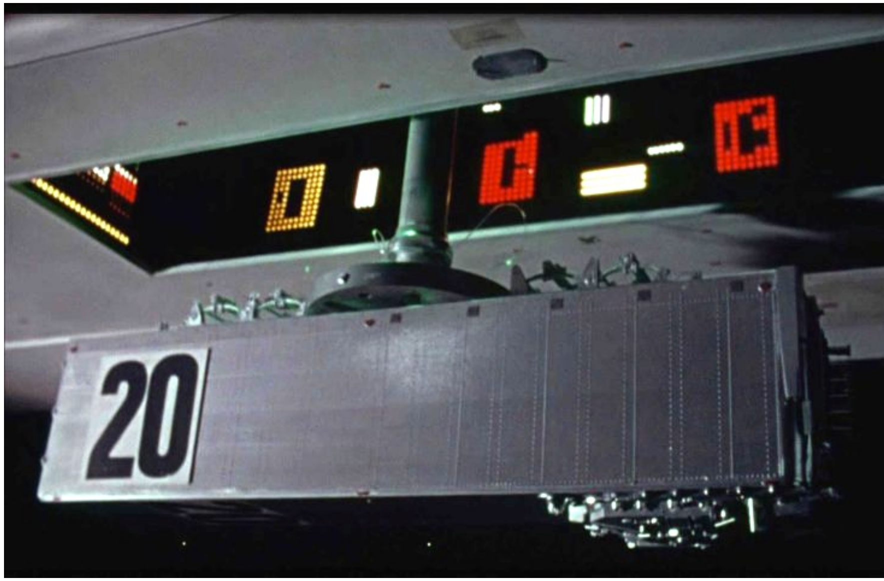
Convolutional Neural Nets II

Hands On



Oliver Dürr

Datalab-Lunch Seminar Series

Winterthur, April 22nd, 2015

Outline

Zurich University
of Applied Sciences

- Motivation for CNN
  - Focus here on image classification
- Frameworks
  - Caffe / **Lasagne**
- Recap MLP / **Demo MLP**
- Recap CNN / **Demo CNN**
- Some Tricks
  - Dropout
  - Training and Test augmentation (Learning Symmetries)
- Demo **CNN with Augmentation**

Code for demos: https://github.com/oduerr/dl_tutorial

History, milestones of CNN

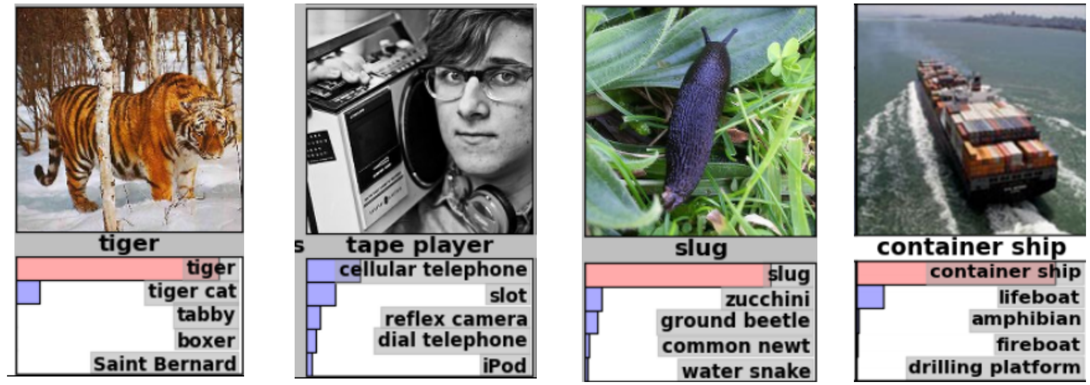
Zurich University of Applied Sciences

- 1980 Kunihiko Fukushima introduction
- 1998 Le Cun (Backpropagation)
- Many Contests won
    - 2011& 2014 MINST Handwritten Dataset
    - 201X Chinese Handwritten Character
    - 2011 German Traffic Signs
- ImageNet Success Story
  - Alex Net (2012) winning solution of ImageNet…

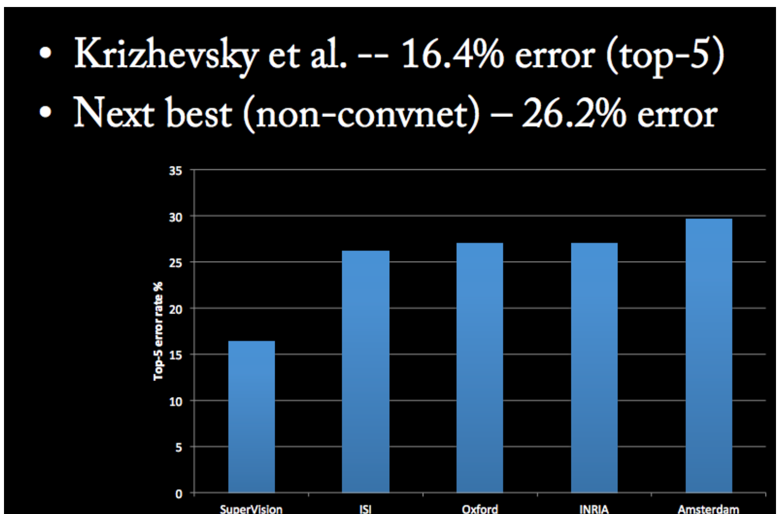
Imagenet 2012,2013,2014

Zurich University
of Applied Sciences

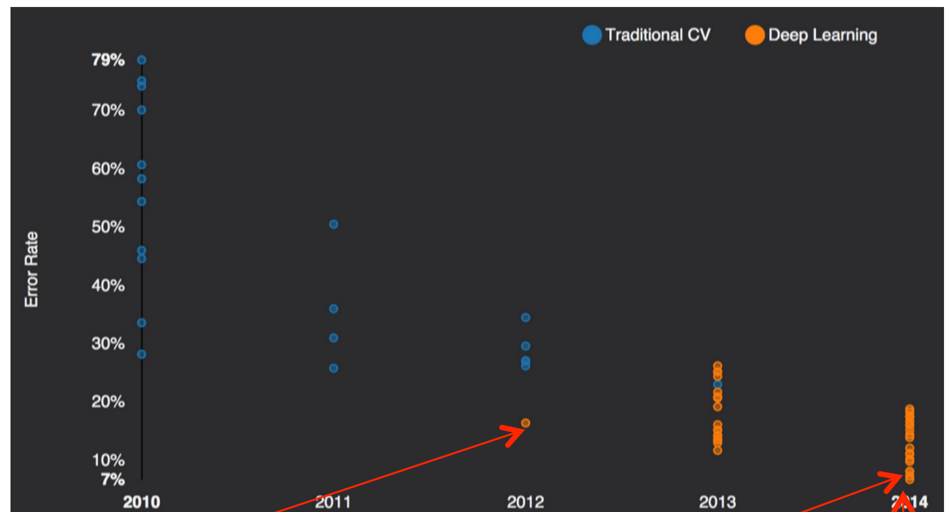
Some Examples
With Alexnet results
1000 Classes



2012



2010-2014



SuperVision
AlexNet 7 layers deep

GoogLeNet 6.7%

Oxfordnet up to 19 layers

http://cs.nyu.edu/~fergus/presentations/nips2013_final.pdf
https://medium.com/global-silicon-valley/machine-learning-yesterday-today-tomorrow-3d3023c7b519

A really convincing fact

Kaggle Plankton Competition (2015)

https://www.kaggle.com/c/datasciencebowl/leaderboard/public

Apps PiVi Run Marathonplan unter Wetter: Kreuzlingen WhatsApp Web Keep » Other Bookmarks

249	↓67	plankton_78	0.994107	23	Tue, 03 Mar 2015 21:03:05 (-3.8d)
250	↓67	Evan Cater	0.995265	18	Thu, 29 Jan 2015 20:48:08 (-21.1h)
251	↓66	I do not use CNN	0.997750	38	Sun, 01 Feb 2015 17:50:52 (-0.4h)
252	↓65	Alepf	0.999827	7	Tue, 10 Mar 2015 17:19:22 (-3d)
253	↓65	rm	0.999885	20	Tue, 24 Feb 2015 16:52:59

There is another bold one

634	↓70	NO-CNN	2.988988	4	Thu, 15 Jan 2015 04:25:27

Frameworks

Overview of frameworks

Zurich University
of Applied Sciences

Disclaimer: „This is a fast changing field. The list is not exclusive“.

