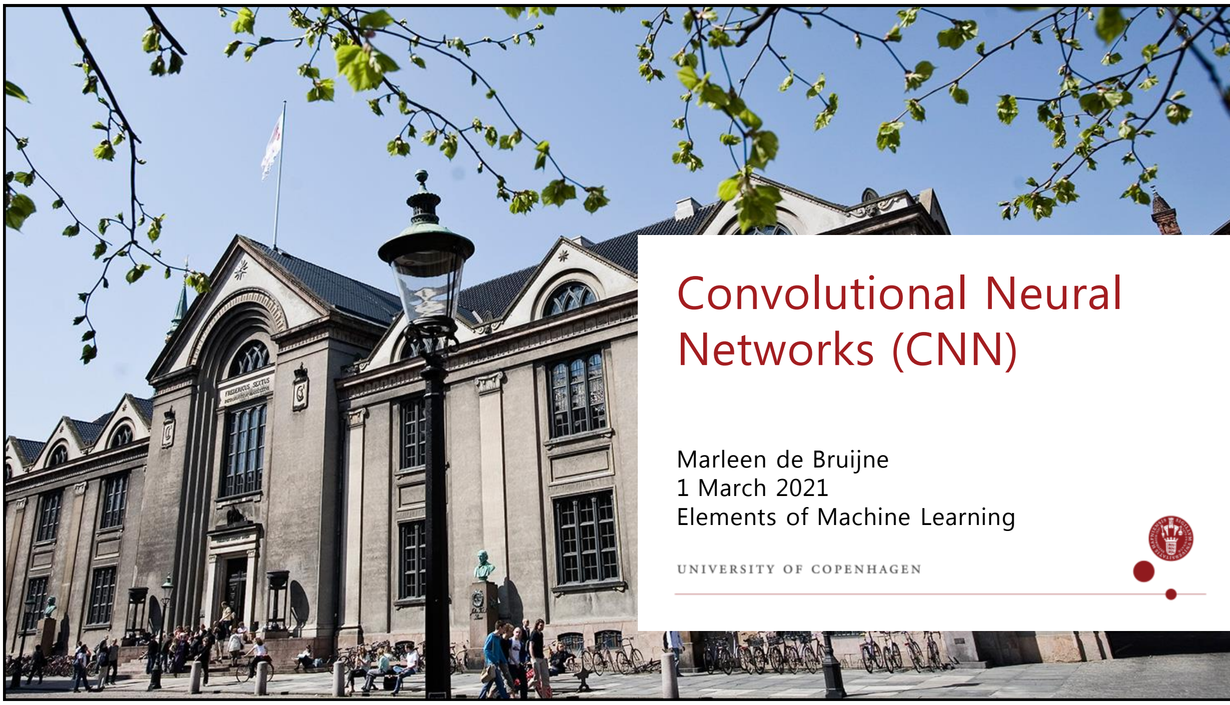


Convolutional Neural Networks (CNN)

Marleen de Bruijne
1 March 2021
Elements of Machine Learning

UNIVERSITY OF COPENHAGEN

1

General remarks

- We are recording
- Please ask questions!
  - I cannot see you when my lecture is up, so:
    - Unmute and say something
    - Or type in the chat (but I may not always see this)

2

Who am I

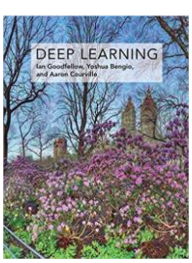
- Marleen de Bruijne
- MSc in Physics
- PhD in Medical Image Analysis
- Now Professor Medical Image Analysis at DIKU and at Erasmus MC – University Medical Center Rotterdam, The Netherlands
- Research:
  - Machine learning approaches to medical imaging
  - Automatic, quantitative analysis
  - Predicting disease
  - Brain, lung, and cardiovascular imaging

3

UNIVERSITY OF COPENHAGEN

This lecture

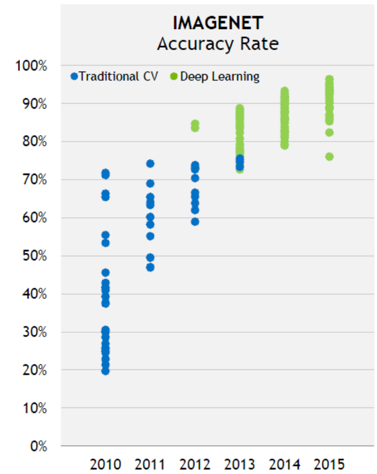
- Why **Convolutional** Neural Networks?
- What is convolution
- Typical components of CNN – what they do and why they work (or not)
- What CNN can and cannot learn
- Some practical considerations



4

Convolutional networks work **really well**

ImageNet Large Scale Visual Recognition Challenge



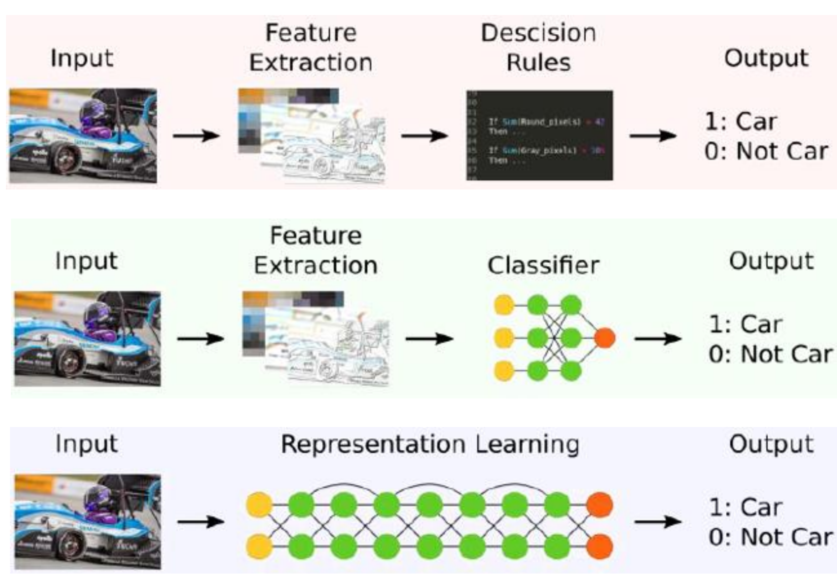
5

What was new?

