

Deep Learning

Lecture 7

Convolutional Neural Networks

Data Augmentation

Prof. Dr. Rainer Herrler

Phone: 09721/ 940-8710

Email: rainer.herrler@fhws.de

Pictures from Wikipedia / Pixabay

Some Pictures generated with Stable Diffusion

Recap

- Various convenience Stuff
- Regularization
- Optimizers
- Saving and Loading Models

Results on CIFAR100

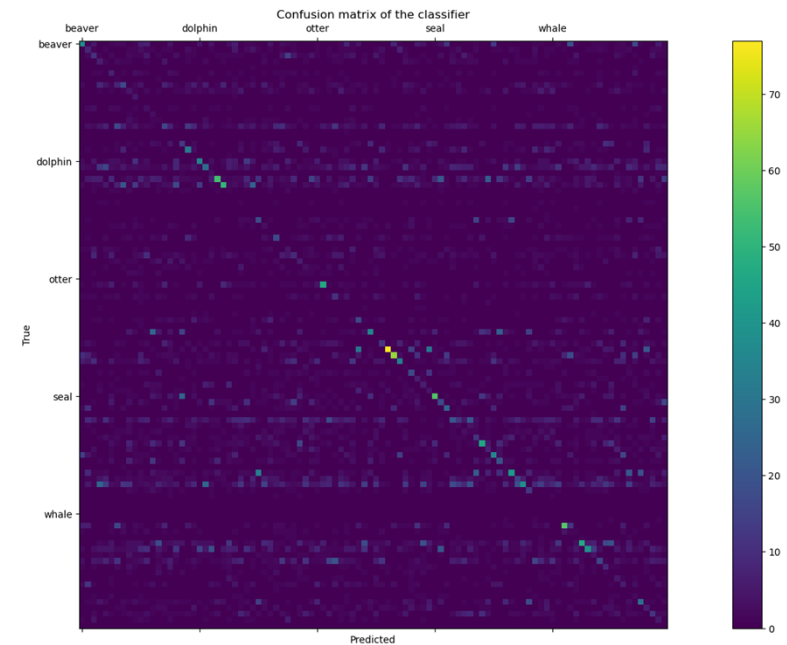
Conclusion:

It's difficult to get high accuracy rates with CIFAR100 because

- Large Input Space (32x32x3)
- Less samples than Fashion-MNIST (50000)
- → Overfits very quickly
- → Regularization necessary
- Hard to optimize / takes a long time, plateaus

We need more techniques to tackle this problem!

Example: [DL_012_A_CIFAR-100-Basic_Layers.ipynb](#)



Limitations of FC-Layers for image classification

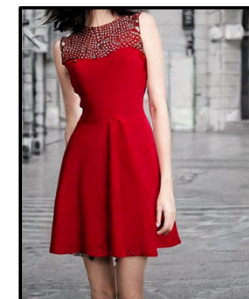
- We were able to see pretty decent results for Fashion-Mnist, but
 - Resolution of the Pictures was very small
 - Position/Size of the objects was relatively similar
- Limitations of Fully connected layers
 - They can in principle learn any structure
 - This doesn't work effectively for higher resolution images
- In a fully connected network no prior structure is considered
 - each pixel is treated independently
 - Pixels are not aware of neighbor pixels
 - Everything has to be learned also special structure



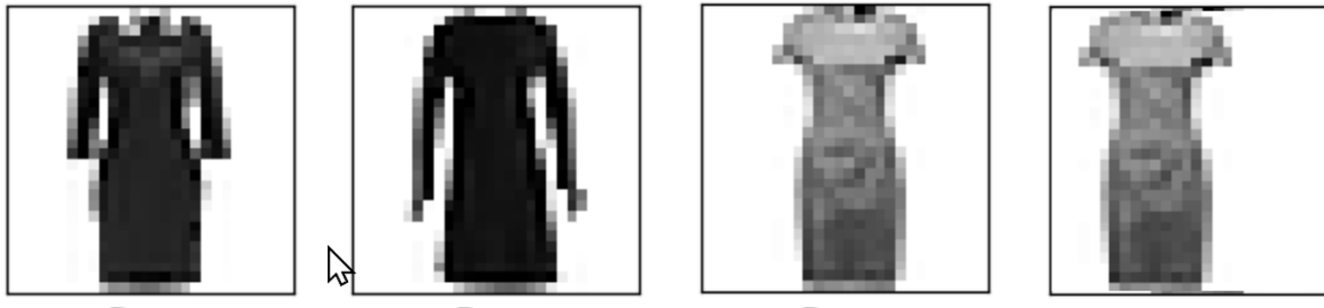
28x28



450x600



It doesn't matter where we find something (patterns should be detected independent of the place)



