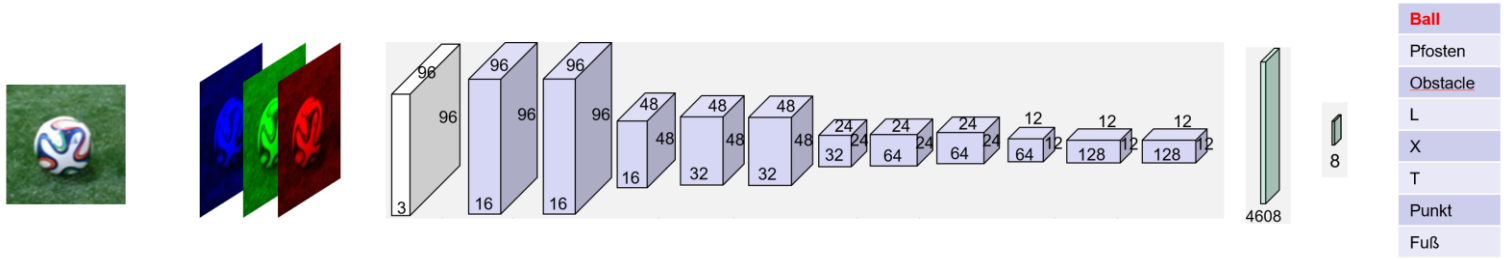
Summer School 2020

Deep Learning

Prof. Dr. Klaus Dorer



Hochschule Offenburg
offenburg.university

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Übersicht

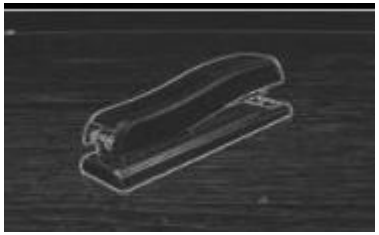
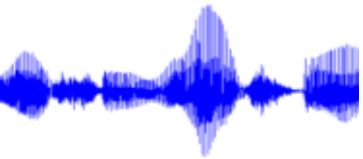
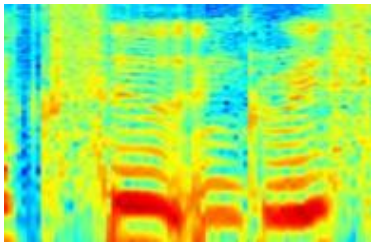
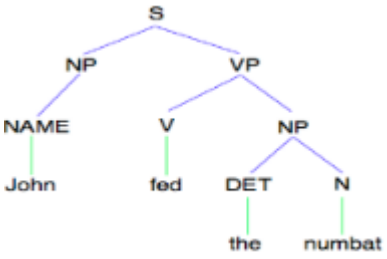
- Neuronale Netzwerke
  - Einführung
  - Modell einer Nervenzelle
  - Perceptron
  - Backpropagation Networks
  - Convolutional Neural Networks

- Ziele
  - Elemente tiefer neuronaler Netze kennen
  - Anwendungsmöglichkeiten einschätzen können

Warum ist Deep Learning so aktuell?

- Mehr Daten
  - Youtube: jede Minute werden 500 Stunden Video hochgeladen
  - Facebook: täglich werden 300 Millionen Bilder hochgeladen
- Mehr Rechenpower
  - GPU Beschleunigung
  - GPU/CPU Cluster
- 'Neue' Deep Learning Ansätze
  - Deep Neural Networks
  - Convolutional Neural Networks
  - Bessere Aktivierungsfunktionen, Optimizer, Initialisierung, …
- Frei verfügbare Frameworks
  - Theano, TensorFlow, DeepLearning4J, …

Maschinelles Lernen vor Deep Learning

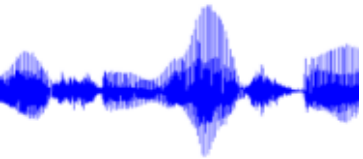
Input	Feature Extraktion (Feature Engineering)	Klassifizierer Detektor	Ergebnis
		SVM, (flaches) neuronales Netz, …	
		HMM, (flaches) neuronales Netz, …	Sprechererkennung, Spracherkennung, ...
		Clustering, HMM, …	Themenerkennung, maschinelle Übersetzung, ...

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

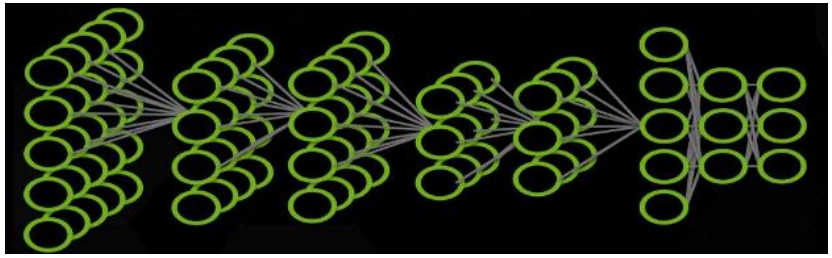
Maschinelles Lernen mit Deep Learning

Input

Ergebnis



Deep
Network



"Tacker"

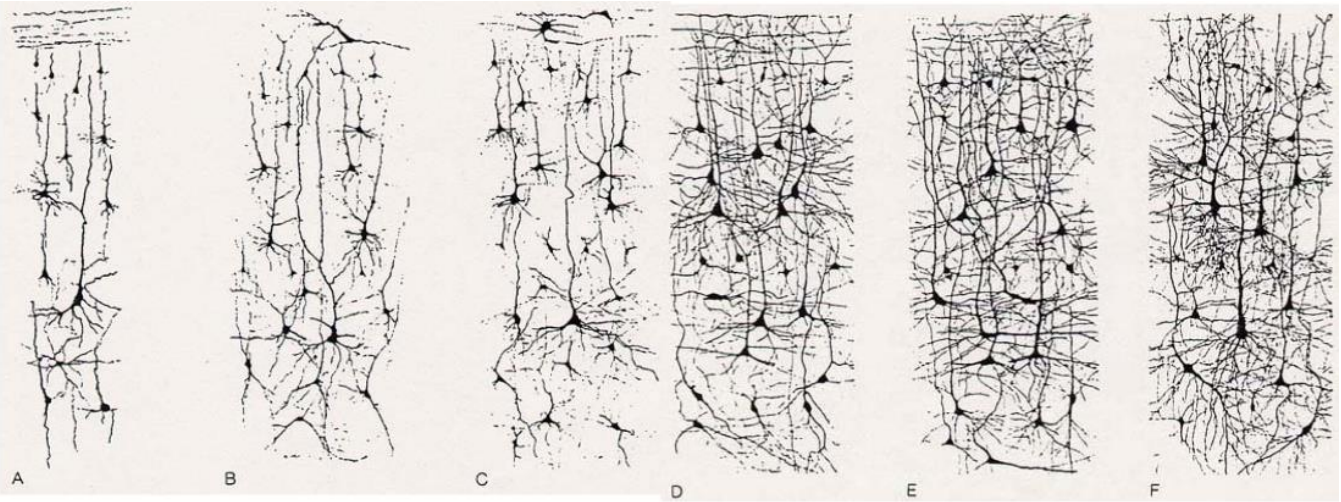
"Tacker"

"Bedienungsanleitung
für einen Tacker"
....

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

Menschliches Gehirn

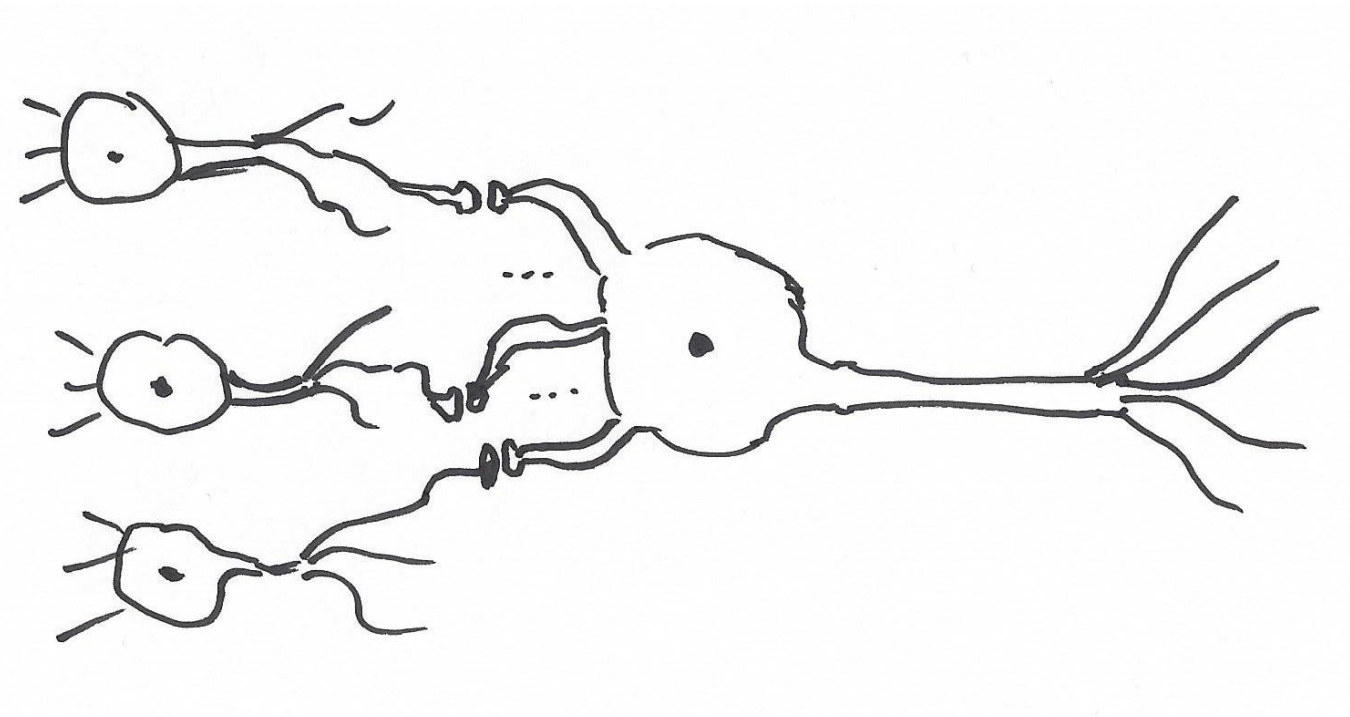
- Lernen erfolgt durch
  - Bildung neuer Verbindungen zwischen Nervenzellen
  - Verstärkung und Abschwächung von Verbindungen



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

Menschliches Gehirn

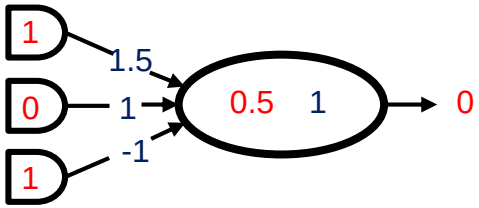


Menschen Lernen Maschinelles Lernen

Deep Learning

Modell einer Nervenzelle

Prof. Dr. Klaus Dorer



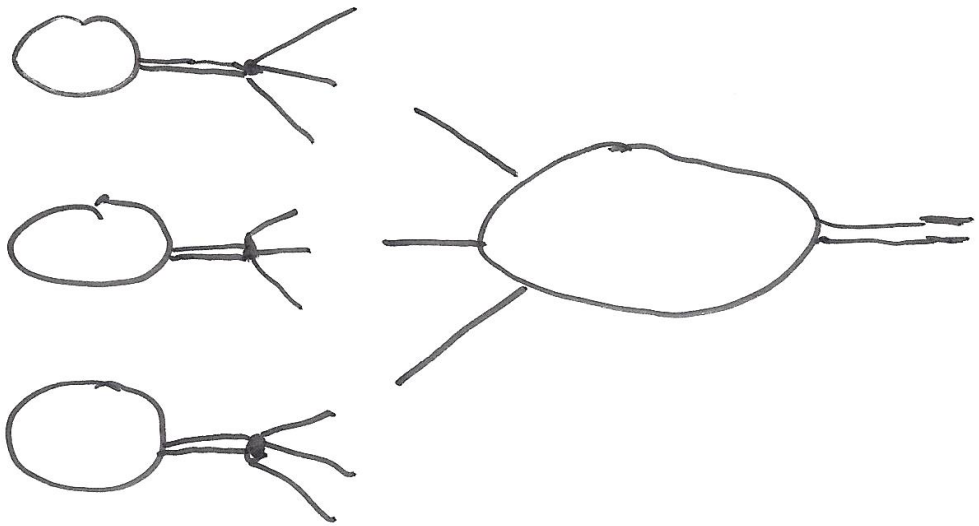
Hochschule Offenburg
offenburg.university

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Übersicht

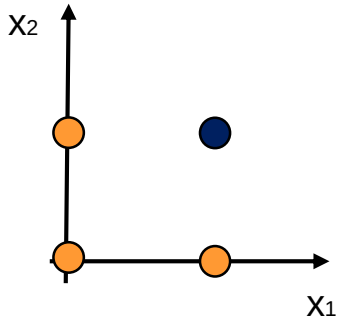
- Neuronale Netzwerke
  - Einführung
  - Modell einer Nervenzelle
  - Perceptron
  - Backpropagation Networks
  - Convolutional Neural Networks

