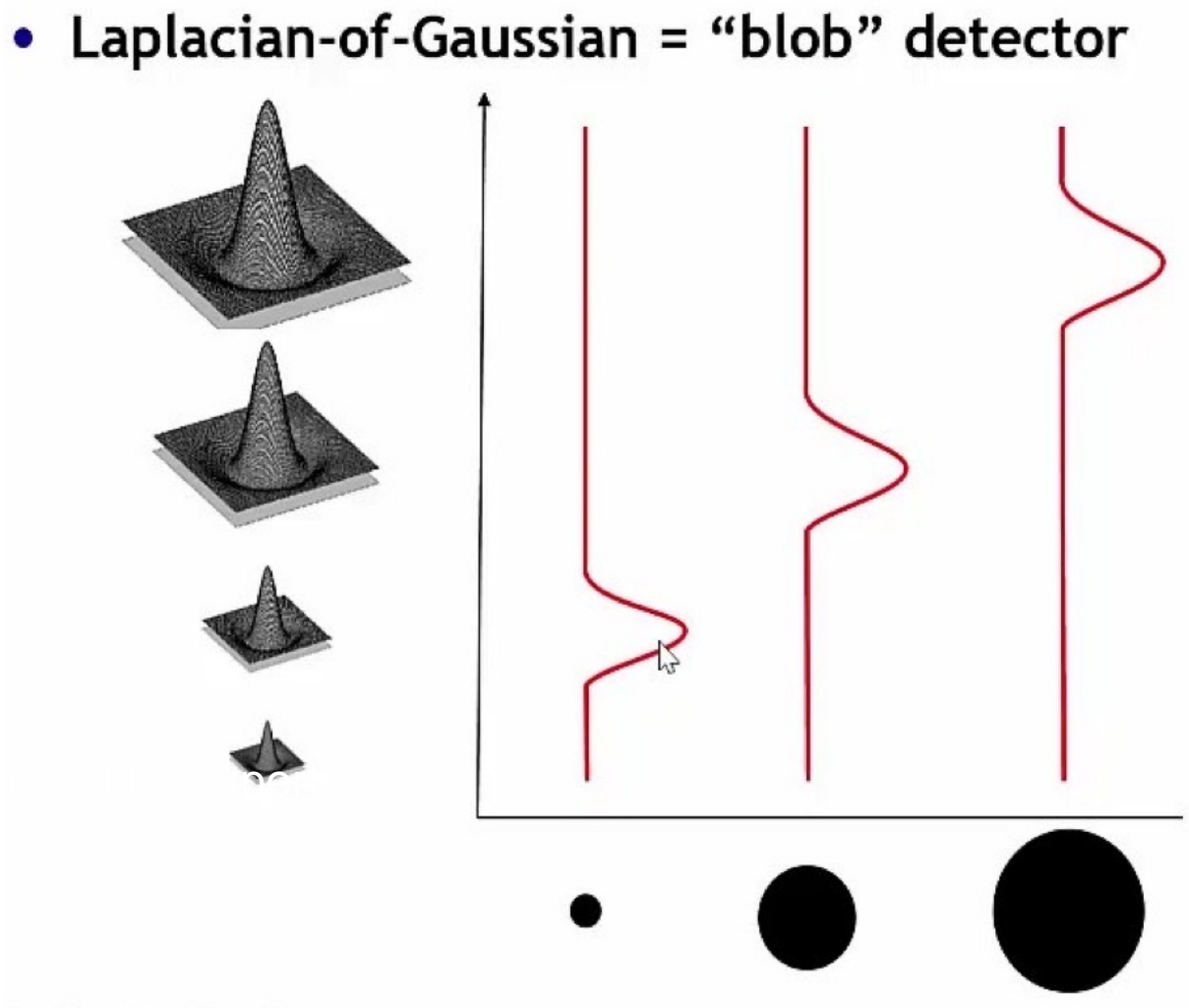
Source: CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

How to match images?



Source: [CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler]

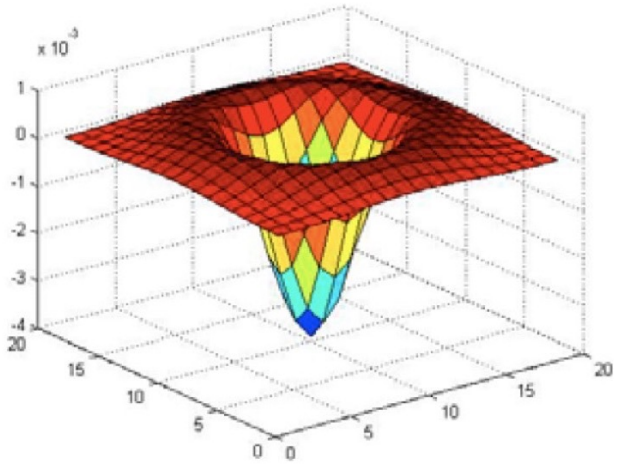
Blob detection

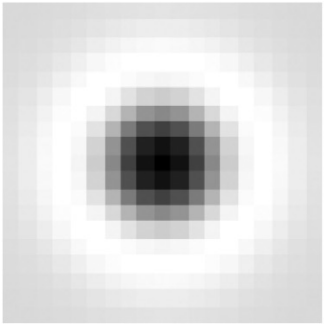


Source: [CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler]

Blob detection

- Laplacian of Gaussian:
  - $\nabla_g^2(x, y, \sigma) = \frac{\partial^2 g(x,y,\sigma)}{\partial x^2} + \frac{\partial^2 g(x,y,\sigma)}{\partial y^2}$ where G is Gaussian
  - $\nabla_g^2(x, y, \sigma) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2+y^2}{2\sigma^2}\right)\exp -\frac{x^2+y^2}{2\sigma^2}$

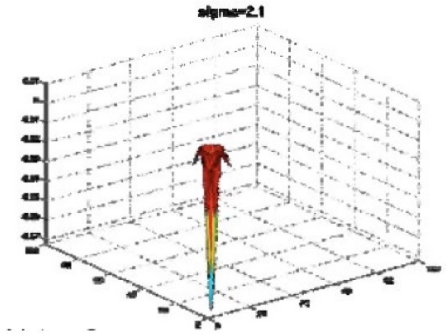


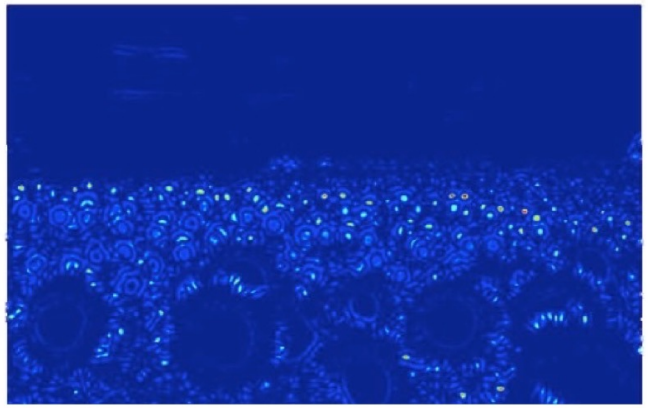


[Source: K. Grauman]

Source: CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

Blob detection

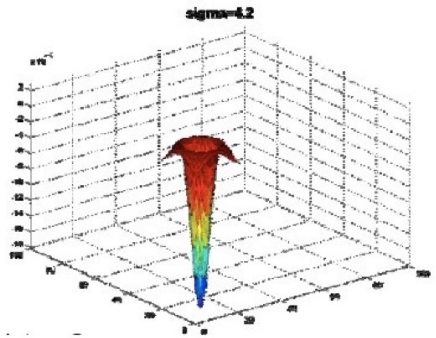


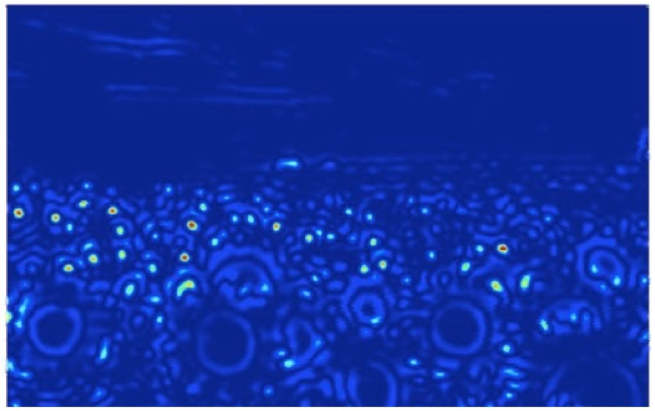


[Source: K. Grauman]

Source: CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

Blob detection

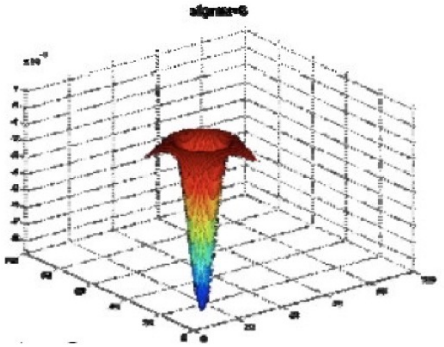


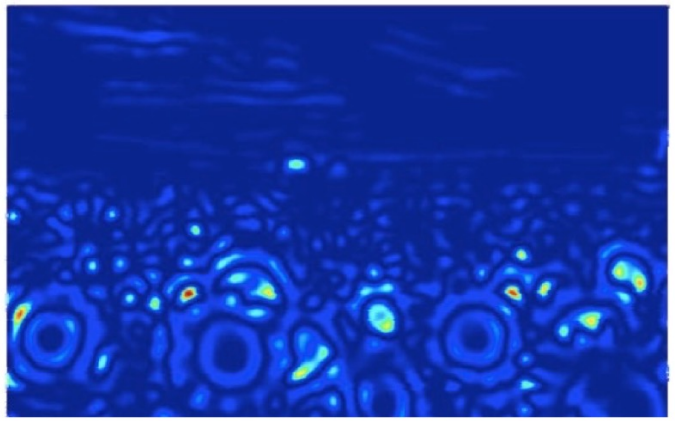


[Source: K. Grauman]

Source: CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

Blob detection

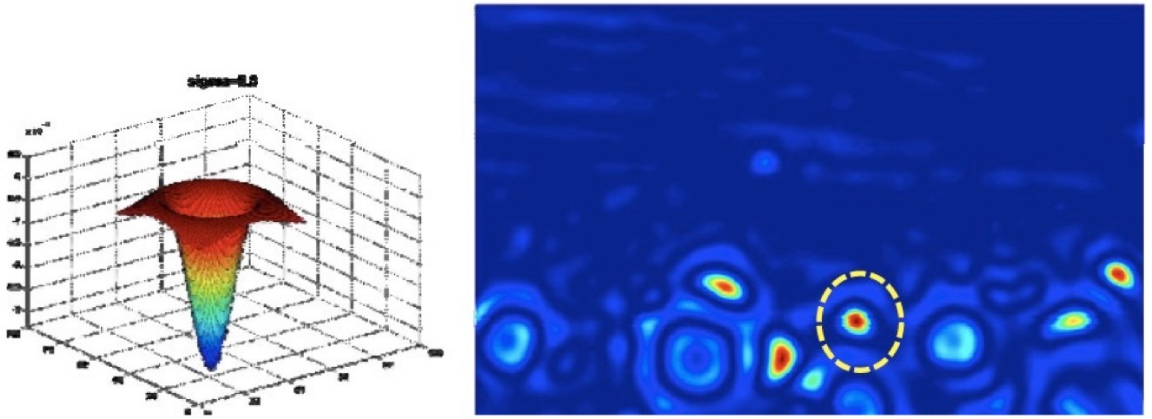




[Source: K. Grauman]

Source: CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

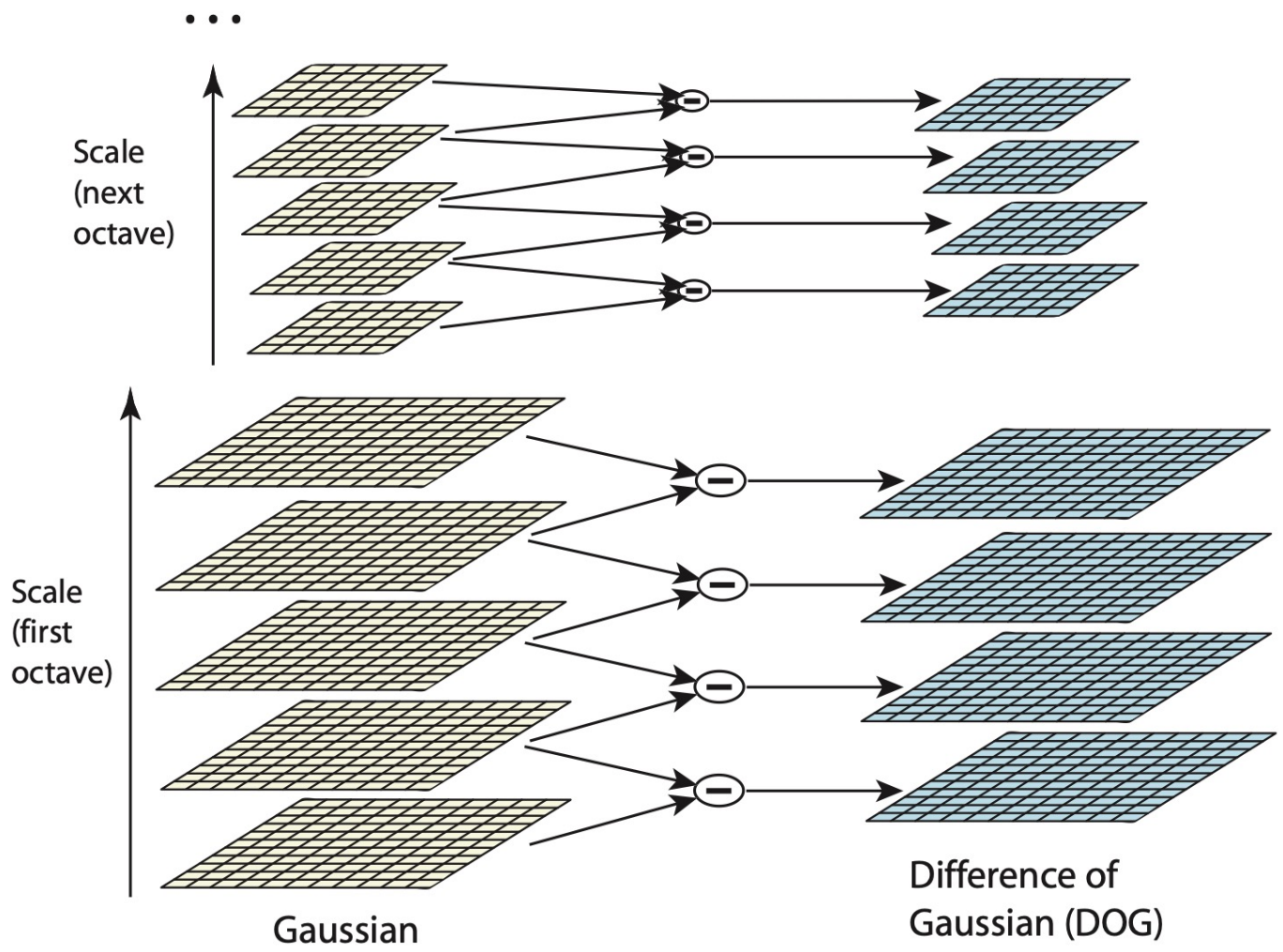
Blob detection



[Source: K. Grauman]

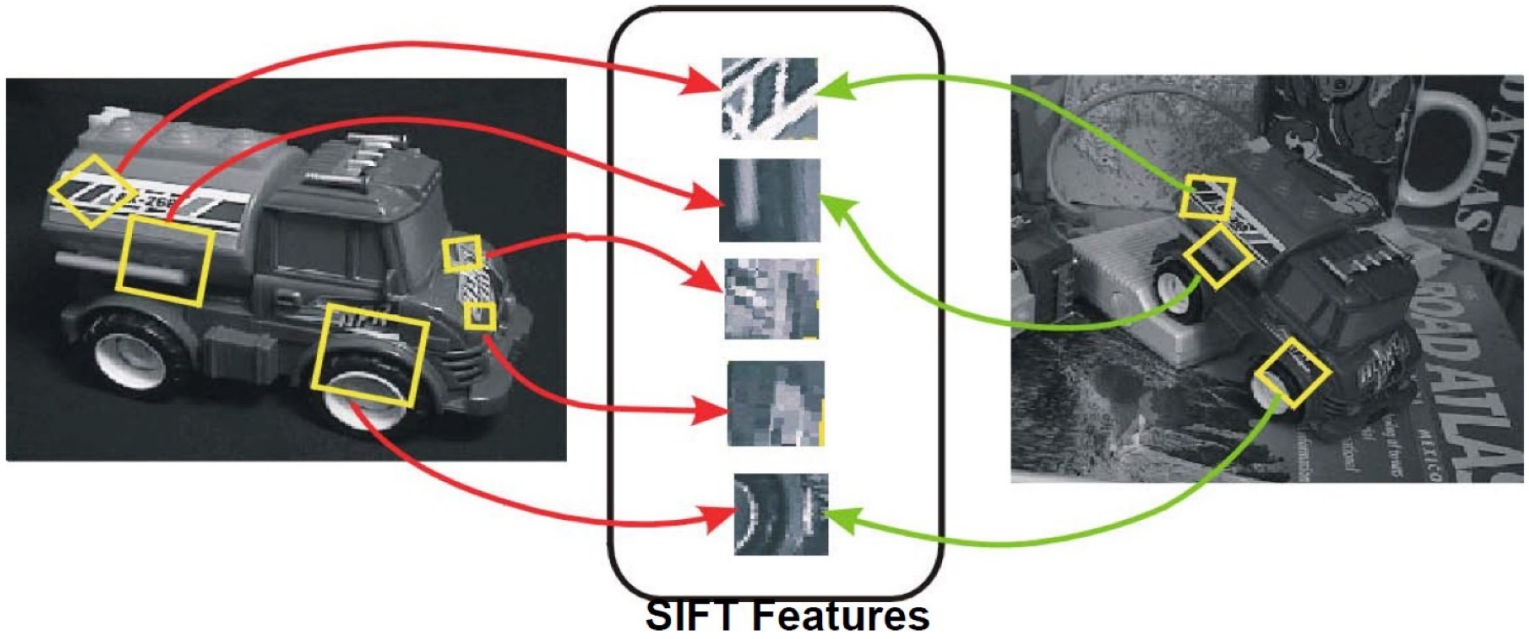
Source: CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

How to get invariant to scale change?



Source: Distinctive Image Features from Scale-Invariant Keypoints

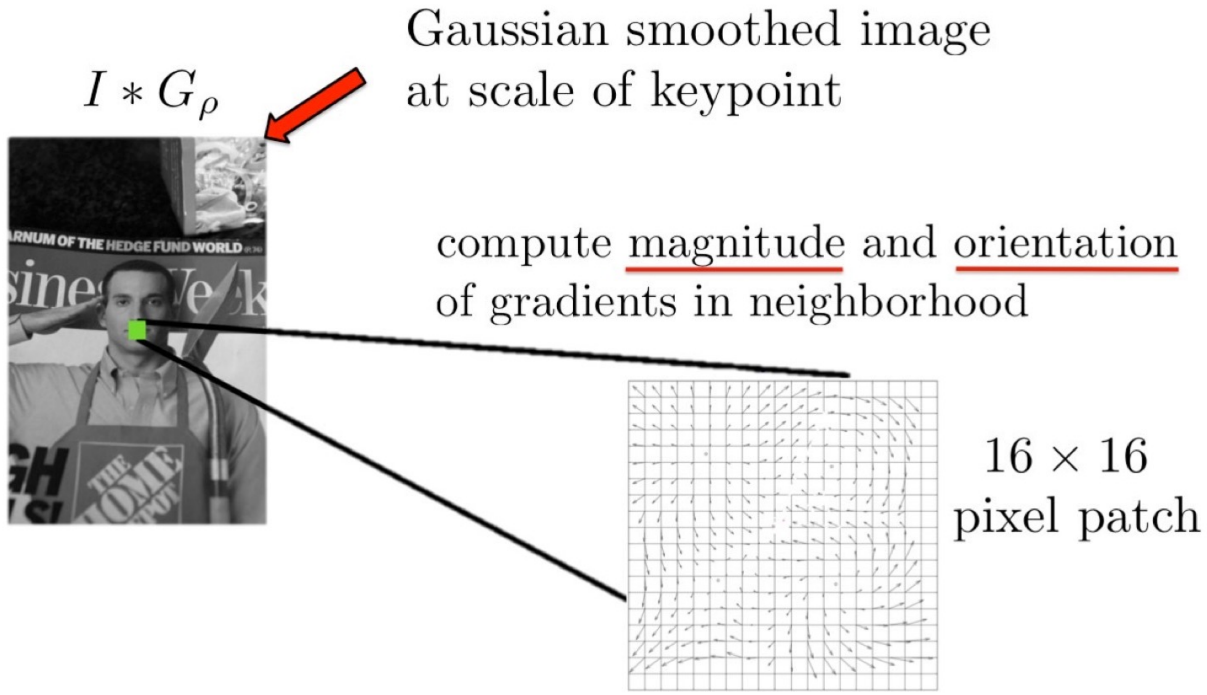
SIFT detector



Source: CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

SIFT detector

- Compute the gradient magnitude and orientation in neighborhood of each keypoint proportional to the detected scale



[Adopted from: F. Flores-Mangas]

Source: CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler

SIFT detector

- Compute the gradient magnitude and orientation in neighborhood of each keypoint proportional to the detected scale

magnitude of gradient:

$$|\nabla I(x,y)| = \sqrt{\left(\frac{\partial(I(x,y) * G_\rho)}{\partial x}\right)^2 + \left(\frac{\partial(I(x,y) * G_\rho)}{\partial y}\right)^2}$$

