

Neuronale Netze

Prof. Dr. Rudolf Kruse
Christoph Doell, M.Sc.

Computational Intelligence
Institut für Wissens- und Sprachverarbeitung
Fakultät für Informatik
kruse@iws.cs.uni-magdeburg.de



Lernen Tiefer Neuronaler Netze (Deep Learning)

Wiederholung

- Lernen Rekurrenter Netze durch *ausfalten*
- Problem des verschwindenden Gradienten

Varianten des Neurons

Autoencoder

Hybrider Lernalgorithmus

Faltung

Pooling

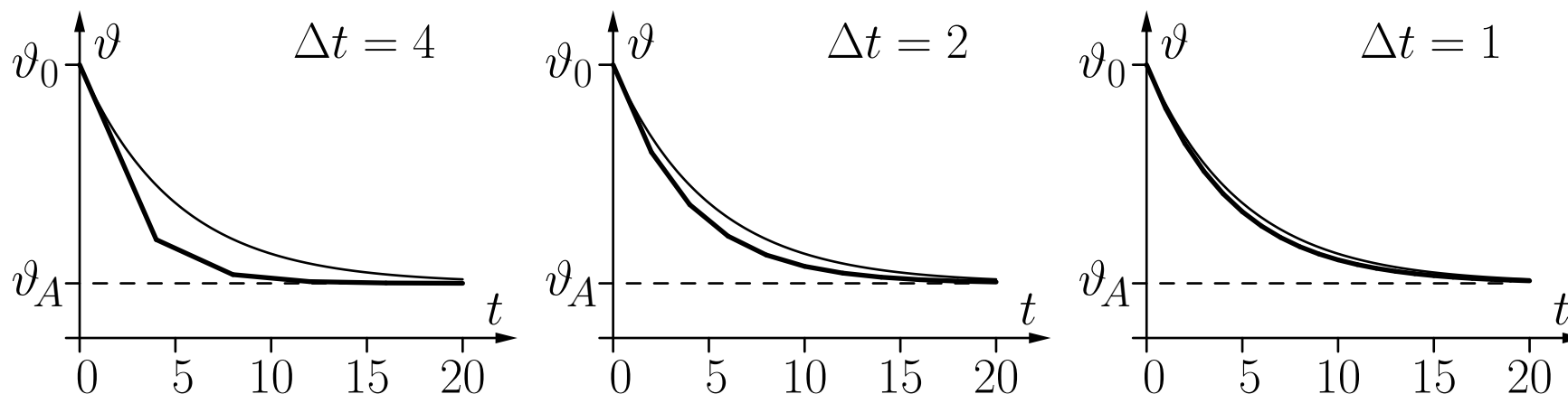
Faltende Neuronale Netze

Anwendungen

Wiederholung: Lernen Rekurrenter Netze

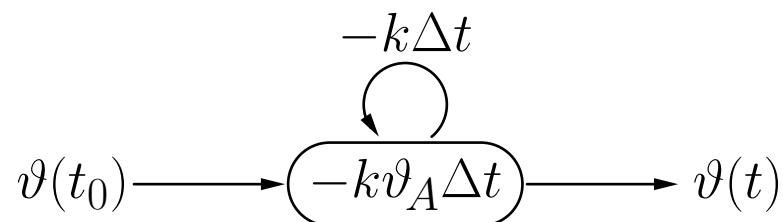
Beispiel: **Newton'sches Abkühlungsgesetz**

Euler–Cauchy-Polygonzüge für verschiedene Schrittweiten:



Die dünne Kurve ist die genaue analytische Lösung.

Rekurrentes neuronales Netz:



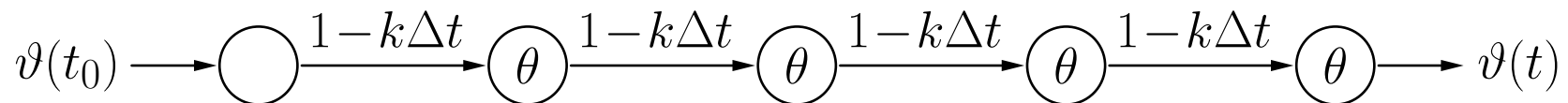
Wiederholung: Fehler-Rückpropagation über die Zeit

Annahme: Wir haben Messwerte der Abkühlung (oder Erwärmung) eines Körpers zu verschiedenen Zeitpunkten. Außerdem sei die Umgebungstemperatur ϑ_A bekannt.

Ziel: Bestimmung des Werts der Abkühlungskonstanten k des Körpers.

Initialisierung wie bei einem MLP: Biaswert und Gewicht der Rückkopplung zufällig wählen.

