TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät für Elektro- und Informationstechnik, Professur für Grundlagen der Elektrotechnik und Elektronik

FliK Modul 2020

Convolutional Neural Networks

Steffen Seitz, Marvin Arnold & Markus Fritzsche

Prof. Ronald Tetzlaff

Dresden, 19-23.10.

Prequel
ML from a bayesian point of view

Prior:
Wahrscheinlichkeitsverteilung der Größe des Effekts, bevor man Daten untersucht

Posterior:

$$P(Effekt|Daten) = \frac{P(Daten|Effekt) \times P(Effekt)}{Konstante *}$$

Falls prior = **Gleichverteilung** („flat prior") = jeder Wert gleichwahrscheinlich

$$= \frac{P(Daten|Effekt)}{Konstante *}$$

Prior → „Basisraten-problem" in der Psycho-logie des Entscheidens:

Bild: weltbild.com

= Ergebnis frequentistischer Statistik, aber bessere Interpretation mit **transparenter** Annahme (man weiß nichts über den Effekt, bevor man die Daten sieht)

Konstante dient nur dazu, dass das Ergebnis als Verteilung interpretiert werden kann; sorgt dafür, dass „Wahrscheinlichkeit über alle möglichen β-Werte" = 1

© Michael Höfler TUD

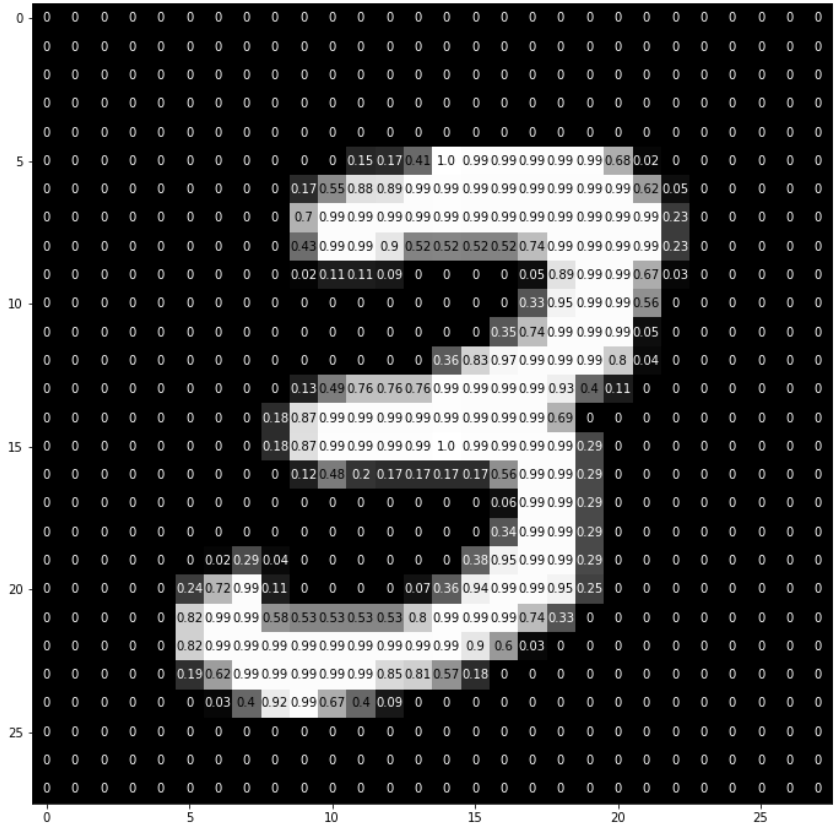
Likelihood

Neural Networks actually calculate (maximize) the **Likelihood**. Given Data → get **model** that best **explains** the data

In this context, the „Effekt" is aquivalent to the class wen would like to detect and **prior** can be seen as the **network architecture**.

If enough **data** is available (i.e. not the case), any **prior** will **vanish** out. But „**bad**" prior **don't exist**. (some just dont help much... :-))

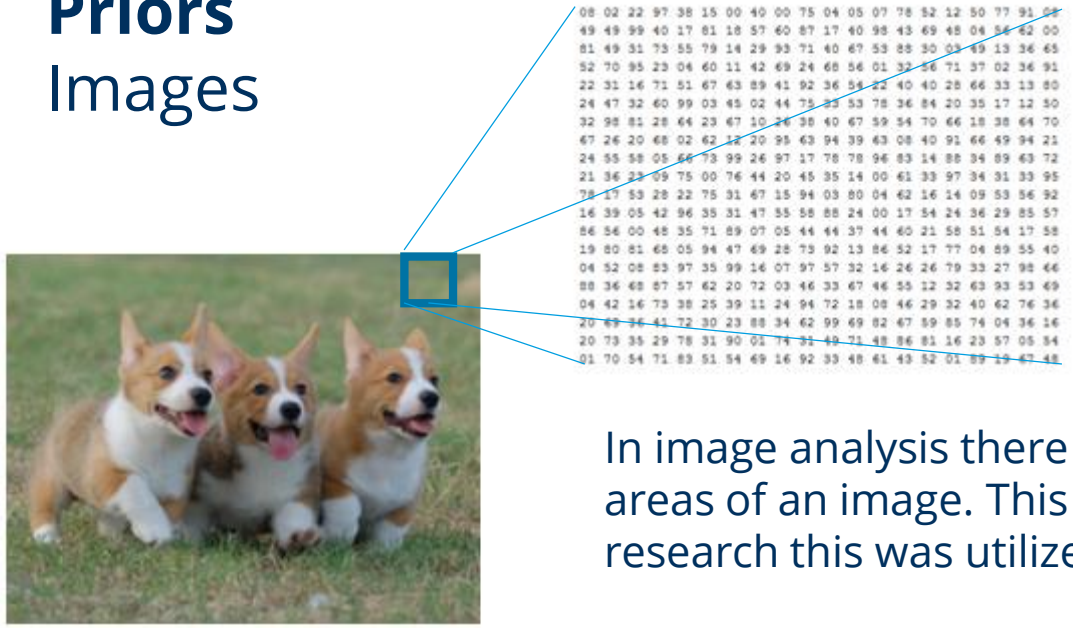
Image Data



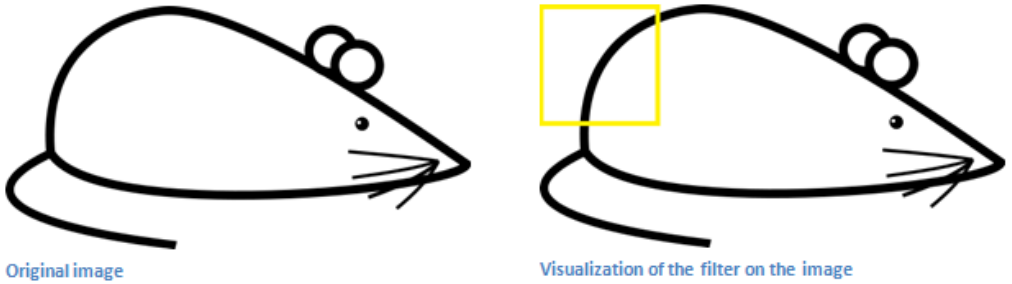
For a computer, images are just numbers. But there is one **prior** from years of image analysis that we can make use off:

Neighbour pixels are often correlated!

Priors
Images



In image analysis there is an algorithm that uses this effect to investigate specific areas of an image. This function is called (discrete) **convolution**. In classic CV research this was utilized together with (predefined) detectors or **filters.**



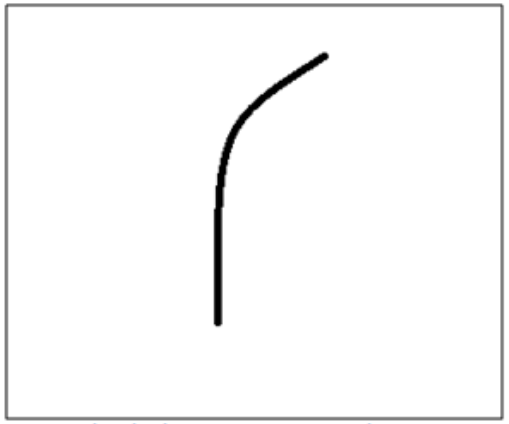
Example: *Sobel Filter* (Edge Detector)

Convolutional Neural Networks
Intuition

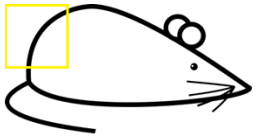
These filters can be **convoluted** with the **image** to **detect** the **feature** the filter was **designed** to detect by human image experts. In general a **filter** is just a matrix of numbers and the (discrete) convolution is simplified to a simple **vector-matrix** product – simple multiplication and summation.