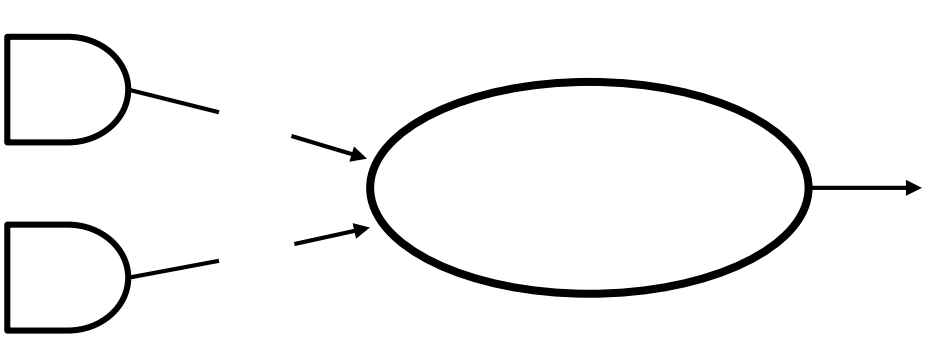
Modell einer Nervenzelle

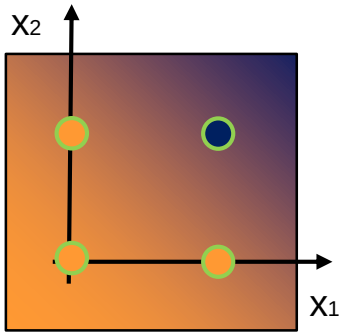


Modell einer Nervenzelle
Beispiel: And Funktion

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



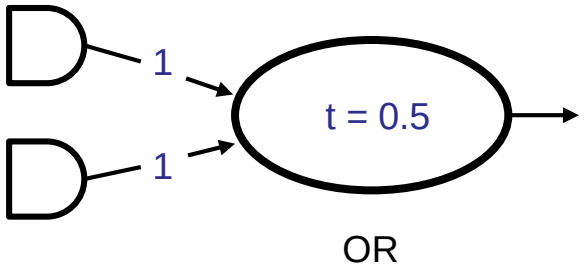




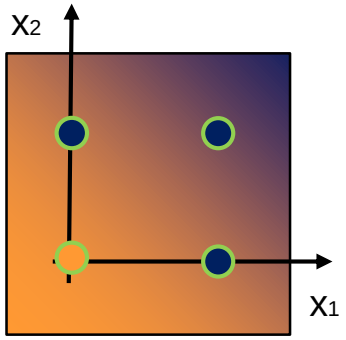
$$o_j = Step_t(\sum_i w_{i,j} x_i) = Step_t(Wx)$$

Modell einer Nervenzelle
Or und Not Funktion

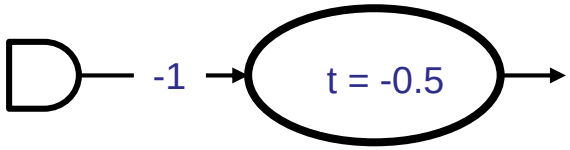
x_1	x_2	y_1
0	0	0
0	1	1
1	0	1
1	1	1



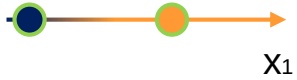
OR



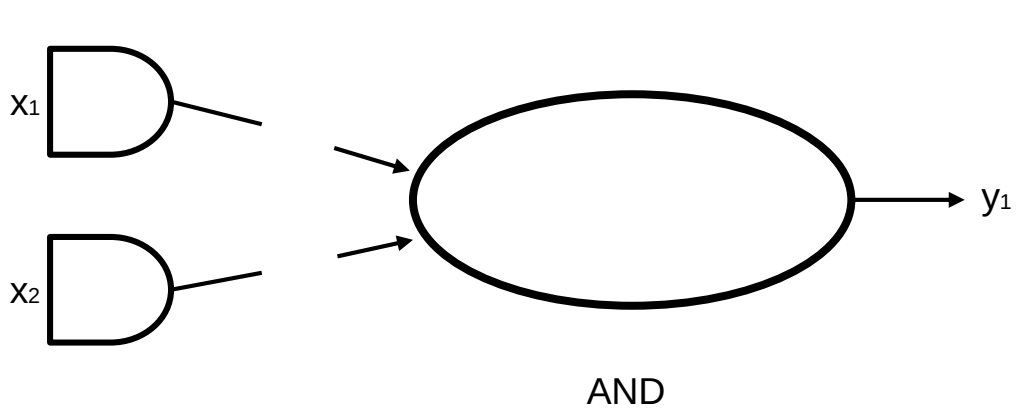
x_1	y_1
0	1
1	0



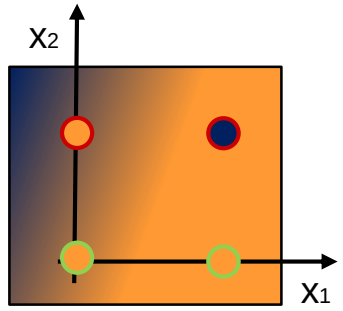
NOT

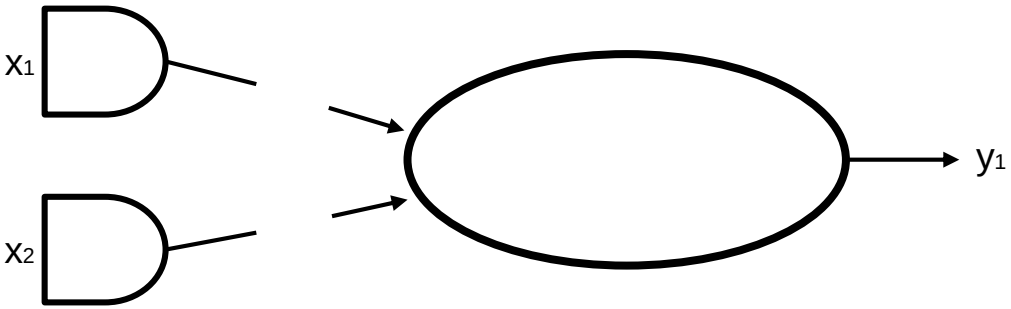


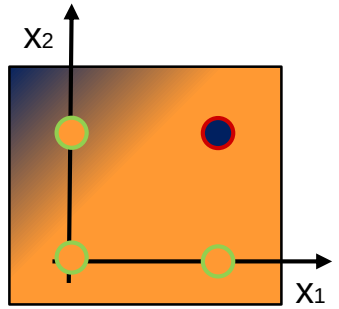
Modell einer Nervenzelle
Passende Werte finden



AND

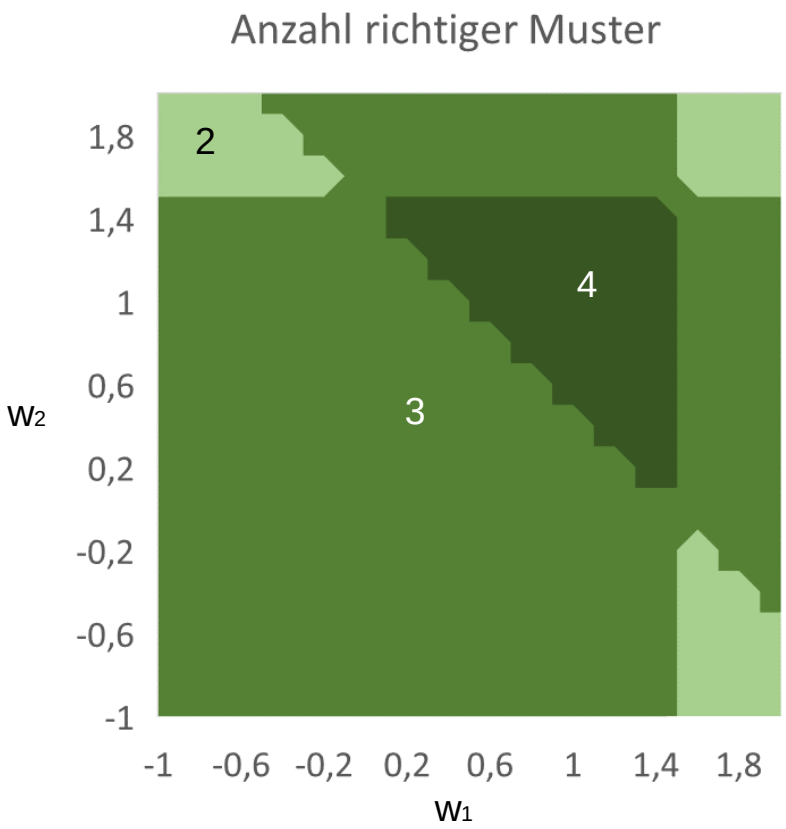




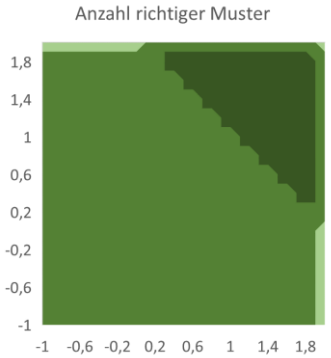


Modell einer Nervenzelle
Passende Werte finden

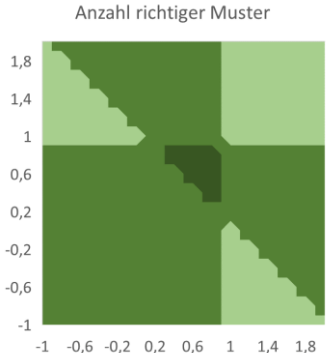
t=1.5



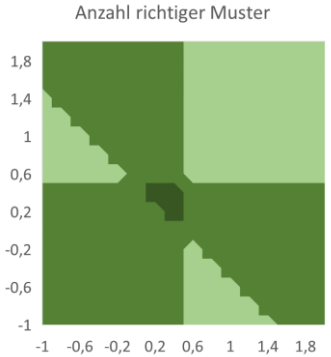
t=2



t=1

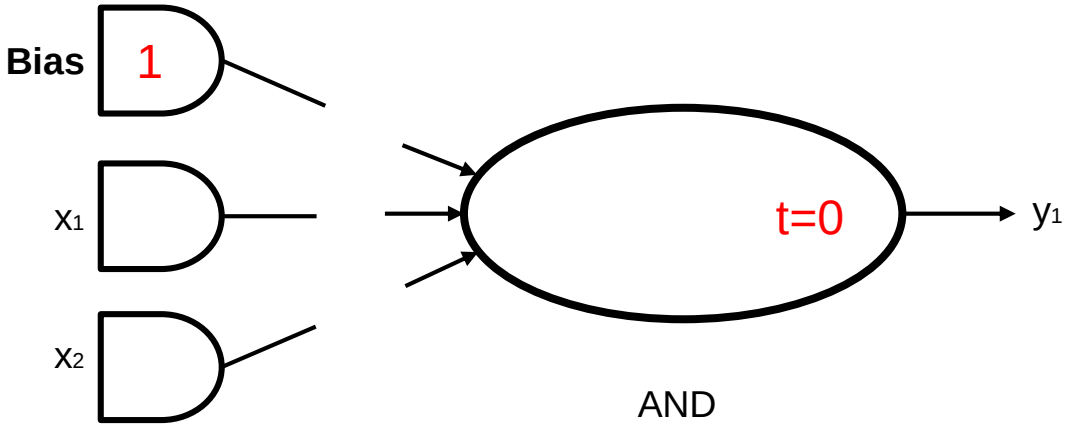


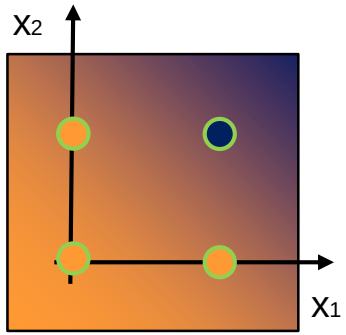
t=0.5



Modell einer Nervenzelle
Passende Werte finden

- Bias





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

Modell einer Nervenzelle
Passende Werte finden

- Gewichte w seien -1 und 1
- Fehlerfunktion (loss function)
  - Betragsfehler (L1 Norm)

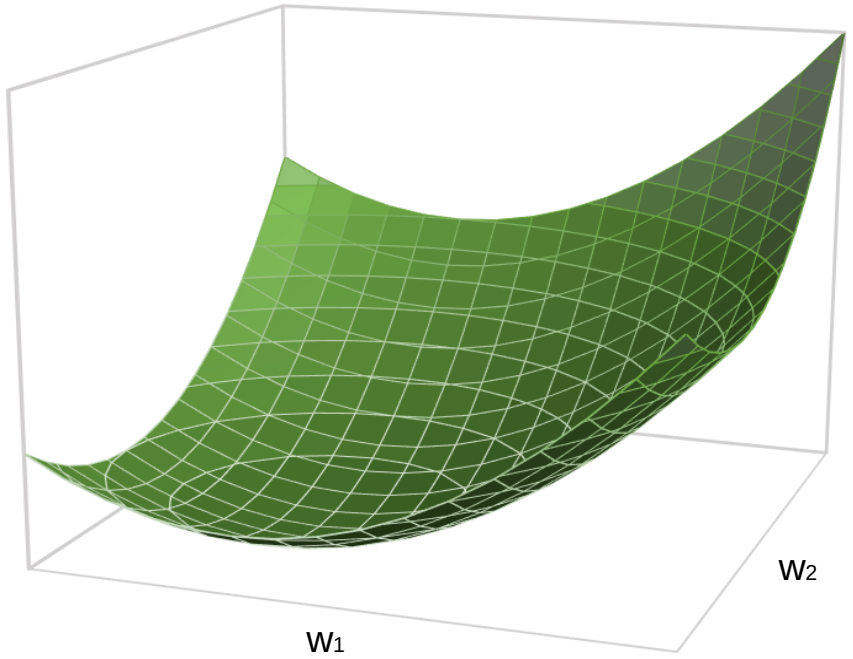
$$l = \sum_i | y_i - o_i | = || y - o ||_1$$

