Deep learning and CNN's

Søren Olsen

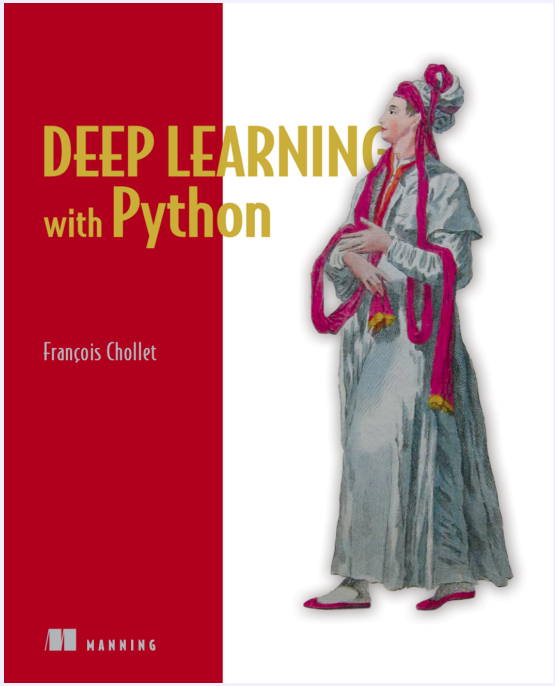
Department of Computer Science
University of Copenhagen

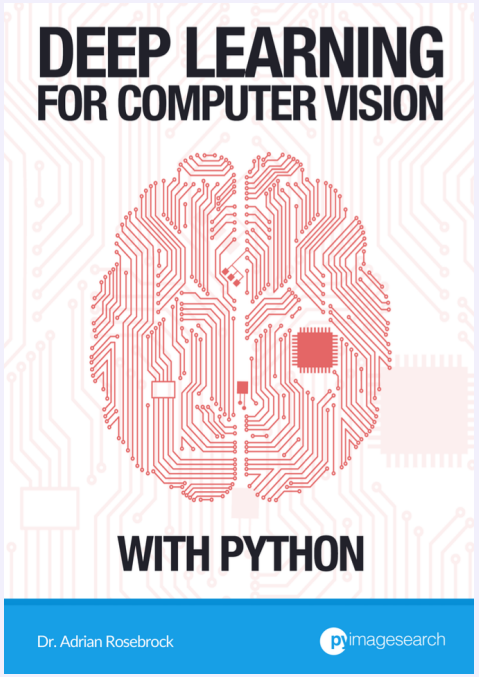
Today

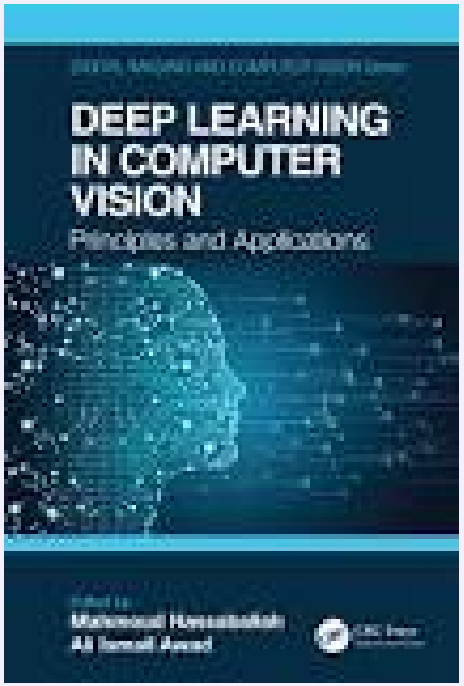
- Litterature and Computer Vision history
- Neural nets
- Convolutional Neural Nets
- CNN Architecture design

Litterature

Ponti et.al. *Everything you wanted to know about Deep Learning for Computer Vision but were afraid to ask*; 2017

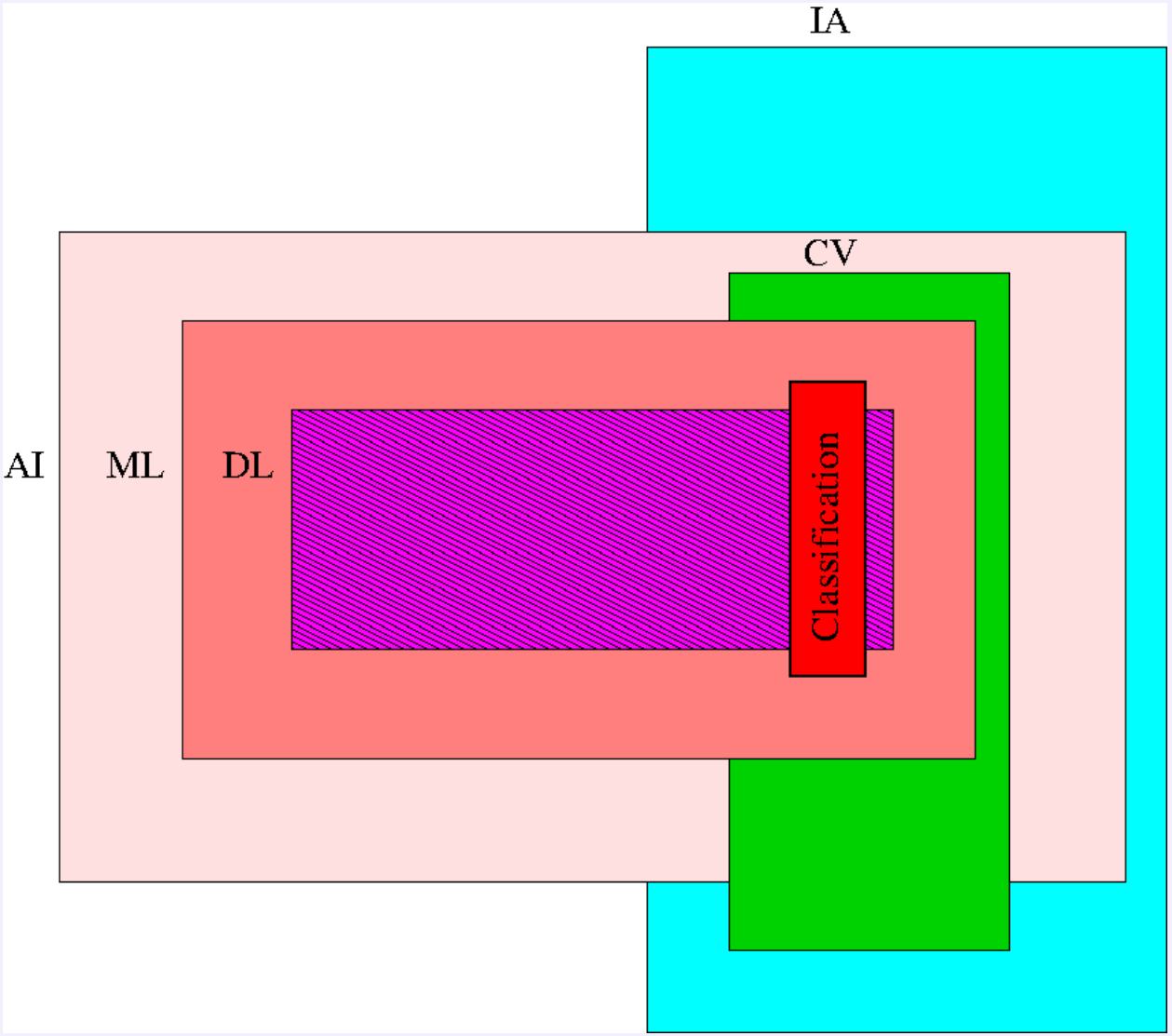






Liu et.al: *A Survey of Convolutional Neural Networks: Analysis, Applications and Prospects*; 2021.