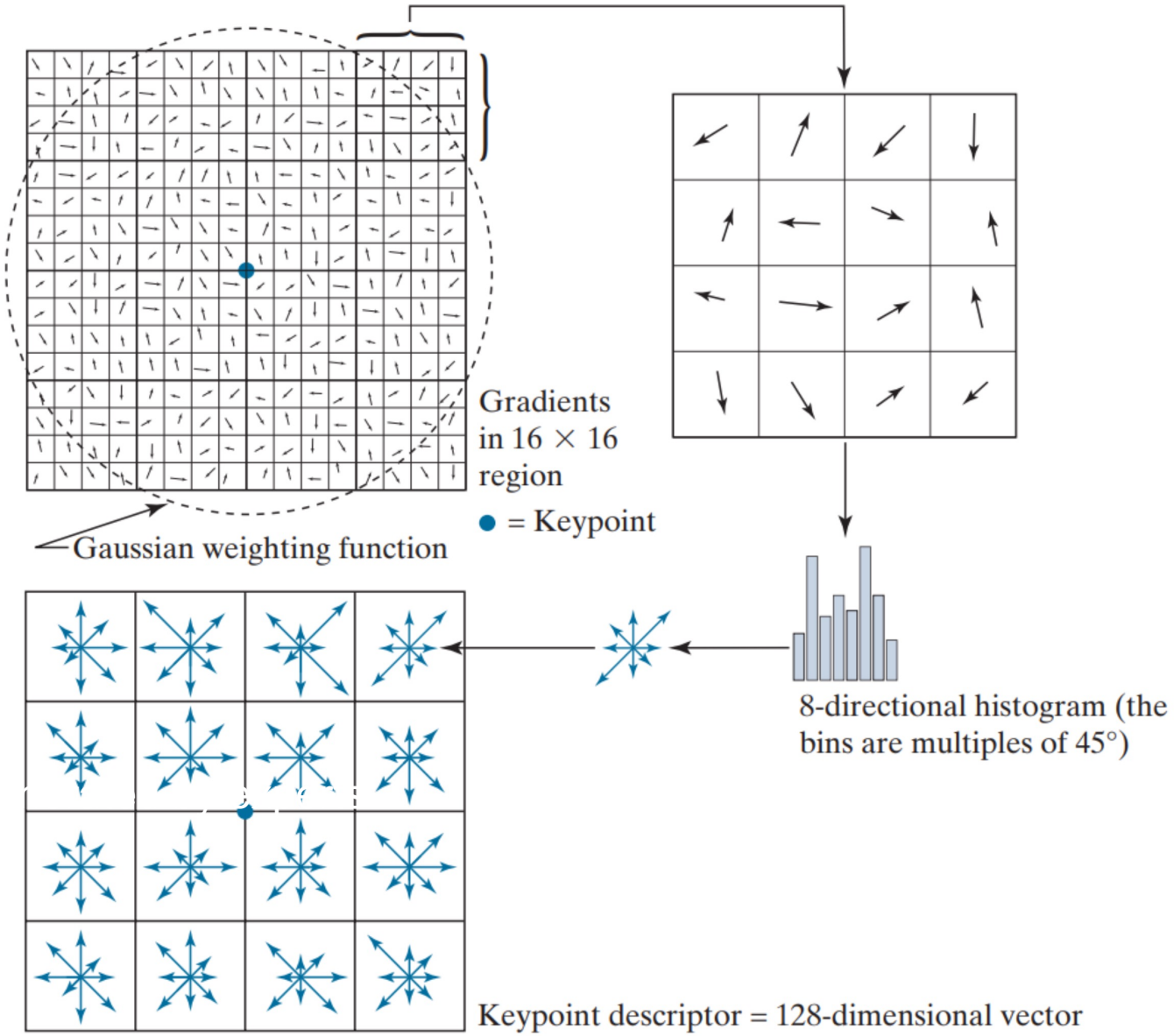
gradient orientation:

$$\theta(x,y) = \arctan\left(\frac{\partial I * G_\rho}{\partial y} / \frac{\partial I * G_\rho}{\partial x}\right)$$

(in case you forgot ;))

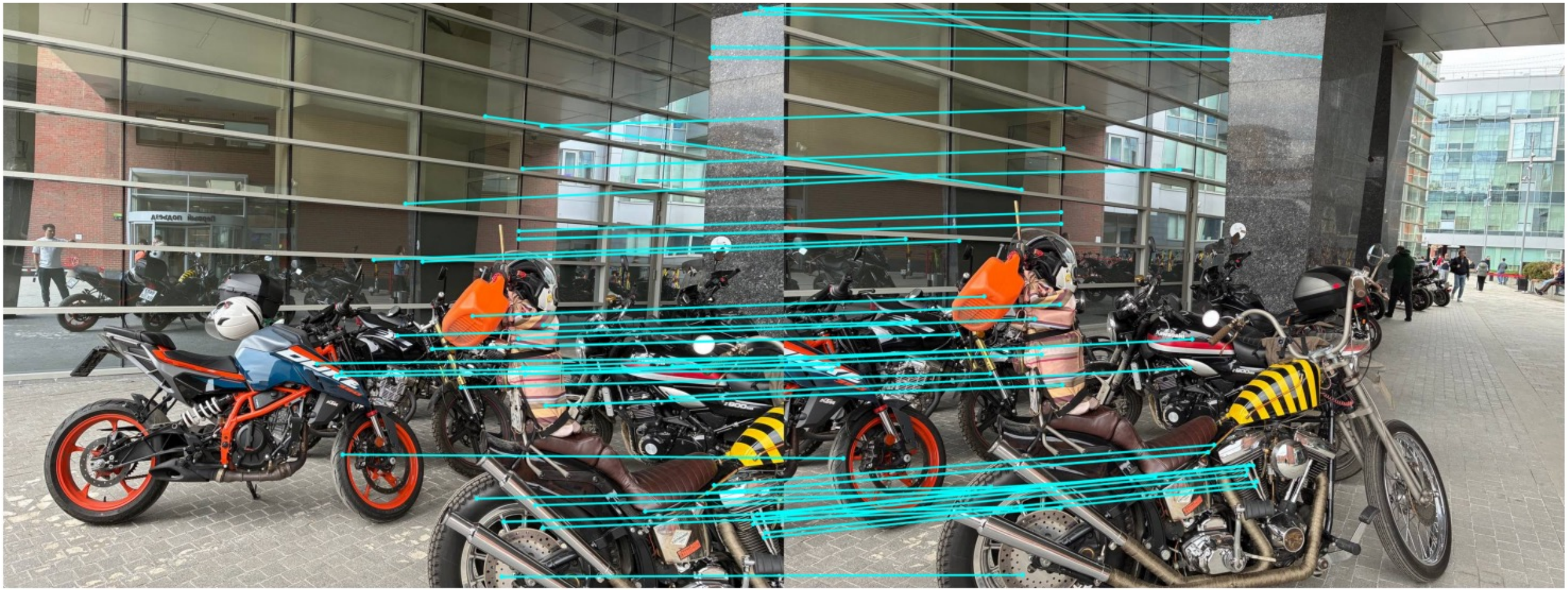
[Adopted from: F. Flores-Mangas]

Source: CSC420 slides by David Lindell, original slides credits: Babak Taati ←Ahmed Ashraf ←Sanja Fidler



