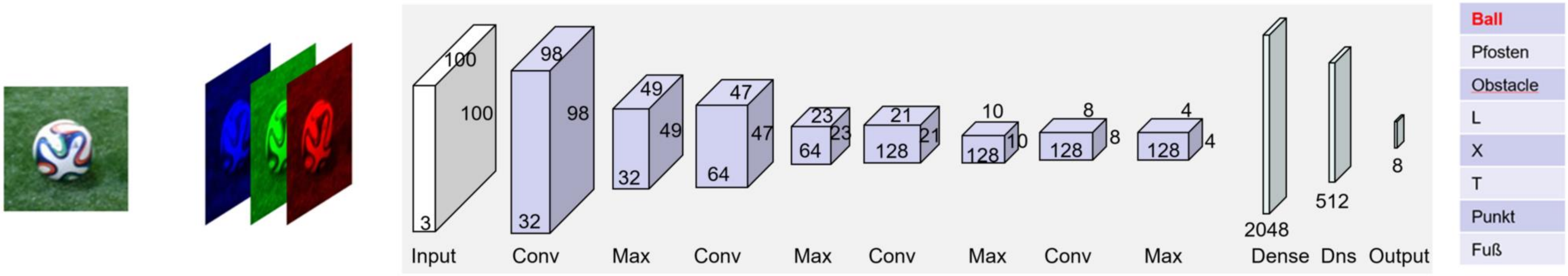
Summer School

Deep Learning

Prof. Dr. Klaus Dorer



Hochschule Offenburg
offenburg.university

Europäischer Fonds für Regionale Entwicklung (EFRE)

Der Oberrhein wächst zusammen: mit jedem Projekt

UpperRhine 4.0

Overview

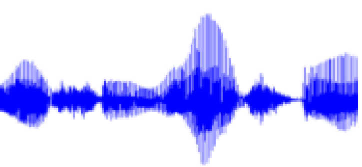
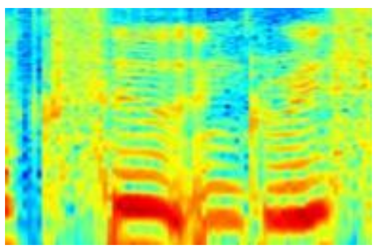
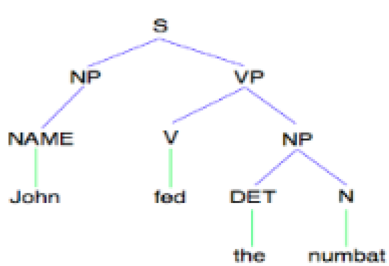
- Neural Networks
  - Introduction
  - Model of a Neuron
  - Perceptron
  - Backpropagation Networks
  - Convolutional Neural Networks

- Goals
  - Know the elements of deep neural networks
  - Have an estimation of their applicability

Why is deep learning so hot?

- More data
  - Youtube: 500 hours video per minute
  - Facebook: 300 milion images per day
- More computing power
  - GPU acceleration
  - GPU/CPU Cluster
- ‚New' Deep Learning approaches
  - Deep Neural Networks
  - Convolutional Neural Networks
  - better activation functions, optimizer, initialization, …
- Freely available frameworks
  - Keras, TensorFlow, Theano, DeepLearning4J, …

Machine Learning before Deep Learning

Input	Feature Extraction (Feature Engineering)	Klassification Detektor	Result
		SVM, (flat) neural network, …	
		HMM, (flat) neural net, …	Language recognition, ...
		Clustering, HMM, …	Machine translation, topic identification, ...

http://on-demand.gputechconf.com/gtc/2015/webinar/deep-learning-course/intro-to-deep-learning.pdf

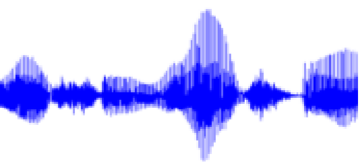
Machine Learning with Deep Learning

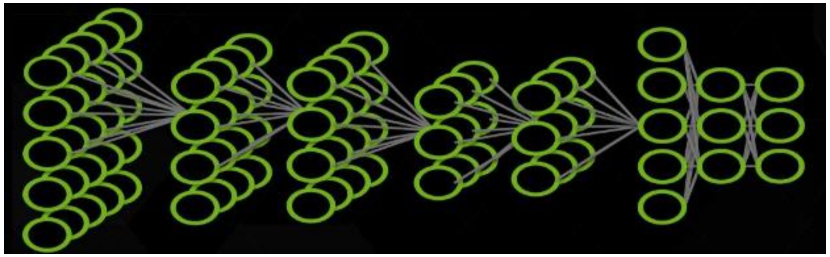
Input

Result

Deep
Network

„stapler"





„stapler"

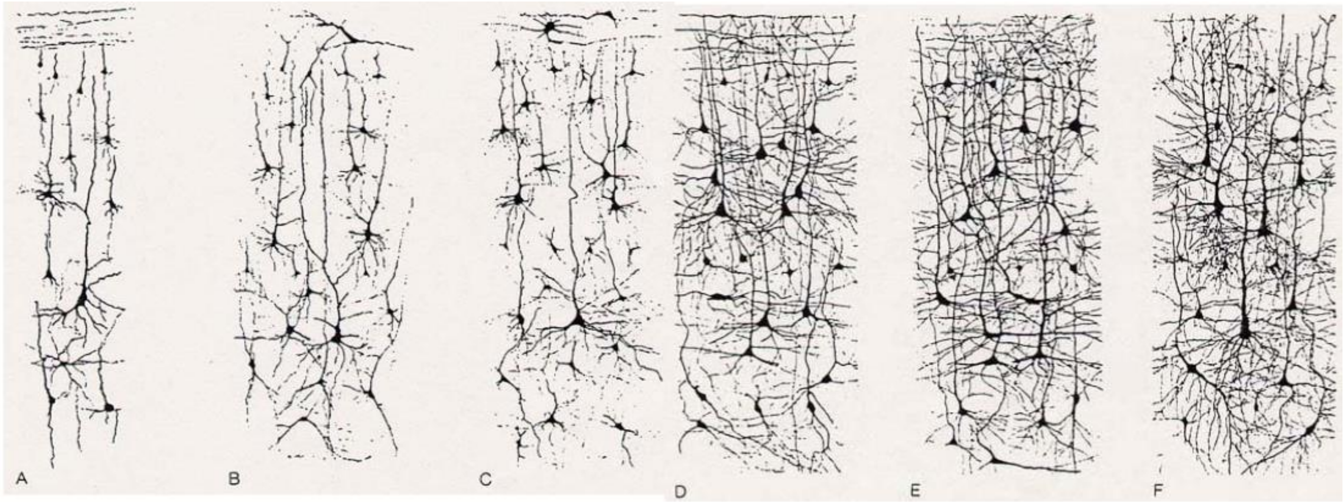
„Manual for a stapler"

....

nach http://on-demand.gputechconf.com/gtc/2015/webinar/deep-learning-course/intro-to-deep-learning.pdf

Human Brain

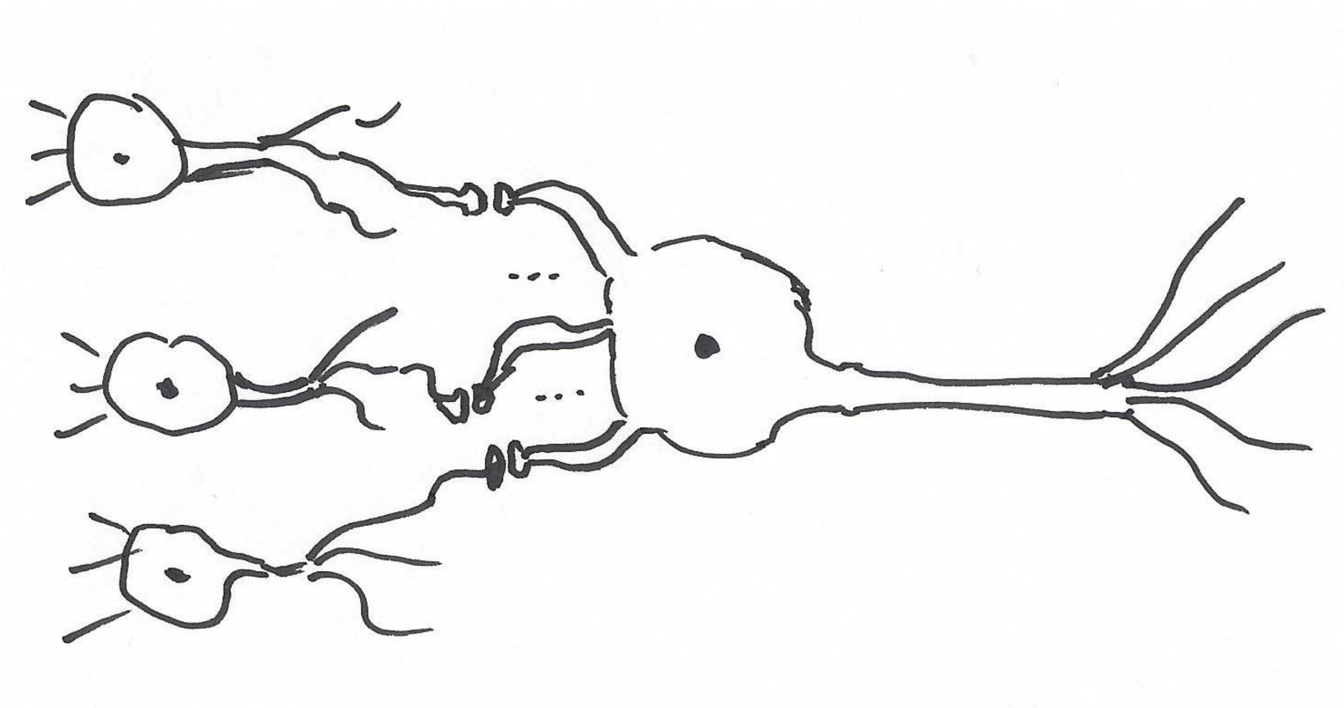
- Learning is performed by
  - Creating links between neurons
  - Reinforcing and weakening links between neurons



Die sechs Bilder vermitteln einen Eindruck von der Entwicklung des Gehirns von der Geburt bis zu einem Alter von zwei Jahren; zum Zeitpunkt der Geburt (A), nach einem Monat (B), nach drei (C), nach sechs (D), nach 15 (E) und nach 24 Monaten (F). Abgebildet ist ein Ausschnitt aus der Großhirnrinde in der Nähe des Broca Sprachareals.