Early AI
in computer vision

"Standard"
Machine Learning
in computer vision

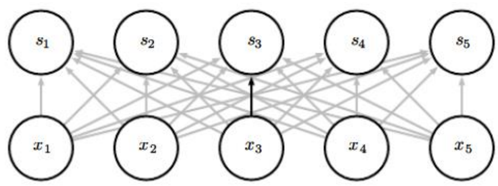
Deep Learning



6

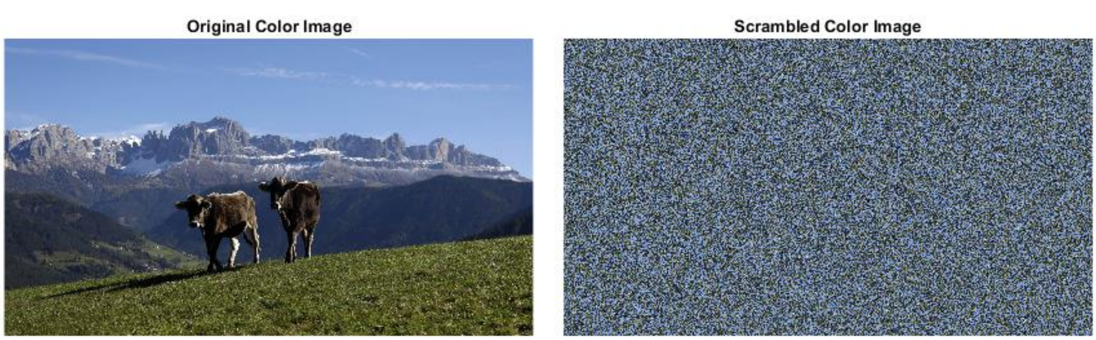
Comparing to “regular” networks as in the previous lecture

- A “regular” (fully connected) network would have weights from every image pixel to every node in the first layer, which is connected to every node in the second layer, which is connected……
- Lots of weights! Good or bad?
- Good, because: this is a very flexible model. It can learn anything!
- Bad because: Large memory footprint. Many computations. Probably, higher complexity (more flexible model) than needed. Risk of overtraining.



- CNN is a way of reducing the number of weights

A fully connected NN learns equally well from these two images



- Can we use image structure better?

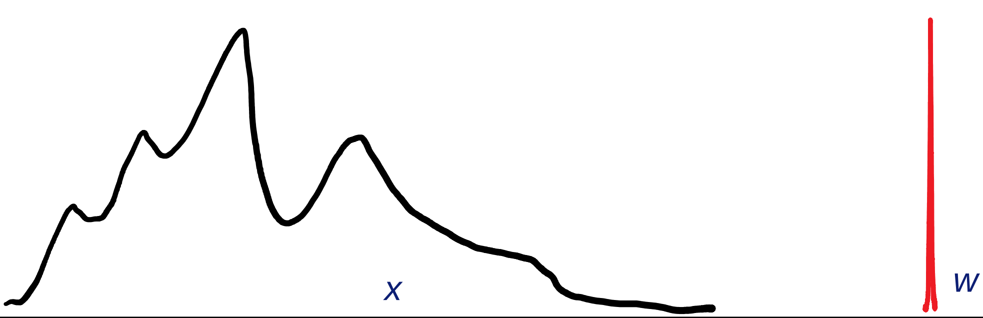
To a fully connected NN, these two images are very different

- Can we use image structure better?
- Model invariant properties?
- A CNN does this (to some extent)

What is convolution?

$$s(t) = \int x(a)w(t-a)da \tag{9.1}$$

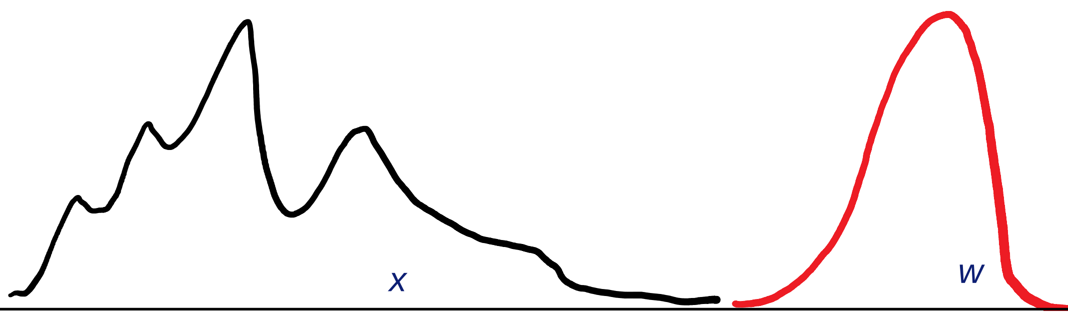
Or: The integral of the multiplication of a function and another function which is reversed and shifted.



