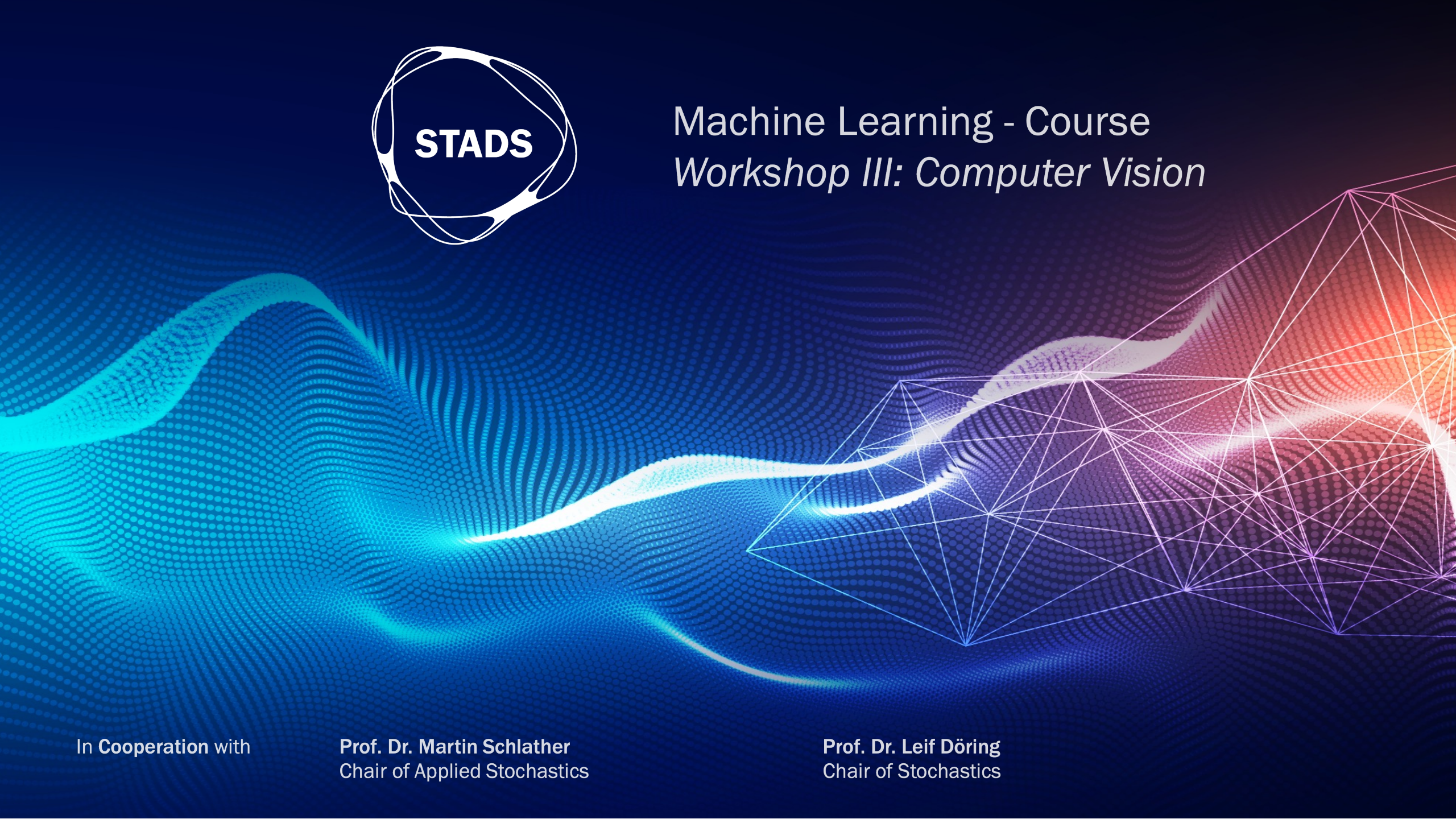


Machine Learning - Course

Workshop III: Computer Vision

In **Cooperation** with

Prof. Dr. Martin Schlather
Chair of Applied Stochastics

Prof. Dr. Leif Döring
Chair of Stochastics

Agenda

Introduction

Theory of CNNs

Programming Project

Agenda

- Introduction
- Theory of CNNs
- Programming Project

What is Computer Vision?

Computer Vision

Computer vision is a field **within artificial intelligence** (AI) that enables computers and system to extract meaningful information from digital **images, videos, and other visual inputs** - and take action or make recommendations based on that information. If AI enables computers to think, **computer vision enables them to see, observe and understand.**[1]

Tasks

1. Image classification
2. Object classification
3. Object localization
4. Object recognition
5. Semantic and instance segmentation

Parts

Classical Machine Learning or Deep Learning

+

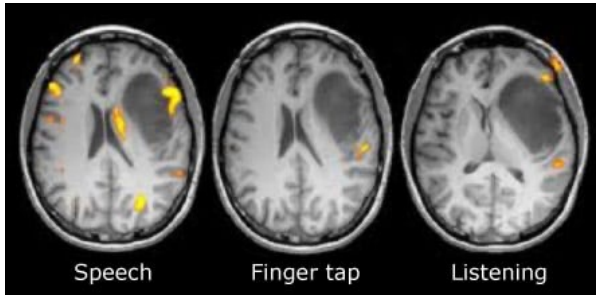
Operations
1. Conv Layer
2. Pooling Layer

1 https://www.ibm.com/de-de/topics/computer-vision

Exciting and important applications

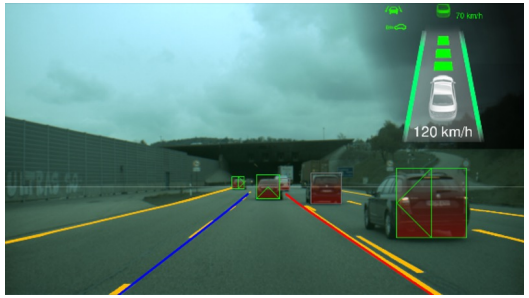
Medicin

fMRI Scans



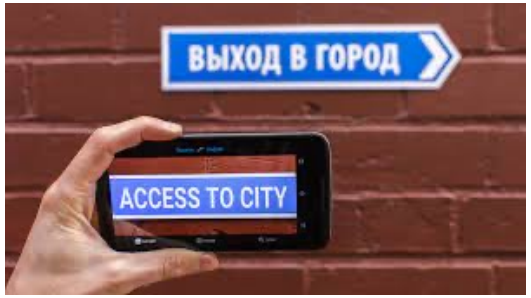
Security

Facial recognition at the airport



Autonomous driving

Recognize people and traffic signs

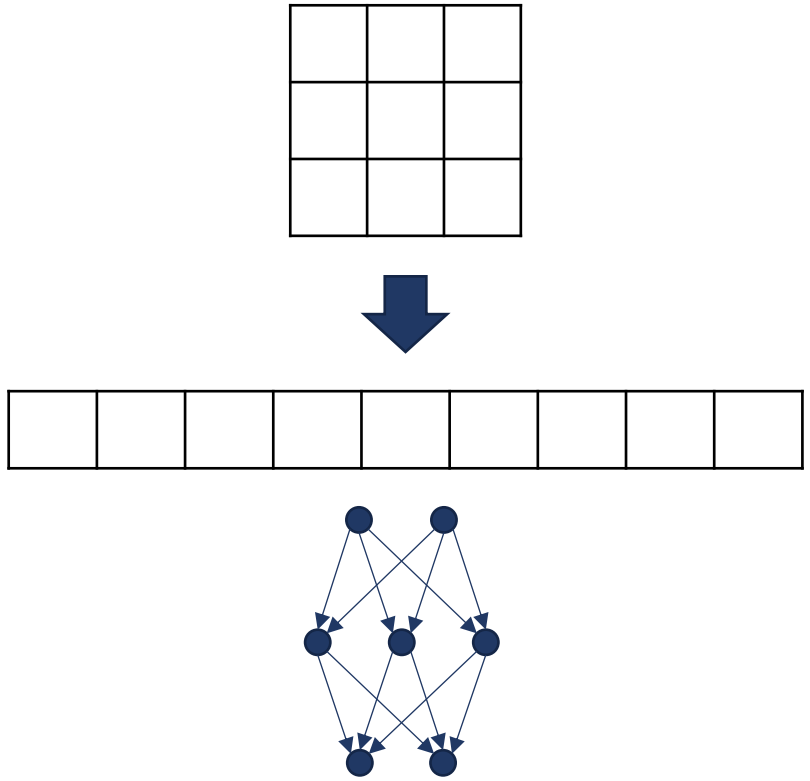


Samrtphone

Face ID, Google Translate

Feedfoward Neural Networks for Images

Naive Idea



Drawbacks

Many Parameter

- One weight per pixel
- E.g. MNIST data set for recognition of clothes: 28x28 pixels

Information Loss

- Loss of information about localities
- Loss when transferring to a vector
- e.g. regions on images, order of frames in videos

No translation variance

- Relevance of the object‘s position
- No information aggregation
- e.g.2x2 Pixel

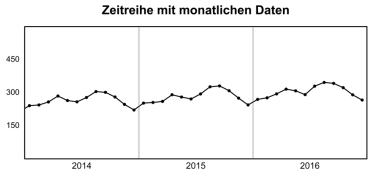
Different input sizes not possible

- Fixed input size of the feedforward neural network

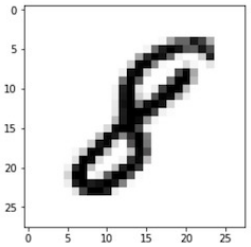
Convolutional Neural Networks for Grid Data

Grid Data

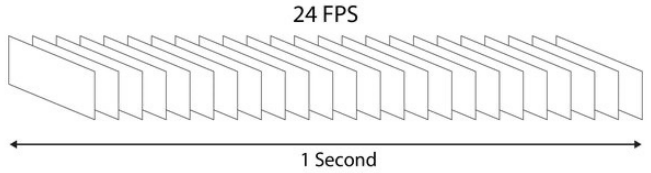
One-dimensional (e.g. Time Series)



Two-dimensional (e.g. images)



Three-dimensional (e.g. videos)



Operations along relevant dimensions

What is relevant?