Like in a puzzle if we detect local structures it helps us to make conclusions on the overall structure



Picture from Pixabay

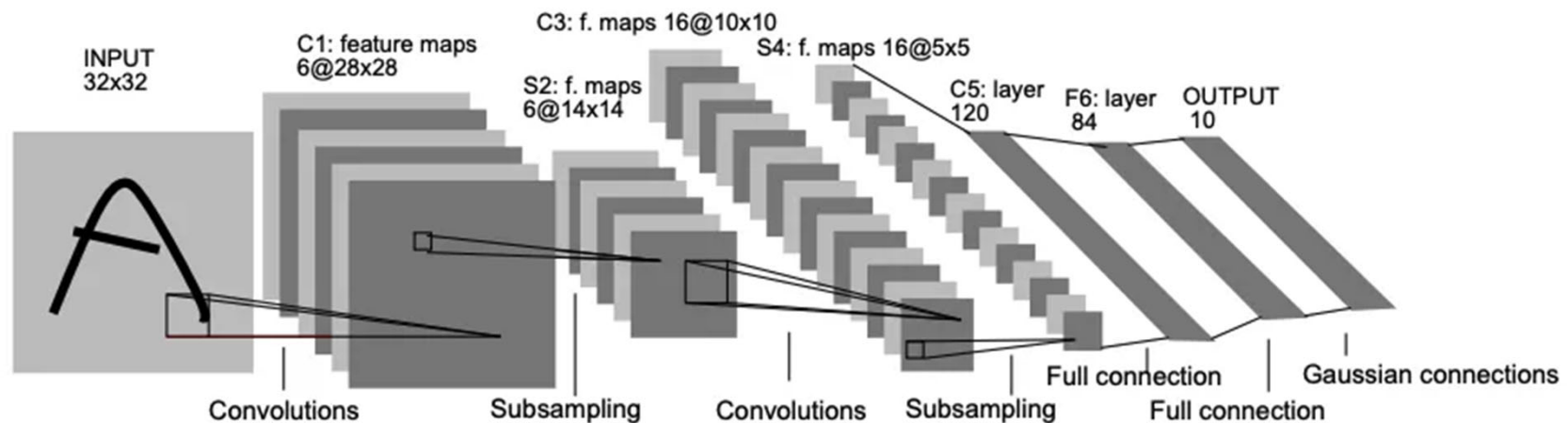
Convolutional Neural networks (CNNs)

Original Paper:

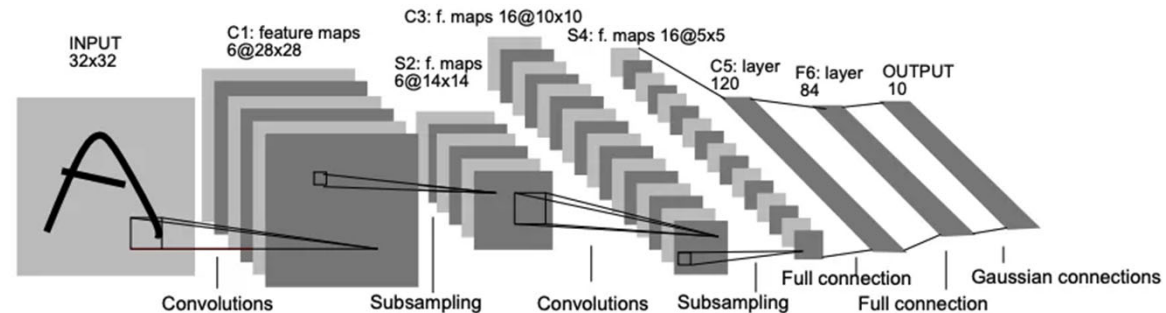
“Gradient-based learning applied to document recognition” *LeCun, Bottou, Bengio, Haffner 1998*



Yann LeCun



Intuitive Idea of CNNs



- We use a small scanner so scan for small Details in the original picture
- The identified “features” are collected in a 2 dimensional structure of lower dimension
- This is done hierarchically
- In the end we have a vector of features that can be fed into a regular Neural Network to determine the label