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter



Visualization of the filter on the image

Example: *Detecting a specific line*

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Basically, in the input image, if there is a **shape** that generally resembles the curve that this **filter is representing**, then all of the multiplications summed together will result in a **large value**!

Convolutional Neural Networks
Intuition

Now let's see what happens when we move our filter.

Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

The value is much **lower**! This is because there was **nothing** in the image section that **responded** to the curve detector filter.

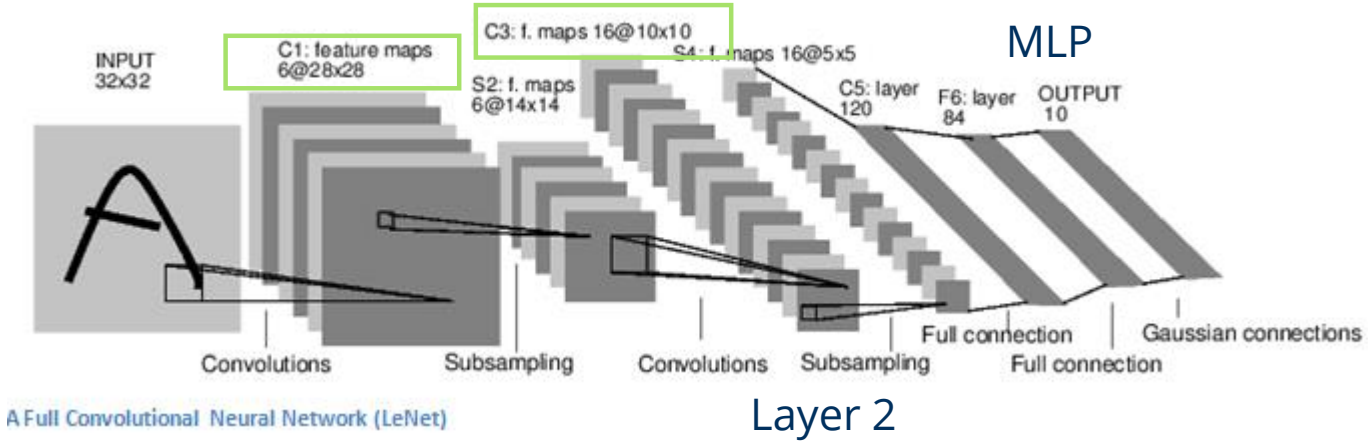
All previous filters we have seen, have been **handcrafted**. But the design of these filters is very **time consuming** and sometimes **not** even **possible** for human experts.

Idea: Use Neural Networks learn the filter!

Convolutional Neural Networks

The go to **algorithm** for **images**!

First approach: **LeNet** (Yann LeCun, 1998)



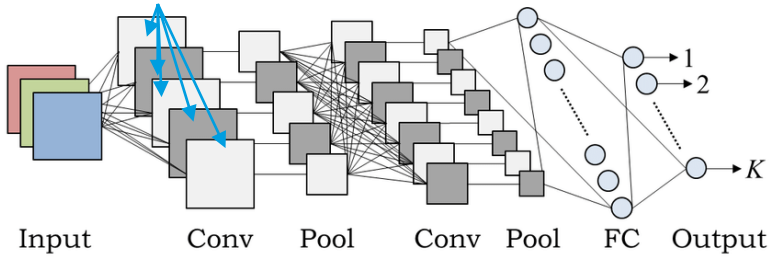
A Full Convolutional Neural Network (LeNet)

MLP

Layer 2

Simple:

feature maps

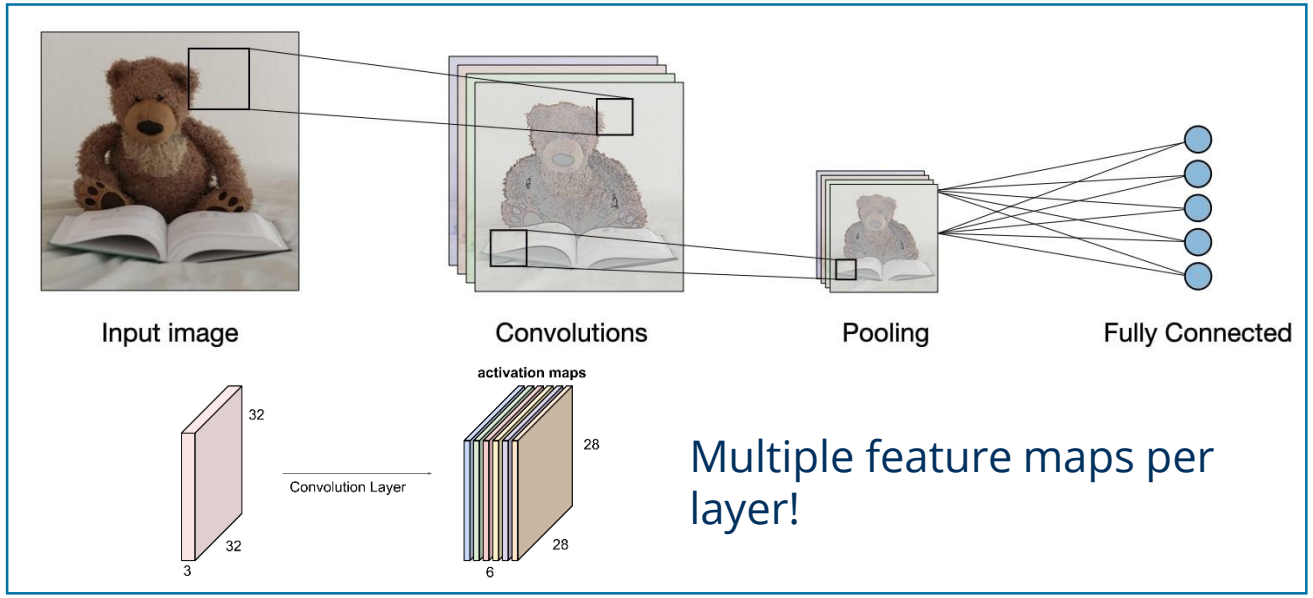


Layer 1 Layer 2

Interpret the **values** of our **filter** as the **weights** from our previous neural networks that we want to optimize. The forwards pass is exactly the already mentioned convolution of this initialized filter with our image. Usually multiple **featuremaps** are initialized and trained. This is **analog** to the **neurons** in the MLP case.

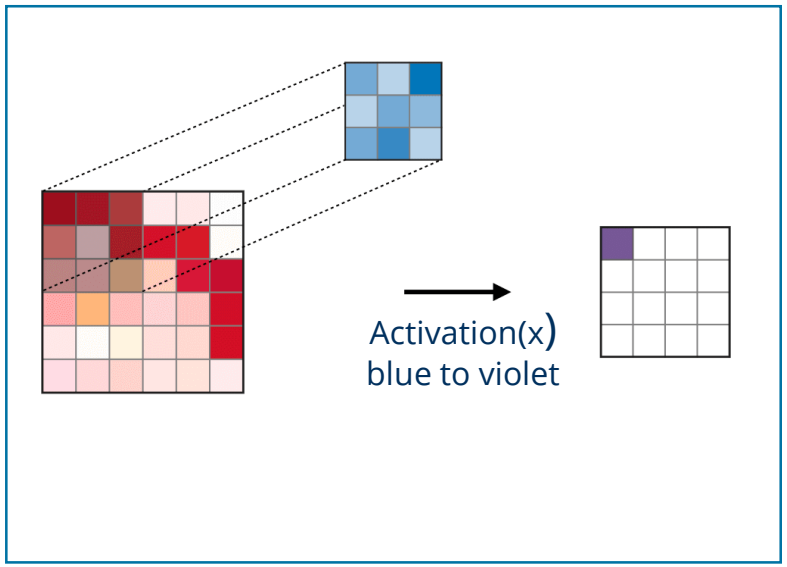
The feed forward path is the already mentioned **sliding convolution** of the filter over the image with two additions:

1) The **output** of the convolution = x is processed by an **activation function** (e.g. ReLu) before handing to the next layer.
2) After each Conv. layer, a **pooling crops** the **receptive field** for the next layer.