Permutation along relevant dimensions leads to a loss of information

Examples

- Time Series
  - Time dimension is important
  - Spatial dimension is not
- Images
  - Spatial dimensions are important
  - Color channels are not
- Videos

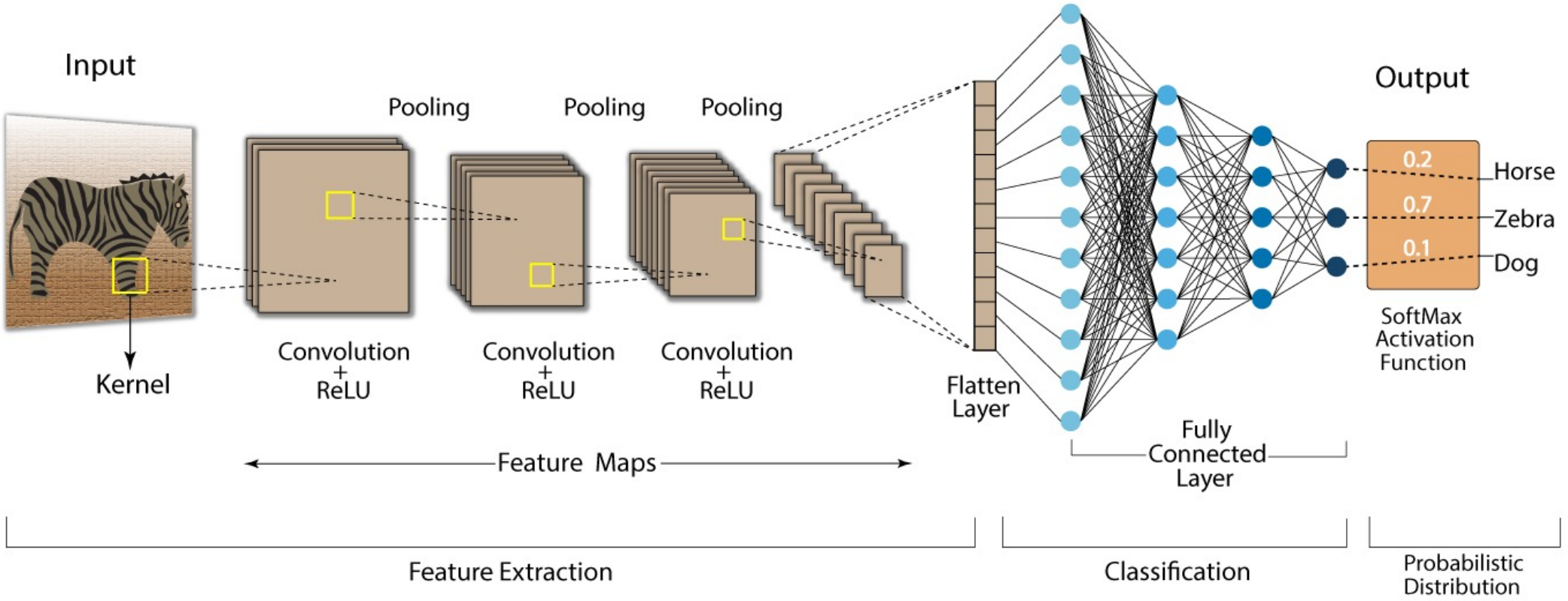
Agenda

Introduction

Theory of CNNs

Programming Project

Convolutional Neural Network (CNN)



https://cdn.zephyrnet.com/wp-content/uploads/2022/03/03070300/basics-of-cnn-in-deep-learning.png

Convolution

A convolution is an operation that generates a third function s from two functions x and w.
For CNNs, we consider **discrete convolution** only.

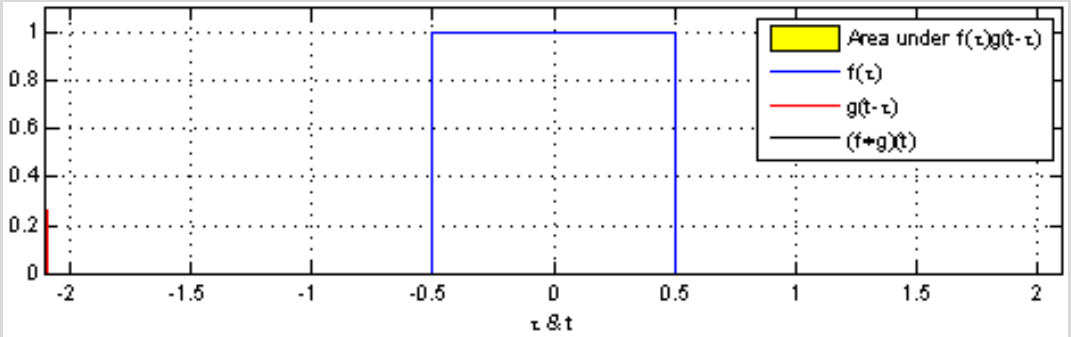
$$s(t) = (x * w)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau) \qquad \left[\, s(t) = (x * w)(t) = \int x(\tau)w(t-\tau)\,\mathrm{d}\tau \,\right]$$

Here $\boldsymbol{x}$ represents the input and $\boldsymbol{w}$ (or also $\boldsymbol{K}$) the so-called kernel (also called filter).
A distinction is made between t = output time, τ = input time and t-τ = 'age' of measurement.

Interpretation

Convolution calculates the weighted average of the Input $\boldsymbol{x}(\boldsymbol{\tau})$ at time $\boldsymbol{t}$.
Where the weight of $\boldsymbol{x}(\boldsymbol{\tau})$ of $\boldsymbol{a}=\boldsymbol{t}$-$\boldsymbol{\tau}$ depends on $\boldsymbol{w}(\boldsymbol{a})$.

Illustration of a continuous convolution (∫ instead of Σ).



https://upload.wikimedia.org/wikipedia/commons/6/6a/Convolution_of_box_signal_with_itself2.gif

Cross-correlation

Instead of the classical convolution the very similar **cross-correlation** is mostly used for CNNs. Nevertheless, it is refered to as convolution. This operation corresponds to the so-called inner product. The cross-correlation can also be interpreted as a "weighted sum“ in a discrete manner.

$$s(t) = (x \star w)(t) = \sum_{\tau=-\infty}^{\infty} x(t+\tau)w(\tau)$$

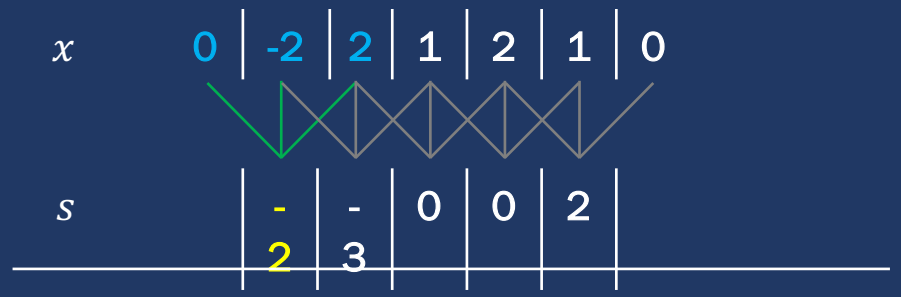
Discrete Convolution

$$s(t) = (x * w)(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau)$$

Example 1 – 1D Discrete Cross-correlation

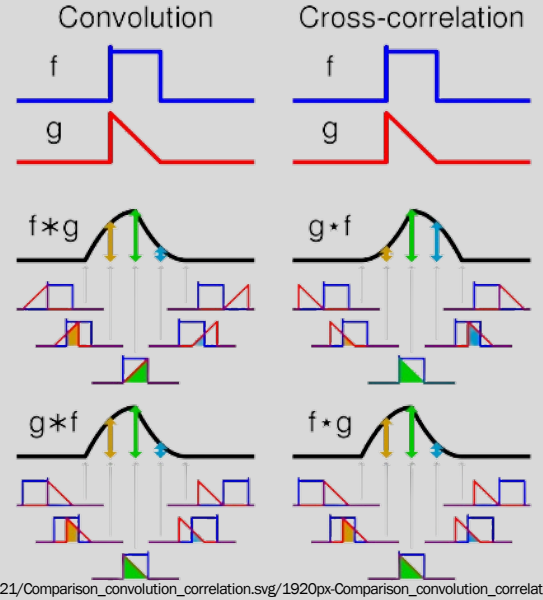
Input x = (0 -2 2 1 2 1 0

Filter w = (1 0 -1) ergibt



Example calculation: 0 * 1 + (-2) * 0 + 2 * (-1) = -2

Convolution
(continuous)
vs
Cross-correlation
(continuous)



https://upload.wikimedia.org/wikipedia/commons/thumb/2/21/Comparison_convolution_correlation.svg/1920px-Comparison_convolution_correlation.svg.png

Use of the Convolution in CNNs

Convolution of data

= "Application of the same filter to all local regions".
= „Examine whether a pattern appears in the data set“

Local Regions = Set of related grid points

- Mostly adjacent point sets of the input x
- Temporal or spatial
- e.g. 3x3 pixels
- Size of the region corresponds to the filter size

Filter (or Kernel) = Feature Detector

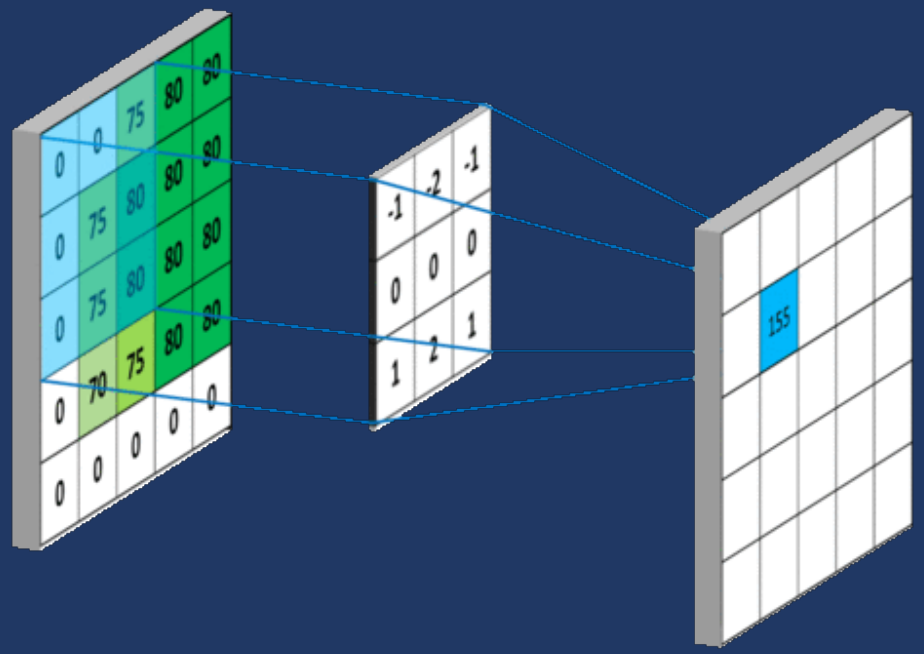
- Contains the weights for the convolution
- Is represented by kernel matrix (K, previously: w)
- Has the size of the local region under consideration
- Has the task to recognize patterns/objects

