ML - ANN

Psychologie 2021

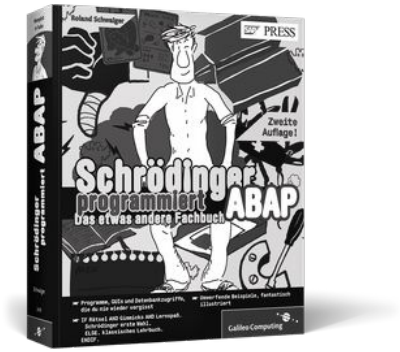
Dr. Roland Schwaiger

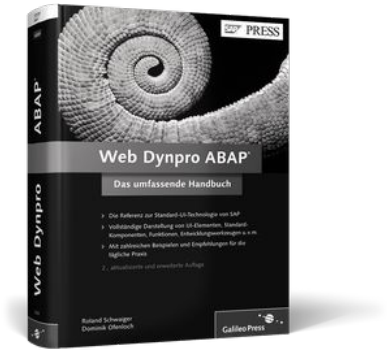
NoR Business Units

Human Rescources

Technologie

Künstliche Neuronale Netze
& Evolutionäre Algorithmen





www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Agenda

- Introduction / Overview
- Starter Kit
- A Simple Neural Network
- Learning for the Simple Neural Network

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

First steps

- Download material from https://www.rheinwerk-verlag.de/neuronale-netze-programmieren-mit-python/
- For Chapter 8:
  - Open and run Kapitel_08-InstallBibliotheken
    - “...mit conda” has not worked for me
    - “...mit pip” has worked