  - Summe der Fehlerquadrate (L2 Norm)

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

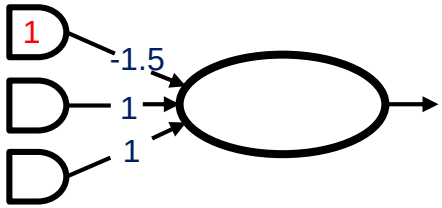
- Gradientenabstieg (gradient descent)

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	



Modell einer Nervenzelle
Zusammenfassung

- Inputberechnung
- Bias
- Aktivierungsfunktion

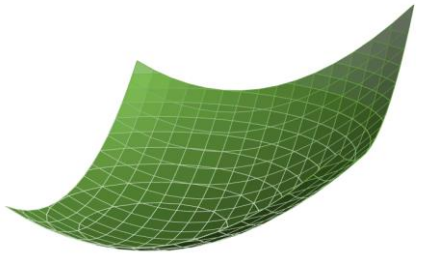


$$o_j = Step_0(\sum_{i+1} w_{i,j} x_i)$$

- Fehlerfunktion

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

- Gradientenabstieg

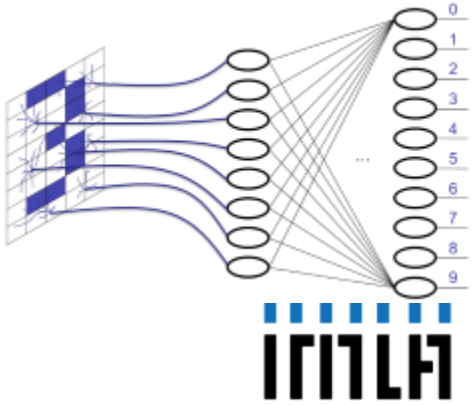


Menschen Lernen Maschinelles Lernen

Deep Learning

Perceptron

Prof. Dr. Klaus Dorer



Hochschule Offenburg
offenburg.university

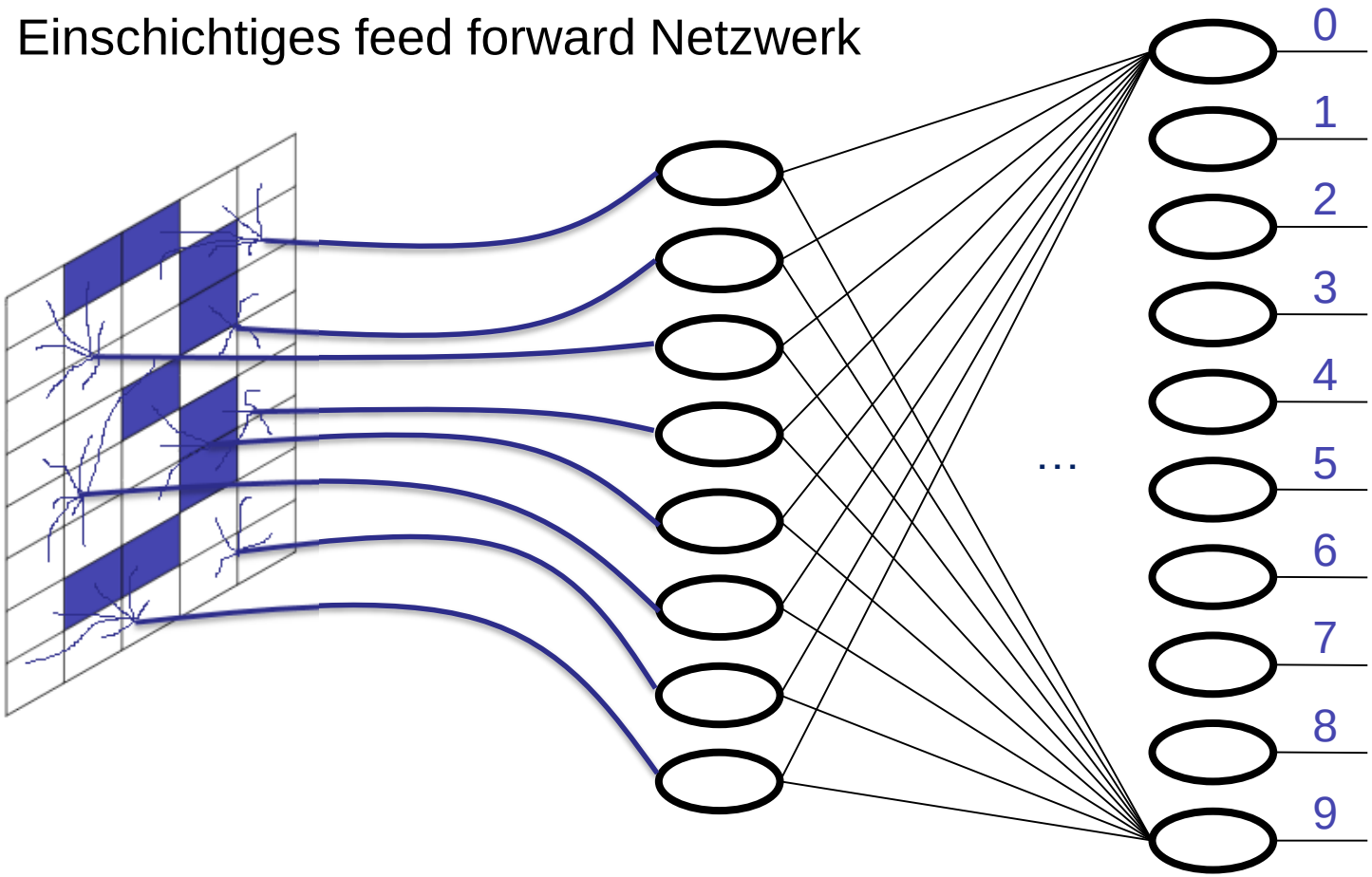
GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Übersicht

- Neuronale Netzwerke
  - Einführung
  - Modell einer Nervenzelle
  - Perceptron
  - Backpropagation Networks
  - Convolutional Neural Networks

Perceptron

- Von Frank Rosenblatt 1958 vorgestellt
- Einschichtiges feed forward Netzwerk



Perceptron

- Berechnung der Outputs

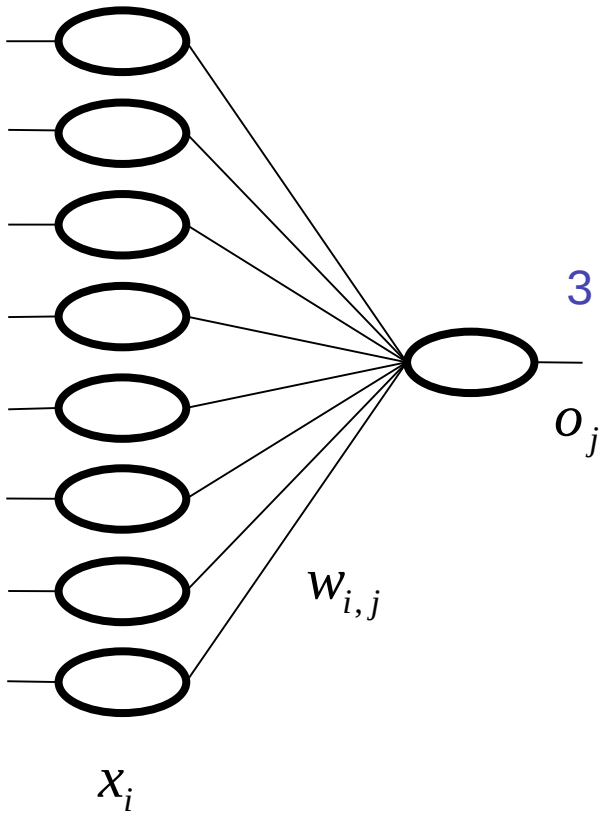
$$o_j = Step_0(\sum_{i+1} w_{i,j} x_i)$$

- Lernregel (Delta Regel)

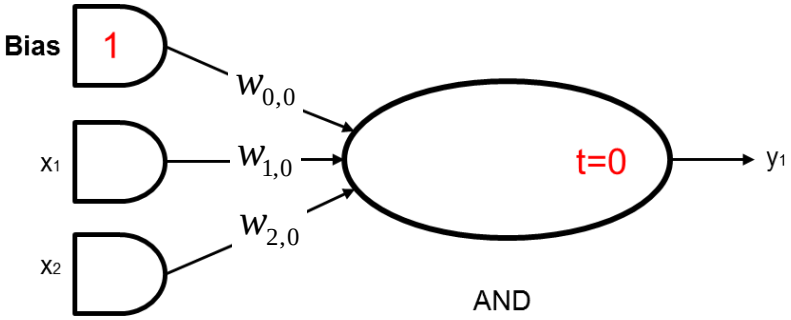
$$w_{i,j} = w_{i,j} + \alpha \cdot x_i \cdot (y_j - o_j)$$

- Algorithmus

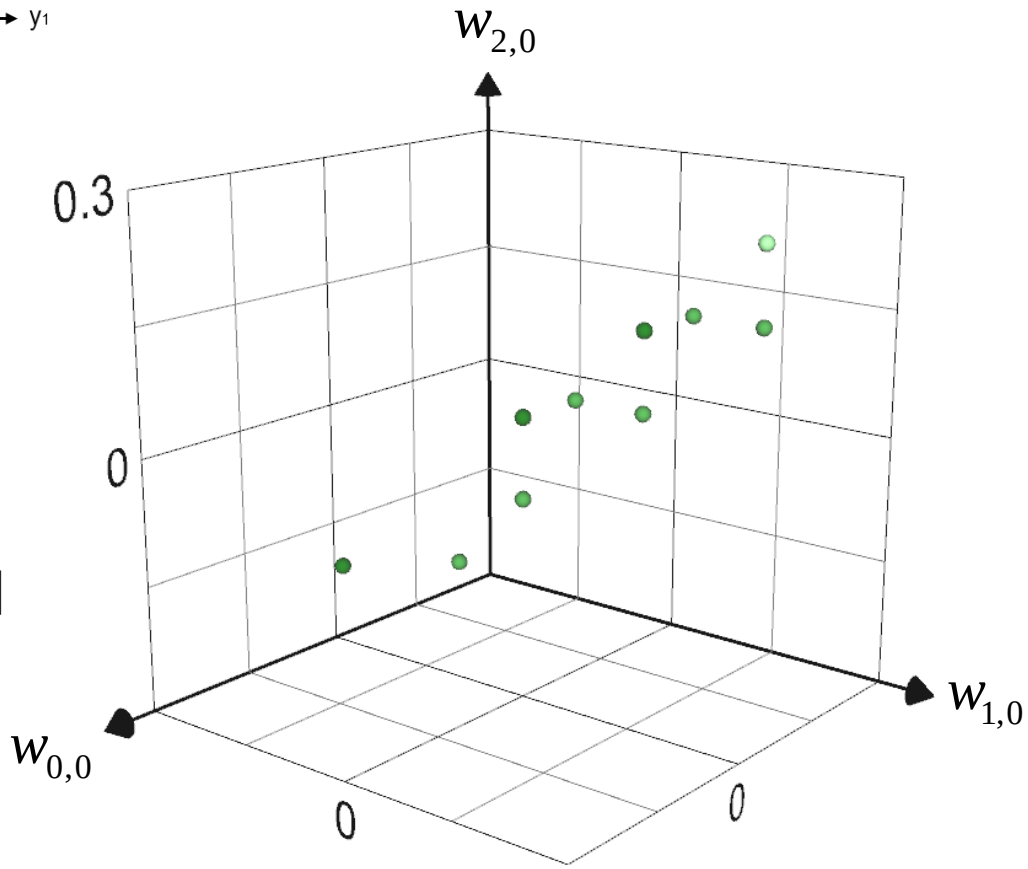
```
initialisiere Gewichte zufällig
do
   for each e in examples
      berechne Output
      passe Gewichte an
while (Fehler zu groß und Abbruch-
       kriterium nicht erreicht)
```



