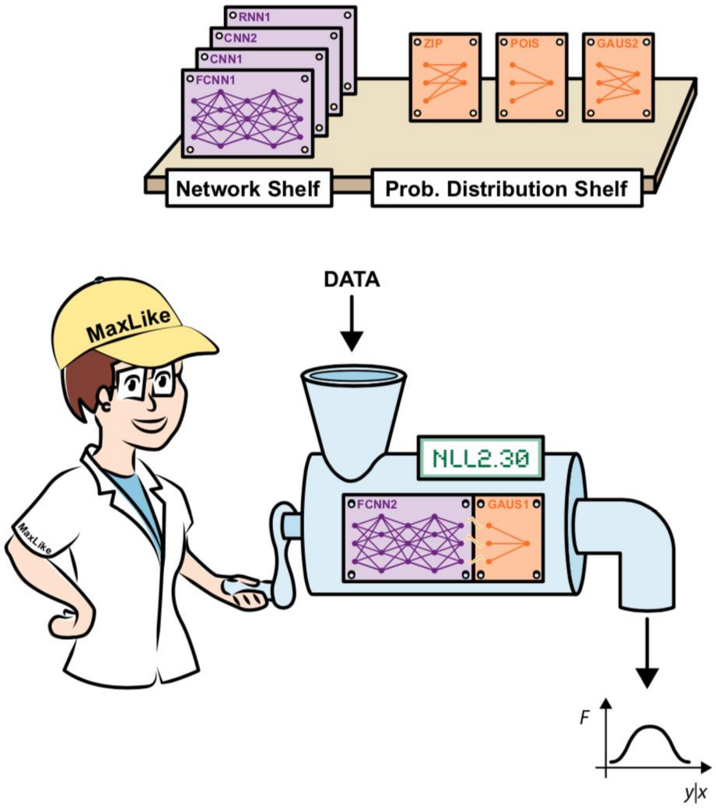
WBL Deep Learning:: Part 2

Beate Sick, Oliver Dürr

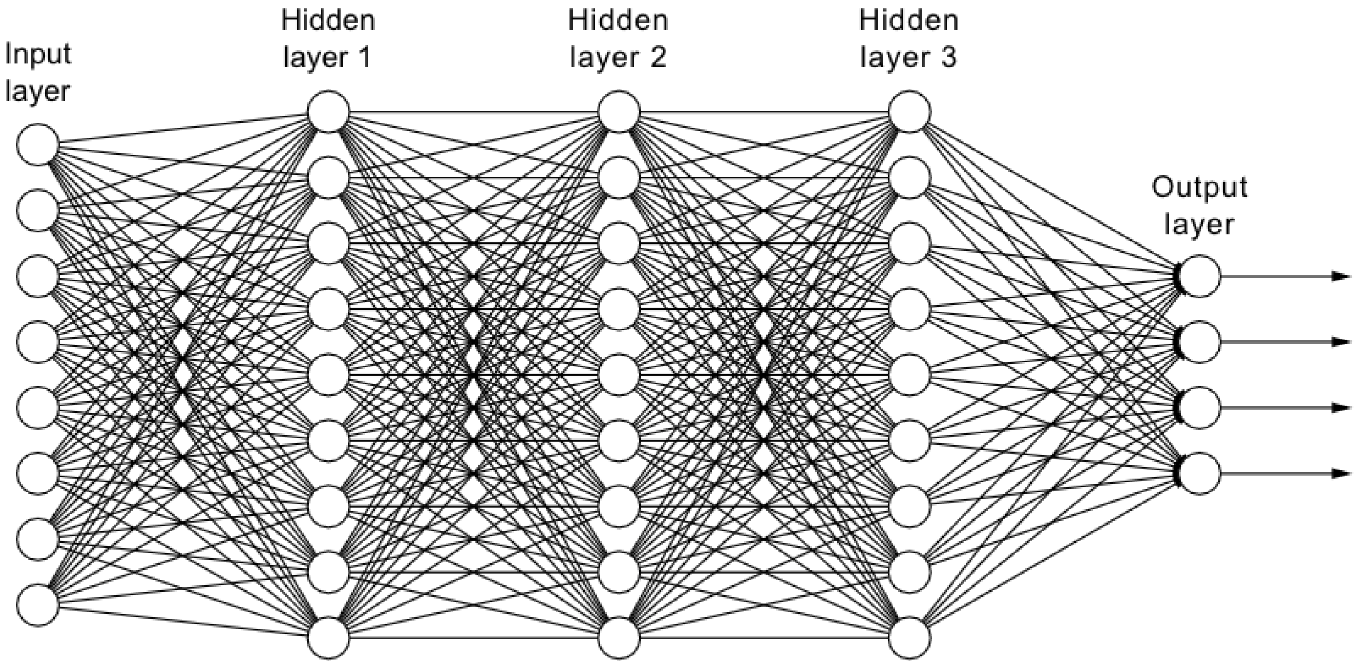
Convolutional Neural Networks

Topics of today

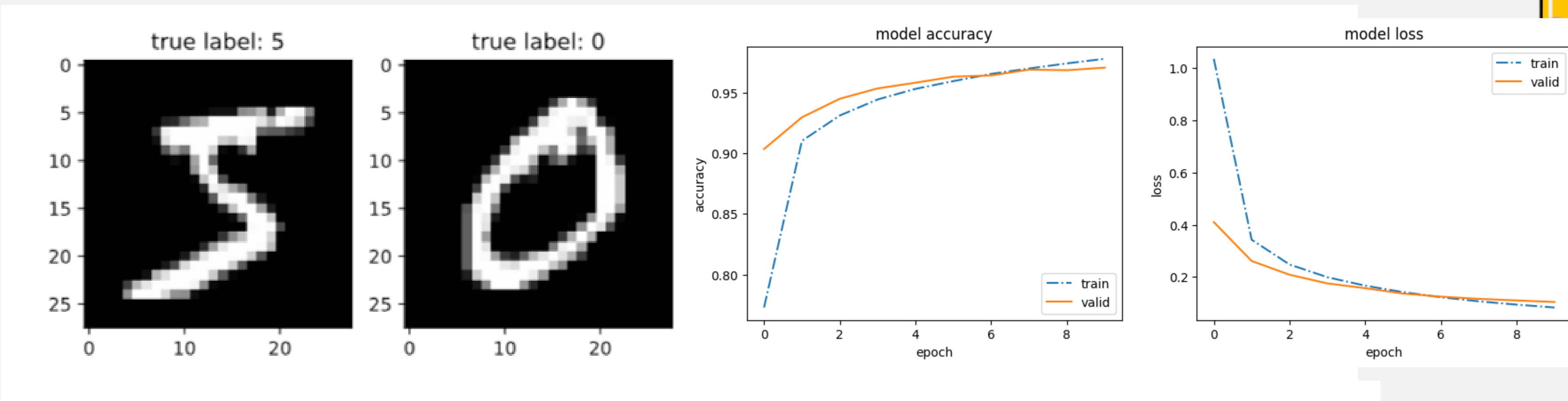
- Convolutional Neural Networks (CNN) for images
  - Motivation of CNN
    - Use local correlation structure of images
  - How das convolution work?
    - Kernel and filter with shared weights
    - Each kernel yields one activation map
  - How das pooling work?
  - Architecture of CNNs