IA and AI



Background/History

Computer Vision (CV) may be viewed as an offspring of Image processing. However, many other disciplines have played a central role in the development of CV.

- Computer Science
- Mathematics
- Physics (optics)
- Perceptual psychology
- Photogrammetry
- etc.

CV is truly multidisciplinary.

IP/IA and CV

Modern image analysis grew up during the 60ies end the 70ies and matured during the 80ies and 90ies. New methods developed gradually in parallel with the access to faster and (in particular) larger computers.

CV developed in USA in the late 70ies. To some degree a single person, David Marr can be credited. Marr was affilliated with MIT (psychology). In 1980 his book Vision gave rise to a burst of research all over the world.

Elements of Marr's theory on vision, e.g. the concept of Scale Space, originally suggested by Witkin, may still be found in many modern approaches and algorithms.

Marr's layers of abstraction

Marr operate with 3 levels of abstraction following the raw image data:

Primal sketch makes explicit features like edges and blobs, and their spatial geometric relationship, including grouping, colinearity and terminations.

2.5D sketch represent rough depth of visible surfaces, discontinuities in depth and orientation all in a viewer centered coordinate frame. Contributing processes include stereo, motion and optical flow, shape from shading etc.

3D model representation describes volumetric objects and shapes arranged hierarchically in a object centered coordinate frame.

Marr imagined a bottom-up process through the layers resulting in an abstract representation of the 3D scene surrounding us.

The 90ies and the 00s

This was the golden age of classical CV. As computers become larger and faster, new reliable methods emmerged and application began to see light. In particular within CV, the appropriate mathematics for expressing e.g. the geometry of multi-view geometry, was established.

A significant development was a number of semi-large databases for performance measuring of methods for segmentation, stereo, optical flow, object recognition etc. was created and made public. This contributed to maturing the field.

As result, industrial applications became more frequent. Examples on popular applications include: Google Street View, Shazam, Vivino, OCR-systems, Postal code readers, JPG, MPEG, MR-scanner reconstructions etc.

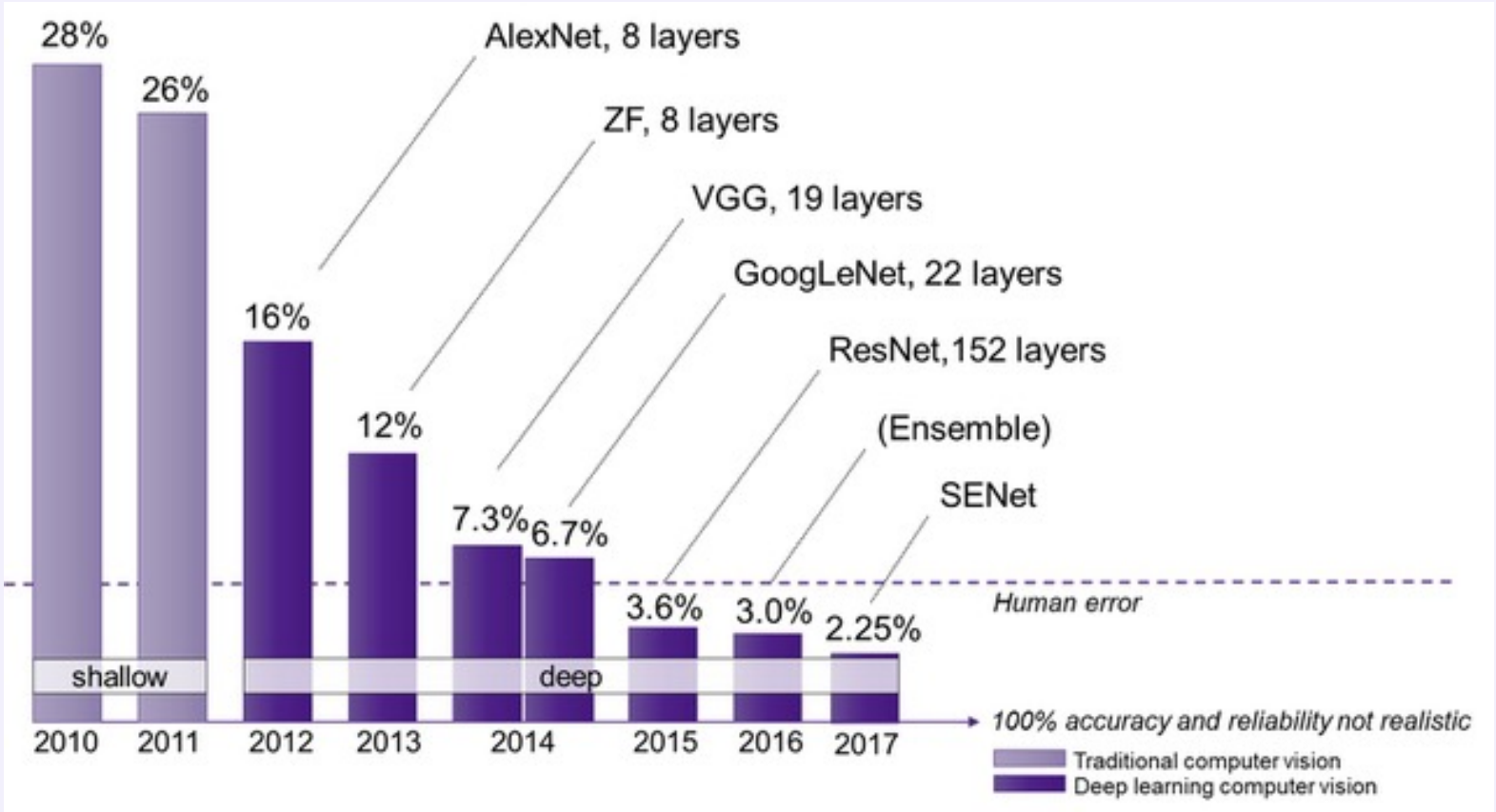
Recent trend - Neural nets

Neural nets have been known since the 40'ies and was (for a few years during the 90ies) very popular. However, the models had far too many unknown parameters, and was far to difficult to train on the available hardware. Also, essential numerical optimization software and the huge amount of needed annotated data was not available.