Survey from the partcipiants of the Kaggle Challenge (Feb 2015)

Most Mentioned

- **lasagne / nolearn**
  - Python based on Theano, very flexible (winning team „Deep See“ used it)
- **caffe**
  - C++ based library, python bindings, convenience functions, many existing / pretrained models

Also used

- Theano (plain vanilla)
  - Symbolic computation of gradients and construction of numerical C-code
- Torch, lua (used by Facebook)
- Cxxnet
- ...

*https://www.kaggle.com/c/datasciencebowl/forums/t/12839/curiousity-survey-add-upvote-your-toolkit

Caffe

- C++ library with python bindings
- Settings (network layout and others) via files
- Documentation: poor
- Feels like a Blindflug
- Up to data components
  - Alexnet,GoogLeNet
- Input: Images or strange DBs
- Data Augmentation: Not possible from python
- Predefined models available

Lasagne / nolearn

- Lasage: python using theano (library define, optimize, and evaluate mathematical expressions on GPU)
- Nolearn is a wrapper around lasagne to provide a similar interface then scikit-learn
- Documentation: poor, but use the source luke
- Feels like you understand /control bells and whistles
- Custom components possible (provided Theano has functionallity)
- Input: Any numpy arrays
- Data Augmentation: Easy
- No predefined models (yet)

We will focus on lasagne

Links for Caffe

We will focuss on Lasagne but some links to Caffe for reference (thanks to Gabriel)

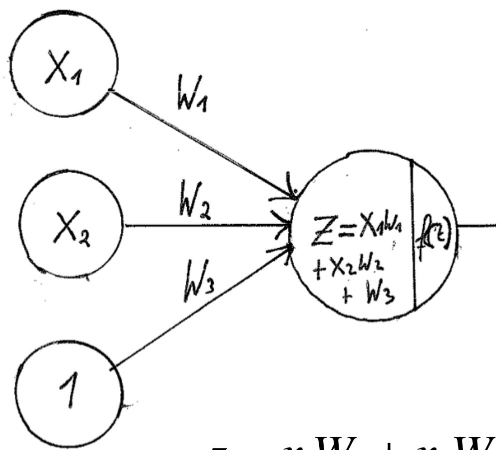
- https://github.com/oduerr/dl-playground/tree/master/python/FaceCaffe
- http://vision.princeton.edu/courses/COS598/2015sp/slides/Caffe/caffe_tutorial.pdf
- http://vision.stanford.edu/teaching/cs231n/slides/caffe_tutorial.pdf
- https://docs.google.com/presentation/...

Recap Neural Nets

Recap Neural Networks: Basic Unit

Zurich University
of Applied Sciences

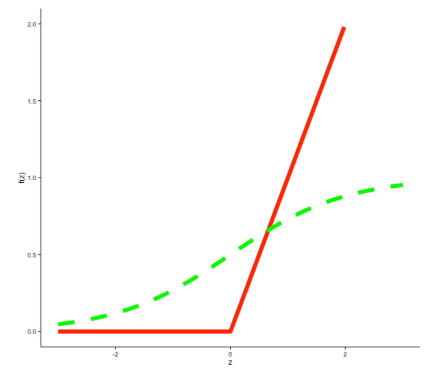
N-D log Regression



$$f(z) = \begin{cases} \dfrac{\exp(z)}{1+\exp(z)} \\ \max(0,z) \end{cases}$$

$$z = x_1 W_1 + x_2 W_2 + W_3 = \theta^T x$$

Activation Function a.k.a. Nonlinearity*f(z)*



Motivation:

Green:
logistic regression.

Red:
ReLU faster
convergence

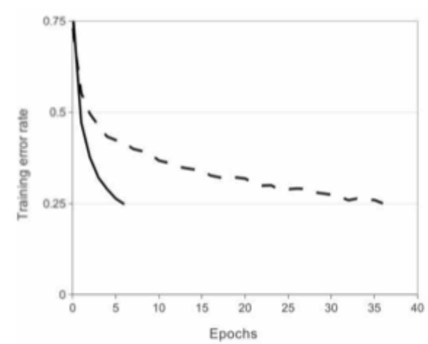


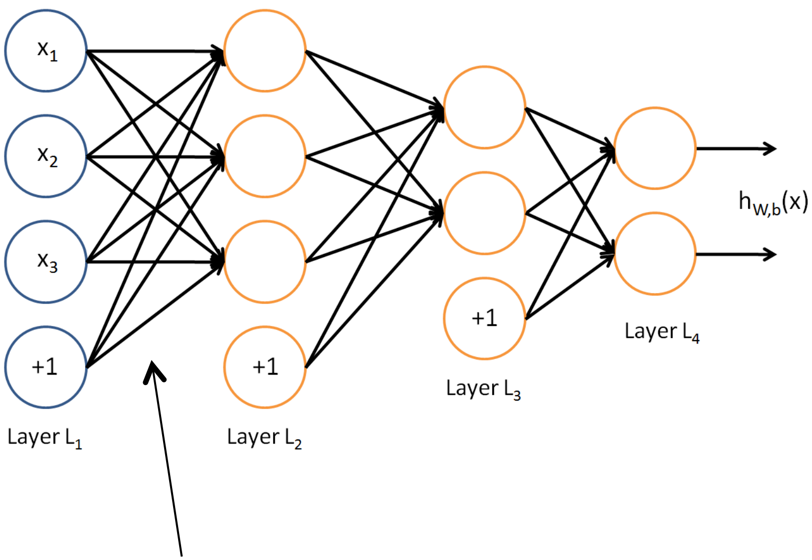
Figure 1: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons

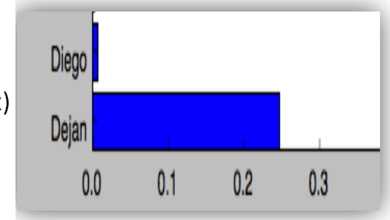
Source:
Alexnet

Krizhevsky et
al 2012

For a more detail explanation see: https://home.zhaw.ch/~dueo/bbs/files/ConvNets_17_Dec_1.pdf

Recap Neural Networks: Stacking things together





Output (Softmax)

$$f(z_i) = \frac{e^{z_i}}{\sum_{i=1}^{N} e^{z_i}}$$

Propability of a class given the input image X

Contains many weights $W^{(l)}_{ij}$

This is just a complex functions of the many weights $\theta = W^{l}_{ij}$ and the input predicting the prob. of a class.