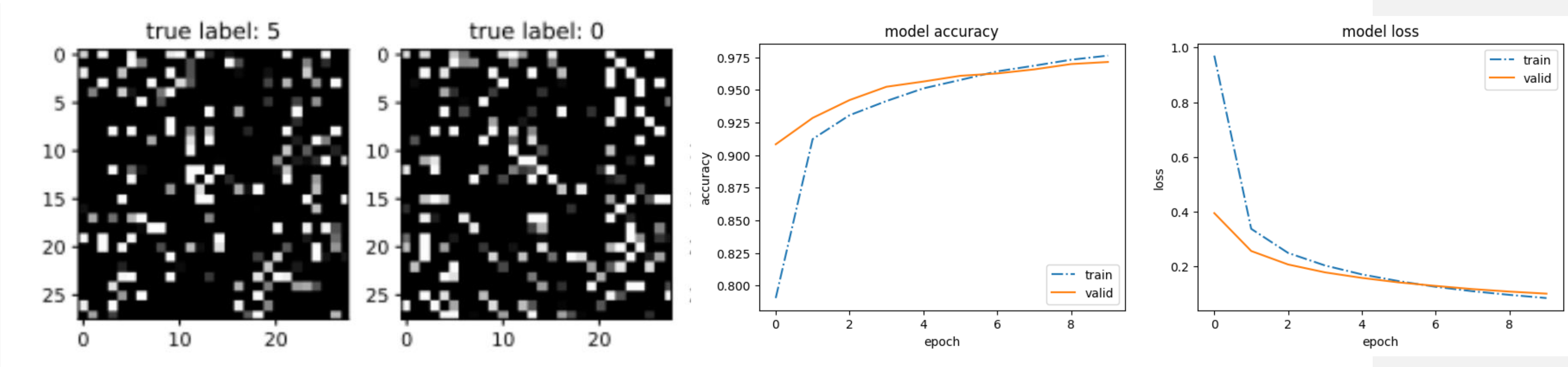


Recall: Architecture of a fully connected NN



MNIST exercise: Does shuffling disturb a fcNN?

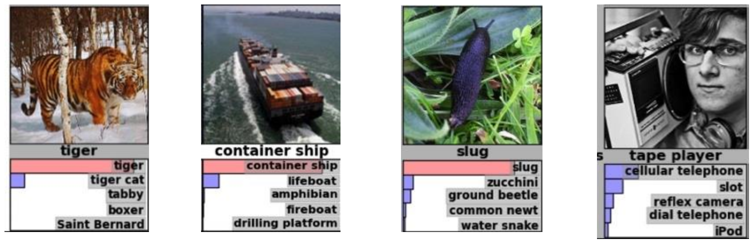




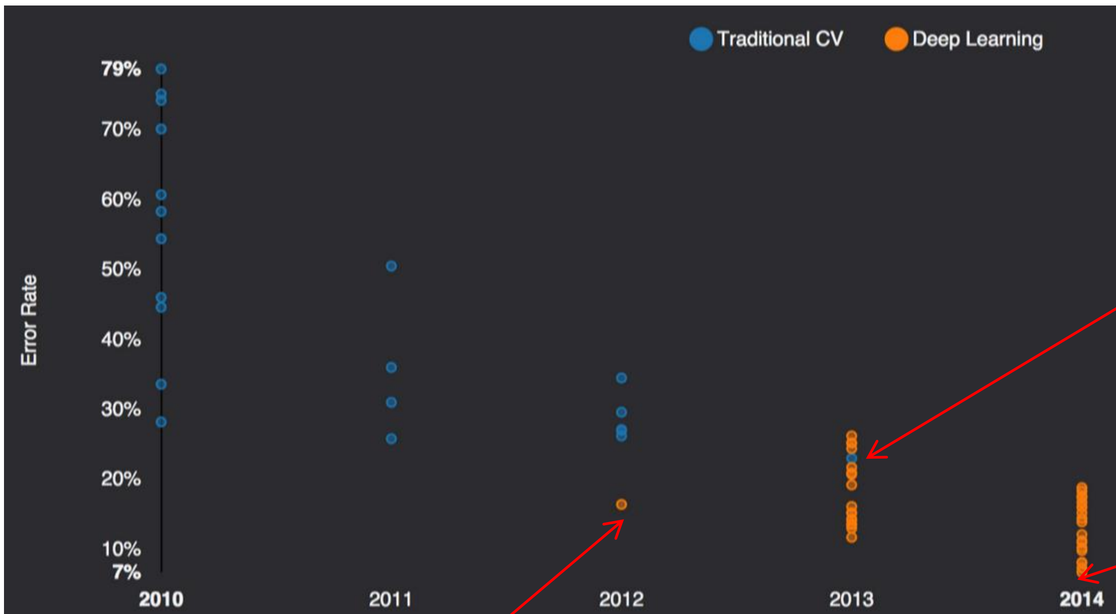
→ The performance of a fcNN is the same on original and shuffled images

Recall: Imagenet challenge

1000 classes
1 Mio samples



…



Human: 5% misclassification

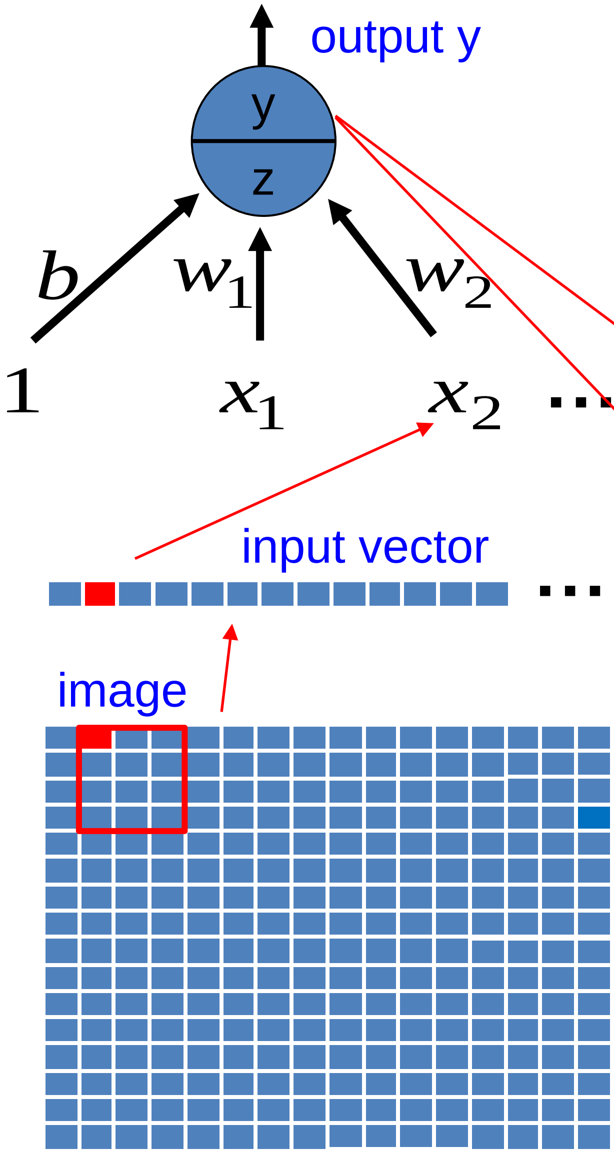
Only one non-CNN approach in 2013

GoogLeNet 6.7%

A. Krizhevsky
first CNN in 2012
Und es hat zoom gemacht

Figure: https://medium.com/global-silicon-valley/machine-learning-yesterday-today-tomorrow-3d3023c7b519