In 2012, **AlexNet** won the ImageNet Large Scale Visual Recognition Challenge by a large margin over all previous methods. Alex net was a so-called Convolutional Neural Net (**CNN**) with more than 60M parameters trained on newly available fast Graphical processing units (GPUs) using The backpropagation algorithm.

This was a game changer

ImageNet results



GPU's

A (now old) Nvidia Titan XP has 3840 Cuda kernels, and delivers about 12 TFLOPs in processing power. Modern Titan RTX is up to 4 times faster.

Popularizing the Cuda platform for parallel programming NVidia has established itself as provider of both hardware and low-level software for Machine Learning and Image Analysis.

Backpropagation

Neural nets are trained by changing their parameter values such that their response becomes closer to the desired output. This is done using (a variant of) The backpropagation algorithm.

First the input is passes through the net and the difference between the output and the desired output is computed. Then, according to a specified loss-function, and using the chain rule of differentiation, the errors are passed backwards through the net, and the parameters of the neurons adjusted slightly to reduce the error.

Automatic differentiation and parallization

Essential for easy design and use of neural nets is that the updating equations for the individual parameters in a large net may be automatically derived and applied.

Automatic symbolic differentiation has been known for decades and available in programs such as Maple and Mathematica. During the period 2014-18, a number of new software platforms spicialized to this task and to application on a parallel architecture such as a GPU, has emmerged, including: Caffe, Theano, Torch/Pytorch, Ternsorflow.

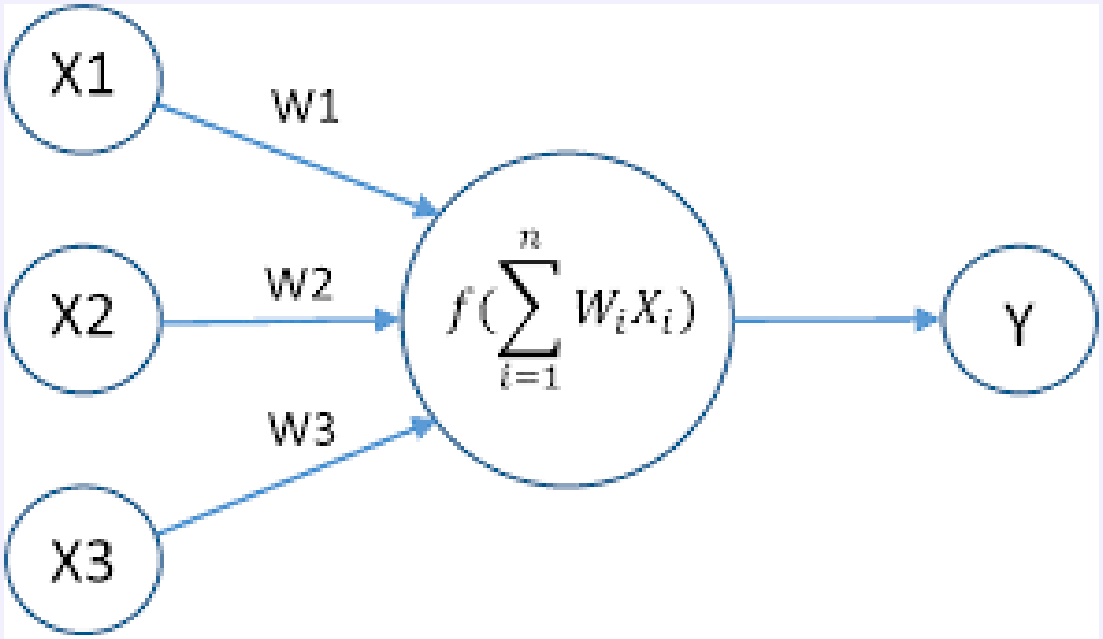
Also programming environments, such as Keras running on top of the abovementioned makes experimantal implementations easy.

Questions ?

Neurons

An (artificial) neuron is a tiny processing unit that computes a linear combination of its input values.

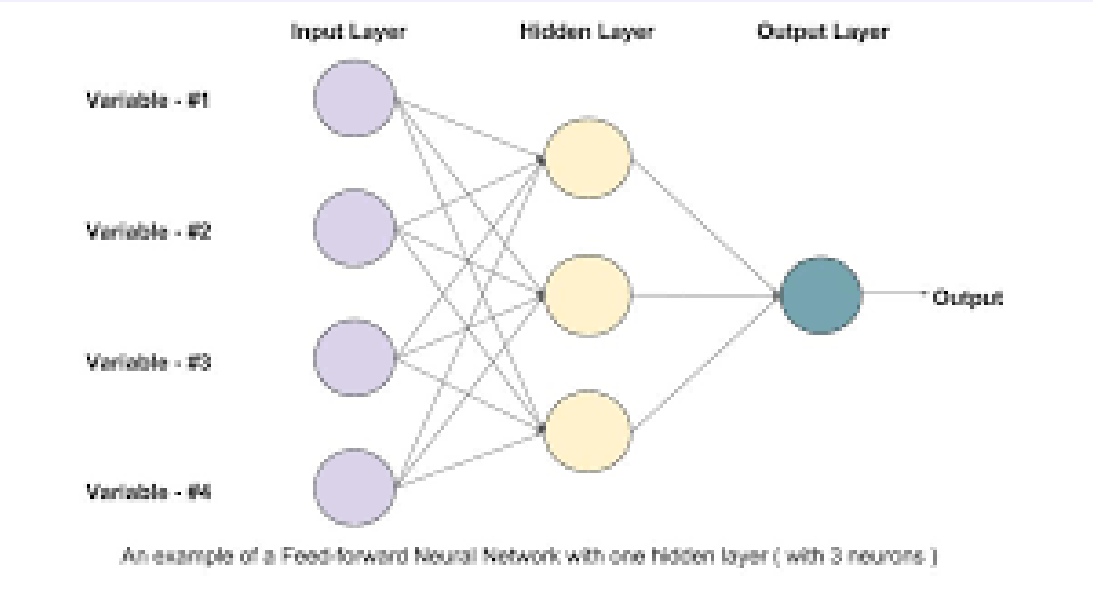
$$y = \sum_i w_i x_i + b = \sum_j w_j x_j$$



The simple linear model seems to have very little in common with neurons found in brains.

Neural Nets

For image analysis you may imagine input neurons attached to all pixels and having all pixel values in a local neighborhood as input. The net is said to be fully connected.



The 19 parameters (of the toy example above) is not chosen by engineering/design, but learned.

Fully connected nets are expensive in the number of parameters and often hard to train.

Convolutional Neural Nets

To reduce the number of parameters all neurons within the same layer may share the same weights w_i. In this case the computation will be a convolution.

The basic processing in a CNN is a number of stacked convolutional layers (CONV). The number of parameters is independent on the image size, but depend on the number of channels.