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Introduction / Overview

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Motivation

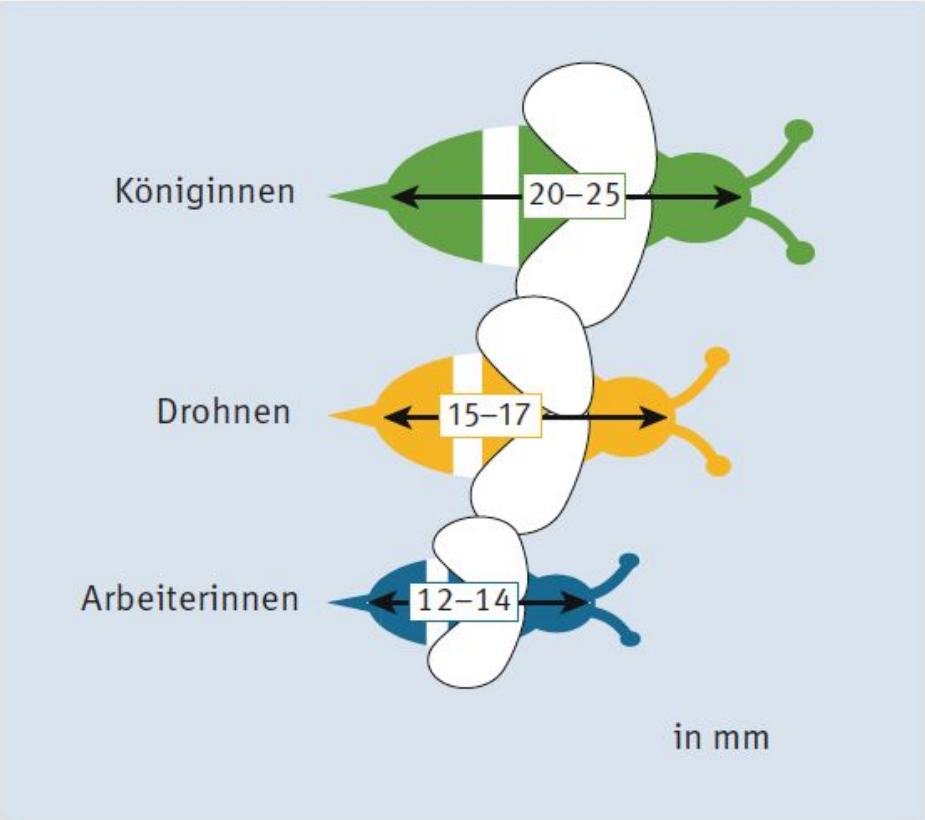


Abbildung 1.1 Die Bienenklassifikation

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

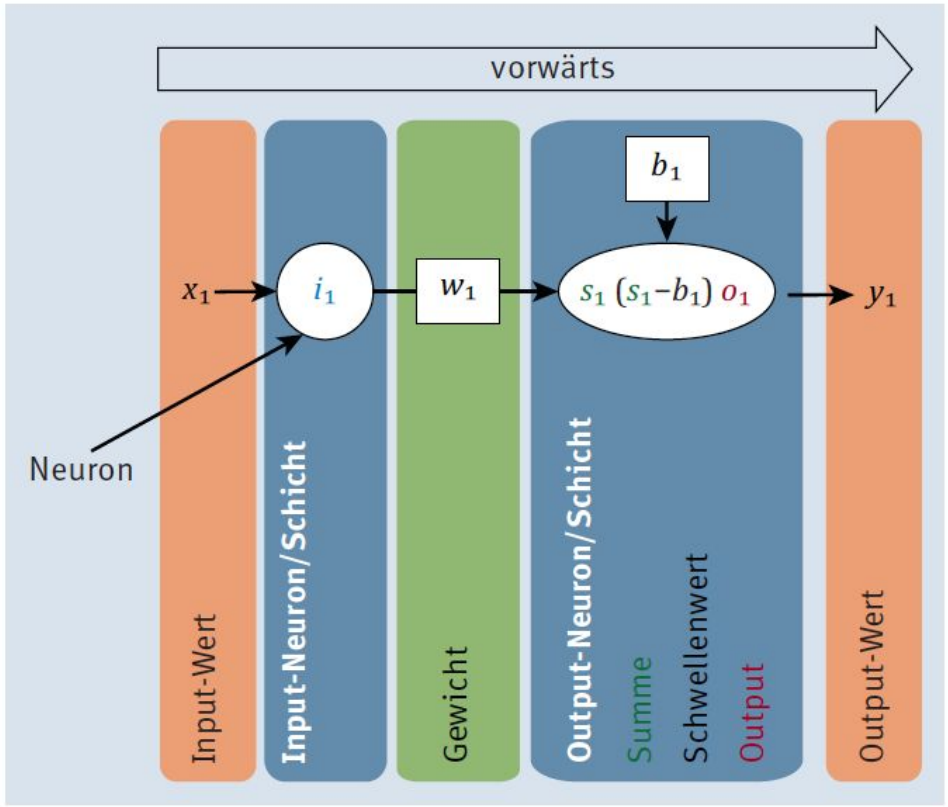


Abbildung 1.2 Ein einfaches KNN

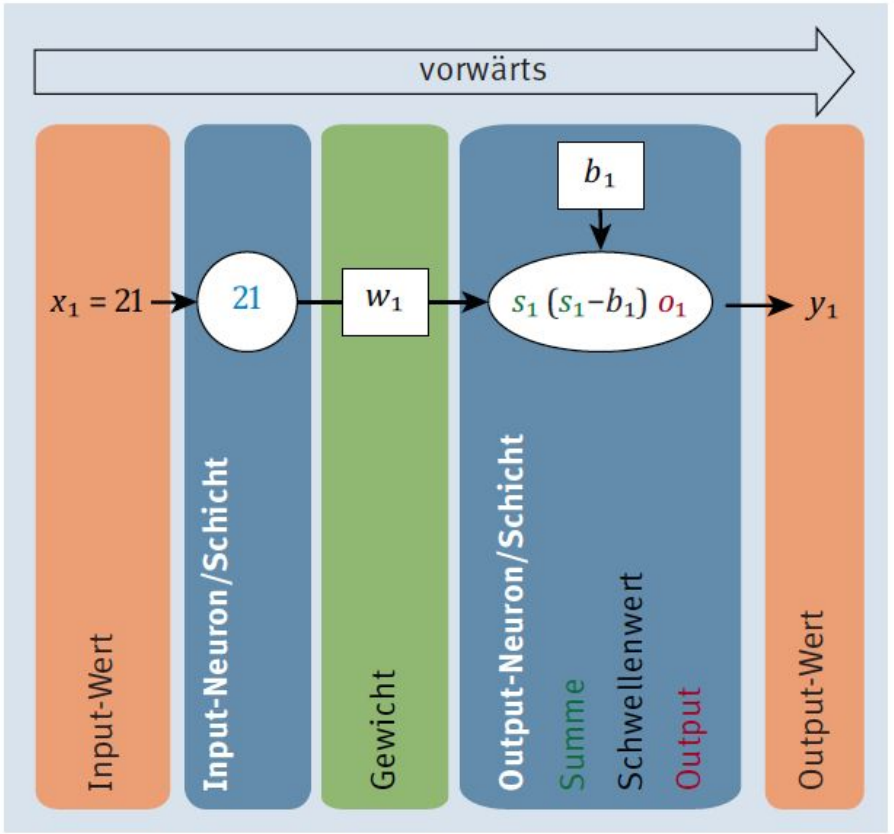


Abbildung 1.3 Der Input-Wert 21 wird an das Netz übergeben.