For a more detail explanation see: https://home.zhaw.ch/~dueo/bbs/files/ConvNets_17_Dec_1.pdf
Figure taken from: http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/

Recap Neural Networks: Training the NN

Zurich University
of Applied Sciences

- Use the training data j=1,...,N_{train} to optimize a costfunction *J* sensitive to misclassication
  - Usual a subset n (minibatch) of the training data is taken for optimization in one go.

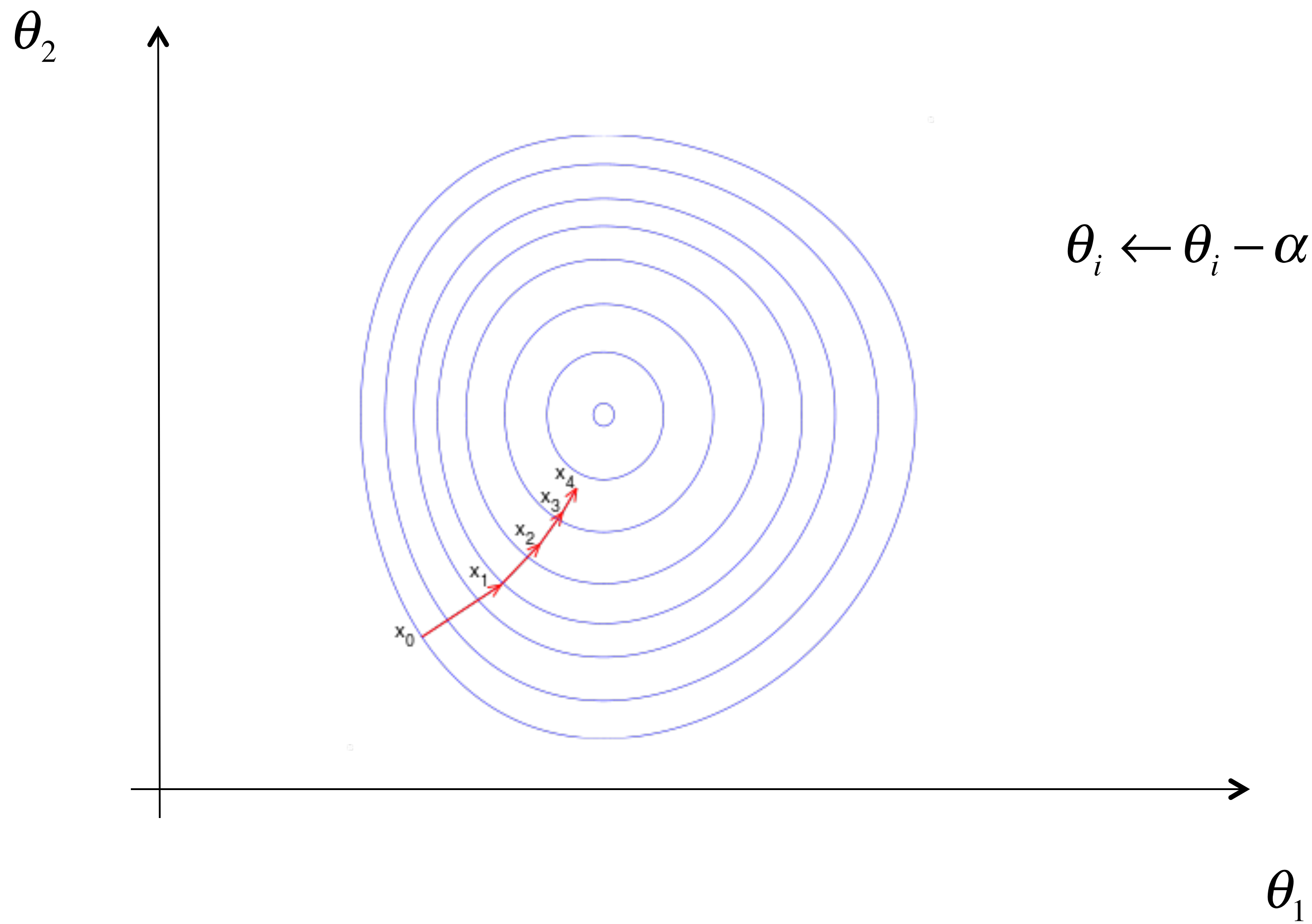
$$-nJ(\theta) = \sum_{i=1}^{n \text{ (Mini Batch Size)}} \text{Cost of Training example } X_i$$

- Motivation of cost function from MaxLikelihhod
- Optimal weights are found in many iterations using gradient descent (α learning rate)

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$$

- Backpropagation a.k.a chainrule is used to calculate the gradient

Illustration of Gradient Descent



$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial J(\theta)}{\partial \theta_i}$$

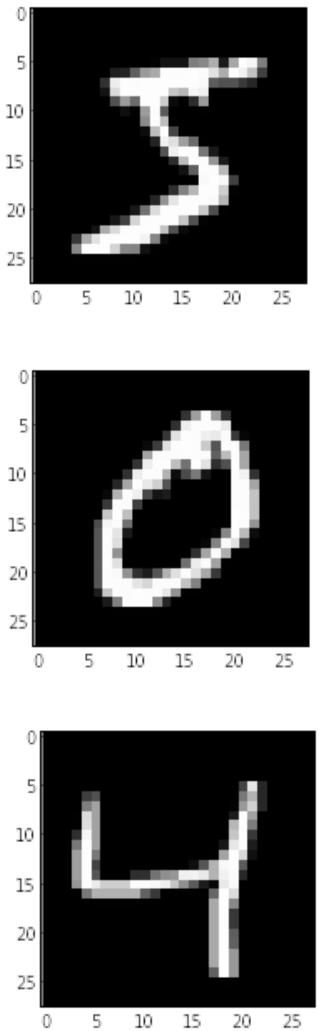
θ_1, θ_2 just two from millions

Zurich University
of Applied Sciences

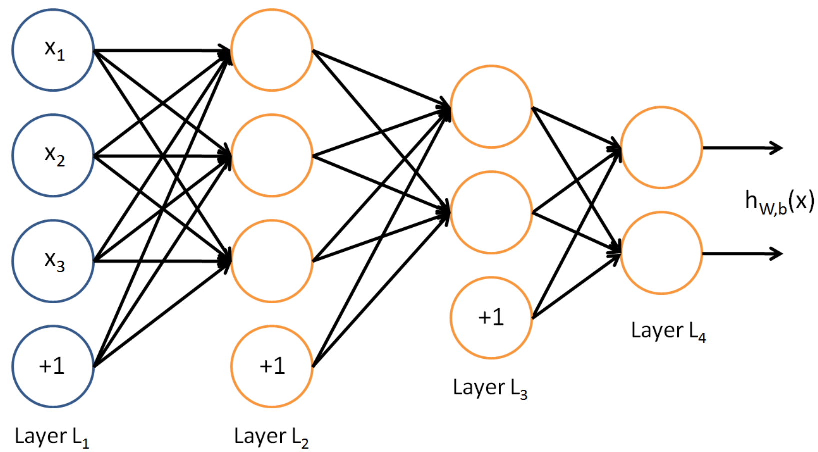
Demo MLP

Definition Data/Network (MLP)