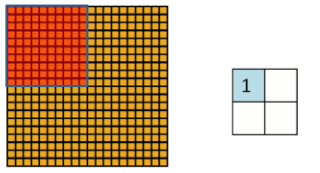


Multiple feature maps per layer!

Convolution

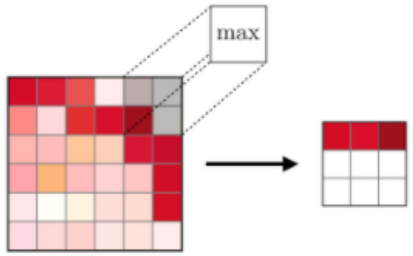
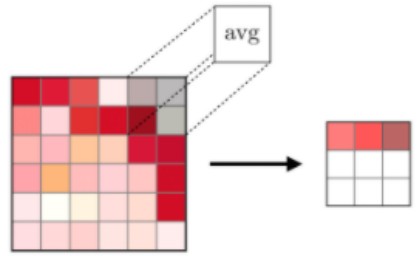


Pooling

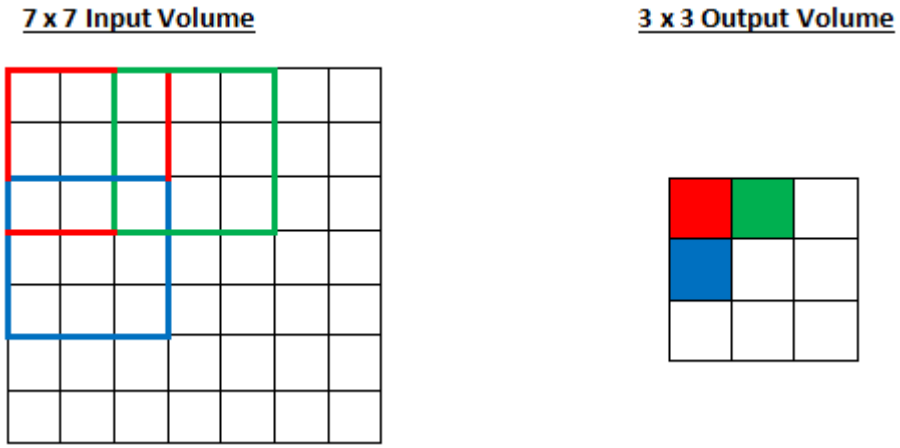


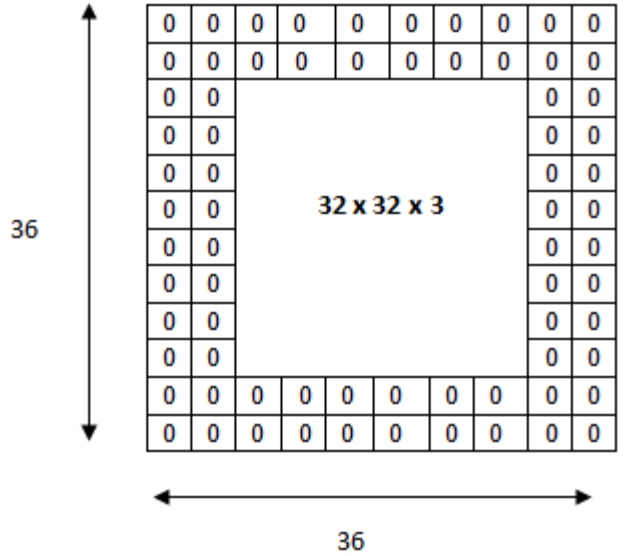
Convolved feature

Pooled feature

Type	Max pooling	Average pooling
Purpose	Each pooling operation selects the maximum value of the current view	Each pooling operation averages the values of the current view
Illustration		
Comments	• Preserves detected features • Most commonly used	• Downsamples feature map • Used in LeNet

Convolutional Neural Networks
Stride & Padding





For a convolutional or a pooling operation, the stride S denotes the **number of pixels** by which the window moves after each operation

Zero-padding denotes the process of adding PP zeroes to each side of the boundaries of the input.

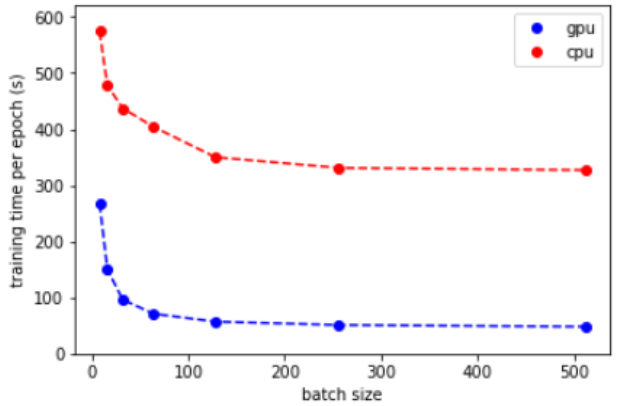
Cifar10

Our previous datasets are far to **easy** for Convnet to solve! Here, we will analyse Data from the **Cifar10** Dataset, a very well known image recognition **benchmark**.

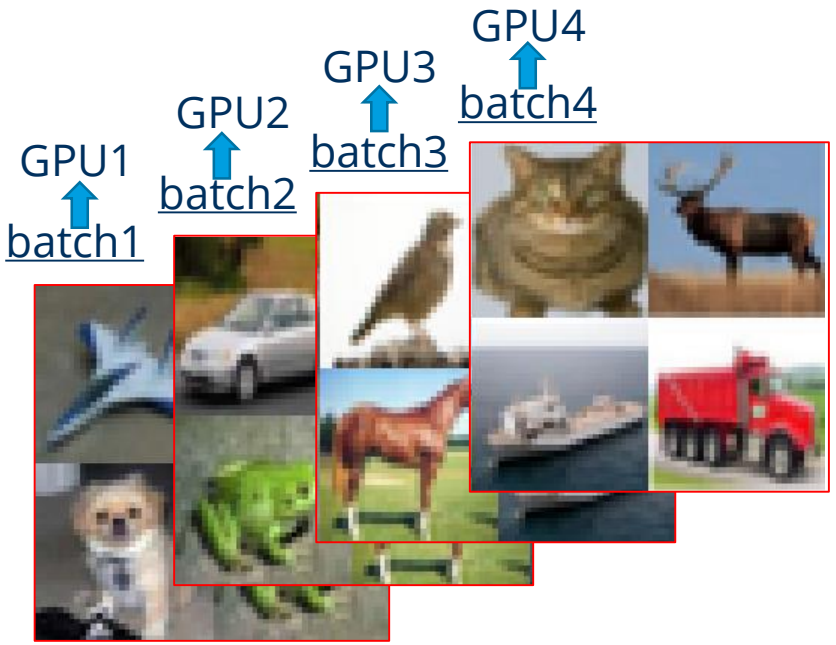
For brevity we will start with only 2 classes in the Dataset: **Cats** and **Dogs**.

Training **time** has become a **limiting factor** in AI research today. But thankfully, **subsets** of **sampels** in a training batch can be calculated in **parallel** on a GPU's:

Remember:
We update weights after each batch!)