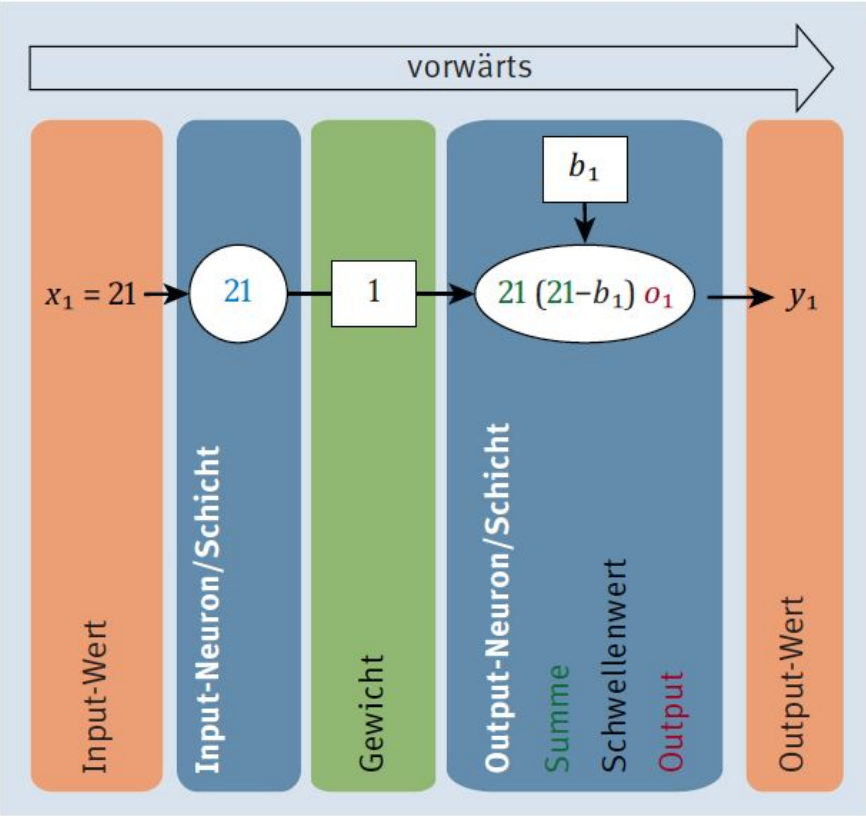


Abbildung 1.4 Der gewichtete Input zum Output-Neuron wird berechnet.