!

$$s(t) = (x \star K)(i,j) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(i+m, j+n) K(m,n)$$

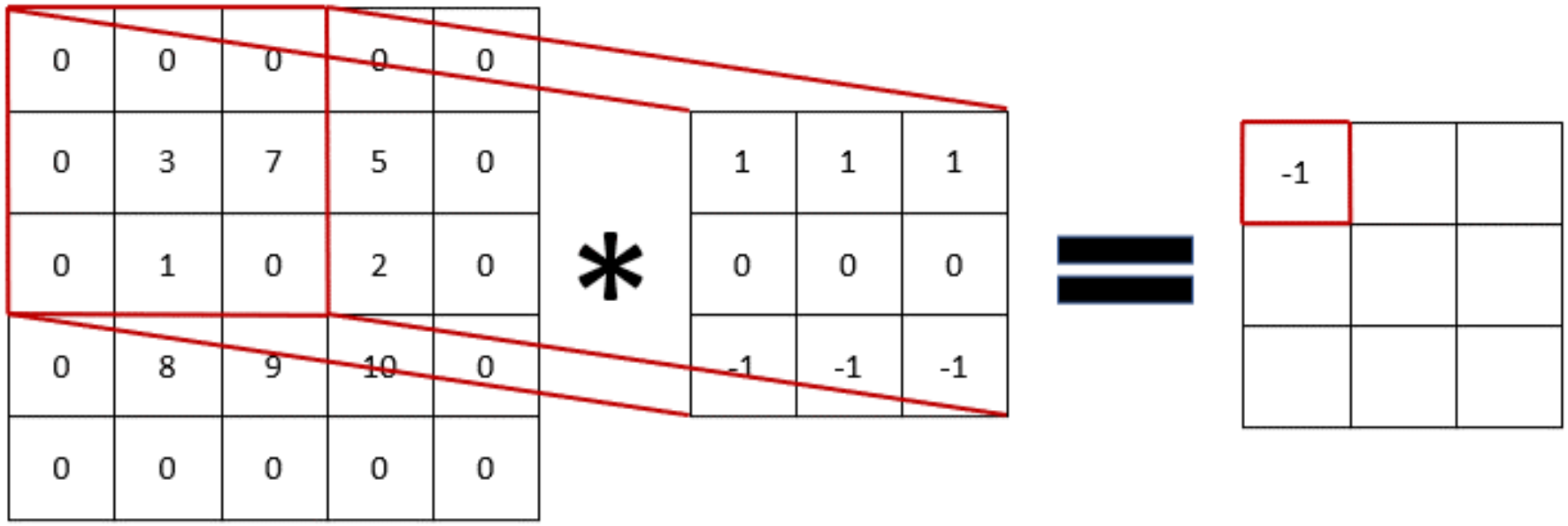
Example 2 – 2D Cross-correlation

Application of a filter (3x3 kernel-matrix) on a local region (3x3 Pixel) of the input (Image Matrix).



https://mlnotebook.github.io/post/CNN1/

Here again to count along ☺

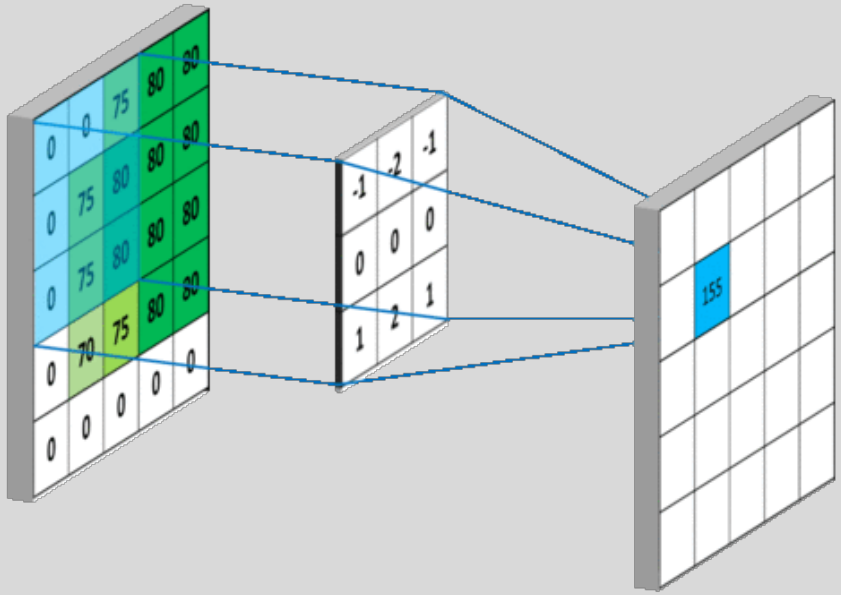


https://machinelearning-blog.de/wp-content/uploads/2022/03/Gif1.gif

Convolution – Zero-Padding

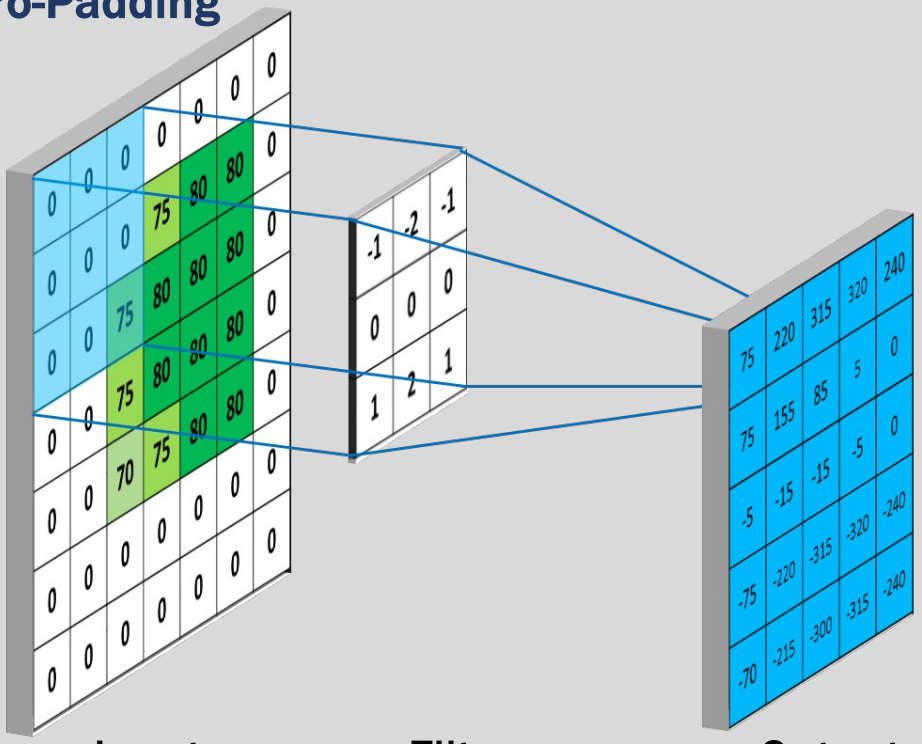
Goal Output-size = Input-size by manually adding zeros

No Zero-Padding



Input	Filter	Output
5x5	3x3	3x3
(m x m)	(f x f)	(m-f+1 x m-f+1)

Zero-Padding



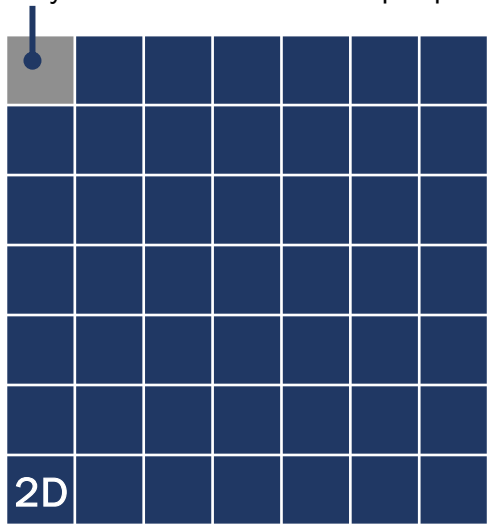
Input	Filter	Output
7x7 (actually. 5x5)	3x3	5x5
(m x m)	(f x f)	(m x m)

https://mlnotebook.github.io/img/CNN/convZeros.png

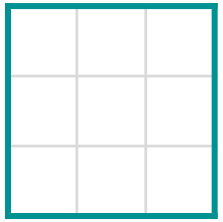
Convolution – Input-size depending on application

Input

Gray-values between 0-255 per pixel

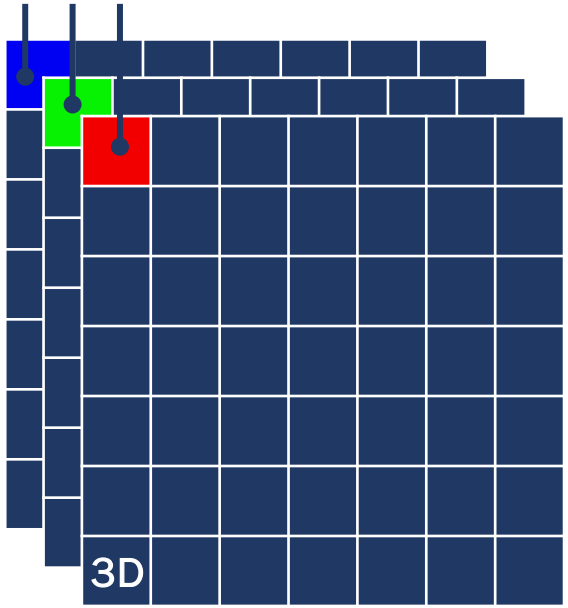


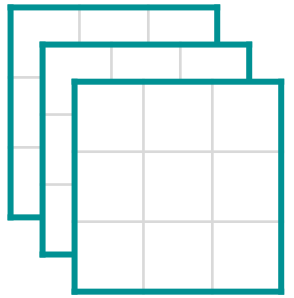
Filter 1x3



Filter 3x3

Color image (RGB) - 3 values between 0-255 per pixel.
Spatial/temporal layers are also possible instead of colors