Zurich University
of Applied Sciences



In reality much more nodes



Images 28x28 =784

500

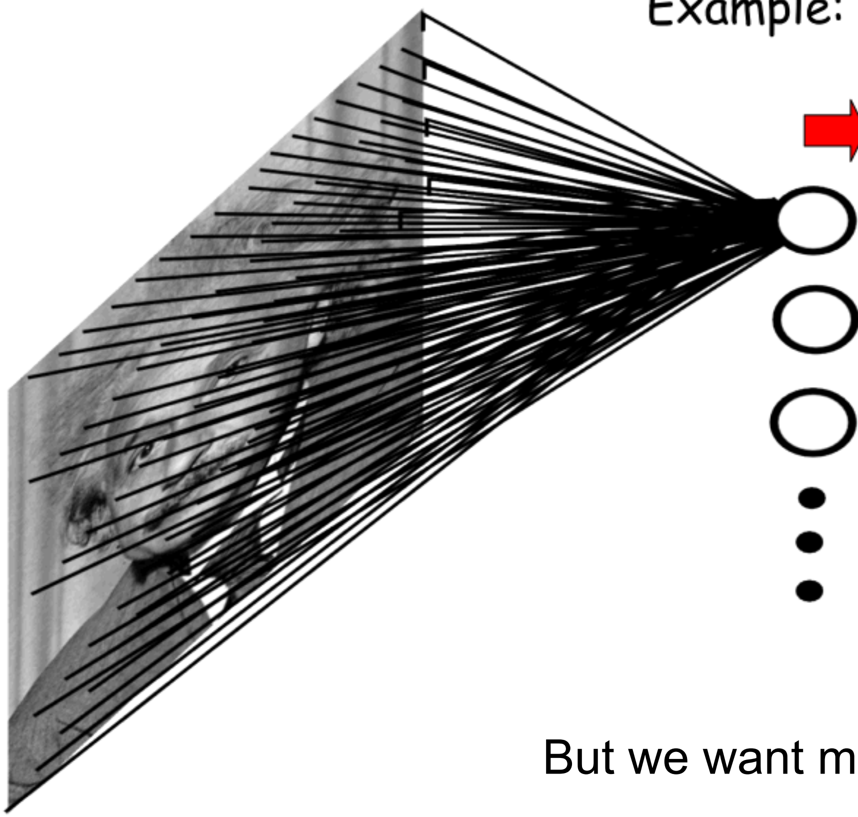
50

10

Now for the Demo…

To many weights

Example: 1000x1000 image
1M hidden units
10^12 parameters!!!



But we want many layers.

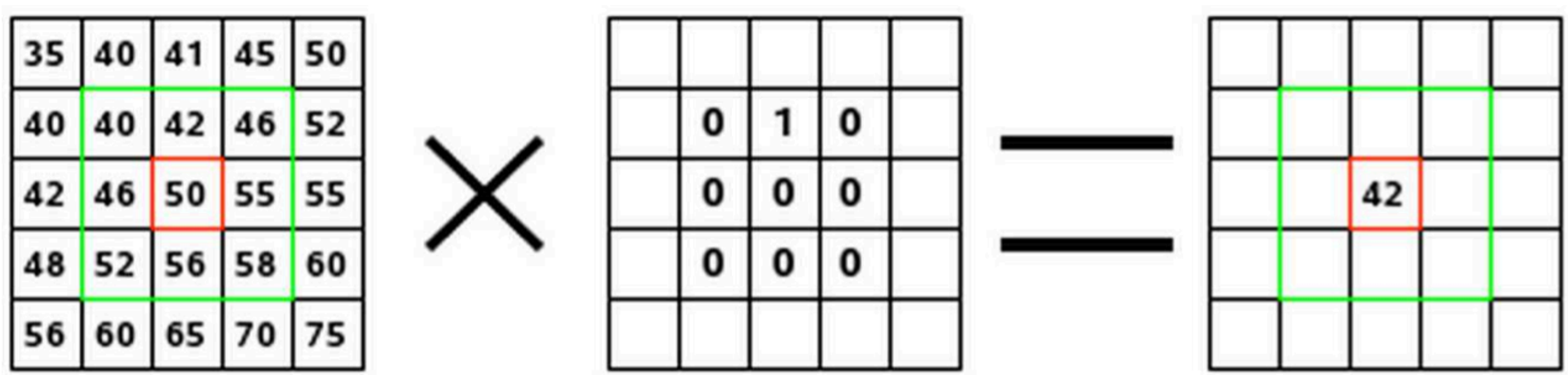
Remedy:

- Weight sharing → Convolution
- Sparse connectivity → Pooling

The convolutional layer
Ingredient I: Convolution

Zurich University
of Applied Sciences

What is convolution?

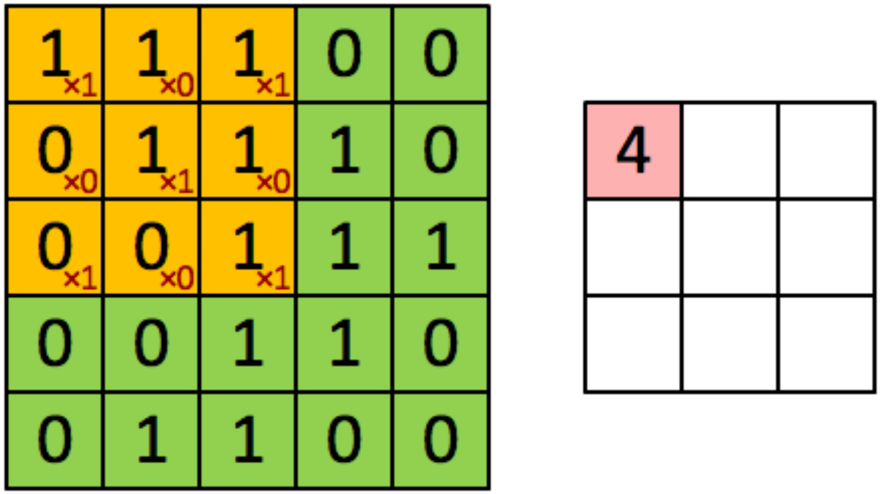


The 9 weights W_{ij} are called Kernel.

The weights are not fixed they are learned!

Gimp documentation: http://docs.gimp.org/en/plug-in-convmatrix.html

The convolutional layer
Ingredient I: Convolution



Image

Convolved Feature

The *same* weights are slid over the image

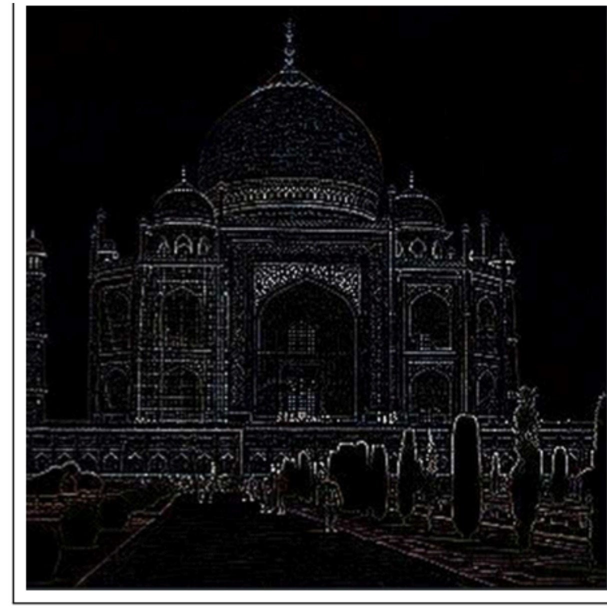
Illustration: http://deeplearning.stanford.edu/tutorial/