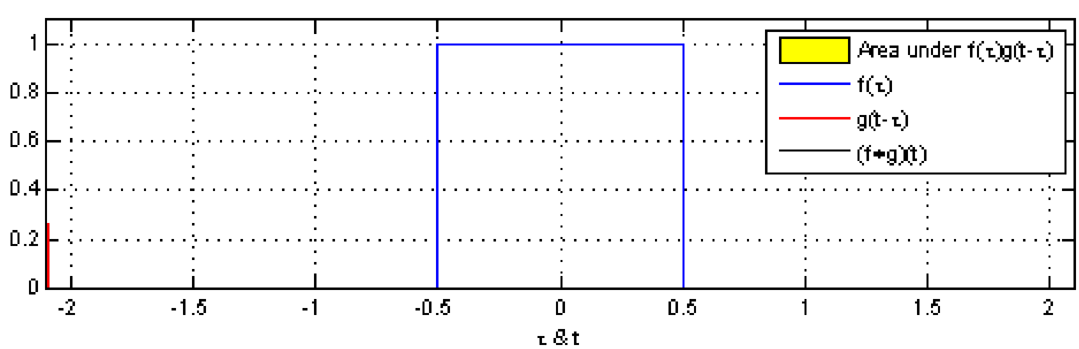
What is convolution?

$$s(t) = \int x(a)w(t-a)da \qquad (9.1)$$

Or: The integral of the multiplication of a function and another function which is reversed and shifted.



One more example
We filter the signal (blue) by convolving with a kernel (red)



Output is maximum at points where the two signals have the same shape

From wikipedia

What is convolution?

$$s(t) = \int x(a)w(t-a)da \tag{9.1}$$

Or: The integral of the multiplication of a function and another function which is reversed and shifted.

In the discrete, 2D version:

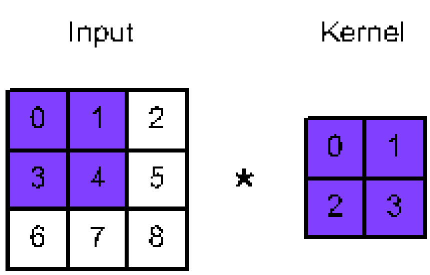
$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{9.5}$$

But in fact, we use **cross-correlation** (no reversing)

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n). \tag{9.6}$$

UNIVERSITY OF COPENHAGEN

Correlation operator, in 2D



Some filtering examples

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Blurring

Some filtering examples

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

Horizontal edge enhancement

Some filtering examples

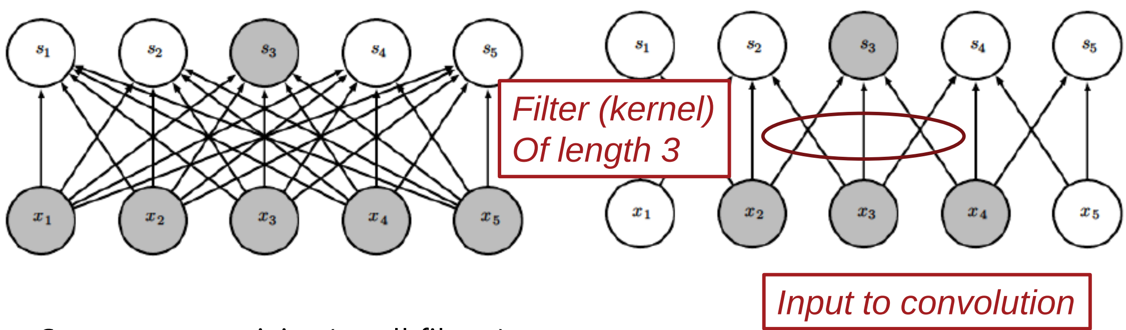
0	-1	0
-1	4	-1
0	-1	0

"Laplacian"
Enhances blobs
(and edges)

Convolutions in image processing, before deep learning

- Choose a set of convolution filters
  - Smoothing
  - 1st order derivatives in different directions (edges)
  - 2nd order derivatives (ridges)
  - Possibly higher order
  - (nonlinear) combinations of them (eg gradient magnitude)
  - At multiple scales
- Apply filters to image -> features
  - (often) Assume local image structure is sufficiently informative - filters are small
  - assume the same type of filters are useful everywhere in the image
- Train classifier (or other learner) on the features

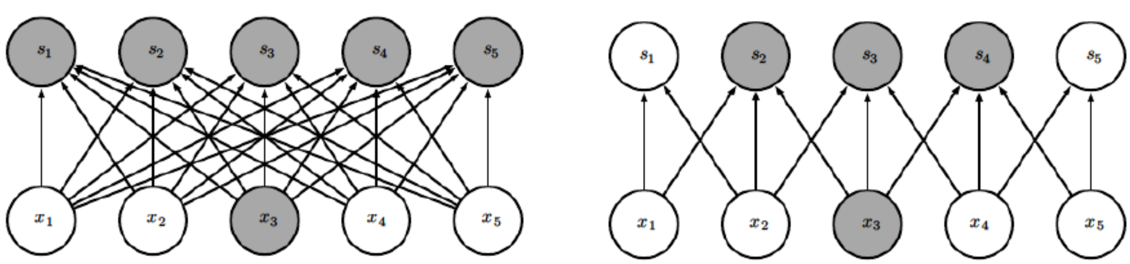
How to use this in a neural network?



- Sparse connectivity (small filters)
- Every node in hidden layer influenced by limited number of input nodes
- Each input node influences limited number of nodes in first layer

19

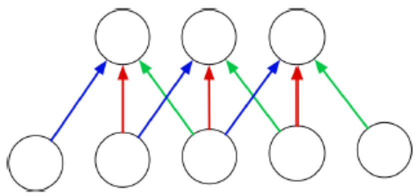
How to use this in a neural network?