32x32 pixel RGB images
10 classes
50,000 training images
10,000 test images

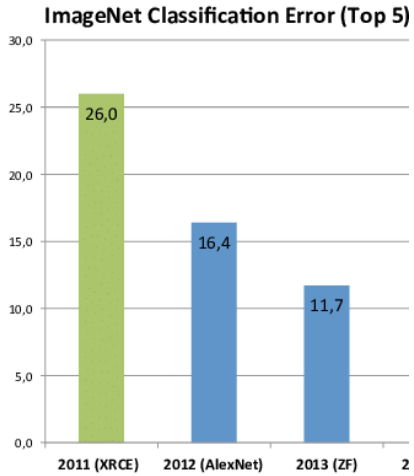


5. Exercise

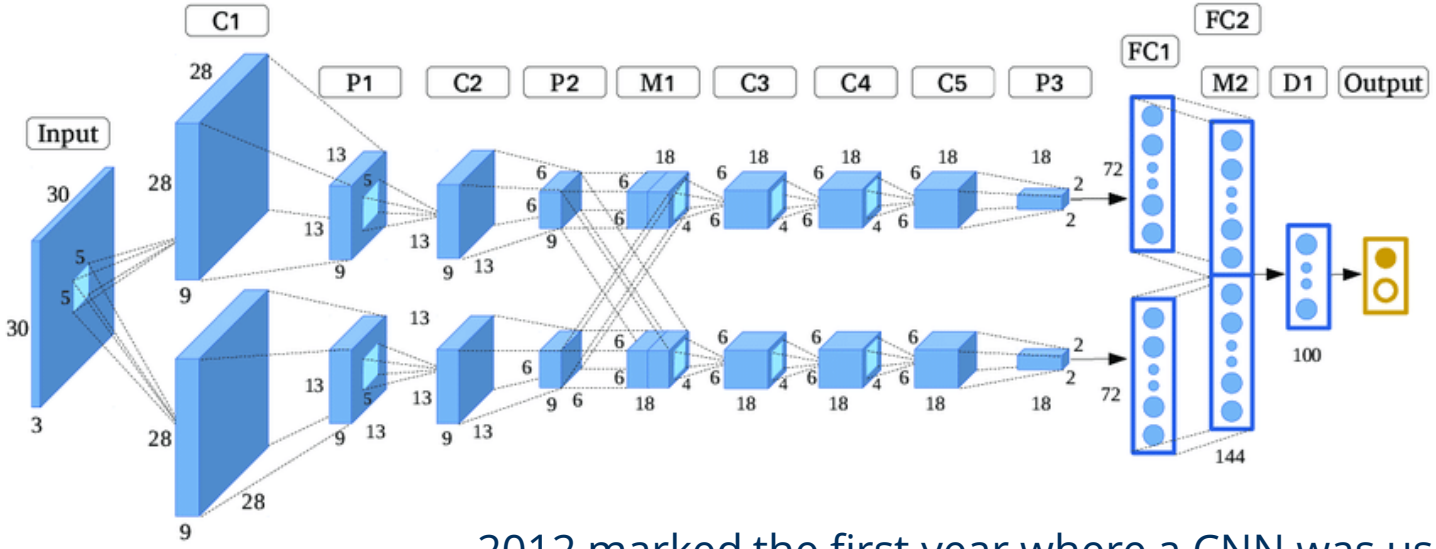
Let's train our first CNN classifier for cats and dogs with Keras!

ImageNet

Another famous dataset for benchmarking Image recognition algorithms. Tiny **ImageNet** dataset has 100,000 images across 200 **classes**.



First remarkebel CNN: **AlexNet** 2012



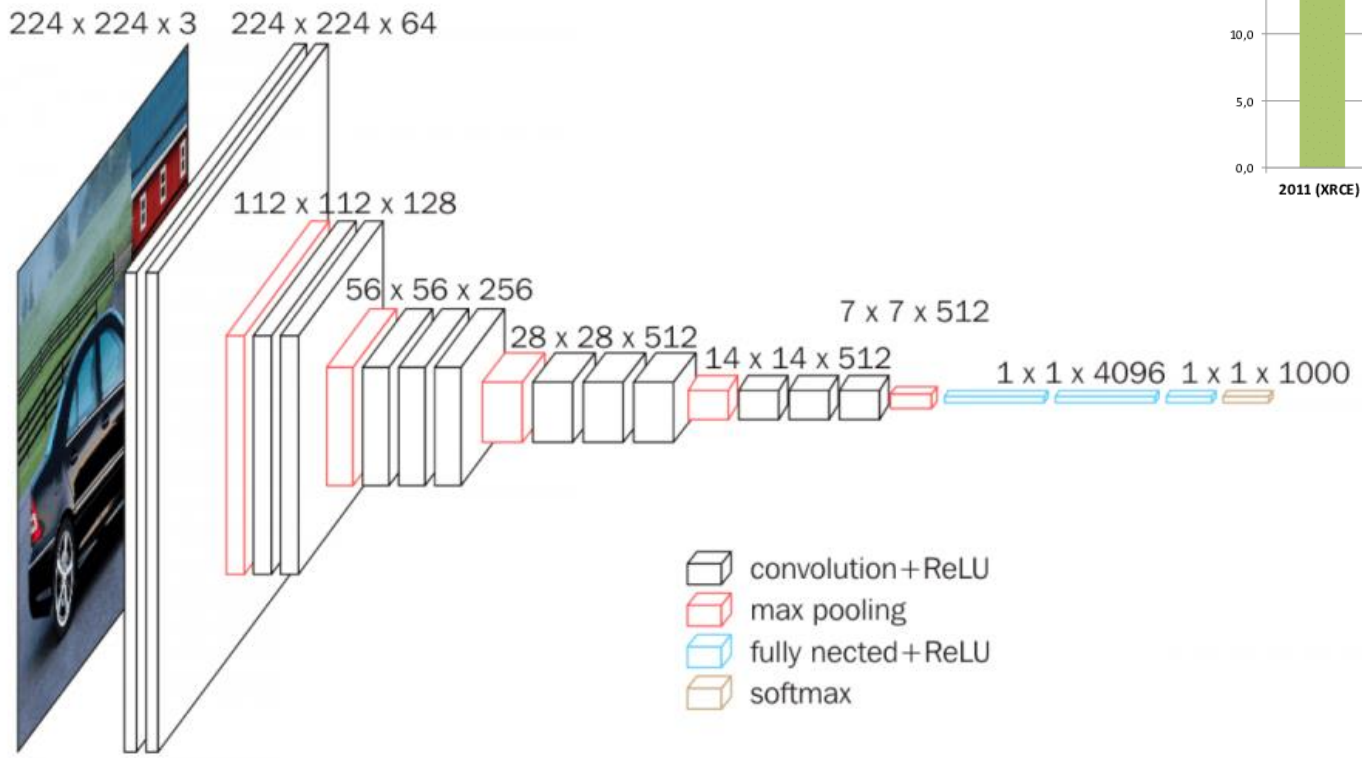
<u>Comparrison to LeNet:</u>

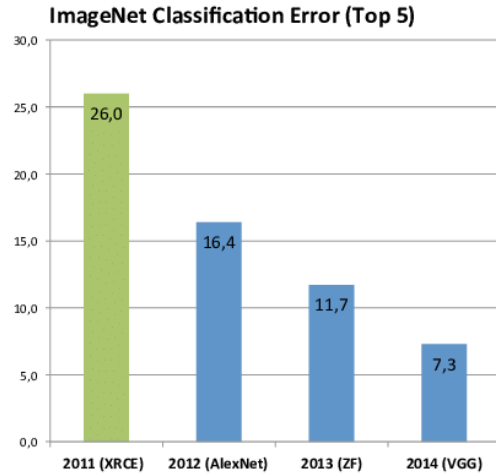
- 2 Paths possible
- X – connections
- Filters number increasing with layer

2012 marked the first year where a CNN was used to achieve a top 5 test error rate of 15.4%.

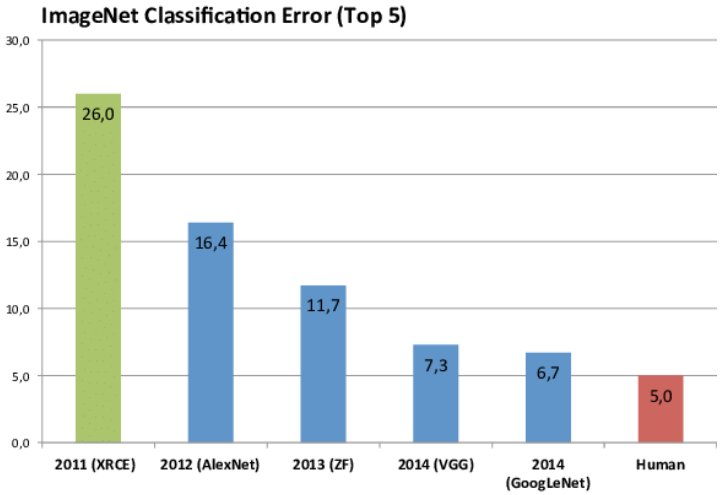
VGG16

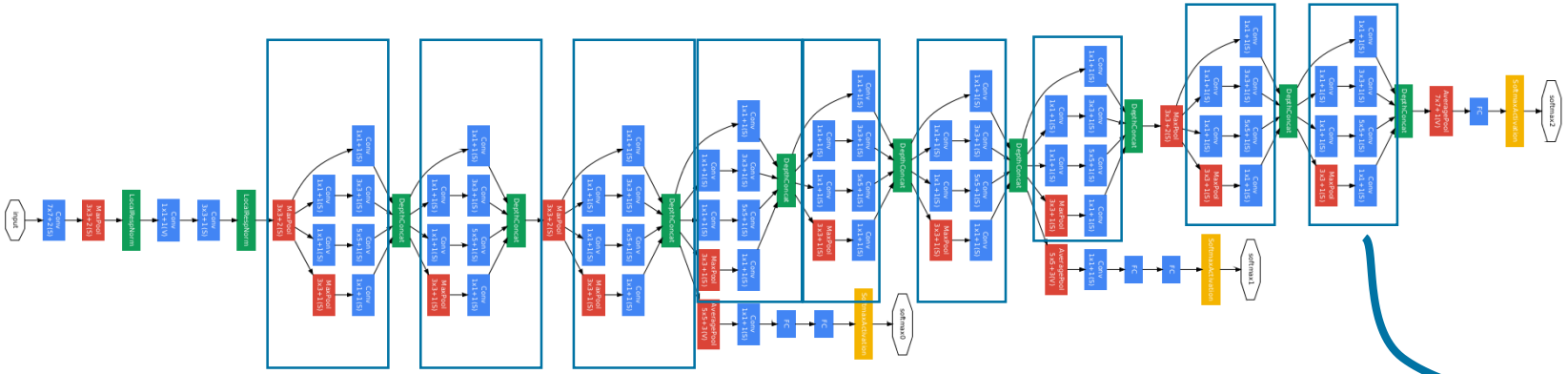
Winner of 2014 with 19 Layers!