Example of a Kernel

Zurich University
of Applied Sciences

Edge enhance
Filter

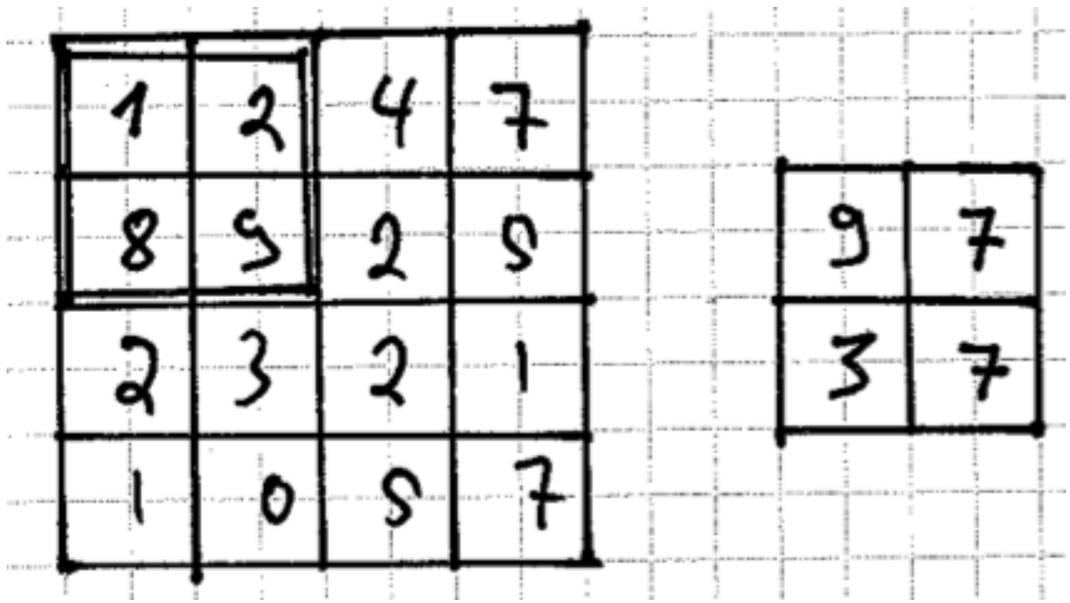
	0	0	0	
	-1	1	0	
	0	0	0	



But again!
The weights are not fixed. They are learned!

Gimp documentation: http://docs.gimp.org/en/plug-in-convmatrix.html

The convolutional layer
Ingredient II: Max-Pooling



Also sliding window versions.

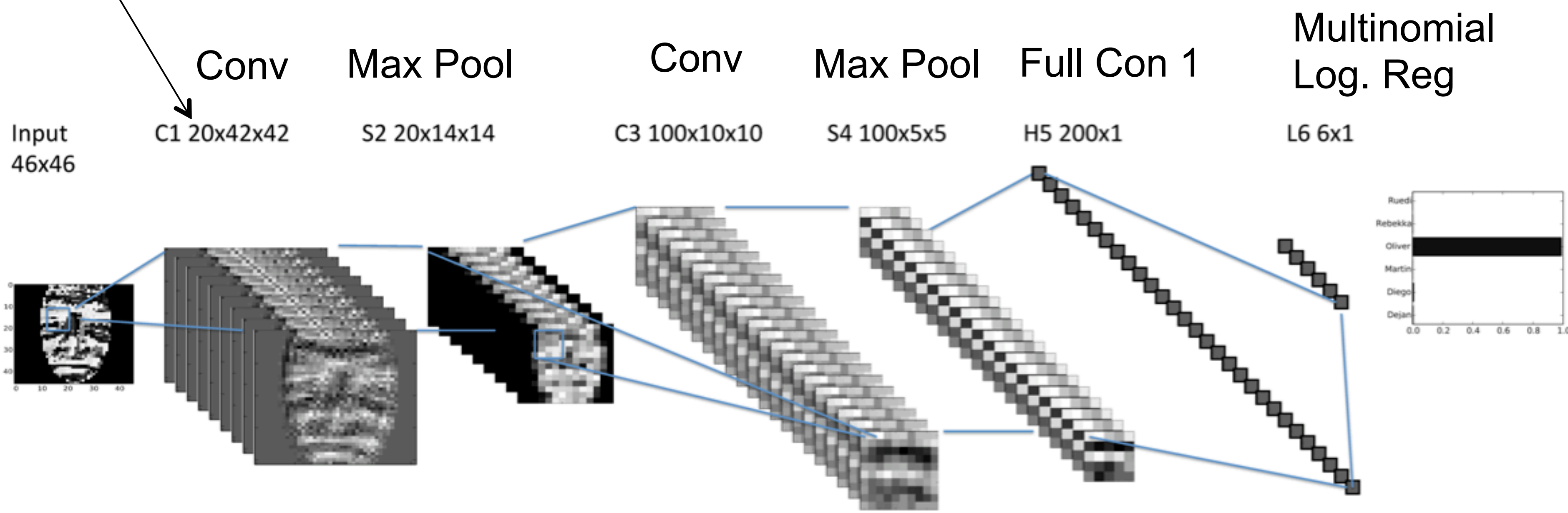
Simply join e.g. 2x2 adjacent pixels in one.

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

A simple version of the CNN (LeNet5 Architecture)

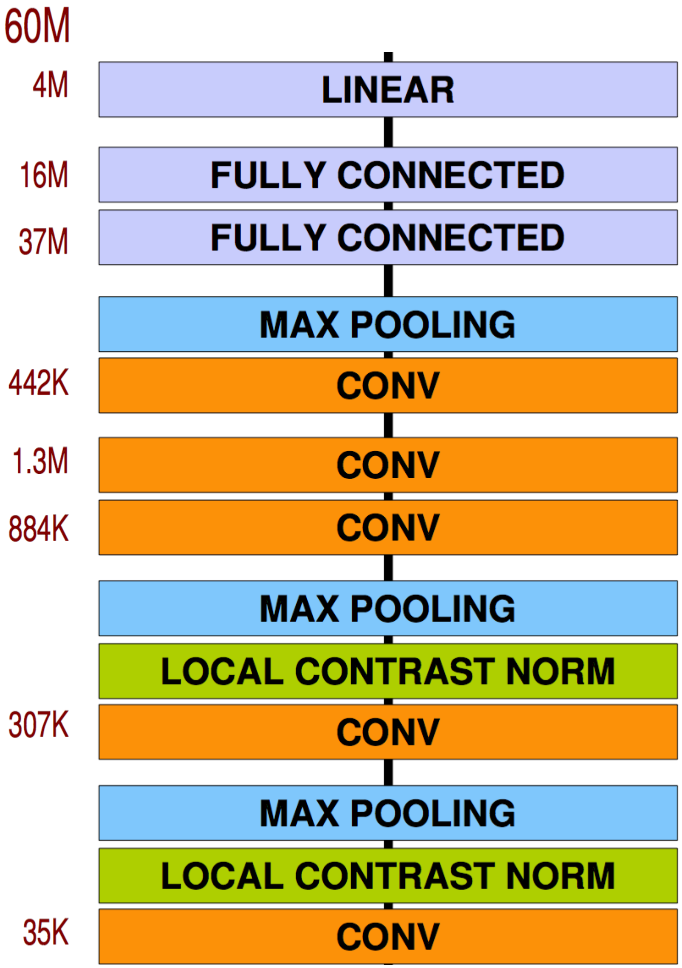
Zurich University of Applied Sciences

20 Kernels a 5x5 weights to go from one to the next

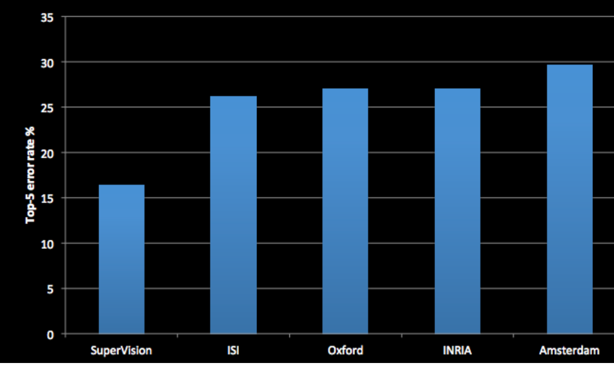


A typical recent architecture (AlexNet, 2012)