Quelle: http://nwg.glia.mdc-berlin.de/media/pdf/education/Legasthenie.pdf

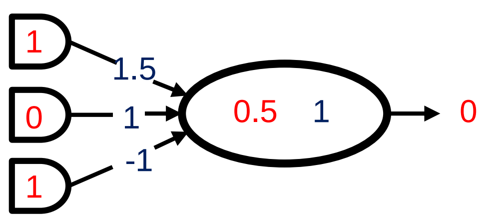
Human Brain



Deep Learning

Model of a Neuron

Prof. Dr. Klaus Dorer



Hochschule Offenburg
offenburg.university

Europäischer Fonds für Regionale Entwicklung (EFRE)

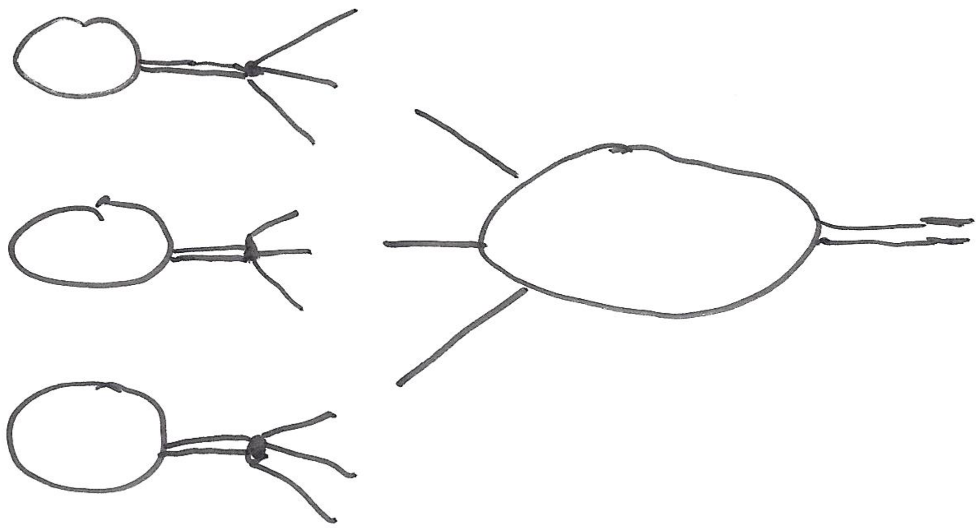
Der Oberrhein wächst zusammen: mit jedem Projekt

UpperRhine 4.0

Overview

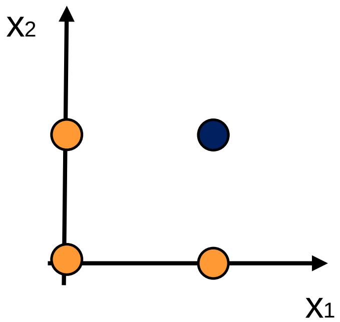
- Neural Networks
  - Introduction
  - Model of a Neuron
  - Perceptron
  - Backpropagation Networks
  - Convolutional Neural Networks

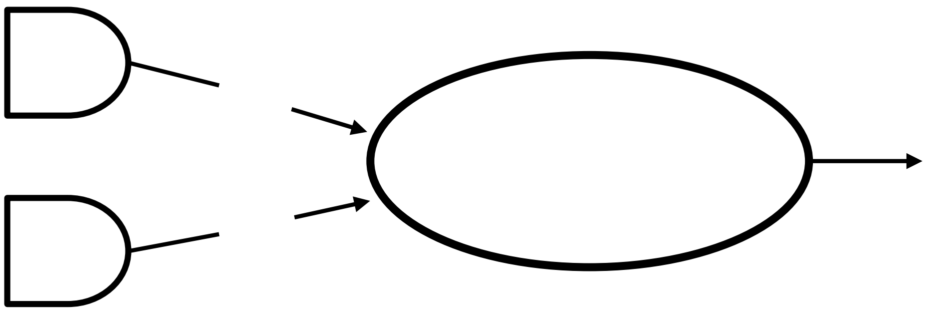
Model of a Neuron

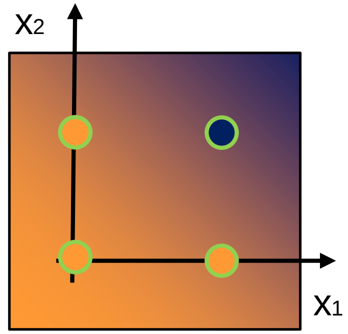


Model of a Neuron
Example: And Function

x_1	x_2	y_1
0	0	0
0	1	0
1	0	0
1	1	1



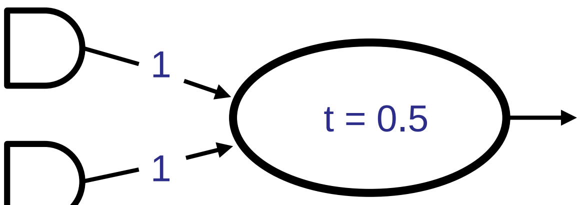




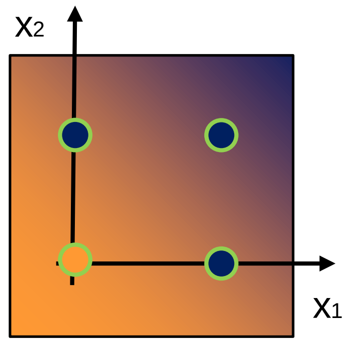
$$o_j = Step_t(\sum_i w_{i,j} x_i) = Step_t(Wx)$$

Model of a Neuron
Or and Not Function

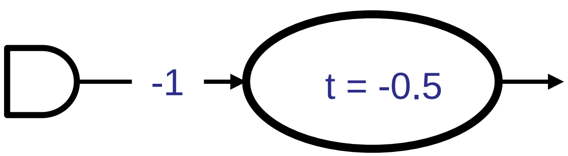
x_1	x_2	y_1
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**



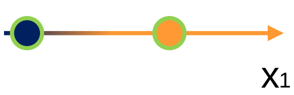
OR



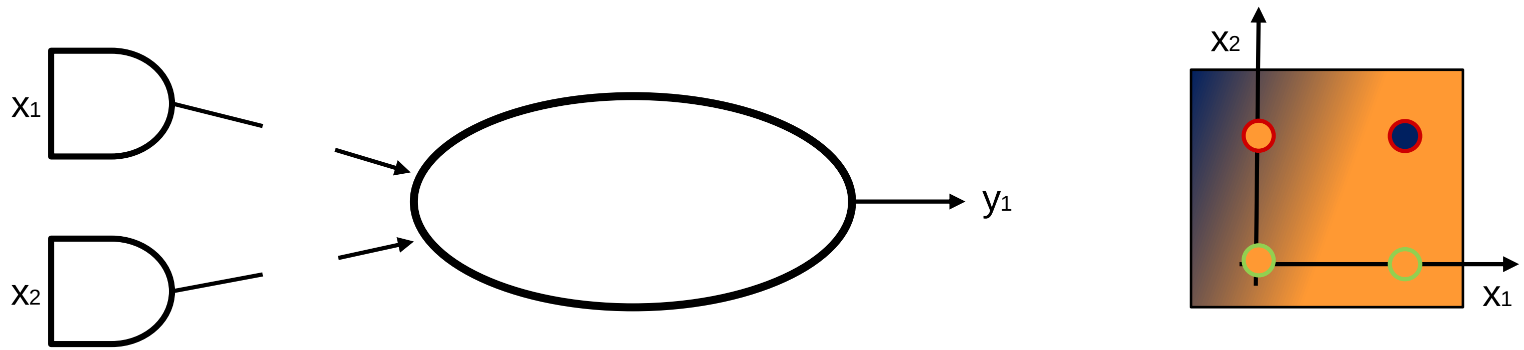
x_1	y_1
0	**1**
1	**0**



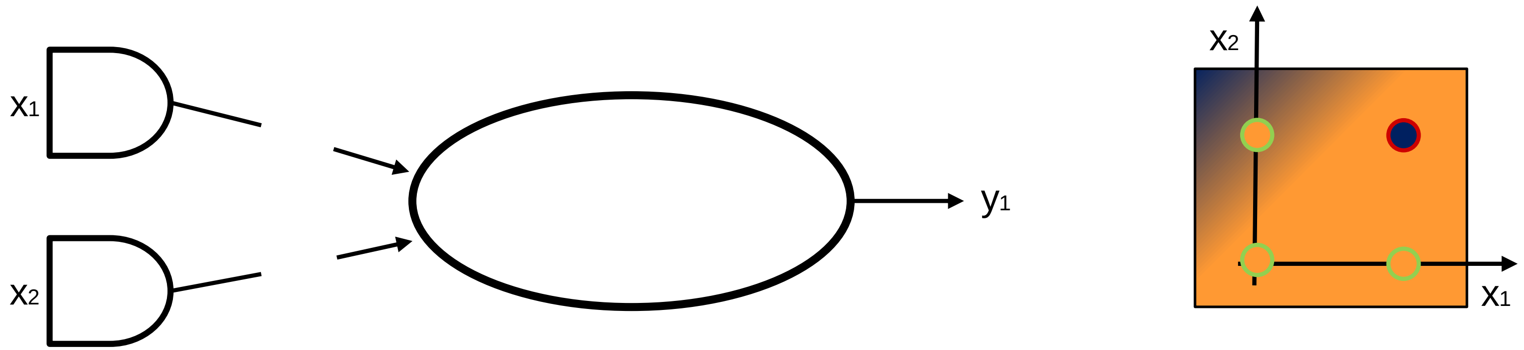
NOT



Model of a Neuron
Find suitable weights

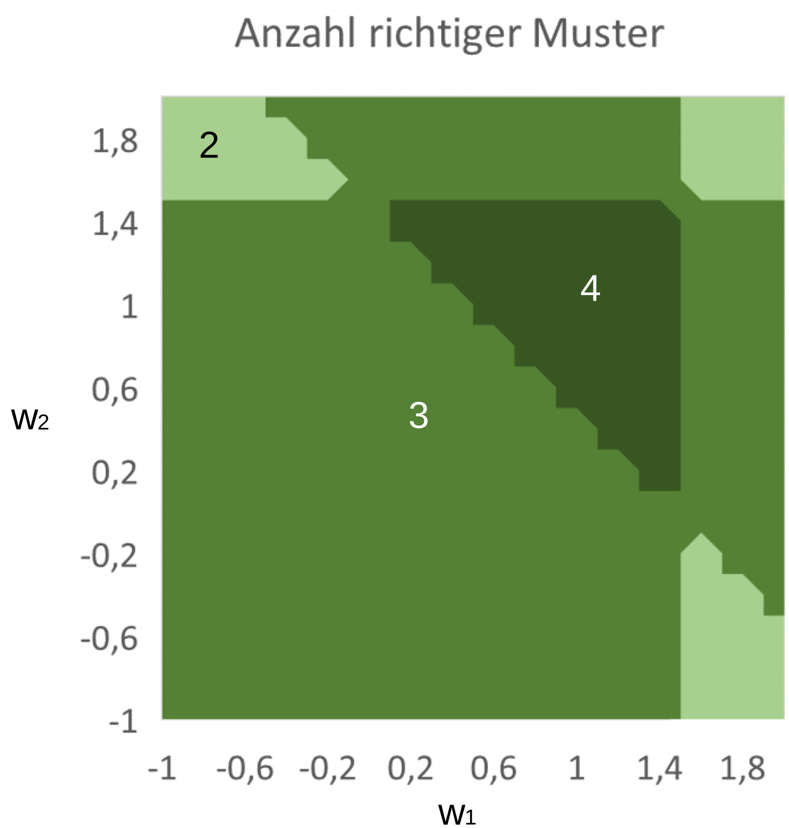


AND

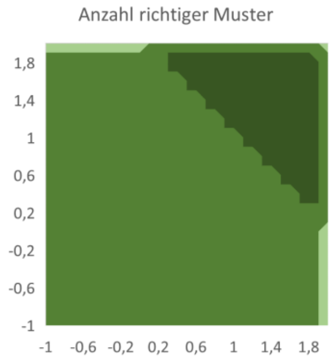


Model of a Neuron
Find suitable weights

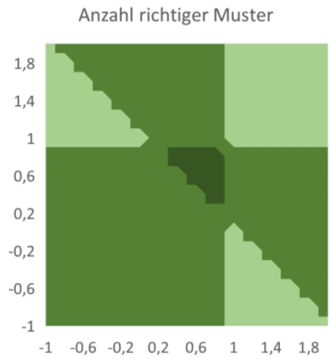
t=1.5



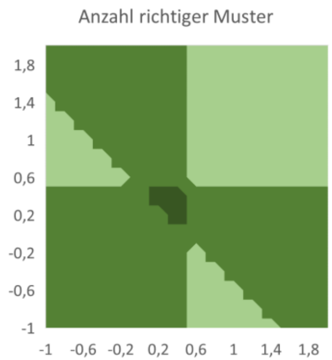
t=2



t=1

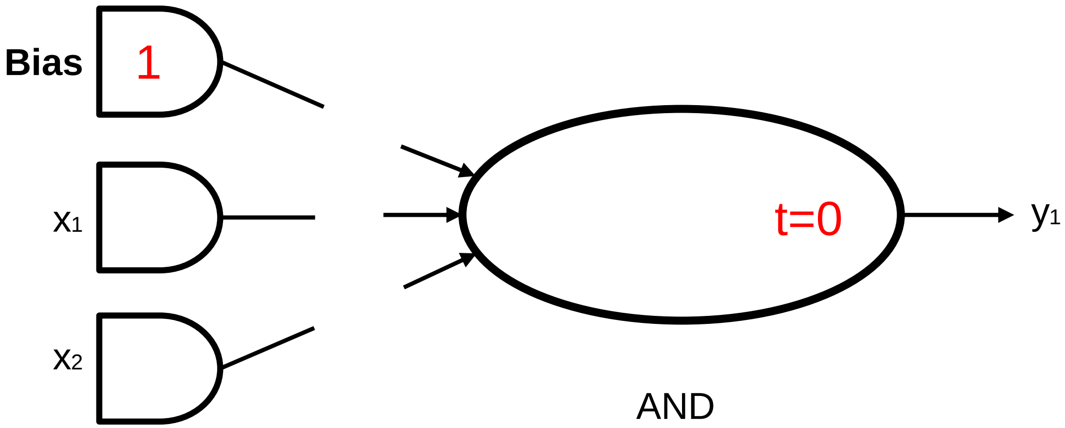


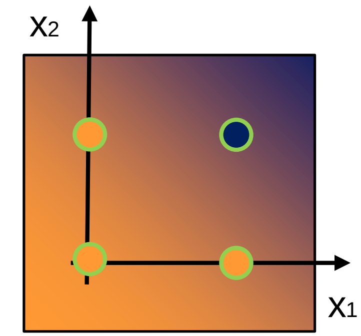
t=0.5



Model of a Neuron
Find suitable weights

- Bias





$$o_j = Step_0\left(\sum_{i+1} w_{i,j} x_i\right)$$

Model of a Neuron
Find suitable weights

- Weights w are -1 and 1
- Loss function
  - Absolute error (L1 Norm)