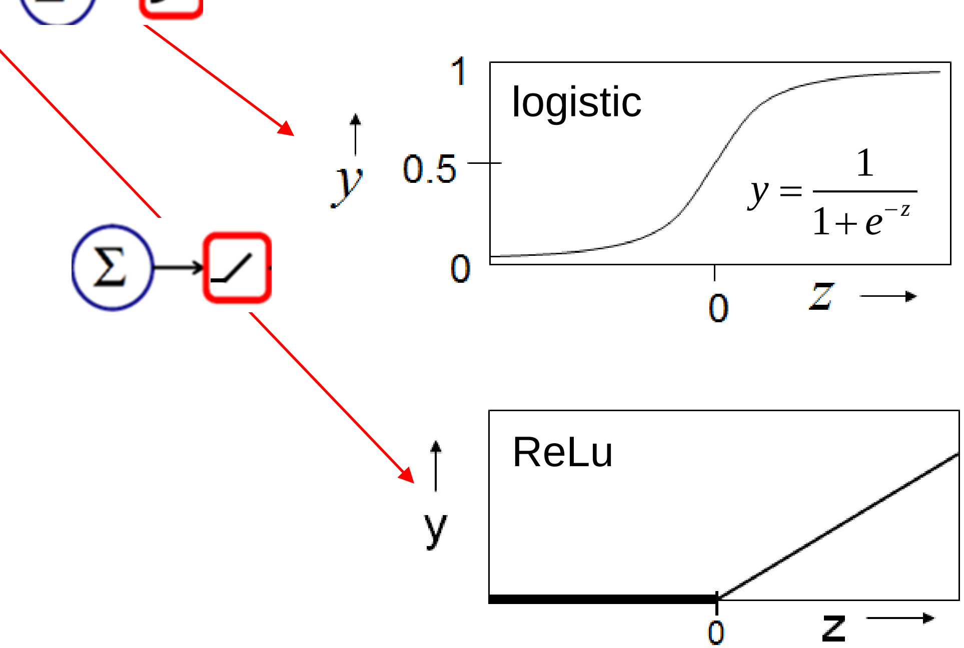
An artificial neuron



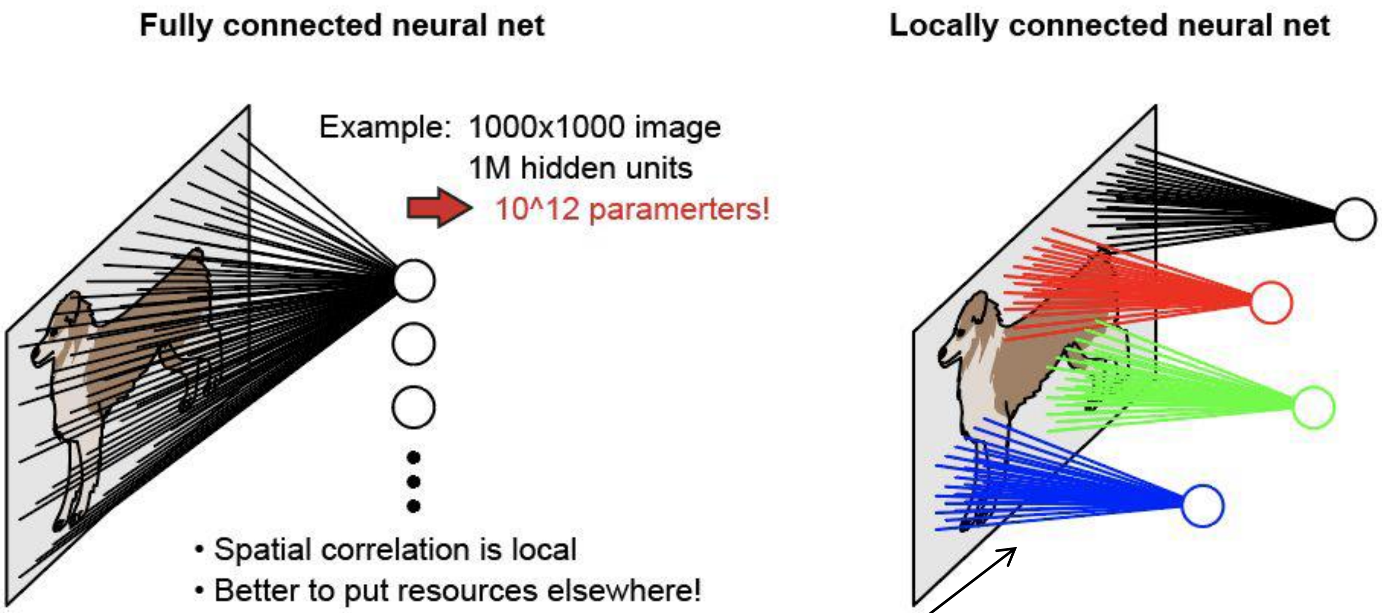
bias weights

$$z = b + \sum_i x_i w_i$$

Different non-linear transformations are used to get from z to output y



Convolution extracts local information using few weights



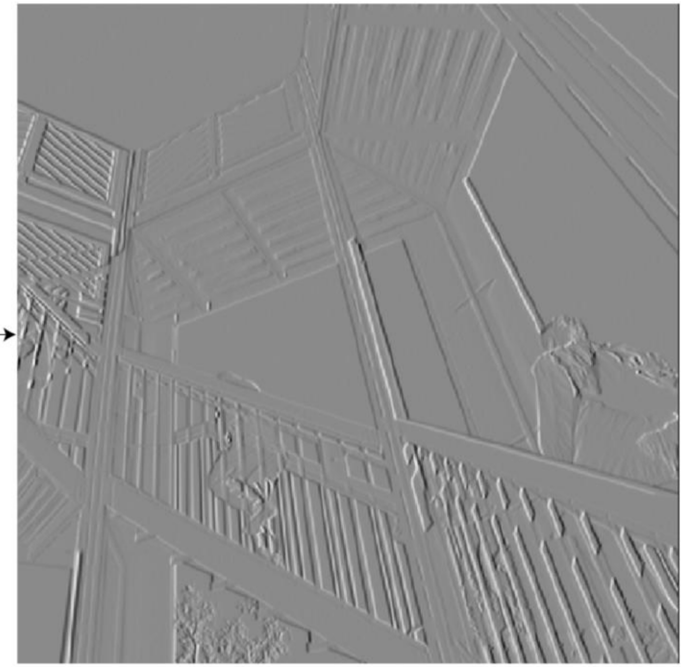
Shared weights:

by using the same weights for each patch of the image we need much less parameters than in the fully connected NN and get from each patch the same kind of local feature information such as the presence of an edge.

