Deep Learning

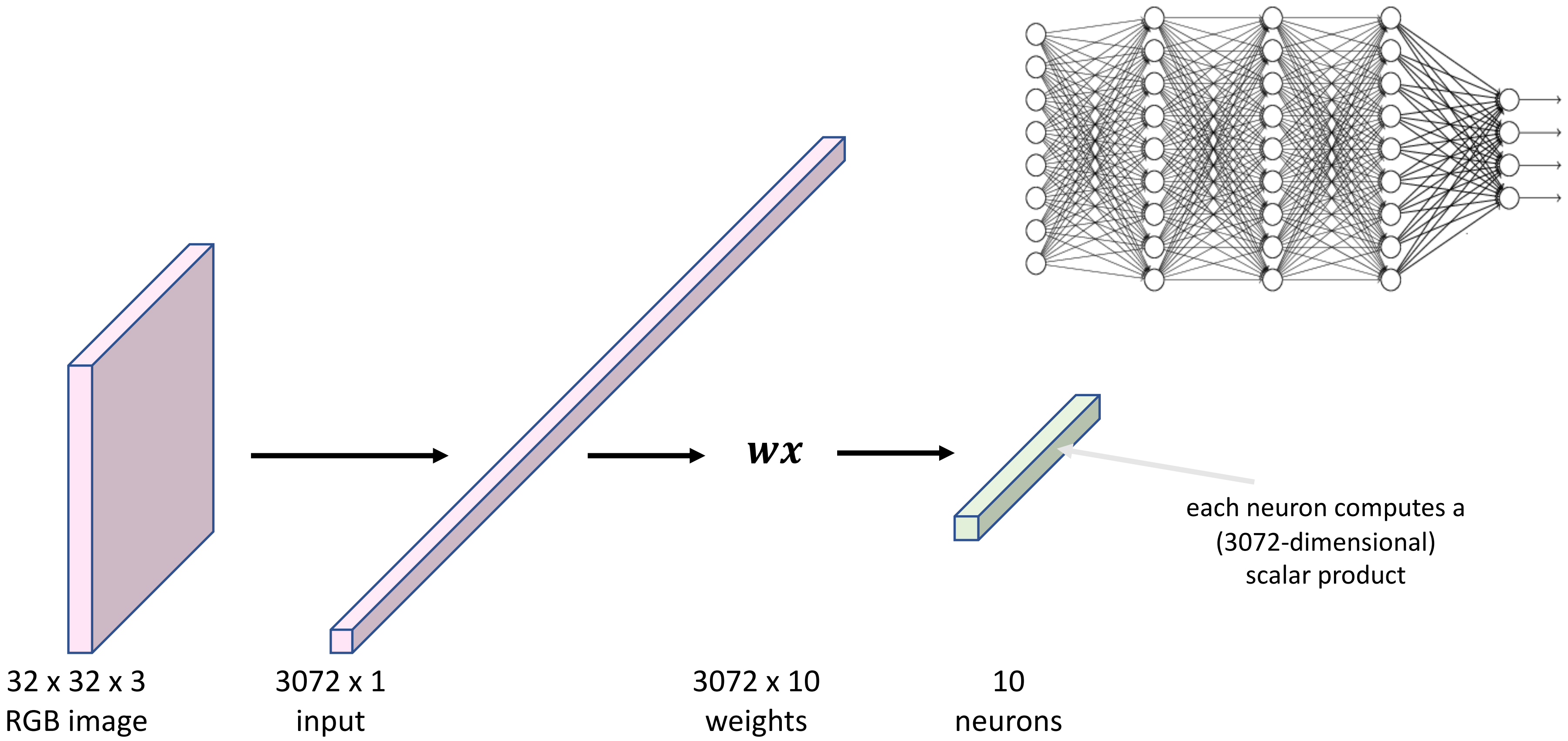
Convolutional Neural Networks

Technische Hochschule Rosenheim
Sommer 2023
Prof. Dr. Jochen Schmidt

Overview

- Linear Filtering & Convolution
- Convolutional Neural Networks
- Architectures

Example: Typical MLP-Structure for Images

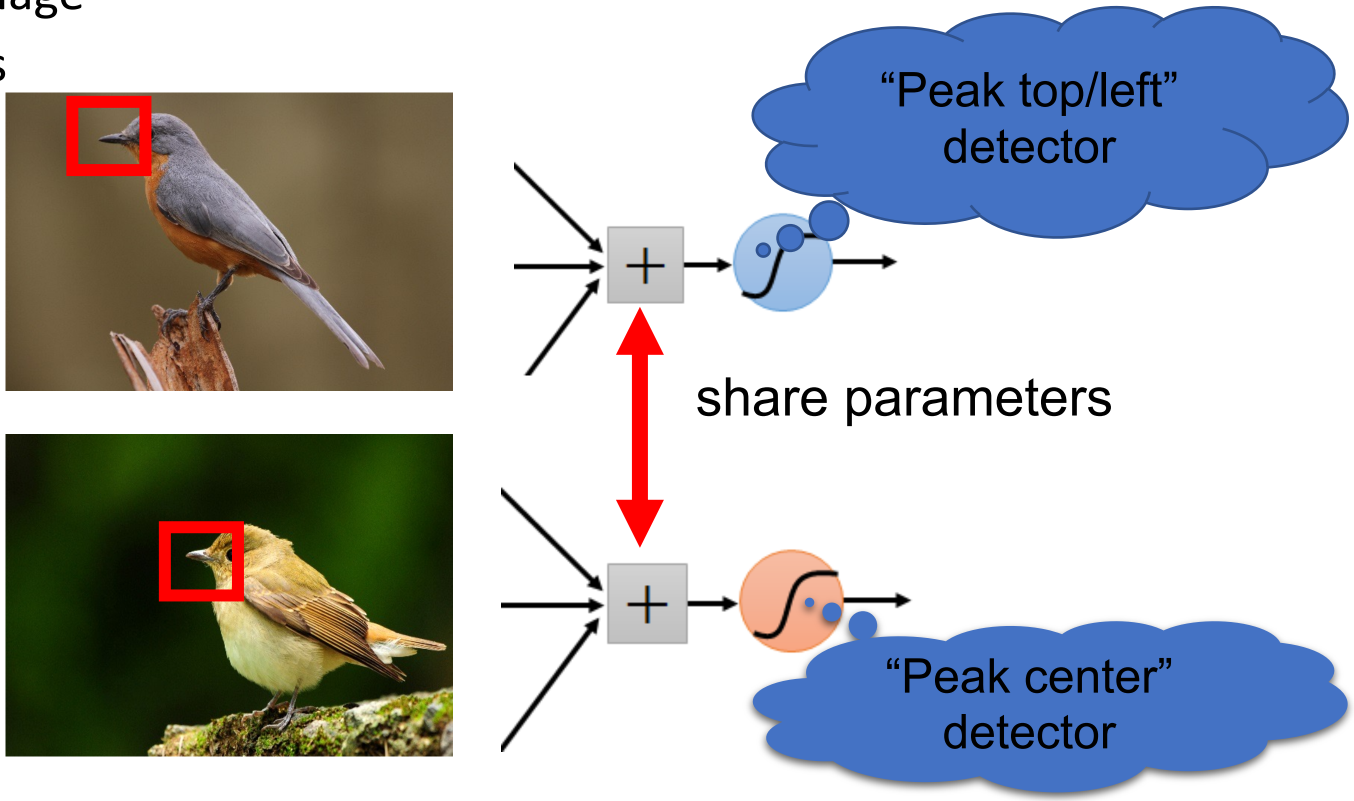


Recognition of Image Parts

- many patterns are smaller than the complete image
- for small regions: less parameters required