Die Zeit zwischen zwei aufeinanderfolgenden Messwerten wird in Intervalle unterteilt. Damit wird die Rückkopplung des Netzes *ausgefaltet*. Liegen z.B. zwischen einem Messwert und dem folgenden vier Intervalle ($t_{j+1} = t_j + 4\Delta t$), dann erhalten wir

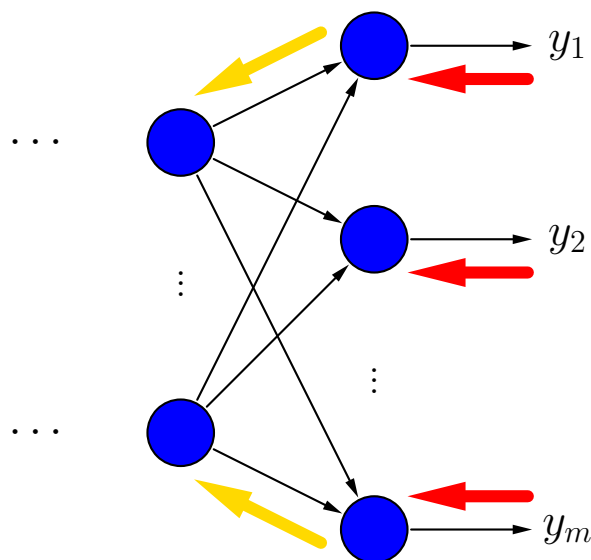
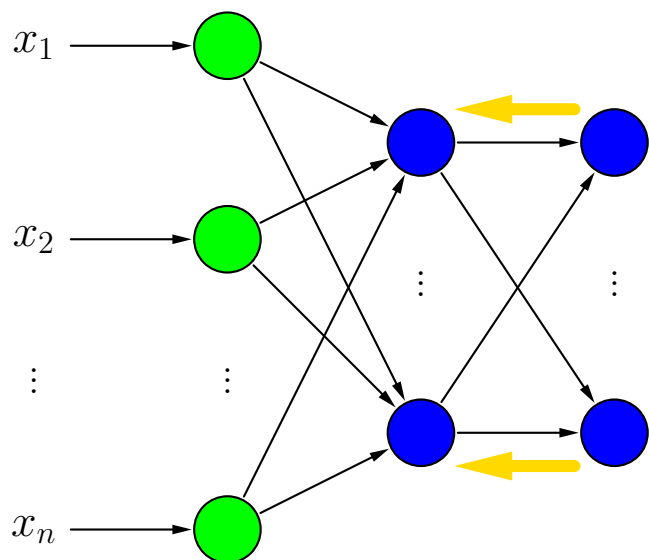


Wiederholung: Fehlerrückpropagation

$$\forall u \in U_{\text{in}} : \text{out}_u^{(l)} = \text{ex}_u^{(l)}$$

Vorwärts-
propagation:

$$\forall u \in U_{\text{hidden}} \cup U_{\text{out}} : \text{out}_u^{(l)} = \left(1 + \exp \left(- \sum_{p \in \text{pred}(u)} w_{up} \text{out}_p^{(l)} \right) \right)^{-1}$$



logistische
Aktivierungs-
funktion
impliziter
Biaswert

Fehlerfaktor:

Rückwärts-
propagation:

$$\forall u \in U_{\text{hidden}} : \delta_u^{(l)} = \left(\sum_{s \in \text{succ}(u)} \delta_s^{(l)} w_{su} \right) \lambda_u^{(l)}$$

$$\forall u \in U_{\text{out}} : \delta_u^{(l)} = \left(o_u^{(l)} - \text{out}_u^{(l)} \right) \lambda_u^{(l)}$$

Aktivierungs-
ableitung:

$$\lambda_u^{(l)} = \text{out}_u^{(l)} \left(1 - \text{out}_u^{(l)} \right)$$

Gewichts-
änderung:

$$\Delta w_{up}^{(l)} = \eta \delta_u^{(l)} \text{out}_p^{(l)}$$

Problem des verschwindenden Gradienten

Probleme:

Gewichtsänderung nimmt in vorderen Schichten exponentiell ab

Lernen dauert zu lang

Zuwenige (gelabelte) Lernbeispiele vorhanden

Konvergenz zu lokalen Minima

Lösungsansätze:

Initialisiere Gewichte nicht zufällig, sondern abhängig vom Datensatz

Verwende schnellere Computer (Lastverteilung auf GPUs)

Sammele mehr gelabelte Lernbeispiele

Kann nicht verhindert werden

Rectified Linear Unit (ReLU)