Source: https://blog.roboflow.com/sift/

best 50 matches

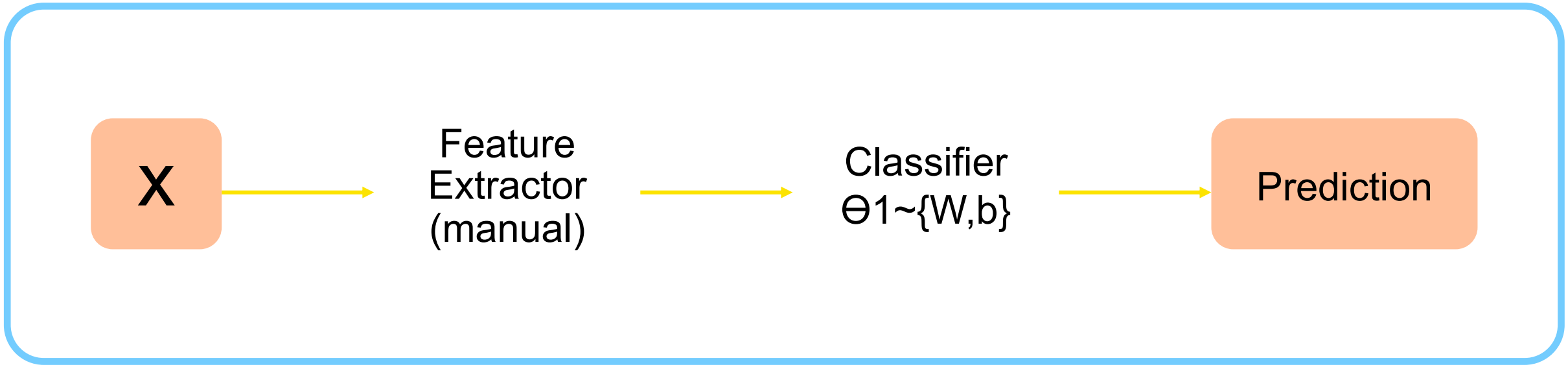


best 50 matches

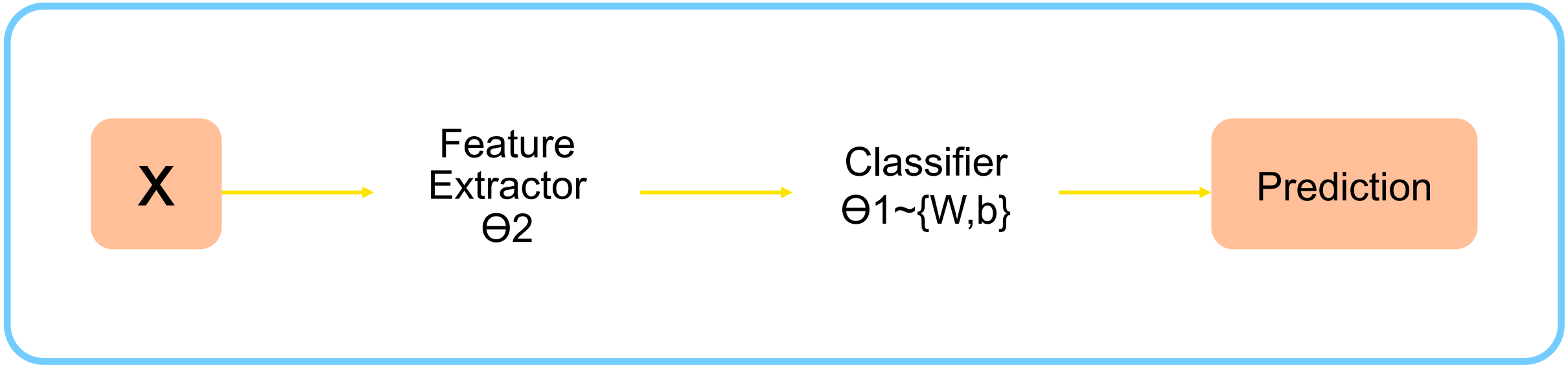
Return to Deep Learning

02

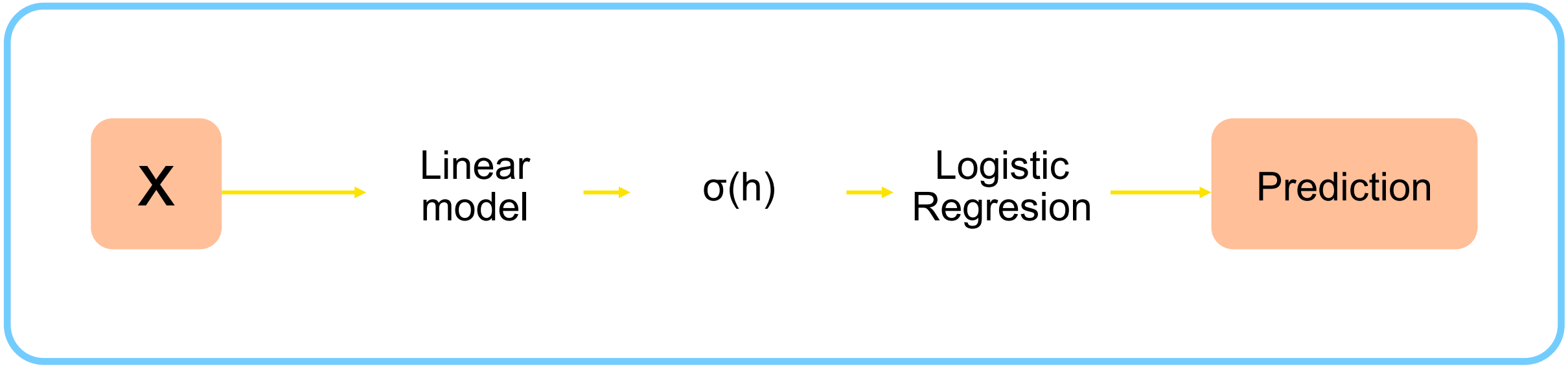
Classic pipeline



NN pipeline



NN pipeline: example



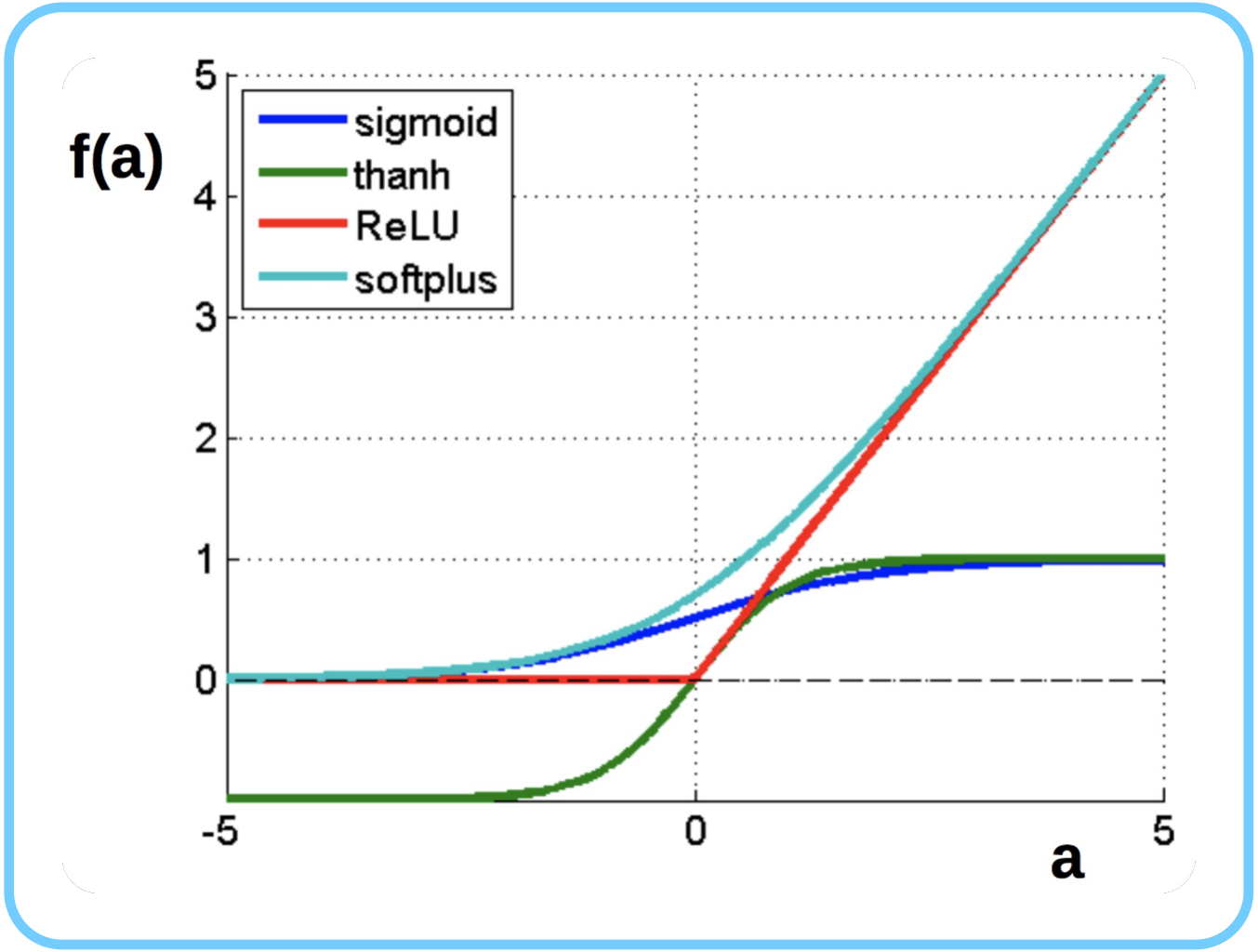
Activation functions: nonlinearities

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

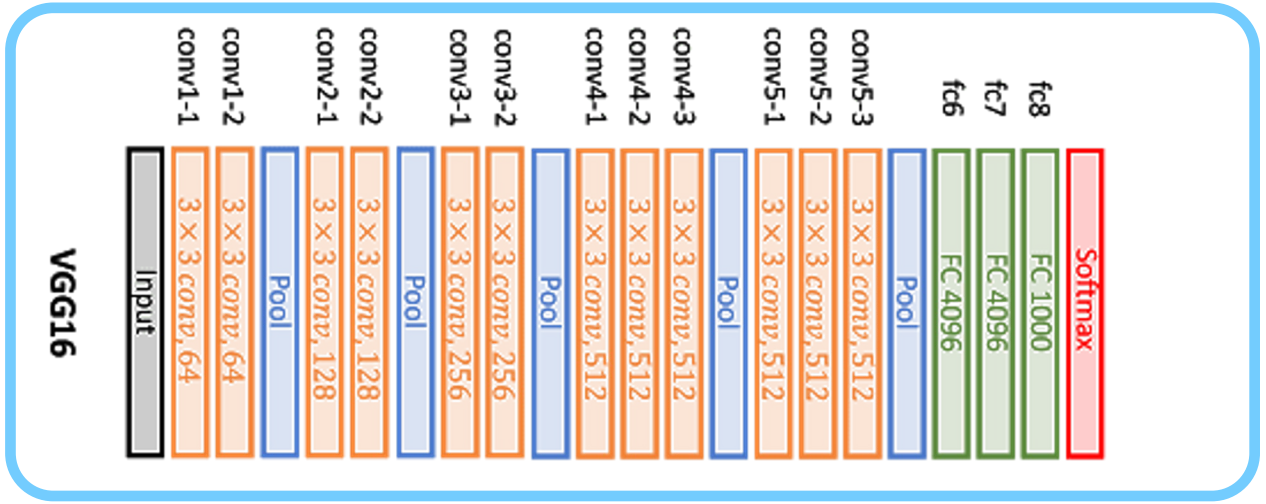
$$f(a) = \tanh(a)$$

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

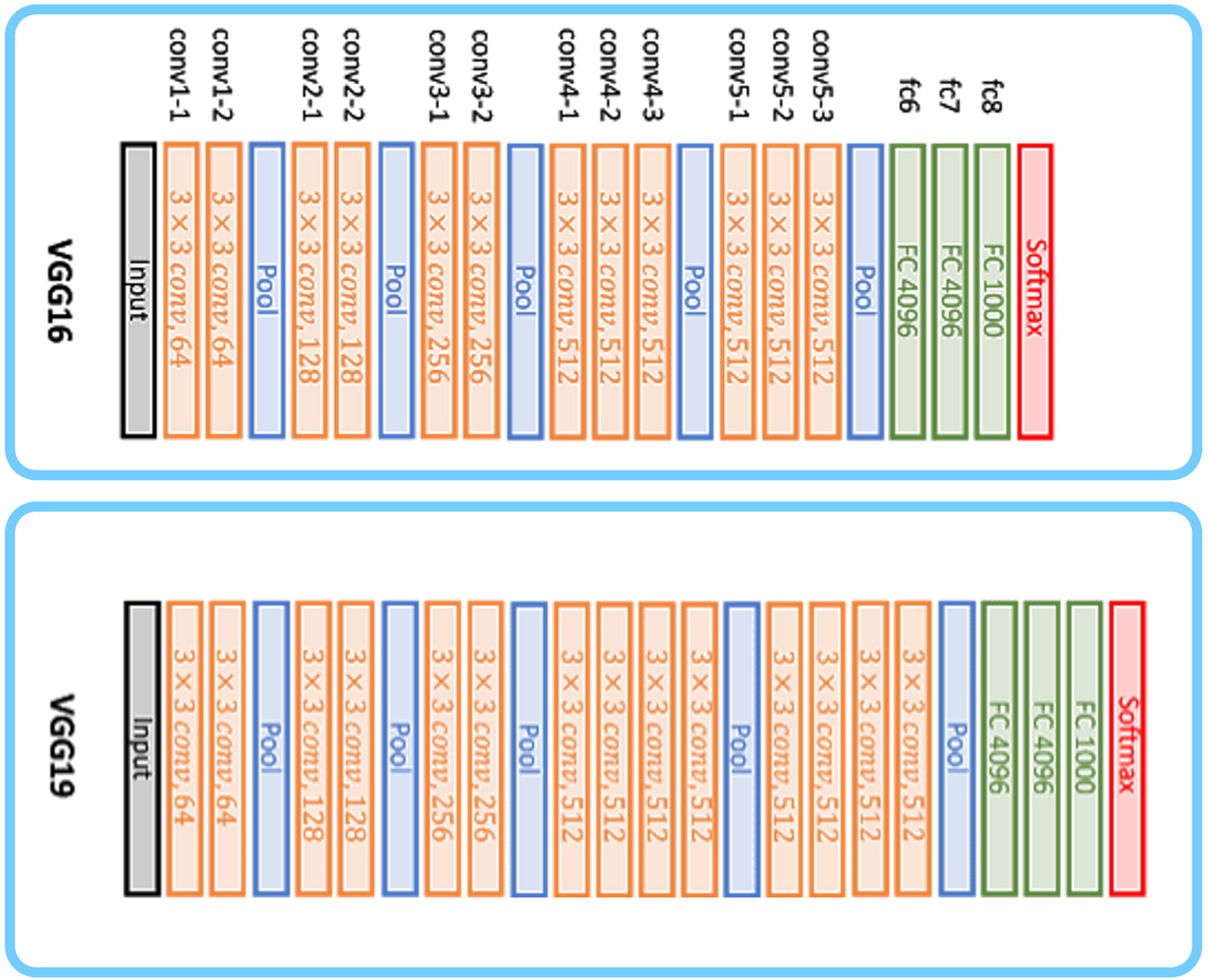
$$f(a) = \log(1 + e^{a})$$

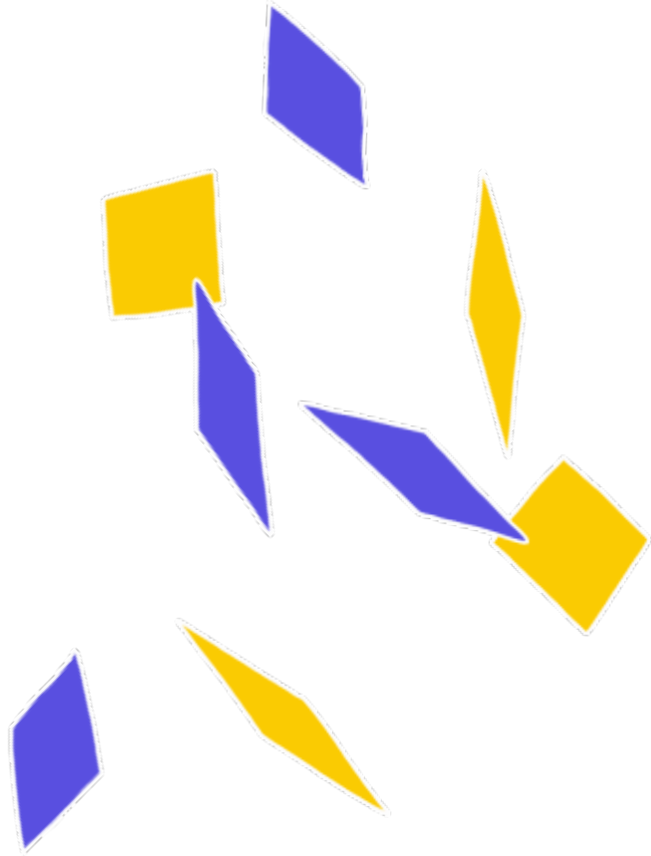


Actually, networks can be deep

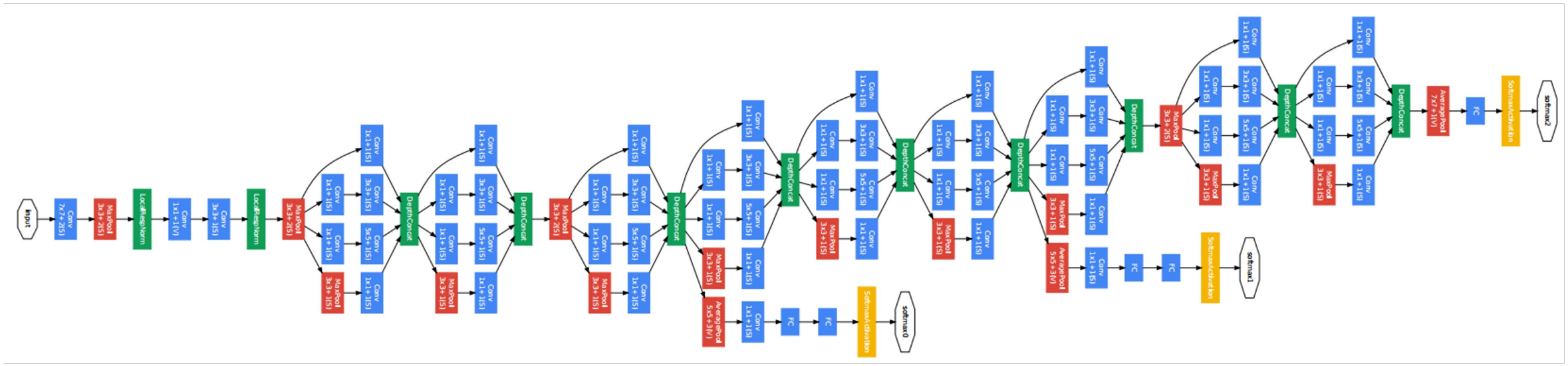


And deeper...


VGG16
Input
3 × 3 conv, 64
3 × 3 conv, 64
Pool
3 × 3 conv, 128
3 × 3 conv, 128
Pool
3 × 3 conv, 256
3 × 3 conv, 256
Pool
3 × 3 conv, 512
3 × 3 conv, 512
Pool
3 × 3 conv, 512
3 × 3 conv, 512
Pool
FC 4096
FC 4096
FC 1000
Softmax
conv1-1
conv1-2
conv2-1
conv2-2
conv3-1
conv3-2
conv4-1
conv4-2
conv4-3
conv5-1
conv5-2
conv5-3
fc6
fc7
fc8
VGG19
Input
3 × 3 conv, 64
3 × 3 conv, 64
Pool
3 × 3 conv, 128
3 × 3 conv, 128
Pool
3 × 3 conv, 256
3 × 3 conv, 256
Pool
3 × 3 conv, 512
3 × 3 conv, 512
Pool
3 × 3 conv, 512
3 × 3 conv, 512
Pool
FC 4096
FC 4096
FC 1000
Softmax



Much deeper...

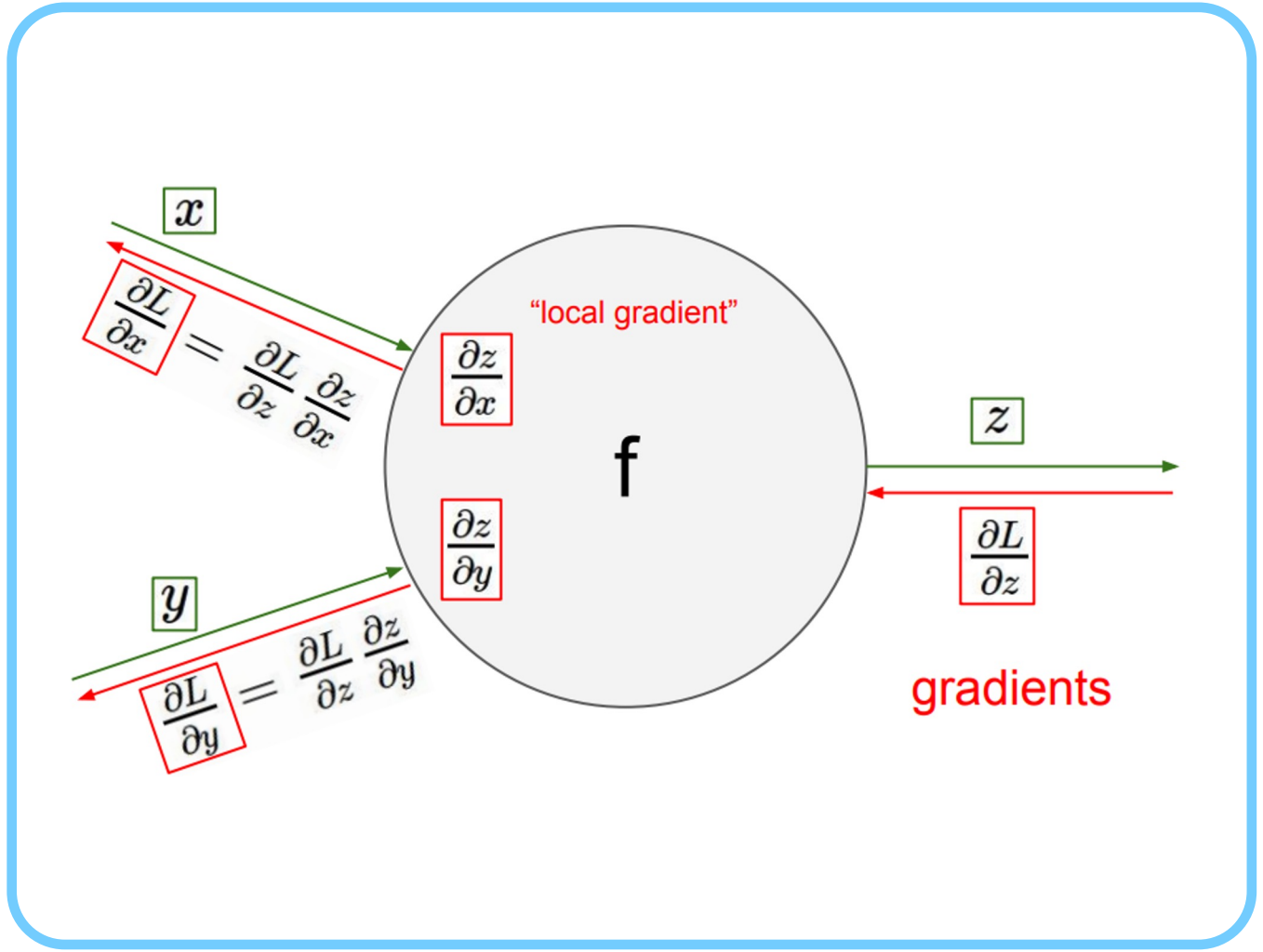

input
Conv 7x7+2(S)
MaxPool 3x3+2(S)
LocalResponseNorm
Conv 1x1+1(V)
Conv 3x3+1(S)
LocalResponseNorm
MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
MaxPool 3x3+1(S)
DepthConcat
AveragePool 5x5+3(V)
FC
SoftmaxActivation
softmax0
softmax1
softmax2

Backpropagation and chain rule

Chain rule is just simple math:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x}$$

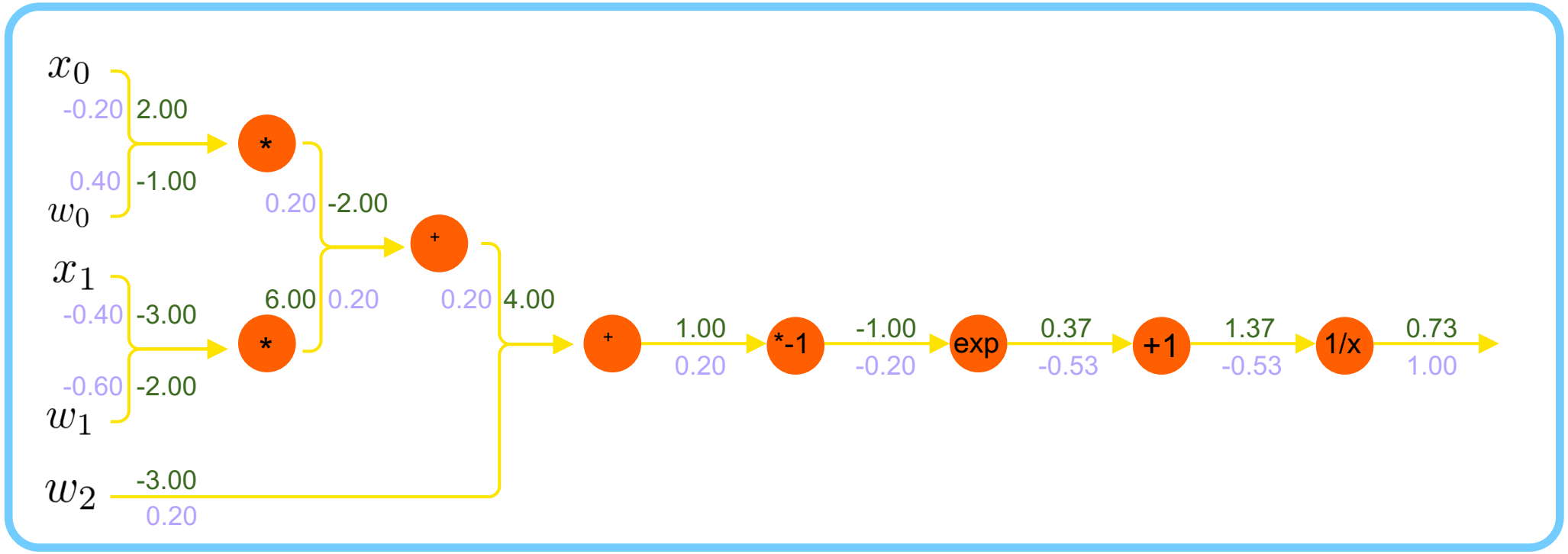
Backprop is just way to use it in NN training



source

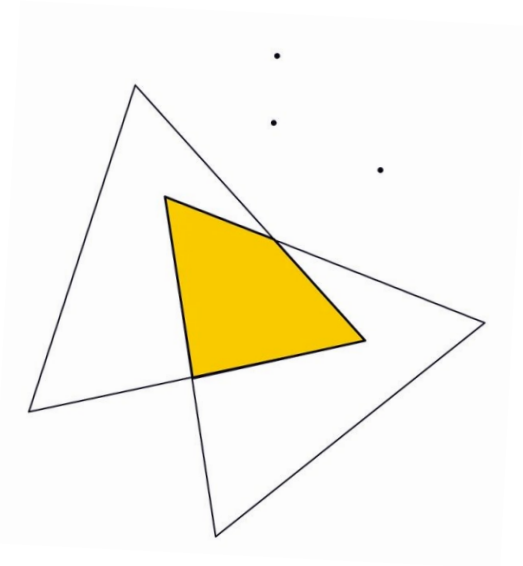
Backpropagation example

$$L(\boldsymbol{w}, \boldsymbol{x}) = \frac{1}{1 + \exp\left(- (x_0 w_0 + x_1 w_1 + w_2)\right)}$$



source

Backpropagation: matrix form

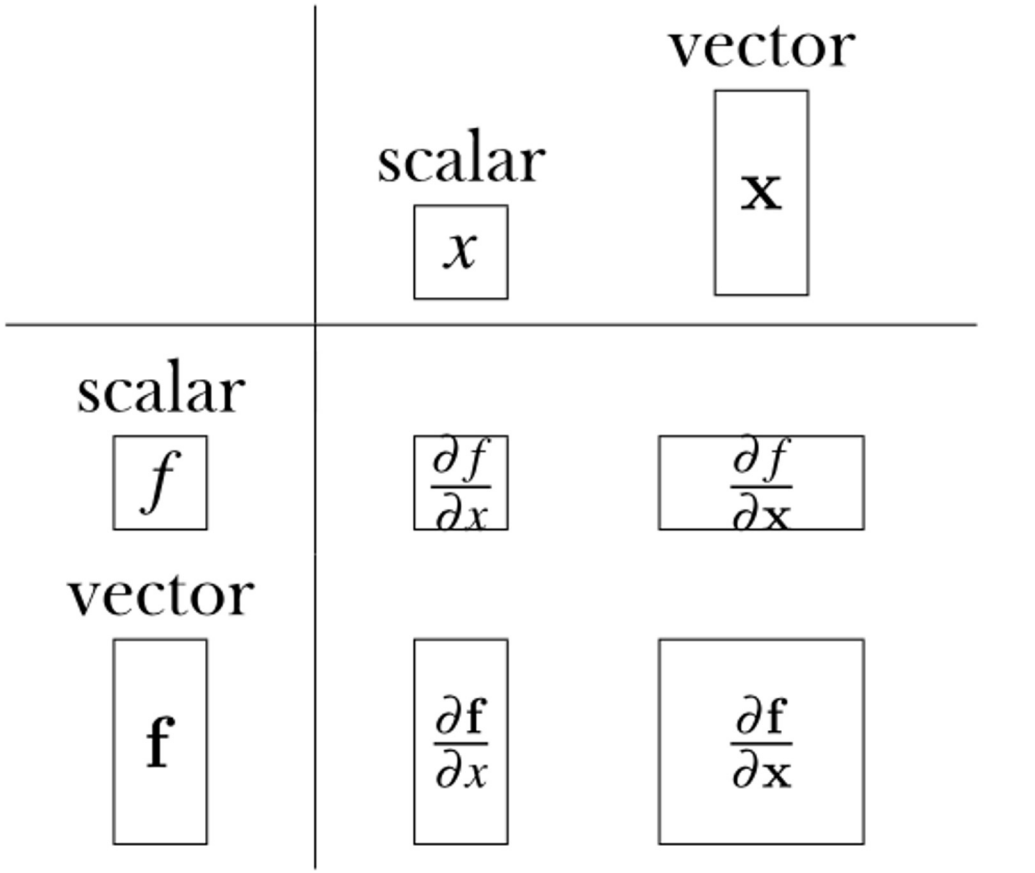


$$
\begin{aligned}
y_1 &= f_1(\mathbf{x}) = x_1 \\
y_2 &= f_2(\mathbf{x}) = x_2 \\
&\vdots \\
y_n &= f_n(\mathbf{x}) = x_n
\end{aligned}
$$

$$
\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \nabla f_1(\mathbf{x}) \\ \nabla f_2(\mathbf{x}) \\ \cdots \\ \nabla f_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}} f_1(\mathbf{x}) \\ \frac{\partial}{\partial \mathbf{x}} f_2(\mathbf{x}) \\ \cdots \\ \frac{\partial}{\partial \mathbf{x}} f_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\mathbf{x}) & \frac{\partial}{\partial x_2} f_1(\mathbf{x}) & \cdots & \frac{\partial}{\partial x_n} f_1(\mathbf{x}) \\ \frac{\partial}{\partial x_1} f_2(\mathbf{x}) & \frac{\partial}{\partial x_2} f_2(\mathbf{x}) & \cdots & \frac{\partial}{\partial x_n} f_2(\mathbf{x}) \\ & & \cdots & \\ \frac{\partial}{\partial x_1} f_m(\mathbf{x}) & \frac{\partial}{\partial x_2} f_m(\mathbf{x}) & \cdots & \frac{\partial}{\partial x_n} f_m(\mathbf{x}) \end{bmatrix}
$$

source

Backpropagation: matrix form



source

Backpropagation: matrix form

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{x}} f_1(\mathbf{x}) \\ \frac{\partial}{\partial \mathbf{x}} f_2(\mathbf{x}) \\ \cdots \\ \frac{\partial}{\partial \mathbf{x}} f_m(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x_1} f_1(\mathbf{x}) & \frac{\partial}{\partial x_2} f_1(\mathbf{x}) & \cdots & \frac{\partial}{\partial x_n} f_1(\mathbf{x}) \\ \frac{\partial}{\partial x_1} f_2(\mathbf{x}) & \frac{\partial}{\partial x_2} f_2(\mathbf{x}) & \cdots & \frac{\partial}{\partial x_n} f_2(\mathbf{x}) \\ & & \cdots & \\ \frac{\partial}{\partial x_1} f_m(\mathbf{x}) & \frac{\partial}{\partial x_2} f_m(\mathbf{x}) & \cdots & \frac{\partial}{\partial x_n} f_m(\mathbf{x}) \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial}{\partial x_1} x_1 & \frac{\partial}{\partial x_2} x_1 & \cdots & \frac{\partial}{\partial x_n} x_1 \\ \frac{\partial}{\partial x_1} x_2 & \frac{\partial}{\partial x_2} x_2 & \cdots & \frac{\partial}{\partial x_n} x_2 \\ & & \cdots & \\ \frac{\partial}{\partial x_1} x_n & \frac{\partial}{\partial x_2} x_n & \cdots & \frac{\partial}{\partial x_n} x_n \end{bmatrix}$$

$$\text{(and since } \frac{\partial}{\partial x_j} x_i = 0 \text{ for } j \neq i\text{)}$$

$$= \begin{bmatrix} \frac{\partial}{\partial x_1} x_1 & 0 & \cdots & 0 \\ 0 & \frac{\partial}{\partial x_2} x_2 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & \frac{\partial}{\partial x_n} x_n \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

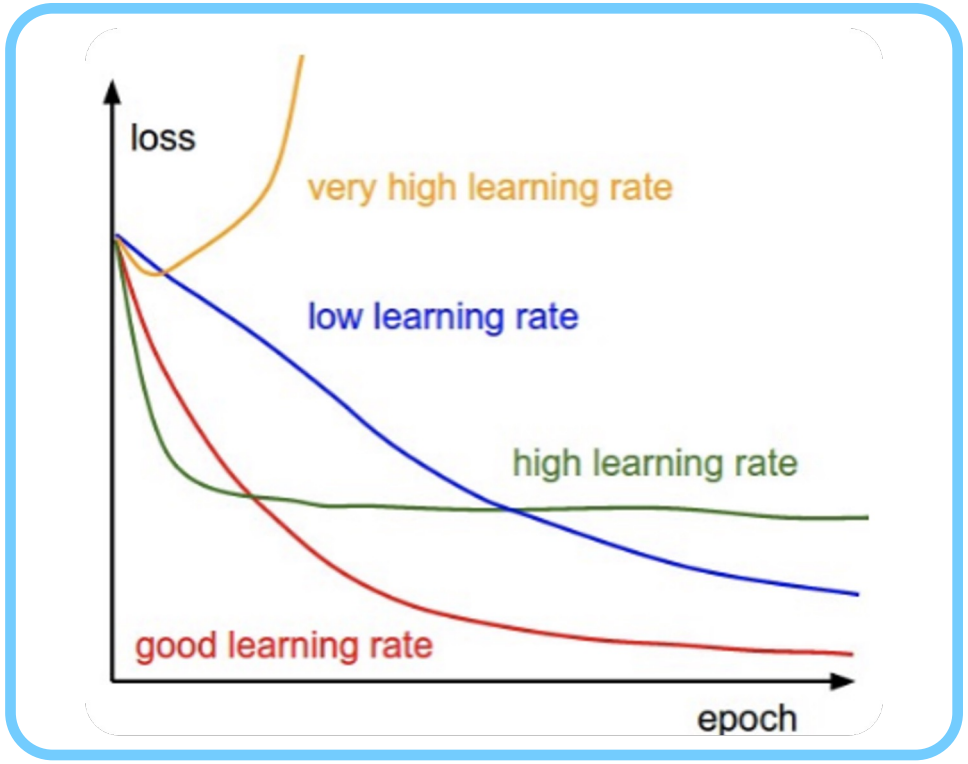
$$= I \quad (I \text{ is the identity matrix with ones down the diagonal})$$