Zurich University of Applied Sciences



- Krizhevsky et al. -- 16.4% error (top-5)
- Next best (non-convnet) – 26.2% error



Senimal paper. introduced 26.2% error → 16.5%

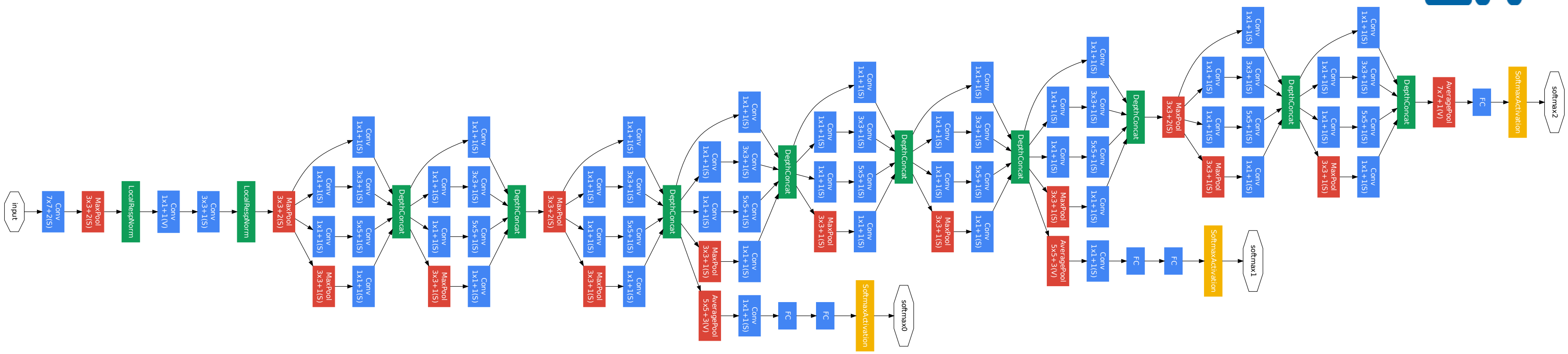
- Dropout (see below)
- ReLU instead of sigmoid
- Parallelisation on many GPUs
- Local Response Normalization (not used widely nowadays)

A bit of a simplification, since Alex Net is build for 2 GPUs and normalization. Caffe code from here

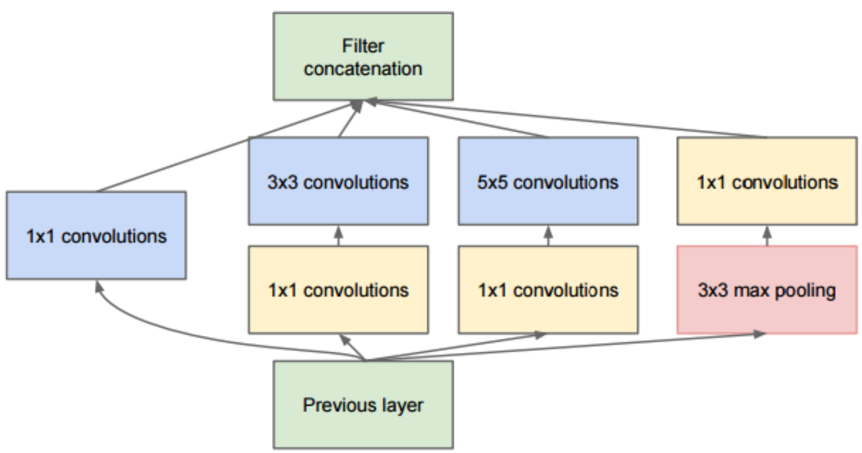
Winning architecture (GoogLeNet, 2014)

Zurich University
of Applied Sciences

zh



The inception module (convolutions and maxpooling)



Few parameters, quite hard to train.
Comments see [here]

[Going deeper with convolution http://arxiv.org/abs/1409.4842]

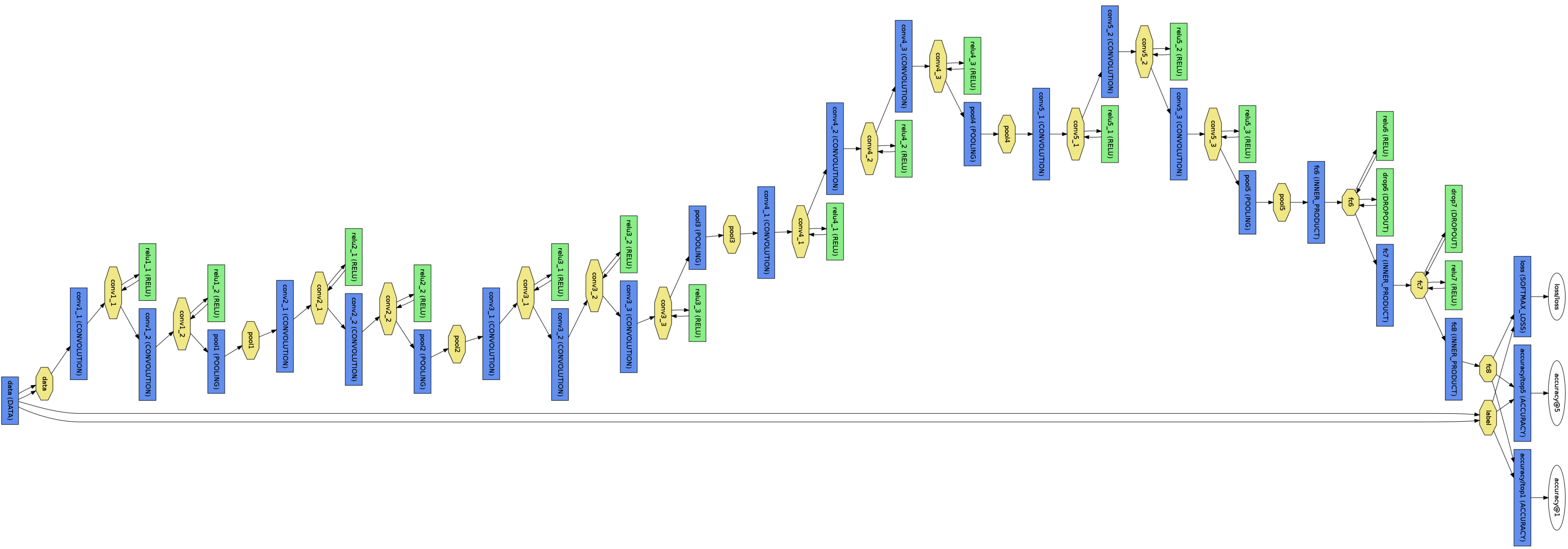
A typical very recent architecture („Oxford Net"(s), 2014)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