Wähle statt Neuron
Rectified Linear Unit

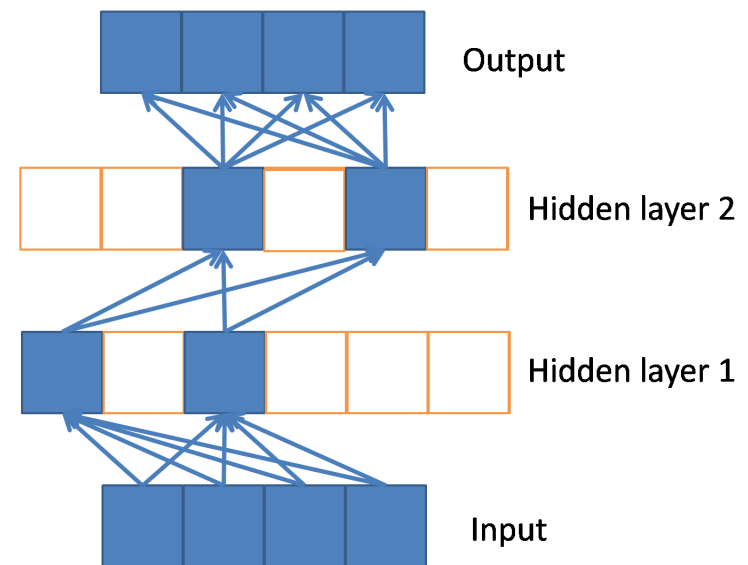
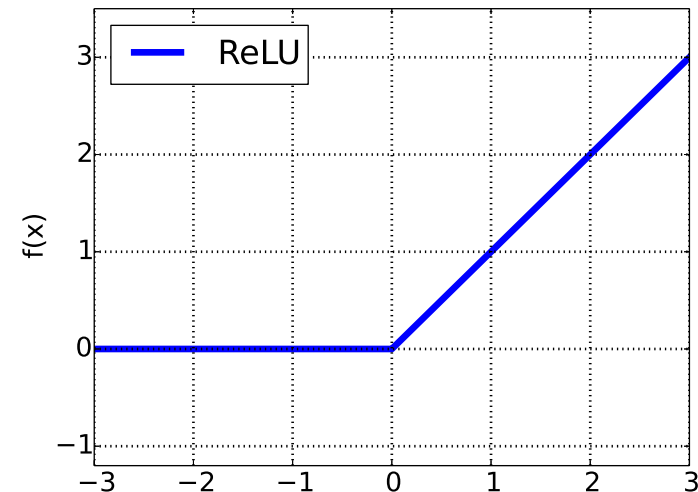
ReLU: $f(x) = \max(0, x)$

Vorteile:

- sehr einfache Berechnung
- Ableitung ist leicht zu bilden
- 0-Werte vereinfachen Lernen

Nachteile:

- kein Lernen links der 0
- mathematisch eher unschön
- Nicht-differenzierbarer
„Knick“ bei 0



[ReLU nach Glorot et. al 2011]

ReLU: Berechnungsvarianten

Softplus:

$$f(x) = \ln(1 + e^x)$$

- „Knick“ wurde beseitigt
- Einige Vorteile auch

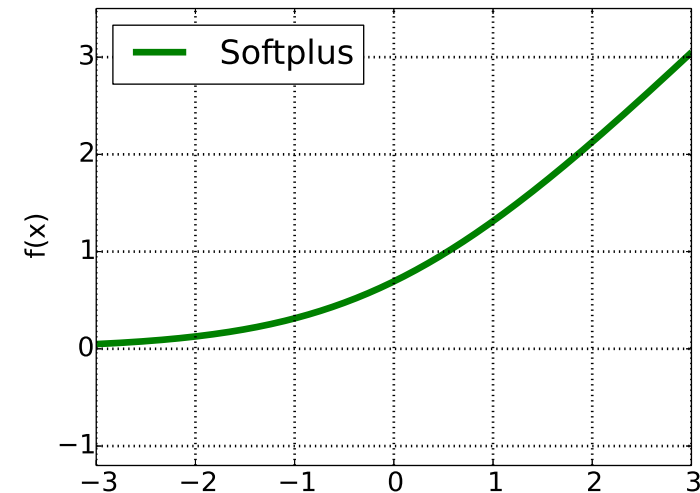
Noisy ReLU:

$$f(x) = \max(0, x + \mathcal{N}(0, \sigma(x)))$$

- Addiert Gaussches Rauschen

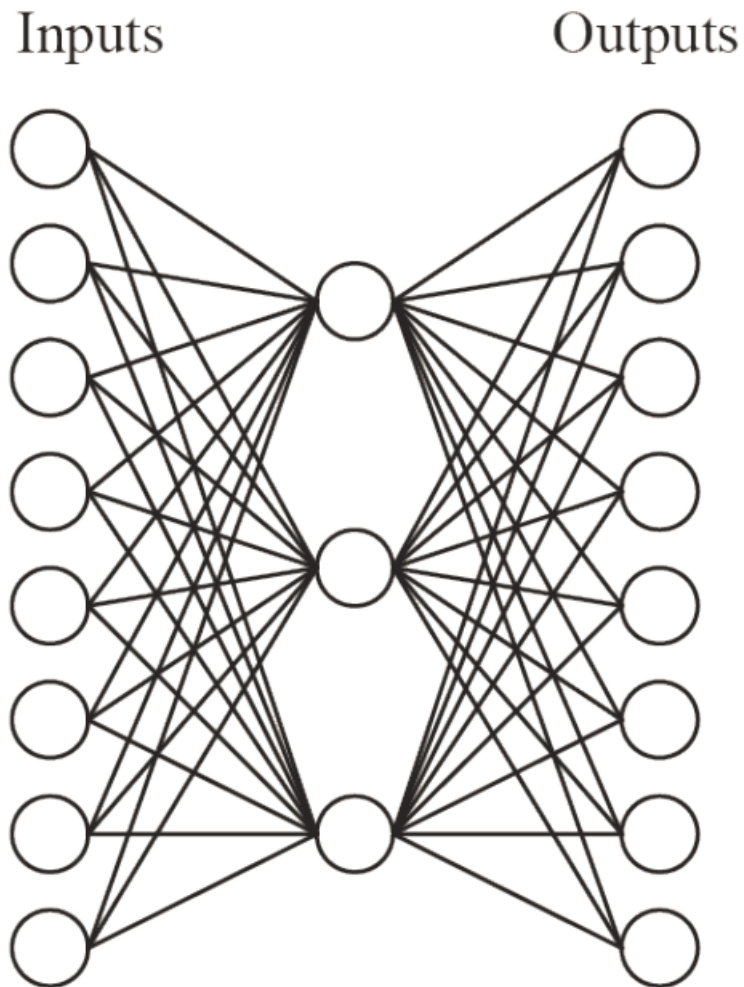
Leaky ReLU

$$f(x) = \begin{cases} x, & \text{falls } x > 0, \\ 0.01x, & \text{sonst.} \end{cases}$$