Example of designed Kernel / Filter

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

Horizontal Sobel kernel



Applying a vertical edge detector kernel

Taken from the great website on convolution: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolution

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

Input X

1	0	1
0	1	0
1	0	1

Kernel W

4	3	4
2	4	3
2	3	4

Result Z

Convolution (let's ignore bias b):

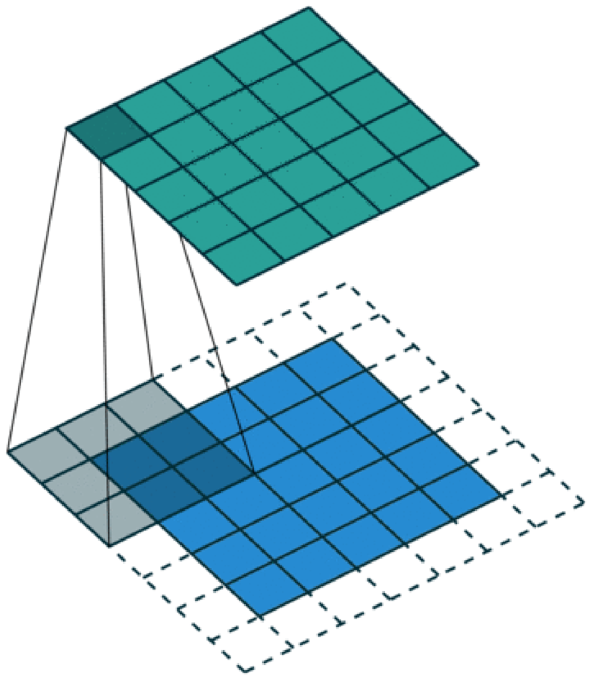
$$z = b + \sum_i x_i w_i$$

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

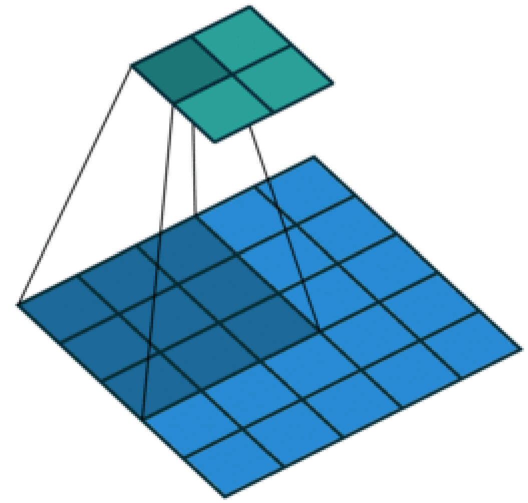
4		

https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Ingredient I: Convolution



Zero-padding to achieve
same size of feature and input



no padding to only use
valid input information

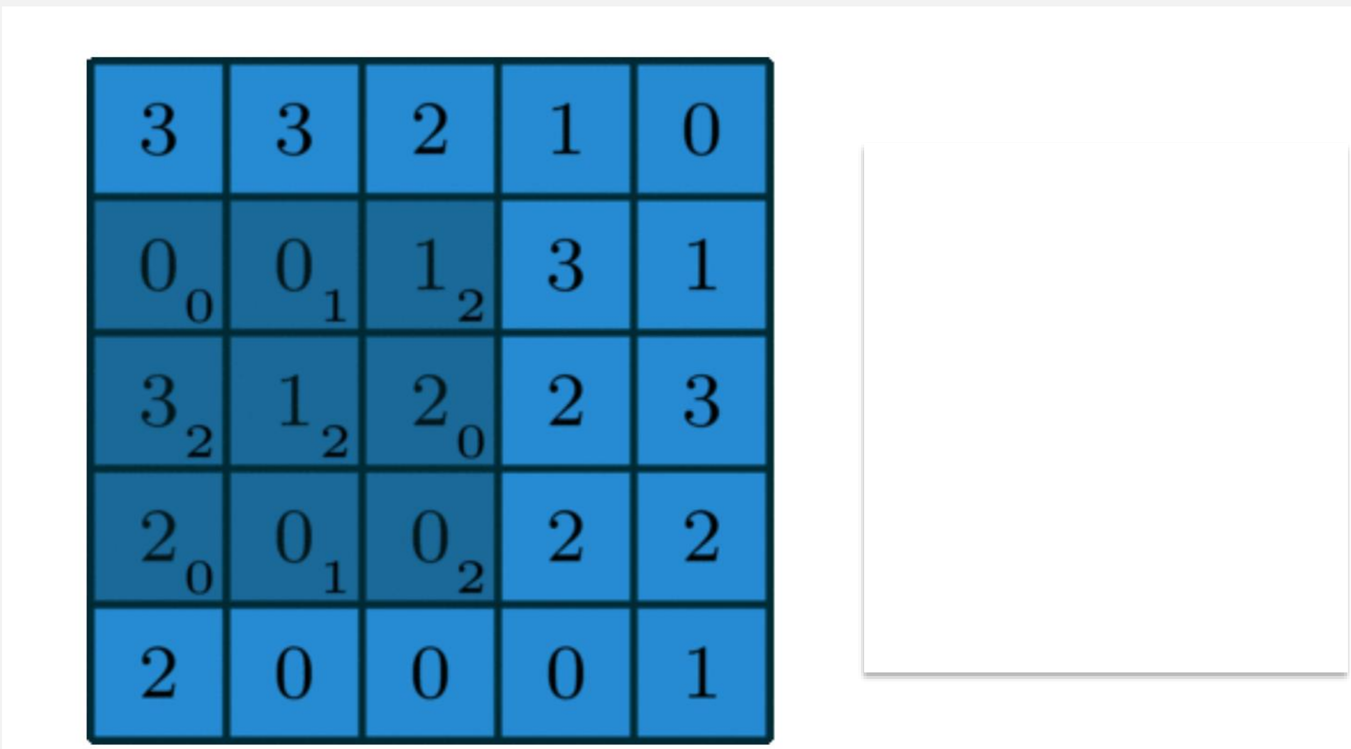
The *same* weights are used at each position of the input image.

Taken from the great website on convolution: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Exercise: Do one convolution step by hand

The kernel is 3x3 and is applied at each valid positon
– how large is the resulting activation map?

The small numbers in the shaded region are the kernel weights.
Determine the position and the value within the resulting activation map.