A net with CONV-layers only could be implemented with only a single layer. To increase the computing repertoire non-linear layers are needed.

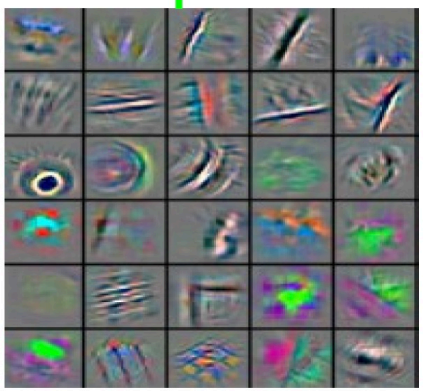
Channels

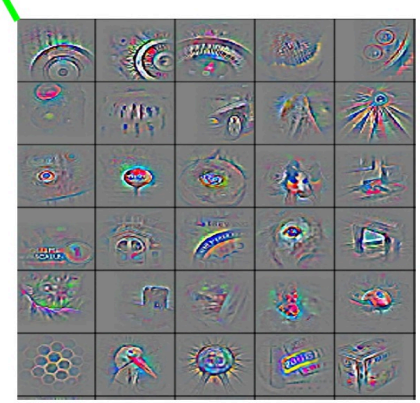
An RGB-image has 3 channels. The number of input channels to the first concolutional layer thus is 3. However, several sub-layers, using the same input may produce different output and producing a different/larger number of output channels.

Usually the number of channels increases (often exponentially) up through the network. This results in a similar increase in the number of parameters as we go up through the net.

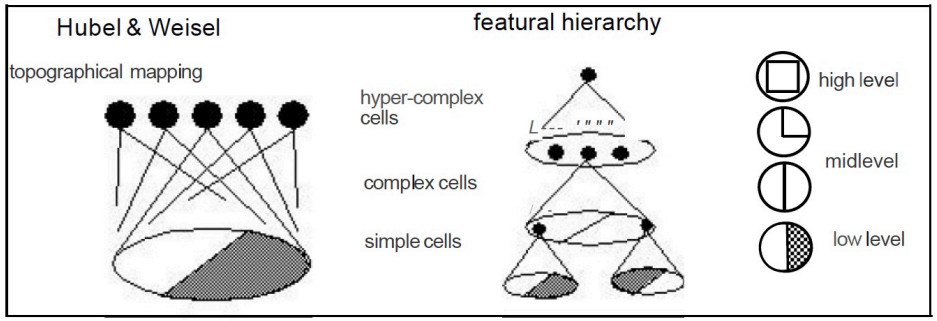
It is part of the engineering/design of the network architecture to choose a small, but sufficient large number of output filters/channels. This is application dependent. Often 8-64 are used at the first layer and the number doubled after each subsampling.

Low-level feature → Mid-level feature → High-level feature → Trainable classifier





Zeiler & Fergus, Visualizing and Understanding Convolutional Networks, 2013



Learning hierarchical representation

Convolutions are 3D

If a filter has a spatial dimension of say $N \times N$ and works on input with C channels, then the number of parameters are $CN^2 + 1$.

A 3D kernel (filter) $w_{i,j,c}$ is called a tensor.

$$y = \sum_{i}^{N} \sum_{j}^{N} \sum_{c}^{C} w_{i,j,c} x(i,j,c) \; + \; b$$

Thus, a 1×1 filter is just a pixelwise linear combination of the channel data.

Class exercise - 0.5 minute

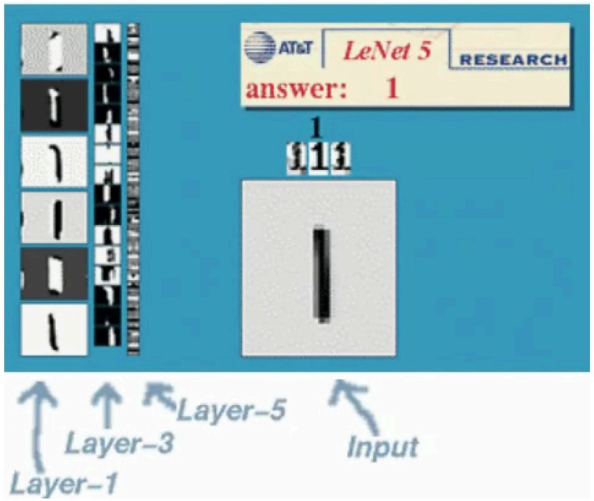
What math can be computed by *n*-layers of convolutions (without any activation function) ?

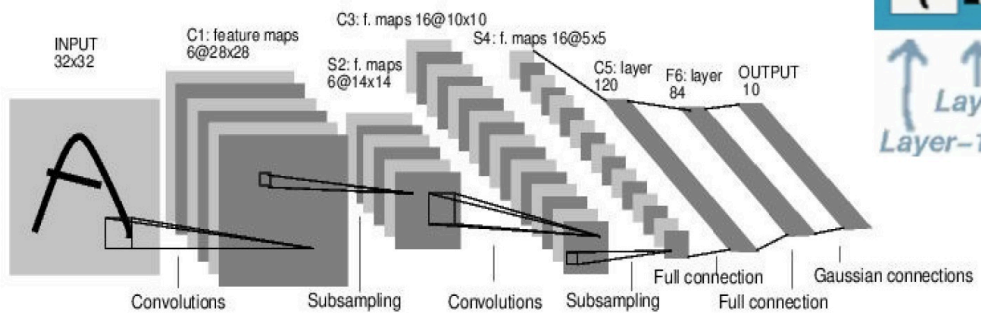
Convolutional Neural Networks

A bit of history:

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]





LeNet-5

RELU's

If the net consisted of convolutional layers only, the final output would be a convolution. To increase the functionality, a non-linear operation is needed. Many has been tried, RELU's have shown superior.

A RELU (REctified Linear Unit) is a simple non-linear operation with no parameters:

$$y = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Often RELU's are used after each convolutional layer (or after a sequence of such).

Class exercise - 0.5 minute

Why do you think a RELU is a (huge) computational advantage compared to e.g. a sigmoid ($= \frac{e^x}{1+e^x}$) ?