- Small pooling
- More than 1 conv before maxpooling.
- No strides (stride 1)
- ReLU after conv and FC
- More traditional, easier to more weights than GoogLeNet
- Caffe Implementation

http://arxiv.org/abs/1409.1556 ImageNet Challenge 2014, 2nd classification

A typical very recent architecture („Oxford Net"(s), 2014)



Definition (16 Layer) taken from:
https://gist.github.com/ksimonyan/211839e770f7b538e2d8#file-readme-md

Zurich University
of Applied Sciences

Demo Lasagne II (Convolutional)

Demo: Nolearn/Lasagne (LeNet Architecture)

Zurich University
of Applied Sciences

```
layers=[
        ('input', layers.InputLayer),
        ('conv1', layers.Conv2DLayer),
        ('pool1', layers.MaxPool2DLayer),
        ('conv2', layers.Conv2DLayer),
        ('pool2', layers.MaxPool2DLayer),
        ('hidden4', layers.DenseLayer),
        ('output', layers.DenseLayer),
        ],
    input_shape=(None, 1, PIXELS, PIXELS),
    conv1_num_filters=32, conv1_filter_size=(3, 3), pool1_ds=(2, 2),
    conv2_num_filters=64, conv2_filter_size=(2, 2), pool2_ds=(2, 2),
    hidden4_num_units=500,
    output_num_units=10, output_nonlinearity=nonlinearities.softmax
```

Have ReLu non-linearity by default

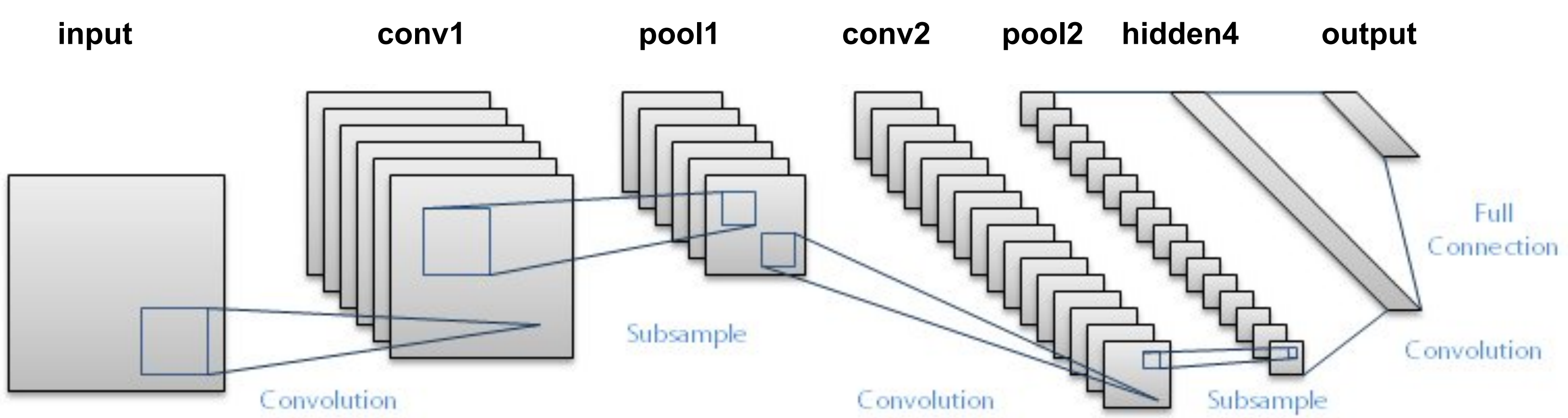
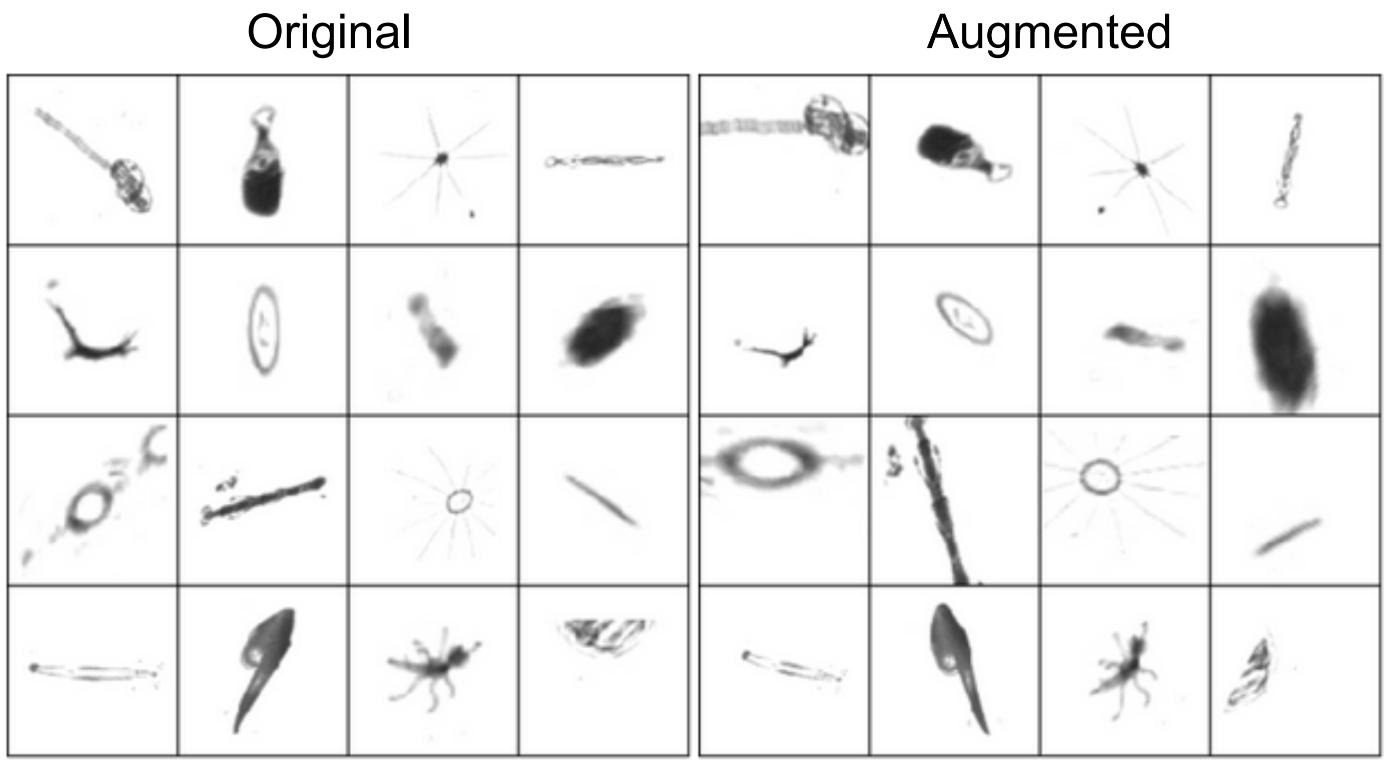


Image taken from: Master Thesis Christopher Mitchell http://www.cemetech.net/projects/ee/scouter/thesis.pdf

(Some) tricks of the trade

Data Augmentation

- Create „new training“ data by „lable preserving transformation“
- Force invariances under translational symetrie (translation, rotation,)