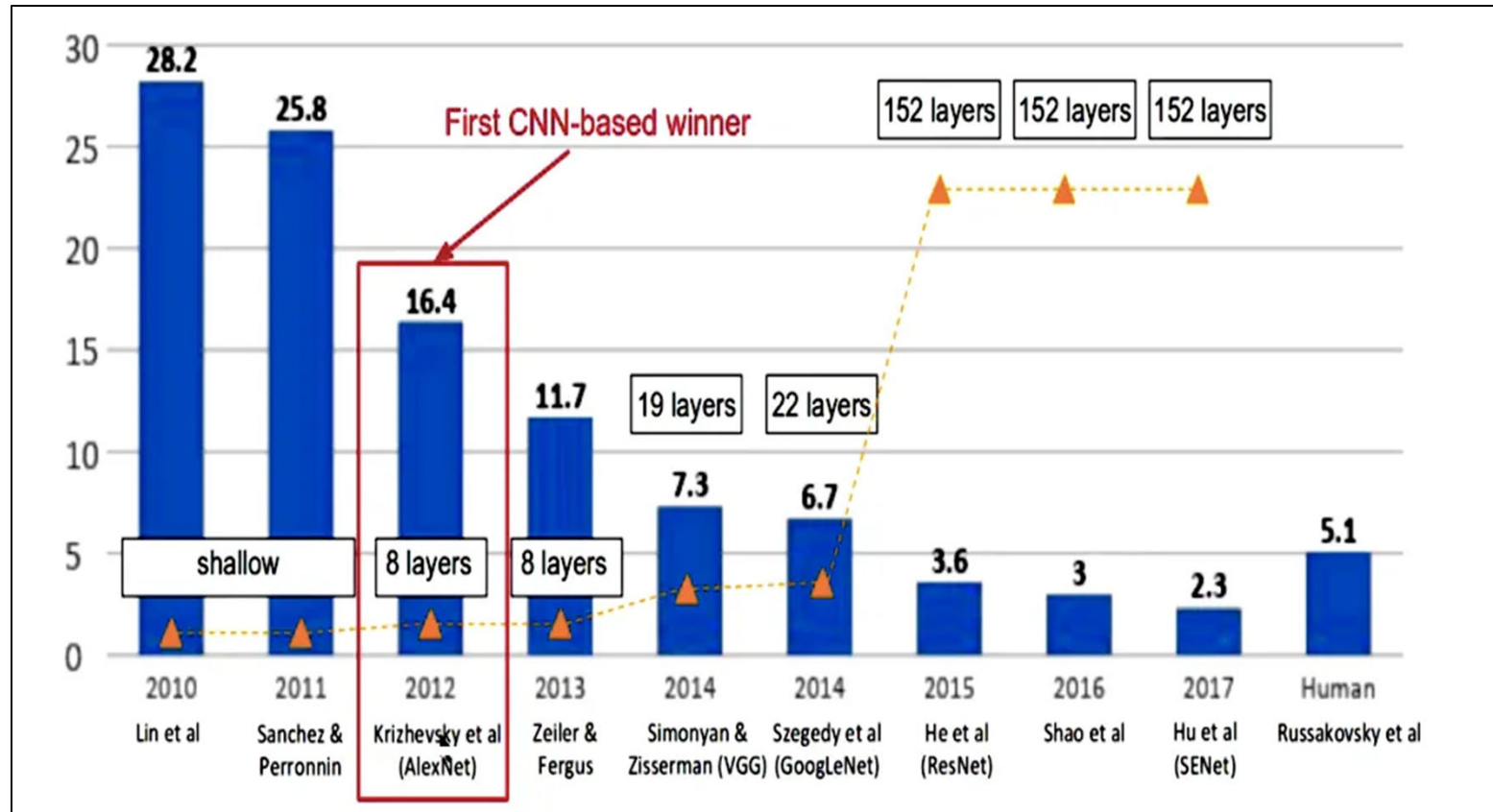
This is best for pictures but also works for other data e.g. Audio, where you have the spatial position of encoded features is of importance.

History of Convolutional Neural networks

- First invented in 1998 they didn't become much attention in the beginning
 - Missing image datasets
 - Limited computational power
- In 2012 the network "Alexnet" won the ImageNet competition with a CNN (first to use ReLU and Dropout)
 - Authors: Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton
- From that time on every year results got better, networks got deeper
- Recommended Article
 - <https://medium.com/appyhigh-technology-blog/convolutional-neural-networks-a-brief-history-of-their-evolution-ee3405568597>

ImageNet Competition (14 Million Pictures)