Class exercise - 0.5 minute

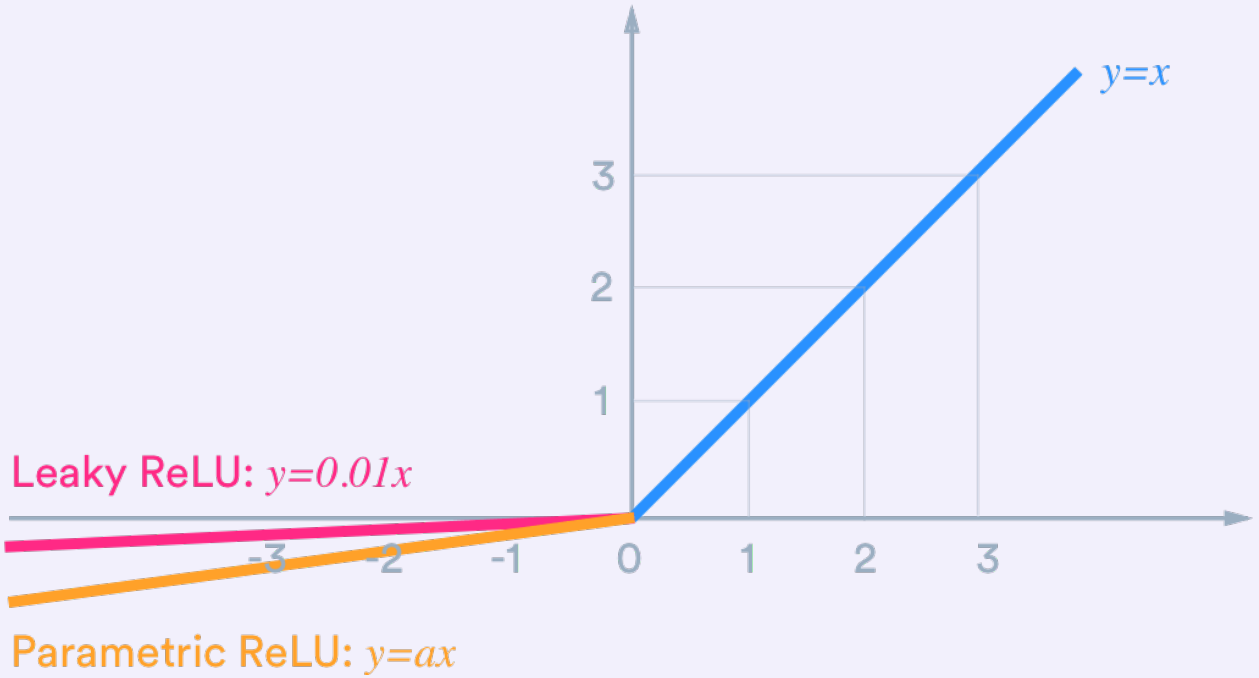
Why do you think a RELU is a (huge) computational advantage compared to e.g. a sigmoid ($= \frac{e^x}{1+e^x}$) ?

Remember the Back-propagation algorithm. What happens when we differentiate a RELU:

$$y = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Leaky RELU

A Leaky RELU does not map all negative values to zero, but let a tiny fraction pass.



There is consensus that RELUs are much better (and faster) than e.g. sigmoid-functions. There is no consensus about leaky RELUs.

Max-pooling

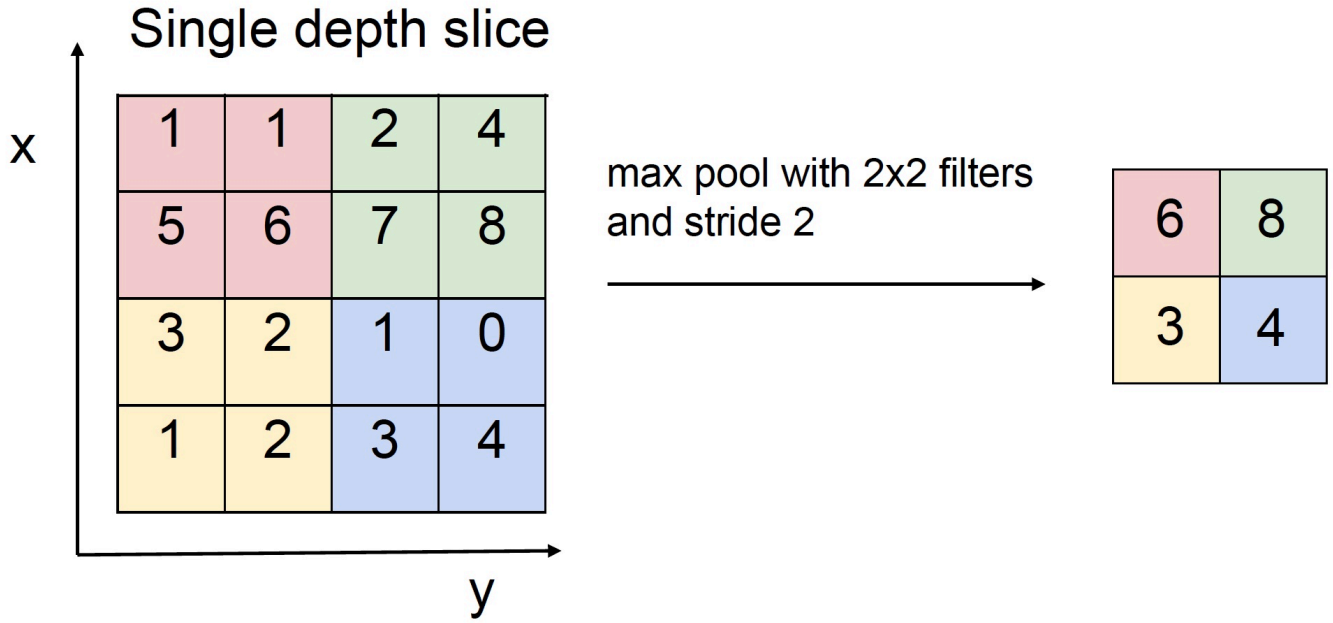
For many purposes, a resolution given by the input image is not needed. In object recognition an approximative position, size and aspect ratio of a bounding box may suffice.

To gather sufficient context information large filters are needed. These cost a lot in the number of parameters. Instead the images are downsampled (usually with a factor 2 in each direction). This may take place repeatably after each RELU. Thus the architecture becomes pyramid-like.

When four pixels are downsampled to one, a pooling operation must be applied. Most often this is a maximum.