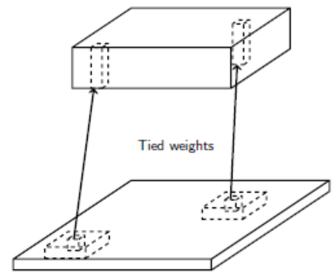


- Sparse connectivity (small filters)
- Each input node influences limited number of nodes in first layer
- Every node in hidden layer influenced by limited number of input nodes

20

How to use this in a neural network?

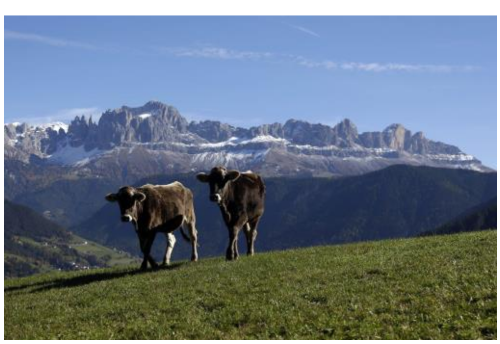




- Weight sharing: same filters/kernels used everywhere in the image
- -> CNN equivariant to translation
- If input is shifted, output remains the same (but shifted the same way)

To a fully connected NN, these two images are very different

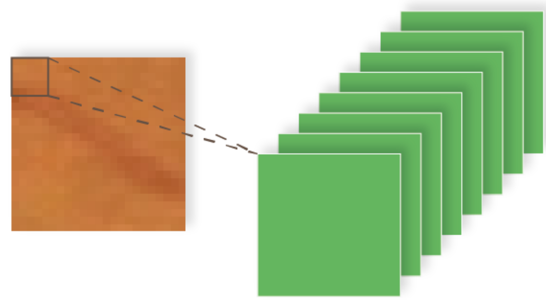
- Feature maps after several layers of CNN will look the same, just shifted
- (Except for boundary effects)



Convolutional neural networks

The filter kernel moves over the image $\rightarrow$ feature map

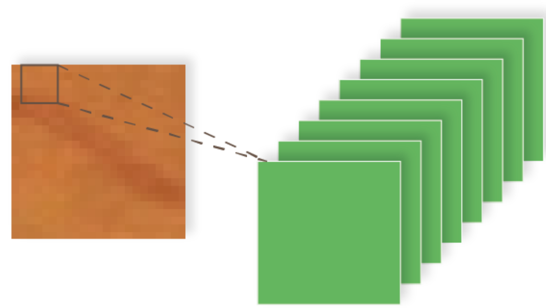
The step size is called **stride** (usually stride = 1)



Convolutional neural networks

The filter kernel moves over the image $\rightarrow$ feature map

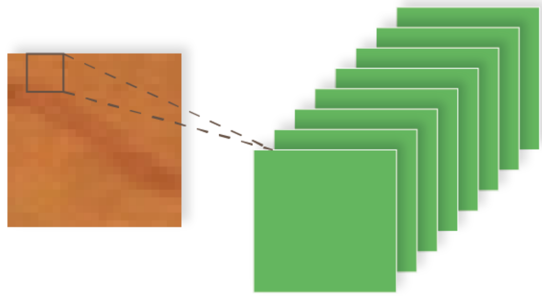
The step size is called **stride** (usually stride = 1)



Convolutional neural networks

The filter kernel moves over the image → feature map

The step size is called **stride** (usually stride = 1)

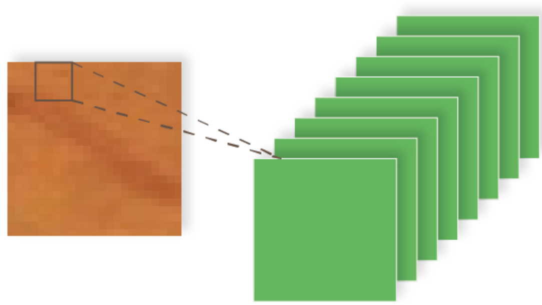


25

Convolutional neural networks

The filter kernel moves over the image → feature map

The step size is called **stride** (usually stride = 1)

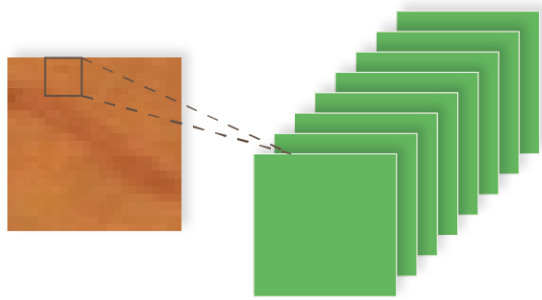


26

Convolutional neural networks

The filter kernel moves over the image → feature map

- The step size is called **stride** (usually stride = 1)

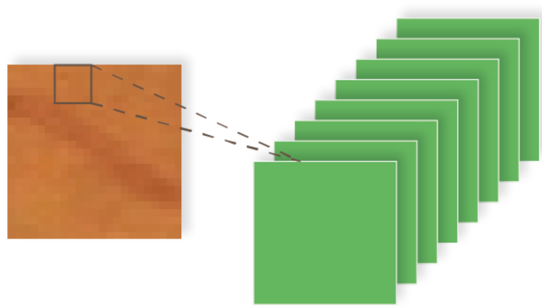


UNIVERSITY OF COPENHAGEN

Convolutional neural networks

The filter kernel moves over the image → feature map

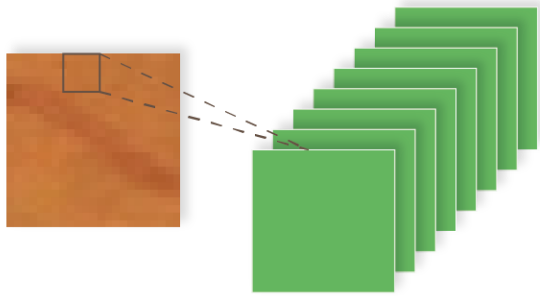
- The step size is called **stride** (usually stride = 1)