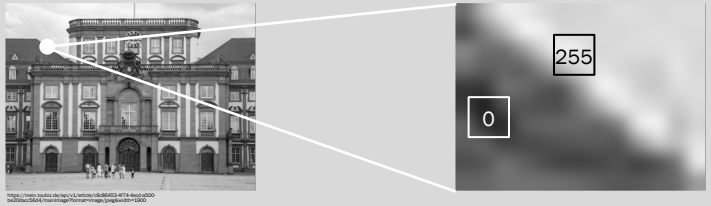




Filter 3x3x3

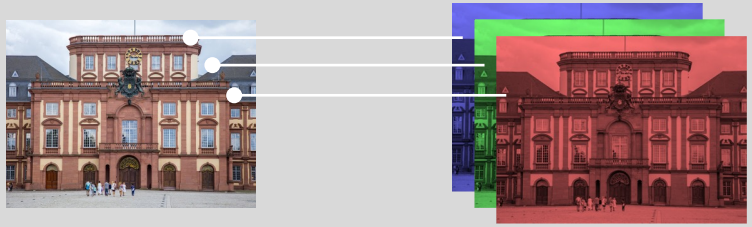
Schwarz-weiß Bilder als Input

(Height x Width) mit [0 (=schwarz),255 (=weiß)]



Farbbilder als Input

(Height x Width x 3) mit [0 (=schwarz),255 (=rot/grün/blau)]

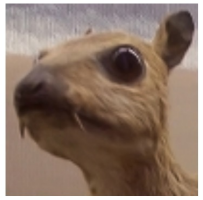


How do certain filters affect the input?

Example 1

Identity

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$



Box blur

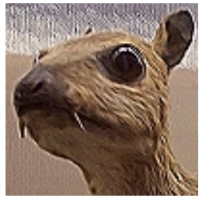
$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Edge detection

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Sharpen

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



https://en.wikipedia.org/wiki/Kernel_(image_processing)

Example 2

Identity

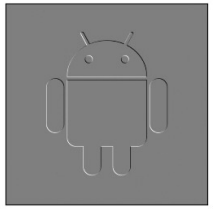
Verticals Detection

-1	0	1
-2	0	2
-1	0	1

Horizontals Detection

-1	-2	-1
0	0	0
1	2	1

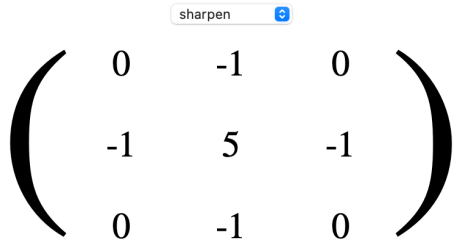
Verticals and horizontals combined



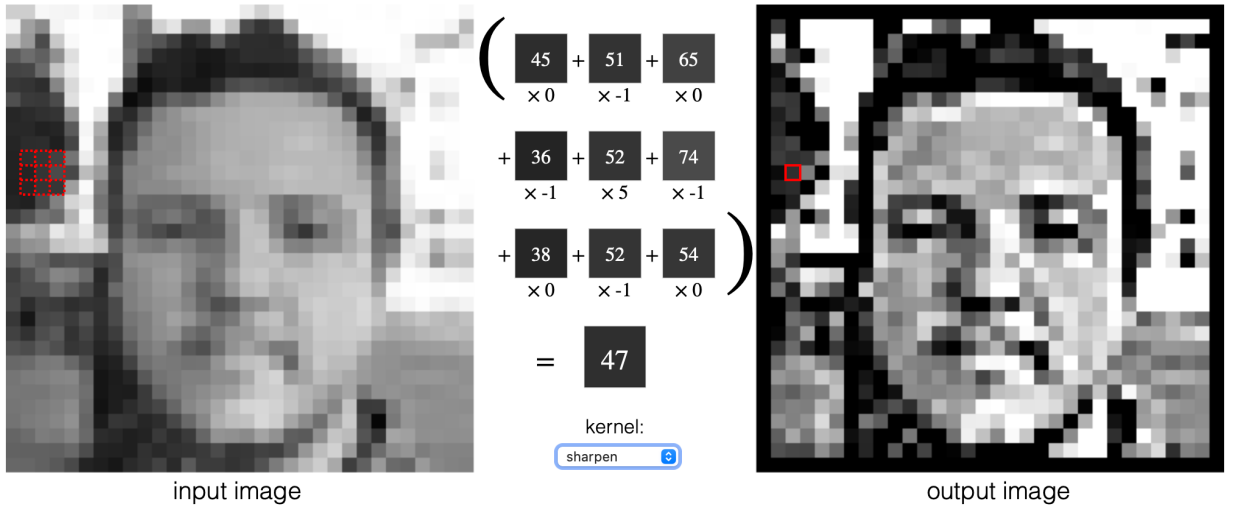
https://mlnotebook.github.io/post/CNN1/

Website to try your own filters

Let's walk through applying the following 3x3 **sharpen** kernel to the image of a face from above.

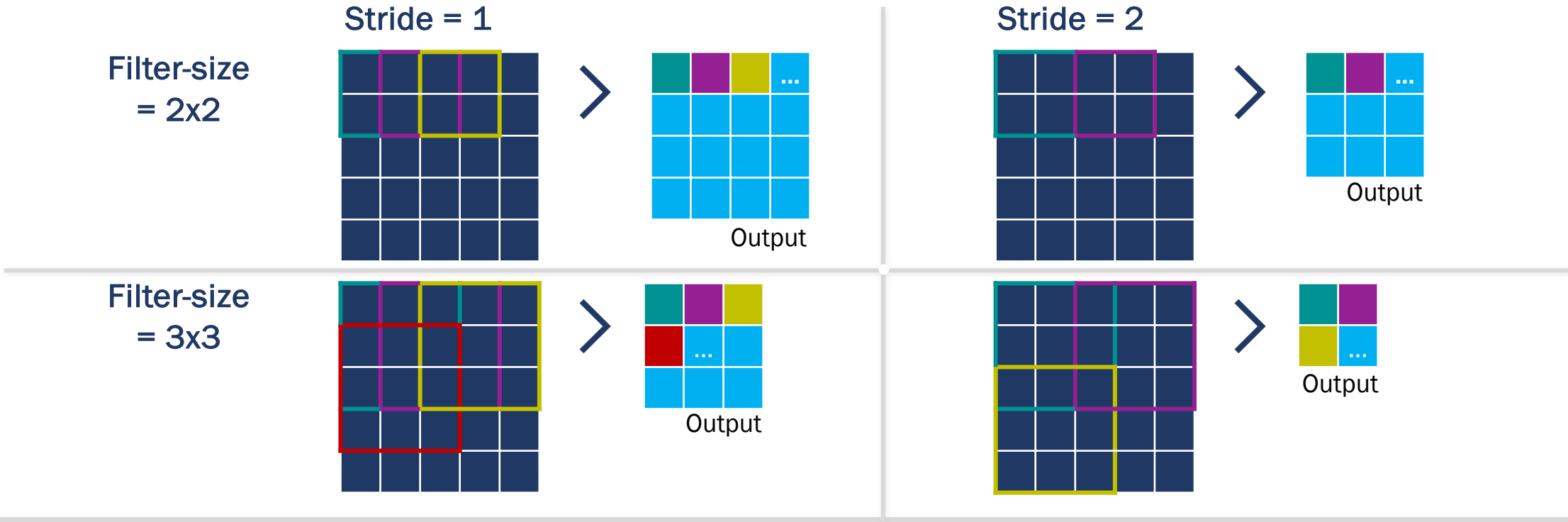


Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.



https://setosa.io/ev/image-kernels/

Convolution – Stride

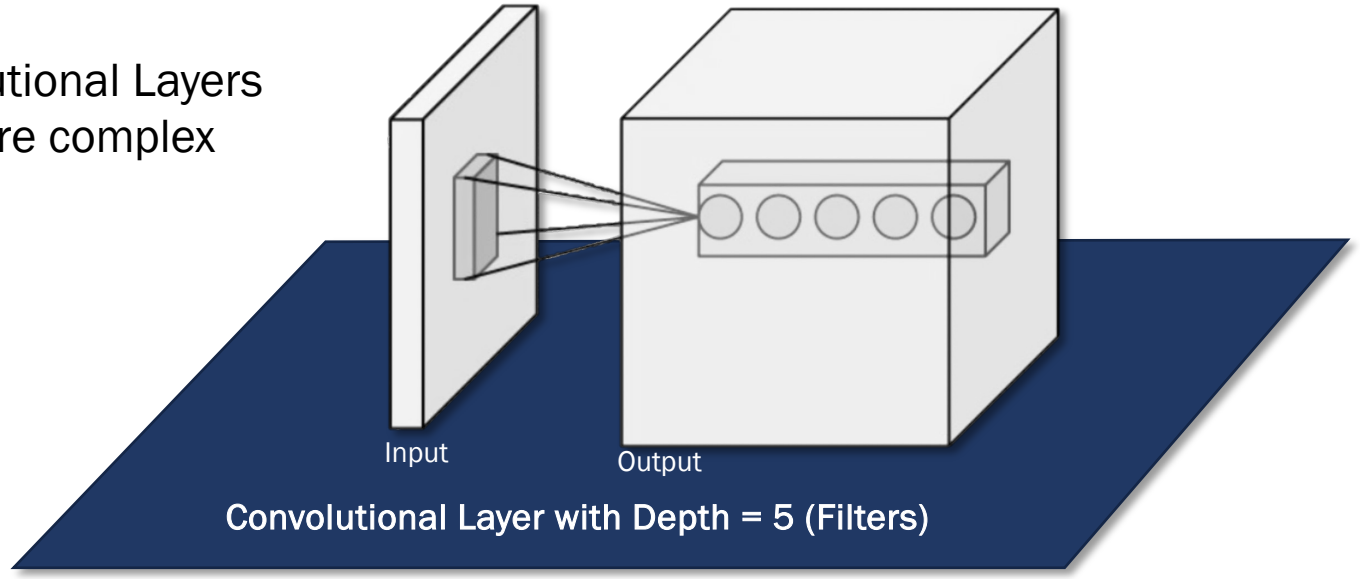


The same filter is shifted systematically over all data points.
The number of pixels by which the filter is shifted is called the stride (in previous examples: stride = 1).
The stride affects the output size ≈ input size / stride.
A small Stride results in more details in the output and features being visible multiple times (due to overlaps in the output).
A larger Stride reduces the output size and thus the complexity of the model and reflects coarser patterns.