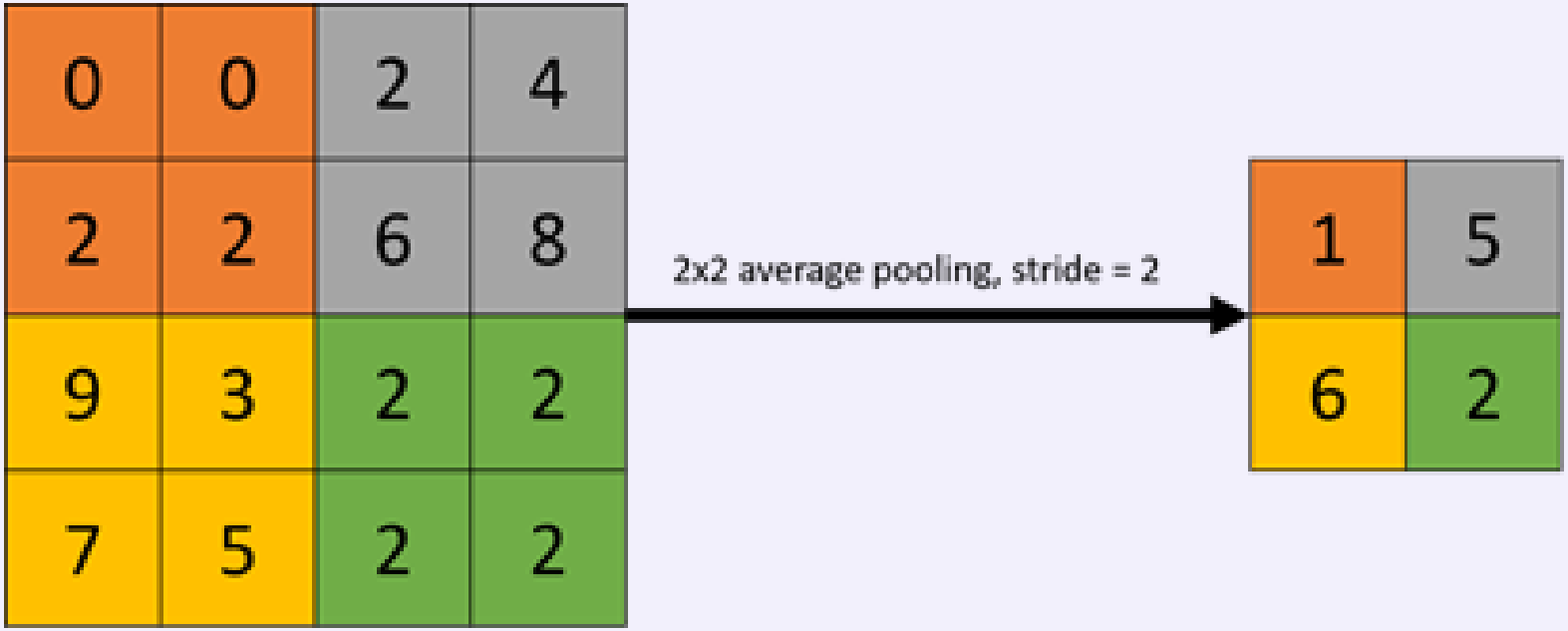
Pooling Layer

MAX pooling



Average pooling

An average is a linear operation and does not provide the non-linearity often needed in learning processes. However, in applications where counting/integration is essential, average pooling is possible.



Questions ?

Other layer types

In object recognition, the last 1-3 layers are fully connected (FC). This part of the netork thus resembles a traditional neural net, and often accounts for half the number of parameters (or more).

Another usefull layer type performs normalization (NORM). This may be done in many ways - one is to make sure all channels contribute with equal amount to the following layer.

Finally, to stabilize the learning of the FC-layers, often a DROPOUT layer is used. During training, this temporarily disconnects say half the neurons from contributing. DROPOUT was used a lot in early nets. Later architectures prefer BATCH-NORMALIZATION.

At the very end/top of most CNN's SOFTMAX-layers, CLASSIFICATION-layers, or REGRESSION-layers are usually applied. We will not discuss those layer types here.

Batch normalization

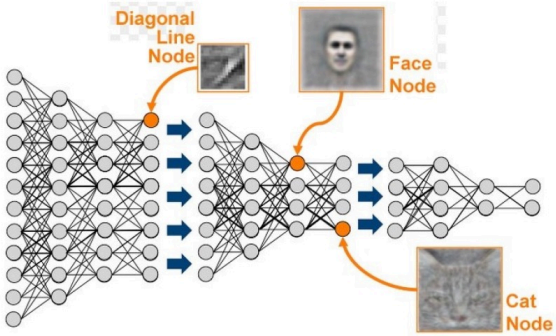
The purpose of Batch Normalization is to perform regularization. For each channel, it simply subtracts the mean and divides by the standard deviation.

$$BN_{\gamma,\beta}(x_I) \;=\; \gamma \left[\frac{x_I \;-\; \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}\right] \;+\; \beta$$

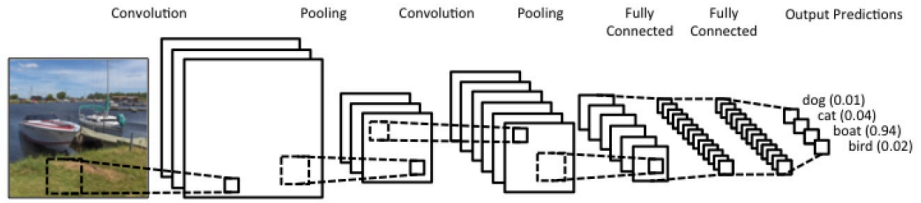
where (γ, β) are papameters that in the simple case may be $(1, 0)$, but otherwise are trainable.

Batch Normaliztion may often be applied immediately after all (or selected) convolutions.

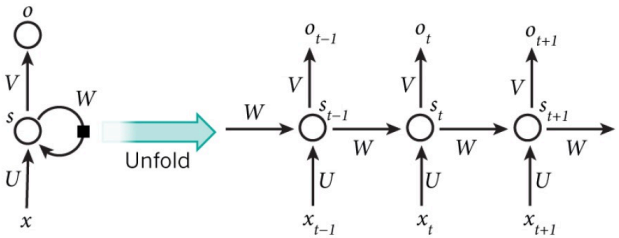
Deep Learning architectures



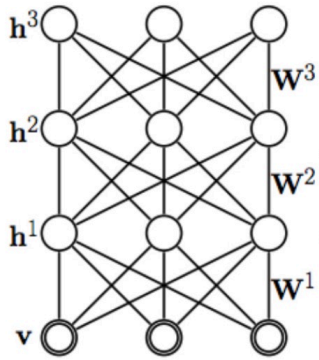
Deep Neural Network (DNN)
all fully connected layers

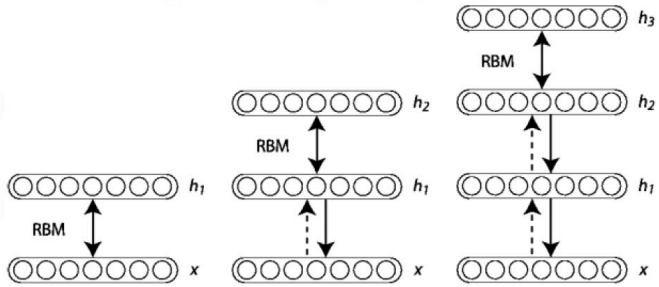


Convolutional Neural Networks (CNN, ConvNets)
include convolutional layers and pooling layers



Recurrent Neural Networks (RNN)





Deep Belief Networks (DBN)

Deep Boltzmann Machines (DBM)

It's so deep

Deep Learning is a name for methods that decompose the input in a hierarchy of gradually more abstract representations.

Deep learning, like Neural Nets is a more general term than CNN. It does not specify neither architecture, nor learning algorithm.

In the early 2010'ies, 5 levels were considered deep. Today this will be shallow. Modern nets often applies 100 layers or more. However, increased depth does not necessarily imply increased performance.