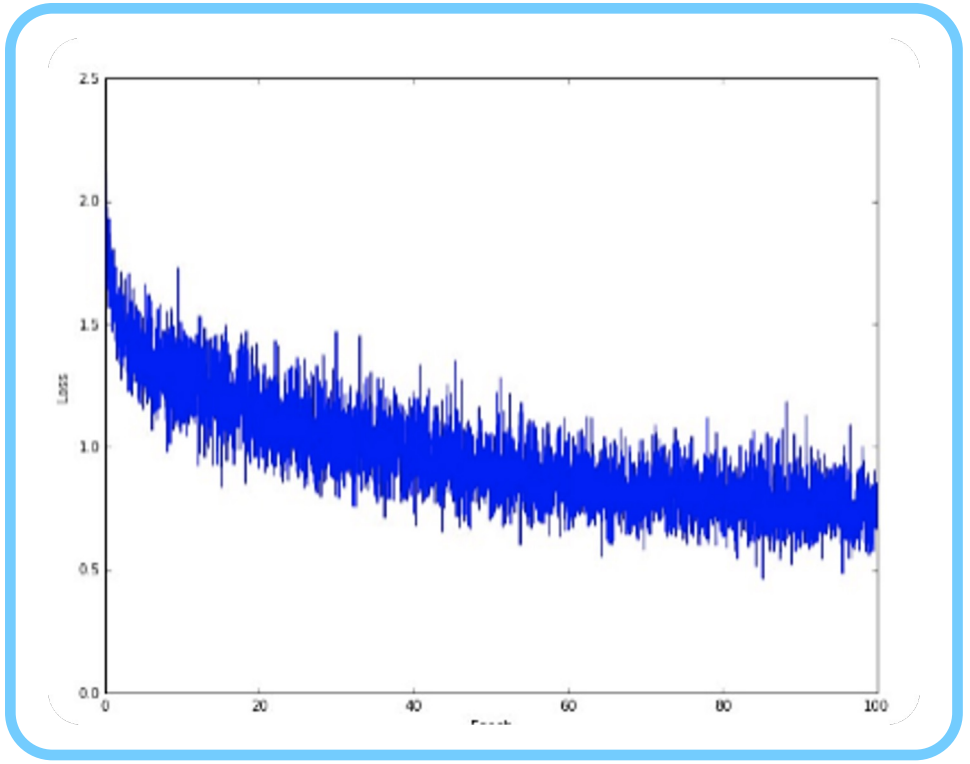
source

Gradient optimization

Stochastic gradient descent is used to optimize NN parameters

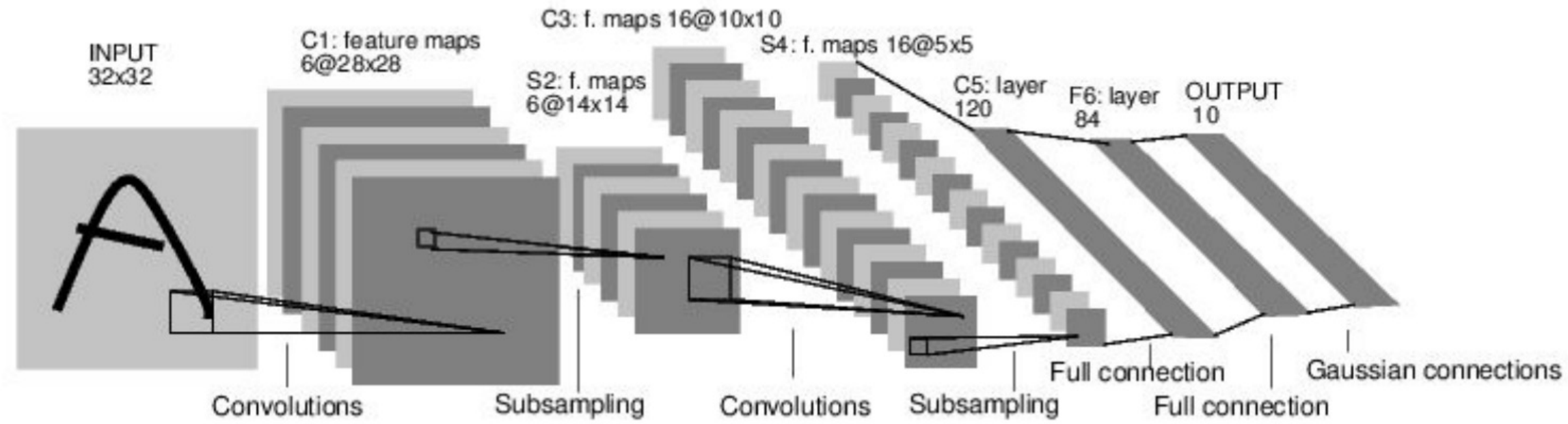
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \frac{dL}{d\boldsymbol{w}}$$





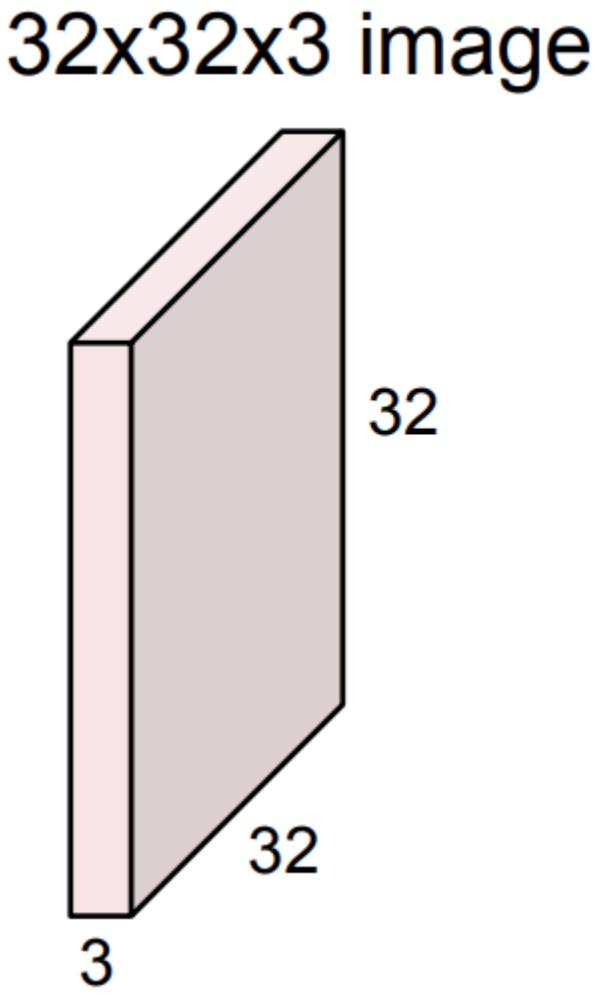
source

Convolutions in Neural Networks



[LeNet-5, LeCun 1998]

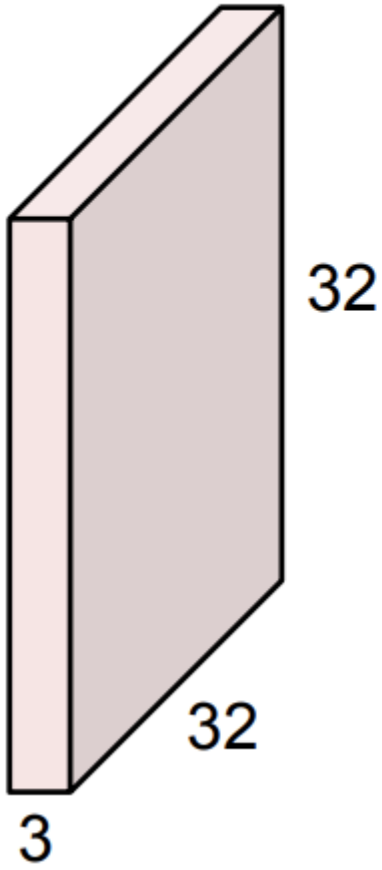
Convolutional layer



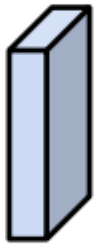
Source: Stanford CS 231 lecture

Convolutional layer

32x32x3 image



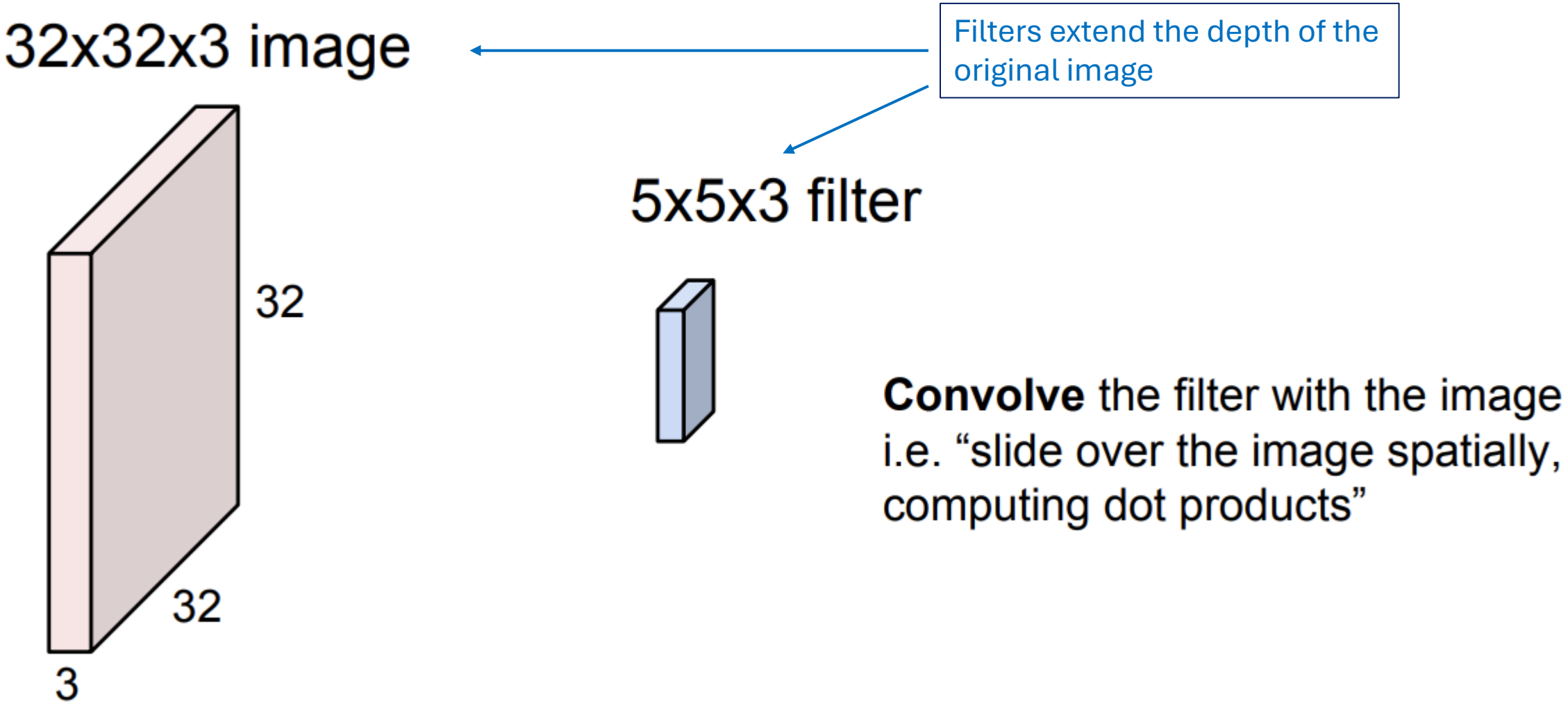
5x5x3 filter



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

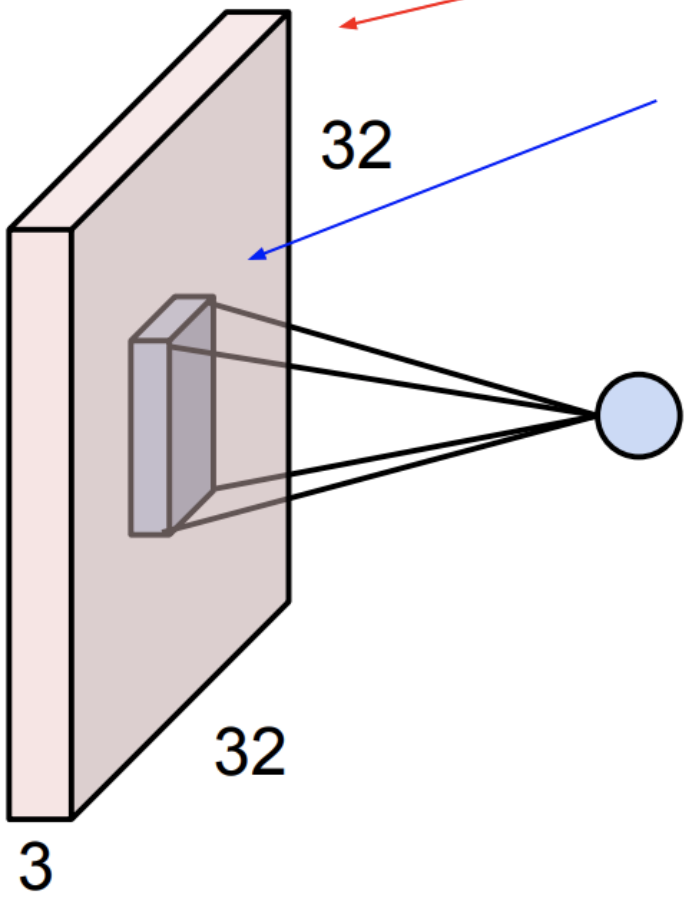
Source: Stanford CS 231 lecture

Convolutional layer



Source: Stanford CS 231 lecture

Convolutional layer



32x32x3 image

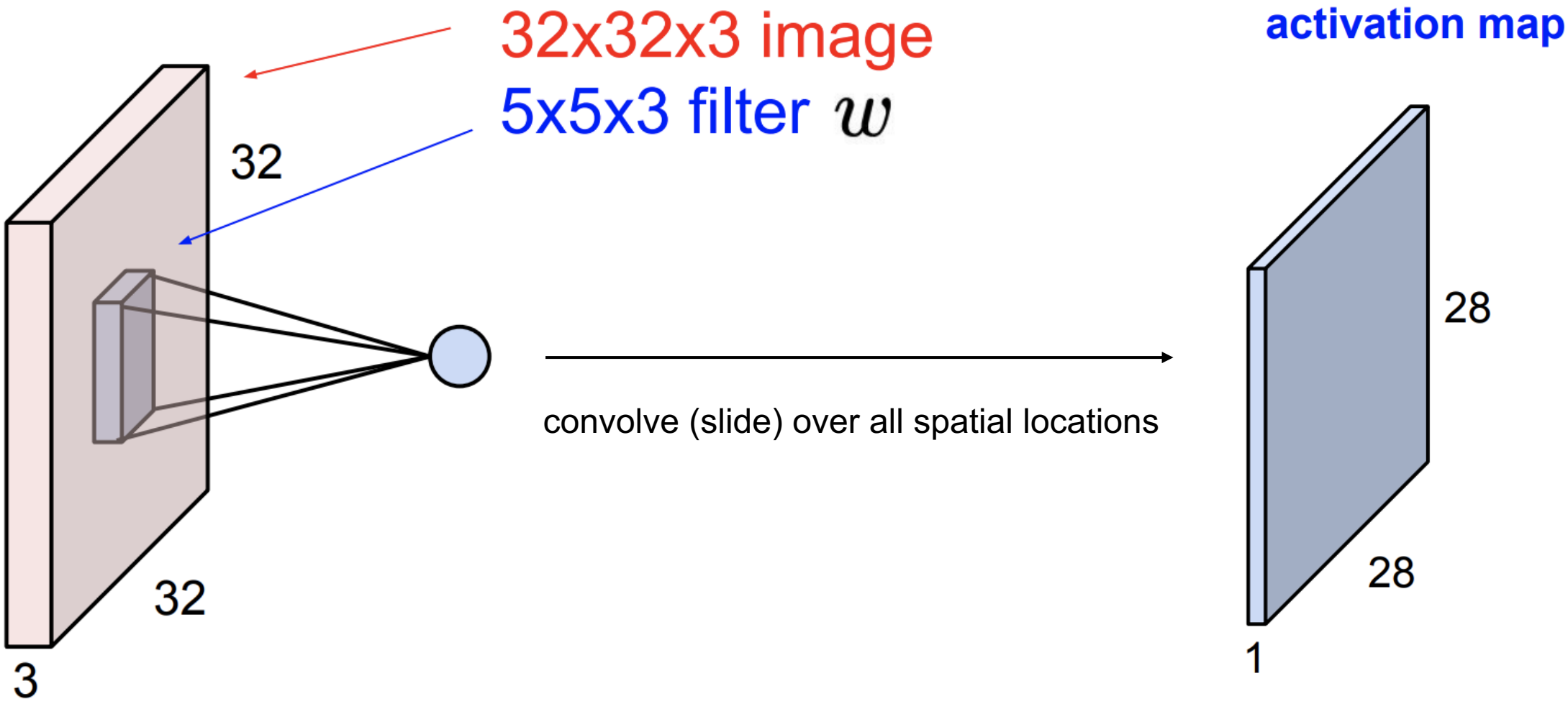
5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