Convolutional Layer – Filter as a Feature Detector

We now know how a filter works, but how is it used for image recognition?

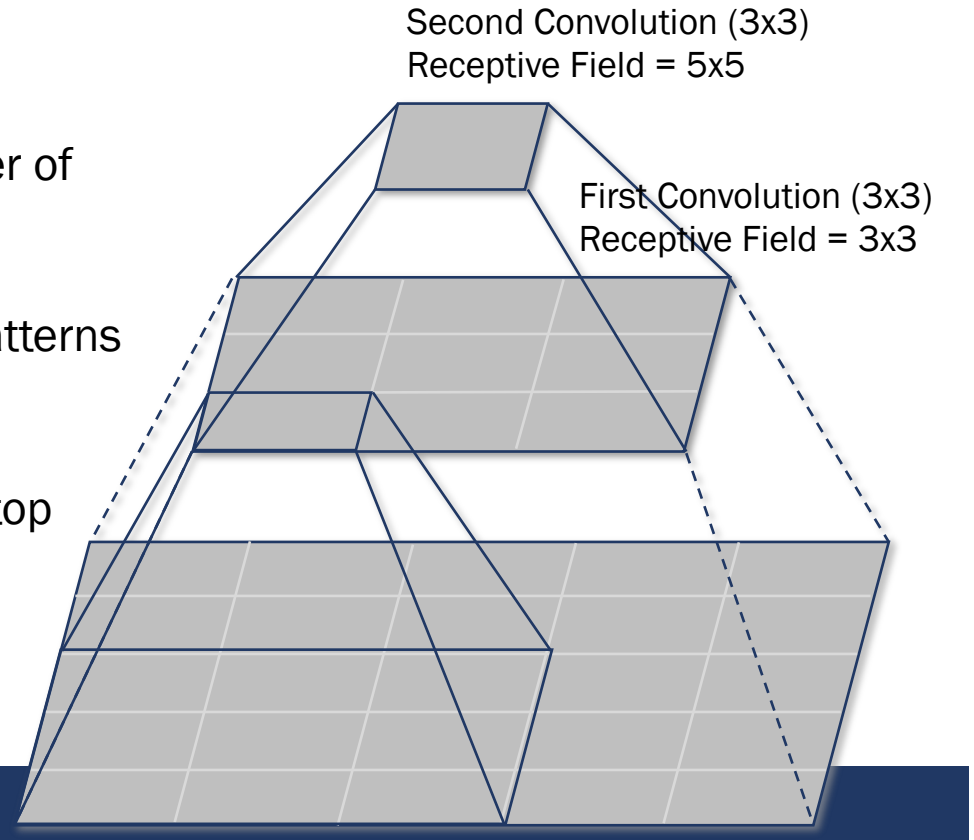
- Input (batch of images) ★ Filter set (feature detectors) = Output
- The filters are learned by the model and initialized randomly at the beginning.
- The size of the filter has an effect on the size of the recognized feature.
- A CNN consists of several Convolutional Layers, each with its own set of filters for different patterns.
- The consecutive connection of Convolutional Layers allows the recognition of more and more complex structures.



Receptive Fields

Each feature in a CNN that is characterized by a filter has a so-called receptive field.

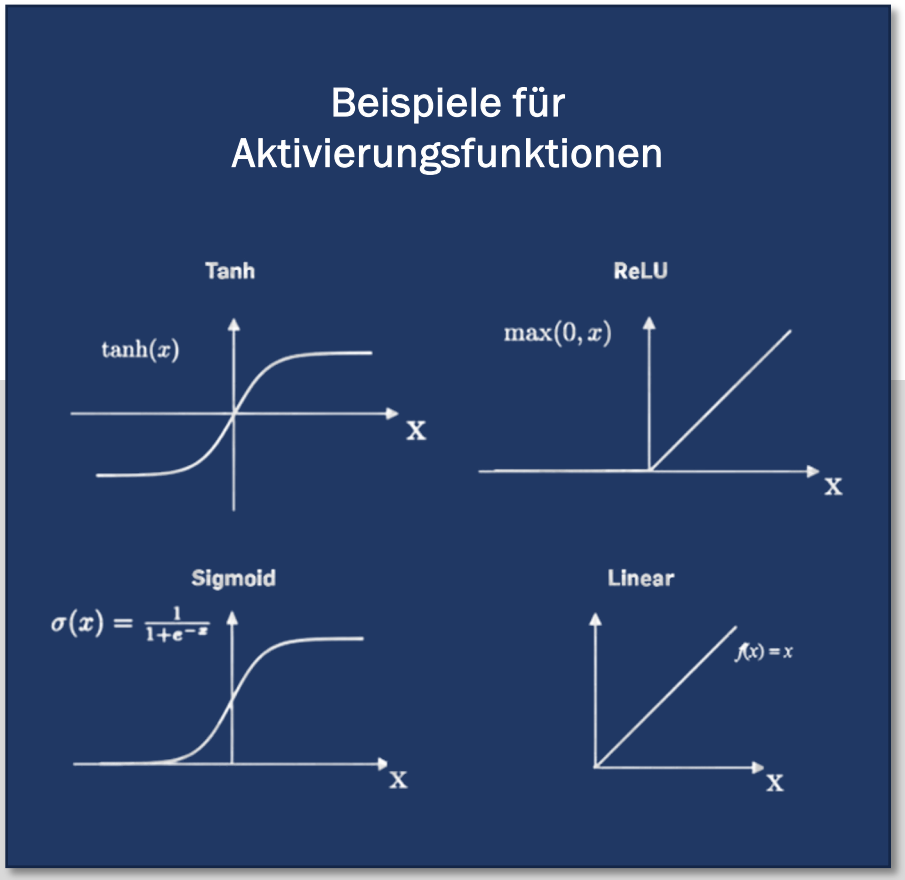
- The Receptive Field denotes the set of pixels that affect a considered feature in Convolutional Layer x.
- The receptive field becomes larger with increasing number of convolution and pooling operations.
- I.e. filters in later Convolutional Layers recognize larger patterns (= patterns where more pixels flow in).
- So instead of using large filters, we use several filters on top of each other.



Convolutional Layer – Non-Linearity

Every convolution is followed by a 'non-linearity' of the output in the form of an activation function.
Why do we need this non-linearity?

- The convolution itself is a linear operation.
- With the help of non-linear activation functions, non-linearities can be incorporated into the patterns.
- The more convolutional layers, each having an activation function, the more complex structures can be recognized.
- The output of the activation function is called feature map.



Input
(=Output of the Convolution)

-249	-91	-37
250	-134	101
27	61	-153

ReLu(x)= max(0,x)

Feature Map

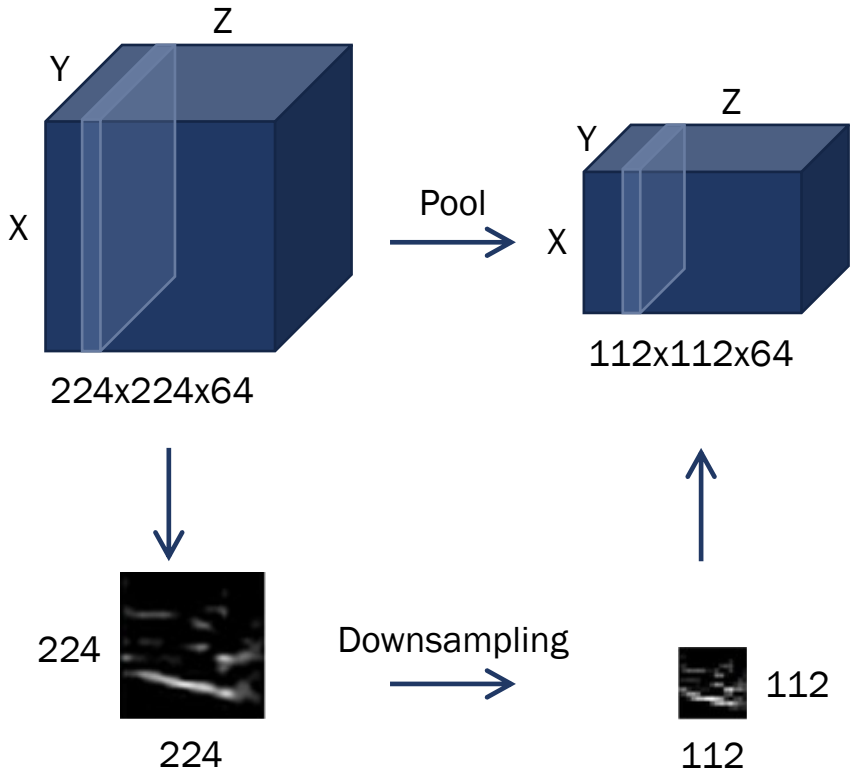
0	0	0
250	0	101
27	61	0

Convolution or Pooling Layer

Pooling Layer - Max and Average Pooling

What is Pooling?

Pooling is an operation that operates like a filter on the input and reduces the output size by increasing the stride.



Max & Average Pooling

Single Depth Slice z

12	20	30	0
8	12	2	0
34	70	37	4
112	100	25	12

Y

X

Max Pooling

2x2 Filter und Stride=2

20	30
112	37

Y

X

or

Average Pooling

(rounded)

2x2 Filter und Stride=2

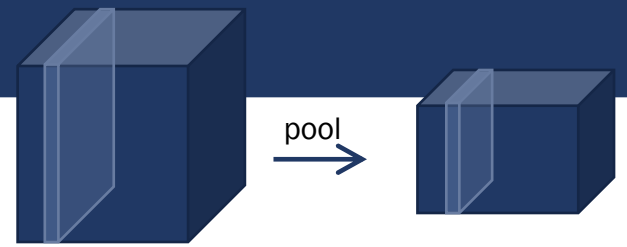
13	8
79	20

Y

X

Pooling Layer

Why do we need Pooling?