Putting it all together

Many CNN's are build using a set of fixed layers, here called a **level**. As example, we may construct a level by two $3 \times 3 \times k$ CONV layers, a RELU and a MAXPOOL layer.

Two 3×3 convolutions could as well be implemented using one 5×5 convolution, but the former use only 18 parameter where the latter use 25, an important saving. Modern CNN's tend to apply 3×3 CONV's only.

Now stacking a number of levels and adding a FC creates a simple CNN.

What is going on?

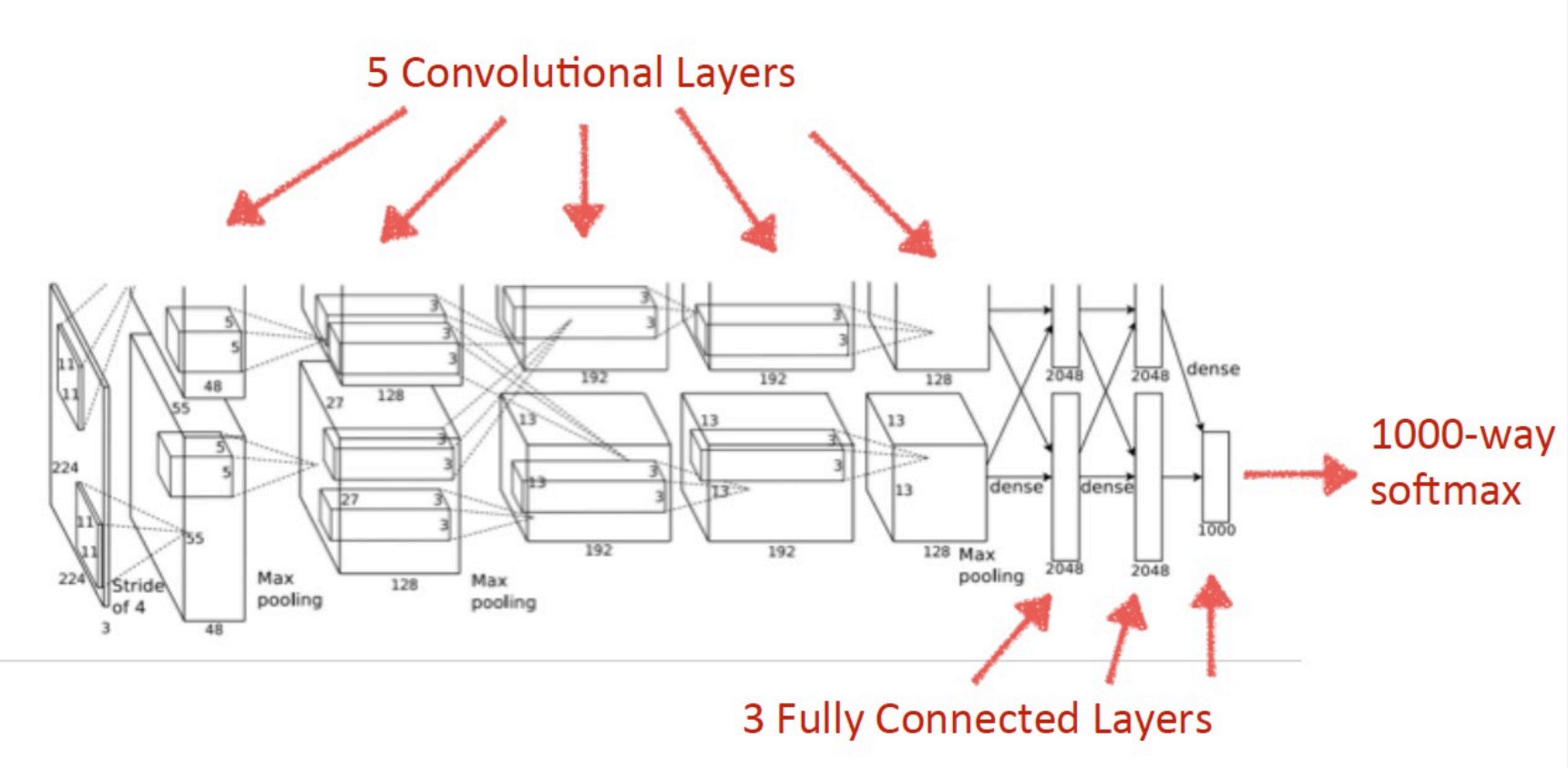
The purpose of the CONV-layers is to do feature extraction. Going up the levels gradually more and more abstract feature maps are build. Finally, classification may be done using af FC-layer.

A CNN may specialize the features to the task represented by the annotated images. This may involve a mix of texture and structural elements, that are not obvious to humans.

In certain restricted application areas, well-trained CNN's have outperformed humans. In other tasks they lack behind, and even the best CNN can only handle what it has been trained to.

CNN's

In general the CNN-architecture should be constructed according to the task that it is supposed to solve. Early nets, like Alex net are rather complicated and involves many parameters (60 Million for Alex Net).



Recent nets try to simplify and cut down the number of parameters.

Class exercise - 0.5 minute

Where in AlexNet (or many other nets) do you think that the major number of parameters are used ?

A simple CNN

Assume the Input size is $N \times N \times 3$:

level	operation	filter size	output sixe
1	Conv	$3 \times 3 \times 4$	$N \times N \times 4$
	Conv	$3 \times 3 \times 8$	$N \times N \times 8$
	RELU		$N \times N \times 8$
	MAXPOOL	2×2	$N/2 \times N/2 \times 8$
2	Conv	$3 \times 3 \times 16$	$N/2 \times N/2 \times 16$
	Conv	$3 \times 3 \times 32$	$N/2 \times N/2 \times 32$
	RELU		$N/2 \times N/2 \times 32$
	MAXPOOL	2×2	$N/4 \times N/4 \times 32$
3	Conv	$3 \times 3 \times 32$	$N/4 \times N/4 \times 32$
	Conv	$3 \times 3 \times 64$	$N/4 \times N/4 \times 64$
	RELU		$N/4 \times N/4 \times 64$
	MAXPOOL	2×2	$N/8 \times N/8 \times 64$
4	Conv	$3 \times 3 \times 128$	$N/8 \times N/8 \times 128$
	Conv	$3 \times 3 \times 128$	$N/8 \times N/8 \times 128$
	RELU		$N/8 \times N/8 \times 128$
	MAXPOOL	2×2	$N/16 \times N/16 \times 128$
5	DROPOUT		$N/16 \times N/16 \times 128$
	FC	M	M
	FC	K	K
	SOFTMAX		K
	CLASSIFICATION		1

Please note that only the CONV-layers and the FC-layers require parameters that must be found during training. Only the FC-layers depend on the image size N. Now lets count the number of parameters.