Perceptron
And Funktion

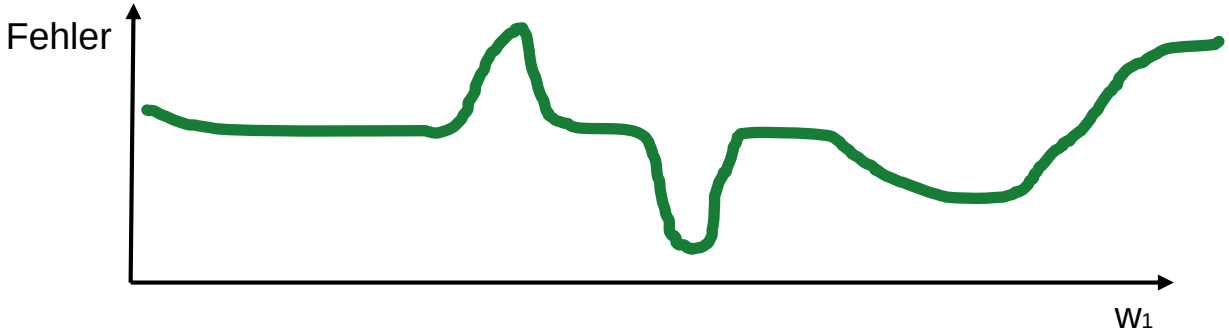


- Zufälliger Anfang
  - $w_{x,0}$ [0.028; -0.258; -0.189]
  - 3 Muster falsch
- 4 Muster anlegen und lernen
  - $w_{x,0}$ [-0.072; -0.158; -0.189]
  - 2 Muster falsch
- …
  - 0 Muster falsch

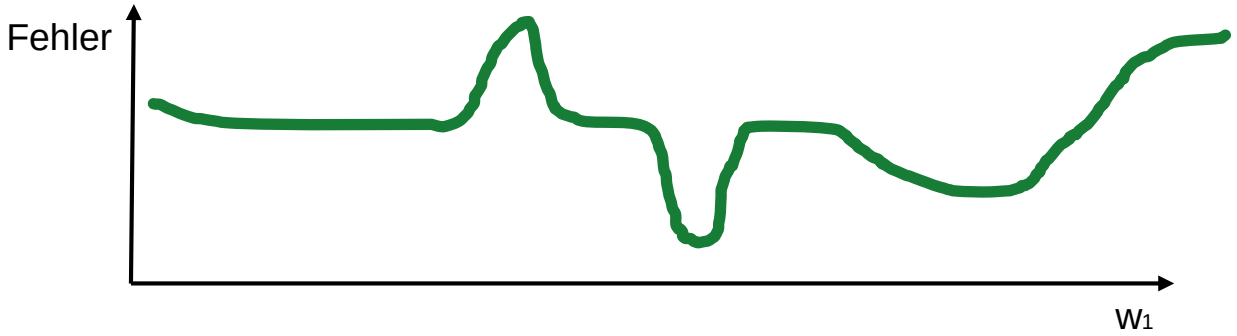


Perceptron
Lernrate und Gradientenabstieg

- Kleine Lernrate



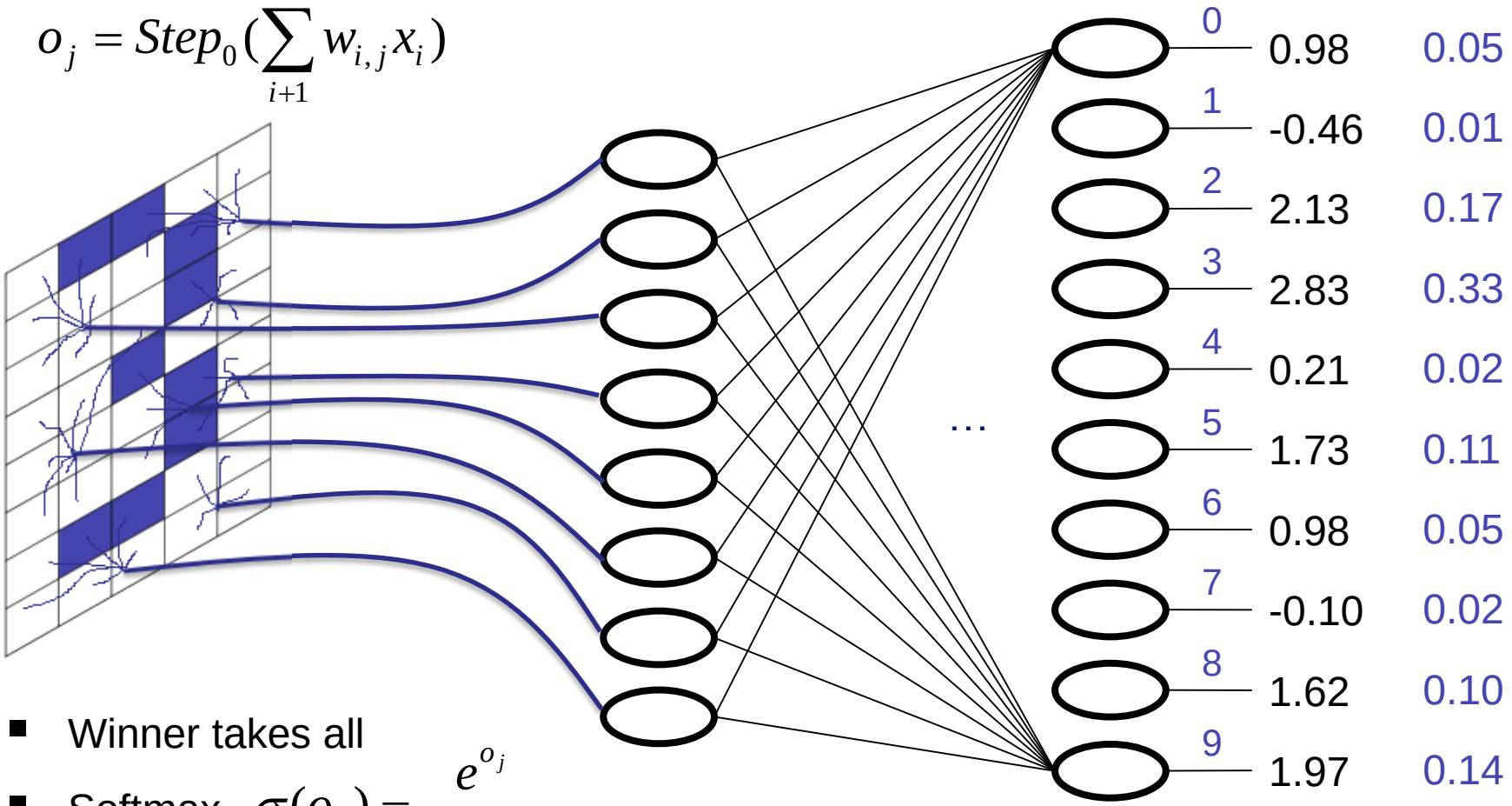
- Große Lernrate



- Stochastic Gradient Descent
- Batch Learning

Perceptron
Reelwertige Inputs und Outputs

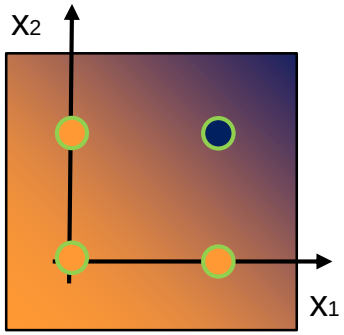
$$o_j = Step_0(\sum_{i+1} w_{i,j} x_i)$$



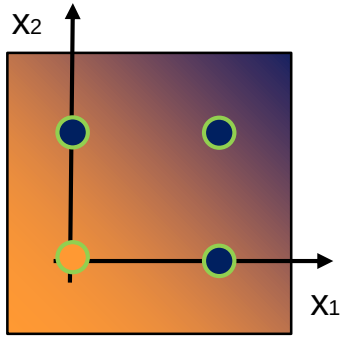
- Winner takes all
- Softmax $\sigma(o_j) = \frac{e^{o_j}}{\sum_k e^{o_k}}$

Perceptron
Ausdrucksfähigkeit

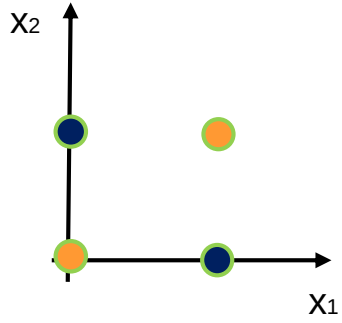
- Kann nur linear separierbare Probleme repräsentieren



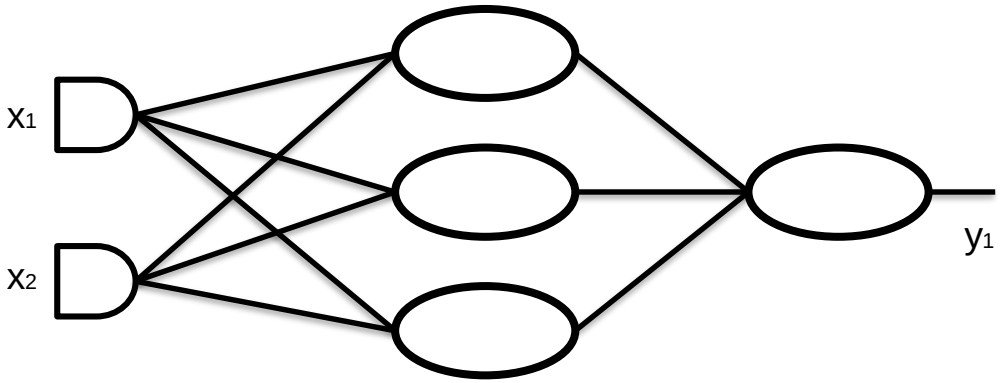
AND



OR



XOR



Perceptron
Zusammenfassung

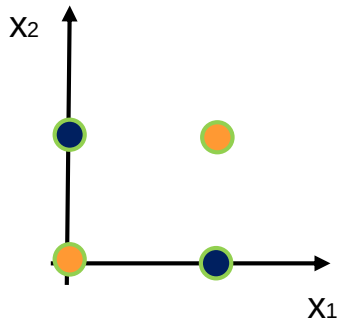
- Lernregel
  - Lernrate
  - Stochastic/Batch Gradient Descent

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

- Outputfunktion
  - Winner takes all
  - Softmax

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

- Problem linearer Separierbarkeit



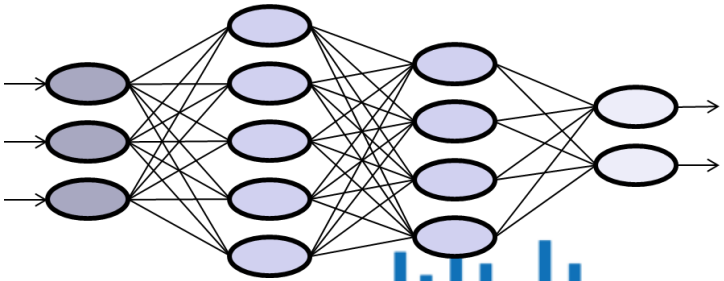
XOR

Menschen Lernen Maschinelles Lernen

Deep Learning

Backpropagation Networks

Prof. Dr. Klaus Dorer



Hochschule Offenburg
offenburg.university

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Übersicht

- Neuronale Netzwerke
  - Einführung
  - Modell einer Nervenzelle
  - Perceptron
  - Backpropagation Networks
  - Convolutional Neural Networks

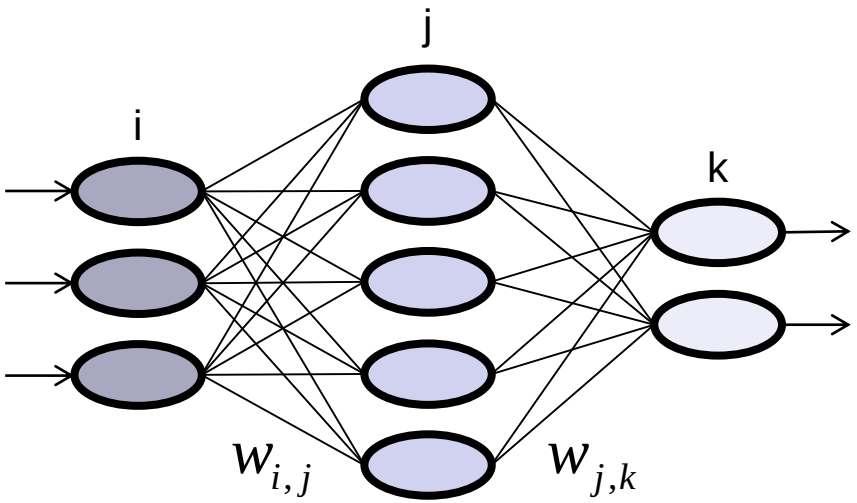
Backpropagation Networks

- Um XOR oder ähnliche Probleme zu lernen benötigt man
  - Mehrschichtige Netzwerke
  - Eine nicht-lineare Aktivierungsfunktion
- Problem
  - Wie propagiere ich den Fehler an den Outputs zurück in die Hidden Units und passe deren Gewichte an?
- Lösung
  - Backpropagation of Error
  - Bryson & Ho 1969, Rumelhart, Hinton & Wiliams 1986