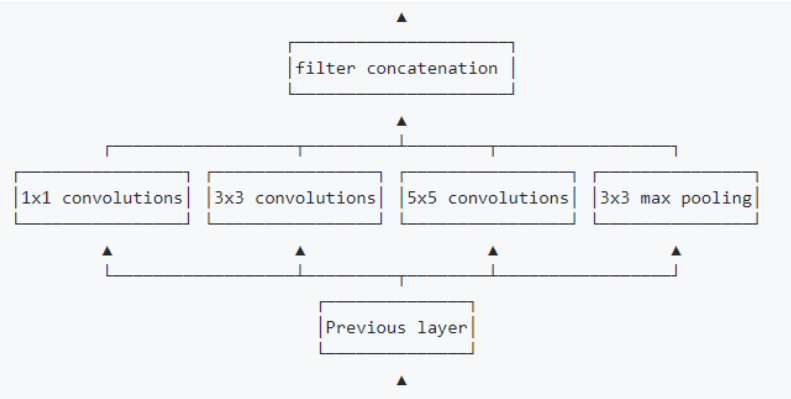
Google LeNet (Inception v1)





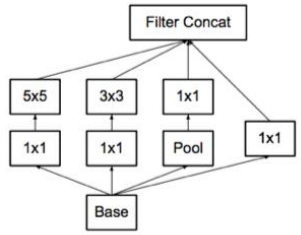
9 Inception modules, **22 layers** (blue)

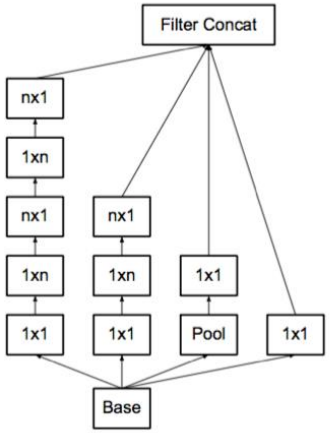
Inception module

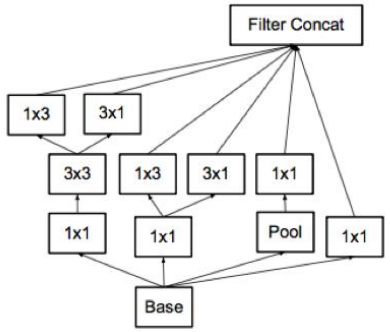


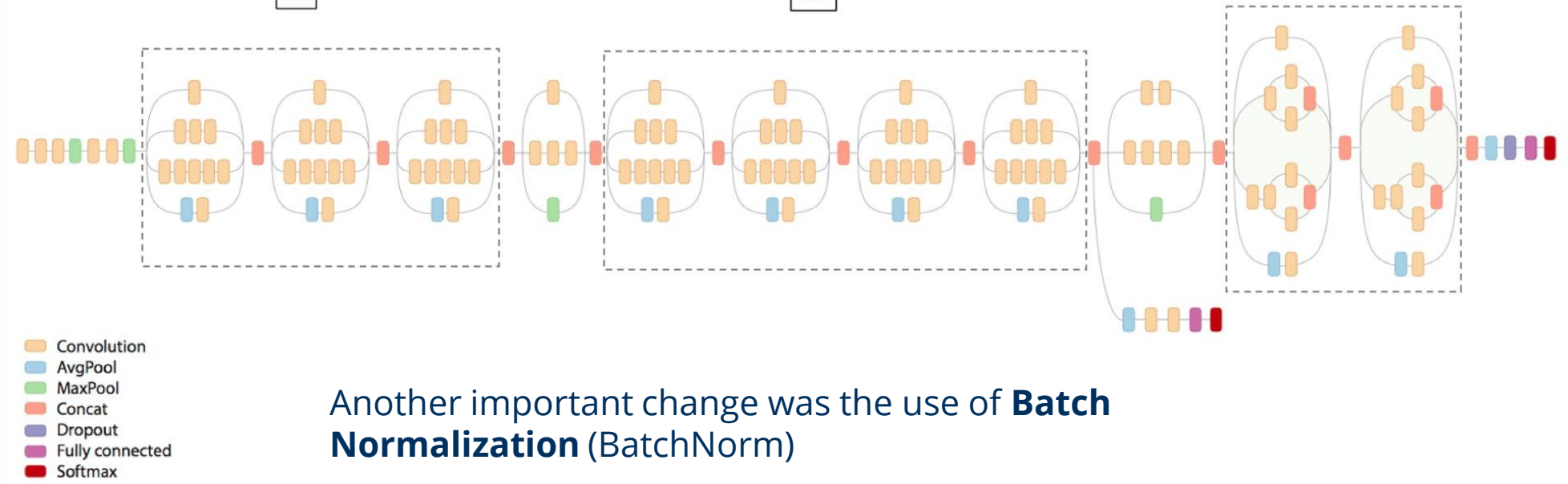
Google LeNet (Inception v2,3)

A 5x5 filter size is **2.78** times more computationally **expensive** than a 3x3 layer with the same number of filters!









Another important change was the use of **Batch Normalization** (BatchNorm)