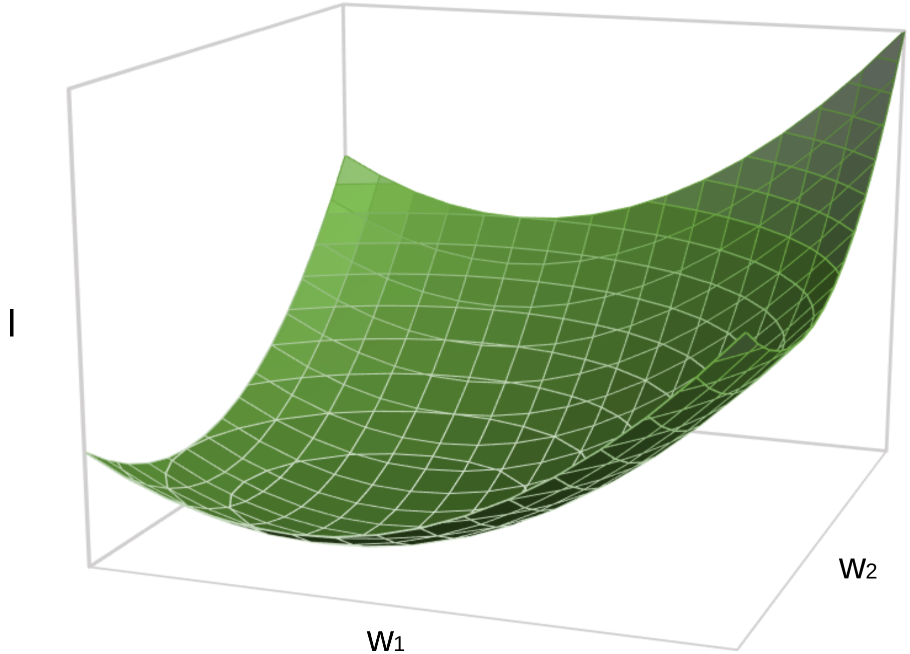
$$l = \sum_i |y_i - o_i| = \| y - o \|_1$$

  - Sum of squared error (L2 Norm)

$$l = \sum_i (y_i - o_i)^2 = \| y - o \|_2^2$$

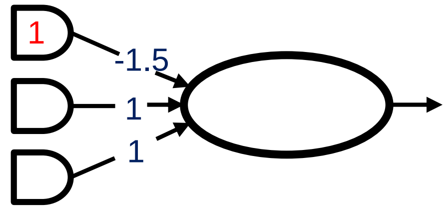
- Gradient descent

x_1	x_2	y_1	o_1	l
0	0	0	0	
0	1	0	0	
1	0	0	1	
1	1	1	0	



Model of a Neuron
Summary

- Input function
- Bias
- Activation function

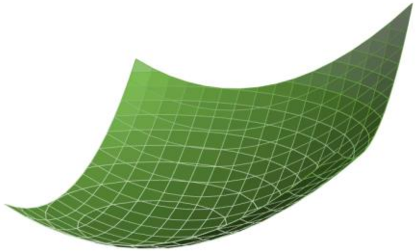


$$o_j = Step_0(\sum_{i+1} w_{i,j} x_i)$$

- Loss function

$$l = \| y - o \|_2^2$$

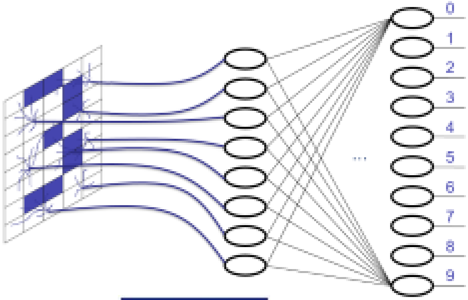
- Gradient descent



Deep Learning

Perceptron

Prof. Dr. Klaus Dorer



Hochschule Offenburg
offenburg.university

Europäischer Fonds für Regionale Entwicklung (EFRE)

interreg Oberrhein Rhin Supérieur

Der Oberrhein wächst zusammen: mit jedem Projekt

Upper Rhine 4.0

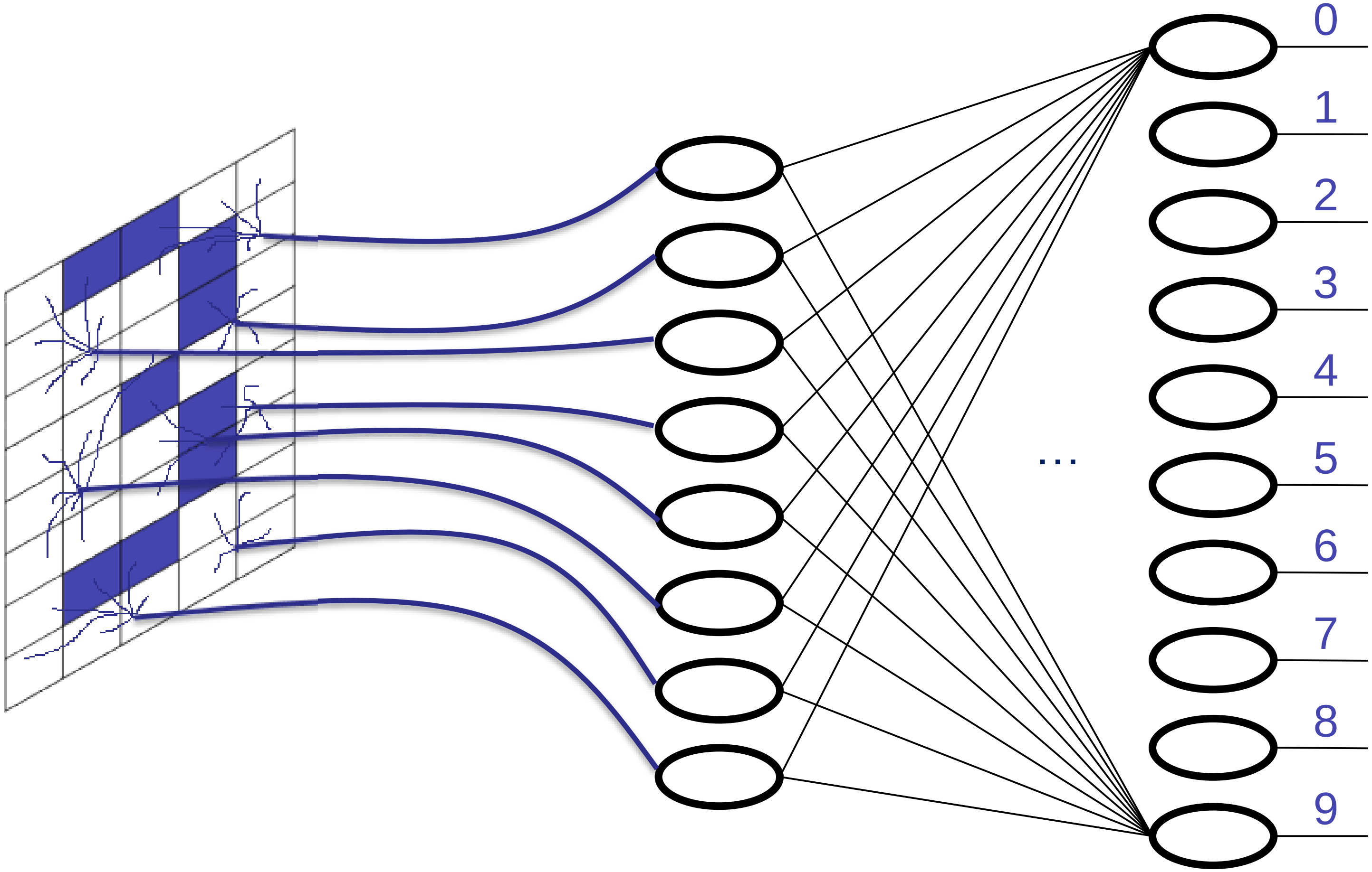
UpperRhine 4.0

Overview

- Neural Networks
  - Introduction
  - Model of a Neuron
  - Perceptron
  - Backpropagation Networks
  - Convolutional Neural Networks

Perceptron

- Single layer feed forward network (1958 Frank Rosenblatt)



Perceptron

- Calculation of output

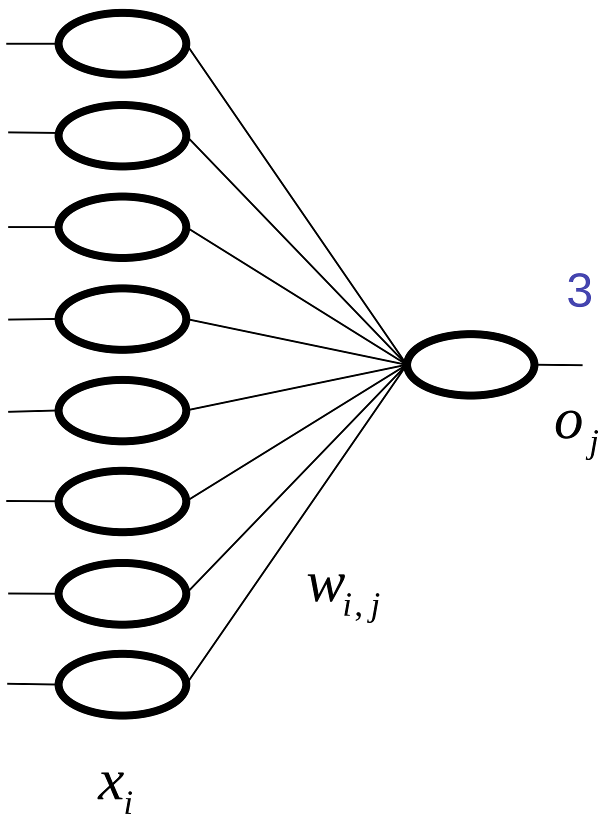
$$o_j = Step_0(\sum_{i+1} w_{i,j} x_i)$$

- Learning rule (delta rule)

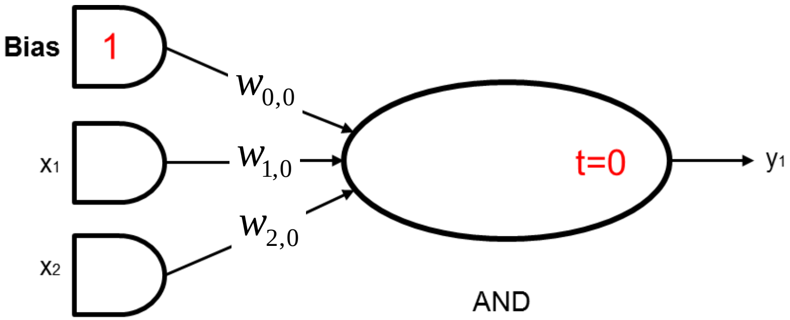
$$w_{i,j} = w_{i,j} + \alpha \cdot x_i \cdot (y_j - o_j)$$