Backpropagation Networks
Lernregel

- Berechnung der Outputs

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



- Lernregel
  - Gewichte zur Outputschicht

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

  - Gewichte zu Hiddenschichten

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

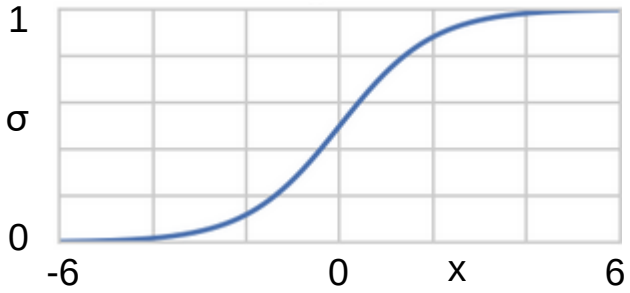
Backpropagation Networks
Algorithmus

- Algorithmus

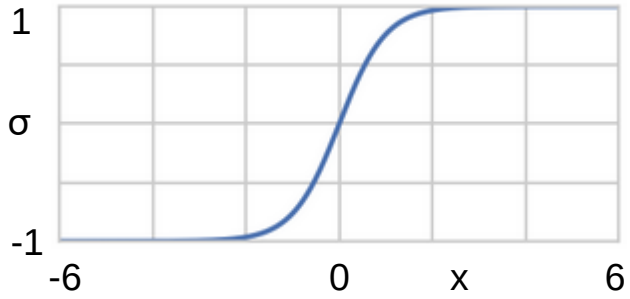
```
initialisiere Gewichte zufällig
do
   for each e in examples
    berechne Output (recall)
    berechne Δ Werte für die Output Units
    wiederhole für jede Schicht (rückwärts von den outputs)
      propagiere Δ Werte zurück zur vorigen Schicht
      berechne neue Gewichte
while (Fehler zu groß und Abbruchkriterium nicht erreicht)
```

Backpropagation Networks
Aktivierungsfunktion

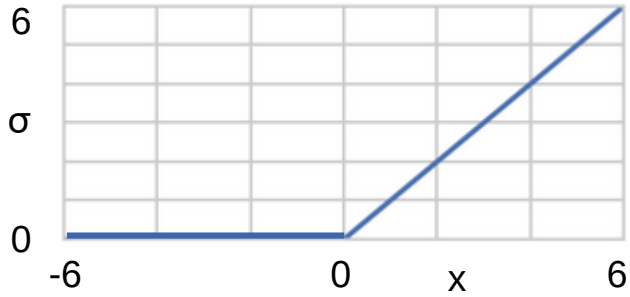
- Sigmoid
  - Funktion $\sigma(x) = \frac{1}{1+e^{-x}}$
  - Ableitung $\sigma'(x) = \sigma(x)(1-\sigma(x))$



- Tanh
  - Funktion $\sigma(x) = \tanh(x)$
  - Ableitung $\sigma'(x) = 1-\sigma(x)^2$

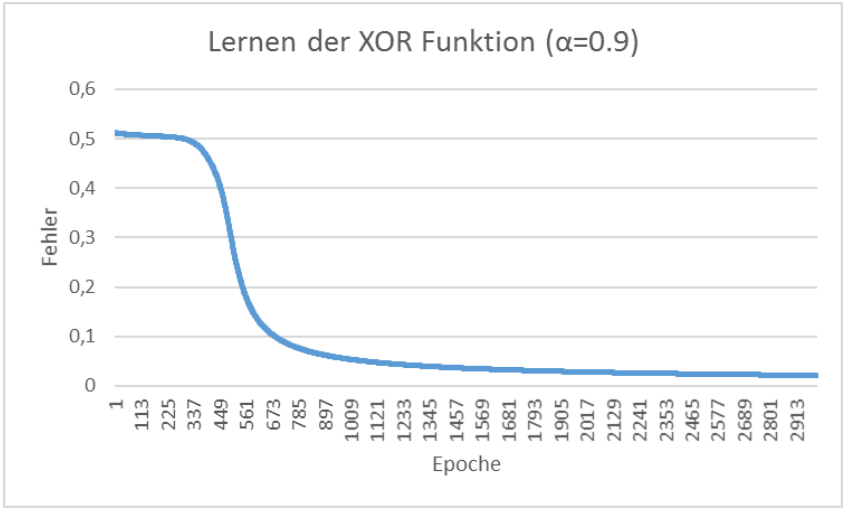


- Relu (Rectified linear unit)
  - Funktion $\sigma(x) = \max(0,x)$
  - Ableitung $\sigma'(x) = \begin{cases} 0, falls\ x < 0 \\ x, falls\ x \geq 0 \end{cases}$

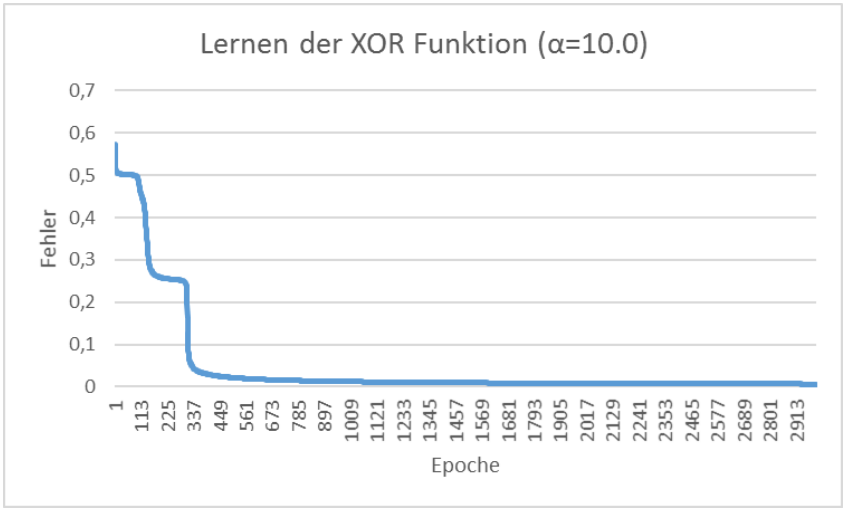


Backpropagation Networks
Beispiel: XOR Funktion

- Ein 3 – 3 – 1 Netzwerk kann das XOR Muster lernen
  - Lernrate 0.9

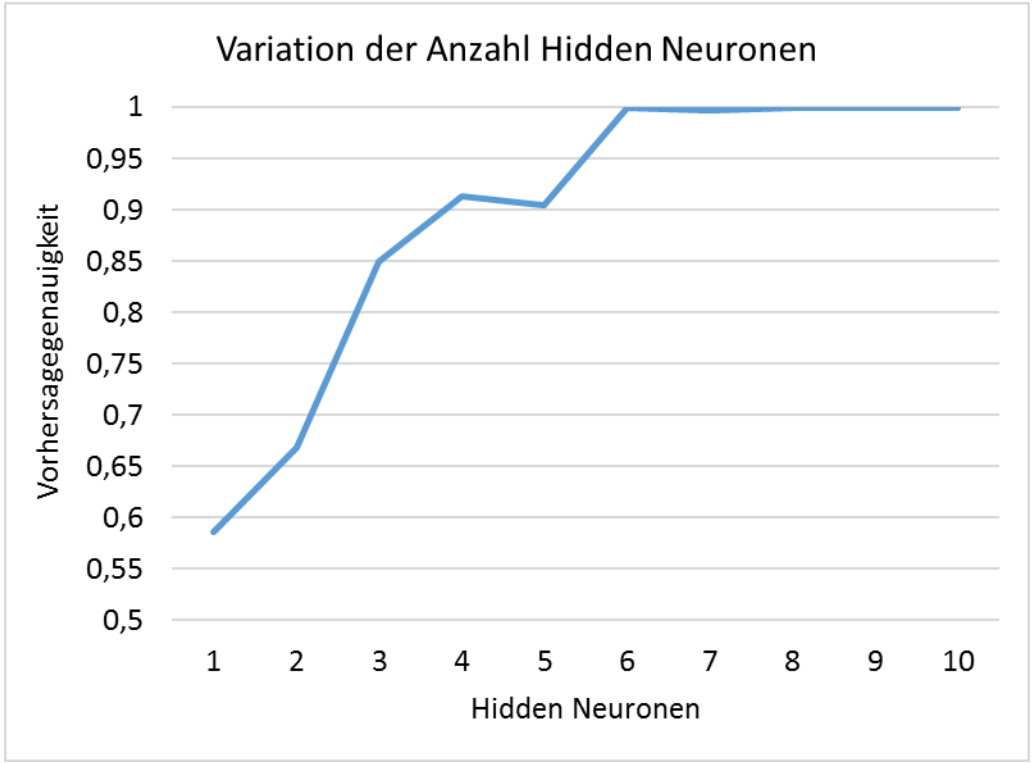


  - Lernrate 10



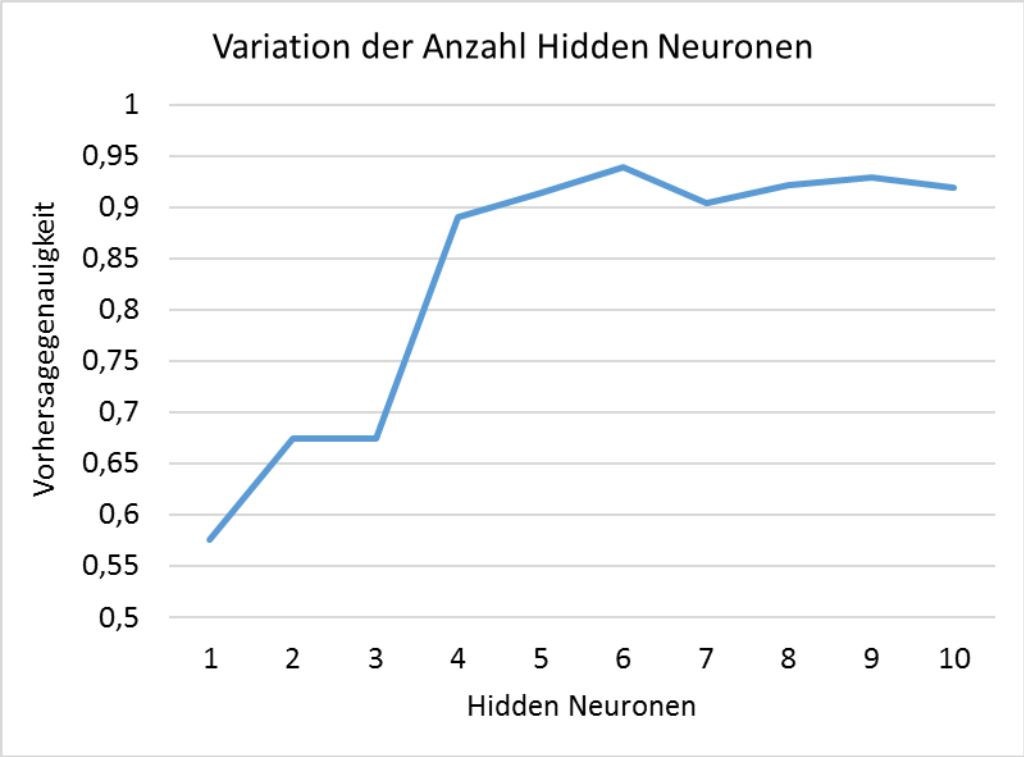
Backpropagation Networks
Beispiel: Klassifikation

- Wie groß muss das Netzwerk sein?
  - Um die Trainingsmuster zu lernen?
  - Test der Vorhersagegenauigkeit auf den Trainingsdaten
- Datensatz
  - 864 Beispiele Training
  - 864 Beispiele Test
  - 6 Attribute (4,4,4,3,3,3)
  - Ziel: ja/nein
- Netzwerk
  - 21 Input Neuronen
  - x Hidden Neuronen
  - 2 Output Neuronen



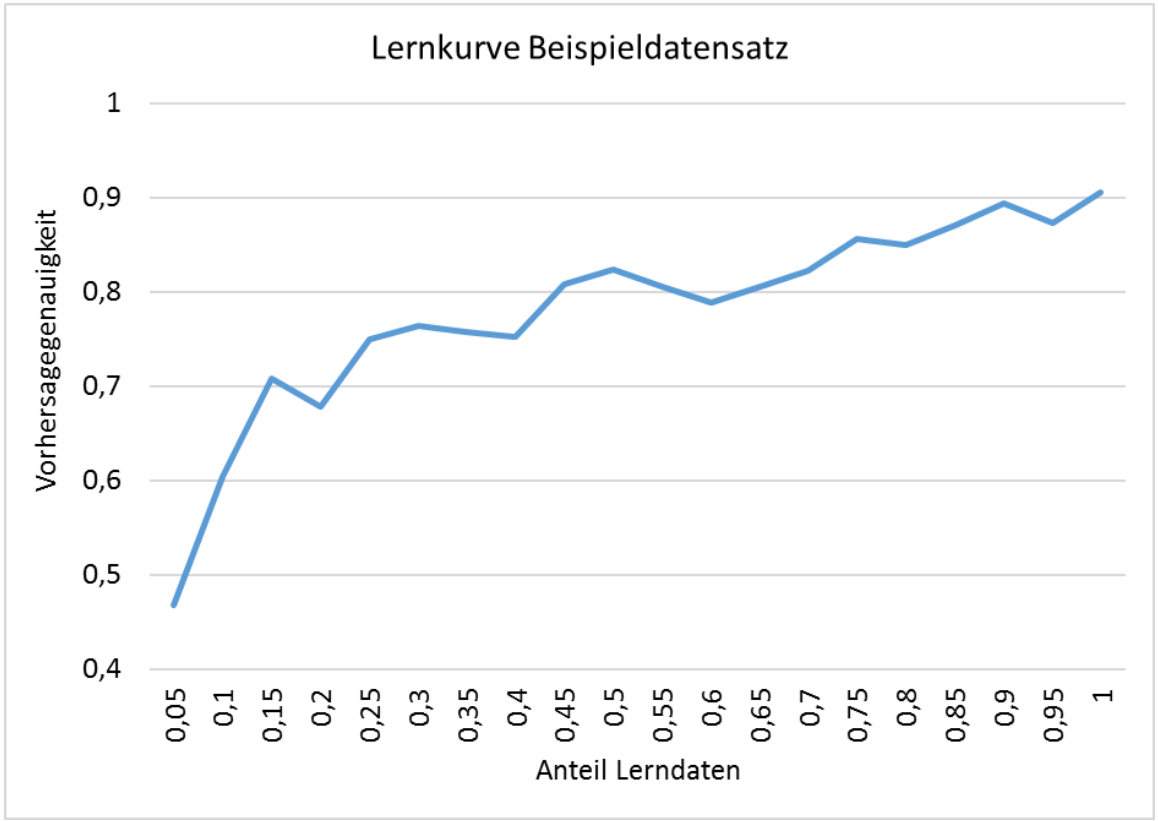
Backpropagation Networks
Generalisierung