Error-rate

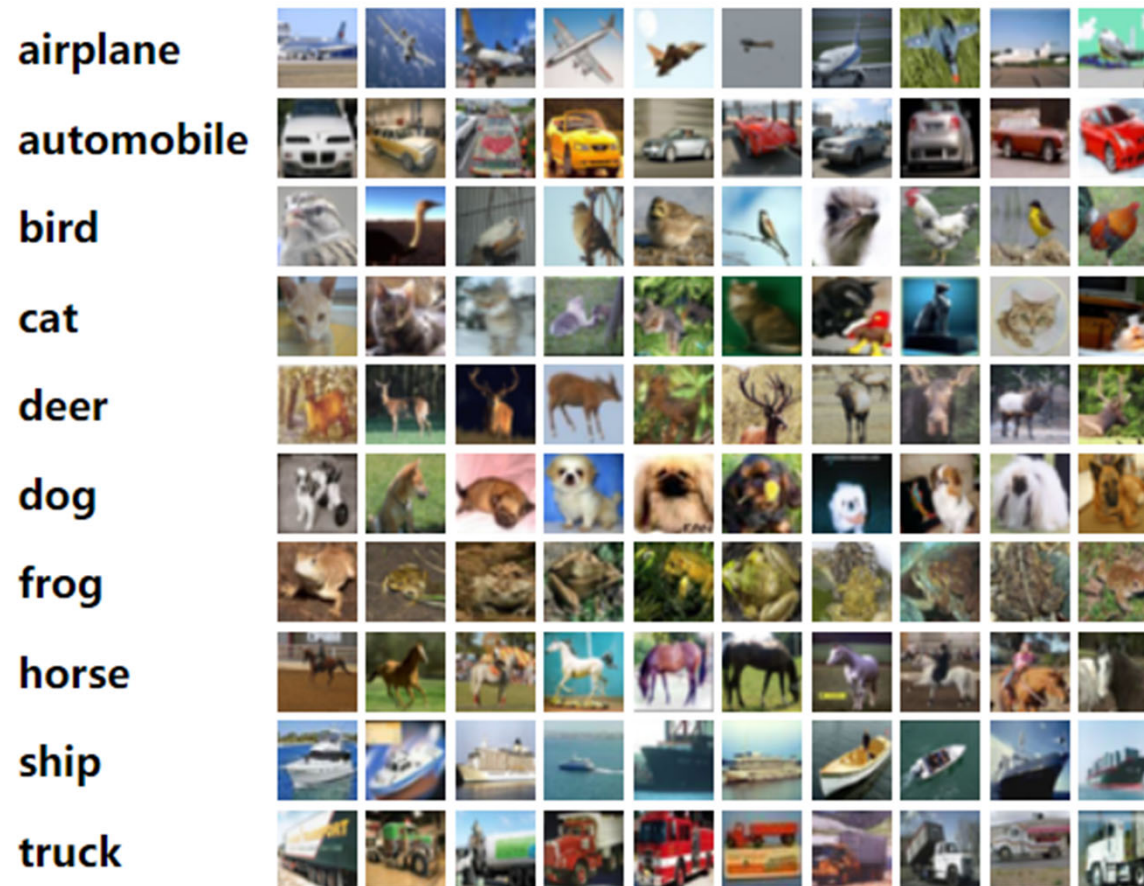


Source: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

Common Image Datasets in Historic order

- Cifar- and MNIST-Datasets
 - Order low-res-Datasets, for education and tests
 - <https://www.kaggle.com/c/cifar-10>
- Pascal VOC “Visual Object Classes” (2005 and later)
 - 20,000 images from 20 different object categories.
- ImageNet (2009)
 - Large dataset with over 14 million images in more than 21,000 categories.
 - helped enable the breakthrough of Deep Learning in image recognition.
- COCO (Common Objects in Context) 2014
 - A dataset with over 330,000 images of 80 different object categories,
- Open Images (2016 and later)
 - A dataset published by Google with over 9 million images used for object detection, recognition, and segmentation.
 - <https://storage.googleapis.com/openimages/web/index.html>

CIFAR 10



Cifar 10

- 10 Classes
- Resolution 32x32x3
- 50000 Training
- 10000 Test

Cifar 100

- 100 Classes
- Resolution 32x32x3
- 50000 Training
- 10000 Test

What is a convolution / cross-correlation - 1D

$$a = [1, 5, 3, 7, 5, 9, 7, 14]$$

$$b = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3} \right]$$

$$a * b =$$

- In a convolution elements of a are elementwise multiplied with b and then added to get a new element of the result. This is done in a sliding window manner
- b is also called a 'kernel' or 'filter'
- Terms come from mathematics and signal processing