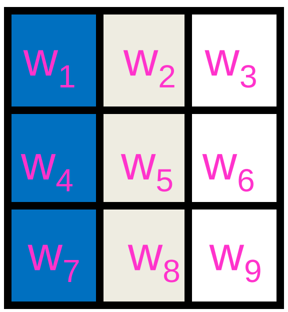


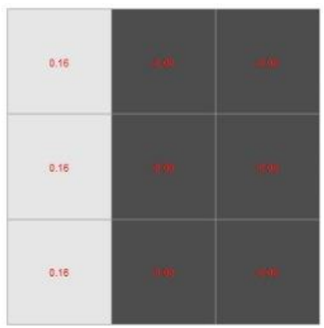
Convolutional networks use neighborhood information and replicated local feature extraction

In a locally connected network the calculation rule

$$z = b + \sum_i x_i w_i$$

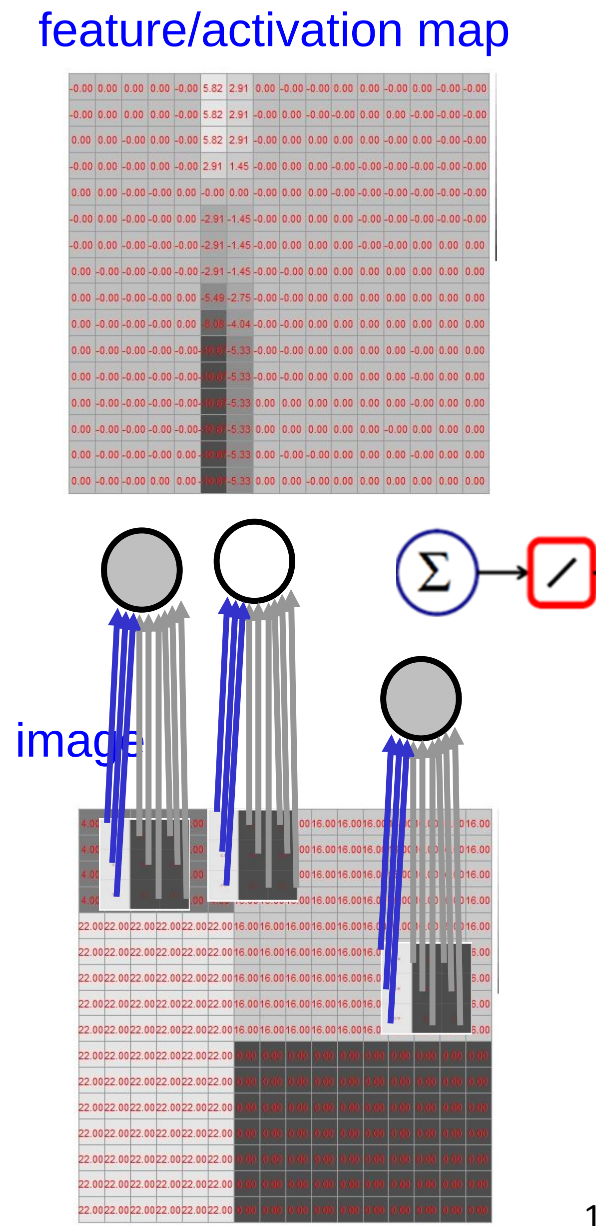
Pixel values in a small image patch are element-wise multilied with weights of a small filter/kernel:



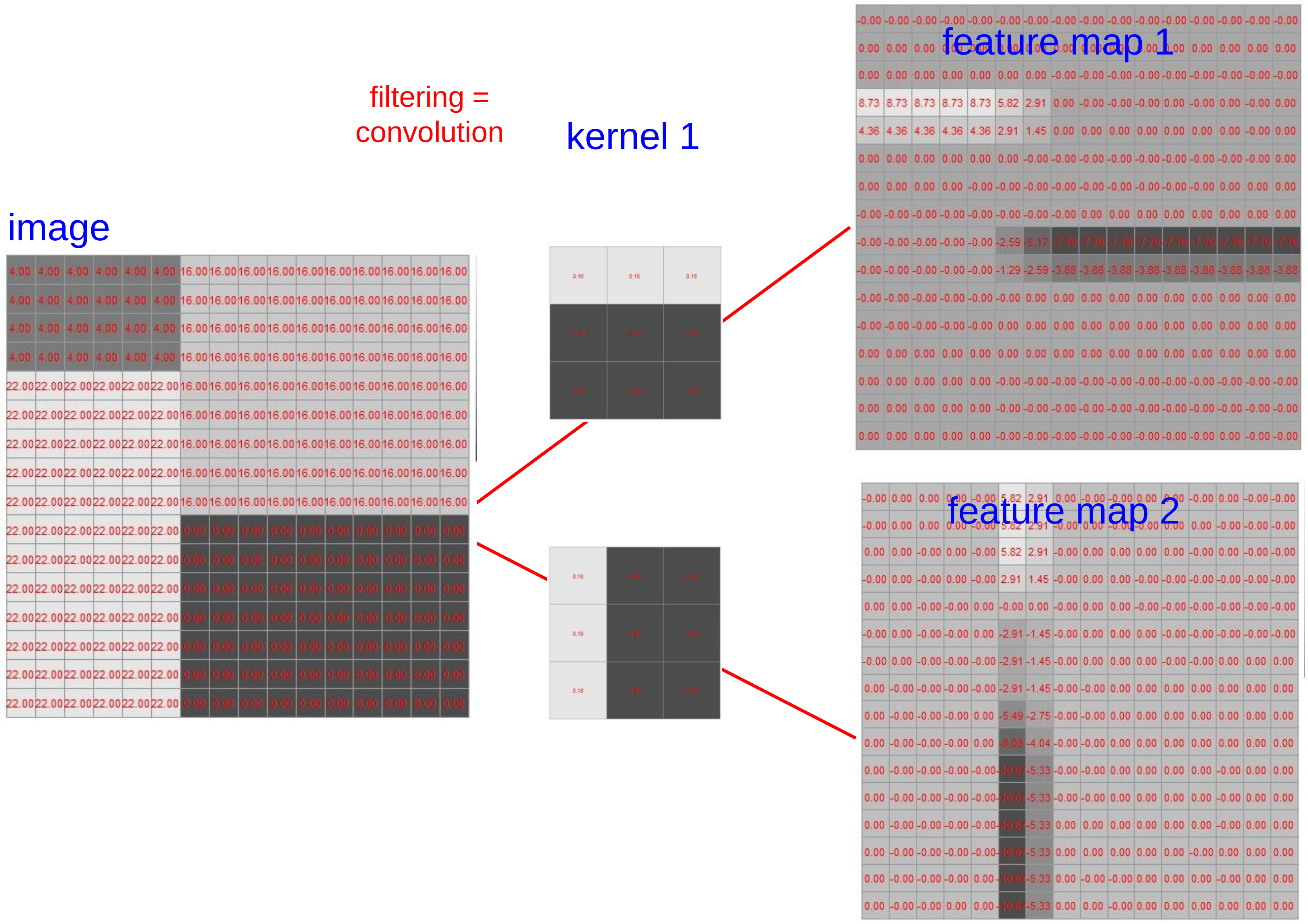


The filter is applied at each position of the image and it can be shown that the result is maximal if the image pattern corresponds to the weight pattern.