Reduction of the output size by a factor

Increased neural network efficiency and faster training

Reduction of sensitivity for small image shifts ("location invariance")

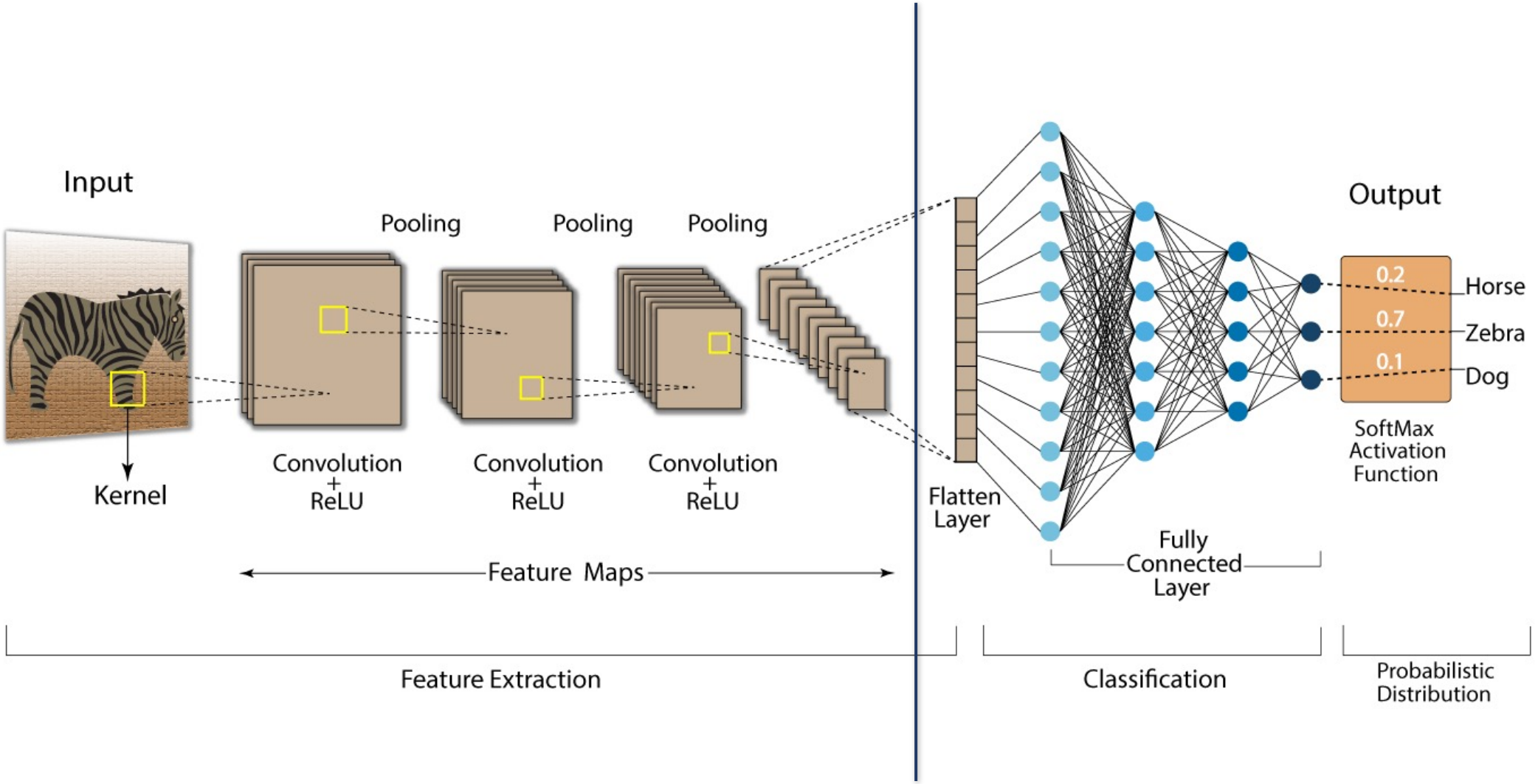
→ Especially useful for computer vision tasks where only the presence of an object is of interest.

Allows classification of images of different sizes

Max Pooling Focusses on extreme values, summarizes the presence of a feature over larger areas

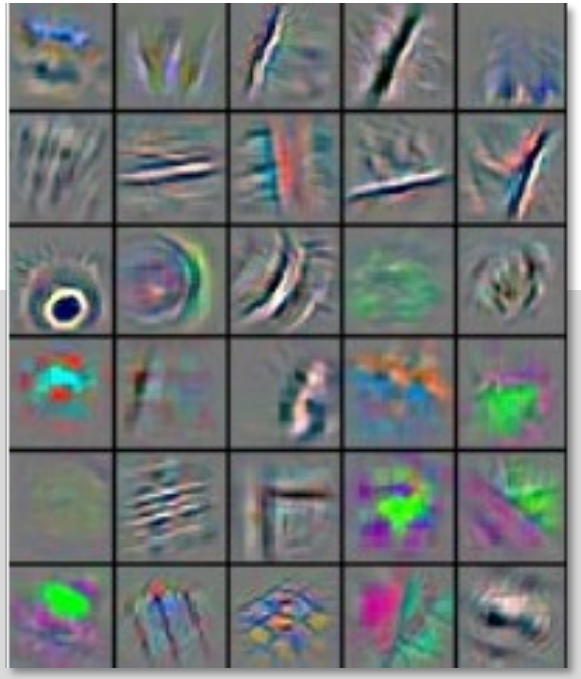
Average Pooling Creates a smoother output, mainly used for size reduction

Network Architecture of a Convolutional Neural Network



Example: Filter in different Convolutional Layers

Filter in Convolutional Layer 1
(= feature of size 7x7x3 Pixel)



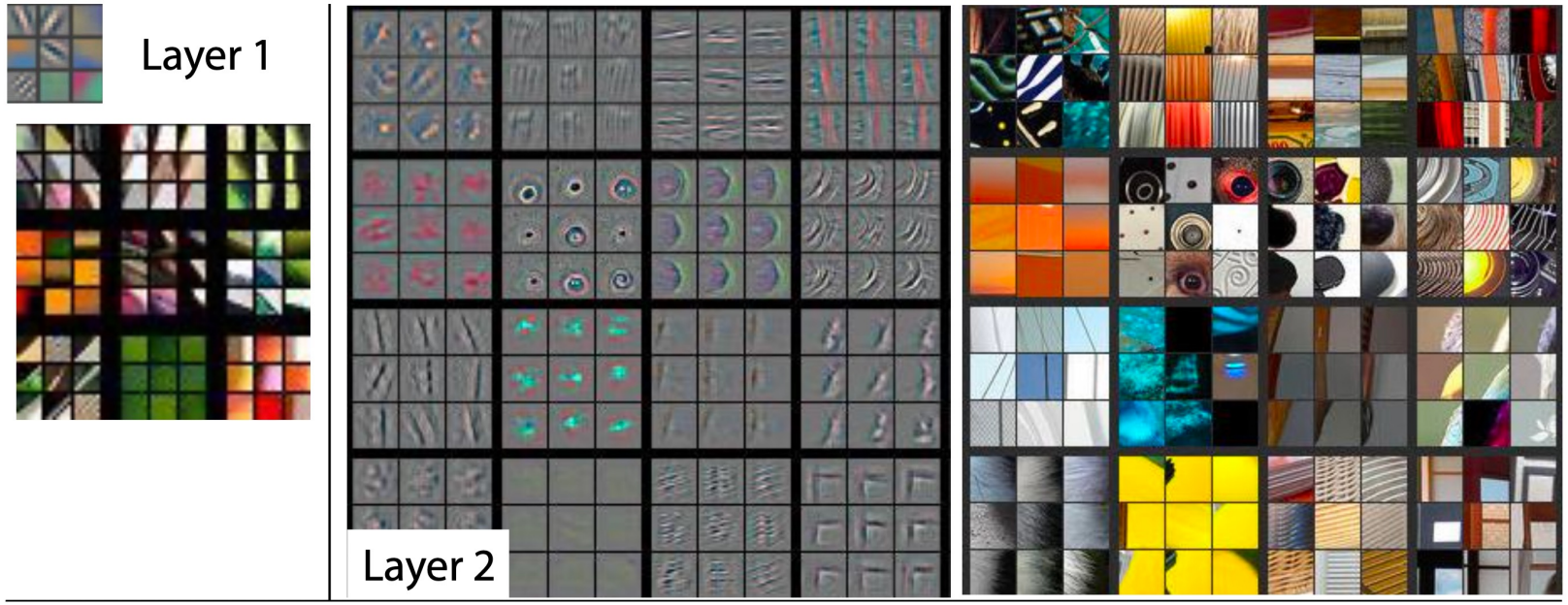
Filter in deeper Layers
(= bigger features)

https://i.stack.imgur.com/5yGWY.png

Example: Edges, Parts and Objects (1/3)

Visualization of features of a fully trained model

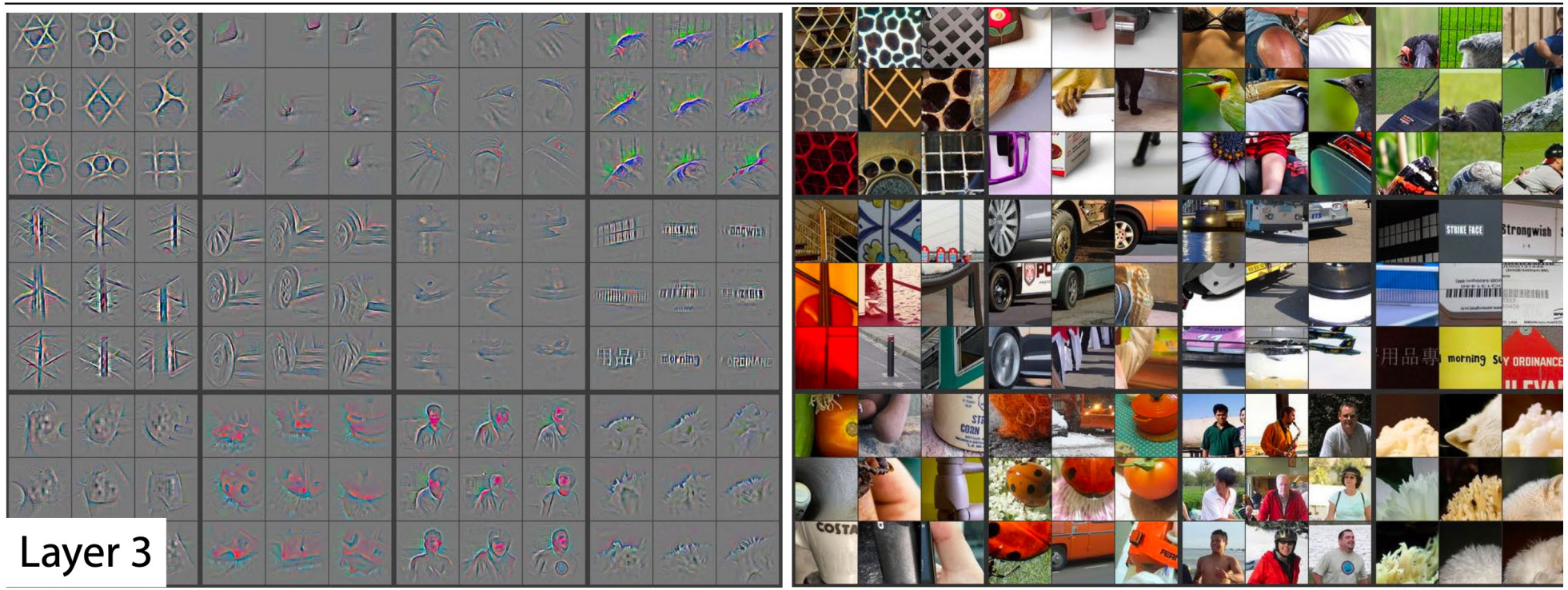
For Layers 2-5, the top 9 activations of some features are shown with their actual image sections.