Extension to 2D convolutions

Image 8x8

0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1

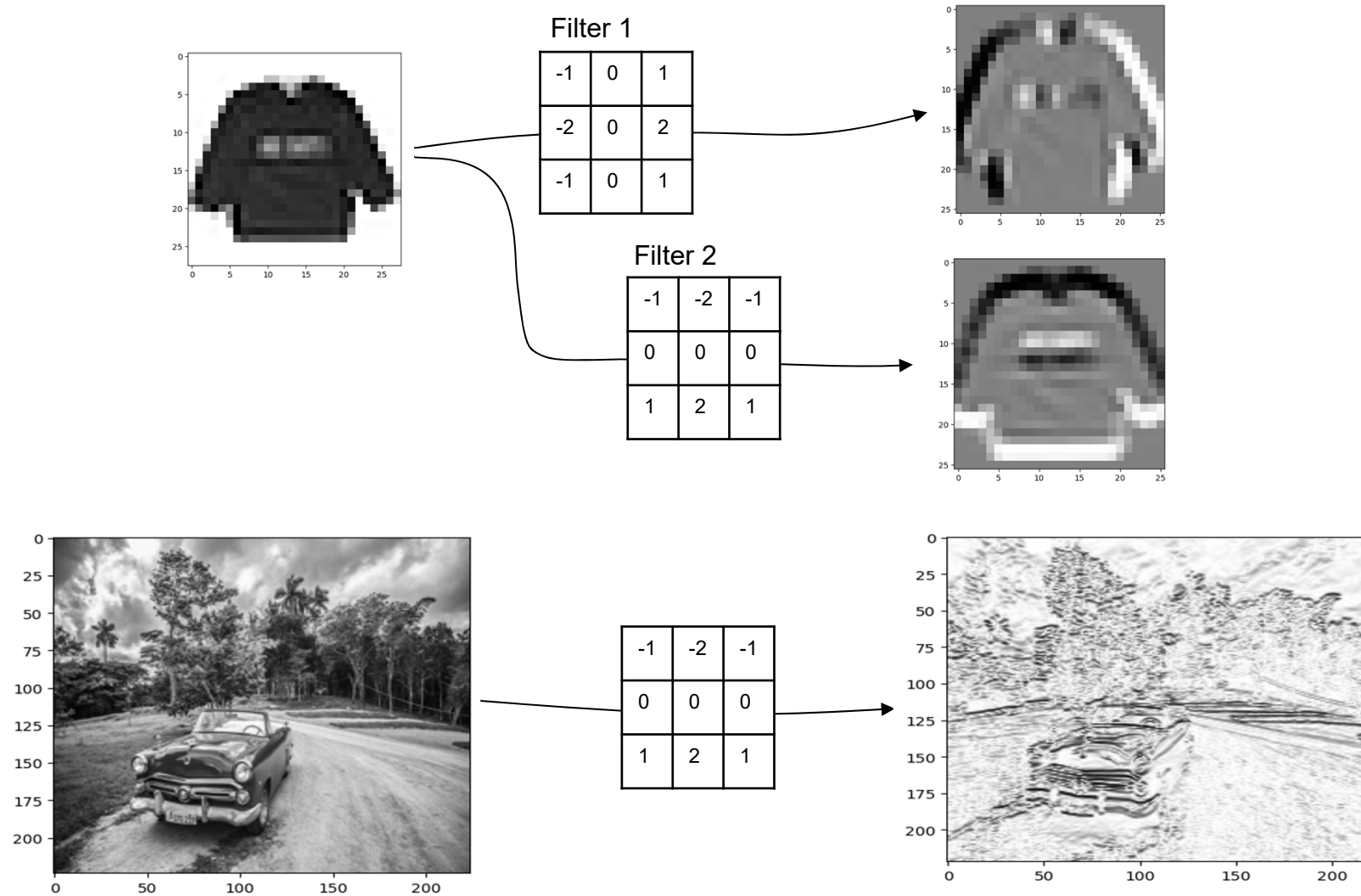
Filter 3x3

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

Result 6x6

0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	0
0	0	4	4	0	1

Sliding window moves from left to right and from top to bottom
For this specific filter we mark edges with high numbers
Result is typically smaller after a convolution

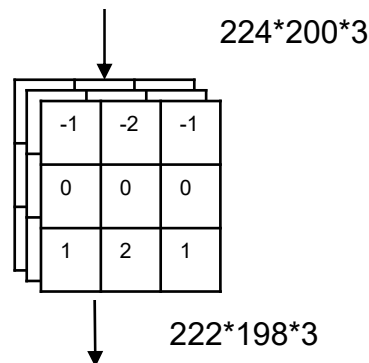


[DL_012_Sobel_Filter_With_Conv2D.ipynb](#)