- Algorithm

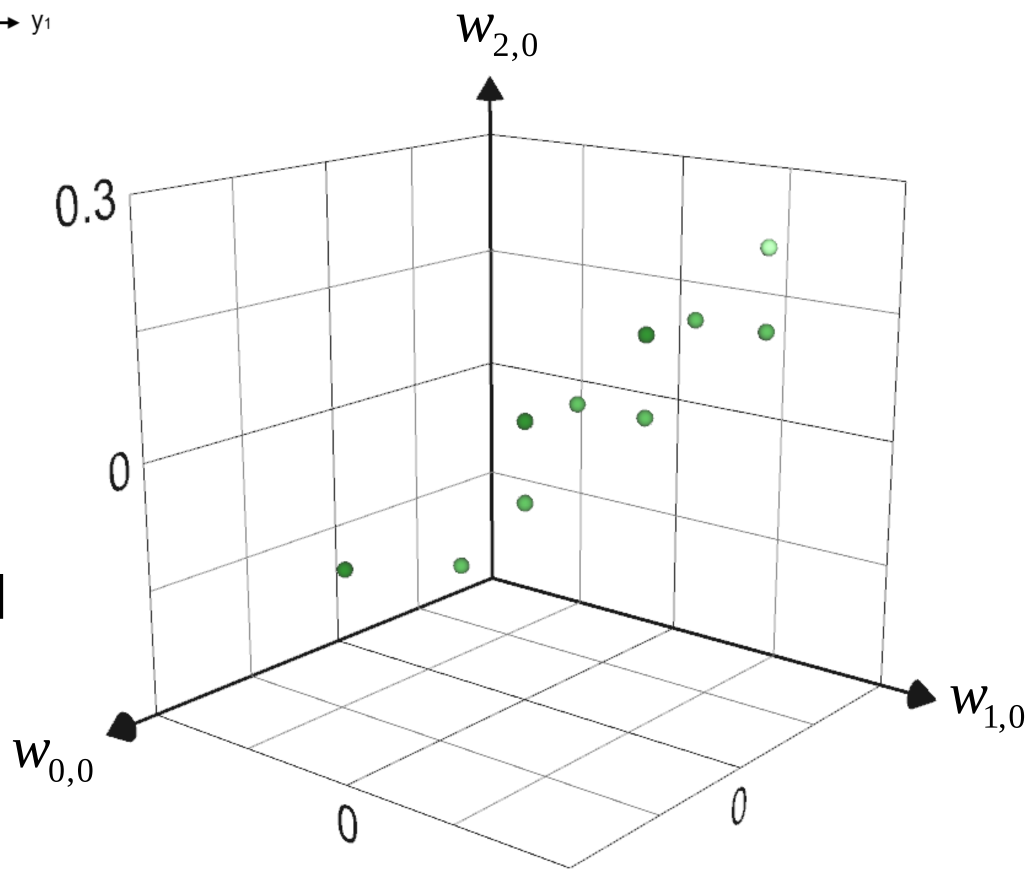
```
Initialize weights randomly
do
   for each e in examples
      calculate output
      adjust weights
while (loss too high and other
         termination criteria not
         reached)
```



Perceptron
And Function



- Random start
  - $w_{x,0}$ [0.028; -0.258; -0.189]
  - 3 patterns wrong
- Train the four patterns
  - $w_{x,0}$ [-0.072; -0.158; -0.189]
  - 2 patterns wrong
- …
  - 0 patterns wrong



Perceptron
Learning Rate and Gradient Descent

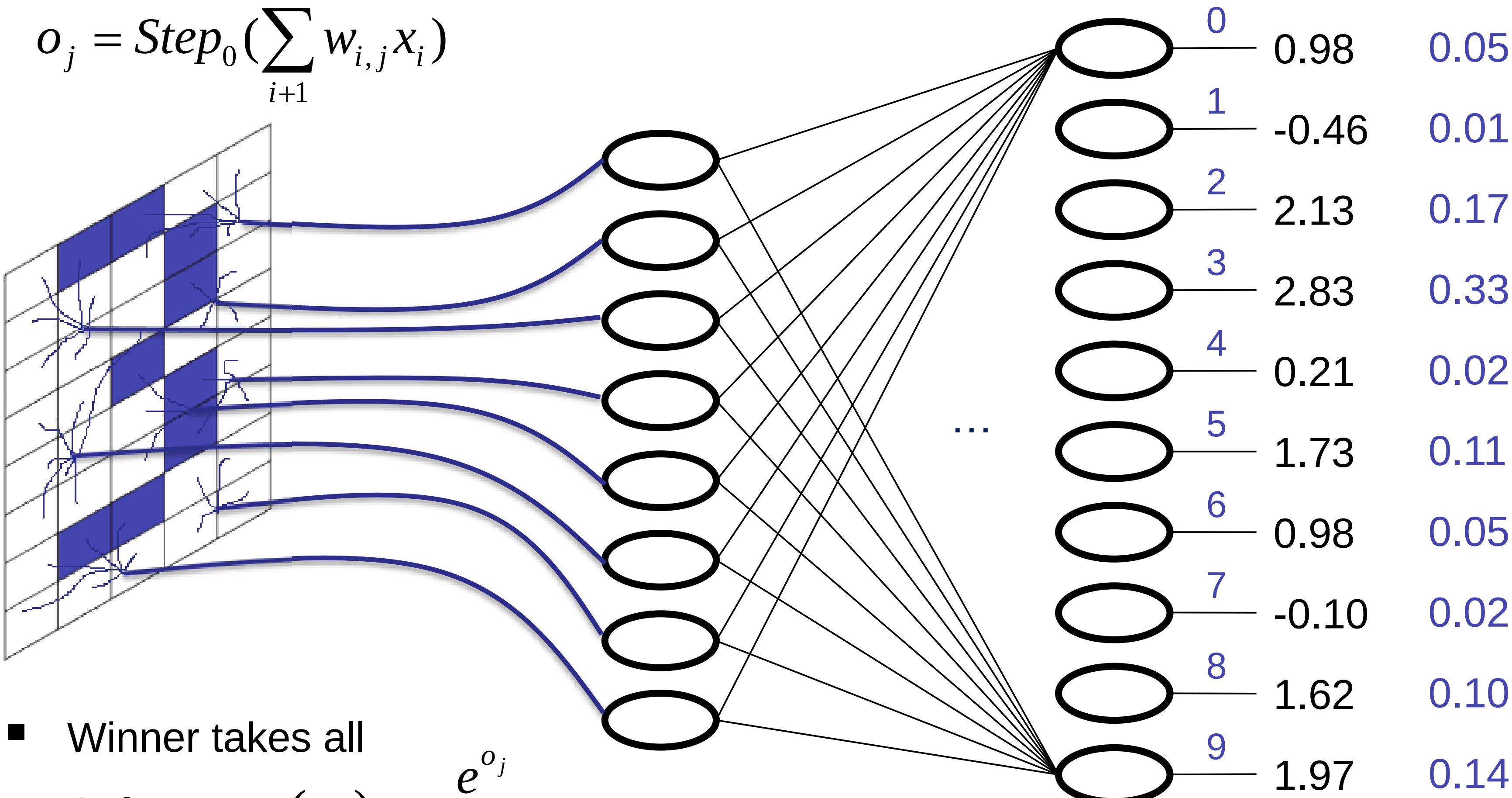
- Small learning rate

- Big learning rate

- Stochastic Gradient Descent
- Batch Learning

Perceptron
Real-valued Inputs and Outputs

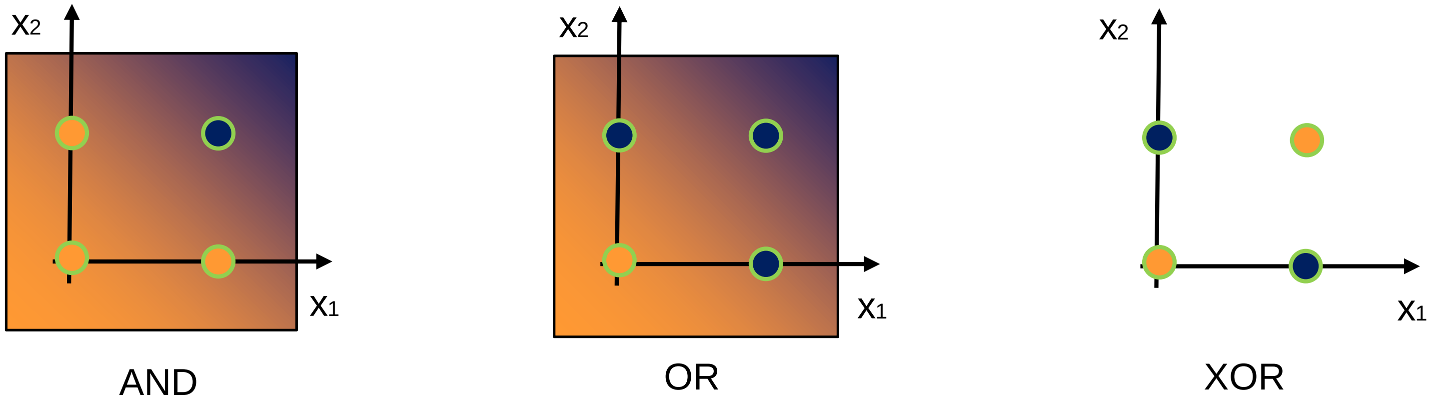
$$o_j = Step_0(\sum_{i+1} w_{i,j} x_i)$$

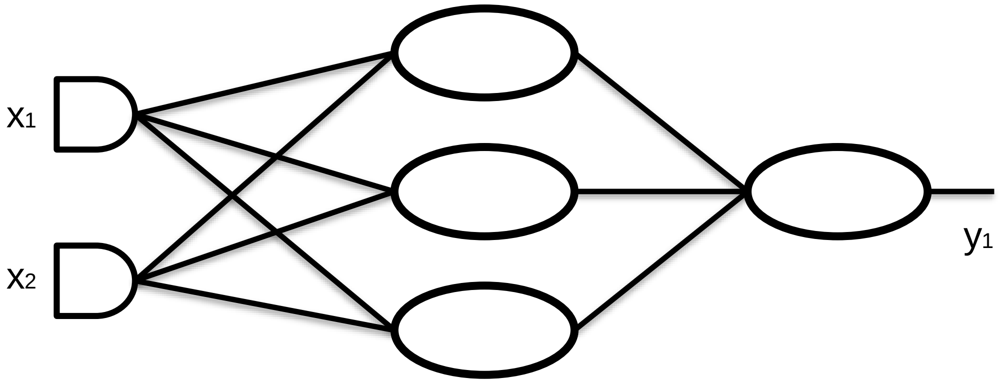


- Winner takes all
- Softmax $\sigma(o_j) = \dfrac{e^{o_j}}{\sum_k e^{o_k}}$

Perceptron
Limitations

- Perceptrons can only represent linearly separable problems





Perceptron
Summary

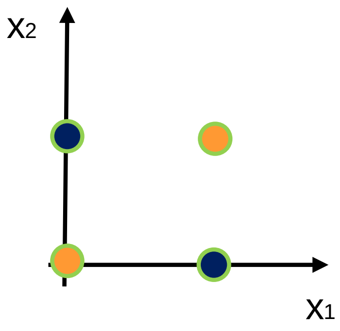
- Learning rule
  - Learning rate
  - Stochastic/Batch Gradient Descent

$$w_{i,j} = w_{i,j} + \alpha \cdot x_i \cdot (y_j - o_j)$$

- Output function
  - Winner takes all
  - Softmax

$$\sigma(o_j) = \frac{e^{o_j}}{\sum_k e^{o_k}}$$

- Problem linear separability

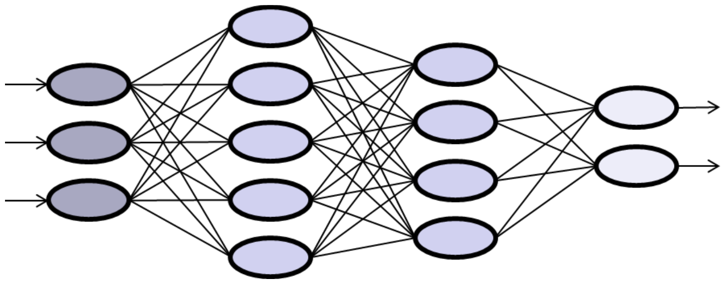


XOR

Deep Learning

Backpropagation Networks

Prof. Dr. Klaus Dorer



Hochschule Offenburg
offenburg.university

Europäischer Fonds für Regionale Entwicklung (EFRE)

Der Oberrhein wächst zusammen: mit jedem Projekt

UpperRhine 4.0

Overview

- Neural Networks
  - Introduction
  - Model of a Neuron
  - Perceptron
  - Backpropagation Networks
  - Convolutional Neural Networks