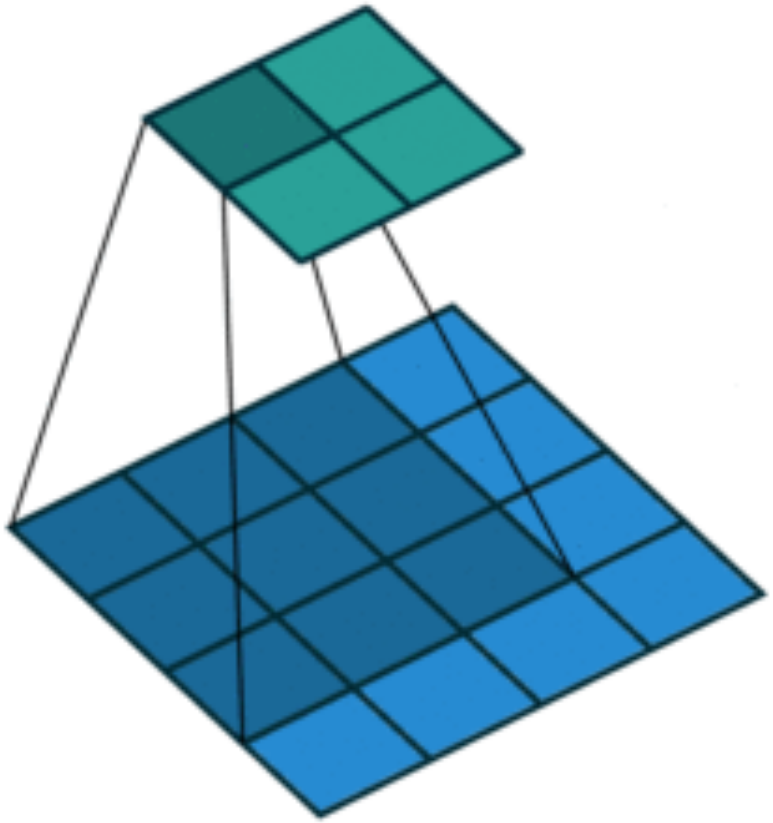
Source: Stanford CS 231 lecture

Convolutional layer



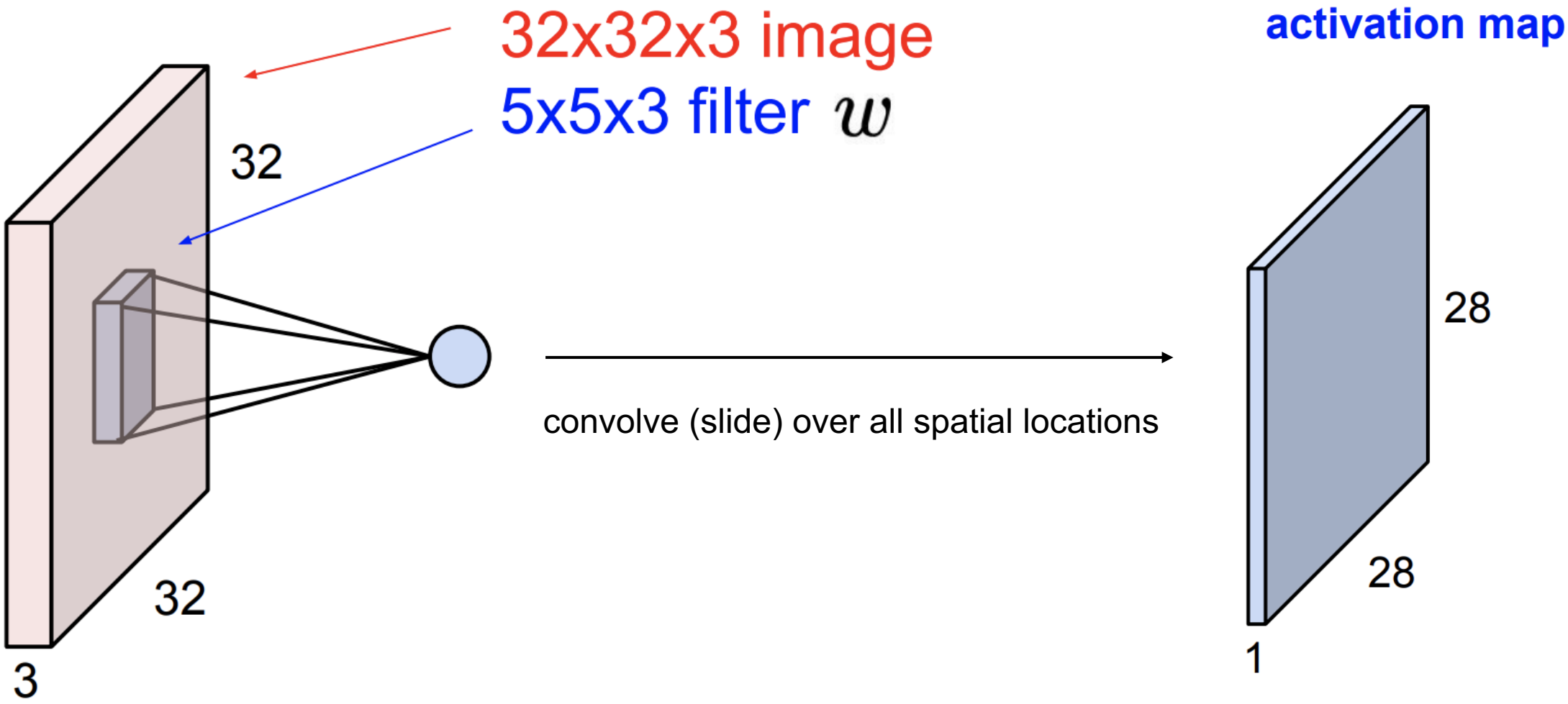
Source: Stanford CS 231 lecture

Convolutional layer



Source: https://github.com/vdumoulin/conv_arithmetic

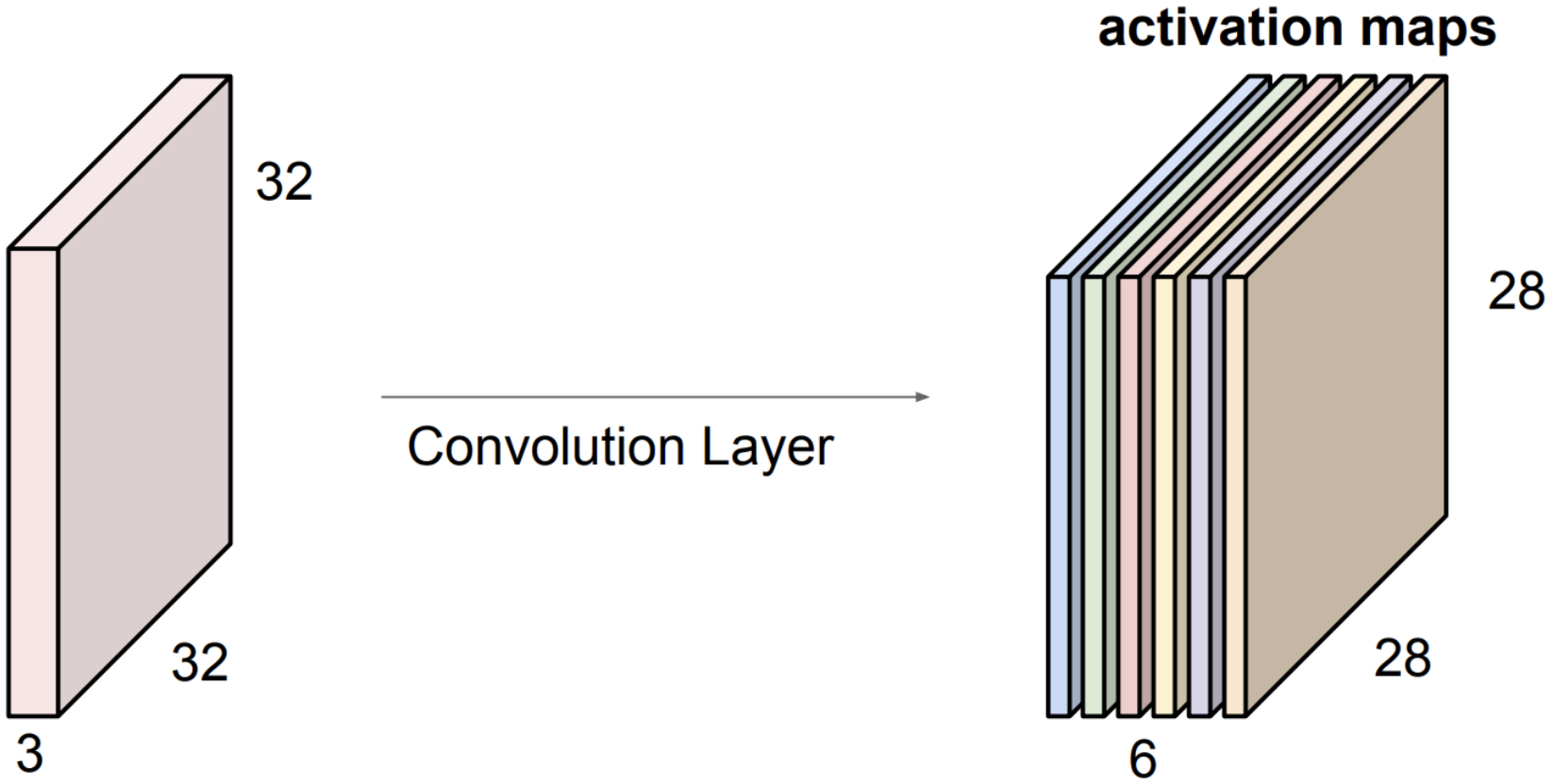
Convolutional layer



Source: Stanford CS 231 lecture

Convolutional layer

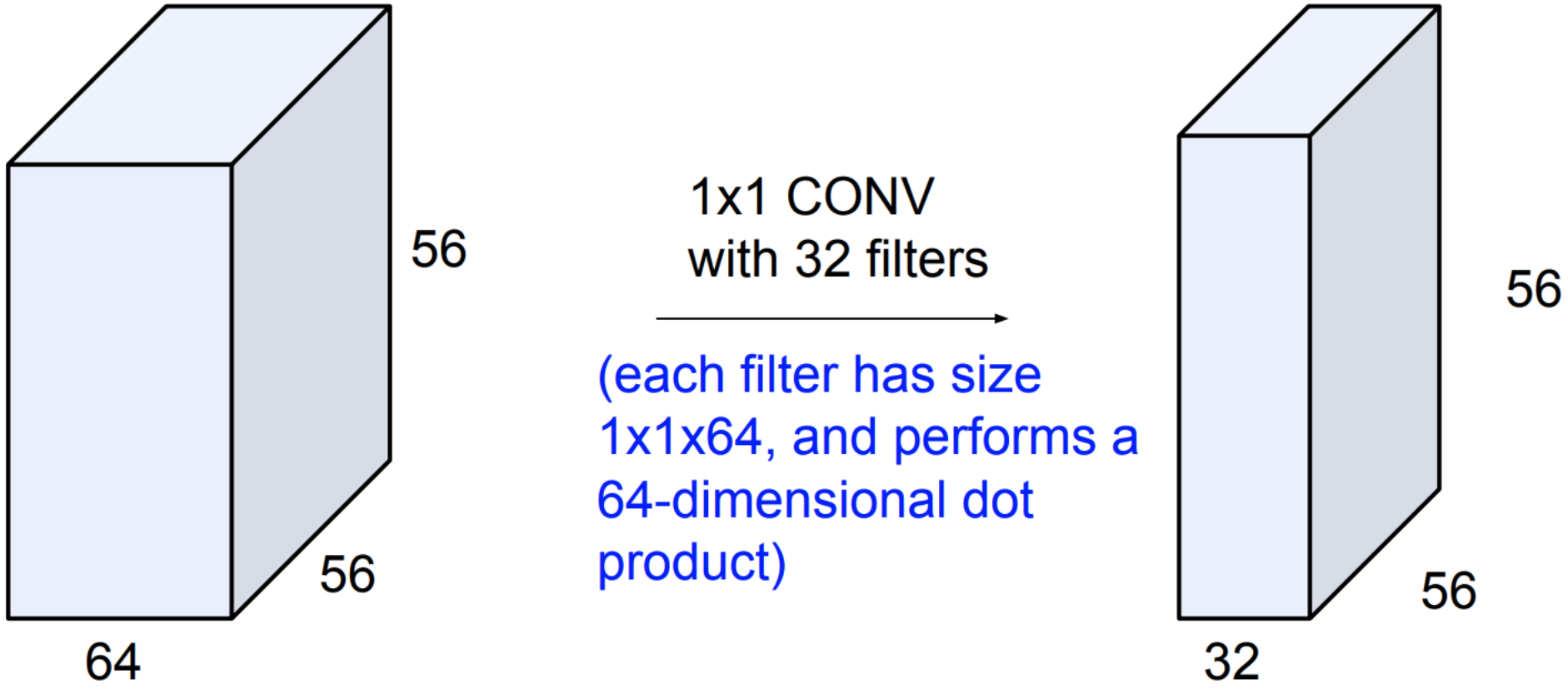
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

Source: Stanford CS 231 lecture

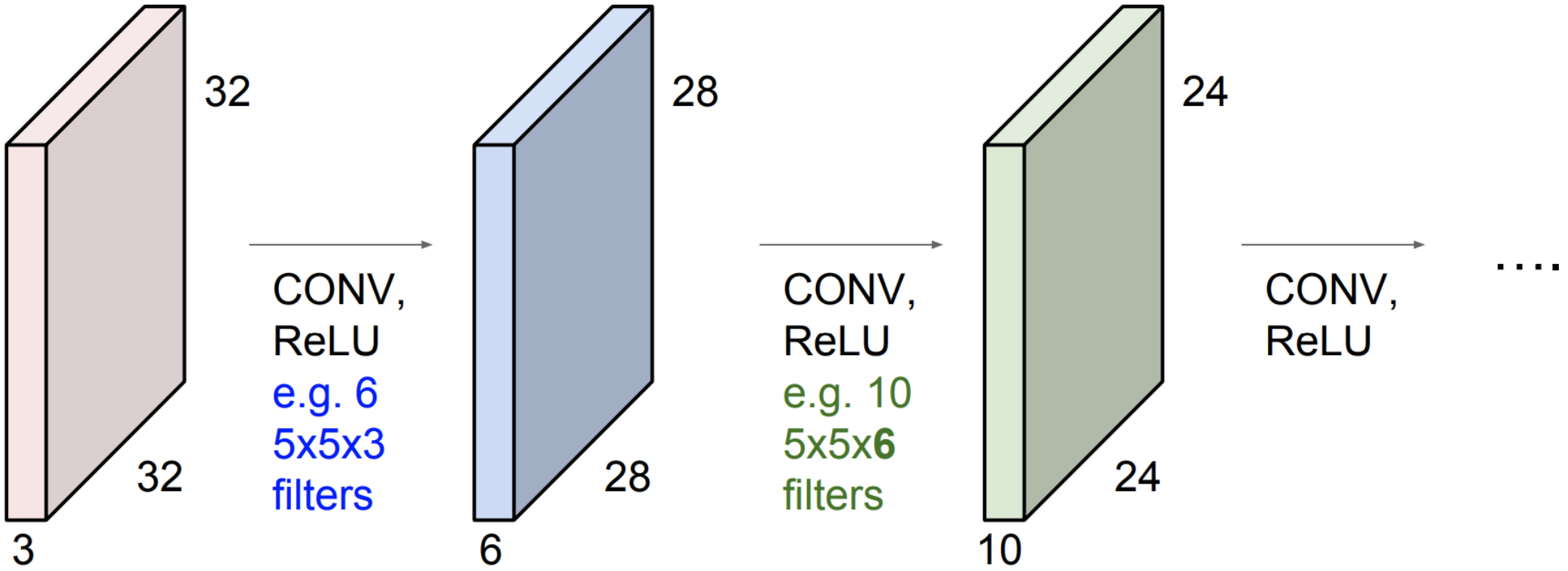
Pointwise (1x1) Convolution



Source: Stanford CS 231 lecture

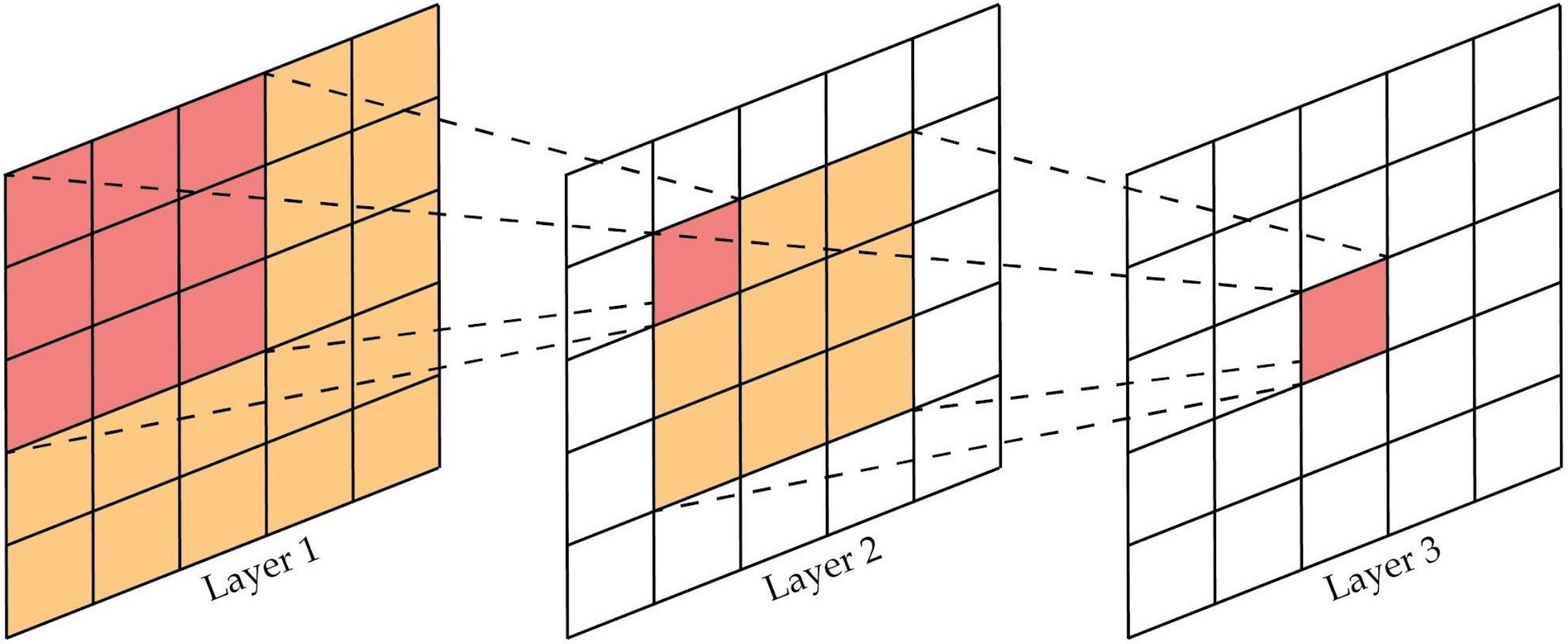
Convolutional layer

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



Source: Stanford CS 231 lecture

Receptive field

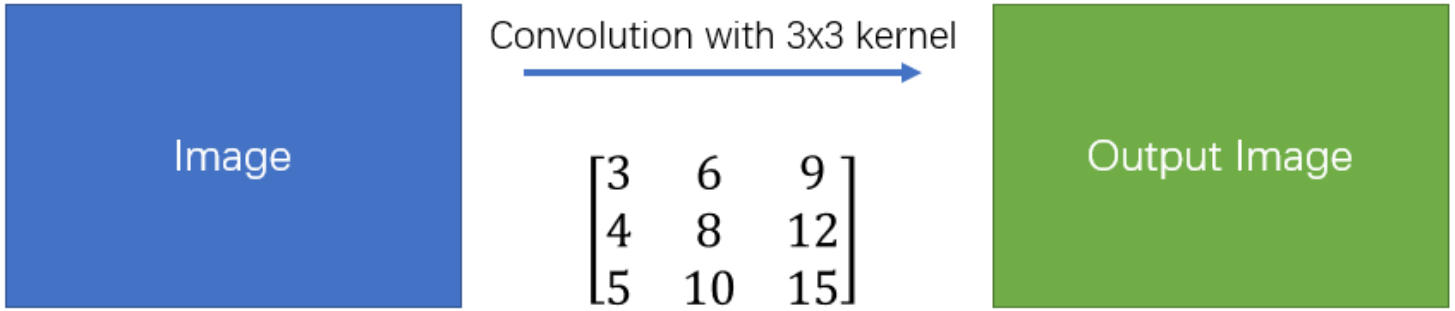


Spatial Separable Convolution

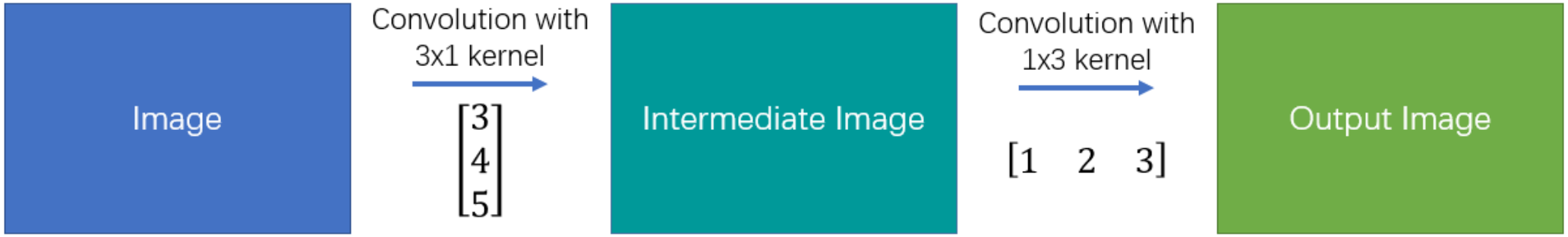
$$\begin{bmatrix} 3 & 6 & 9 \\ 4 & 8 & 12 \\ 5 & 10 & 15 \end{bmatrix} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