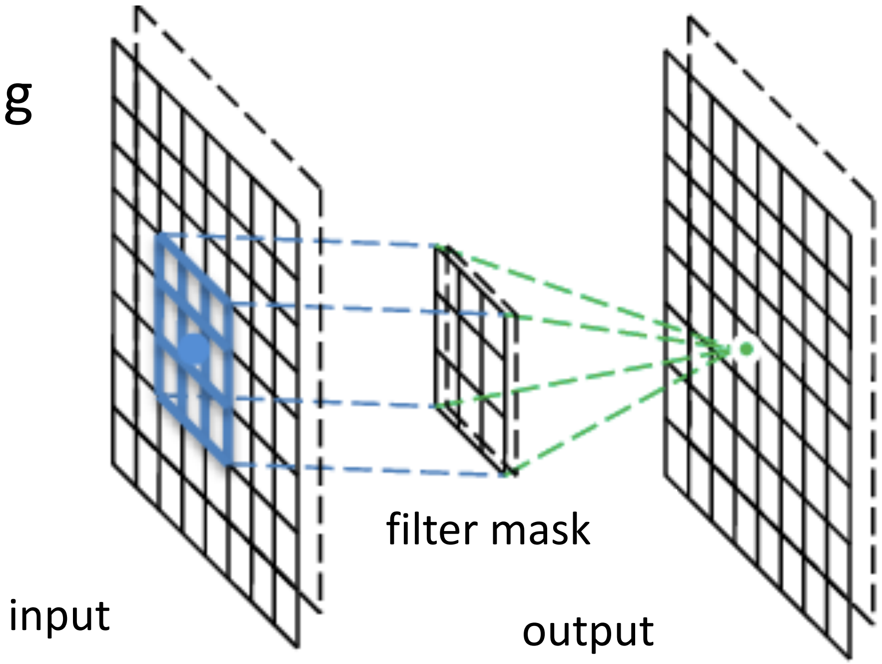
Similar Patterns

- similar patterns can be found in different image locations
- Idea: Train many small detectors that
  - move over the image
  - share parameters



Convolutional Layer

- hence: CNN – Convolutional Neuronal Network
- consists of (linear) convolution filters
- the filter masks are learned during training
- first used with backpropagation in LeNet (1989-1998): LeCun, Bottou, Bengio, Haffner. Gradient-Based Learning Applied to Document Recognition. Proc. of the IEEE 86(11): 2278-2324, 1998.



Linear Filters & Convolution

2D-Convolution (Faltung)

original image f

+ filter mask g

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \xrightarrow{\text{Convolution}}$$

filtered image f'

- mirror filter mask horizontally and vertically,
- move filter mask over image,
- compute weighted sum mask/underlying gray value
  $\longrightarrow$ new value for central pixel

These will be three-dimensional in a CNN!

$$\underbrace{f'(x,y)}_{\text{filtered image}} = f * g = \sum_{i=-r}^{r} \sum_{j=-r}^{r} \underbrace{f(x-i, y-j)}_{\text{original image}}\, \underbrace{g(i,j)}_{\text{filter mask}}$$

Filter mask:

- Size $(2r+1) \times (2r+1)$
- (i,j) coordinate system, (0, 0) in mask center,
- i right, j down (as in image)

Low-pass Filter

- removes high frequencies
- reduces image noise
- results in smoothing of image

3x3 Mean $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

3x3 Gaussian $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

Low-pass Filter

5x5 Mean $\frac{1}{25}\begin{pmatrix}1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\\ 1 & 1 & 1 & 1 & 1\end{pmatrix}$

5x5 Gaussian (with $\sigma = 1$) $\frac{1}{273}\begin{pmatrix}1 & 4 & 7 & 4 & 1\\ 4 & 16 & 26 & 16 & 4\\ 7 & 26 & 41 & 26 & 7\\ 4 & 16 & 26 & 16 & 4\\ 1 & 4 & 7 & 4 & 1\end{pmatrix}$

Influence of Filter Size

Mean filter, Sizes: 3x3, 5x5, 11x11, 21x21

Original

High-pass Filter

- removes low frequencies
- edge detection
- widely used:
  - Sobel
    - based on computing the first order partial derivatives
    - result: two edge images (horizontal and vertical direction)
    - edges = large values (maxima of derivative)
  - Laplace
    - based on computing the second order derivatives (the Laplace operator)
    - edges = zero crossings
    - more subjective to noise compared to Sobel

Sobel

- first order partial derivatives

$$f_x = \frac{\partial f(x,y)}{\partial x} \qquad \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad f_y = \frac{\partial f(x,y)}{\partial y} \qquad \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Range?

- gradient strength:

$$s = \sqrt{f_x^2 + f_y^2}$$

- gradient direction:

$$\theta = \arctan\left(\frac{f_y}{f_x}\right)$$

Note: use atan2(y, x)

High-pass Filter – Sobel

horizontal

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

vertical

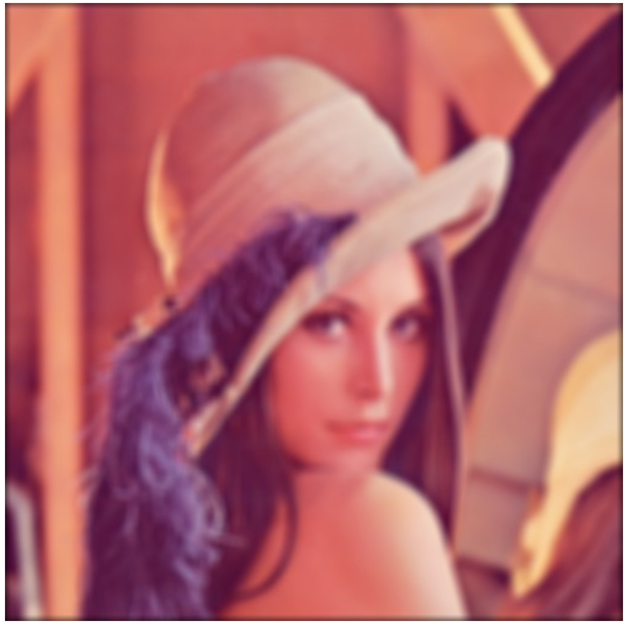
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

combined (gradient strength image)

(images converted to gray-scale range & inverted)

Caution with Color Images!