[Softplus nach Glorot et. al 2011]

Autoencoder



Erstellt eine Kodierung der Daten

Lernt Gewichte mit Rückpropagation

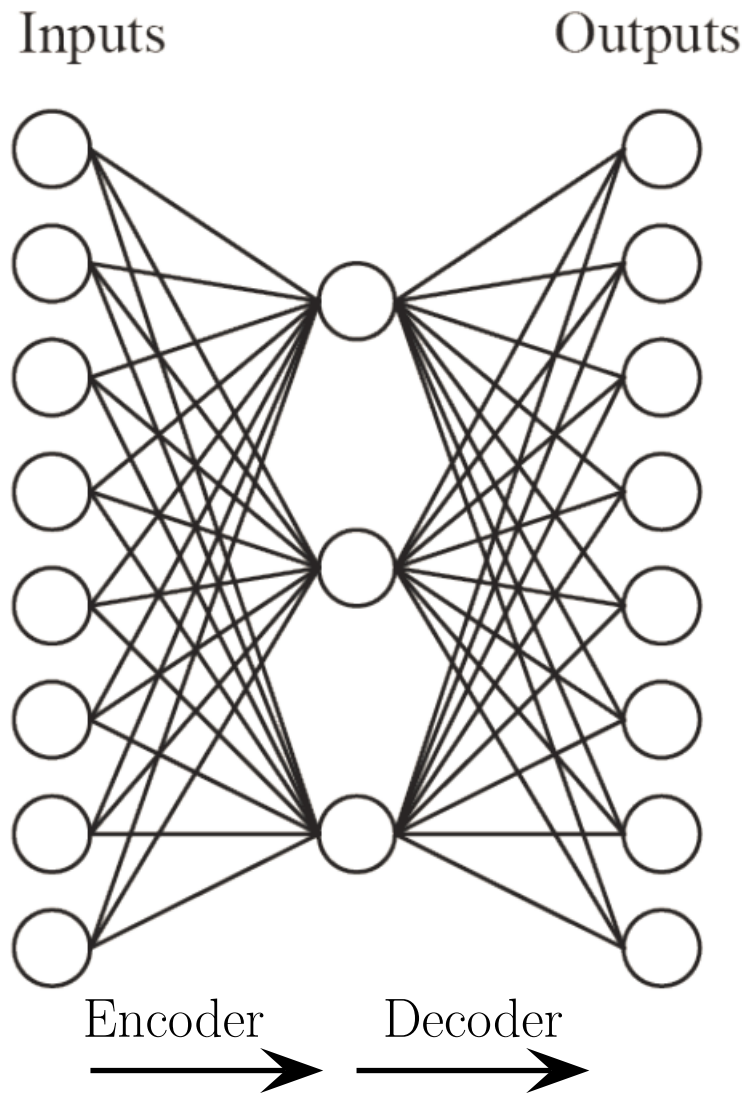
Durch **unüberwachtes** Lernen

Fehler ist $|out - in|^2$

Inputs	Versteckte Gewichte	Outputs
10000000 →	<h1>?</h1>	→ 10000000
01000000 →		→ 01000000
00100000 →		→ 00100000
00010000 →		→ 00010000
00001000 →		→ 00001000
00000100 →		→ 00000100
00000010 →		→ 00000010
00000001 →		→ 00000001

[Grafiken nach T. Mitchell, Machine Learning, McGraw Hill, 1997]