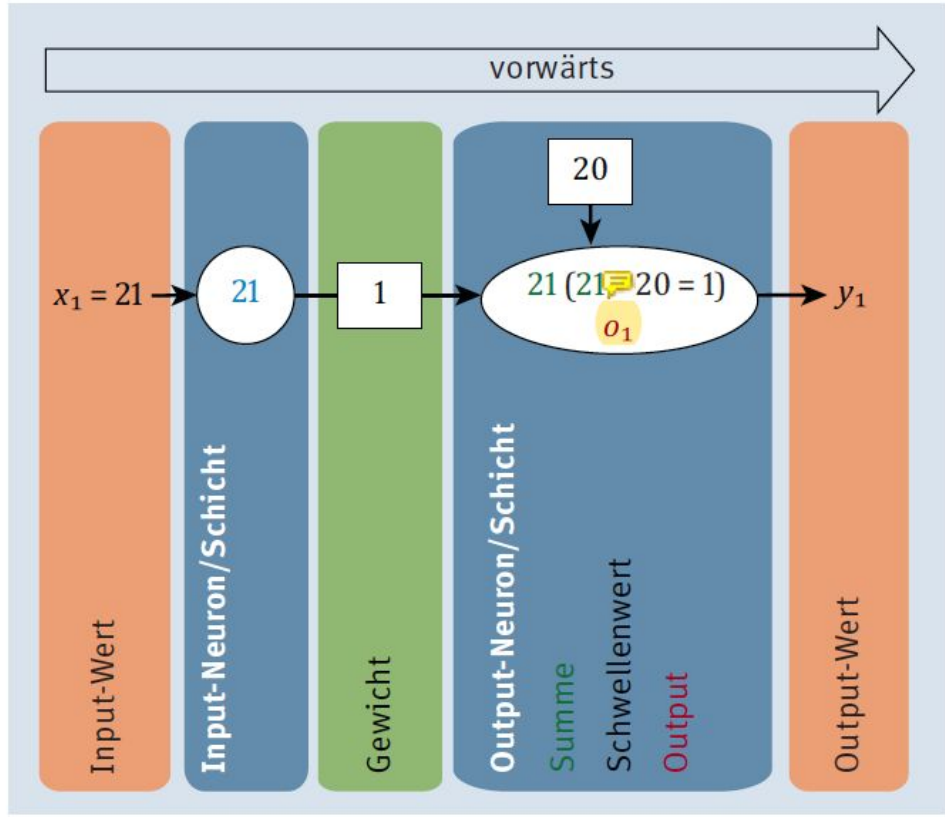


Abbildung 1.5 Schwellenwertberechnung

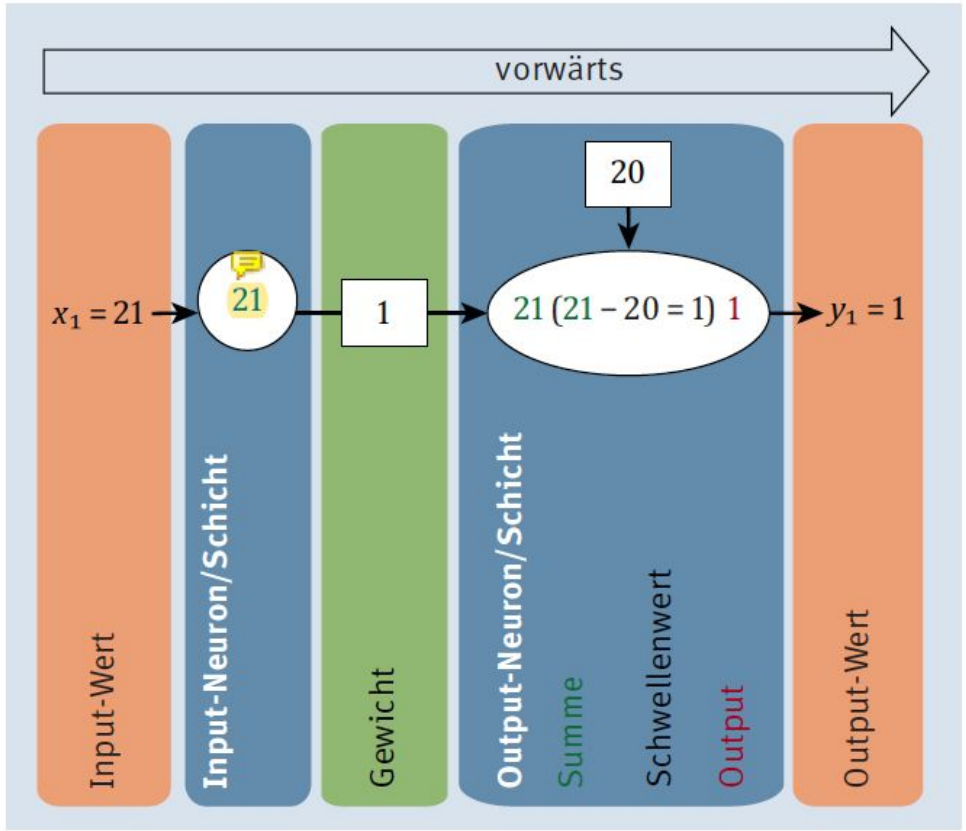


Abbildung 1.6 Output-Berechnung