- Low-pass
  - filter each RGB-channel separately
    - but: RGB is unsuitable for linear interpolation
  - better: use CIELUV or CIELAB color space
    - but: Conversion from RGB is computationally expensive (non-linear)
  - combine all channels using tensors (→ CNN)
    - does not really solve the RGB-problem (linear operation)
    - but results in single value combining all channels (no color image)



Gaussian 11x11, $\sigma = 5$

- High-pass filter
  - filtering channels separately does not really make sense
  - combine all channels using tensors (→ CNN)

Remarks

- Filter masks typically have odd size (3x3, 5x5, …) → symmetric about the current pixel
- they are not necessarily square

- Design of special-purpose filters is possible
- filters can be concatenated, resulting a single new linear filter mask (cf. convolution equation)

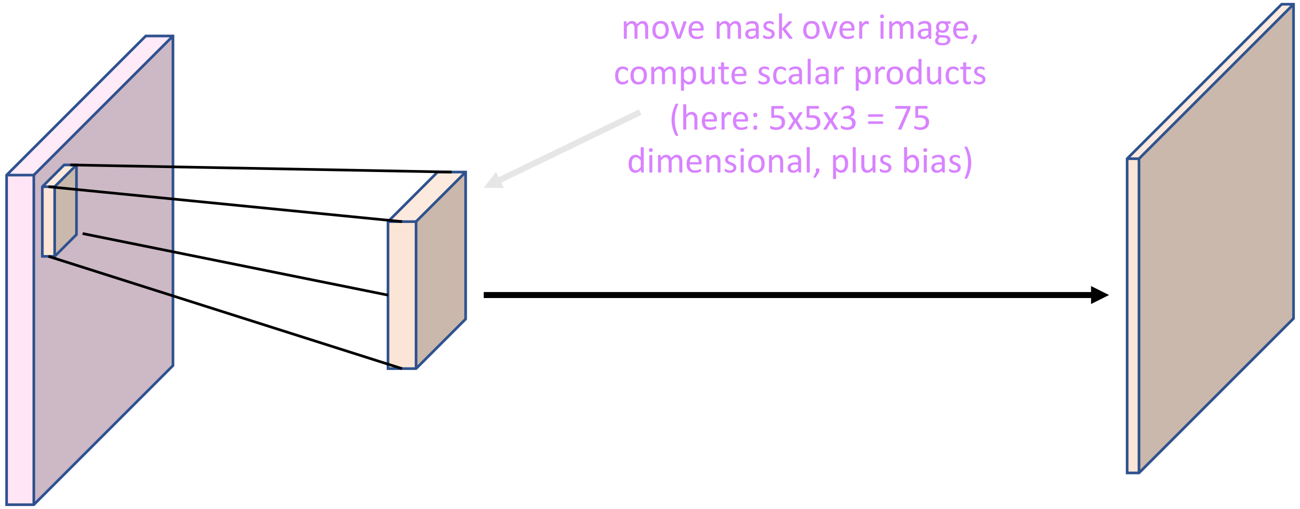
- in image processing, convolution is usually computed in image space
- convolution using FFT makes sense with large filter sizes only

CNN

Convolutional Layer

retains spatial
structure

Convolution:
move mask over image,
compute scalar products
(here: 5x5x3 = 75
dimensional, plus bias)