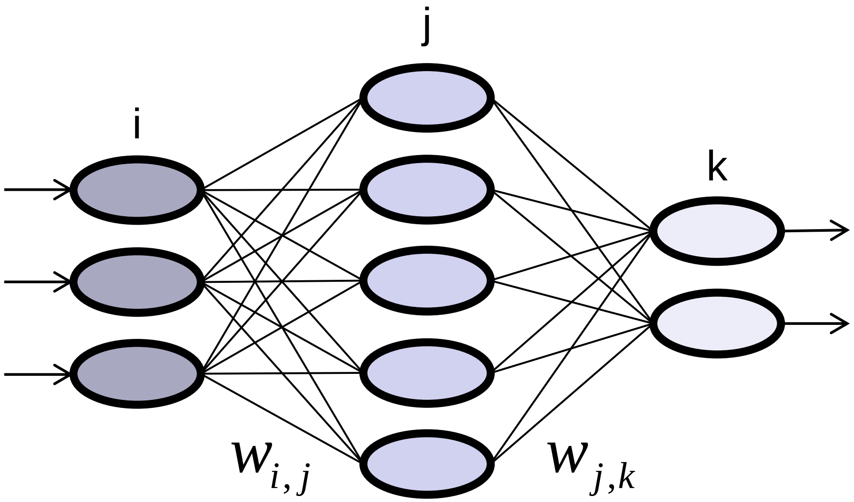
Backpropagation Networks

- To learn XOR oder similar problems we require
  - Multi-layer networks
  - A non-linear activation function
- Problem
  - How to adjust the weights of a hidden layer?
  - How can we propagate the error of the output back to previous layers?
- Solution
  - Backpropagation of Error
  - Bryson & Ho 1969, Rumelhart, Hinton & Wiliams 1986

Backpropagation Networks
Learning Rule

- Calculation of outputs

$$o_j = \sigma(\sum_{i+1} w_{i,j} x_i)$$



- Learning rule
  - Weights to output

$$w_{j,k} = w_{j,k} + \alpha \cdot o_j \cdot \Delta_k \text{ mit } \Delta_k = \sigma'(in_k) \cdot (y_k - o_k)$$

  - Weights to a hidden layer

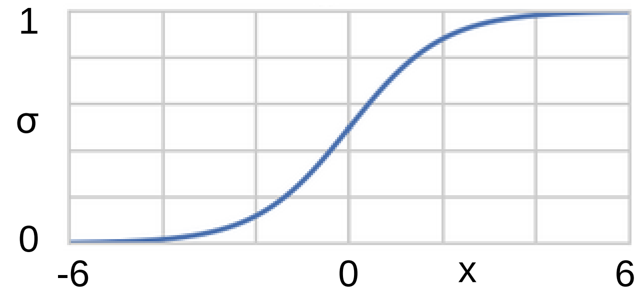
$$w_{i,j} = w_{i,j} + \alpha \cdot o_i \cdot \Delta_j \text{ mit } \Delta_j = \sigma'(in_j) \cdot \sum_k w_{j,k} \Delta_k$$

Backpropagation Networks
Algorithm

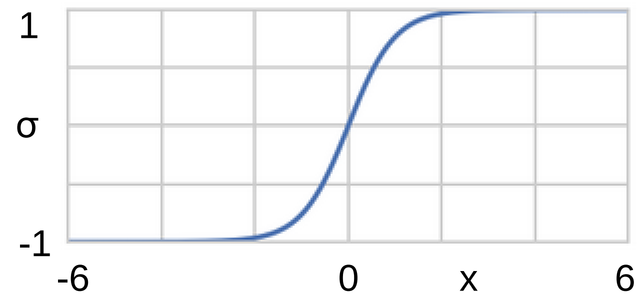

```
initialize weights randomly
do
  for each e in examples
    calculate output (recall)
    calculate Δ values for output units
    repeat for each layer(backward from the outputs)
      propagate Δ values back to previous layer
      adjust weights
while (loss too big and other stop criteria not reached)
```

Hochschule Offenburg
offenburg.university

- Sigmoid
  - Function $$\sigma(x) = \frac{1}{1+e^{-x}}$$
  - Derivative $$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

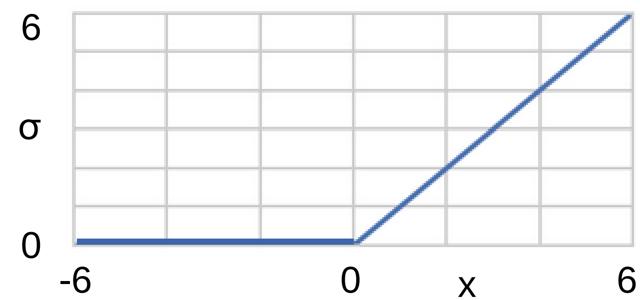


- Tanh $$\sigma(x) = \tanh(x)$$



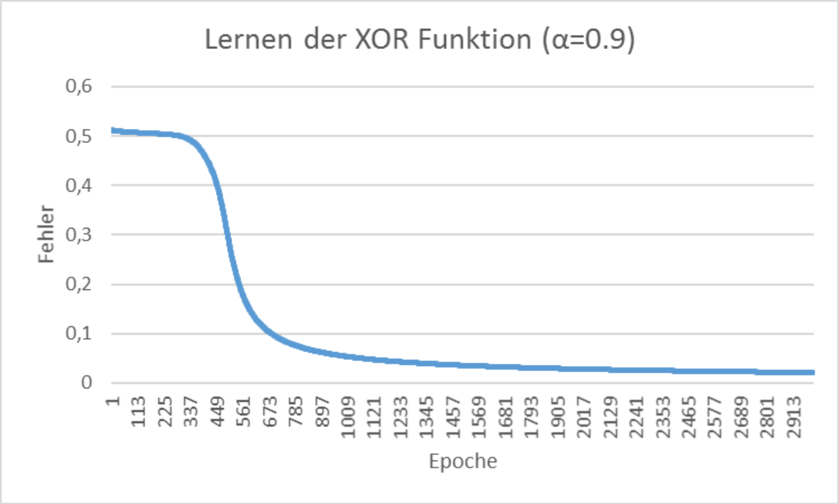
- Relu $$\sigma(x) = \max(0, x)$$

  (Rectified linear unit)

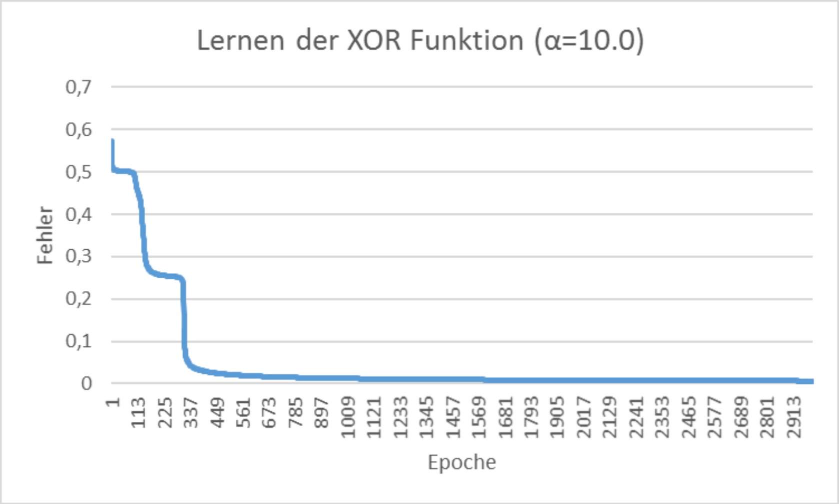


Backpropagation Networks
Example: XOR Function

- A 3 – 3 – 1 network can learn XOR
  - Learning rate 0.9

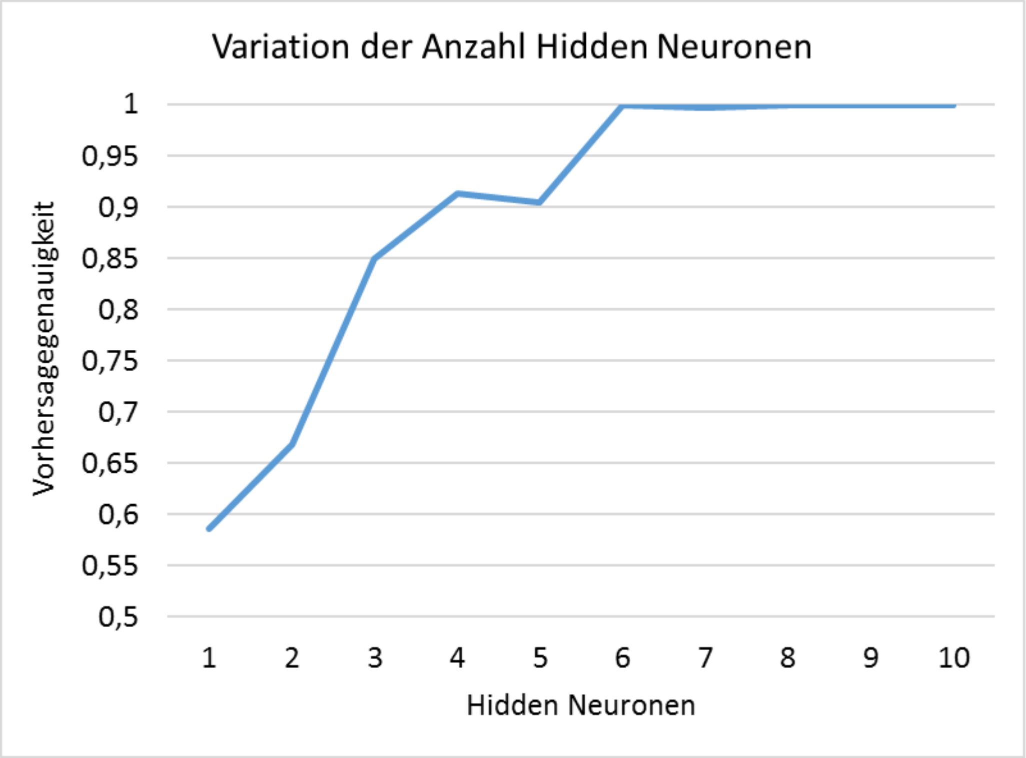


  - Learning rate 10



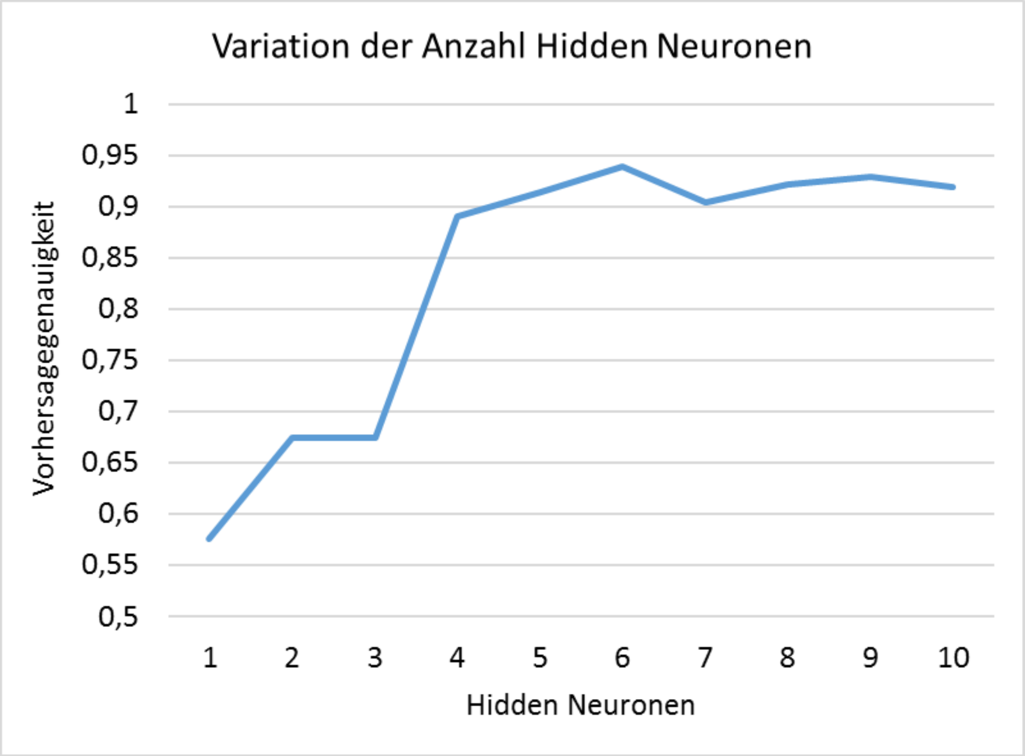
Backpropagation Networks
Example: classification