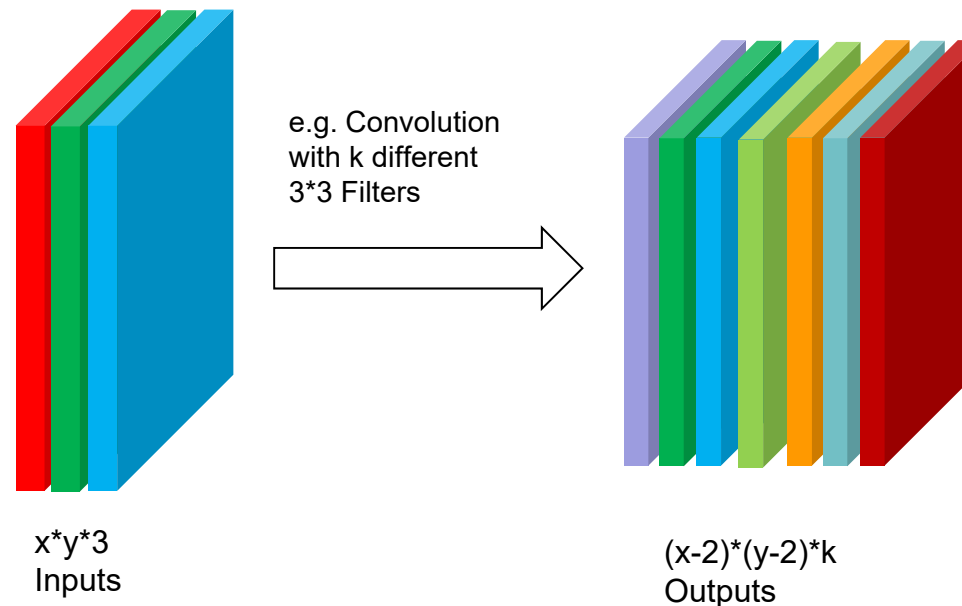
What did we learn in the Jupyter Notebook about Conv2D Layers ?

- A convolution layer can be created in Tensorflow by instantiating `keras.layers.Conv2D`
- A layer can have an arbitrary number of filters aka kernels
- A convolution filter has convolution parameters as well as a bias value
- We also can set an activation function at the layer (in our example sigmoid to keep results in range 0 to 1)
- New Python features:
 - Using the library Pillow (successor of PIL - Python Image Library) can be used to read images from the filesystem, convert size and colorscheme etc
 - Reformating matrixes with `reshape` and `expand_dims`

How do convolutions work with Color Information ?

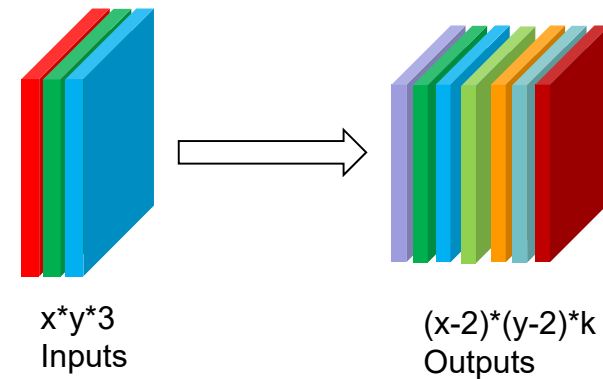


- Color Information is kept in 3 separate channels (R, G, B)
- We can use 3-Dimensional kernels to do the convolution for each channel



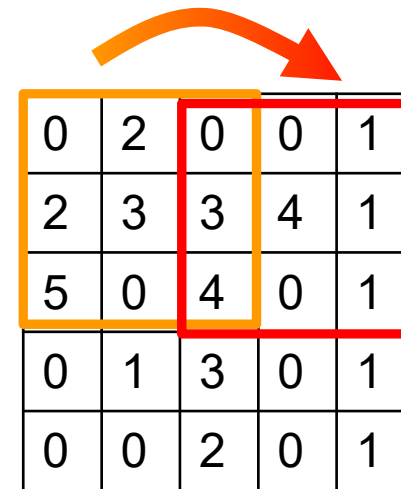
Training convolutional layers

- Kernels in Convolutional Layers are randomly initialized
- Like weights in FC-Layers they are updated by backpropagation to minimize the loss
- Each kernel can learn to identify different “features” like
 - Horizontal/ vertical lines
 - Curves, Color Patterns
- Local structures in the image are leading to the same result independent on where in the image they are
- Efficiency: Compared to fully connected layers they have far less weights to update



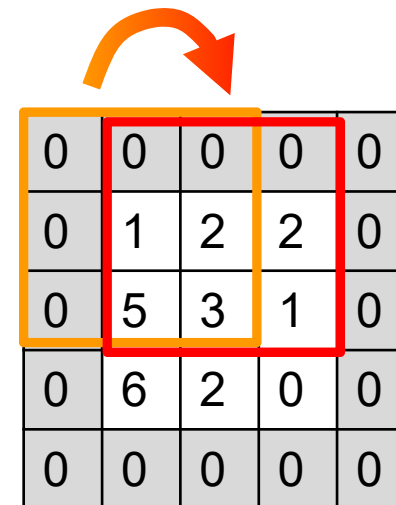
Stride and Padding

- Stride
 - Instead of shifting the tile by a step of one pixel, we can also configure a higher value
 - Stride can be different in different directions
 - Higher Stride reduces the size of the output (downsampling effect)
- Padding
 - Usually convolutions make the image smaller
 - Kernels cannot catch properties at the corners