32 x 32 x 3
RGB image

5 x 5 x 3
Filter

= weights $\boldsymbol{w}$

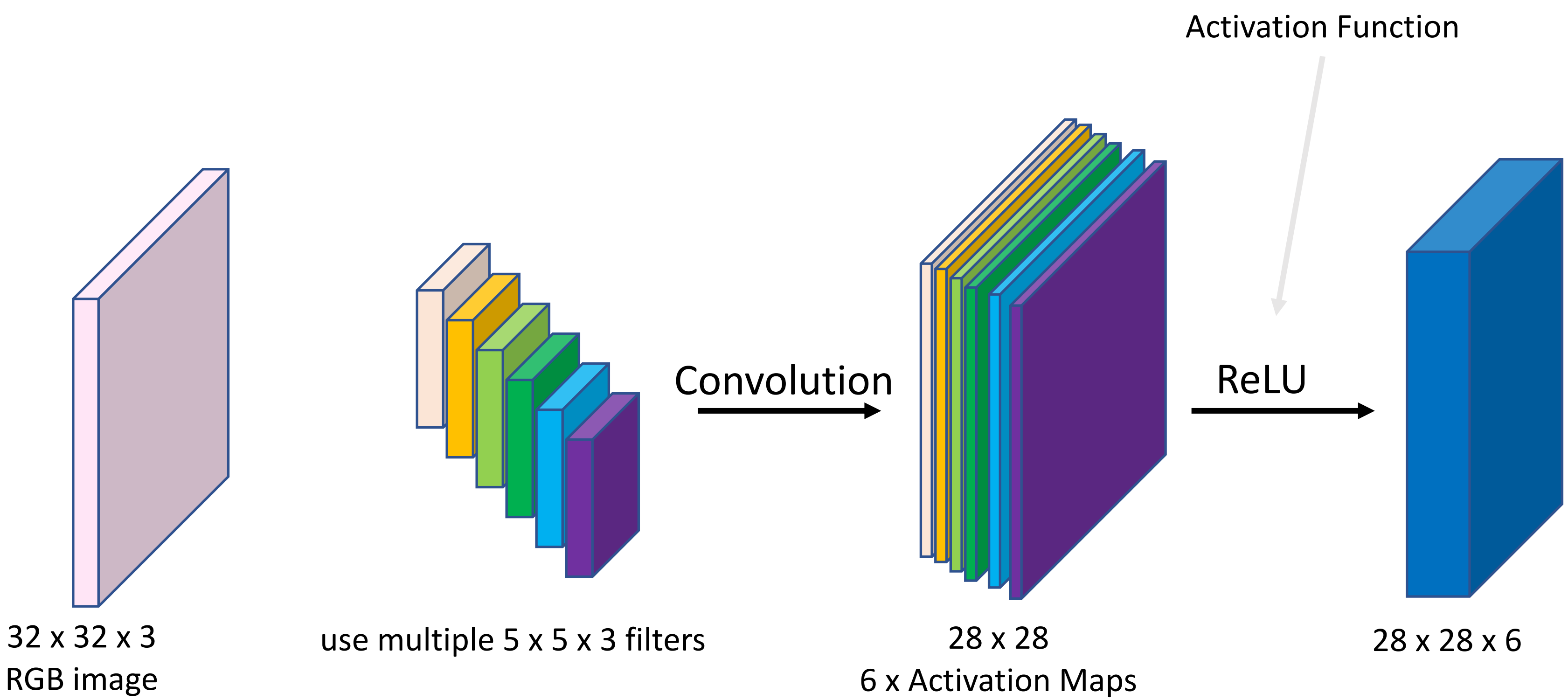
28 x 28 x 1
Activation Map

Filter processes all color channels
The mask is a 3D tensor

Determined automatically
during training

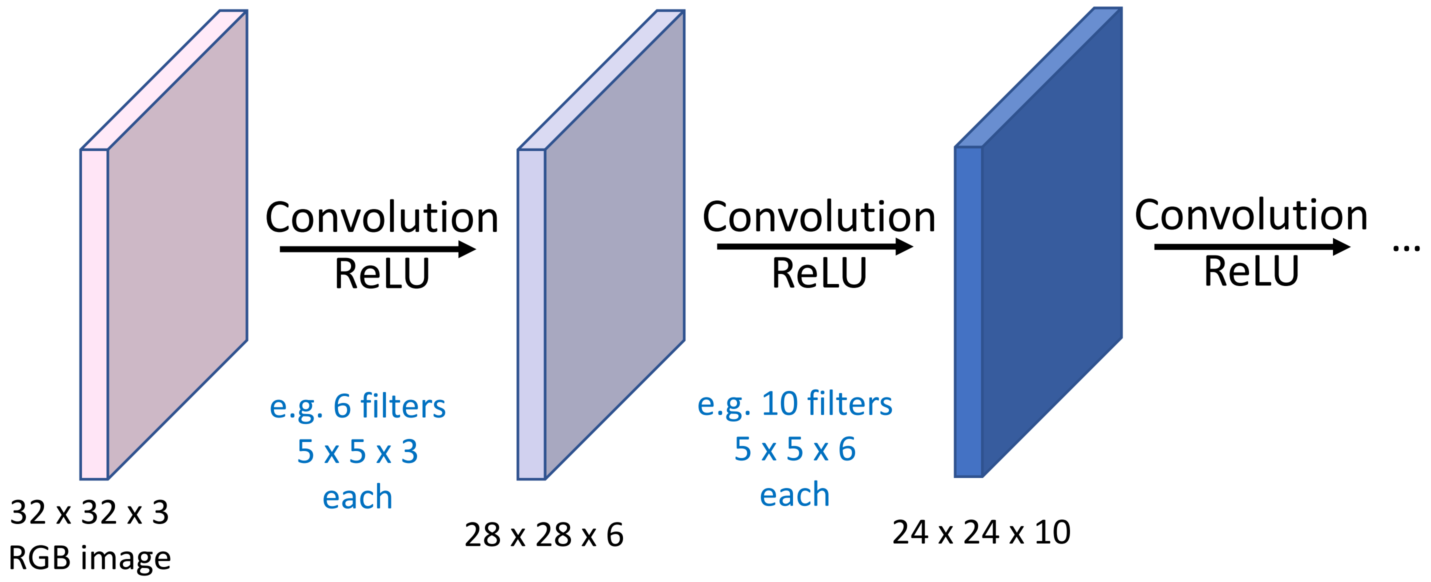
Each filter results in a **single**
value for each location

Convolutional Layer



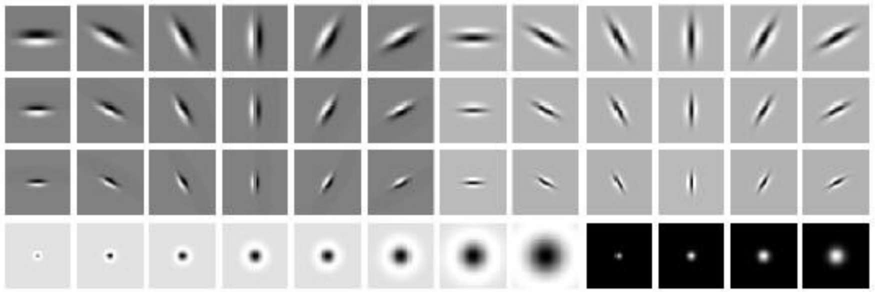
Convolutional Layer

Convolution and activation are now repeated several times
Idea: Combine low-level features, combine again etc.



Convolution = Feature Extraktion

Filter bank with k filters

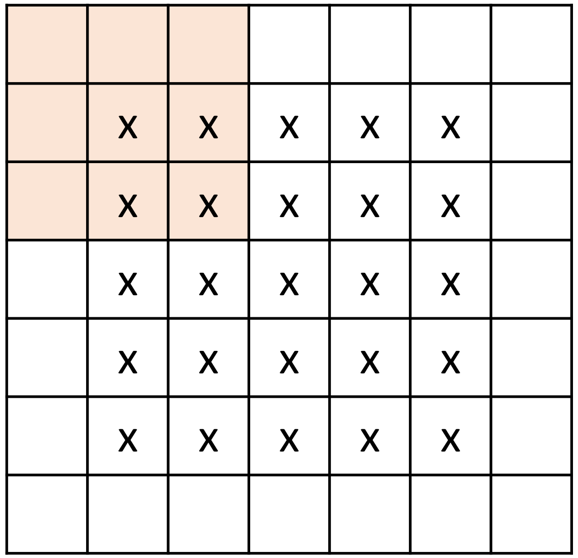


k Feature Maps

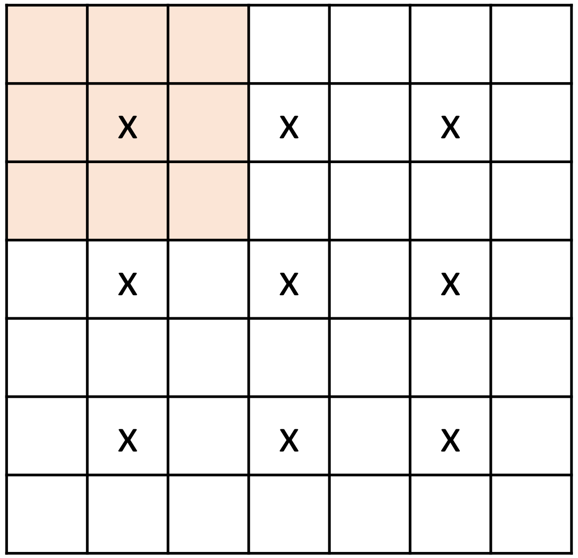
Hyperparameter – Stride

the filter mask can be moved by more than one pixel (stride)
this differs from the “normal“ convolution operation

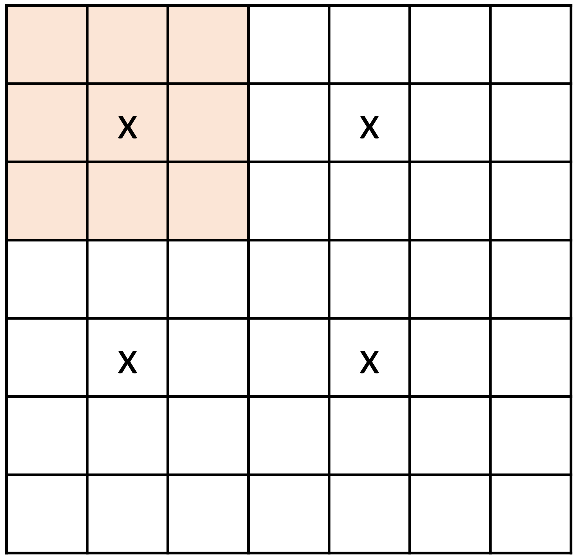
Example: 7x7 image with 3x3 filter



Stride 1
Output: 5x5

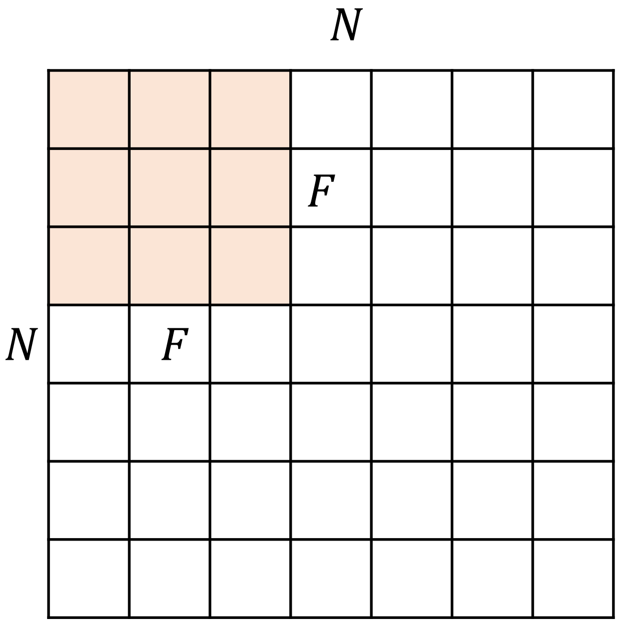


Stride 2
Output : 3x3



Stride 3
asymmetric border – stride does not match

Hyperparameter – Stride



Stride S

Size of output: $\dfrac{N-F}{S}+1$

If result is integer: Stride and filter size match

Example $N=7, F=3$:

$S=1: \frac{7-3}{1}+1=5$

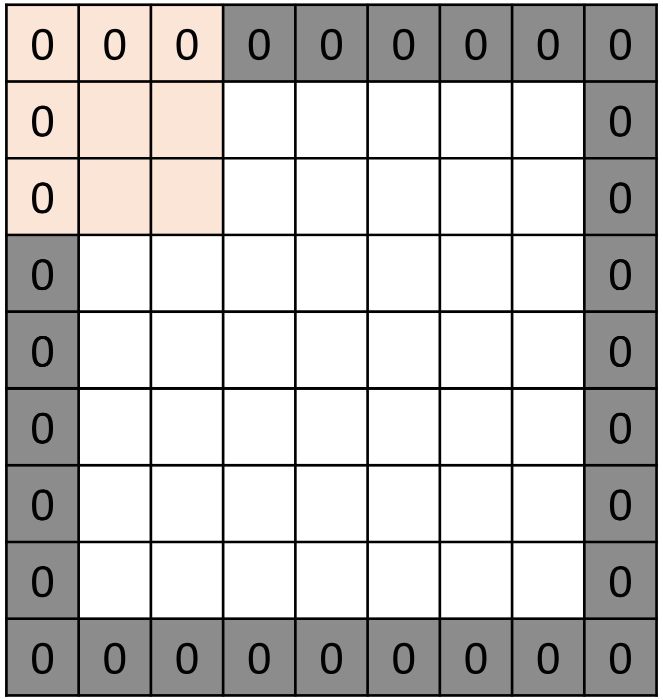
$S=2: \frac{7-3}{2}+1=3$

$S=3: \frac{7-3}{3}+1=2{,}33$

Hyperparameter – Pad

Technische Hochschule Rosenheim

- Problem: Input size for a layer is getting smaller and smaller
- Solution: Padding of border
  - with zeros (Zero-Padding)
  - with copies of the border pixels



For filter size $F \times F$

$\frac{F-1}{2}$ values are lost at the border

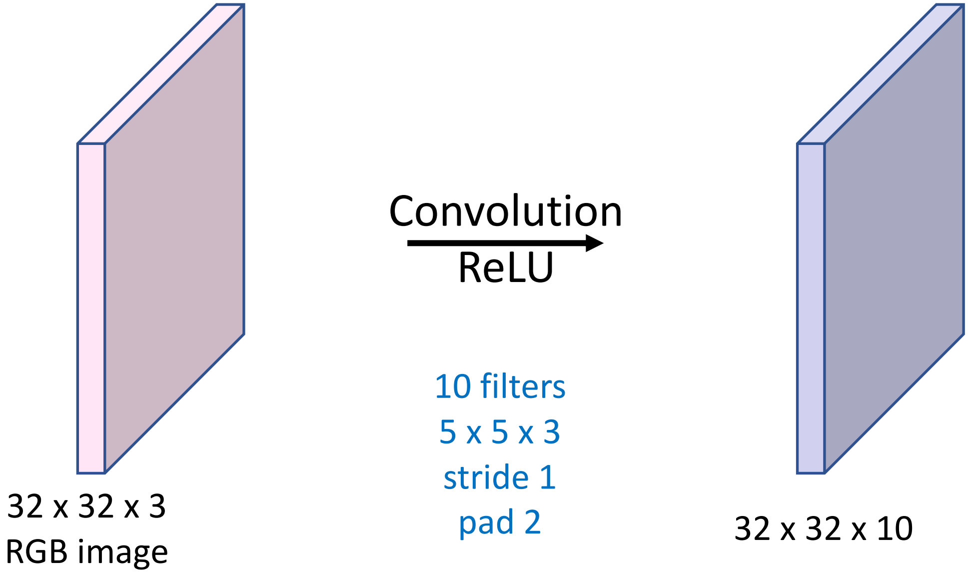
Examples:

$F = 3$: Padding with 1

$F = 5$: Padding with 2

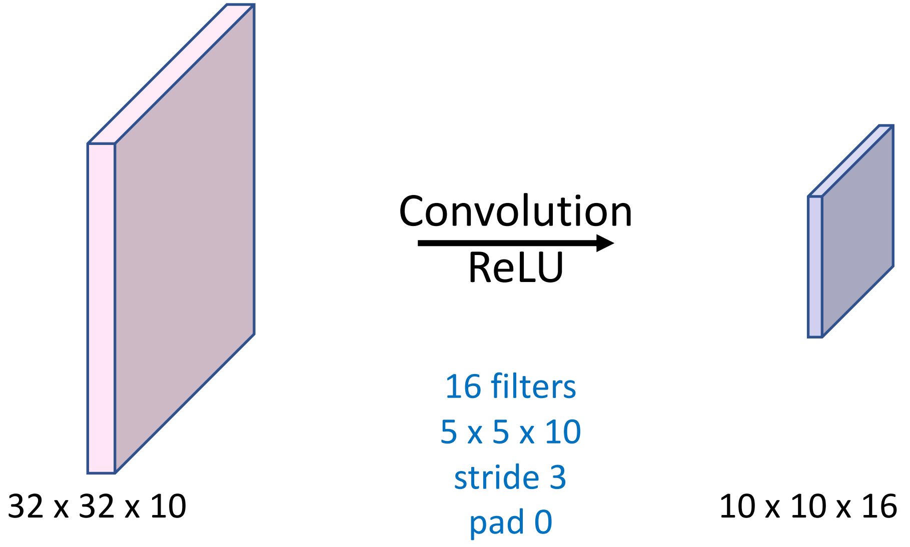
$F = 7$: Padding with 3

Example



Number of parameters for this layer:
each filter has $5 \cdot 5 \cdot 3 + 1 = 76$ parameters (+1 because of bias)
10 filters, total: $76 \cdot 10 = 760$ parameters

Example



Number of parameters for this layer:
each filter has $5 \cdot 5 \cdot 10 + 1 = 251$ parameters (+1 because of bias)
16 filters, total: $251 \cdot 16 = 4016$ parameters