The results form again an image called feature map (=activation map) which shows at which position the feature is present.

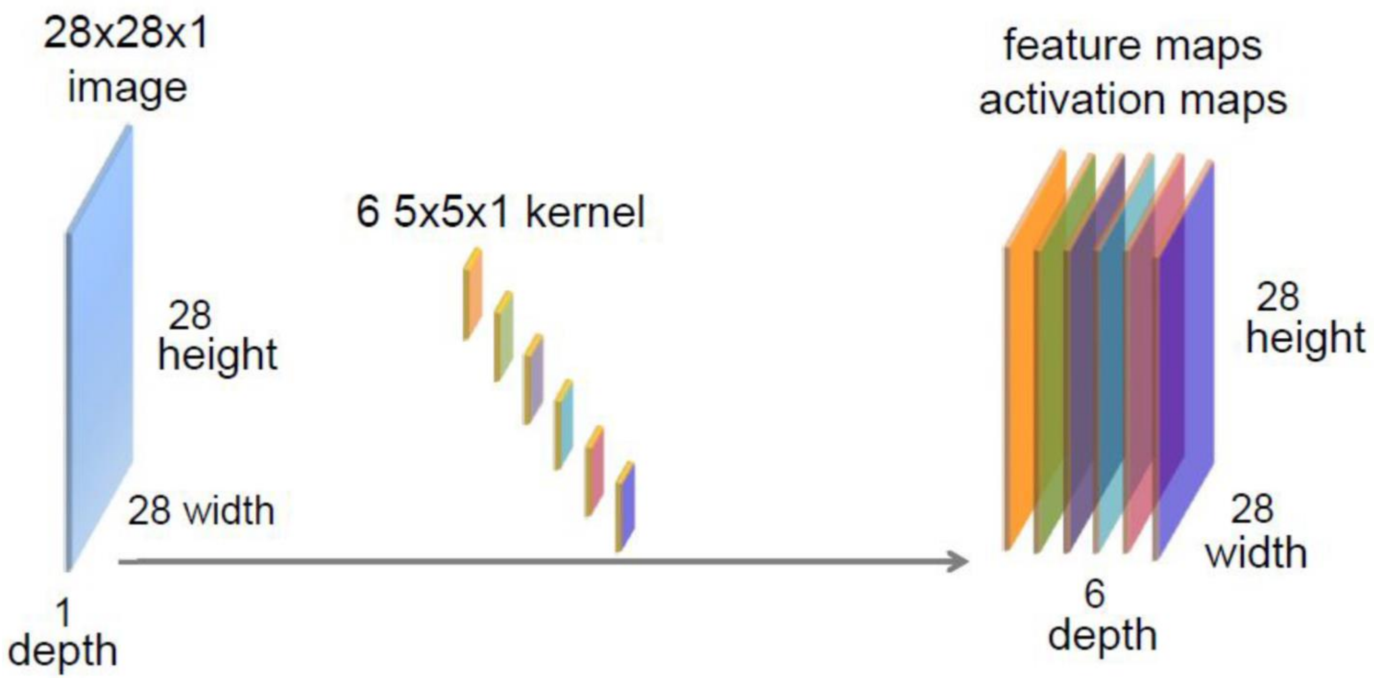


Convolutional networks use neighborhood information and replicated local feature extraction



The weights of each filter are randomly initiated and then adapted during the training.

Convolution layer with a 1-chanel input and 6 kernels

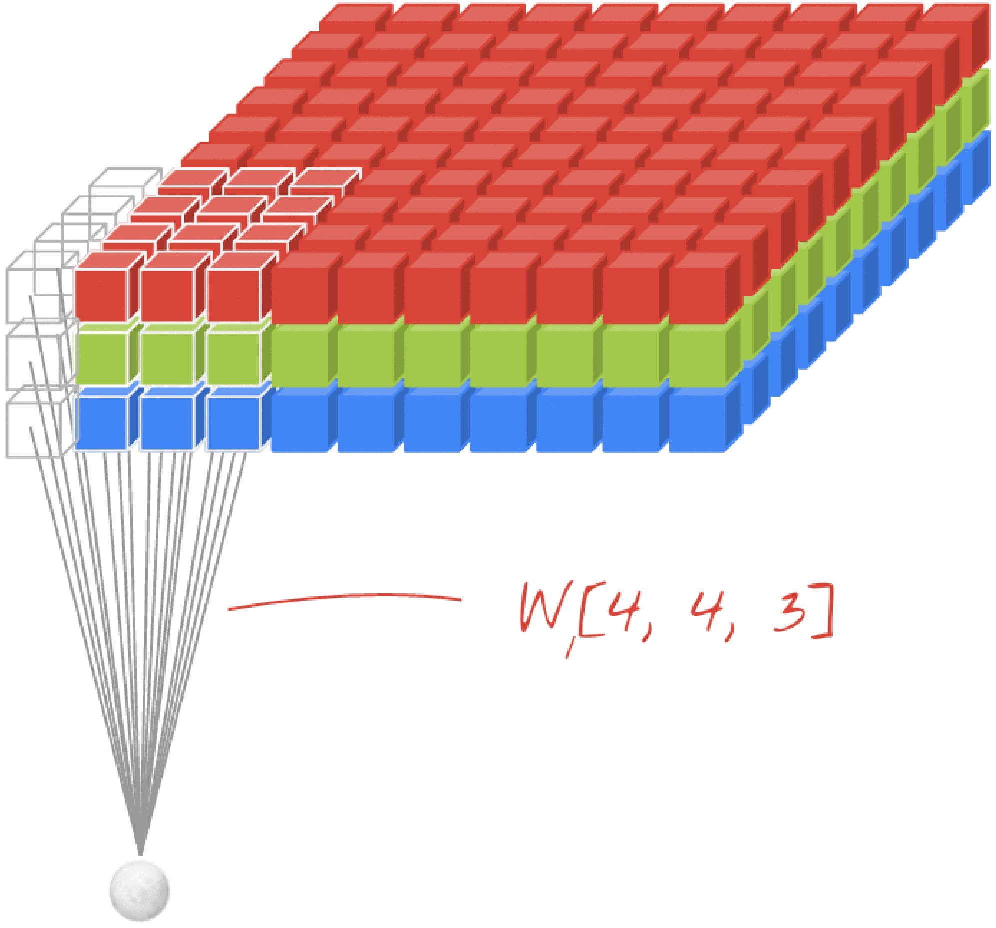


Convolution of the input image with 6 different kernels results in 6 activation maps.

If the input image has only one channel, then each kernel has also only one channel.