- What is a good size of the network?
  - To represent the training patterns?
  - Test of accuracy on training data
- Data
  - 864 patterns training
  - 864 patterns test
  - 6 attributes (4,4,4,3,3,3)
  - Class: yes/no
- Network
  - 21 input neurons
  - x hidden neurons
  - 2 output neurons



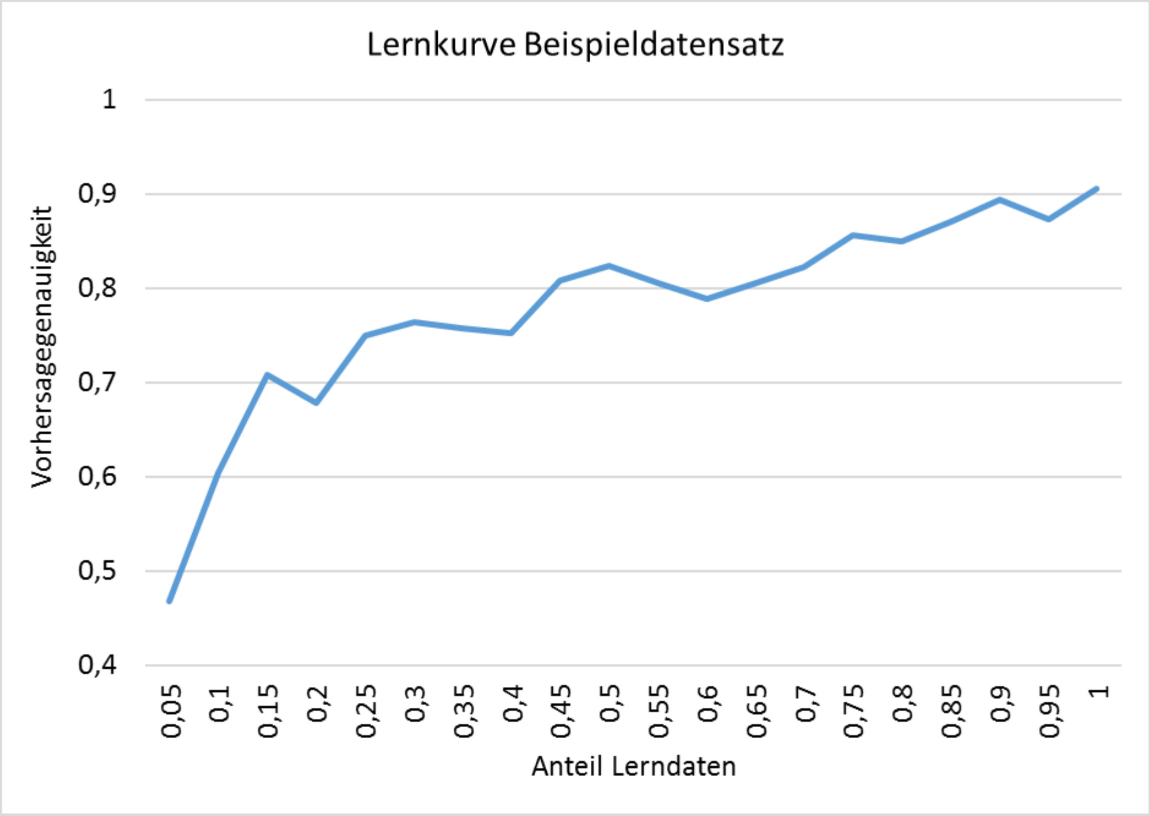
Backpropagation Networks
Generalization

- What is a good size of the network?
  - To properly predict unknown patterns?
  - Test of accuracy on test data



Backpropagation Networks
Learning Curve

- How many patterns are required for the network to generalize?
  - The more the better



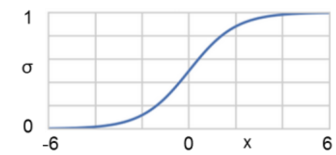
Backpropagation Networks
Summary

- Learning rule

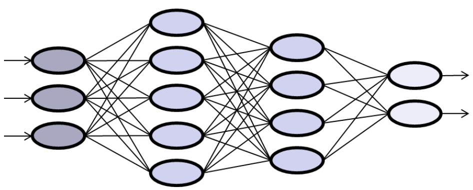
$$w_{i,j} = w_{i,j} + \alpha \cdot o_i \cdot \Delta_j \text{ mit } \Delta_j = \sigma'(in_j) \cdot \sum_k w_{j,k} \Delta_k$$

- Activation function

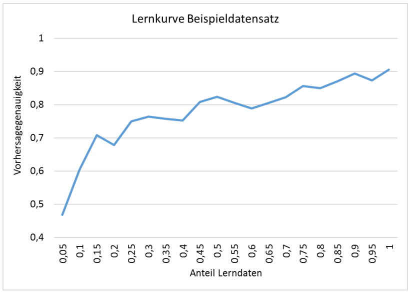
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



- Network size



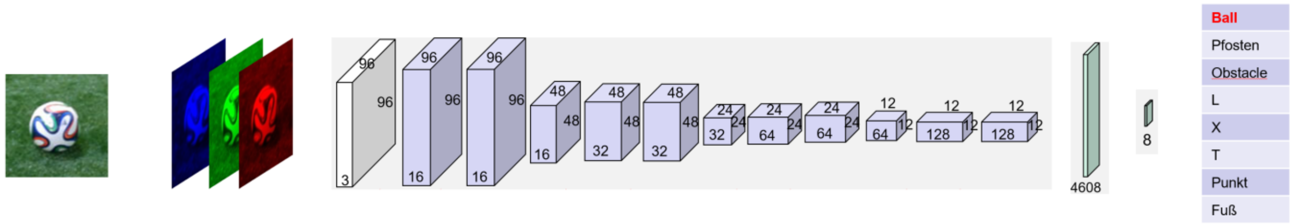
- Learning curve



Deep Learning

Convolutional Neural Networks

Prof. Dr. Klaus Dorer



Hochschule Offenburg
offenburg.university

Europäischer Fonds für Regionale Entwicklung (EFRE)

Der Oberrhein wächst zusammen: mit jedem Projekt

UpperRhine 4.0

Overview

- Neural Networks
  - Introduction
  - Model of a Neuron
  - Perceptron
  - Backpropagation Networks
  - Convolutional Neural Networks
    - Structure
    - Application
    - Deep Learning Frameworks

Convolutional Neural Networks

- Sweaty is a soccer robot
- It needs to see things on its camera images
- No matter where in the picture



- No matter how bright the light



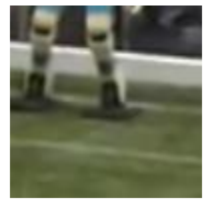
- No matter what color, pattern



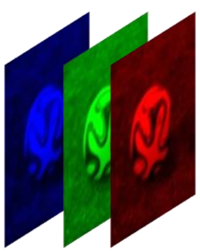
Convolutional Neural Networks

- Input is a 3D matrix
  - Here: 100x100 Pixel, 3 color chanels (Red, Green, Blue)

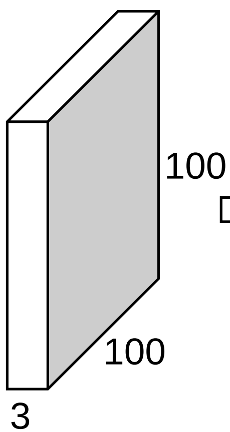




- Output is a feature-vector
  - Ball X-Line Robot L-Line Obstacle



Input

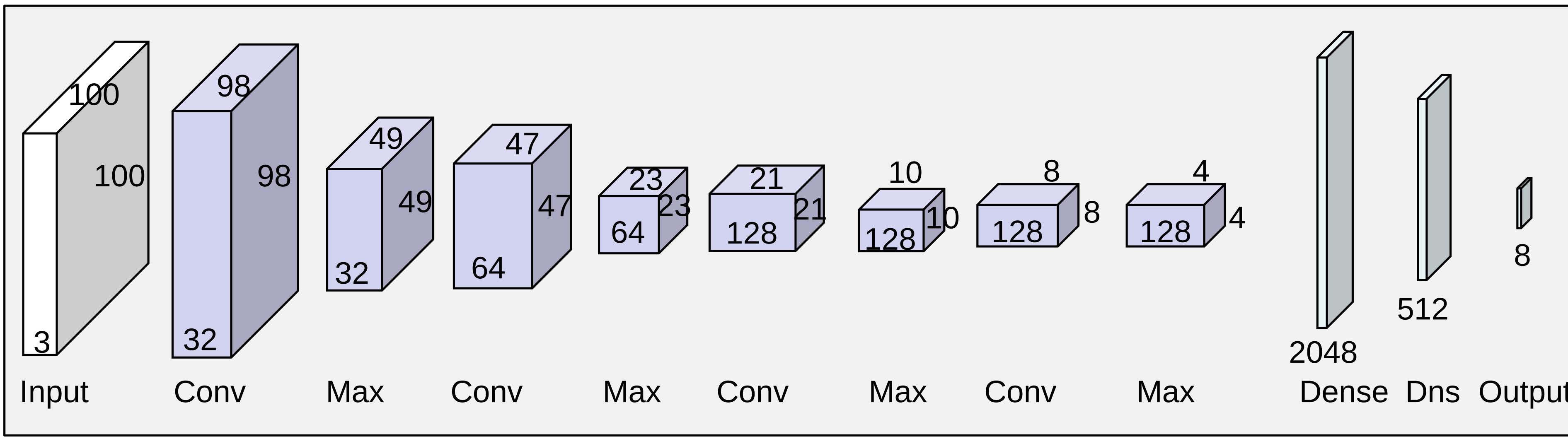


Network

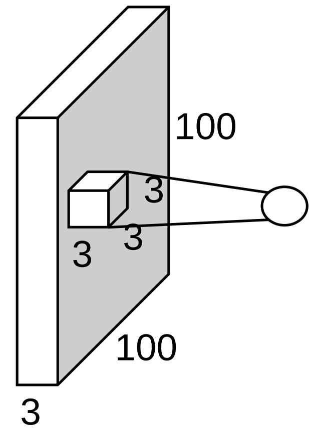
- **Ball**
- Goal Post
- Obstacle
- L
- X
- T
- Penalty Spot
- Foot

8

Convolutional Neural Networks
Example Architecture



Convolutional Neural Networks
Convolution Layer



- Filter (kernel) runs (convolves) over the input
- Calculates input for neuron in activation map
- Activation function here: ReLU
  - Y = ReLU(Wx+b)

W:

1	0	1
0	1	0
-1	0	1

x:

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

b:

-1

o:

	3		

ReLU (W * x + b) = o

- Repeated with n filters: n activation maps
- Padding: how do we deal with pixels at the border
- Strides: step size, > 1 means activation map shrinks

Convolutional Neural Networks
Pooling Layer

- Reduces the size of the activation map
- Pooling function
  - Usually Max-Pooling
  - Sometimes Avg-Pooling
- Example
  - 2x2 filter with 2,2 strides

2	3	5	7
6	4	3	2
1	2	3	2
0	1	1	0

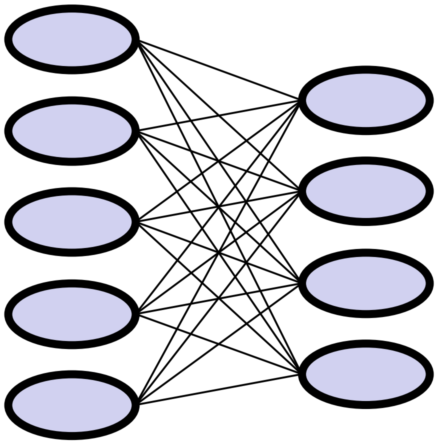
Input

6	7
2	3

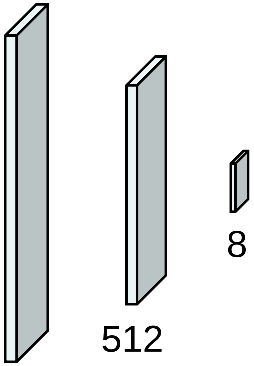
Output
Max- Pooling

Convolutional Neural Networks
Fully Connected Layer

- Already known: backpropagation layer
- Contains a majority of learnable weights
  - In our example
    - 2048 * 512 + 512 = 1.049.088 weights
    - 512 * 8 + 8 = 4.104 Gewichte
  - To compare with
    - 3*3*3*32 + 32 = 896 weights in the first convolution layer