Hyperparameters – Convolution

- Number K and size F of filters
- Stride S
- Size of padding P
- typical values:
  - K = power of 2, e.g. 32, 64, 128, 512
  - $F = 3, \quad S = 1, P = 1$
  - $F = 5, \quad S = 1, P = 2$
  - $F = 5, \quad S = 2, P =$ matching
  - $F = 1, \quad S = 1, P = 0$
- transforms a layer of size $W \times H \times D$ into a layer of size $W' \times H' \times D'$:

$$W' = \frac{W - F + 2P}{S} + 1, \; H' = \frac{H - F + 2P}{S} + 1, D' = K$$

- Number of weights: $(F \cdot F \cdot D) \cdot K + K$

Pooling

- scaling does not change the object
- objective: smaller-sized layers

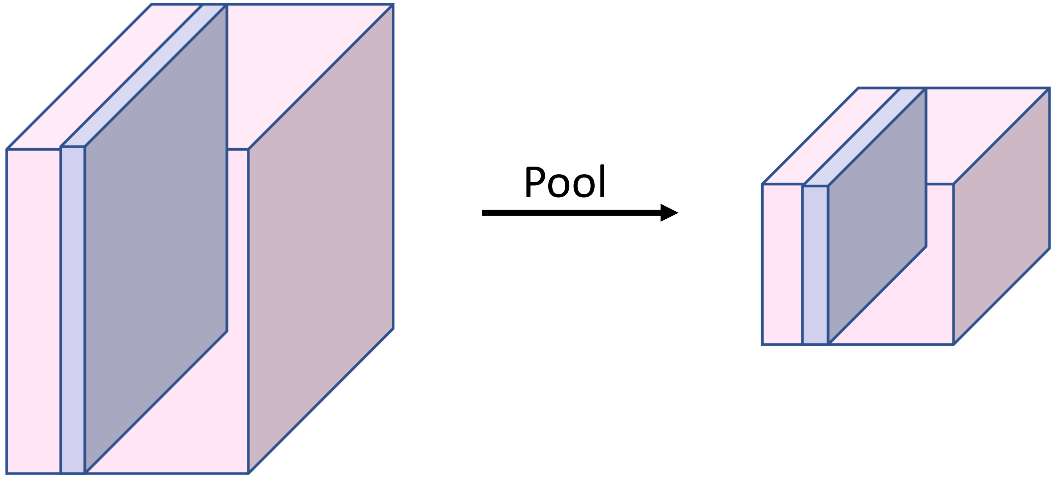
Bird

Scaling (subsampling)

Bird

Pooling

each activation map is processed separately



200 x 200 x 16

100 x 100 x 16

Pooling

MAX-Pooling: Use the largest element within a windows of size $F \times F$

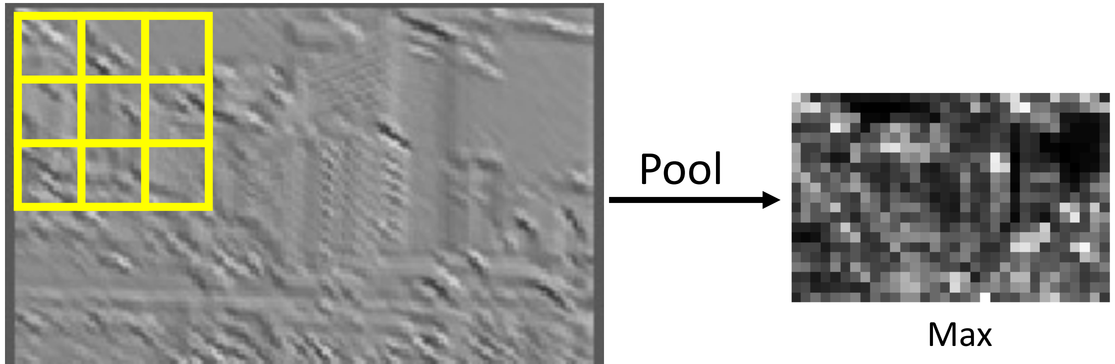
Average-Pooling: Use the mean value of all elements within a windows of size

Example: MAX-Pooling using 2x2 windows and stride $S = 2$

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

Pool →

7	5
9	8



Hyperparameters – Pooling

- Size F of windows
- Stride S
- Typical values:
  - $F = 2, \quad S = 2$
  - $F = 3, \; S = 2$

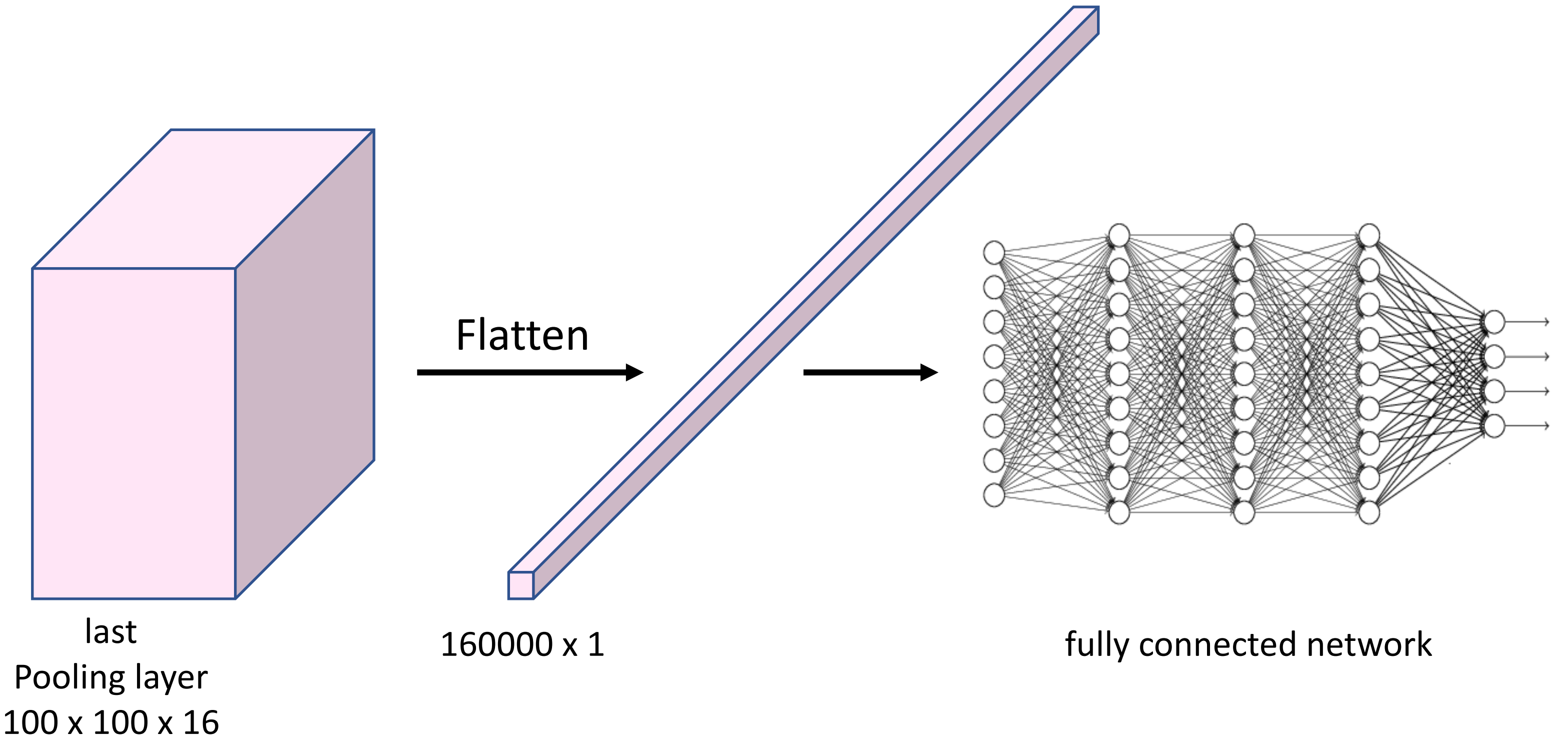
- transforms a layer of size $W{\times}H{\times}D$ into a layer of size $W'{\times}H'{\times}D'$:

$$W' = \frac{W-F}{S} + 1, H' = \frac{H-F}{S} + 1, D' = D$$

- Number of weights: none

Fully Connected Layers / Flatten

- at the end: fully connected layers as before (MLP)
  → Flattening

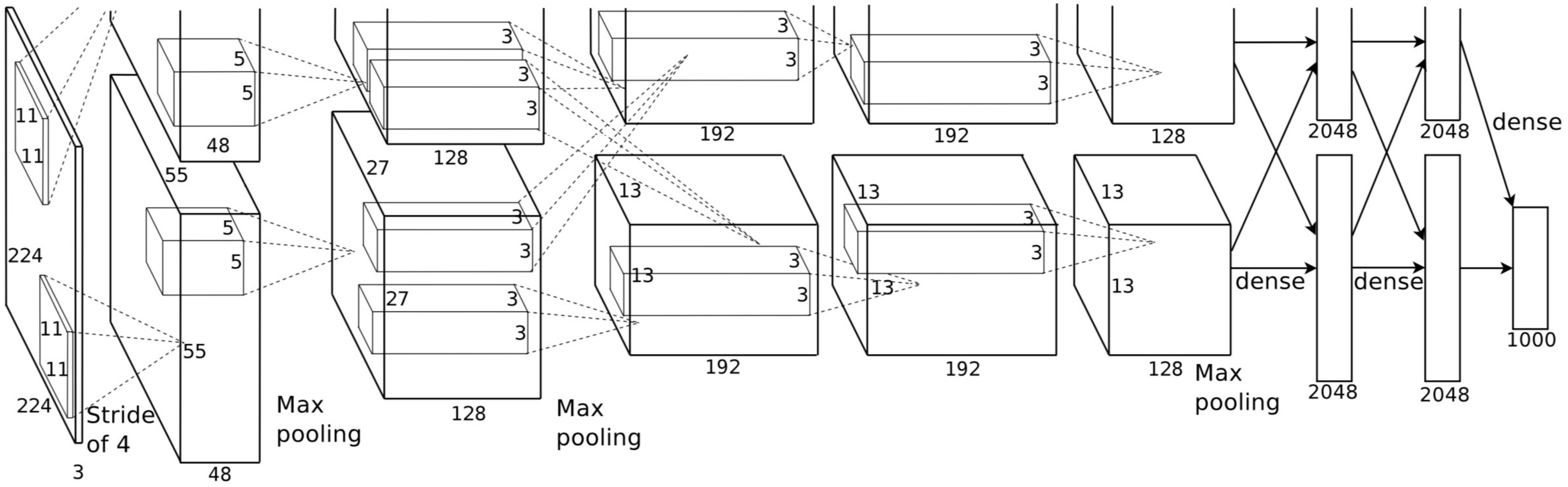


Typical Standard Architecture

$$\text{Input image} \rightarrow \underbrace{\underset{n\text{ times}}{\text{convolution ReLU}} \rightarrow \text{pool}}_{m\text{ times}} \rightarrow \underset{k\text{ layers}}{\text{fully connected ReLU}} \rightarrow \text{output Softmax}$$

- n ca. 3, up to ca. 5
- m large
- $0 \leq k \leq 2$
- General tendency:
  - use smaller filter sizes and deeper architecture
  - away from pooling/fully connected layers towards pure convolutional layers

Alexnet



ImageNet Classification Challenge 2012
- 1000 classes
- 1.2 million training images
- 50,000 validation images
- 150,000 test images

Network:
- 650,000 neurons
- 60 million parameters
- used CNN with ReLU on GPU for the first time

Pre-Processing:
- Scale/Crop images to 256 x 256
  (training uses random crops of size 224x224 from these)
- Subtract mean RGB image

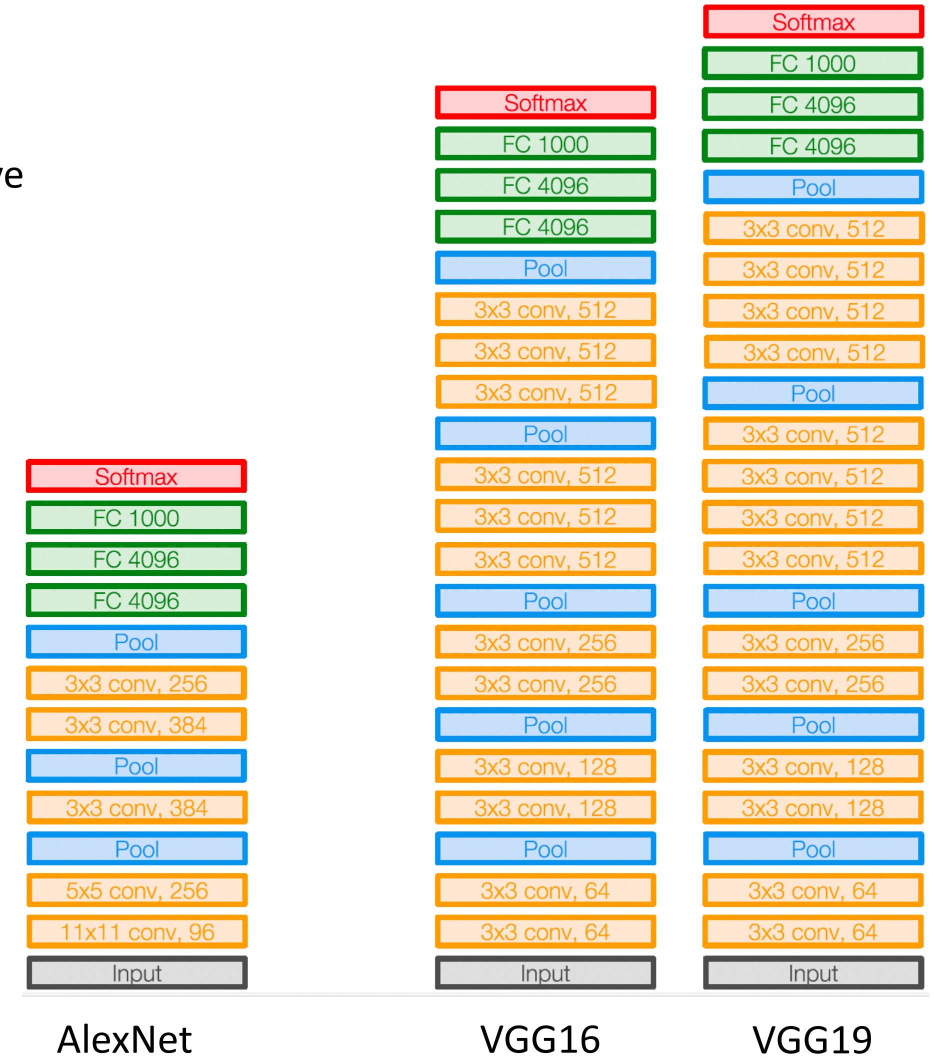
Krizhevsky, Sutskever, Hinton: ImageNet Classification with Deep Convolutional Neural Networks. Commun. ACM 60(6):84-90, 2017.