Convolutional neural networks

The filter kernel moves over the image → feature map

The step size is called **stride** (usually stride = 1)

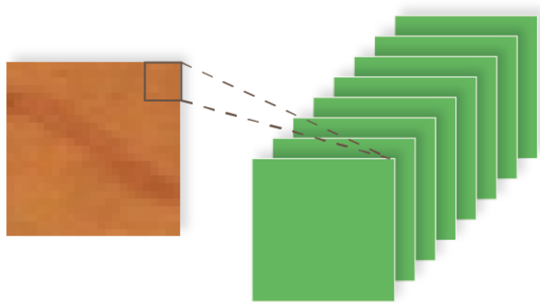


UNIVERSITY OF COPENHAGEN

Convolutional neural networks

The filter kernel moves over the image → feature map

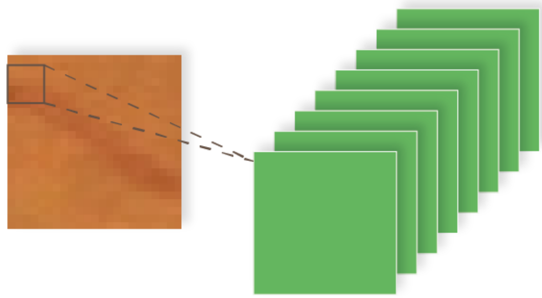
The step size is called **stride** (usually stride = 1)



Convolutional neural networks

The filter kernel moves over the image $\rightarrow$ feature map

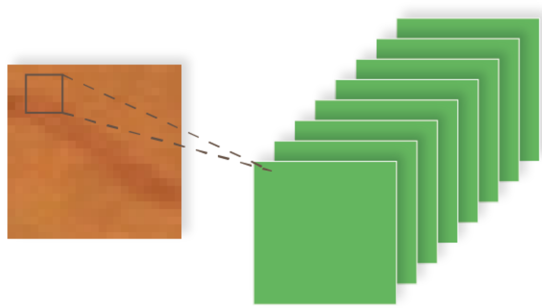
The step size is called **stride** (usually stride = 1)



Convolutional neural networks

The filter kernel moves over the image $\rightarrow$ feature map

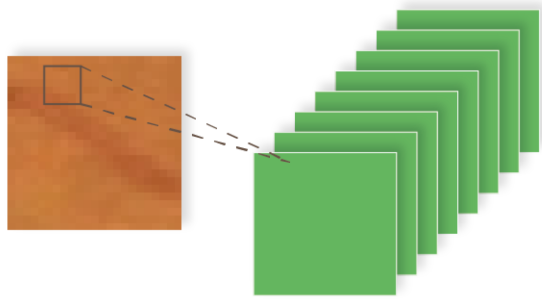
The step size is called **stride** (usually stride = 1)



Convolutional neural networks

The filter kernel moves over the image $\rightarrow$ feature map

The step size is called **stride** (usually stride = 1)



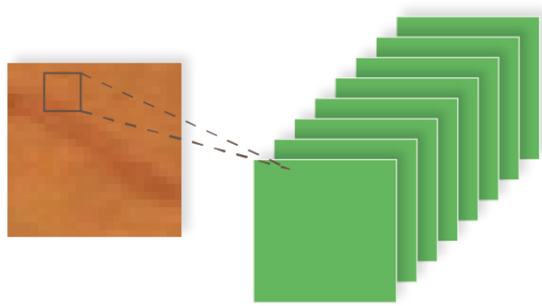
33

UNIVERSITY OF COPENHAGEN

Convolutional neural networks

The filter kernel moves over the image $\rightarrow$ feature map

The step size is called **stride** (usually stride = 1)

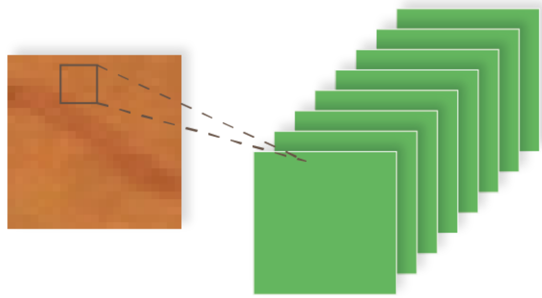


34

Convolutional neural networks

The filter kernel moves over the image → feature map

The step size is called **stride** (usually stride = 1)

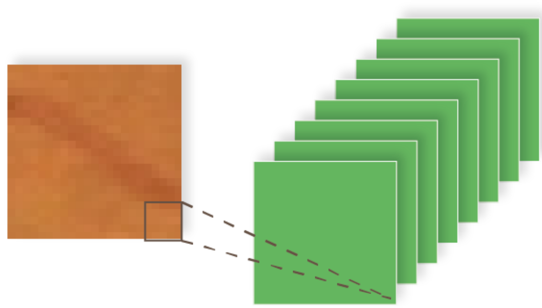


35

Convolutional neural networks

The filter kernel moves over the image → feature map

The step size is called **stride** (usually stride = 1)



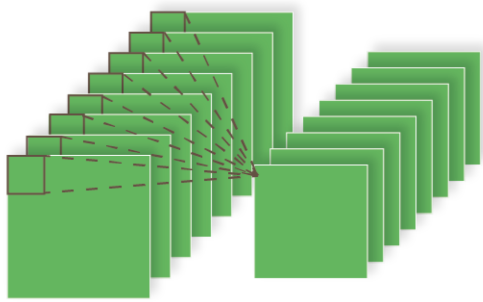
M output channels (feature maps)

36

Convolutions can be stacked

Input channels (features, RGB, ⋯) are combined

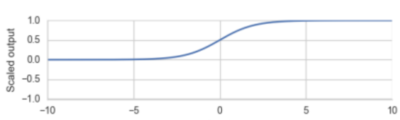
8 x 5x5 weights



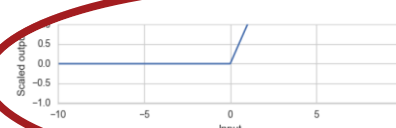
Something missing here?