Batch Normalization

Input

Conv

BN

Conv

BN

FC

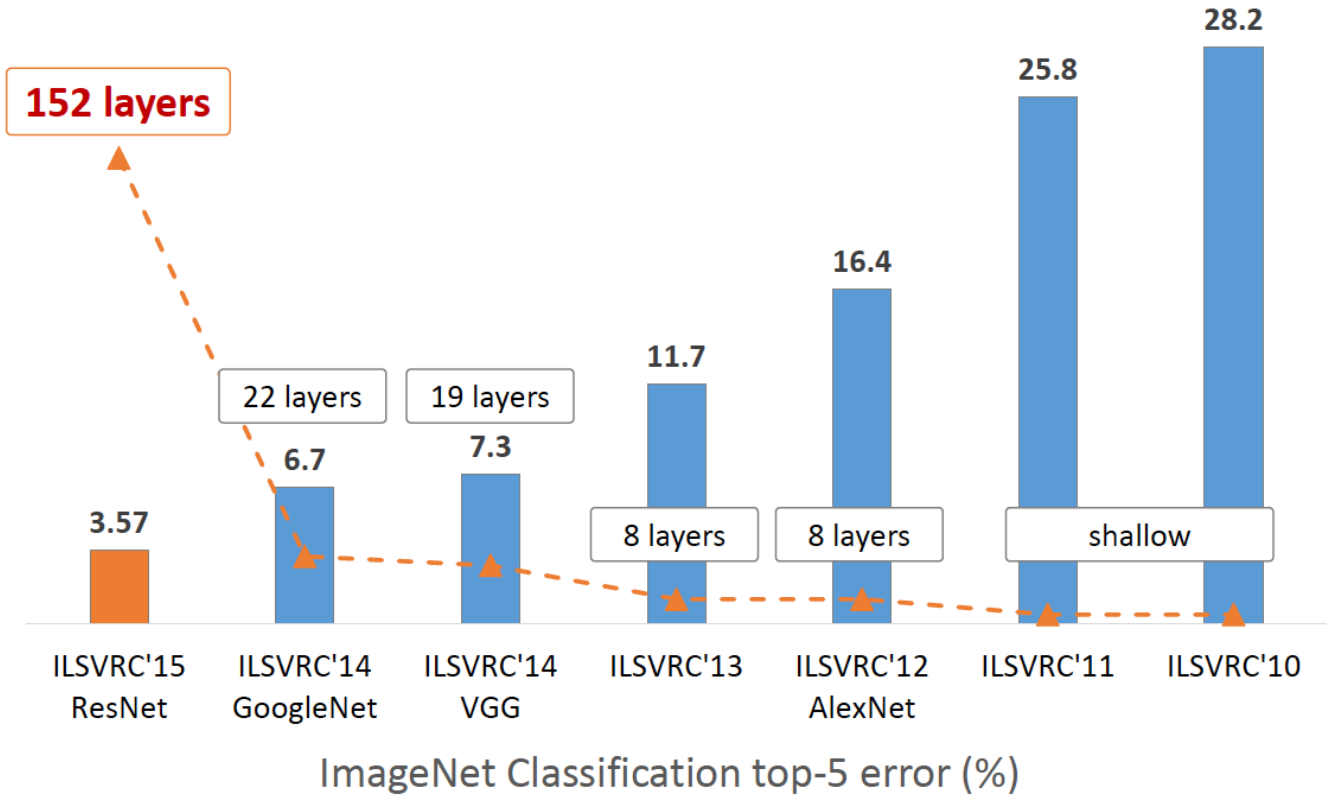
BN

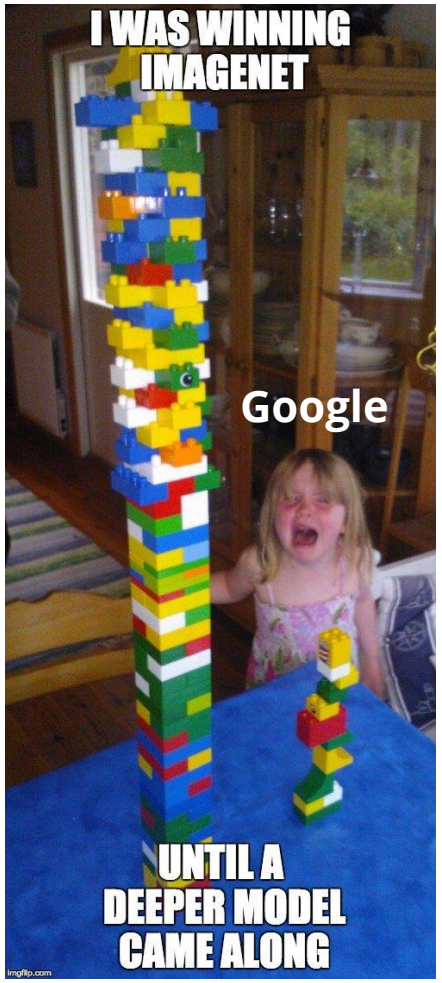
Softmax

Output

6. Exercise

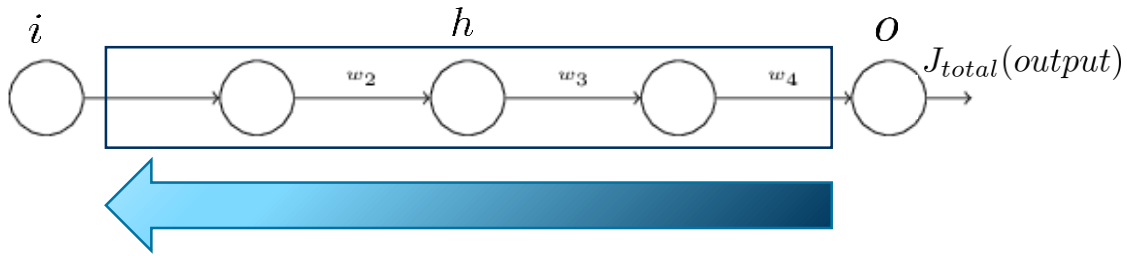
Let‘s regularize our first CNN classifier with BatchNorm!


152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

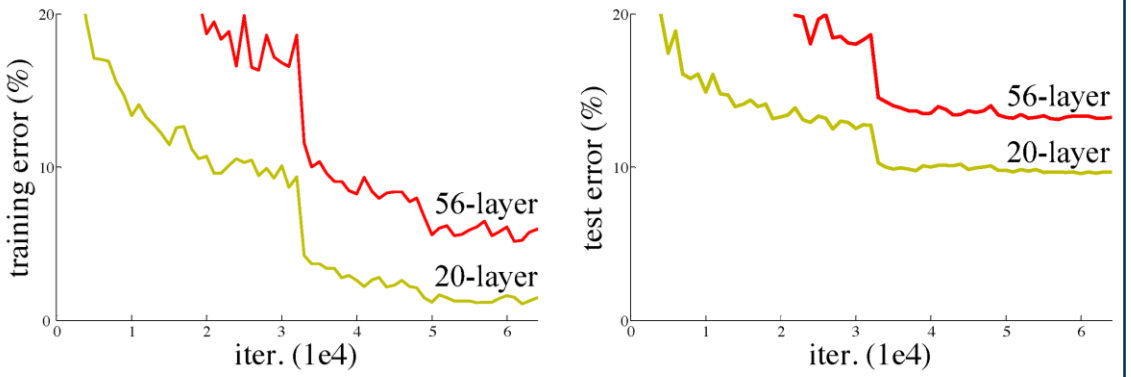

I WAS WINNING IMAGENET
Google
UNTIL A DEEPER MODEL CAME ALONG

Vanishing Gradient

Is learning better networks as easy as stacking more layers?



Cifar10 for deeper VGG Network

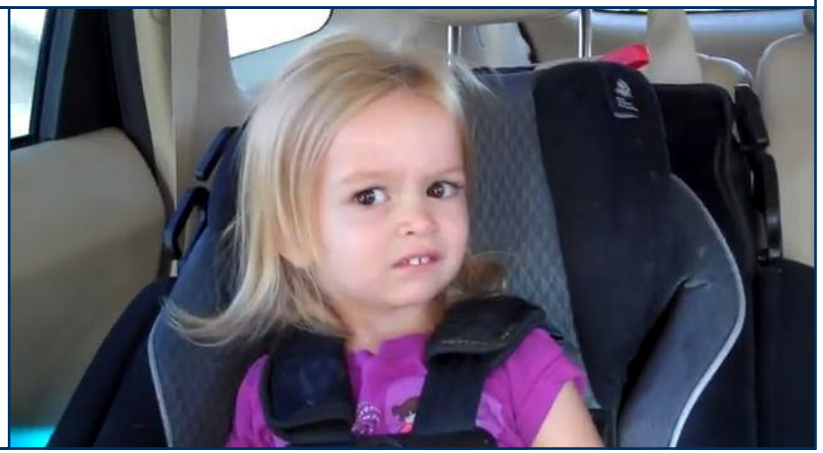


$$\frac{\partial J_{total}(output)}{\partial w_1} = \prod_{k=1}^{K} (w_i \underbrace{\sigma'(z_k)}_{\leq 0.25})$$

$K \ldots$ Number of layers

mostly: $|w_i| < 1$ → Glorot/He: $\sigma(w) = 1/n$, $\mu(w) = 0$

Regularization is also supposed to keep them small!