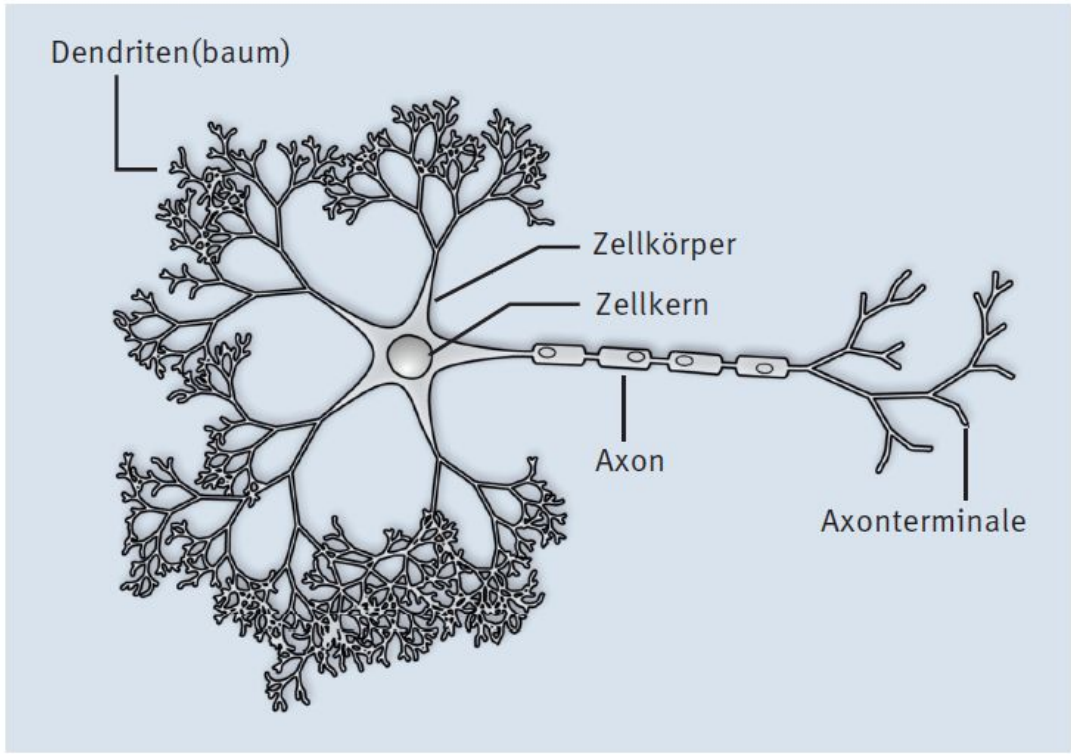


Abbildung 1.7 Schematische Darstellung eines Neurons (von Nicolas.Rougier/CC-BY-SA-3.0, https://commons.wikimedia.org/w/index.php?curid=2192116)