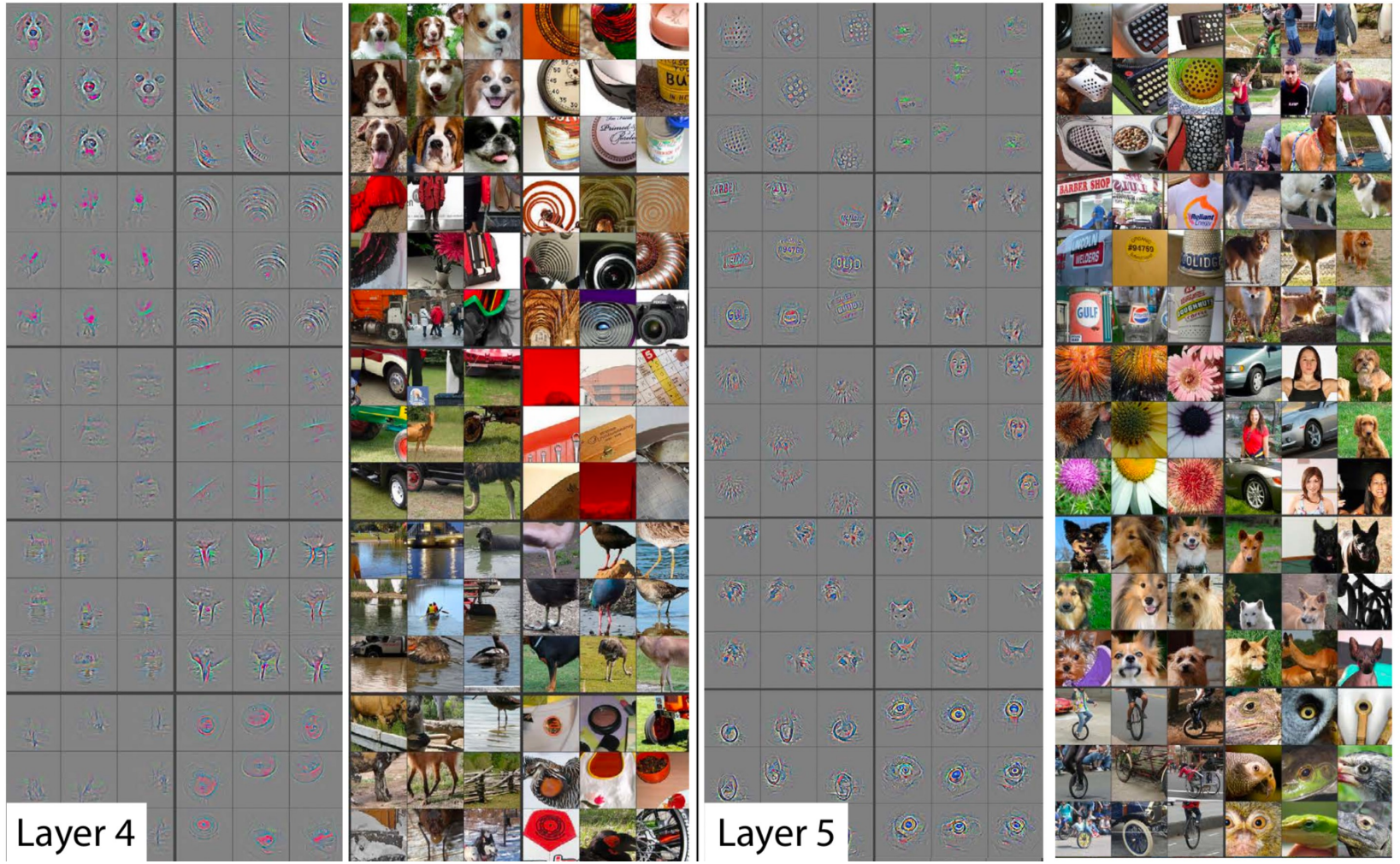
https://arxiv.org/pdf/1311.2901v3.pdf

These image sections are created by mapping the activations back into the input (pixel) space to see which pixels have caused this activation. This was done here with a so-called Deconvolutional Neural Network.

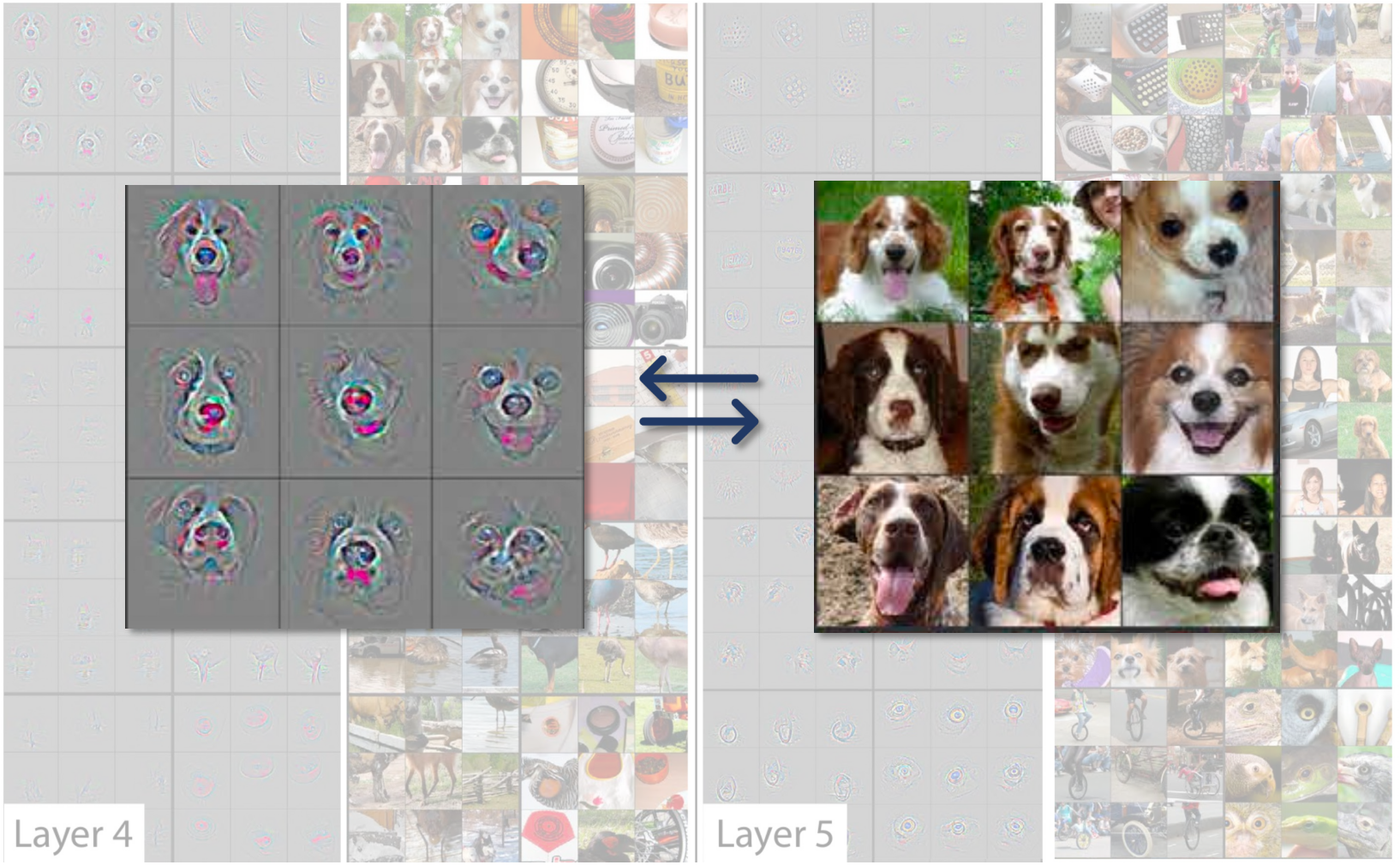
Example: Edges, Parts and Objects (2/3)



Example: Edges, Parts and Objects (3/3)



Example: Edges, Parts and Objects (3/3)



Excursion: Data Augmentation

Advantages of Data Augmentation

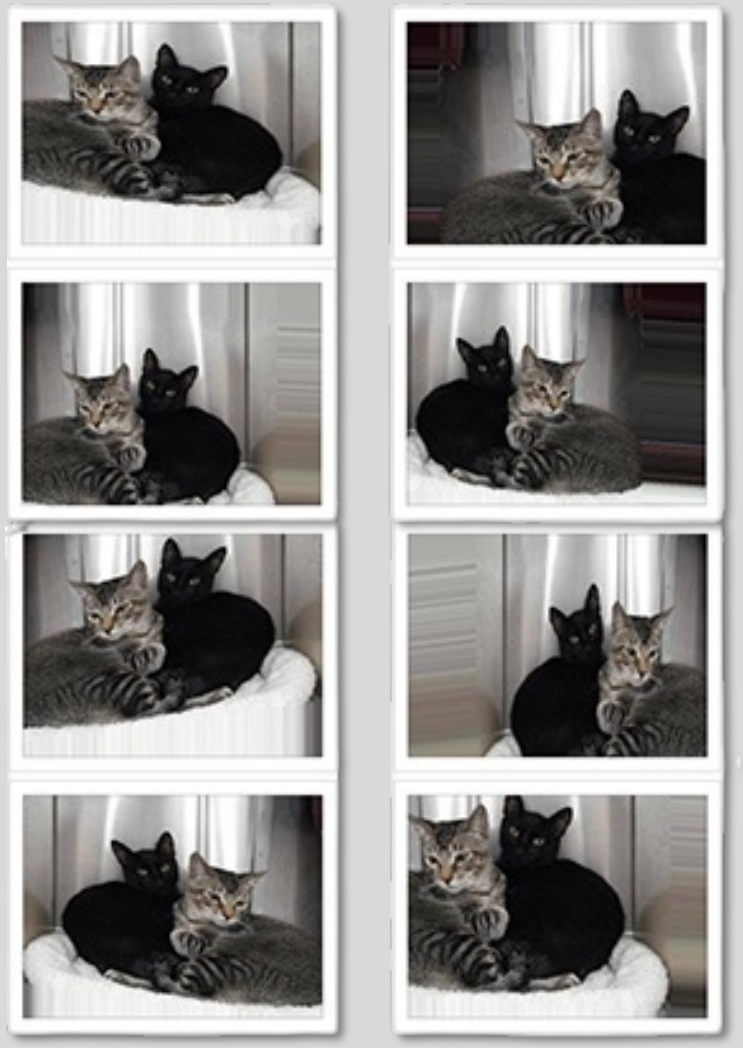
- Creation of more input data for a better trained model
- Recognition of the objects to be identified under different variations

How does Data Augmentation work?