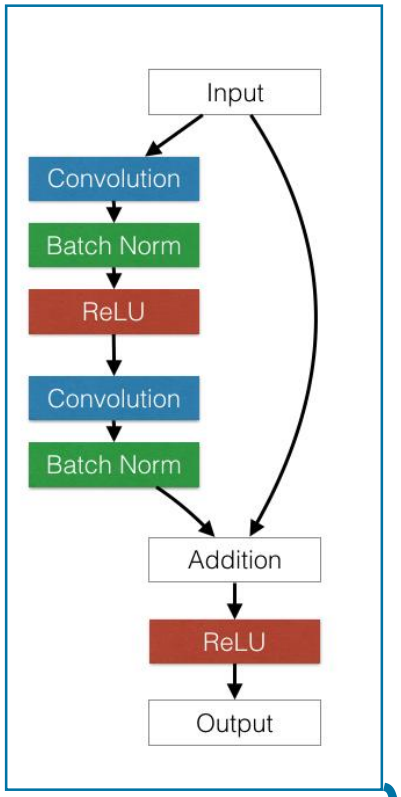


ResNet

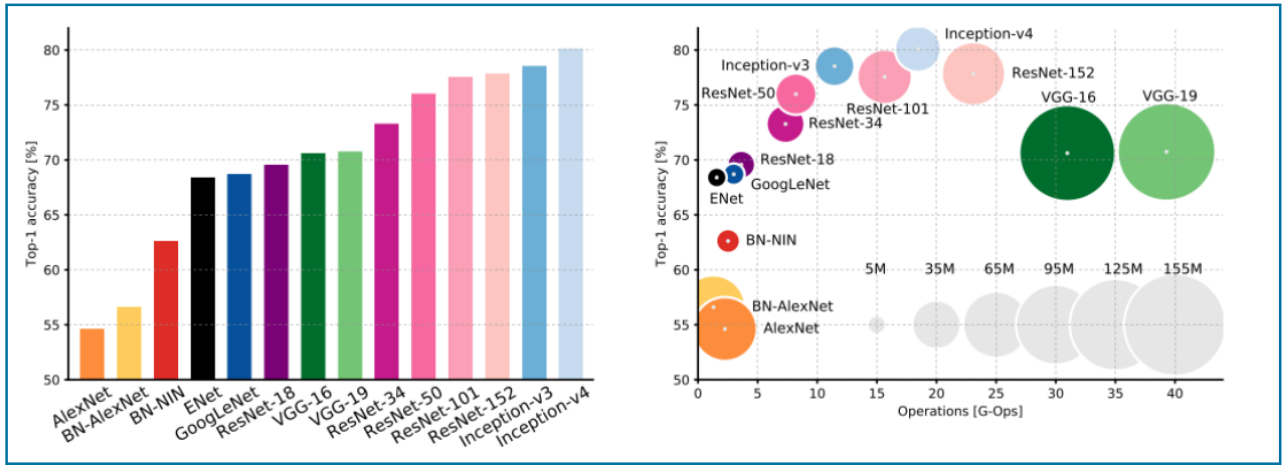
AlexNet, 8 layers
(ILSVRC 2012)

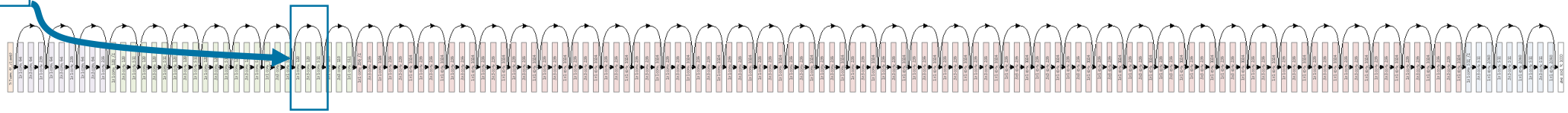
VGG, 19 layers
(ILSVRC 2014)

GoogleNet, 22 layers
(ILSVRC 2014)



ResNet introduced **skip connections** that allowed a flexible gradient highway. It is not a fix to vanishing gradient but it allowed models more **resilient**, so larger models can be trained!





ResNet, 152 layers
(ILSVRC 2015)

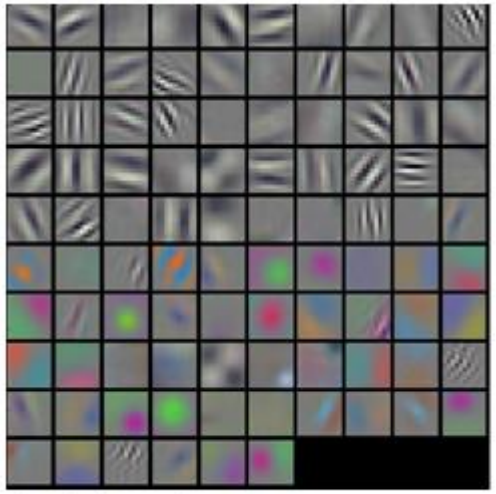
7. Exercise

Let's train a ResNet an other CNN Classifiers on the full Cifar10 dataset!

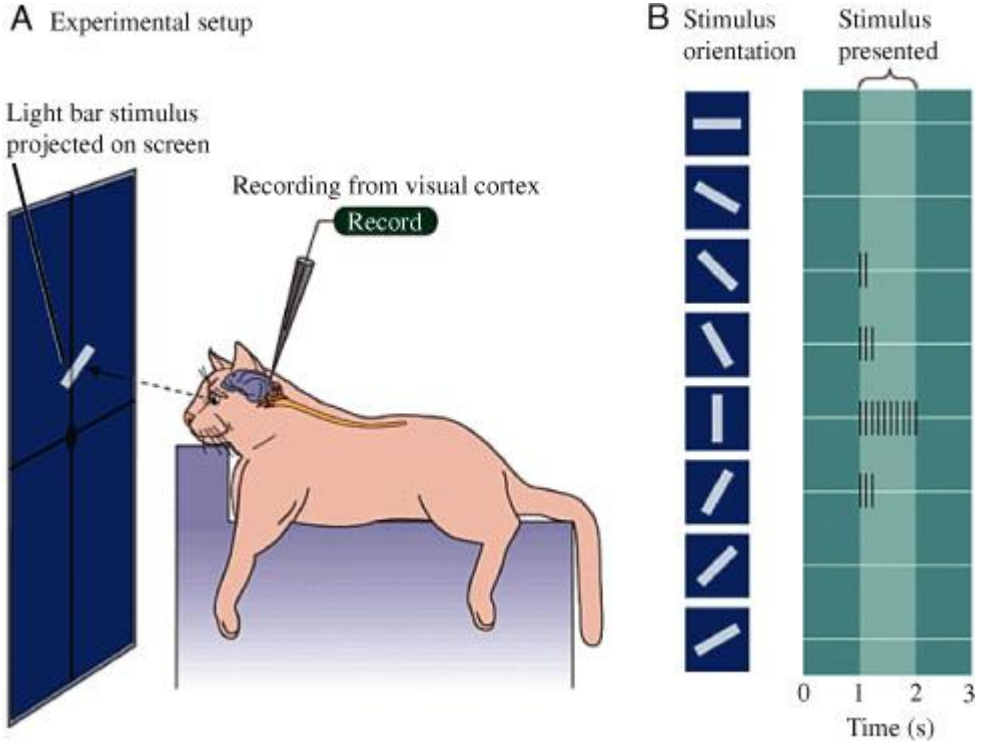
Feature Visualization
Activation Map plotting

Now that we know the basics, we can **plot** the output of the **feature maps activation** given an input image. This is a test how the respective layer is reacting and gives a hint what the network might have learned while training.

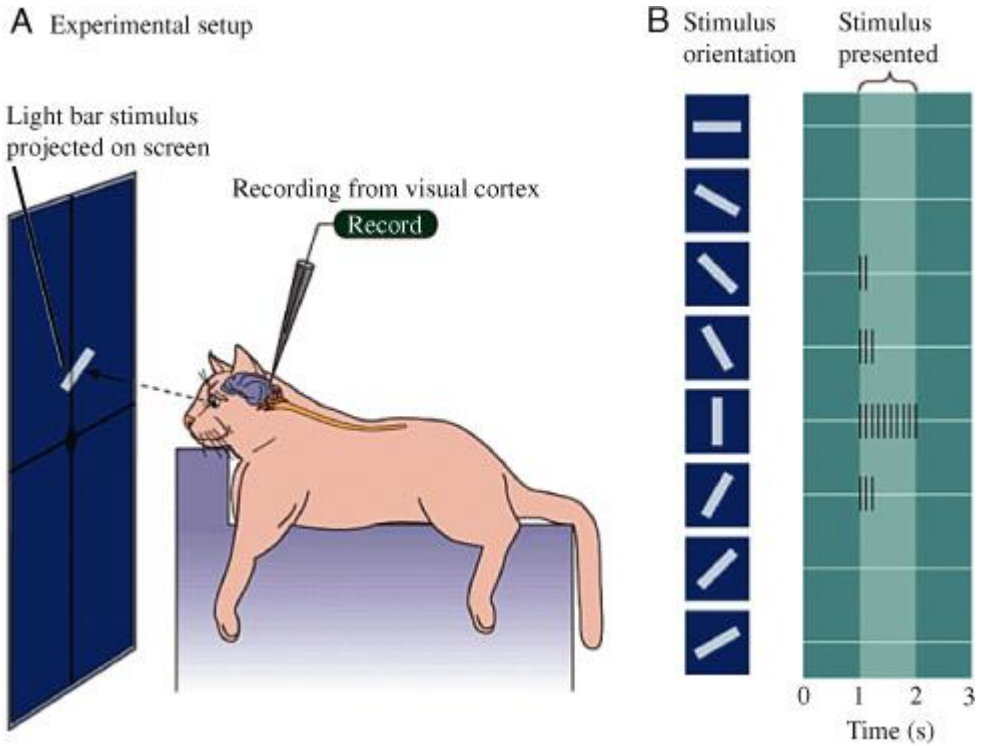
The results are fascinating, especially if you compare them to the experiment from **Hubel** and **Wiesel** 1962.