- On a pretrained network it may suffice to only learn these layers on new images



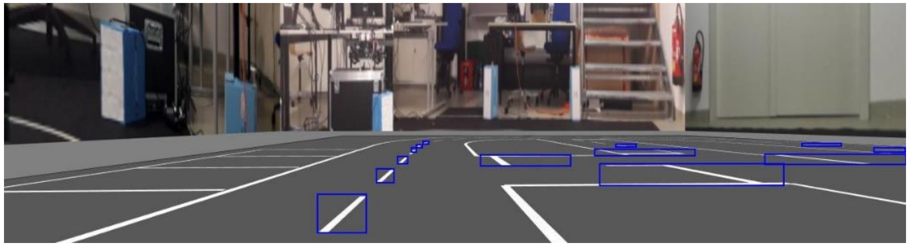
Convolutional Neural Networks
Learning

- Image tagging
  - A teacher has to assign the proper class to each image
  - In case of object localization also a bounding box is required
- Image learning
  - Present training image to network
  - Network calculates output and loss
  - Network performs gradient descent on loss function
- Accuracy is measured on test data
- Application
  - Stored network cann be applied to live images
  - Fast recall

Convolutional Neural Networks
Overfitting

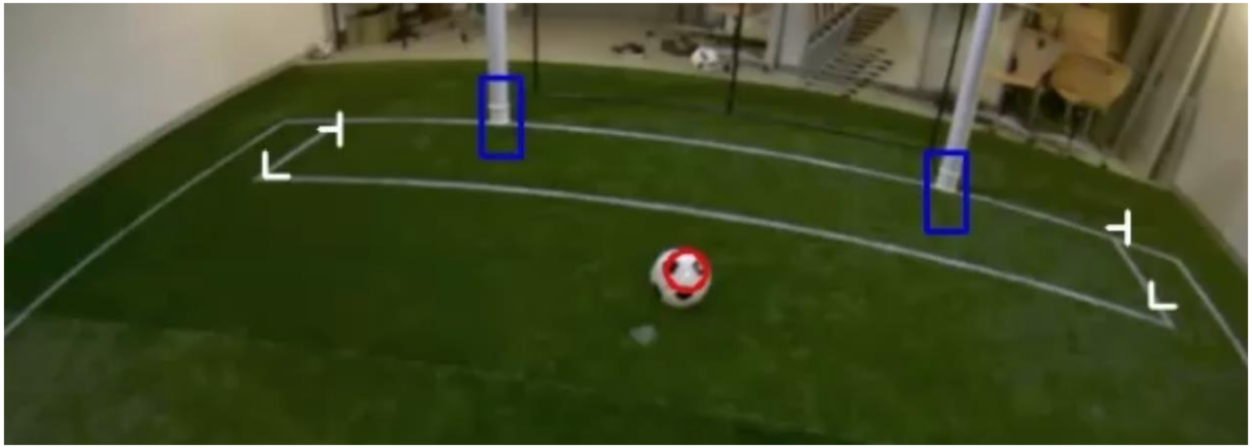
- Train many images
- What if we do not have too many?
  - Augment existing images

  - Create synthetic images

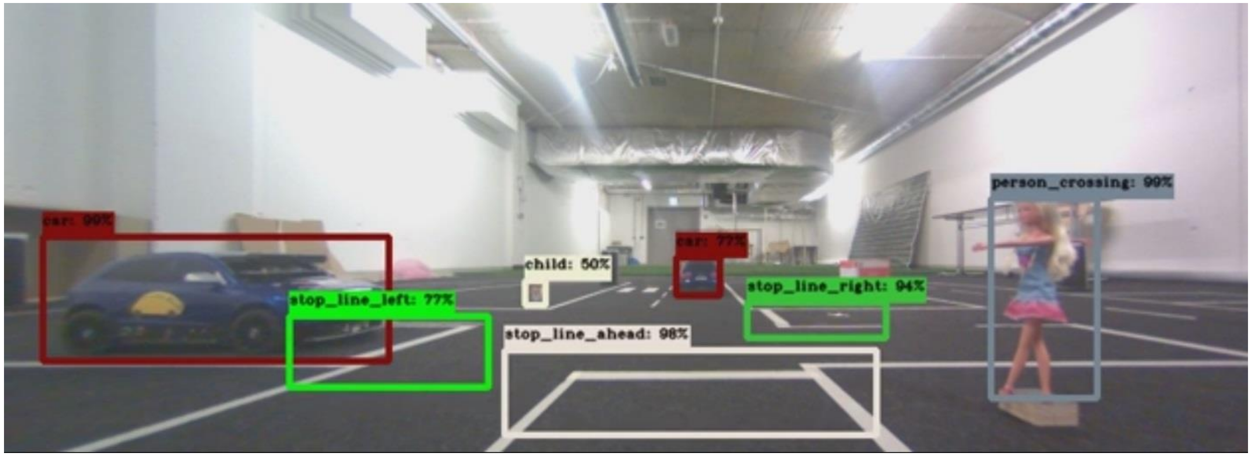


- Dropout
  - Do not use a fraction (e.g. 0.5) of random neurons during learning
  - Avoids single ‚important' connections

- RoboCup

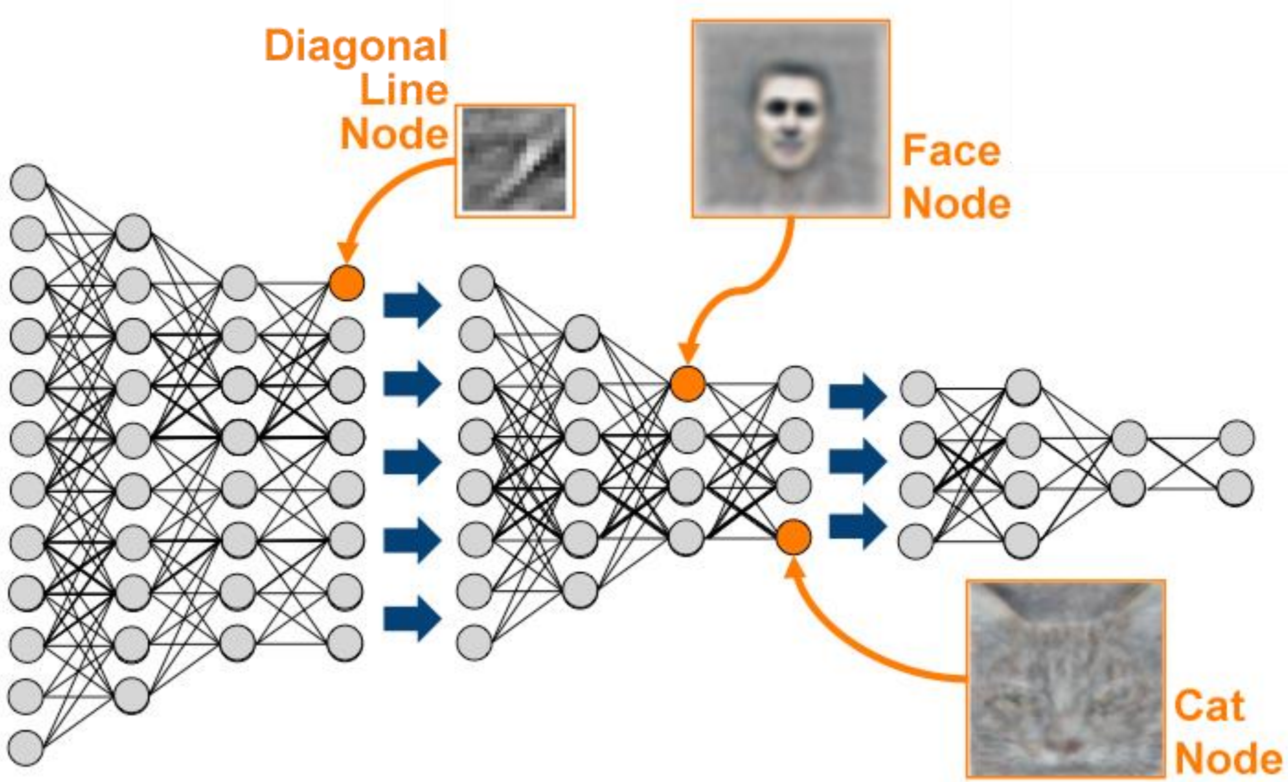


- AudiCup



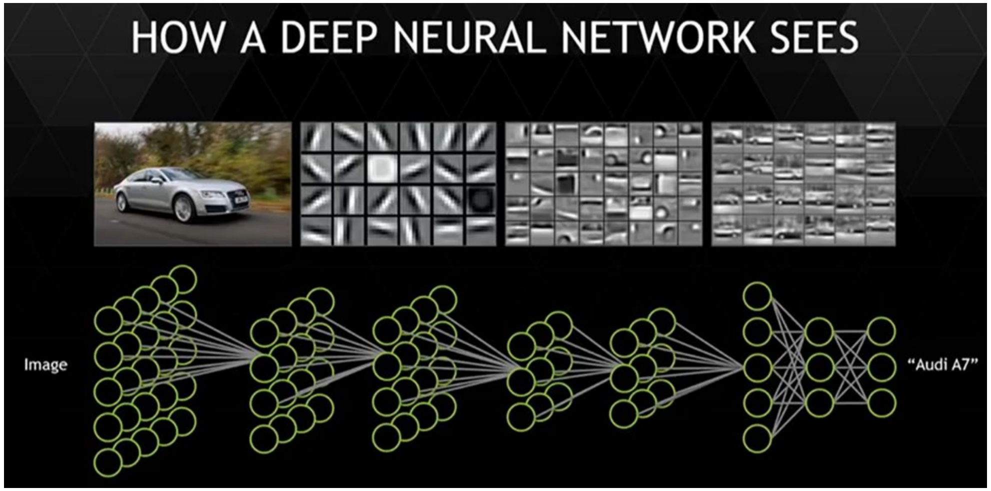
Deep Learning
Applications: Google

- Input
  - 10 mio images (200 x 200 pixel)
- Learning
  - 1 Mio weights
  - 16.000 cores
  - 3 days



Deep Learning
Applications: NVIDIA

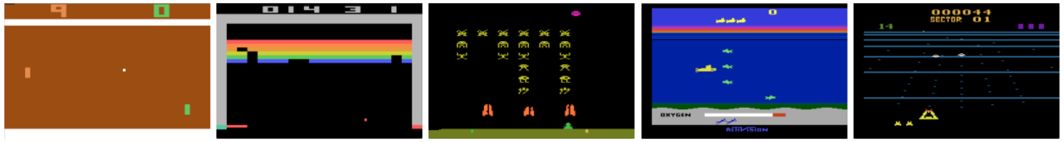
- Example NVIDIA
  - First layer detects lines and circles
  - Parts of a car
  - Cars
  - Car types
- Learning
  - Days
    (on a GPU cluster)
- Recall
  - 2 Megapixel
  - 30 fps
  - 75 objects



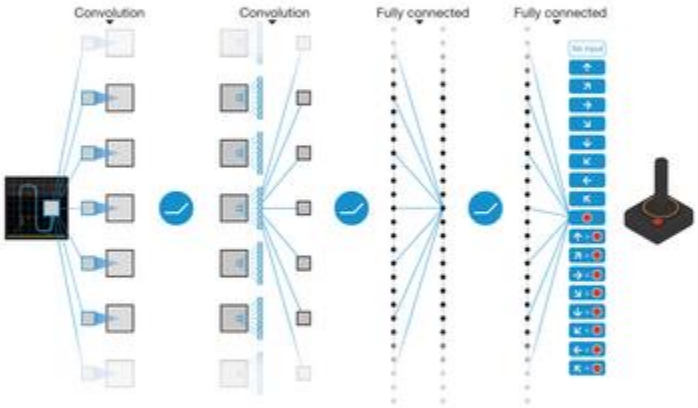
https://www.youtube.com/watch?v=zsVsUvx8ieo