Spatial Separable Convolution

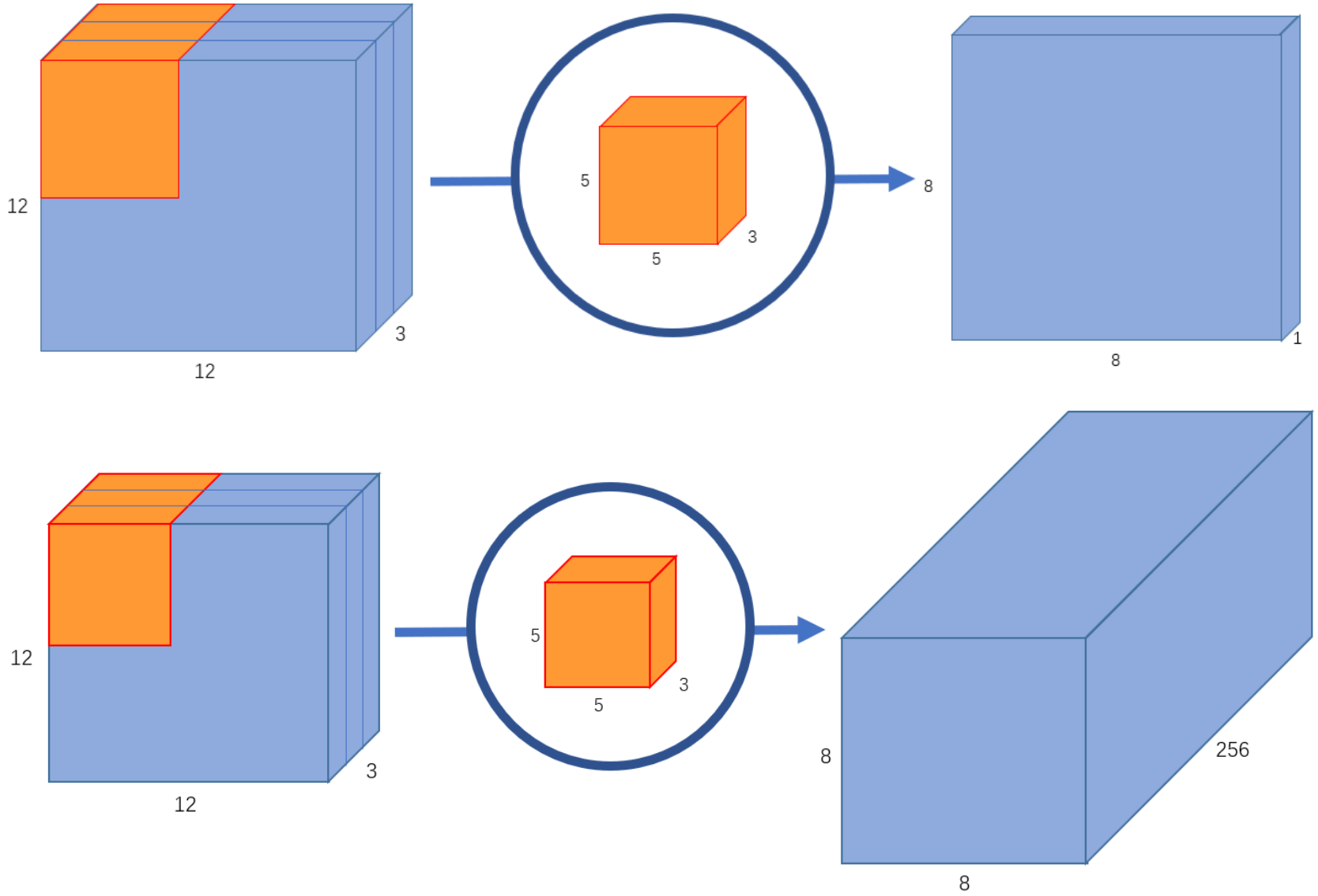
Simple Convolution



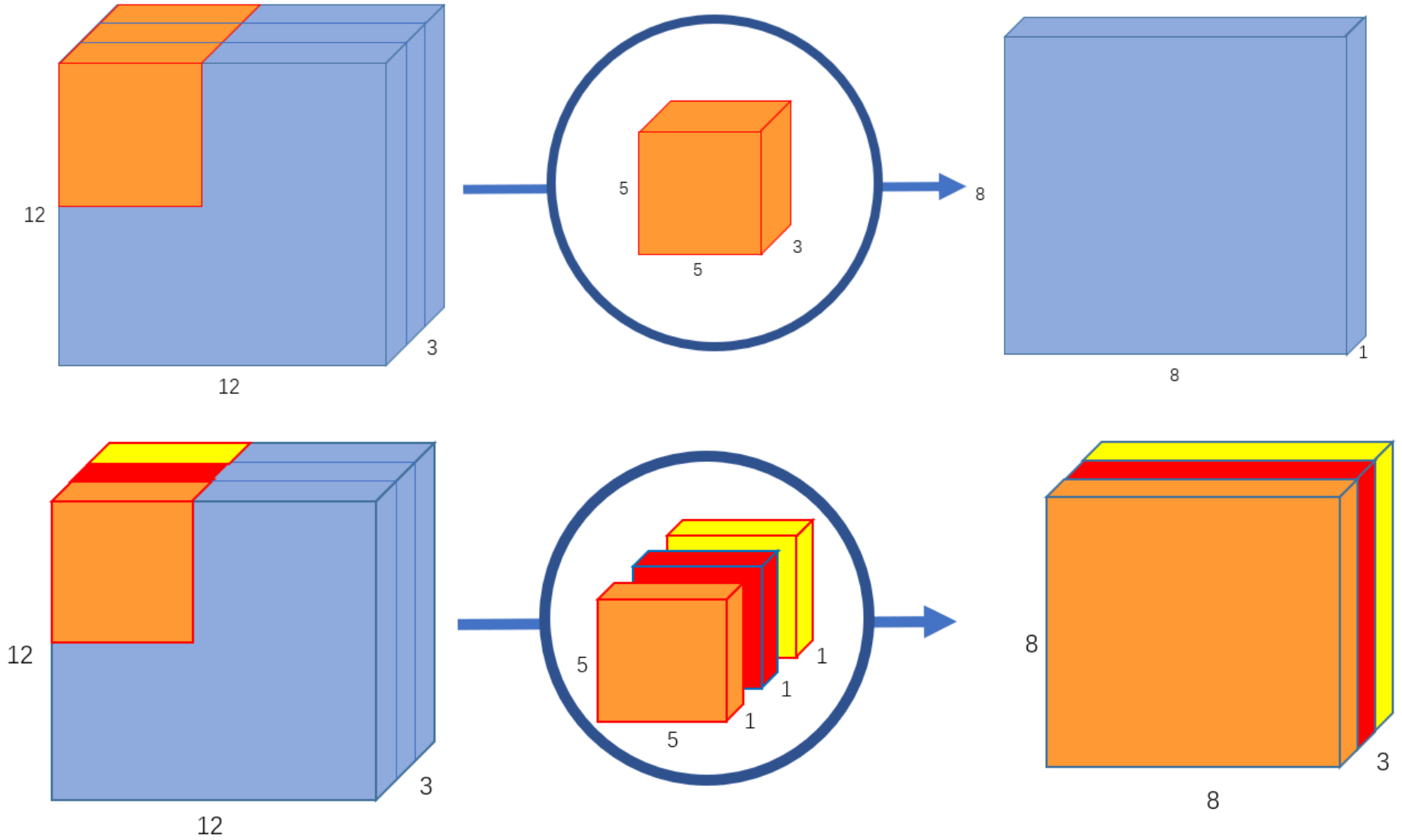
Spatial Separable Convolution



Normal Convolution

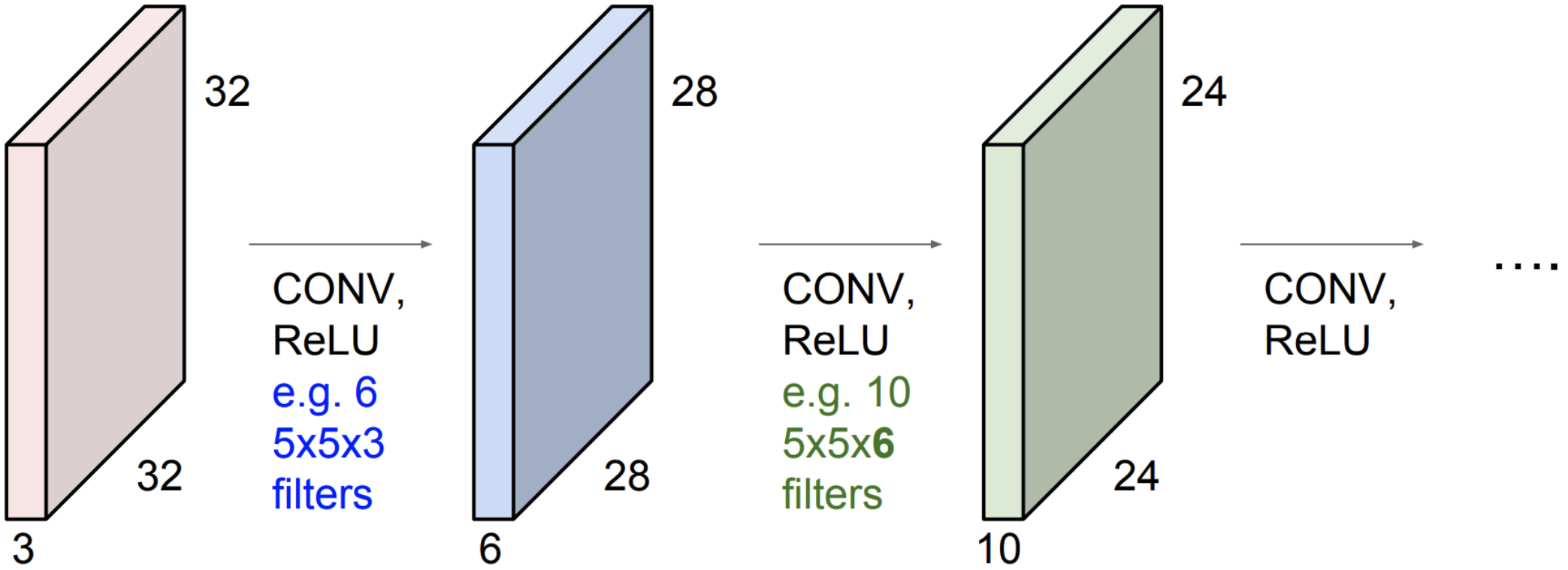


Depthwise Convolution



Convolutional layer

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



Source: Stanford CS 231 lecture

Vanishing gradient

Vanishing gradient is present in **all** deep neural network architectures.

- Due to chain rule / choice of nonlinearity function, gradient can become vanishingly small during backpropagation
- Lower levels are hard to train and are trained slower
- Solutions
  - skip-connections (ResNet)

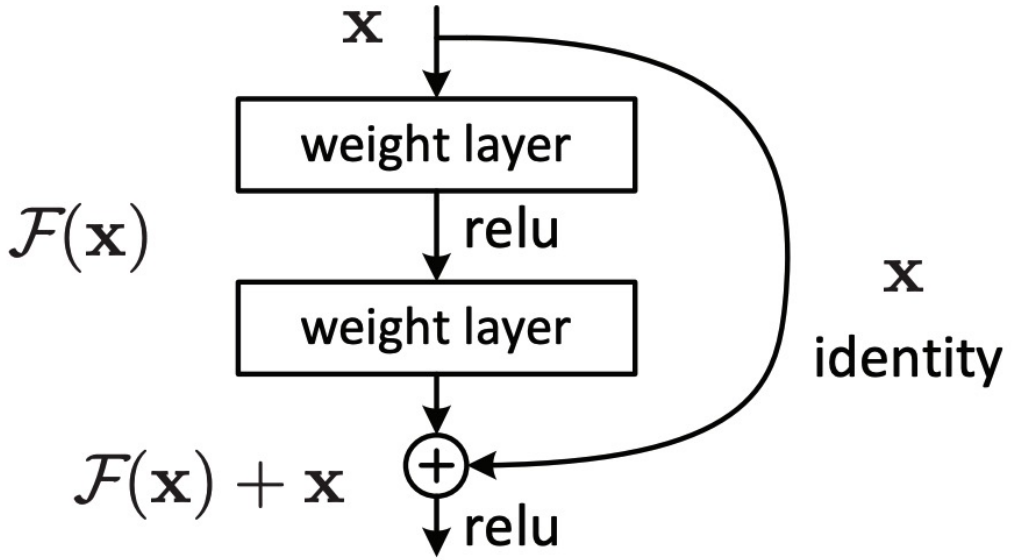
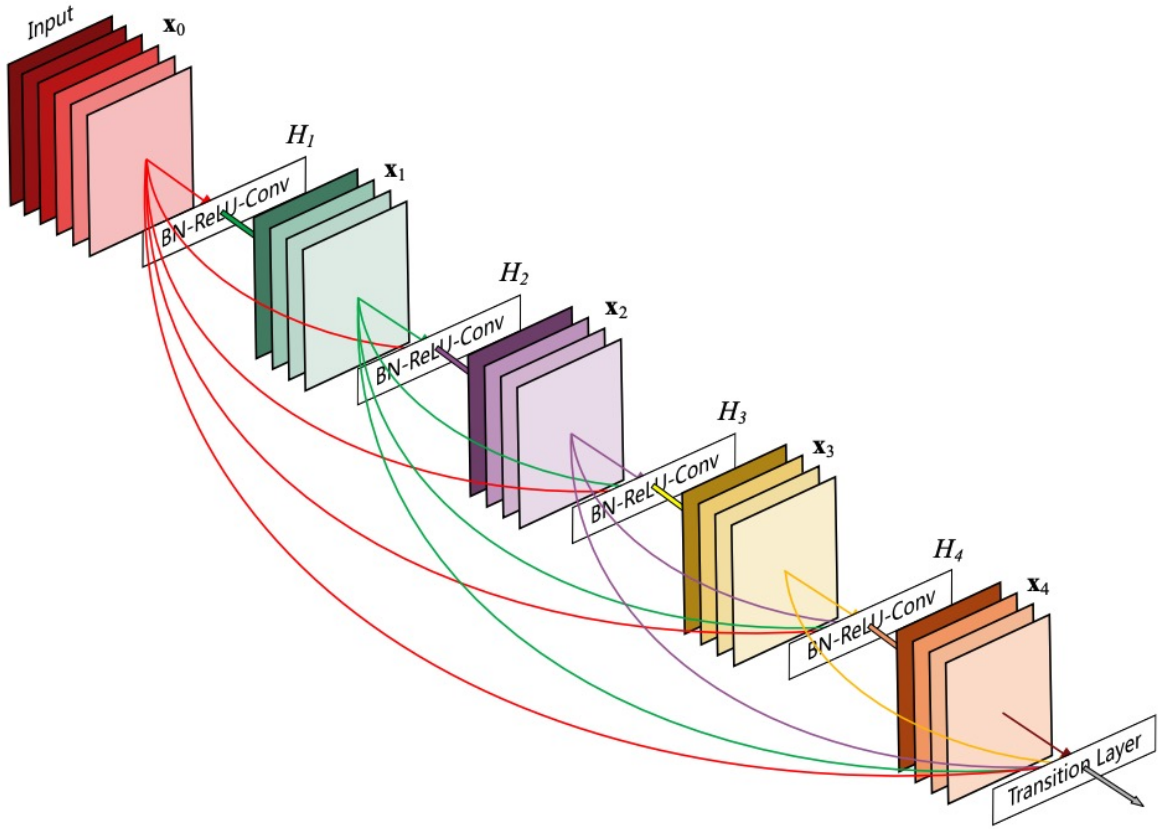


Figure 2. Residual learning: a building block.

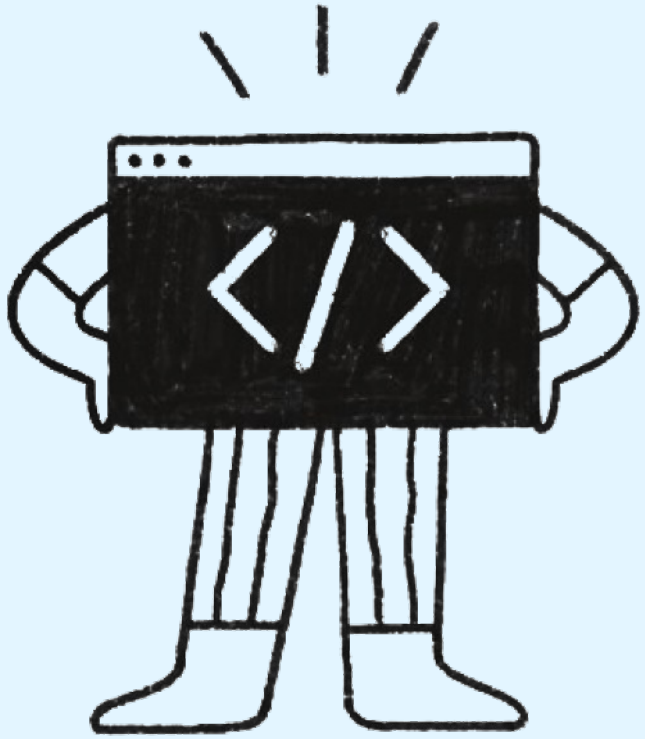
Vanishing gradient

Vanishing gradient is present in all deep neural network architectures.

- Due to chain rule / choice of nonlinearity function, gradient can become vanishingly small during backpropagation
- Lower levels are hard to train and are trained slower
- Solutions
  - skip-connections (ResNet)
  - dense connections (DenseNet)



Classification architectures overview

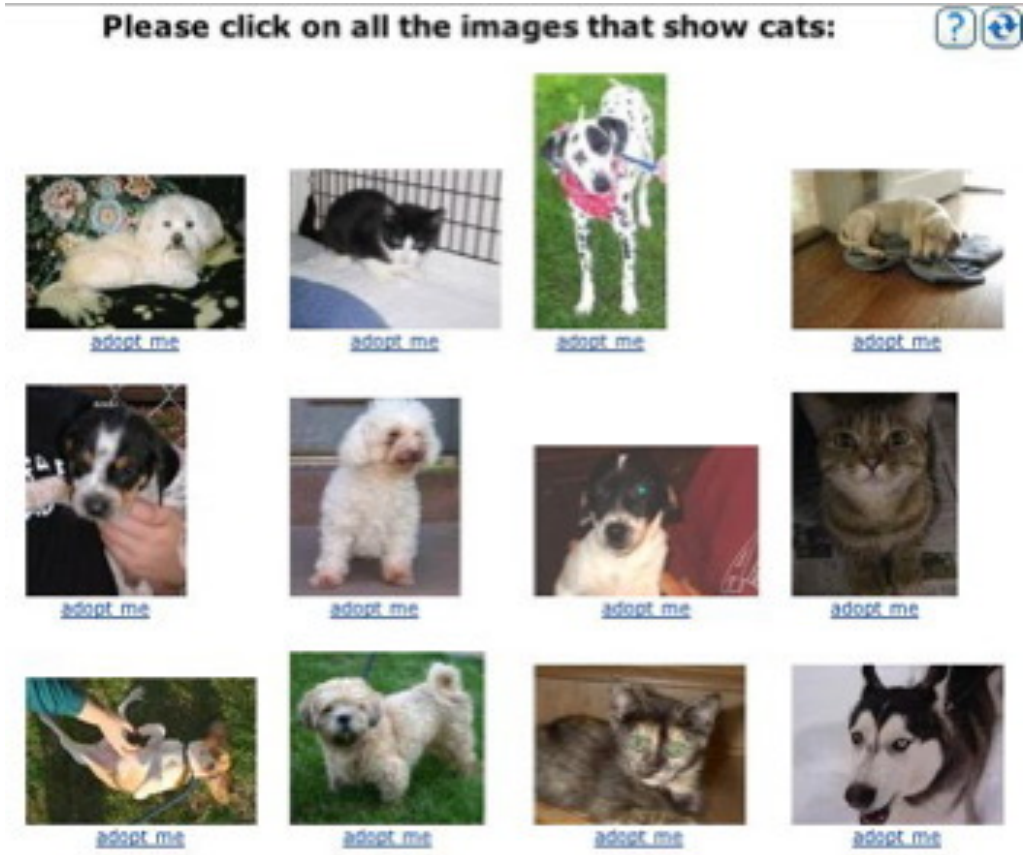


03

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



Elson et al

Image classification now

2014: Asirra on kaggle

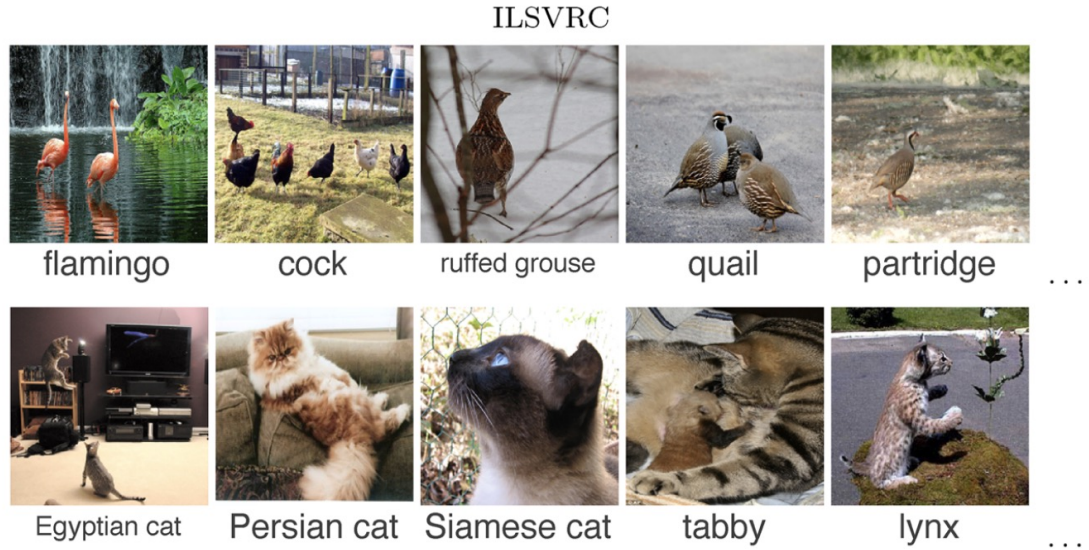
[Competition page]

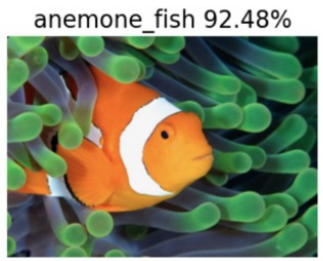
#	△	Team	Members		Score	Entries	Last	Solution
1	—	Pierre Sermanet			0.98914	5	10y	
2	▲ 4	orchid			0.98308	17	10y	
3	—	Owen			0.98171	15	10y	
4	—	Paul Covington			0.98171	3	10y	
5	▼ 3	Maxim Milakov			0.98137	24	10y	
6	▼ 1	we've been in KAIST			0.98102	8	10y	
7	▲ 1	Doug Koch			0.98057	6	10y	
8	▲ 2	fastml.com/cats-and-dogs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

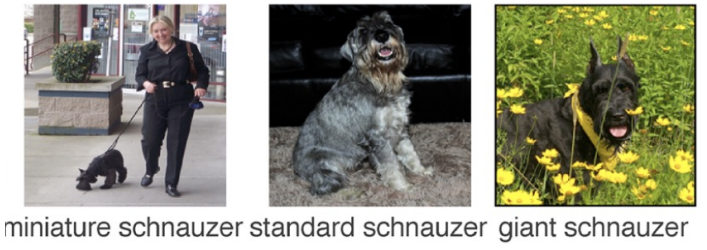
Last Submission: 02/02/2014, 1:43:19 GMT+4