Visualizations of filters



fascinating experiment by Hubel and Wiesel in 1962


A Experimental setup
Light bar stimulus
projected on screen
Recording from visual cortex
Record
B Stimulus
orientation
Stimulus
presented
0
1
2
3
Time (s)

8. Exercise

Let‘s plot our activation maps of the CNN classifier with Keras!

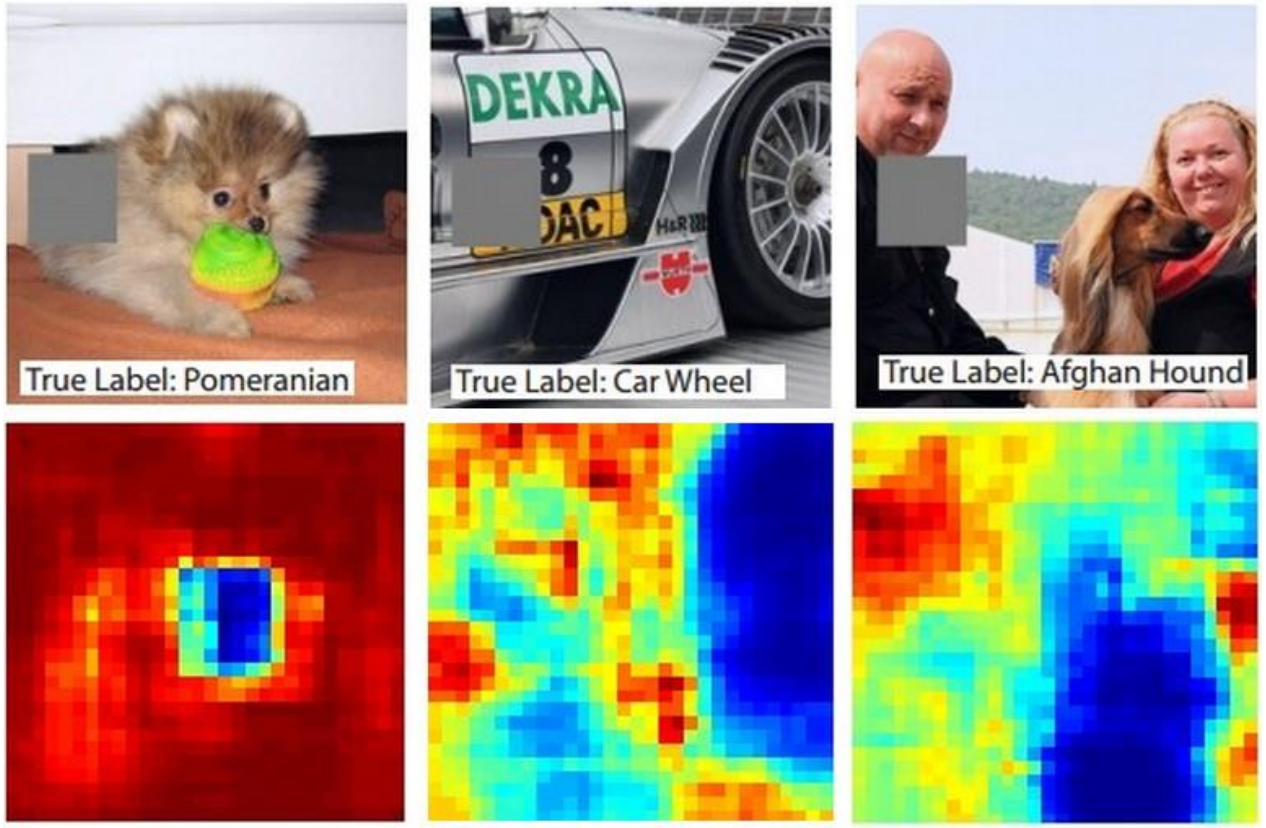
Feature Visualization
Occluding parts of the image

Matthew Zeiler:
Visualizing and Understanding Convolutional Networks (2013)

What part of an Image is **relevant** to a Neural Networks decision?

A native approach to this is the **occlusion** of specific sections in an Image.

Here we **iterate** over regions of the image, set a patch of the image to be all **zero**, and look at the **probability** of the class.

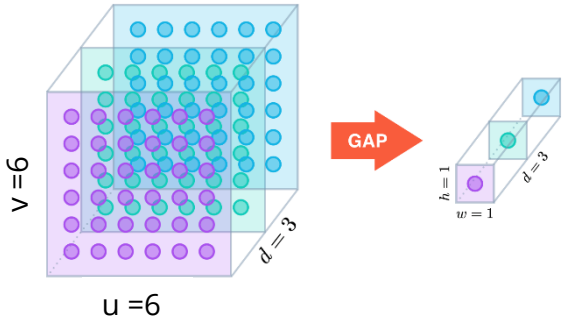


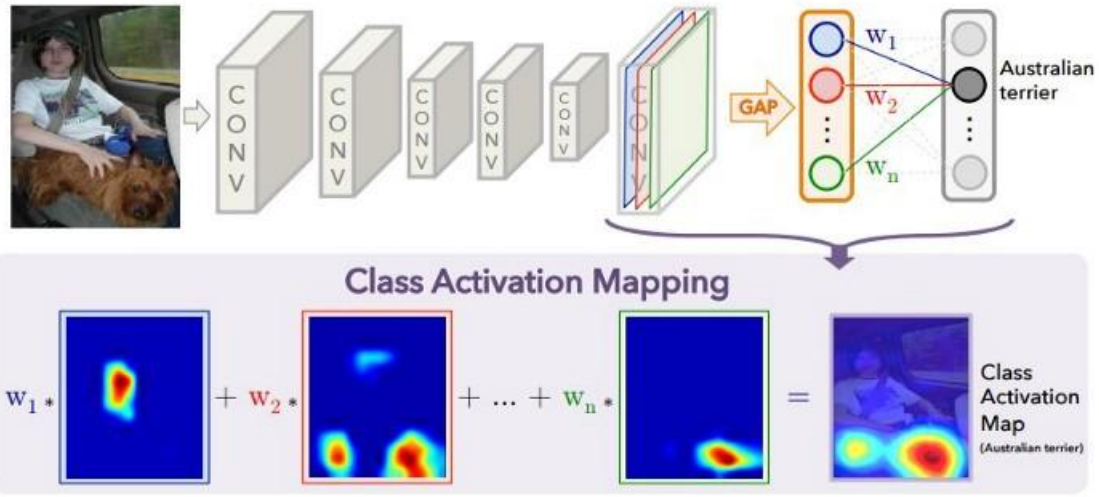
Class Activation Maps (CAM)
CAM and GradCam

We have seen that Convolutional feature maps **retain** spatial information, which is **lost** in fully-connected layers (MLP). **Last Conv Layer** can be thought as the important **features** for the classification.

→ Classificate from these feature Maps after using **GAP** and **sum** the **weighted positive** feature Maps =CAM (make them positive by ReLu(x))
→ Compute the GAP pooled gradient of the last layer

Global Average Pooling (GAP)





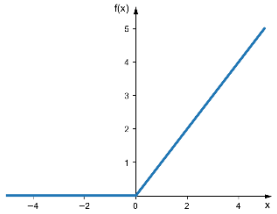
Number of FMaps
Weight (Cam)

$$L^c_{Grad-CAM} \sim= \sum_{k=1}^{K} \alpha^c_k A^k$$ = CAM

Feature Map

$$\alpha^c_k = \frac{1}{uv} \sum_{i=1}^{u} \sum_{j=1}^{v} \frac{dy^c}{dA^k_{i,j}}$$ GAP of Gradient (GradCam)

ReLu



$$L^c_{Grad-CAM} = ReLU\left(\sum_{k=1}^{K} \alpha^c_k A^k\right)$$ = GradCAM

ReLu because we are only interested in the **features** that have a **positive influence** on the class of interest

9. Exercise

Lets train a feature visualisation algorithm! (GradCAM)