Autoencoder



Nutze für Dekodierung die transponierte Gewichtsmatrix der Encodierung

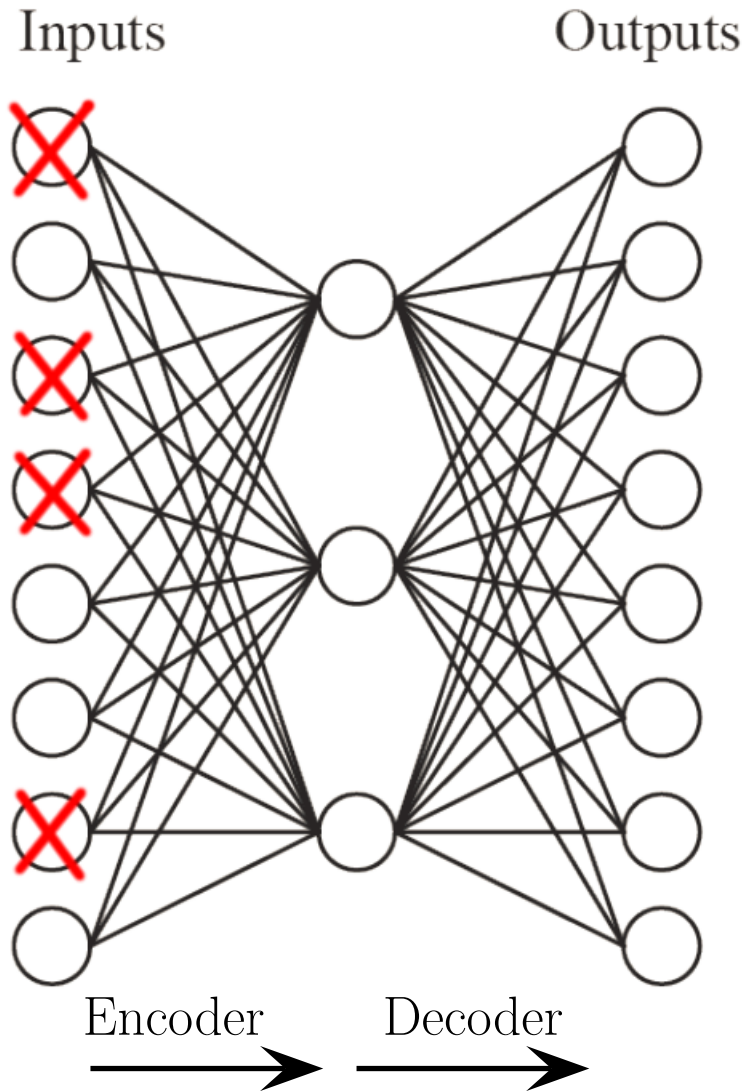
Ergebnis nach 5000 Lerniterationen:

Binäre Kodierung annähernd erreicht

Inputs		Versteckte Gewichte				Outputs
10000000	→	.89	.04	.08	→	10000000
01000000	→	.15	.99	.99	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.01	.11	.88	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

[Grafiken nach T. Mitchell, Machine Learning, McGraw Hill, 1997]

Rauschreduzierender (Denoising) Autoencoder



Gewünschte Eigenschaft:

Robustheit bei Ausfall von Neuronen

Ansatz:

- Nutze zum Lernen nur 50% der Neuronen
- Wähle diese zufällig beim Lernen

Verwende zum Auswerten alle Neuronen

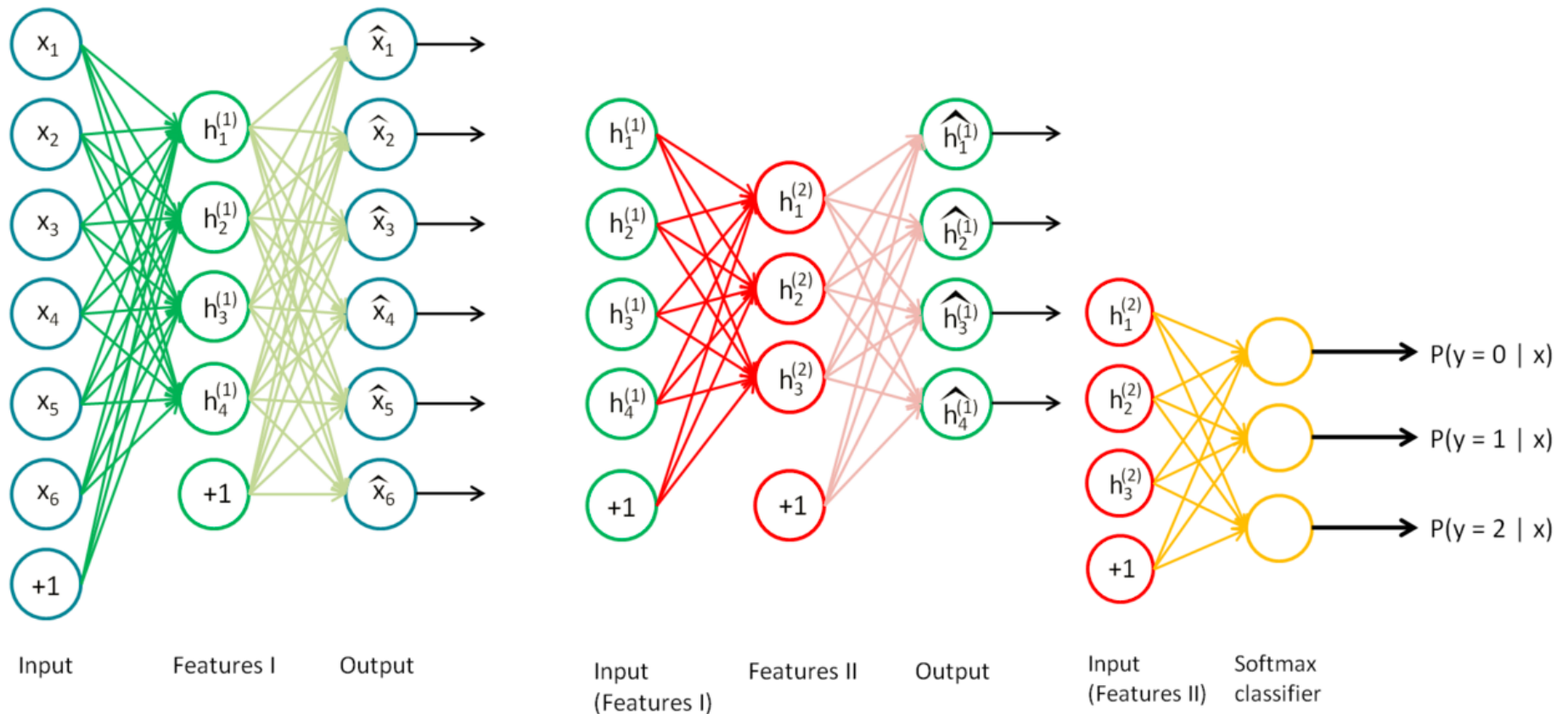
Halbiere alle Gewichte

Ergebnis:

- Robustere Repräsentation
- Verbesserte Generalisierung

Stapeln von Autoencodern

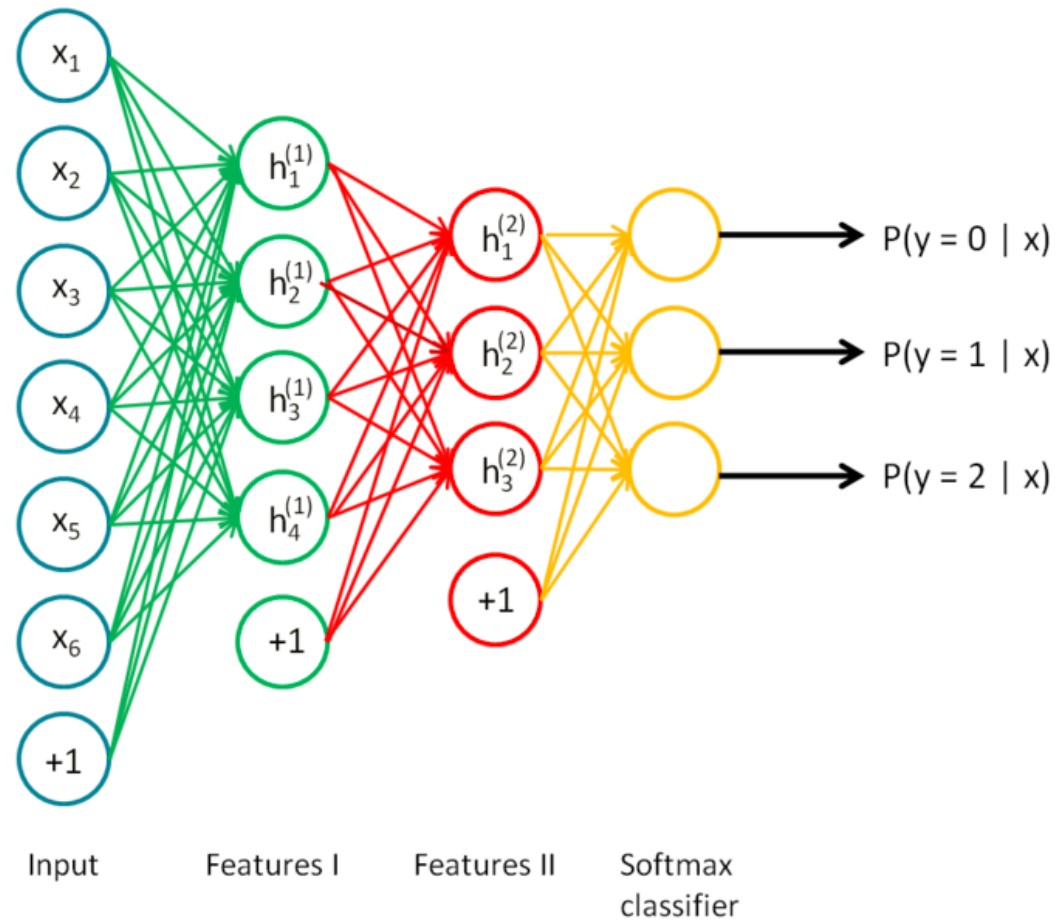
Stapelte Autoencoder, um die besten Features zu erhalten



[http://ufdl.stanford.edu/wiki/index.php/Stacked_Autoencoders]

Stapeln von Autoencodern

Nutze die (vor)gelernten Features zur Klassifikation



[http://ufdl.stanford.edu/wiki/index.php/Stacked_Autoencoders]

Hybrider Deep Learning Algorithmus

1. Definiere für die Lernaufgabe geeignete Netzstruktur
2. Erstelle entsprechend der Struktur Autoencoder und lasse sie mit Rückpropagation einzeln lernen
3. Verwende nur die Encoder, ihre Gewichte und eine weitere vollständig vernetzte, zufällig initialisierte Schicht zur Klassifikation
4. Lasse das so vortrainierte Netz mit Rückpropagation lernen

Faltung (Convolution)

1	1	1	0	0
0	1	1	1	0
0	0	1 _{x1}	1 _{x0}	1 _{x1}
0	0	1 _{x0}	1 _{x1}	0 _{x0}
0	1	1 _{x1}	0 _{x0}	0 _{x1}

Image

4	3	4
2	4	3
2	3	4

Convolved
Feature

Featuretransformation

Schiebe einen „Filter“ über die Features und betrachte die „gefilterten“ Features