IMAGENET competition

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



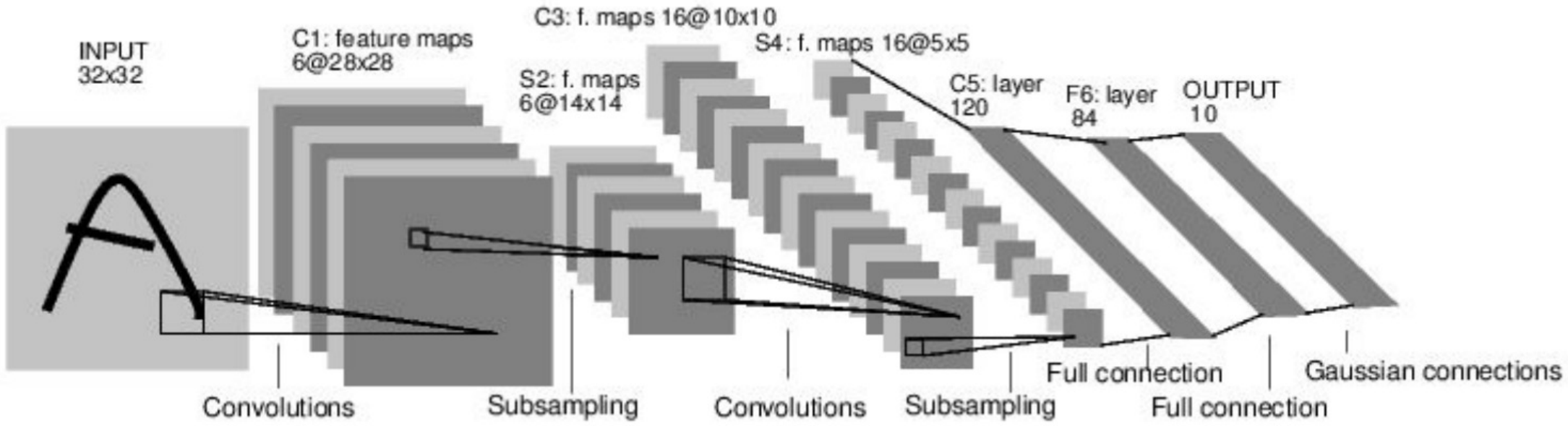




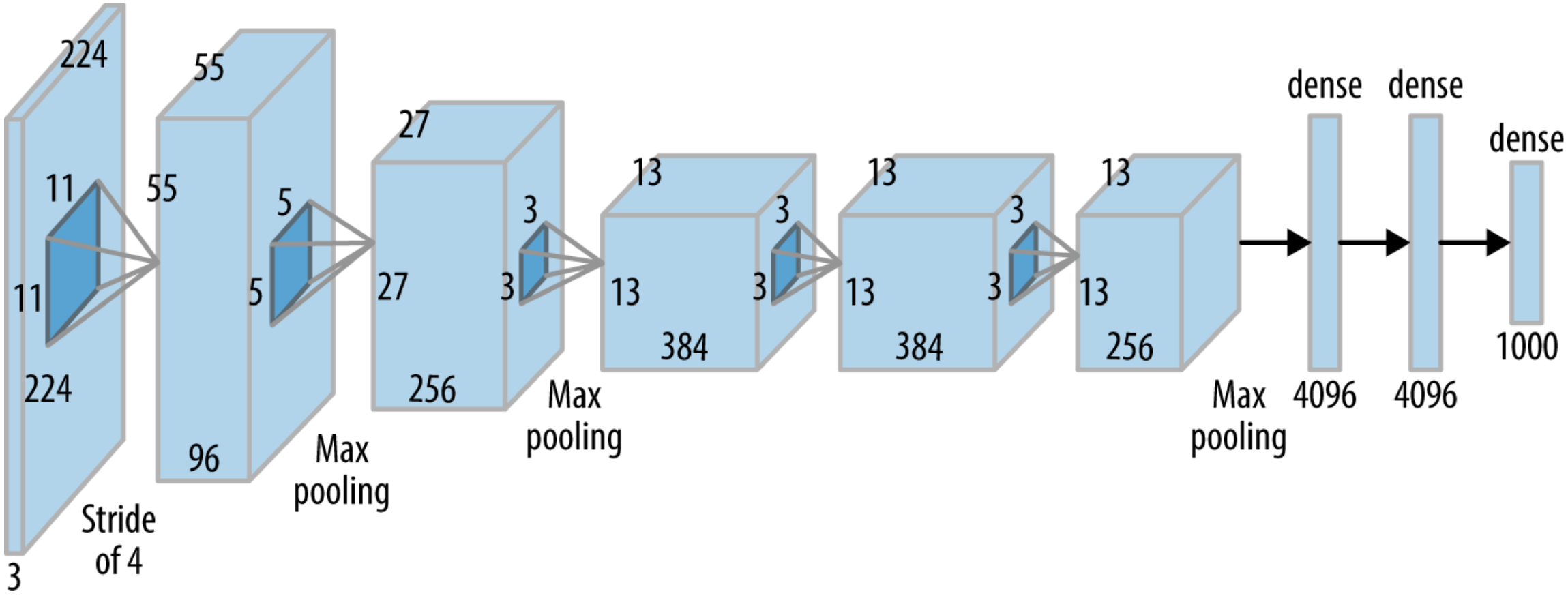
IMAGENET dataset

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

LeNet-5


INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections

AlexNet

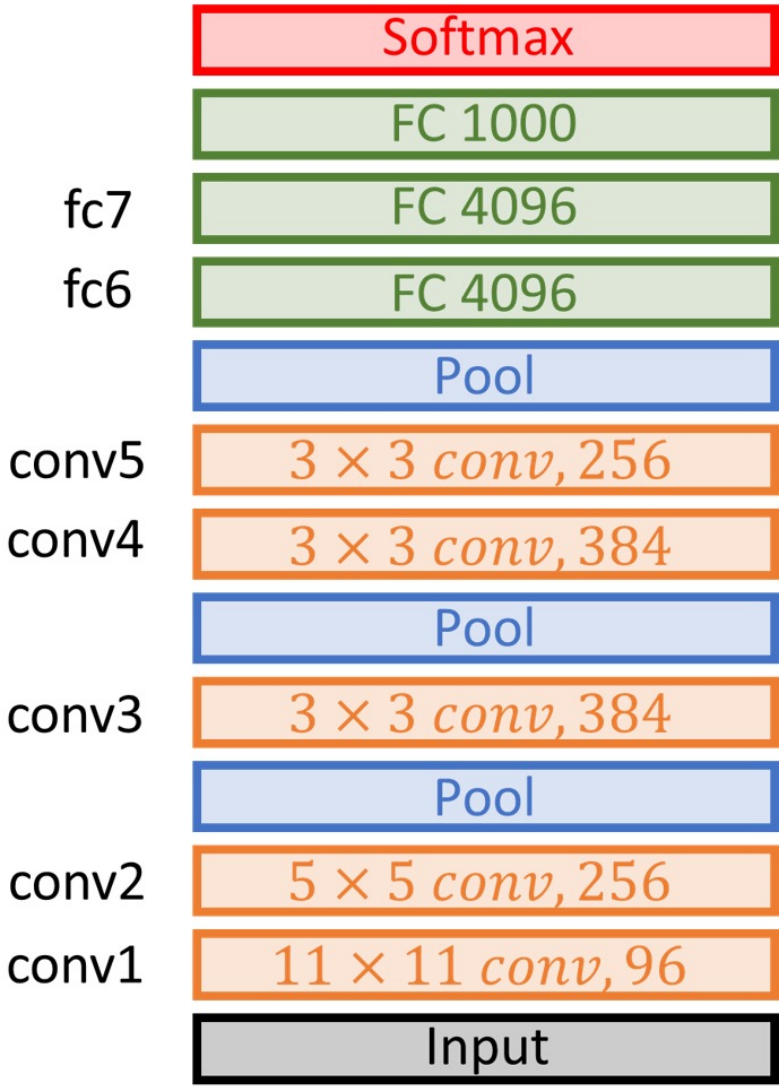

224
224
3
11
11
Stride
of 4
55
55
96
5
5
Max
pooling
27
27
256
3
3
Max
pooling
13
13
384
3
3
13
13
384
3
3
13
13
256
Max
pooling
dense
4096
dense
4096
dense
1000

AlexNet

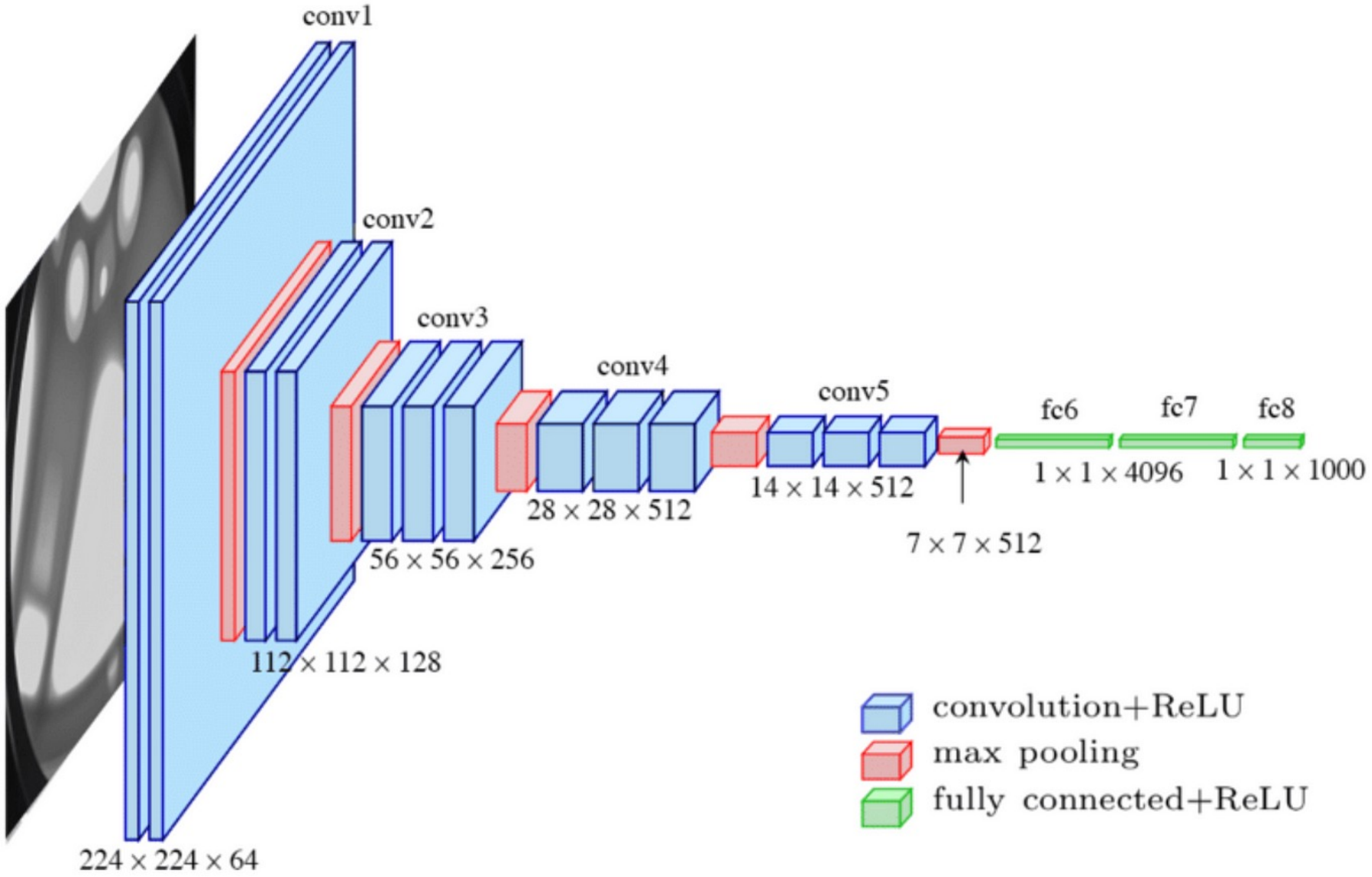
- Krizhevsky et al. 2012
- Used 2 GPUs for training
- First win of ImageNet competition with Deep Neural Networks
- Used big convolutions
- Fully connected at the end
- 7 layers

Takeouts:

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

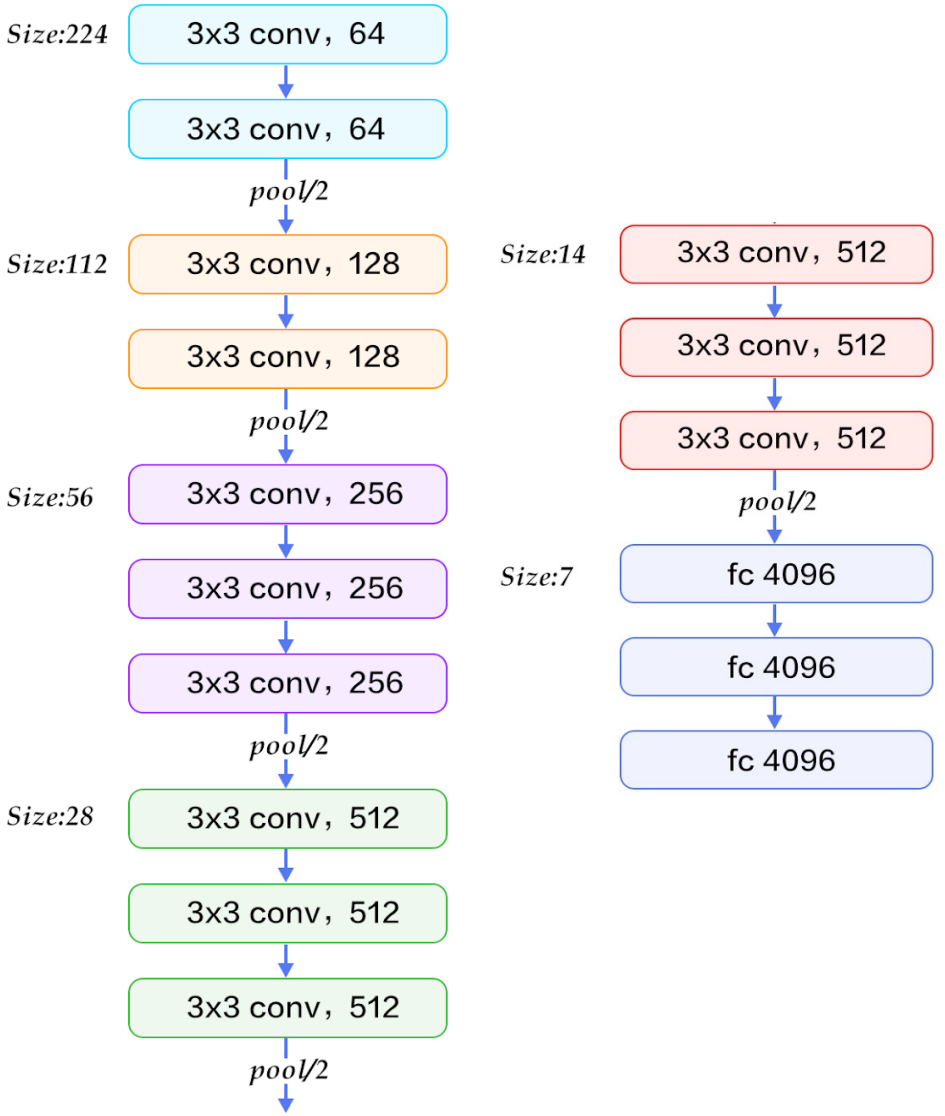


VGGNet

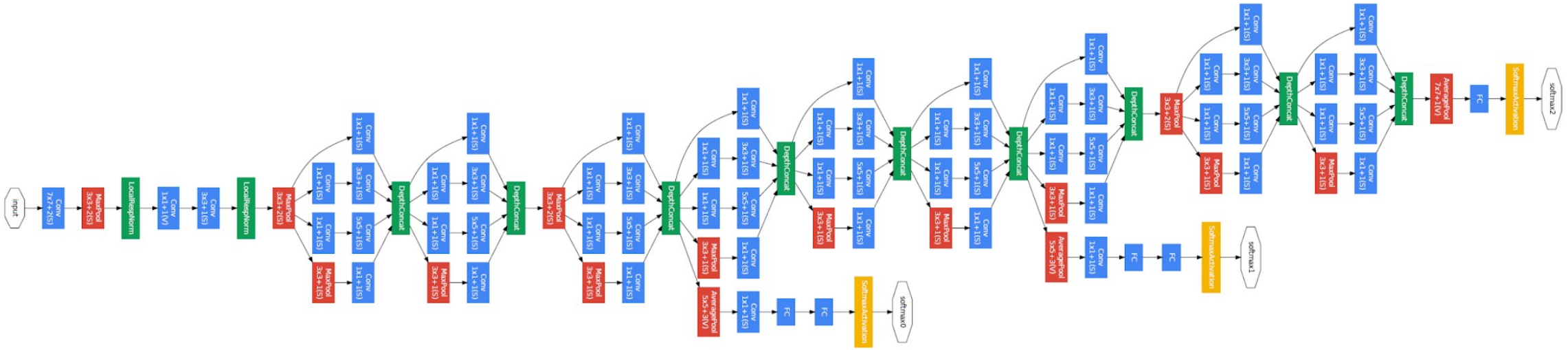

conv1
conv2
conv3
conv4
conv5
fc6
fc7
fc8
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU

VGGNet

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



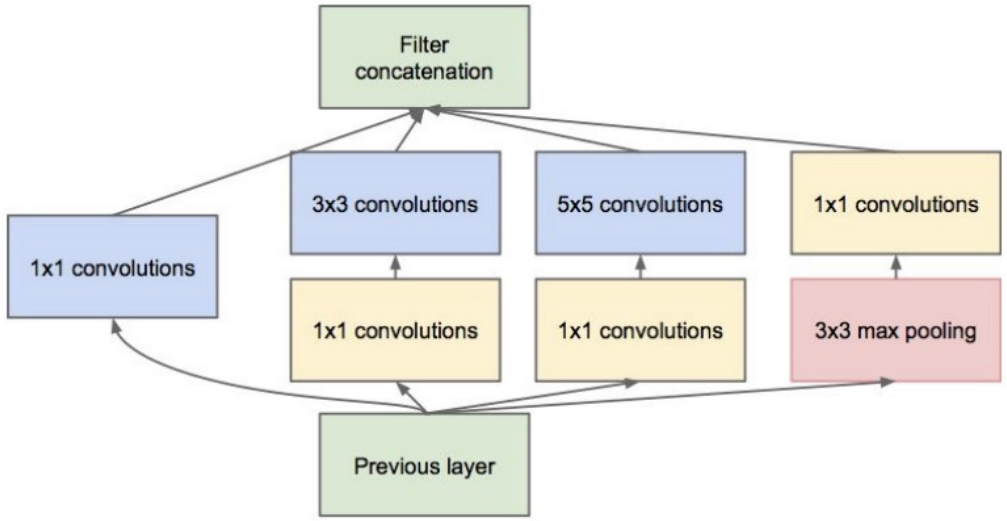
GoogLeNet



Inception module

[Szegedy et al., 2014]

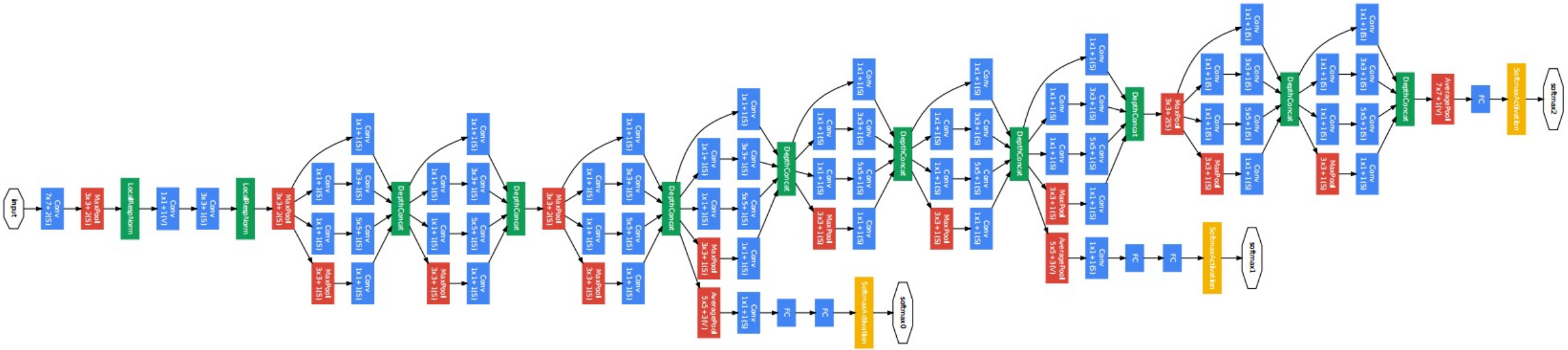
GoogLeNet

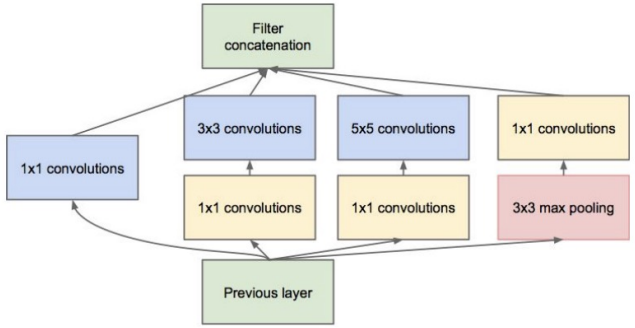


Inception module

ILSVRC 2014 winner (6.7% top 5 error)

GoogLeNet



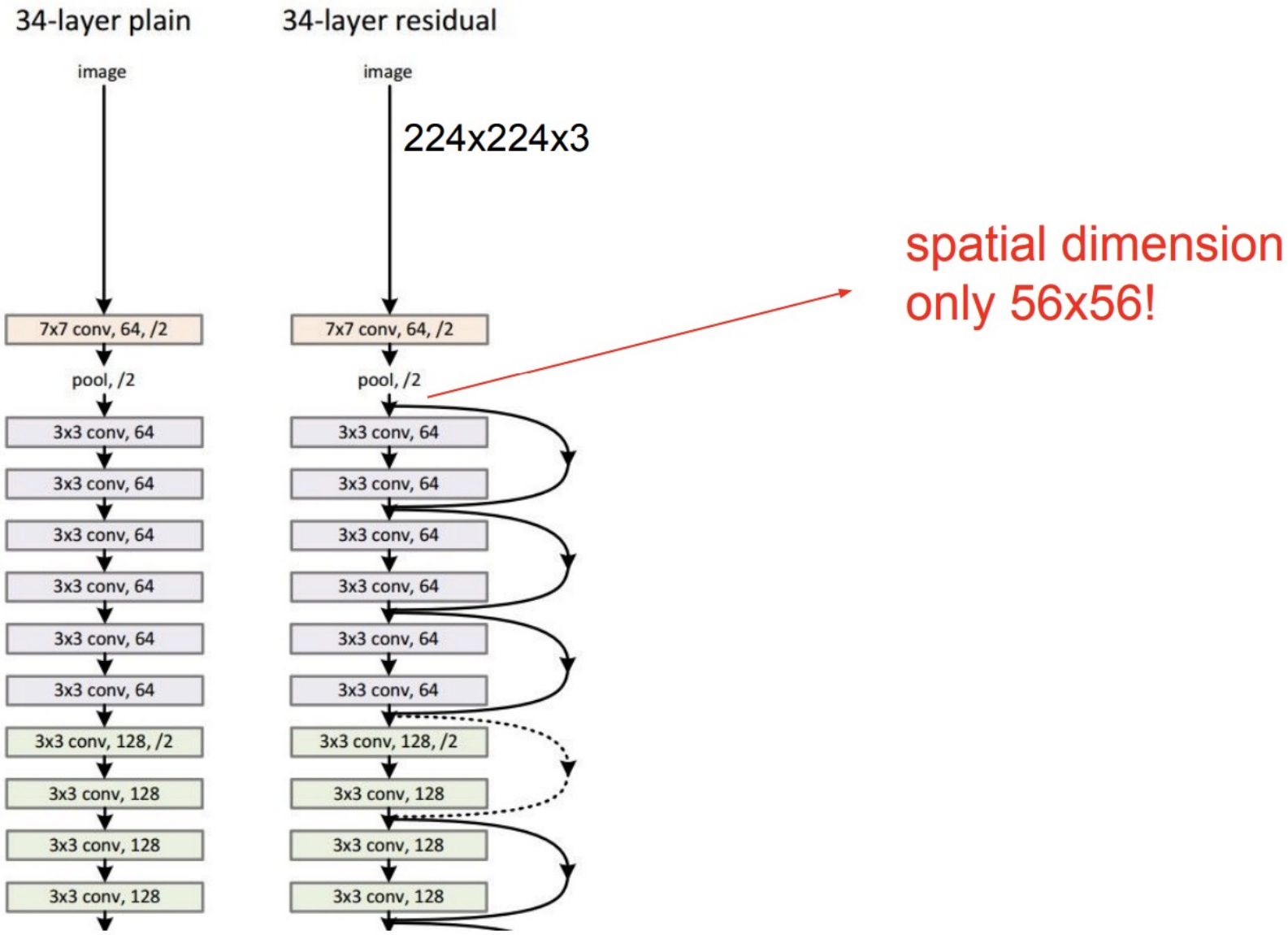


Inception module

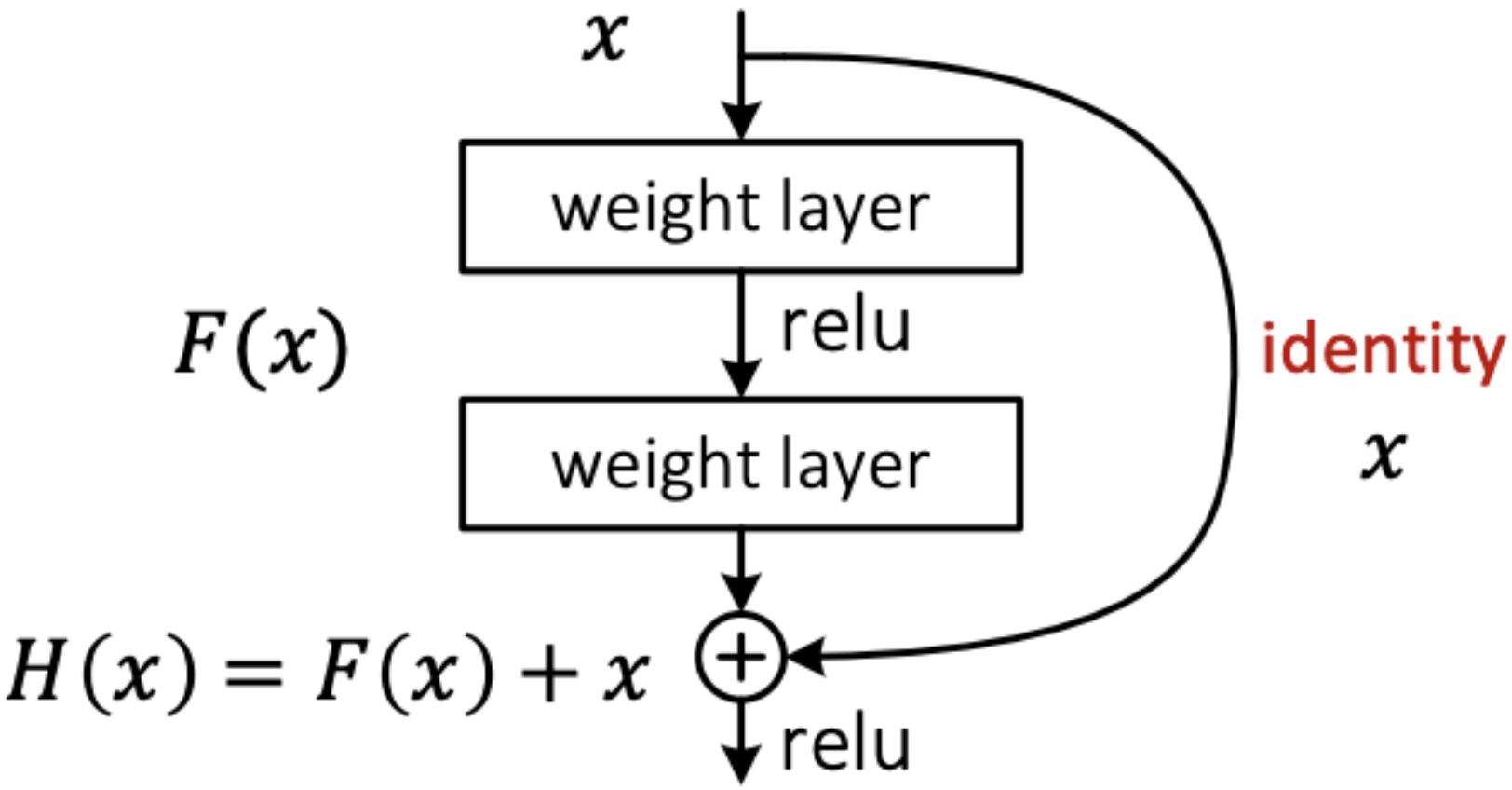
ILSVRC 2014 winner (6.7% top 5 error)