N input channels *M output channels*

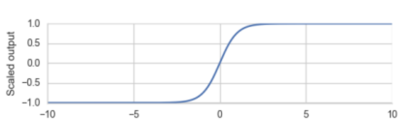
Non-linearities



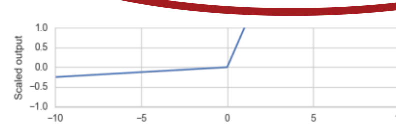
Sigmoid



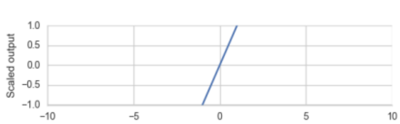
ReLU



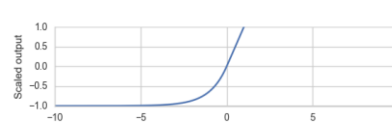
tanh



Leaky ReLU

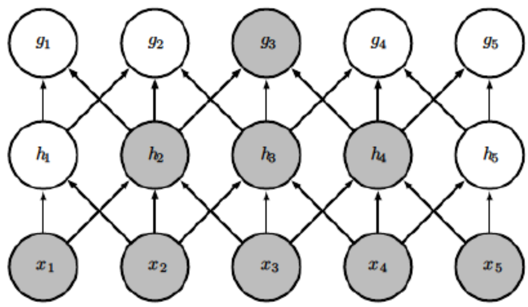


linear



eLU

Stacking convolutions increases the receptive field



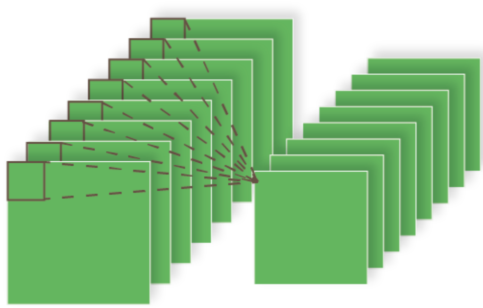
- Compare 1 layer of 5x5 and 2 layers of 3x3 convolutions
- What is the receptive field? Which has more parameters? Which has more non-linearities?
- In practice, we typically use more layers of small kernels

39

Convolutions can be stacked

Input channels (features, RGB, ⋯) are combined

8 x 5x5 weights

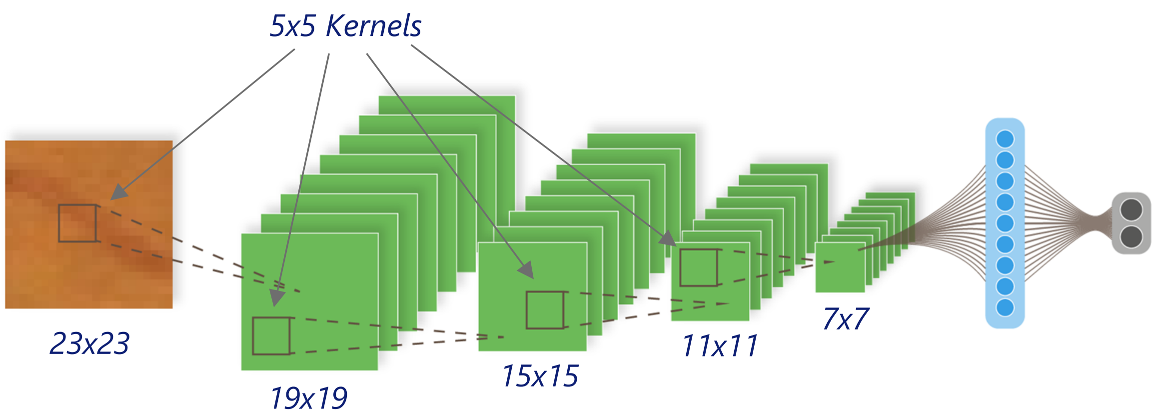


Feature maps in the 2nd layer are smaller

N input channels *M output channels*

40

Valid convolutions / border handling

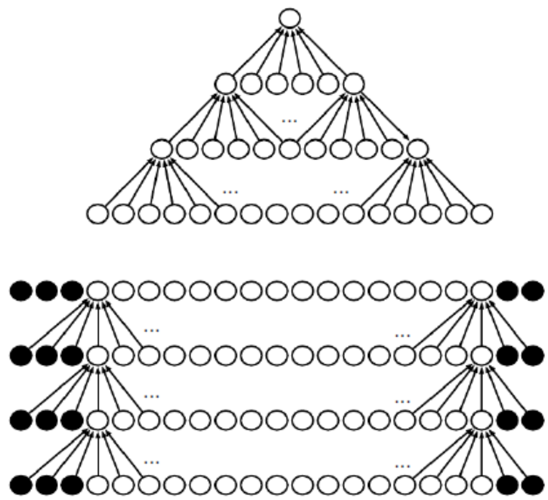


Strategies to handle boundaries

- “valid” convolution – output is smaller by k-1
- “same” convolution – use zero padding outside the image so that output is same size as input
  - Border pixels have less influence in the output
- “full” convolution – even more zero padding – output larger by k-1