Multipliziere Originalfeature mit Filter und Summiere

Originalraum: 5x5

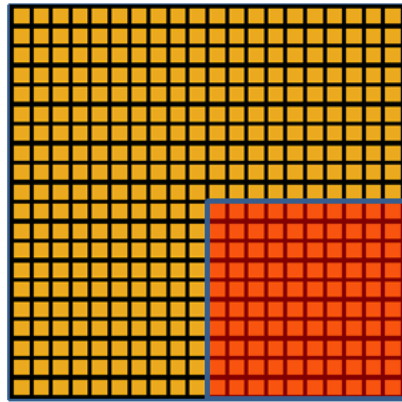
Filtergröße: 3x3

Neue Featuregröße: 3x3

Featureraum kann sich vergrößern

[http://ufdl.stanford.edu/wiki/index.php/Feature_extraction_using_convolution]

Pooling



Convolved
feature

1	7
5	9

Pooled
feature

Featuretransformation

Schiebe einen „Filter“ über die Features und betrachte die „gefilterten“ Features

Betrachte den Bereich entsprechend der Filtergröße

Max Pooling: Nimm maximalen Wert

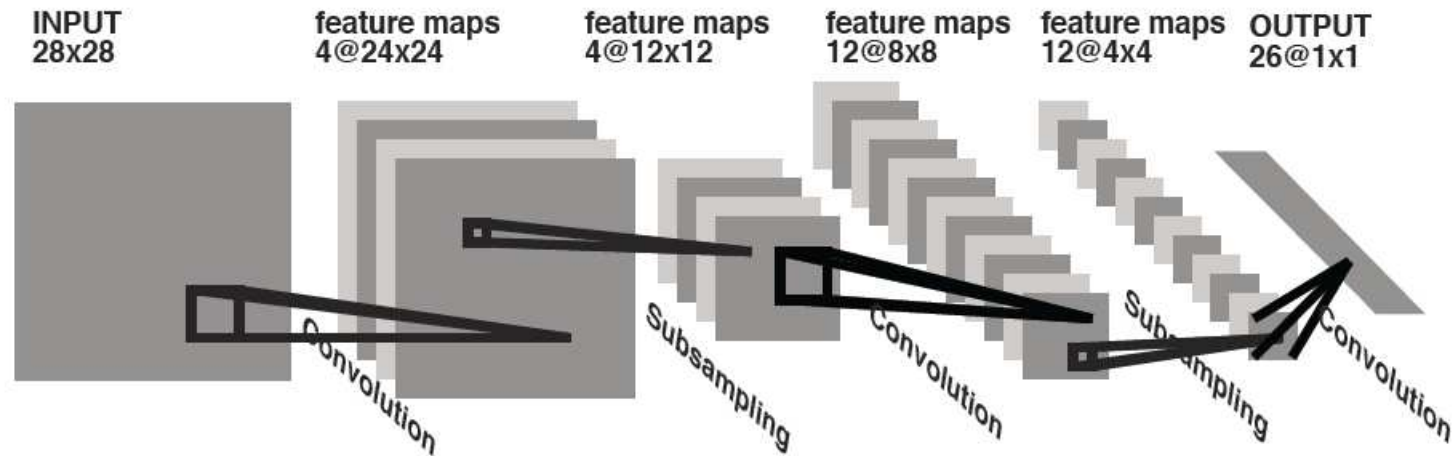
Mean Pooling: Nimm Mittelwert

Featureraum wird kleiner

[http://deeplearning.stanford.edu/wiki/index.php/File:Pooling_schematic.gif]

Faltendende (Convolutional) Neuronale Netze

[Y. Bengio and Y. Lecun, 1995]

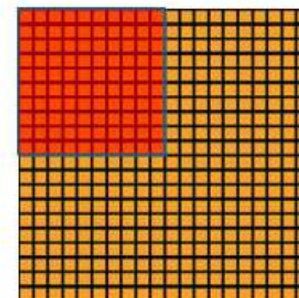


1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

2D input

4		

convolved feature



convolved feature

1	

pooled feature

http://ufdl.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Anwendung: IQ-Test

Lösen von verbalen Verständnisfragen in IQ-Tests [Wang et al. 2015]

Verbale IQ-Tests beinhalten hier 5 Arten von Fragen:

Analogie 1, Analogie 2, Klassifikation, Synonym, Antonym

Beispielfrage(Analogie 1): Isotherm verhält sich zu Temperatur wie isobar zu?

(i) Atmosphäre, (ii) Wind, (iii) Druck, (iv) Ausdehnung, (v) Strömung

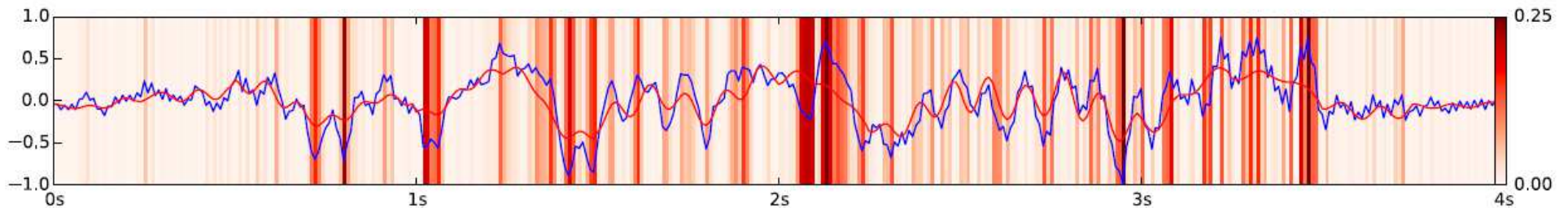
Ansatz:

- Klassifiziere den Fragentyp mit Hilfe einer SVM
- Benutze für jeden Fragentyp einen dafür erstelltes Tiefes Neuronales Netz
- Nutze zum Lernen von zusammengehörenden Wörter eine große Datenbasis (Wiki2014)

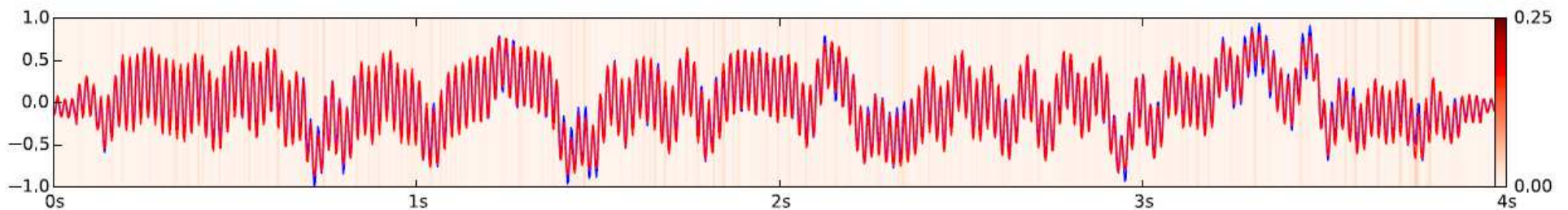
Ergebnis: DeepLearning schneidet etwas besser ab, als Bachelor-Absolventen

Rhythmus-Rekonstruktion durch EEG-Analyse

100-50-25-10 @ 100Hz



100-50-25-10 @ 400Hz



[Quelle: Sebastian Stober, DKE-Kolloquium 03.06.2014]

German Traffic Sign Recognition Benchmark (GTSRB)



Wurde analysiert bei der International Joint Conference on Neural Networks (IJCNN) 2011

Problemstellung:

Ein Bild, mehrere Klassen Klassifikationsproblem

Mehr als 40 Klassen

Mehr als 50.000 Bilder

Ergebnis:

Erste übermenschliche visuelle Mustererkennung

Fehlerraten:

Mensch: 1.16%, NN:0.56%

Stallkamp et al. 2012

Verwendetes Netz:

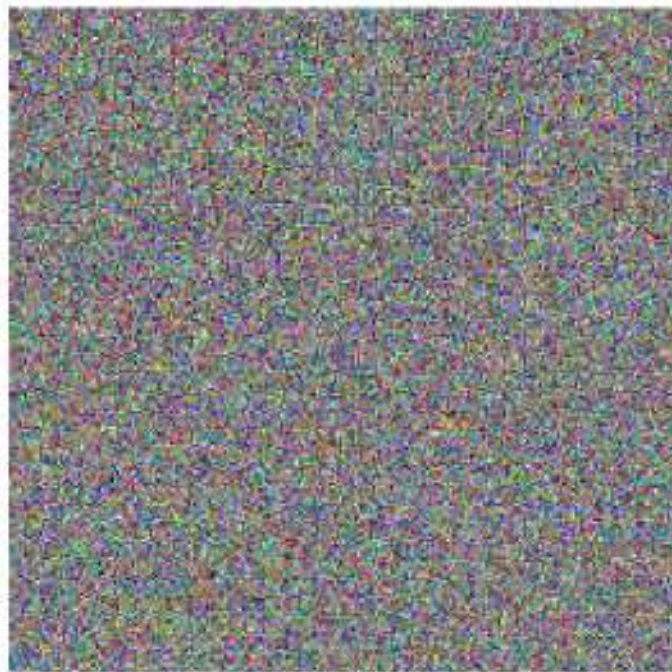
Input, Conv., Max., Conv., Max., Conv., Max, Full, Full

[Details zu den Gewinnern](#)

Visualisierung von gelernten Neuronalen Netzen

Neuronale Netze zur Objekterkennung in Bildern

Was erkennt ein Neuronales Netz in Rauschen, wenn es Bananen gelernt hat?



→
optimize
with prior



[Mehr Beispiele](#)

[Quelle: Heise: Wovon träumen neuronale Netze?](#)

Deep Learning Libraries

Theano

<http://deeplearning.net/software/theano/>

Python Implementierung für GPU-Verarbeitung von mathematischen Ausdrücken

Pylearn2

<http://deeplearning.net/software/pylearn2/>

Python Implementierung zu maschinellem Lernen, basierend auf Theano

Torch

<http://torch.ch/>

LuaJIT und C/CUDA Implementierung

Wird verwendet bei Facebook, Google, Twitter

DL4J

<http://deeplearning4j.org/>

Java Implementierung (multiplatform Unterstützung)

Kompatibel mit Spark, Hadoop