- Wie groß muss das Netzwerk sein?
  - Um unbekannte Muster richtig zu klassifizieren?
  - Test der Vorhersagegenauigkeit auf den Testdaten



Backpropagation Networks
Lernkurve

- Wie viele Beispiele benötigt das Netzwerk, um zu generalisieren?
  - Je mehr desto besser



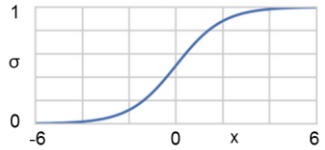
Backpropagation Networks
Zusammenfassung

- Lernregel

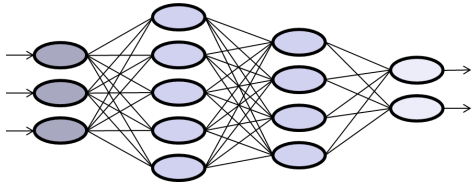
$$w_{i,j} = w_{i,j} + \alpha \cdot o_i \cdot \Delta_j \text{ mit } \Delta_j = \sigma'(in_j) \cdot \sum_k w_{j,k} \Delta_k$$

- Aktivierungsfunktion

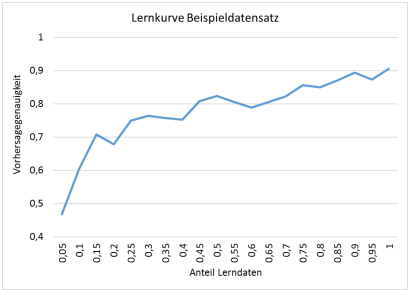
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



- Netzwerkgröße



- Lernkurve

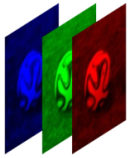


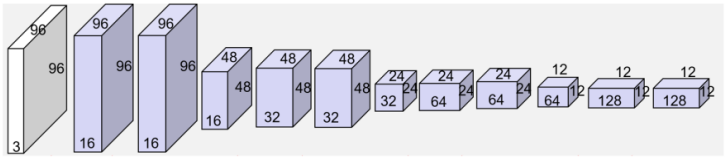
Menschen Lernen Maschinelles Lernen

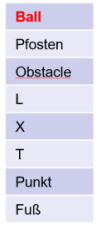
Deep Learning

Convolutional Neural Networks

Prof. Dr. Klaus Dorer







Hochschule Offenburg
offenburg.university

GEFÖRDERT VOM
Bundesministerium
für Bildung
und Forschung

Übersicht

- Neuronale Netzwerke
  - Einführung
  - Modell einer Nervenzelle
  - Perceptron
  - Backpropagation Networks
  - Convolutional Neural Networks
    - Funktionsweise
    - Anwendungsbeispiele
    - Deep Learning Frameworks

Convolutional Neural Networks

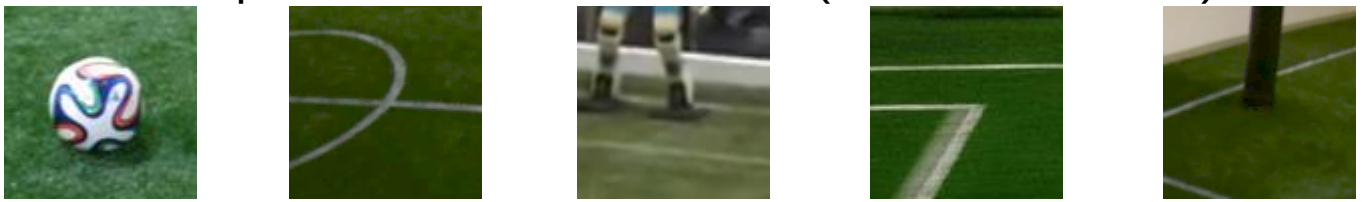
- Sweaty will Fußball spielen
- Dazu muss er Objekte auf Bildern erkennen
- Es ist egal, wo im Bild ein Gegenstand ist

- Es soll möglichst egal sein, wie hell es ist

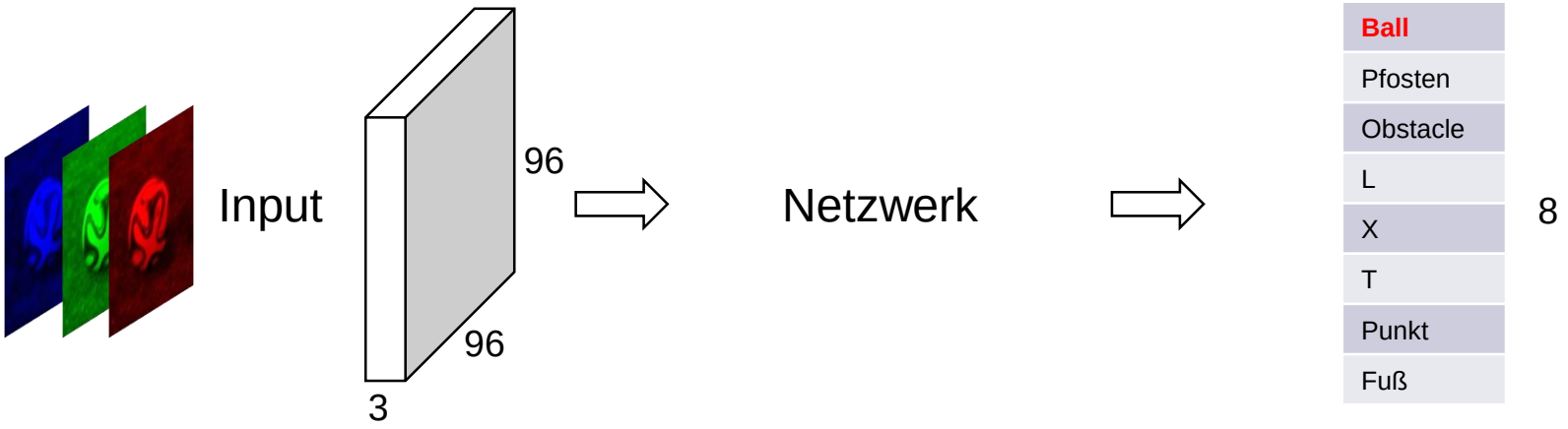
- Es soll egal sein, wie der Ball aussieht

Convolutional Neural Networks

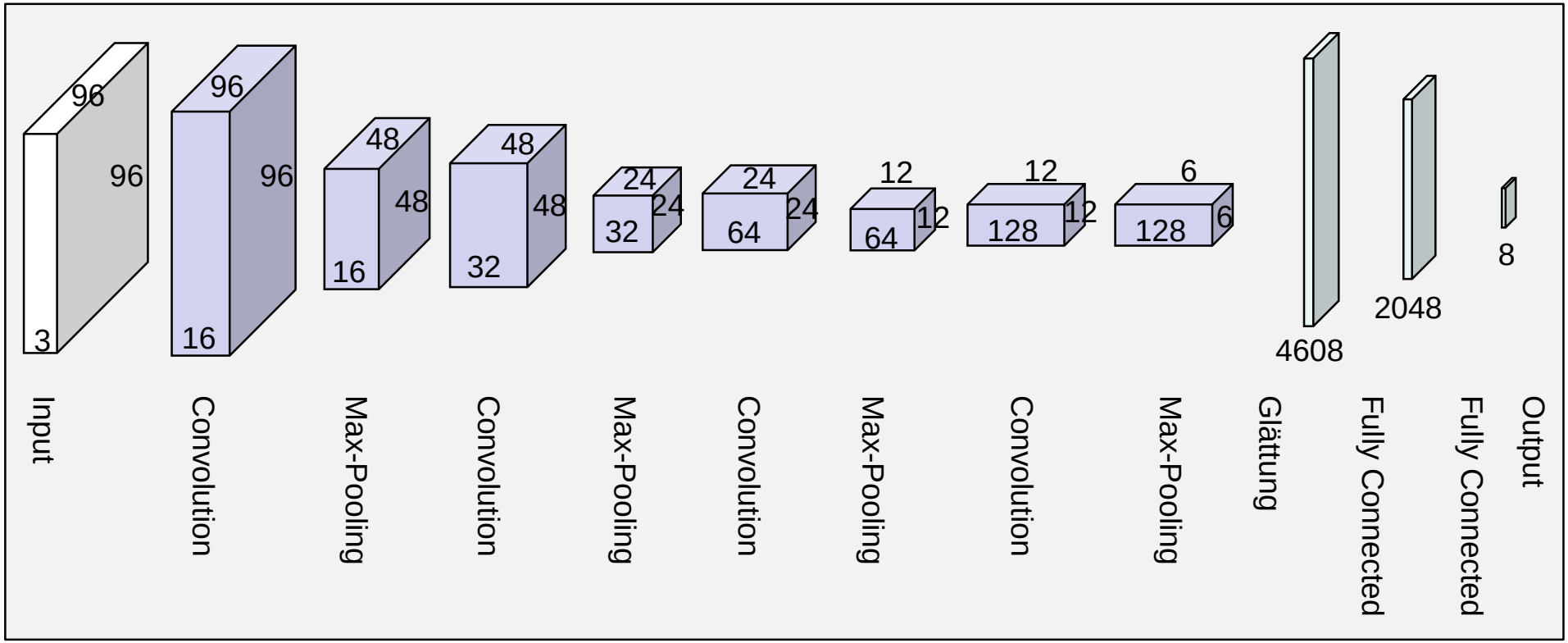
- Input ist 3D
  - Unser Beispiel: 96x96 Pixel, 3 Kanäle (Red, Green, Blue)



- Output ist Feature-Vektor
  - Ball X-Linie Roboter L-Linie Obstacle

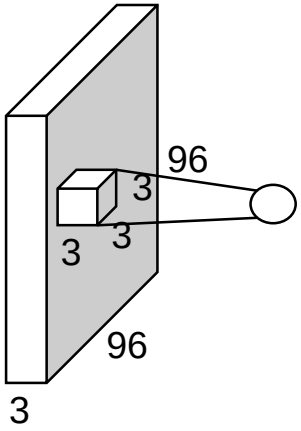


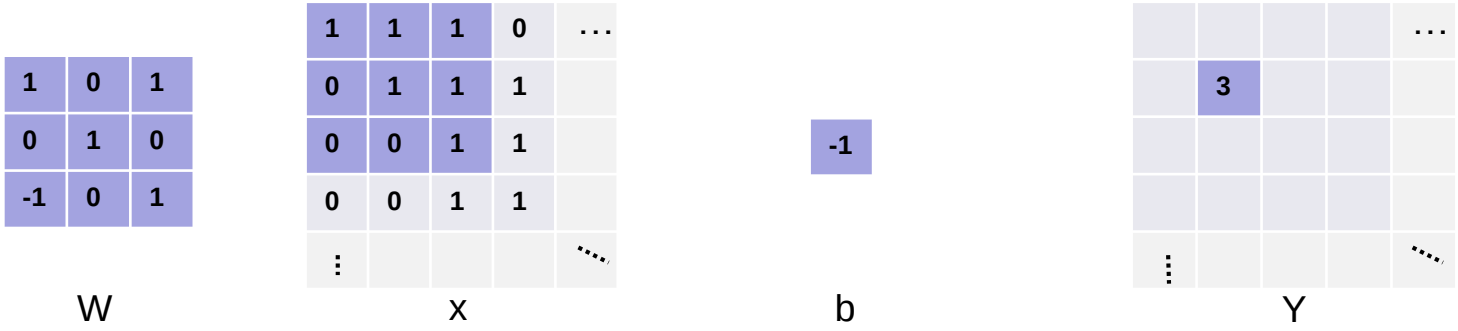
Convolutional Neural Networks
Beispielarchitektur