Animated convolution with 3 input channels

3 color channel input image

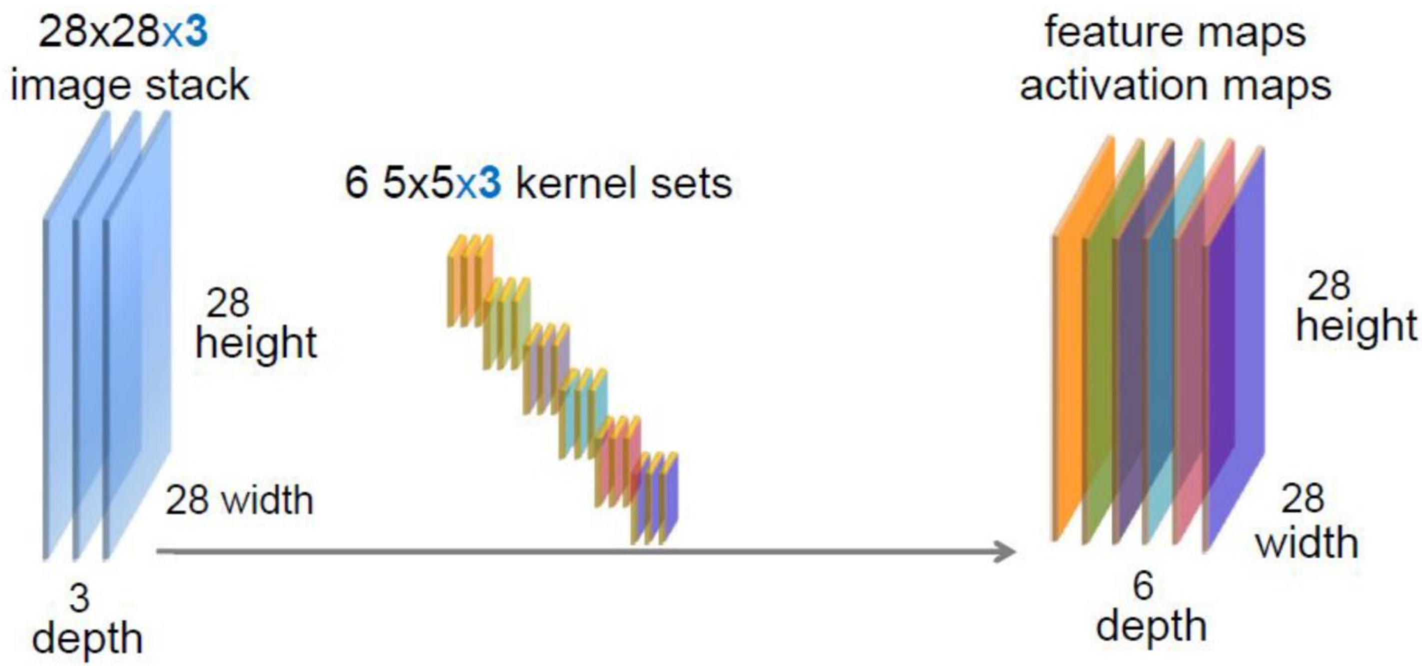


$$z = b + \sum_i x_i w_i$$

$W[4, 4, 3]$

Animation credits: M.Gorner, https://codelabs.developers.google.com/codelabs/cloud-tensorflow-mnist/#10
For an example with number see convolution demo in: https://cs231n.github.io/convolutional-networks/

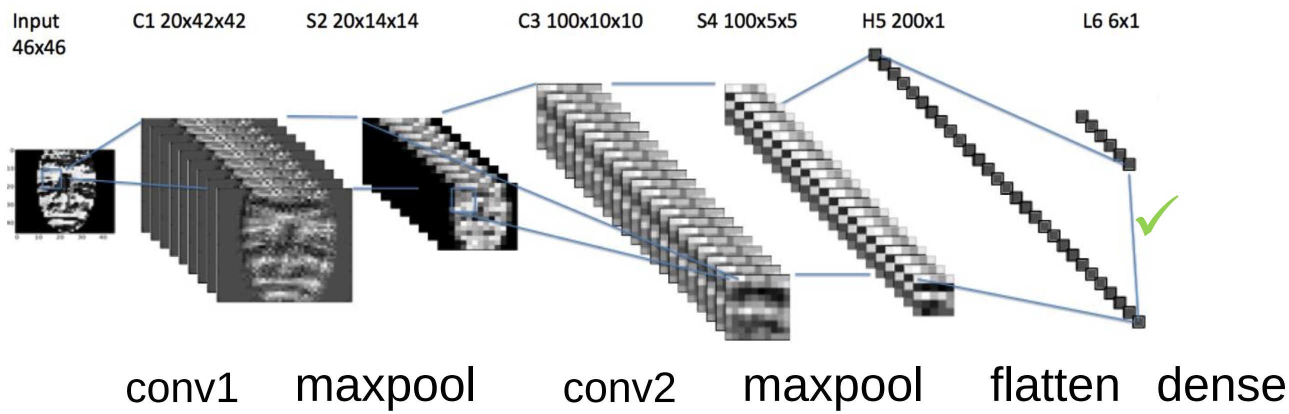
Convolution layer with a 3-chanel input and 6 kernels



Convolution of the input image with 6 different kernels results in 6 activation maps.

If the input image has 3 channels, then each filter has also 3 channels.

CNN ingredient II: Maxpooling Building Blocks reduce size



4x4

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

→

2x2

20	30
112	37

Simply join e.g. 2x2 adjacent pixels in one by taking the max.