VGGNet

- 8 layers (AlexNet) → 16-19 layers (VGG16/19)
- 3x3 convolution only, stride 1, pad 1; 2x2 max-pool stride 2

- a series of three 3x3 convolution layers has the same effective receptive field as a single 7x7 filter layer
  - but: three 3x3 is deeper, with more non-linearities
  - and has less parameters:
    - one 7x7 layer with depth d has $49d^2 + d$ weights
    - three 3x3 layers only $27d^2 + d$

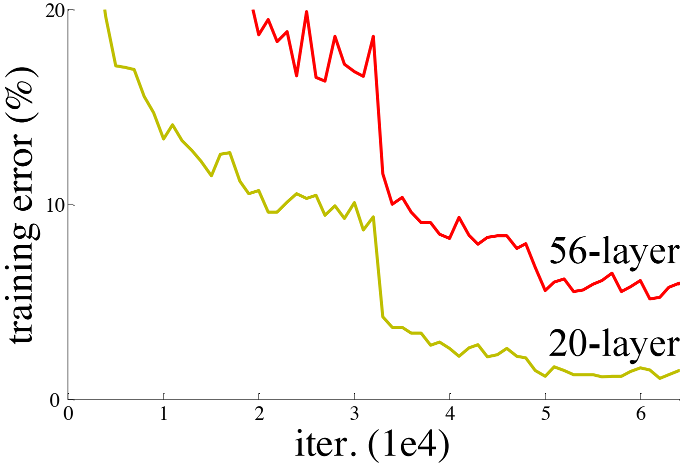
VGG16: 138 million parameters
VGG19: 144 million parameters

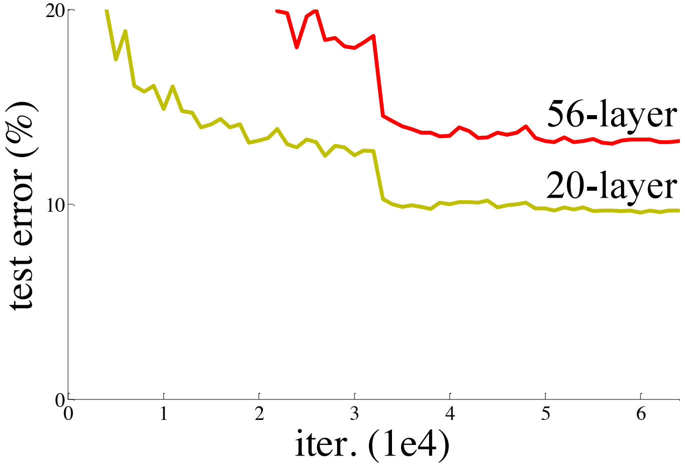


K. Simonyan, A, Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition“. International Conference on Learning Representations, 2015. https://arxiv.org/abs/1409.1556

So, more and more Layers?

What happens when we use more layers and deeper networks?



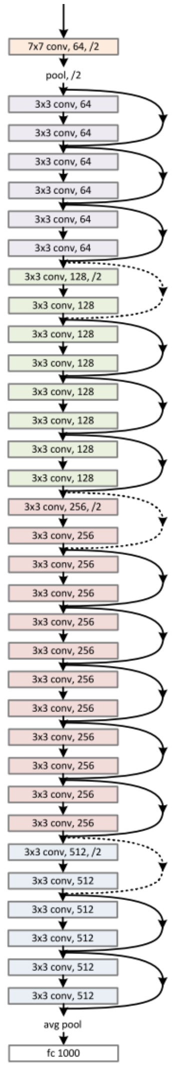


The model with 56 layers is obviously worse – in training as well as test

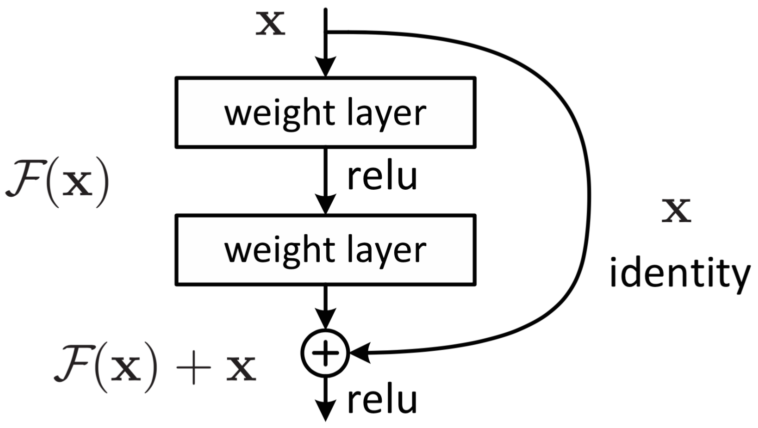
- The deeper network is worse. But this is not caused by overfitting.
- Conjecture: the optimization problem is harder for deeper networks

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition, 2015.
https://arxiv.org/abs/1512.03385

ResNet – Residual Neural Network



- Connections can be skipped
- No sequence of fully connected layers at the end
- Batch Normalization



Idea: A deeper network should be at least as good as a flat one.

Problem:

- when there is no change from one block to the next, we’d just need an identity mapping
- in a standard CNN this is cumbersome: has to be created by training weights

Solution in ResNet:
Copy trained layers from flat model, set additional layers to identity mapping.

Result:

- shortcuts without additional parameters
- when identity is required: just set weights to zero

K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition, 2015.
https://arxiv.org/abs/1512.03385

Classification, Localization, Detection, Segmentation

Classification

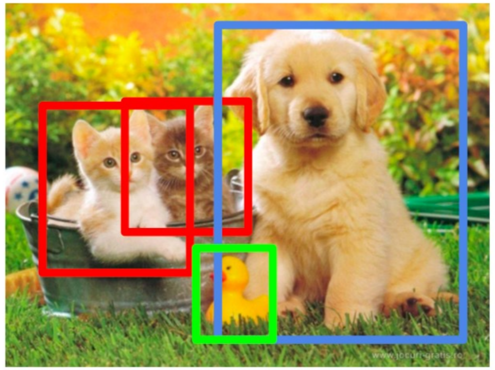
Cat

Localization

Cat

single object

Detection



Cat, Duck, Dog

multiple objects

Segmentation

Cat, Duck, Dog

semantic segmentation
do not differentiate instances,
only care about pixels

More on that: **Computer Vision** class winter term

images: Li, Karpathy, Johnson, CS231n,
lecture 8, Winter 15/16, Stanford

Sources

- Goodfellow, Bengio, Courville: Deep Learning, MIT Press, 2017.
  http://www.deeplearningbook.org/
- Li, Johnson, Yeung: CS231n: Convolutional Neural Networks for Visual Recognition. Vorlesung Stanford University, 2018.
  http://cs231n.stanford.edu/
- Li: Deep Learning and Its Applications. Lecture University of Waterloo, 2017.
  https://cs.uwaterloo.ca/~mli/cs898-2017.html
- Original research articles as stated on the slides