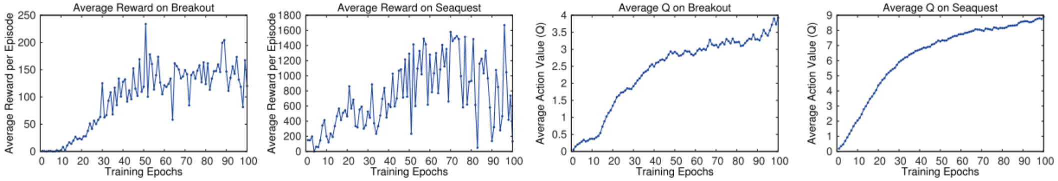
Deep Learning
Applications: DeepMind

- Deep Reinforcement Learning of computer games



- Input
  - 84*84*4 downsampled live video input
- Network
  - 4 layers, 2 convolutional (8x8, 4x4), 2 fully connected
- Result
  - 4 of 7 games played better than human expert

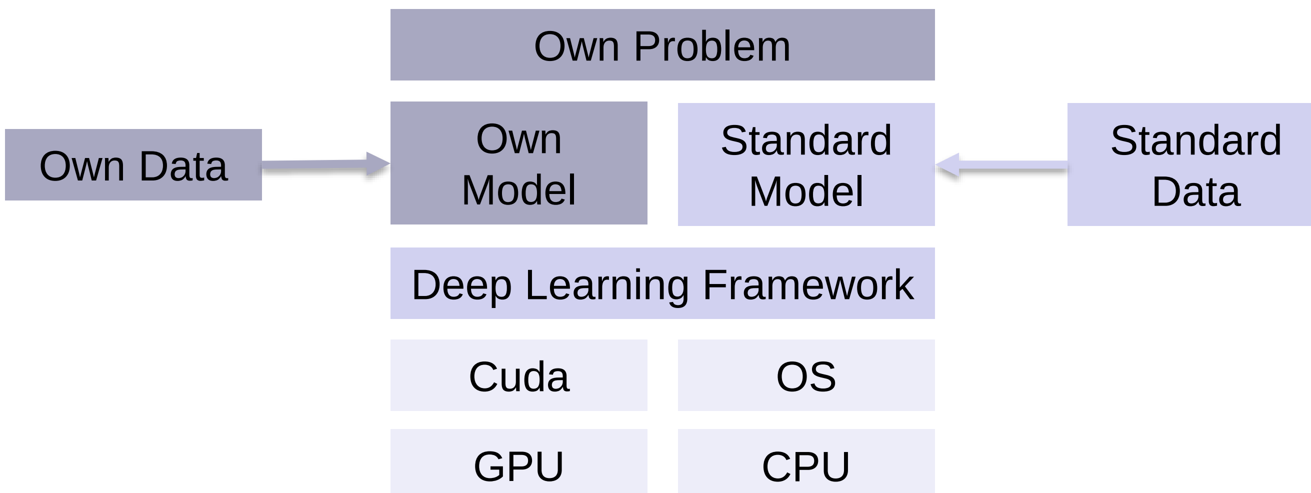




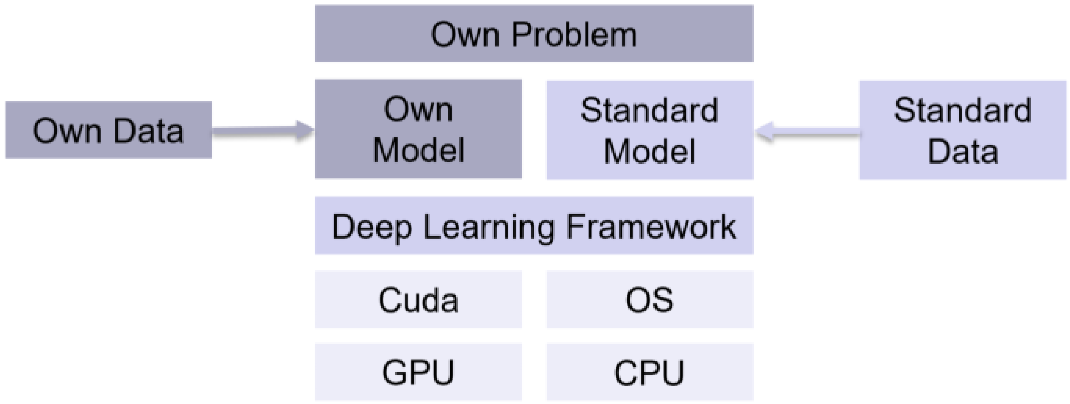
https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf und http://www.nature.com/nature/journal/v518/n7540/full/nature14236.h

Deep Learning
Frameworks

Hochschule Offenburg
offenburg.university

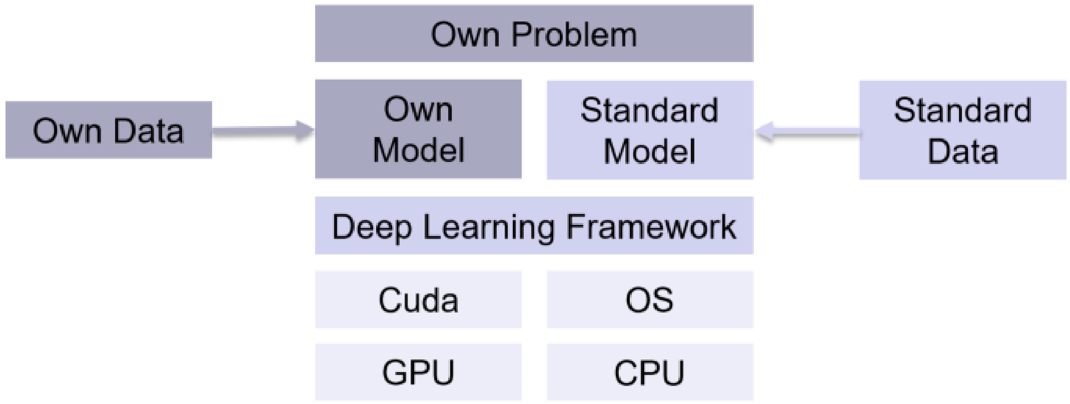


- TensorFlow
  - Google Brain team
  - https://www.tensorflow.org/
- Torch, PyTorch
  - Communities
  - http://pytorch.org/
- Deeplearning4j
  - Skymind engineering team, Deeplearning4j community
  - https://deeplearning4j.org/
- Caffe
  - Berkeley Vision and Learning Center
  - http://caffe.berkeleyvision.org/
- Caffe2
  - Facebook
  - https://research.fb.com/downloads/caffe2/



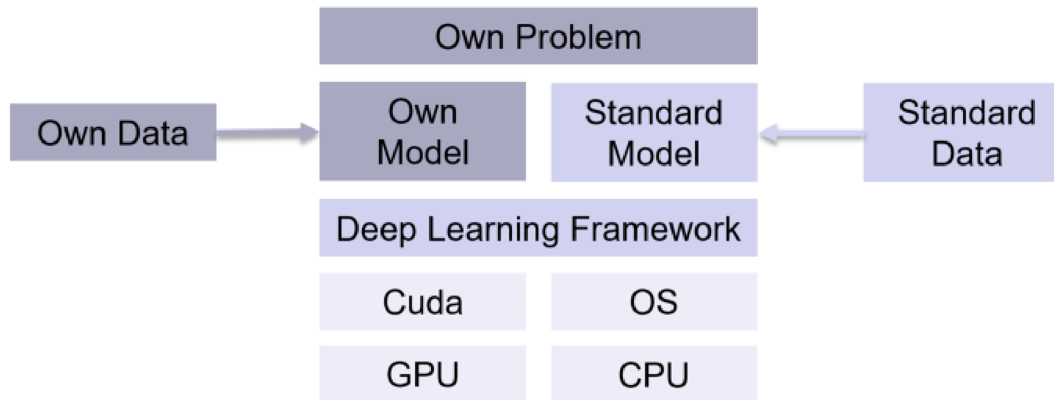
Deep Learning
Standard Data

- ImageNet
  - 14 Mio tagged images
  - 21.000 categories
  - http://www.image-net.org
- MNIST
  - 70.000 hand written digits
  - http://yann.lecun.com/exdb/mnist/
- COCO (common objects in context)
  - 200.000 tagged and segmented images
  - http://cocodataset.org/#home
- Music, faces, speech, texts, …



Deep Learning
Standard Network Models

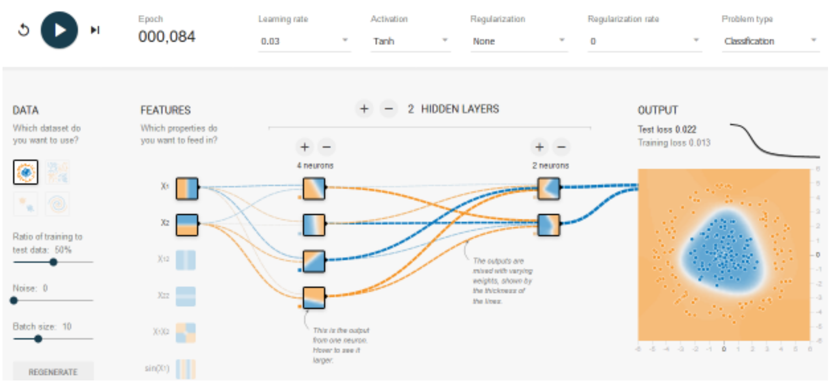
- LeNet-5 (1990)
  - 5 layers (4,1)
  - MNIST
- AlexNet (2012)
  - 8 layers (5,3)
  - ImageNet (16.4% Fehler)
- GoogLeNet (2014)
  - 22 layers
  - ImageNet (6.7%)
- ResNet-152 (2015)
  - 152 layers
  - ImageNet (3.6%), COCO



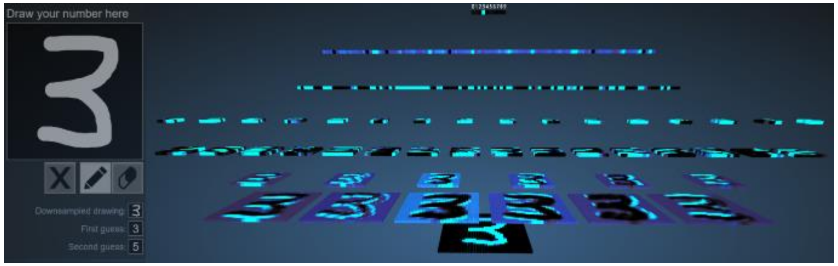
- VGGNet
- Mobilenet
- Inception
- …

Deep Learning
Visualizations

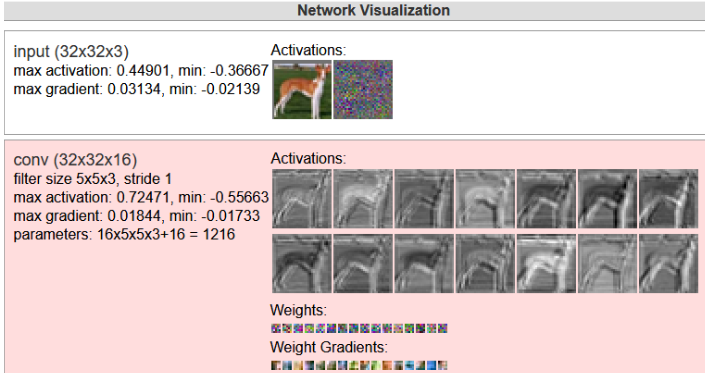
- Google Playground
  - http://playground.tensorflow.org



- 3D digit recognition (Adam Harley)
  - http://scs.ryerson.ca/~aharley/vis/conv



- ConvnetJS (Andrej Karpathy)
  - https://cs.stanford.edu/people/karpathy/convnetjs



Deep Learning
Summary

- Deep Convolutional Neural Networks
  - Convolution layers
  - Max layers
  - Dense layers
- Applications
  - Image recognition, speech recognition, predictive maintenance, …
- Many pretrained networks available for download

Europäischer Fonds für Regionale Entwicklung (EFRE)
Der Oberrhein wächst zusammen: mit jedem Projekt
UpperRhine 4.0