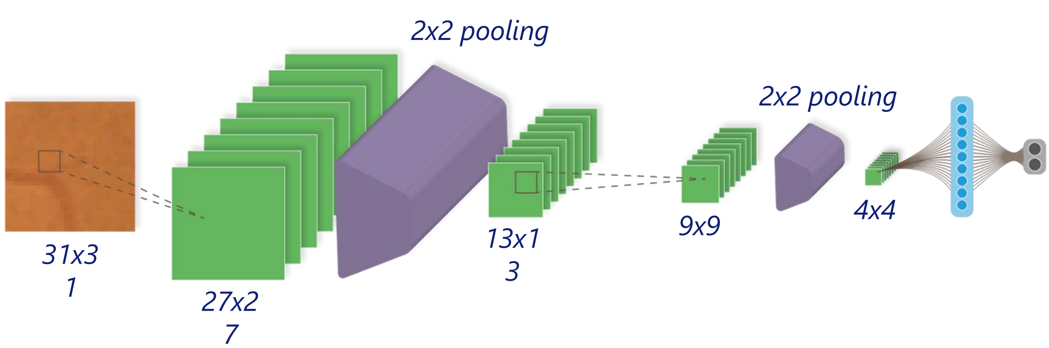
- Consider the effect of zero padding!

Zero padding versus valid convolutions



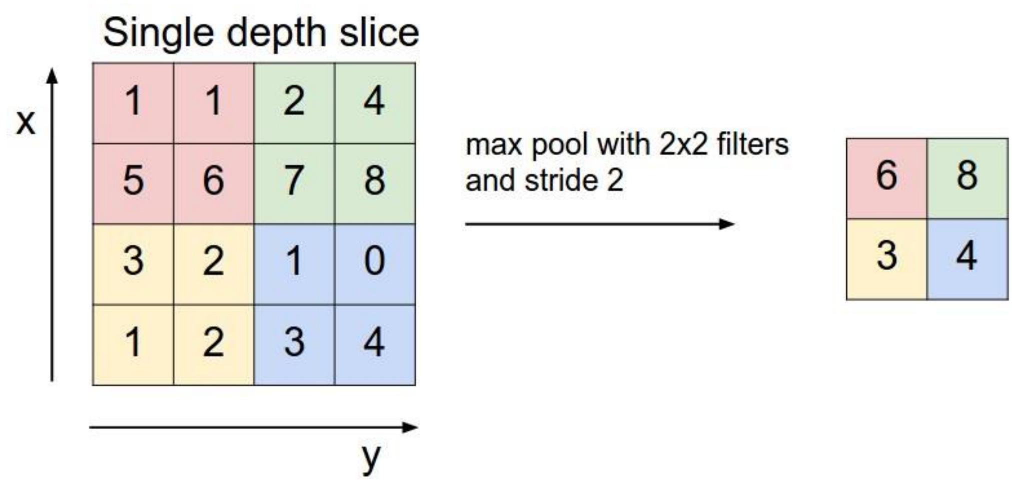
43

Many CNNs include pooling layers



44

Max Pooling and Average Pooling most common



45

Pooling

Why use pooling?

- Reduce memory footprint
- Reduce computations
- Invariance to small transformations

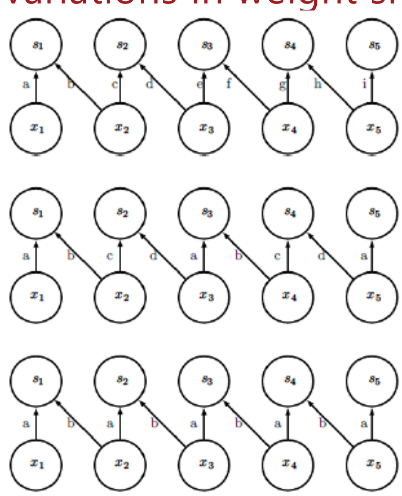
Why not use pooling?

- Loose location information
- Loose fine detail

46

Variants of convolution

Tiled convolution – variations in weight sharing



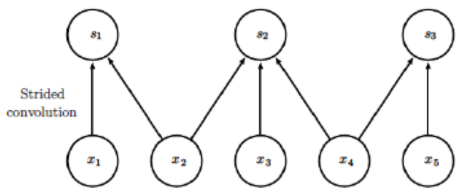
Locally connected

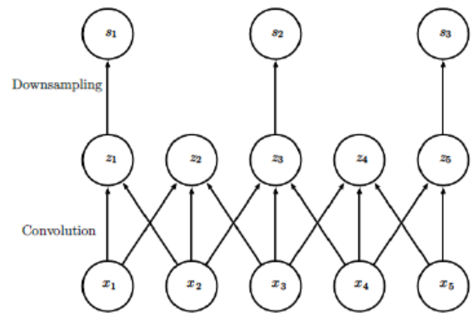
Tiled convolution

Standard convolution

- Not common in practice

Strided convolution

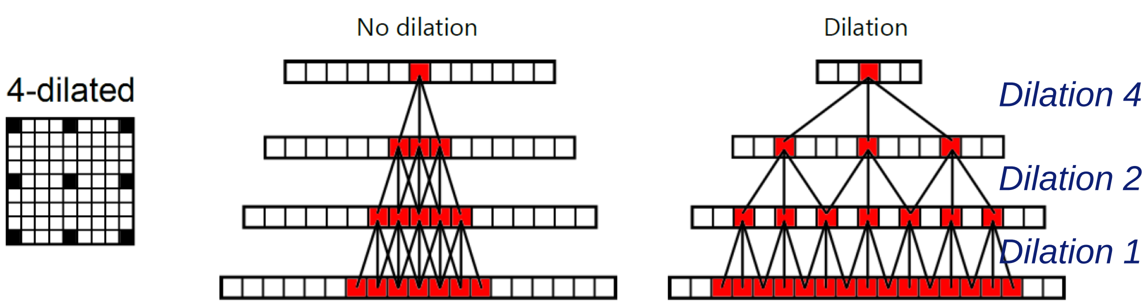




- Downsamples the input efficiently
- Effect similar to pooling, but loose some information