[Szegedy et al., 2014]

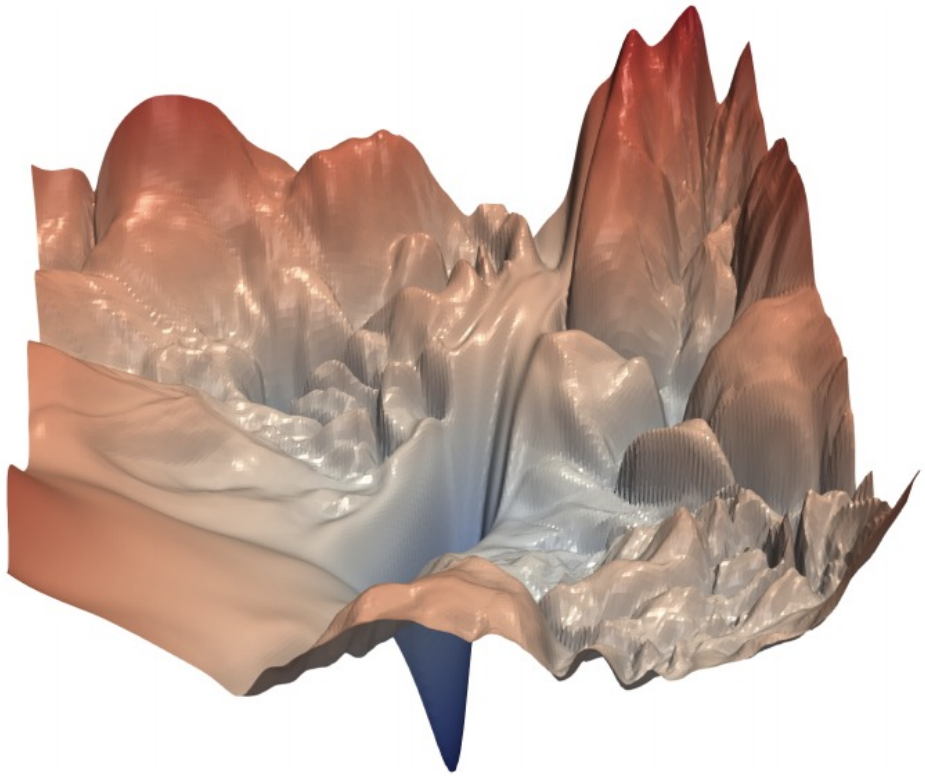
ResNet


34-layer plain
34-layer residual
image
224x224x3
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
spatial dimension
only 56x56!

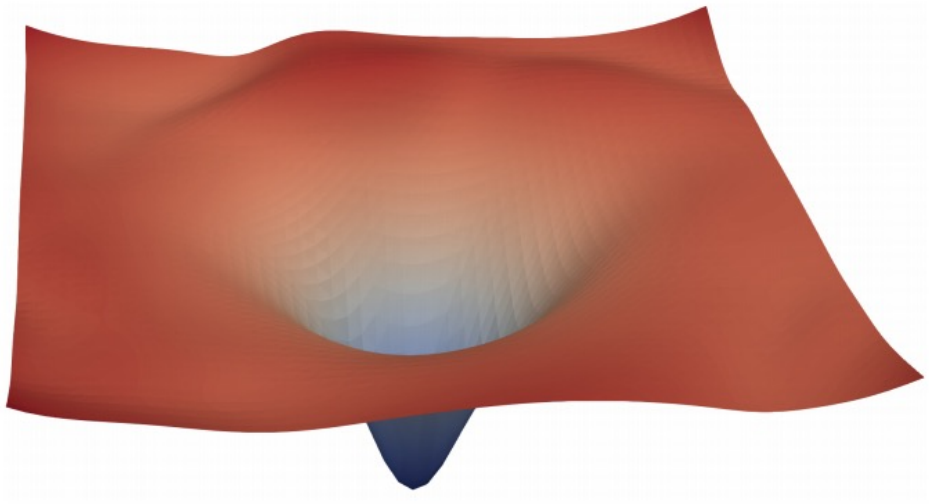
Residual Block



Residual Block

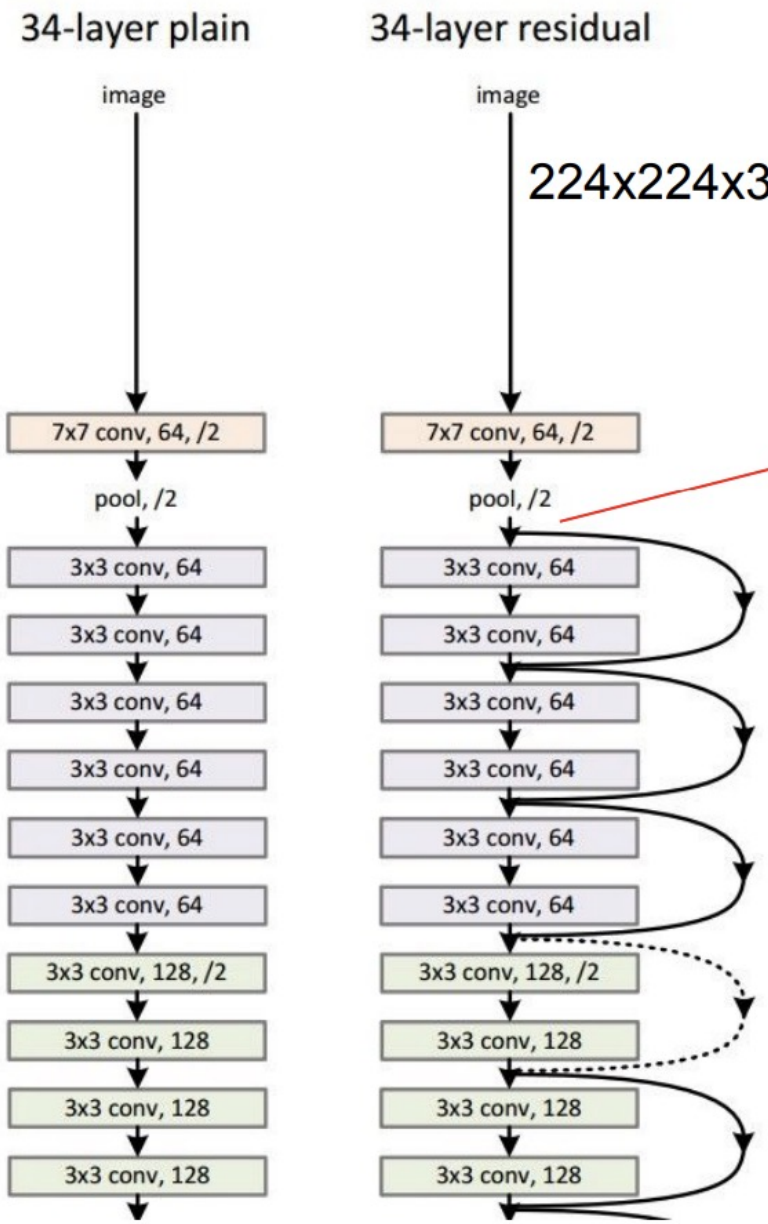


(a) without skip connections



(b) with skip connections

Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.



spatial dimension
only 56x56!

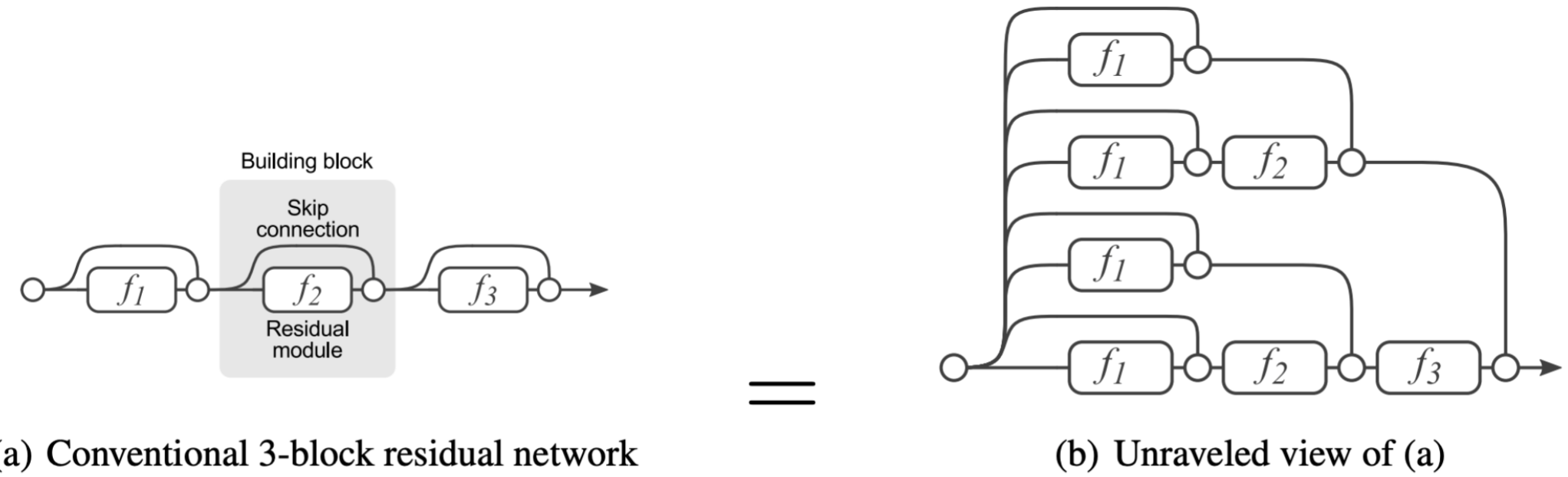
- Batch Normalization after every conv layer
- No dropout used

ImageNet 2015 winner

ILSVRC 2015 winner (3.6% top 5 error)

ResNet is implicit ensemble

Residual Networks Behave Like Ensembles of Relatively Shallow Networks



(a) Conventional 3-block residual network

(b) Unraveled view of (a)

Paper: https://arxiv.org/pdf/1605.06431.pdf

ResNeXt

Same as ResNet, but scalable

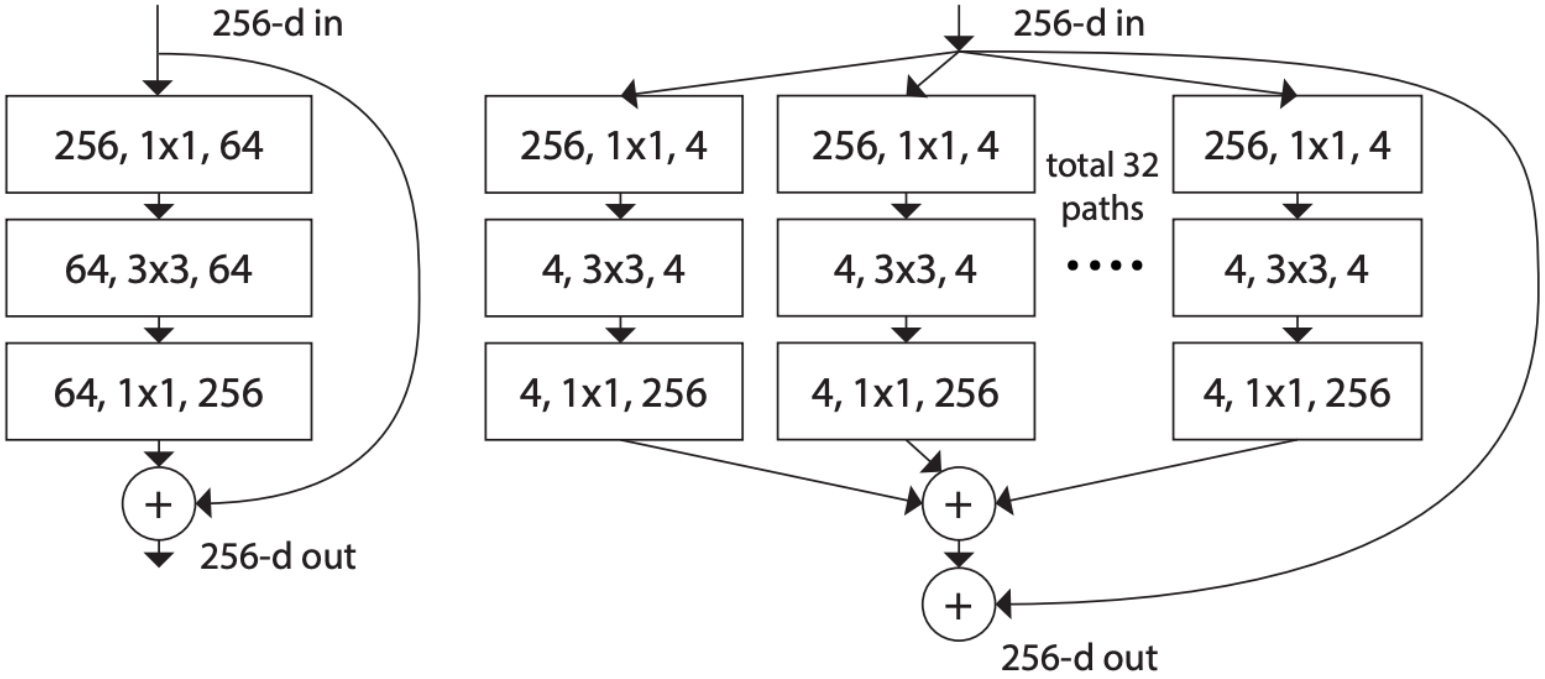


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

SENet

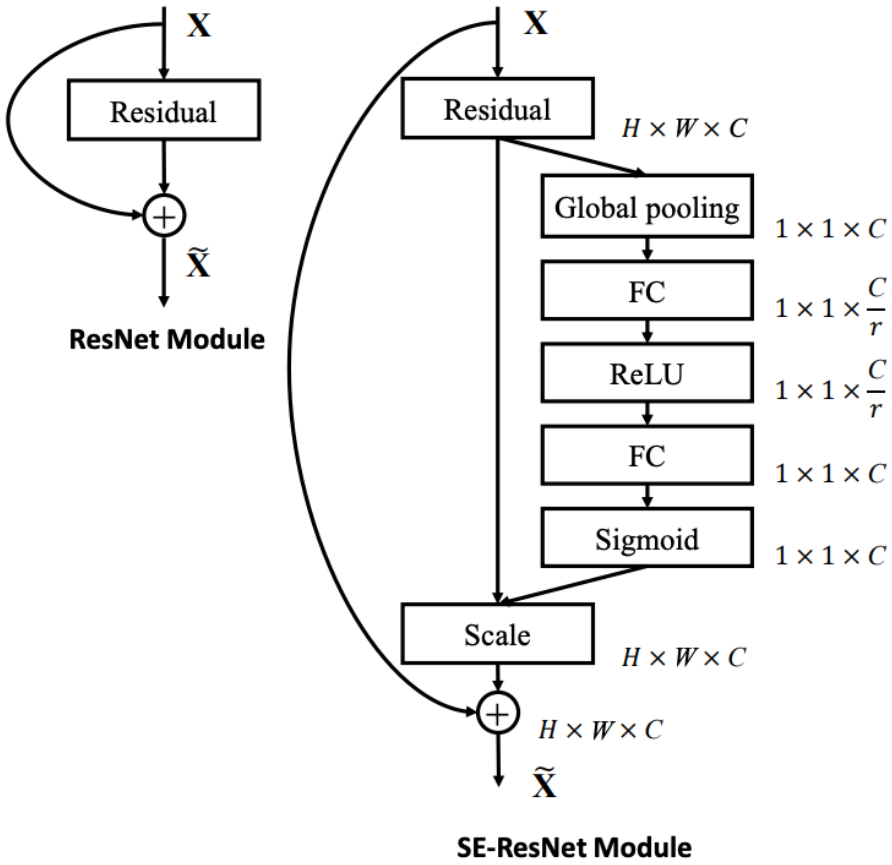
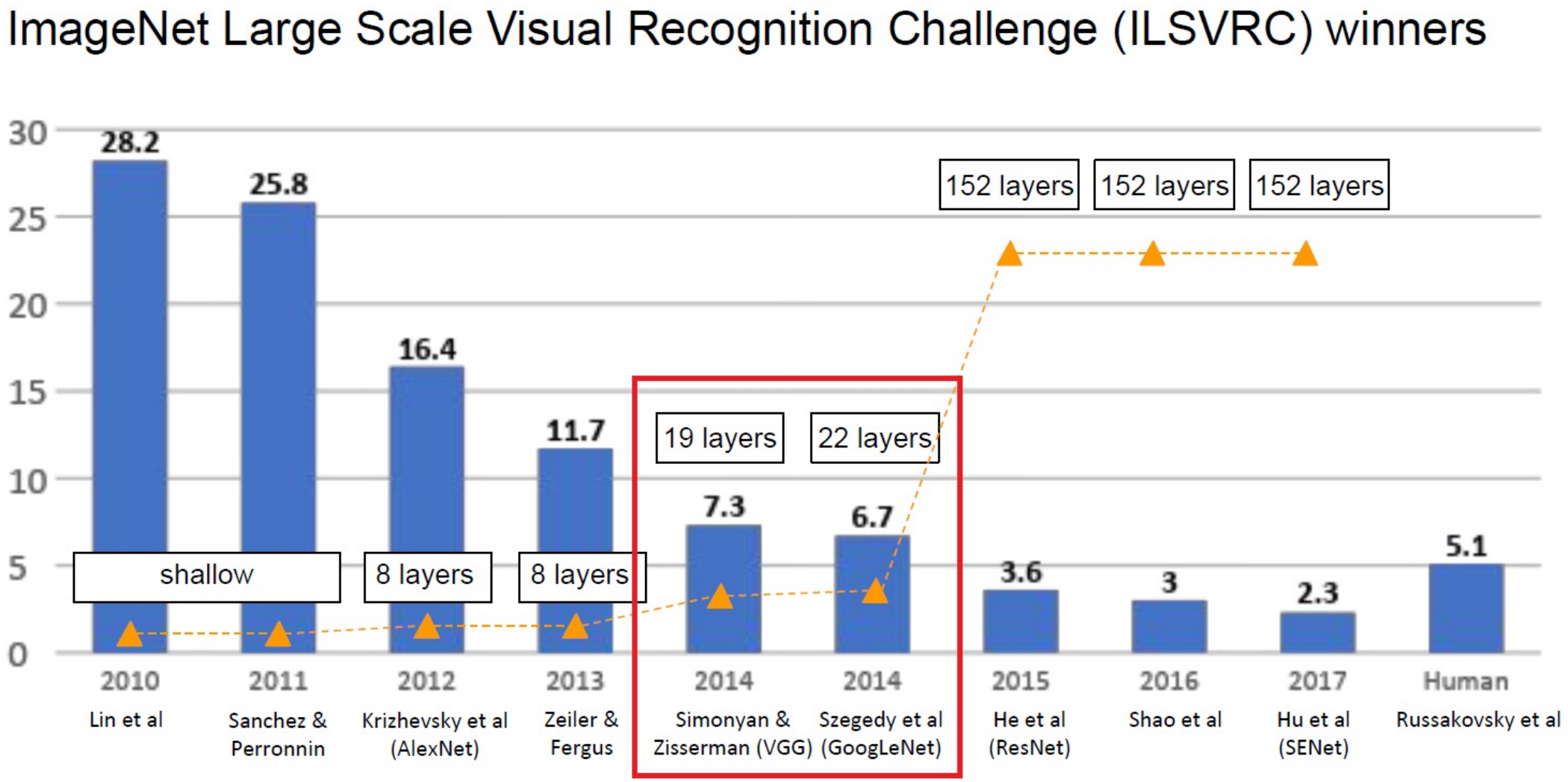


Fig. 3. The schema of the original Residual module (left) and the SE-ResNet module (right).

ImageNet summary



Канал стажировок Яндекса

Спасибо за внимание

Доп. материалы

Y&Y