0	2	0	0	1
2	3	3	4	1
5	0	4	0	1
0	1	3	0	1
0	0	2	0	1

Stride = 2



0	0	0	0	0
0	1	2	2	0
0	5	3	1	0
0	6	2	0	0
0	0	0	0	0

Zero-
Padding

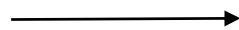
Questions

- Which dimension has the output tensor if we put a batch of 5 RGB-images with a resolution of 30x30 pixels through a convolutional layer with four 5x5 kernels (no padding, stride=1)
- Same as above now with stride of 3 and padding.
- How many parameters has a Conv2D Layer for RGB-images with 100x100 pixels and ten 3x3-Filters

Pooling Layers

- Problem
 - Convolutions with multiple kernels make the output dimension larger from layer to layer
 - Also we want to have a transition from local features to higher level features (classifying what can be seen in larger part of the icture)

0	0	1	3
0	2	2	0
1	0	5	1
0	0	0	0

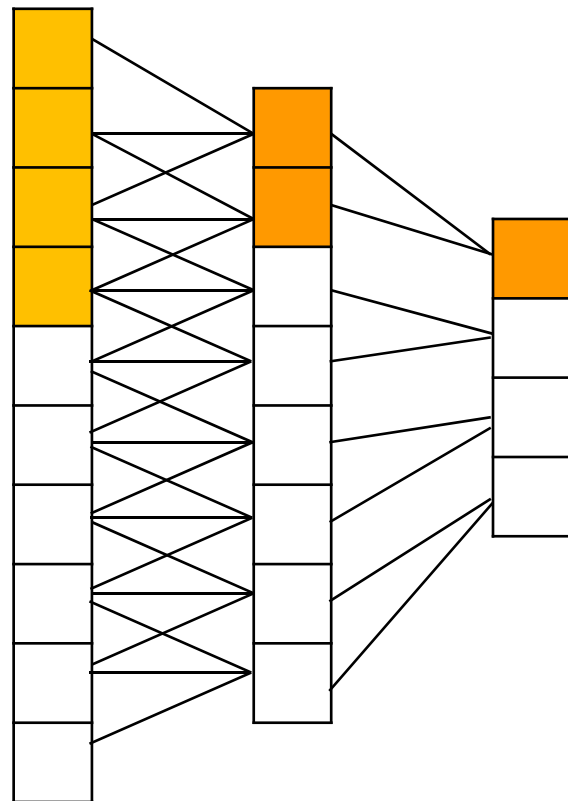


2	3
1	5

MaxPooling - Layer

- Effects
 - The image gets downsampled, the dominant activation remains
 - Mean is shifted

Receptive field extension by stacking layers



Convolution
Size 3
Stride 1

Maxpooling
Size 2

- The receptive field of a neuron is the size of the pixels that can influence the
- By stacking Convolutions and Padding the receptive field of later layers is growing

CNN Layers in Keras

```
tf.keras.layers.Conv2D(  
    filters,  
    kernel_size,  
    strides=(1, 1),  
    padding='valid',  
    data_format=None,  
    dilation_rate=(1, 1),  
    groups=1,  
    activation=None,  
    use_bias=True,  
    kernel_initializer='glorot_uniform',  
    bias_initializer='zeros',  
    kernel_regularizer=None,  
    bias_regularizer=None,  
    activity_regularizer=None,  
    kernel_constraint=None,  
    bias_constraint=None,  
    **kwargs  
)
```

```
tf.keras.layers.MaxPool2D(  
    pool_size=(2, 2),  
    strides=None,  
    padding='valid',  
    data_format=None,  
    **kwargs  
)
```