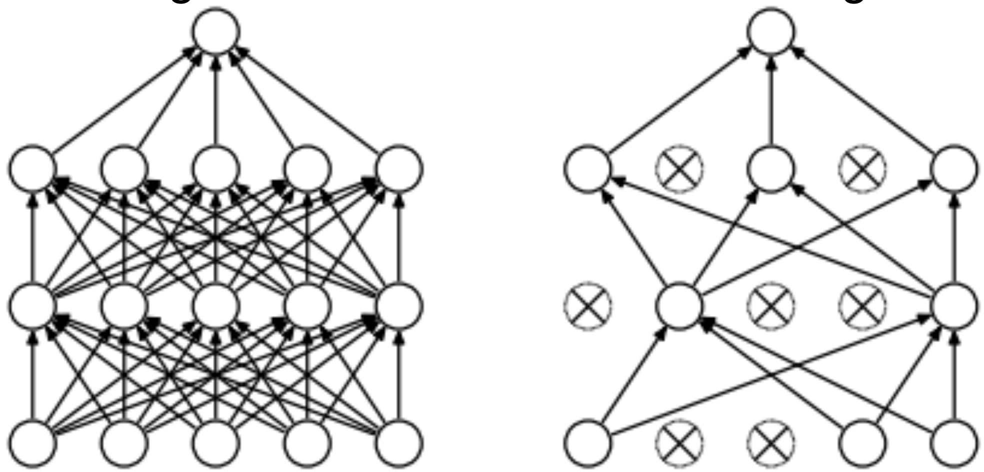
Pre-processed images (left) and augmented versions of the same images (right).

Taken from the winning solution of the plankton challenge. http://benanne.github.io/2015/03/17/plankton.html

Dropout

Original

Training



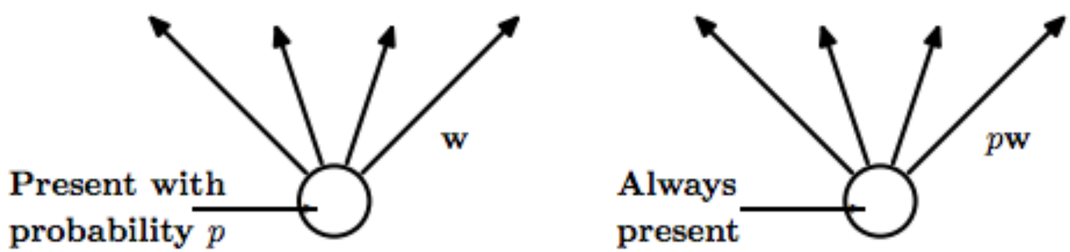
(a) Standard Neural Net

(b) After applying dropout.

At each mini-batch remove random nodes “dropout”

Training time

Test Time



(a) At training time

(b) At test time

Idea: Averaging over many different configuration (exact in case of linear).
Typically 10% performance invcrease

Srivastava et al., Journal of Machine Learning Research 15 (2014) 1929-1958

Zurich University of Applied Sciences

Demo Lasagne III (Convolutional with Training-Data Augmentation)

Attic

Taking a closer look: Convolution

Zurich University of Applied Sciences

zh aw

Documentation:

- The convolutional layer is finished with a nonlinearity:
  - Possible:
    - identity (nothing, linear), rectify (default), tanh, softmax (good for last layer), sigmoid

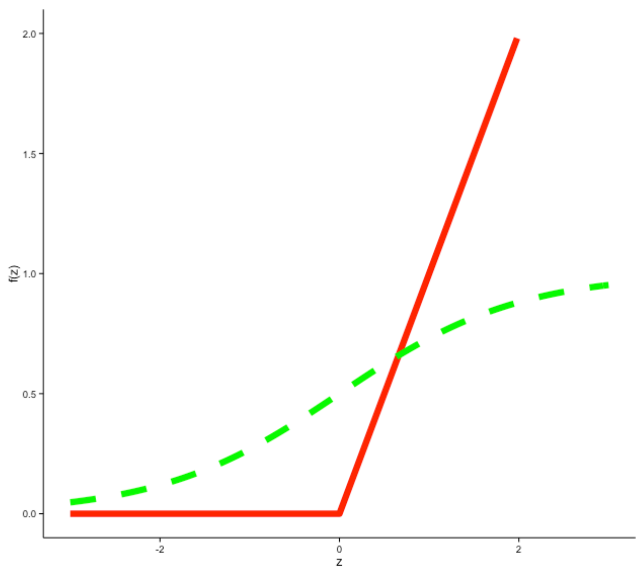
- Different sizes
  - no padding / padding via
    - conv1_border_mode=‘valid‘ #(None, 1, 28, 28) → (None, 32, 26, 26)
    - conv1_border_mode='same‘ #(None, 1, 28, 28) → (None, 32, 28, 28)
    - conv1_border_mode=‘full‘ #(None, 1, 28, 28) → (None, 32, 30, 30) #????

  - stride schrittweite default (1,1)
    - conv1_strides=(2,2) #(None, 1, 28, 28) →(None, 32, 13, 13)

Observation:

Seems to take quite a while for compiling

Cost Functions: ReLU



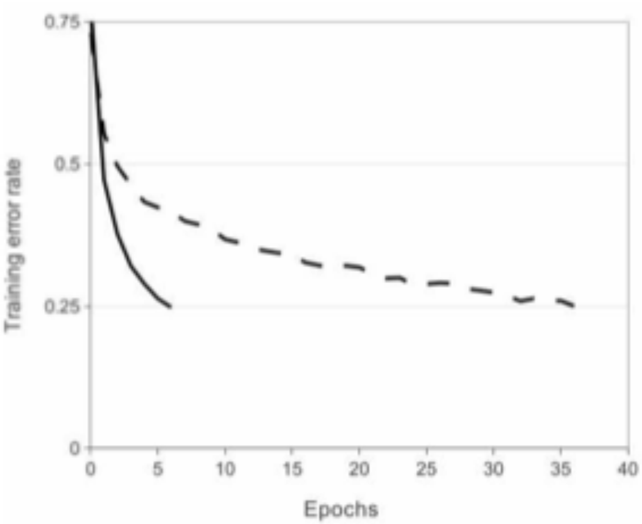


Figure 1: A four-layer convolutional neural network with ReLUs **(solid line)** reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons

Source: Krizhevsky et al 2012

Six Times faster Convergence, than traditional approach.

Intuition: Backpropagation

Generell Notes on Optimization

Zurich University of Applied Sciences

- Gradient Descent (only first order)
- Newton Taylor Expansion 2nd Order using Hessian
-

Taking a closer look: Learning Rate & Momentum

Zurich University
of Applied Sciences

- Nice Description [Caffe Tutorial]()
- Nice Visualization:
    - http://danielnouri.org/notes/2014/12/17/using-convolutional-neural-nets-to-detect-facial-keypoints-tutorial/

- Problem with (stochastic) Gradient Descent are Valleys. You bounce up and down the walls and don‘t descent the slope.
    - Solutions Momentum, Nesterov Momentum (NAG)

$$V_{t+1} = \mu V_t - \alpha \nabla L(W_t)$$
$$W_{t+1} = W_t + V_{t+1}$$

$$V_{t+1} = \mu V_t - \alpha \nabla L(W_t + \mu V_t)$$
$$W_{t+1} = W_t + V_{t+1}$$

- Put mu=0 and we have (S)GD

Taking a closer look: Learning Rate & Momentum

Zurich University
of Applied Sciences

- AdaGrad use all historic information

- Hessian Free Optimizer