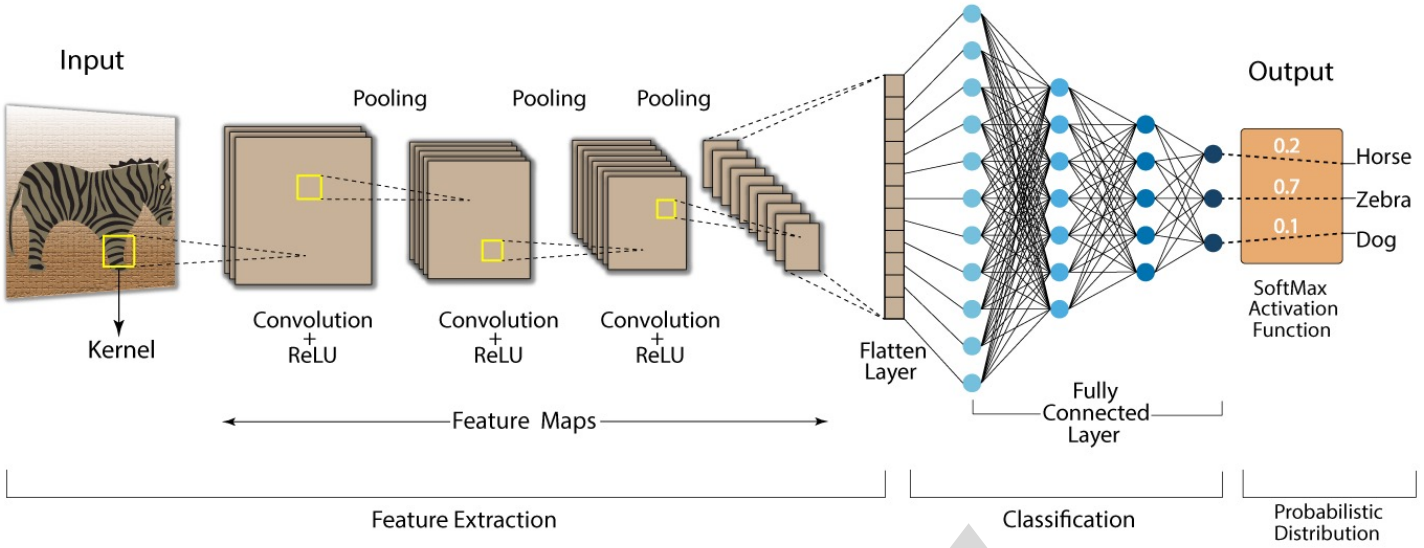
- First the object of desire has to be identiefied in the picture
- Then various tools are applied
  - Rotation
  - Cropping
  - Zoom
  - Horizontal flip
  - Noise
  - Etc.

Why does Data Augmentation work?

- A cat rotated by 15°, is still a cat!



Network architecture of Convolutional Neural Networks



After the desired number of Convolutional and Pooling Layers follow some Fully Connected Layers.

- These layers learn, as seen last week, for example, a classification of the input.
- The previous Convolutional and Pooling Layers are useful for extracting the relevant features that serve as input to the first "normal" Hidden Layer.
- The very last layer eventually produces the output.

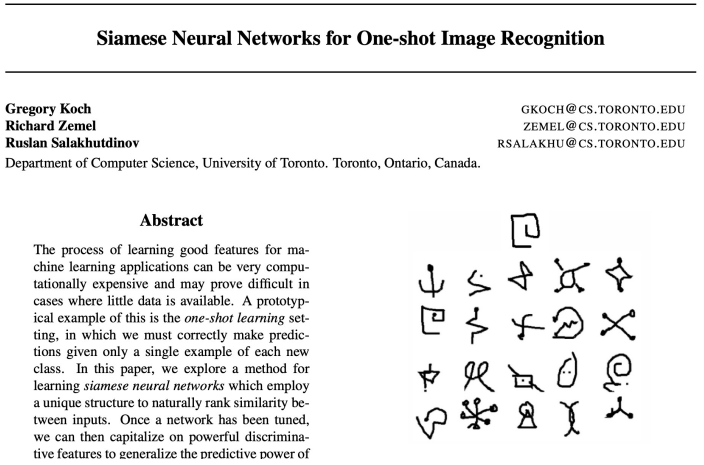
Hyperparameter - Overview

n_h	height of feature map
n_w	width of feature map
n_c	number of channels in the feature map
f	size of filter
s	stride length
d	depth
η	learning rate
μ	momentum
λ	regularization weight
b	batch size

} next week

Outlook for next week

- Application: Neural networks for face recognition
- Theory (working through a modern paper)
- Practice (Implementing the CNN in Tensorflow).



Siamese Neural Networks for One-shot Image Recognition

Gregory Koch GKOCH@CS.TORONTO.EDU
Richard Zemel ZEMEL@CS.TORONTO.EDU
Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Department of Computer Science, University of Toronto. Toronto, Ontario, Canada.

Abstract

The process of learning good features for machine learning applications can be very computationally expensive and may prove difficult in cases where little data is available. A prototypical example of this is the *one-shot learning* setting, in which we must correctly make predictions given only a single example of each new class. In this paper, we explore a method for learning *siamese neural networks* which employ a unique structure to naturally rank similarity between inputs. Once a network has been tuned, we can then capitalize on powerful discriminative features to generalize the predictive power of

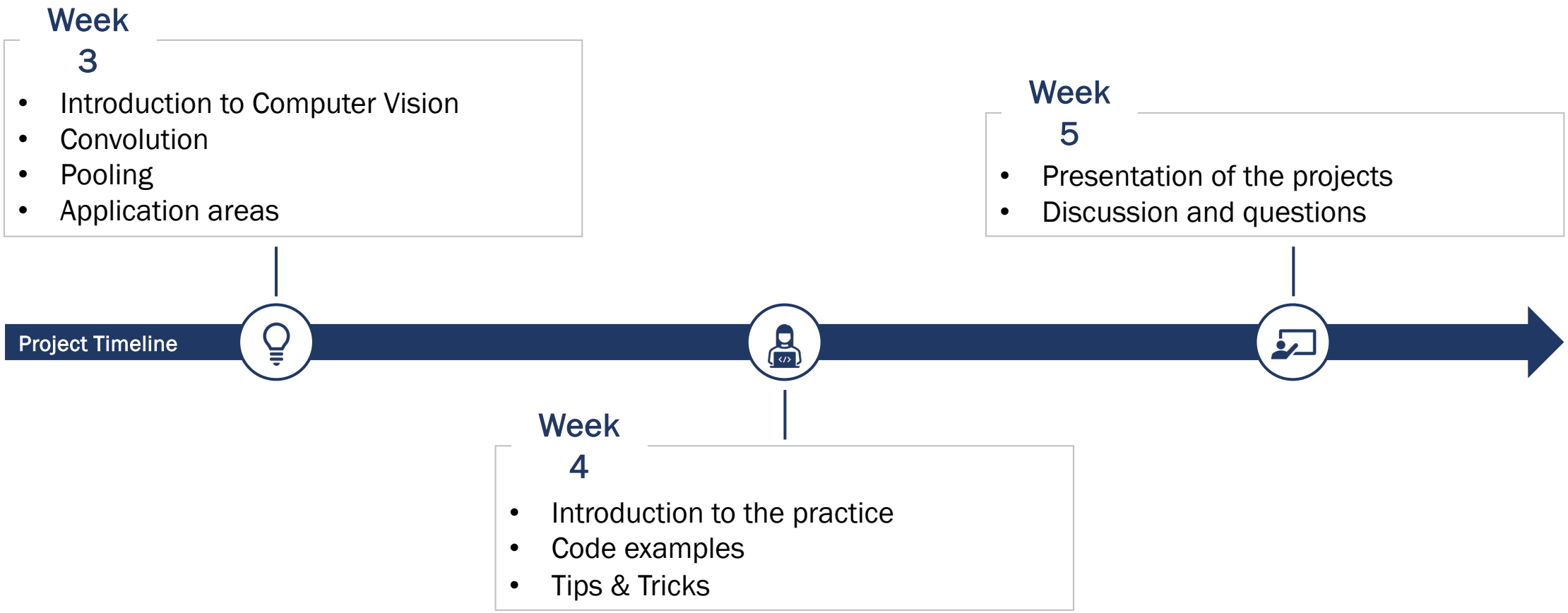
Agenda

Introduction

Theory of CNNs

Programming Project

Process


Week 3
Introduction to Computer Vision
Convolution
Pooling
Application areas
Week 5
Presentation of the projects
Discussion and questions
Project Timeline
Week 4
Introduction to the practice
Code examples
Tips & Tricks

Programming Projects

Details

Who?
Groups of 2-3 Persons

What?
- Working on the provided notebook **or** a self-selected paper, project.
- Understanding instead of copying!
- Presentation of the model and the results
  - 3-8 slides
  - 10 min presentation

When?
- Presentation on November 08, 2023
- Notebook/script, slides by November 06 to
  - mariam.arustashvili@stads.de

For questions at any time contact mariam.arustashvili@stads.de or write in the Teams Channel

Topics

MNIST
- Image classification
- Notebook in the Teams Channel

Handwritten ZIP Codes
- Yann LeCun

http://yann.lecun.com/exdb/publis/pdf/lecun-89e.pdf

fMRI
- fMRI Volume classification
- 3D CNNs

https://www.sciencedirect.com/science/article/pii/S1053811920308144

Your own ideas/interests...