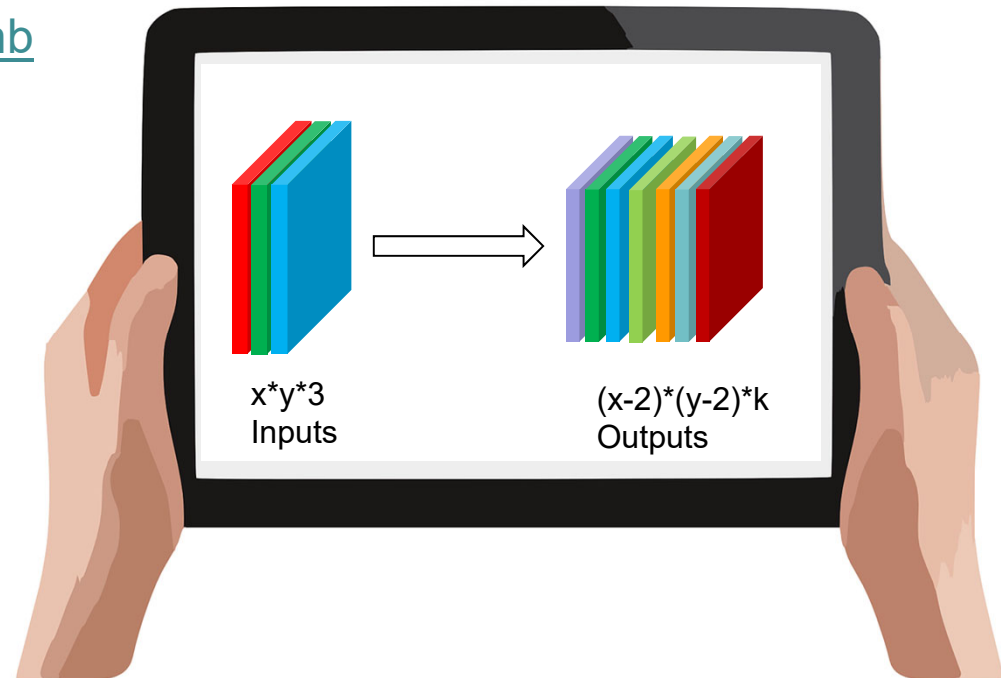
Comments:

- Versions also available for 1D and 3D
- Aliases available MaxPooling2D

A First CNN for Fashion MNIST

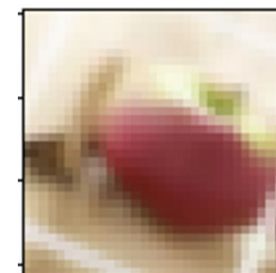
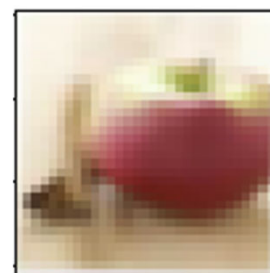
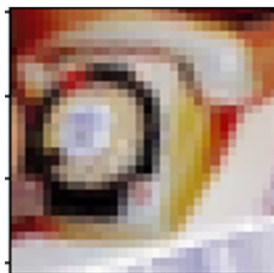
[DL_013_FashionMNistCNN.ipynb](#)

With 1st small Exercise



Data Augmentation

Extend your training data



Data Augmentation

- **Definition:** Data Augmentation creates more samples by slight modifications that usually don't have an effect on the predicted class
- Example Modifications for Images
 - Shifting, Scaling, Rotation
 - Adjusting Contrast, Saturation, Brightness
 - Adding noise
- Example Modification for Audio
 - Add Pitch
 - ...
- <https://www.datacamp.com/tutorial/complete-guide-data-augmentation>

Data Augmentation with Tensorflow

- Option 1: Using special Layers in the model
 - RandomFlip
 - RandomRotation
 - With every epoch a randomly modified sample is being shown
 - We can create an extra Network for augmentation
 - These layers apply the random modification just in training
 - https://keras.io/api/layers/preprocessing_layers/image_augmentation/
- Option 2: Using the ImageDataGenerator
 - Here augmentation is not part of the model itself, but the data preparation
 - https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator
[or](#)
- Option 3: Use Keras-Function to read from Directory
 - Augmentation can be done while “streaming” data from the filesystem
 - `tf.keras.preprocessing.image_dataset_from_directory`
 - <https://keras.io/api/preprocessing/image/>

Data Augmentation Notebook

[DL_014_DataAugmentation.ipynb](#)



What did we learn from the last Notebook

- Data Augmentation helps to fight overfitting
 - **accuracy: 0.9400** - val_accuracy: **0.2145** (without augmentation)
 - **accuracy: 0.3980** - val_accuracy: **0.3218** (with augmentation)
- Sequential-Networks can be nested
 - Composition of networks can be done
 - Take care that input and output layers fit together and that input_size for first layer is given
- We have written a own training-Loop for several epochs
 - Often this is not necessary but it gives you greater flexibility

2nd larger Exercise



Task: Try to create a CNN for CIFAR 100

- Difficulties
 - Many classes and few samples
 - Complex networks will overfit very fast
 - Regularization is needed!
- Realistic outcome on regular laptops:
 - Rates around >35 % accuracy should be easily achievable
 - More complex networks (with current means) can reach 60% accuracy and more

[DL_015_CIFAR-100-CNNs.ipynb](#)