Number of parameters

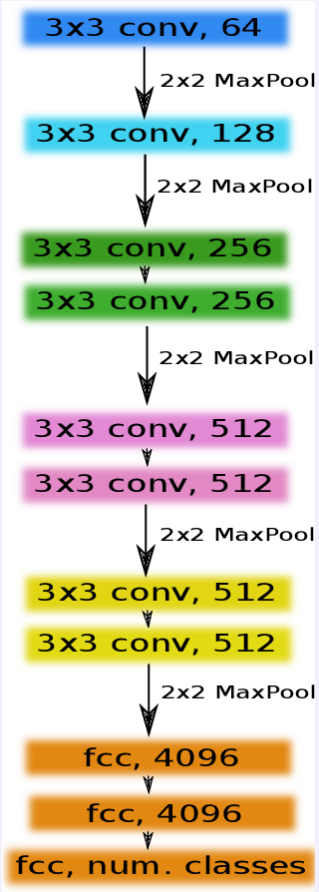
Say we have K classes and M neurons in first FC-layer

level	operation	filter size	parameters
1	Conv	$3 \times 3 \times 4$	$4 * 3 * 3 * 3 = 112$
	Conv	$3 \times 3 \times 8$	$8 * 3 * 3 * 4 = 288$
2	Conv	$3 \times 3 \times 16$	$16 * 3 * 3 * 8 = 1152$
	Conv	$3 \times 3 \times 32$	$32 * 3 * 3 * 16 = 4608$
3	Conv	$3 \times 3 \times 32$	$64 * 3 * 3 * 32 = 18.432$
	Conv	$3 \times 3 \times 64$	$64 * 3 * 3 * 64 = 36.864$
4	Conv	$3 \times 3 \times 128$	$128 * 3 * 3 * 64 = 73.728$
	Conv	$3 \times 3 \times 128$	$128 * 3 * 3 * 128 = 147.456$
5	FC	M	$(N/16)^2 * 128 * M = N^2 * M/2$
	FC	K	$K * M$

Feature levels contribute with a total of 282.640 parameters. Say $N = 200$ (small image patches), $K = 10$, and $M = 100$, then FC-layers amounts to $2.000.000 + 1000 = 2.001.000$ parameters. So in total we have about 2.28 M parameters where the FC-layers are responsible for most. To estimate all those parameters we need efficient machine learning and **a lot of data**.

VGG-net

VGG won the ILSVRG (Image Net Large Scale Visual Recognition Competition) in 2014. It has later been made popular in a range of variants (VGG11, VVG16, VGG19).



The Image Size

Note that in all CNN's the size of the input image is fixed. The size determines the number of neurons in the FC-layers.

Often context information (i.e. image size) is determining the performance. In general, when enlarging image size, the CNN depth as well as the number of channels should also be increased.

Thus increasing image size, increases the number of parameters, making learning harder and increases the risk of overfitting.

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Case Study: VGGNet

[Simonyan and Zisserman, 2014]

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: **224*224*64=3.2M** params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = **102,760,448**
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

Most memory is in early CONV

Most params are in late FC

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

Modern CNN's

The succes of AlexNet created a boost of research in particular within CV. Many far better nets have since 2012 been developed and the trend continues.

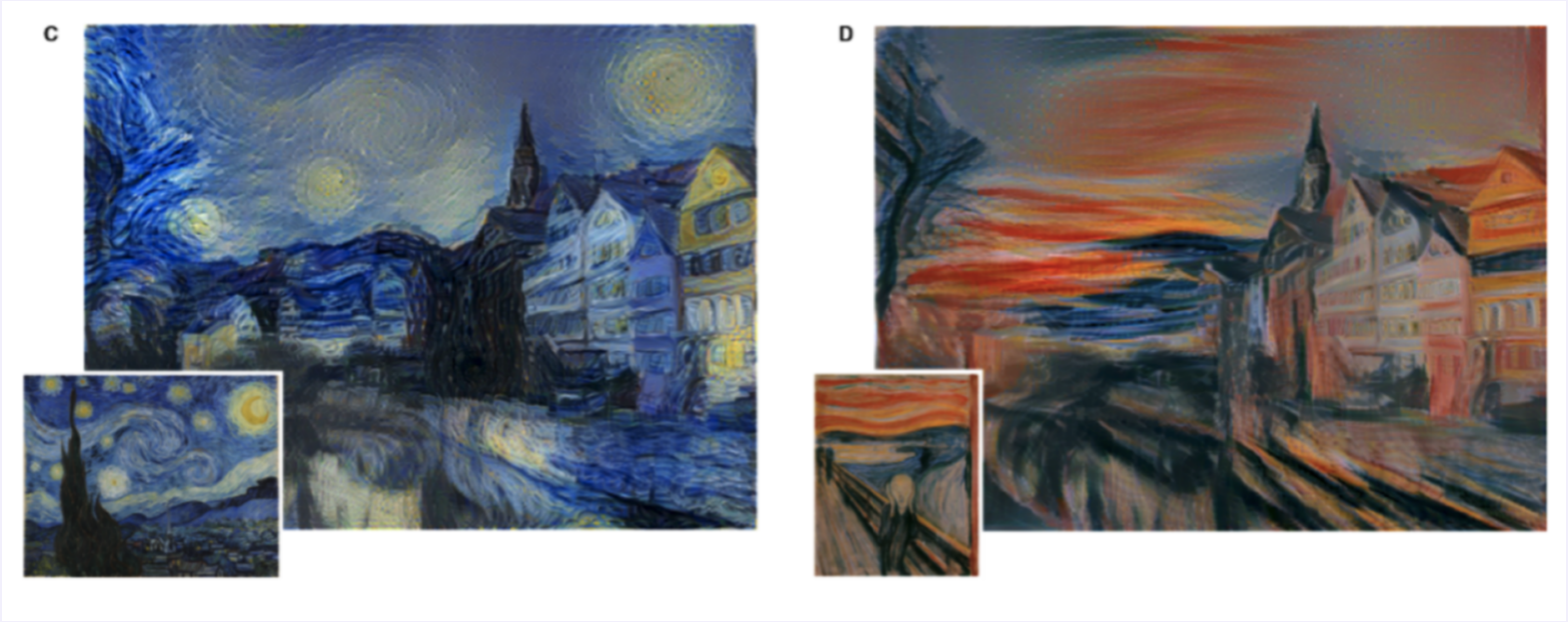
CNN's have shown superior in a number of areas including:

- Visual object recognition
- Image segmentation and scene interpretation

CNN's are inappropriate when shift-invariance cannot be assumed (and convolutions consequently does not apply). Also, they require **HUGE** amounts of annotated data, access to fast GPU's and days and weeks of training.

CNN in art 1

CNN in art 2



CNN for faking images

Next time

- What can a neuron compute ?
- Regularization, Optimization and Loss functions
- Pretraining and transfer learning
- A gentle introduction to the zoo of network architecture
- A bit on object detection and image segmentation
- Course evaluation