Convolutional Neural Networks
Convolution Layer

- Filter wandert über Input
- Berechnet Feature Input für Neuron in Activation Map
- Aktivierungsfunktion: ReLU
  - Y = ReLU(Wx+b)





- Wird mit n Filtern wiederholt: n Activation Maps
- Padding: Wie wird mit dem Rand umgegangen
- Strides: Schrittweite, > 1 bedeutet Activation Map wird kleiner

Convolutional Neural Networks
Pooling Layer

- Reduziert die Größe der Activation Map
- Poolingfunktion
  - Meist wird Max-Pooling verwendet
  - Alternative Avg-Pooling
- Reduzieren die Informationsmenge
- Beispiel
  - 2x2 Filter mit 2,2 strides

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

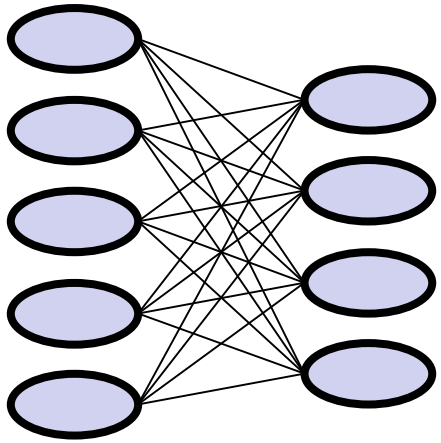
Vorher

6	7
2	3

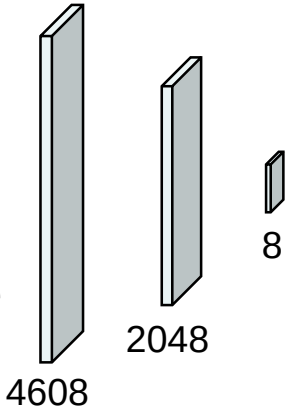
Nachher
Max- Pooling

Convolutional Neural Networks
Fully Connected Layer

- Bereits bekannt: Backpropagation Schichten
- Manchmal genügt es, für ein neues Bilderkennungs-problem nur diese Schichten neu zu lernen
- Hier stecken die meisten lernbaren Gewichte



- Beispiel Ballerkennung
  - 4608 * 2048 = 9.437.184 Gewichte
  - 2048 * 8 = 16.384 Gewichte
- Zum Vergleich
  - In der ersten Convolution Schicht sind 27*16 Gewichte



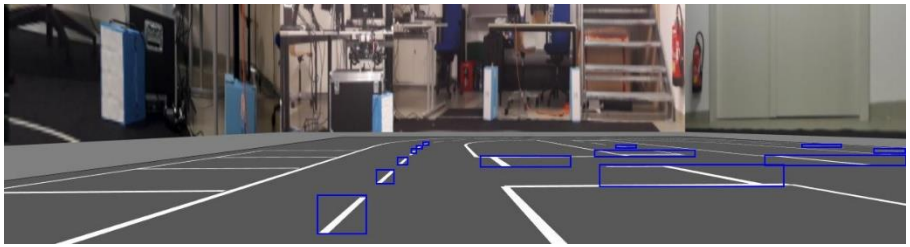
Convolutional Neural Networks
Lernen

- Bilder taggen
  - Für jedes Bild wird die richtige Klassifikation gespeichert
  - Gegebenenfalls eine bounding box
- Bilder lernen
  - Trainingsdatensatz ans Netzwerk anlegen
  - Netzwerk berechnet Output und Fehler
  - Netzwerk ermittelt mit Gradientenabstieg Gewichtsanpassungen, um den Fehler zu reduzieren
- Qualität auf Testdatensatz überprüfen
- Anwenden
  - Gespeichertes Netzwerk auf Live Bildern betreiben
  - Schneller Recall

Convolutional Neural Networks
Overfitting

- Viele Bilder zeigen
- Wenn nicht genügend vorhanden?
  - Vorhandene Bilder mehrfach verwenden

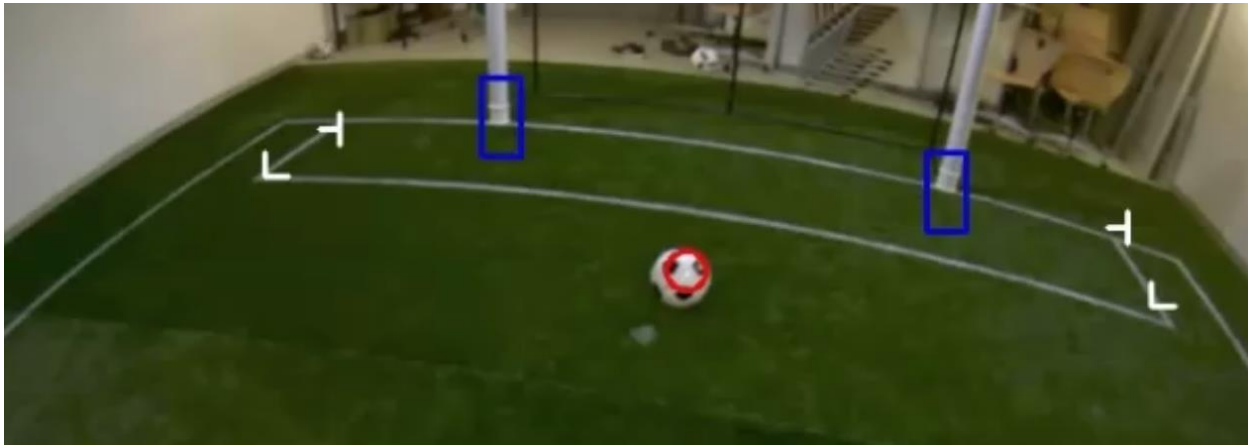
  - Synthetische Bilder generieren



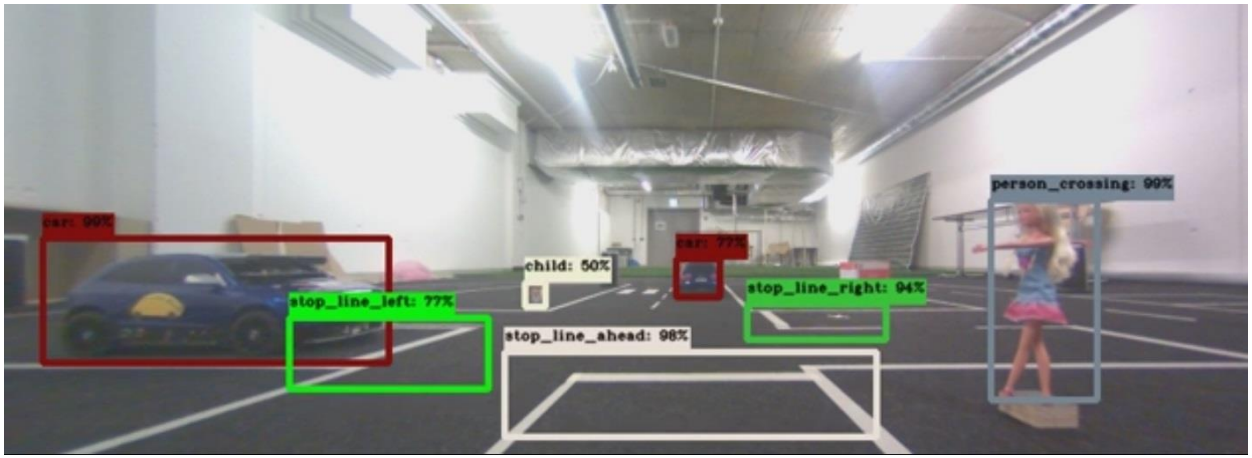
- Dropout
  - Beim Lernen einen bestimmten Prozentsatz zufällig ausgewählter Verbindungen nicht berücksichtigen
  - Vermeidet einzelne ‚wichtige' Verbindungen

Deep Learning
Beispiel Hochschule

- RoboCup

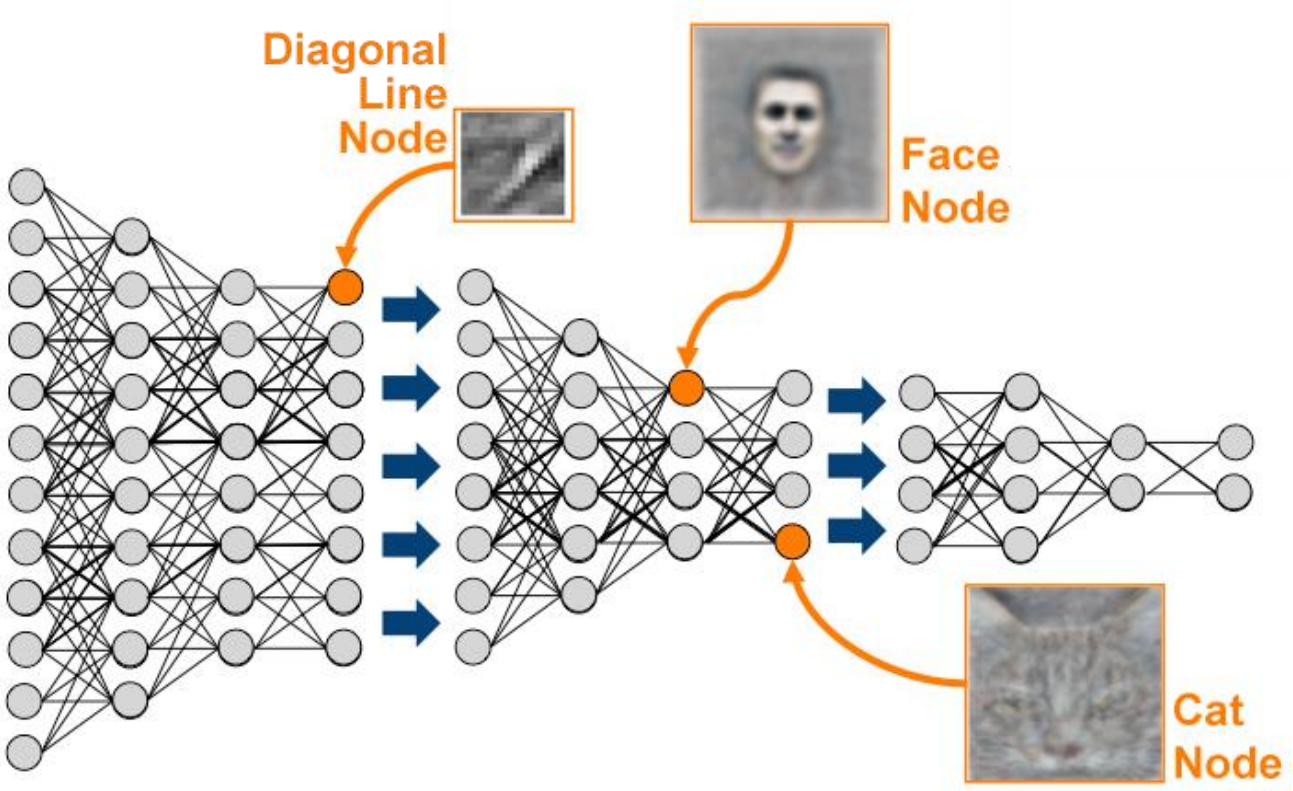


- AudiCup



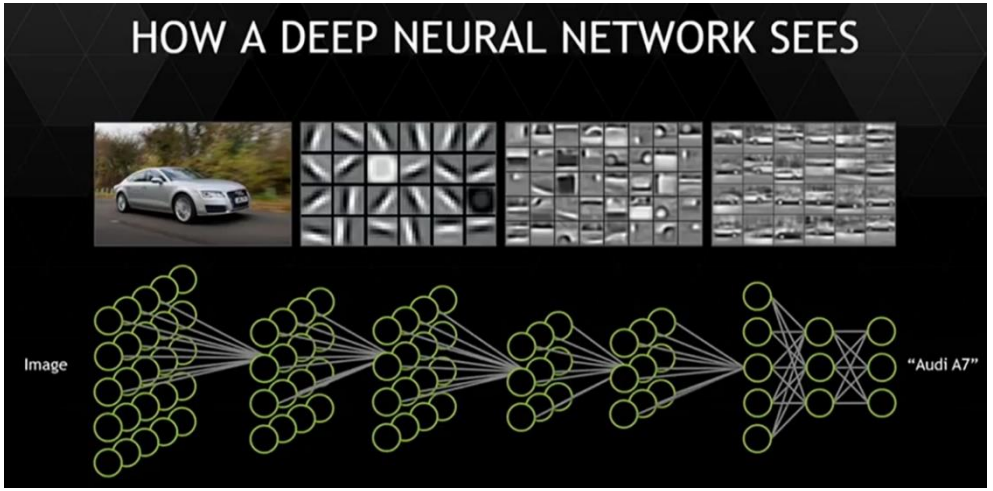
Deep Learning
Beispiel Google

- Input
  - 10 Mio Bilder (200 x 200 pixel)
- Learning
  - 1 Mio Gewichte
  - 16.000 cores
  - 3 Tage