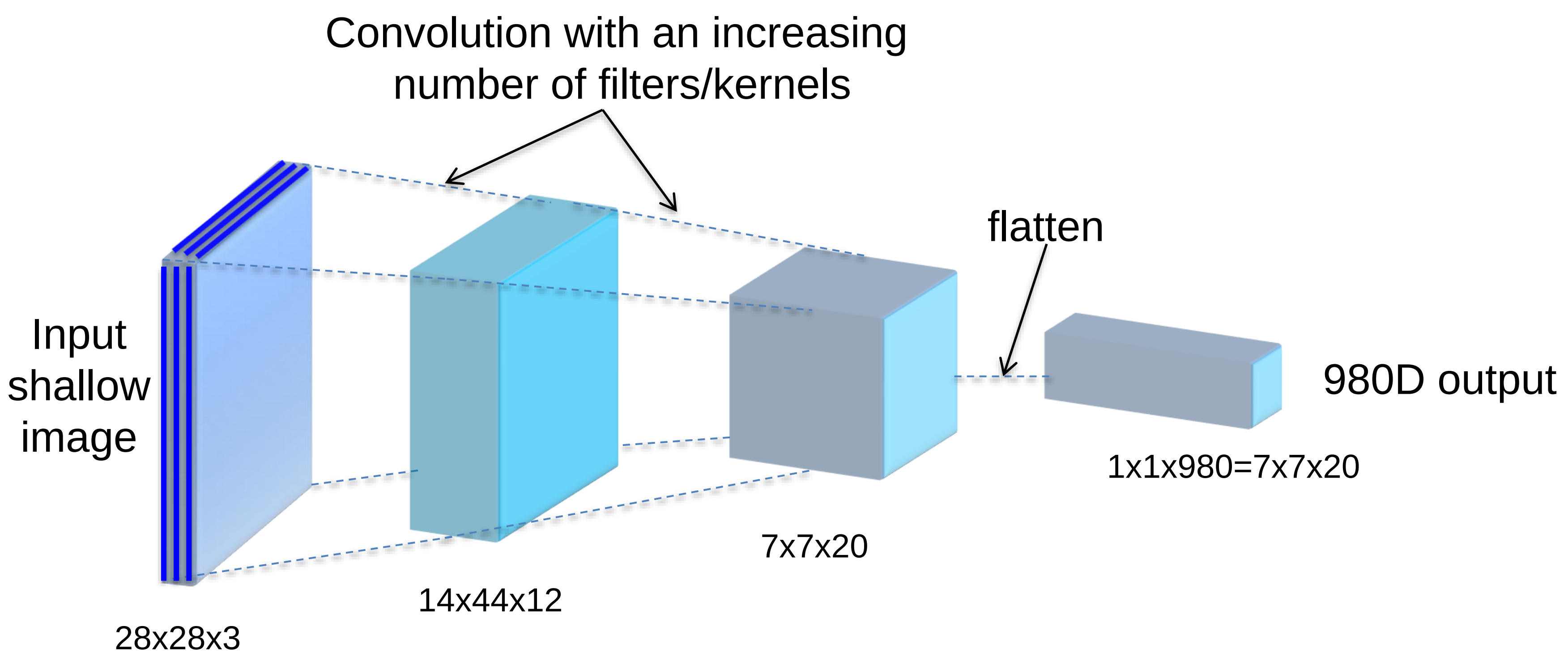
→ less weights in model

→ Less train data needed

→ increased performance

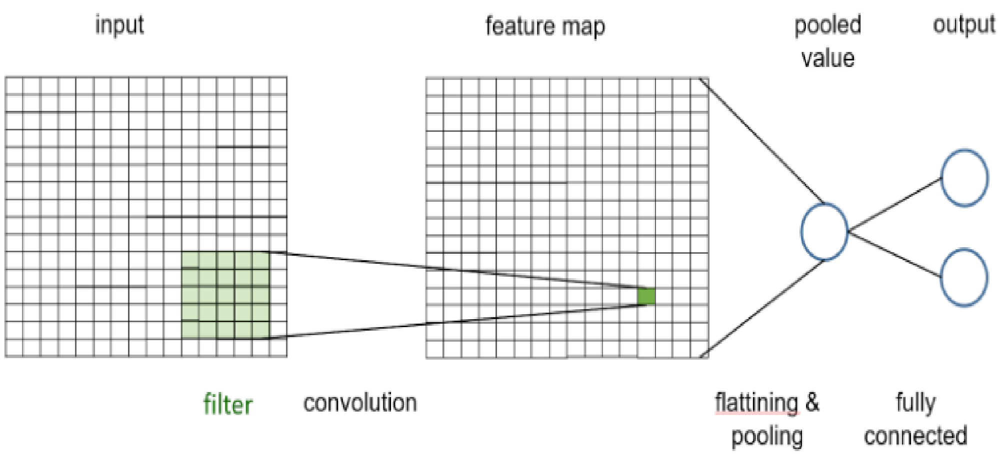
Hinton: „The pooling operation used in convolutional neural networks is a big mistake and the fact that it works so well is a disaster“

Typical shape of a classical CNN



Spatial resolution is decreased e.g. via max-pooling while more abstract image features are detected in deeper layers.

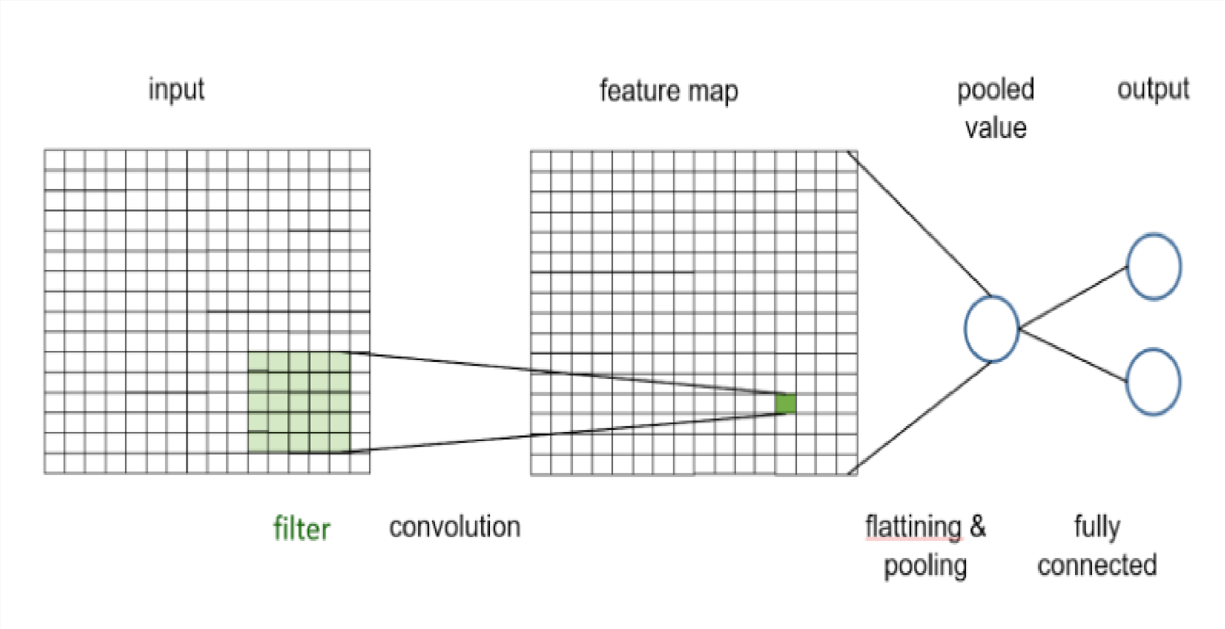
Building a very simple CNN with keras




```
model <- keras_model_sequential()
model %>%
  layer_conv_2d(filters=,
                kernel_size = c(5,5),
                padding = 'same',
…                input_shape = …,
                activation = 'linear') %>%
  # take the max over all values in the activation map
  layer_max_pooling_2d(pool_size = …) %>%
  layer_flatten() %>%
  layer_dense(units = 2,activation = 'softmax')
```

Fill the gaps!

Exercise: Artstyle Lover



Open NB in: https://github.com/tensorchiefs/dl_course_2020/blob/master/notebooks/05_cnn_edge_lover.ipynb

Summary

- NNs work best when respecting the underlying structure of the data.
  - Use fully connected NN for tabular data
  - Use convolutional NN for data with local order such as images
- CNNs exploit the local structure of images by local connections and shared weight (same kernel is applied at each position of the image).