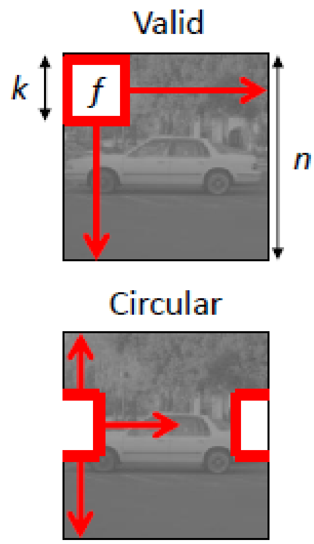
Dilated convolution

- Skip positions in the kernel. Why?

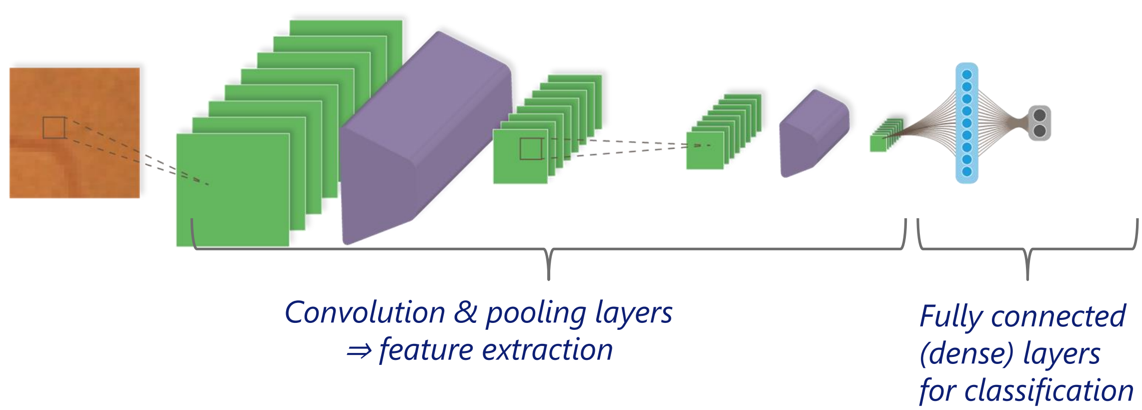


- Quickly increase receptive field, without pooling
- Drawback: large memory footprint

Circular convolution/padding with something else than 0



A typical (simple) CNN for image classification: combine convolution layers and fully connected layers



- How do we optimize this?
- What if images have different sizes?

What can networks learn

- Global pooling versus fully connected layers in the end

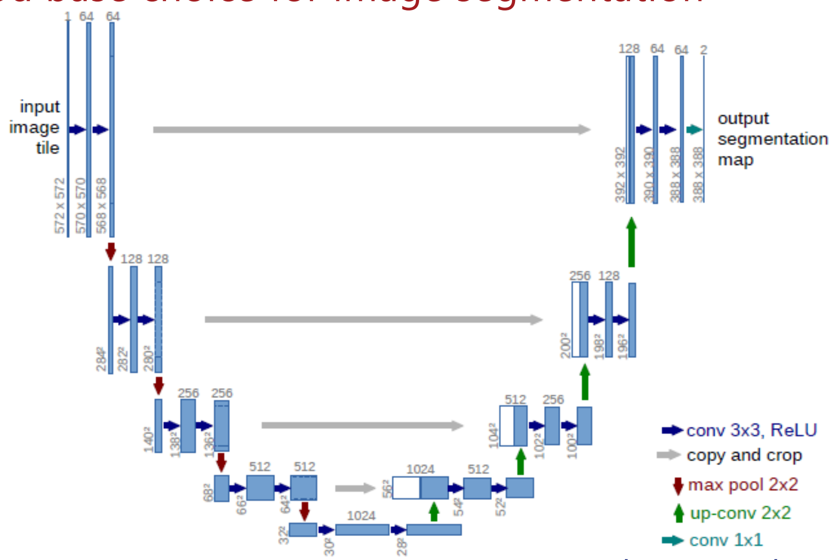
- Global pooling learns **presence** of specific features (average/max)
- Fully connected layers learn also **spatial configuration**

53

CNN for semantic segmentation

54

Unet: good base choice for image segmentation



Ronneberger et al, MICCAI 2015

55

Do we need to learn the features?

- Common (early) approach:
- Download a **pretrained** model (eg, trained in ImageNet)
- Fine tune:
  - Update weights using backprop
  - In all layers, or a subset
- Why does this work?
- Even random convolution kernels can work quite well (Saxe et al, NIPS 2011)

56

Has the machine learned to "see" as a human?

NO

But they are becoming very good at predicting our decisions

57

Some practical considerations

- When working with images, consider if preprocessing is needed
  - Does the target data "look" similar to the training data? What are the differences?
  - Remember that CNN are invariant only to translations (to some extent) and are sensitive to changes in intensity/color/orientation/scale
- What receptive field do you need?
- Bias-variance trade off, overfitting and underfitting. SEE NEXT LECTURE.
- Trend towards deeper models, but need to take dataset size into account

58

- With material from
- http://www.deeplearningbook.org/
- Roger Grosse http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/
- Nikolas Lessmann
- Jan van Gemert
- Dive into deep learning https://d2l.ai/
- https://aishack.in/tutorials/image-convolution-examples/