Deep Learning
Beispiel NVIDIA

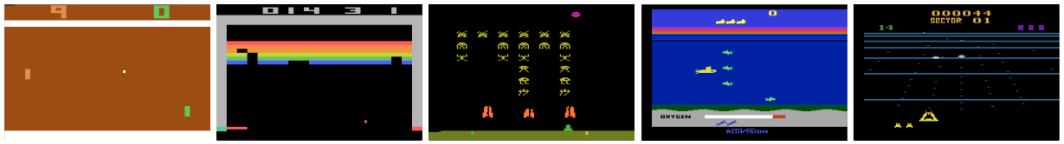
- Beispiel NVIDIA
  - erste Schicht erkennt Linien und Kreise
  - Fahrzeugteile
  - Fahrzeuge
  - Fahrzeugtyp
- Aufwand Lernen
  - Tage
    (auf GPU Cluster)
- Leistung Recall
  - 2 Megapixel
  - 30 fps
  - 75 Objects



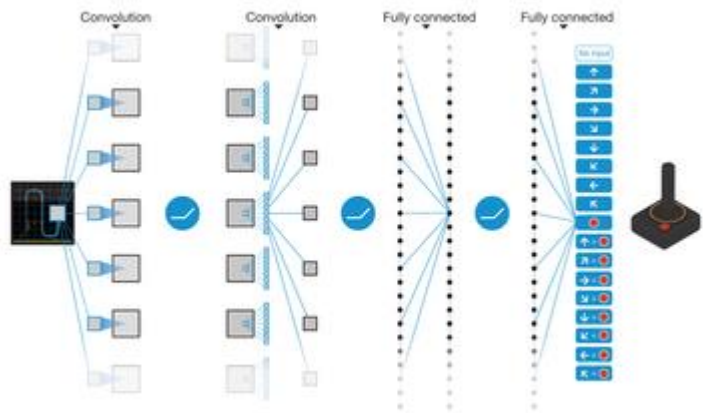
https://www.youtube.com/watch?v=zsVsUvx8ieo

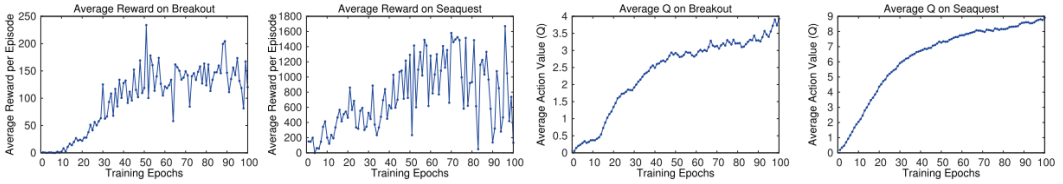
Deep Learning
Beispiel DeepMind

- Deep Reinforcement Learning von Computer Spielen



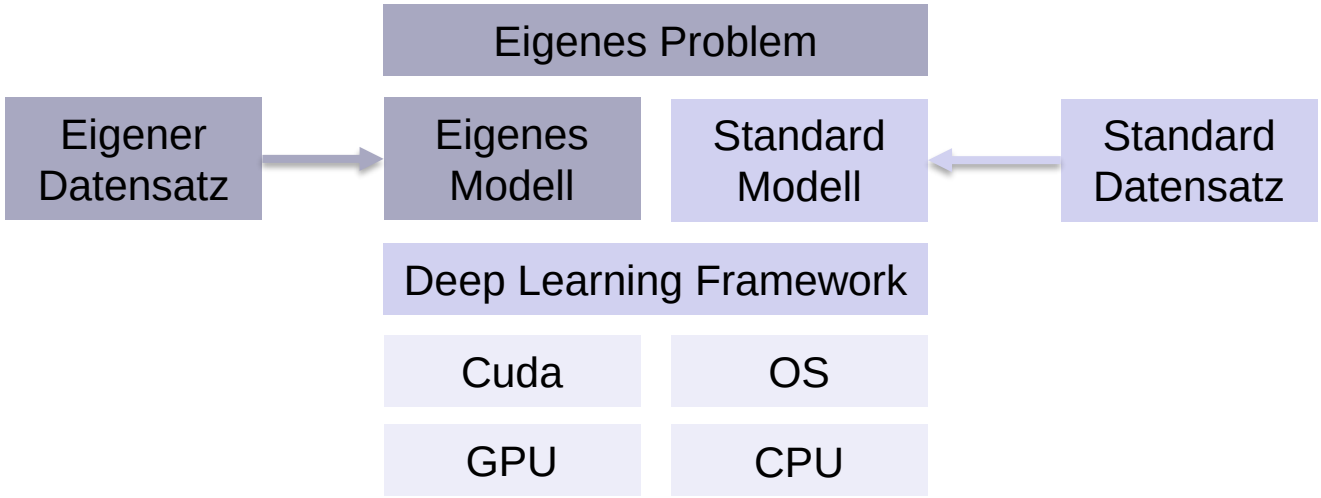
- Input
  - 84*84*4 downsampled live Video Input
- Netzwerk
  - 4 Layers, 2 Convolutional (8x8, 4x4), 2 fully connected
- Ergebnis
  - 4 der 7 Spiele besser als Human Expert





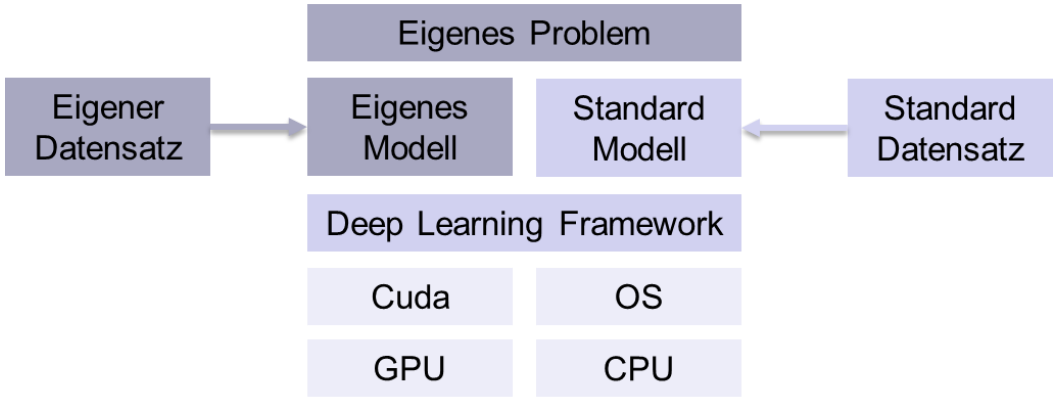
https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf und http://www.nature.com/nature/journal/v518/n7540/full/nature14236.h

Deep Learning
Frameworks



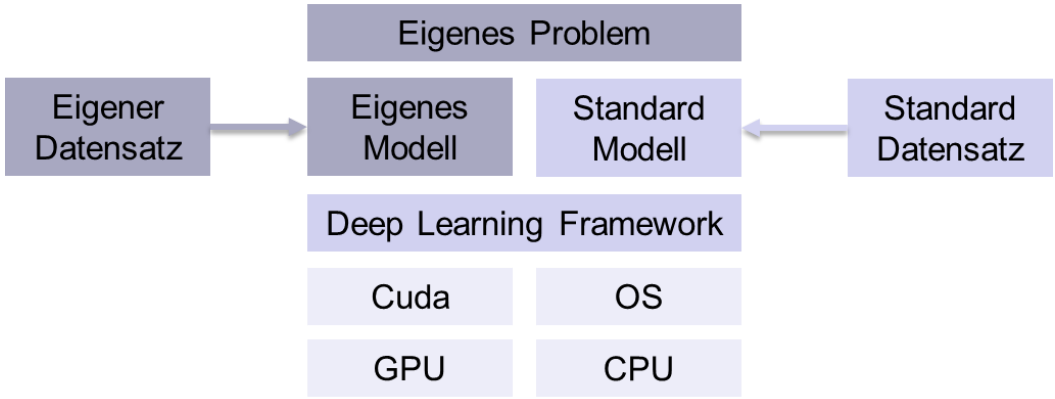
Deep Learning
Bekannte Frameworks

- 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/



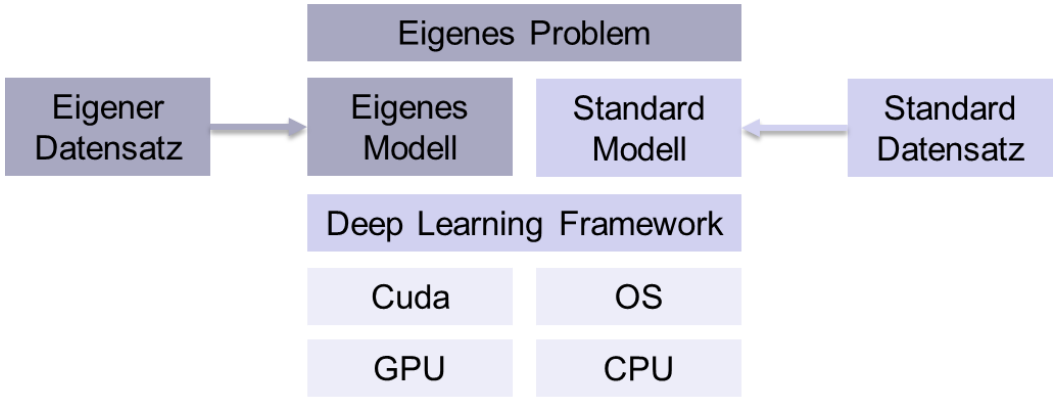
Deep Learning
Standard Datensätze

- ImageNet
  - 14 Mio getaggte Bilder
  - 21.000 Kategorien
  - http://www.image-net.org
- MNIST
  - 70.000 handgeschriebene Ziffern
  - http://yann.lecun.com/exdb/mnist/
- COCO (common objects in context)
  - 200.000 getaggte und segmentierte Bilder
  - http://cocodataset.org/#home
- Musik, Gesichter, Sprache, Texte, …



Deep Learning
Standard Modelle

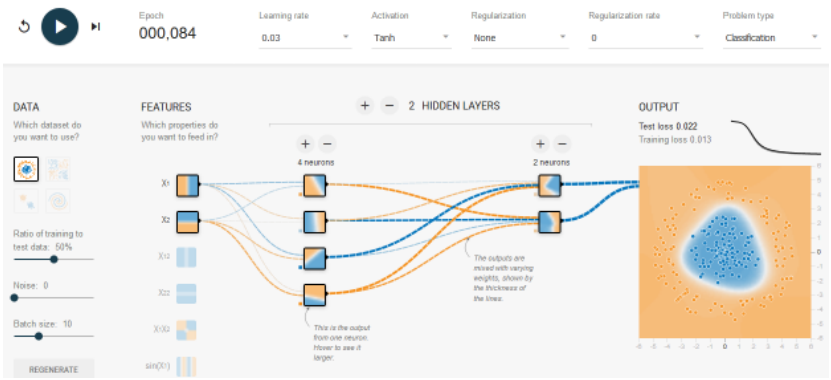
- LeNet-5 (ab 1990)
  - 5 Schichten (4,1)
  - MNIST
- AlexNet (2012)
  - 8 Schichten (5,3)
  - ImageNet (16.4% Fehler)
- GoogLeNet (2014)
  - 22 Schichten
  - ImageNet (6.7%)
- ResNet-152 (2015)
  - 152 Schichten
  - ImageNet (3.6%), COCO



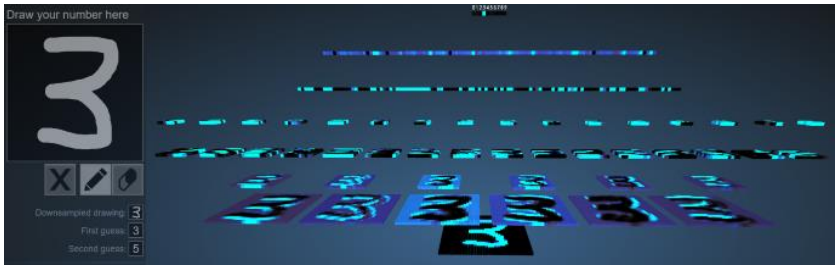
- VGGNet
- Mobilenet
- Inception
- …

Deep Learning
Visualisierungen

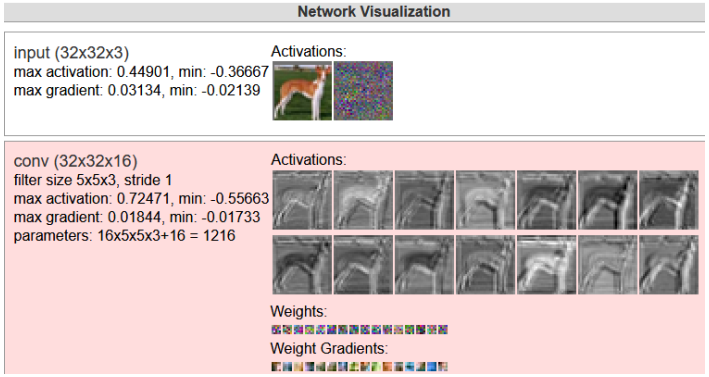
- Google Playground
  - http://playground.tensorflow.org



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



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



Deep Learning
Zusammenfassung

- Deep Convolutional Neural Networks
  - Convolution layers
  - Max layers
  - Dense layers
- Zahlreiche Anwendungsfelder
  - Bilderkennung, Spracherkennung, Texterkennung, …
- Viele vortrainierte Netze verfügbar