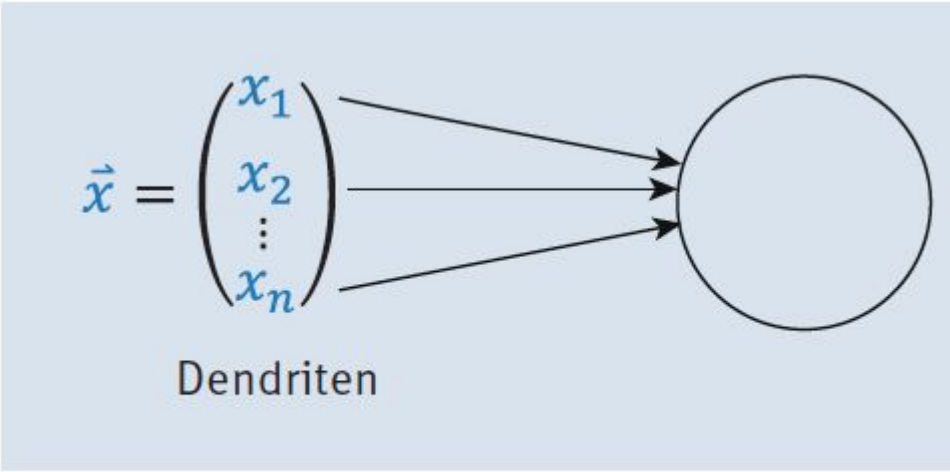


Abbildung 1.8 Künstliche Dendriten

$$\sum_{i=1}^{n} x_i \cdot w_i$$

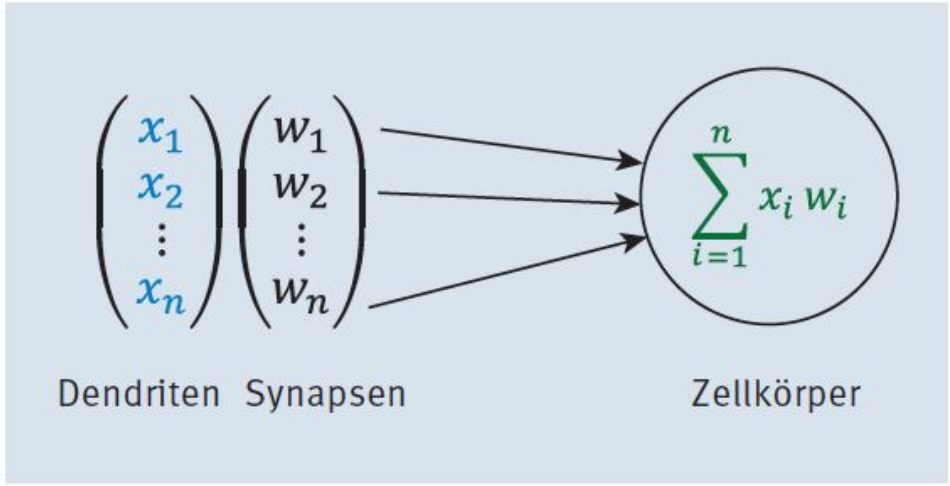


Abbildung 1.9 Künstliche Dendriten mit Synapsen(gewichten)

Activation Function

$$y = f_{\text{akt}}\left(\sum_{i=1}^{n} x_i w_i\right)$$

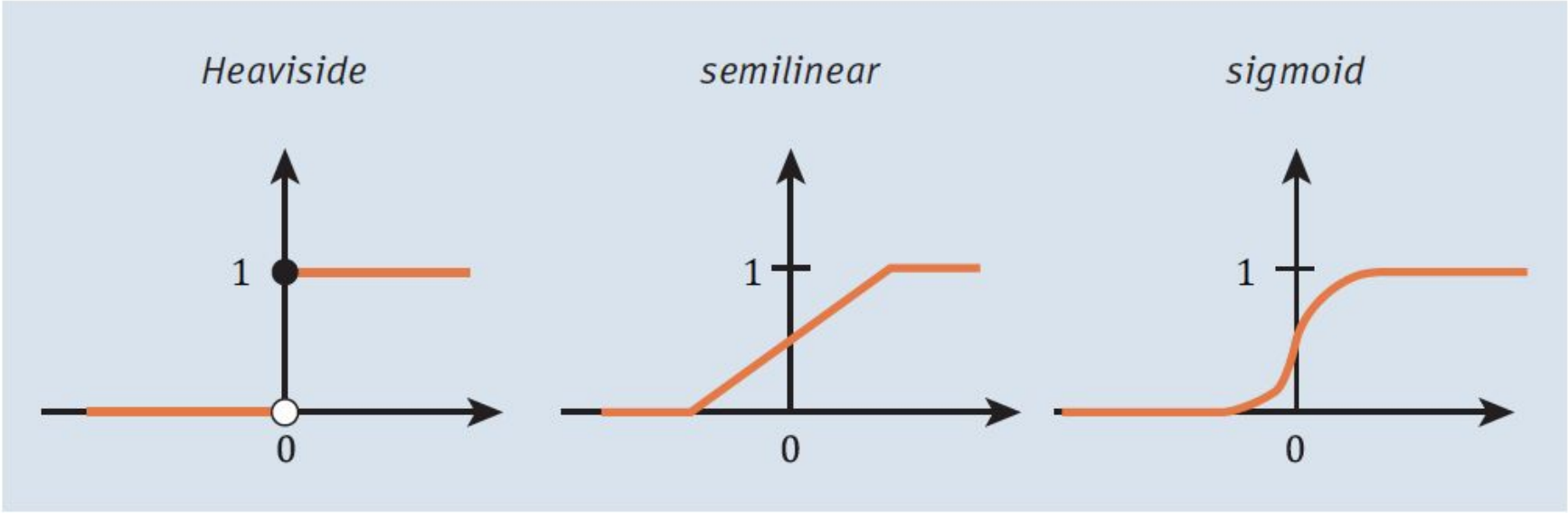


Abbildung 1.10 Beispiele für die Aktivierungsfunktion fakt

Artificial Neuron

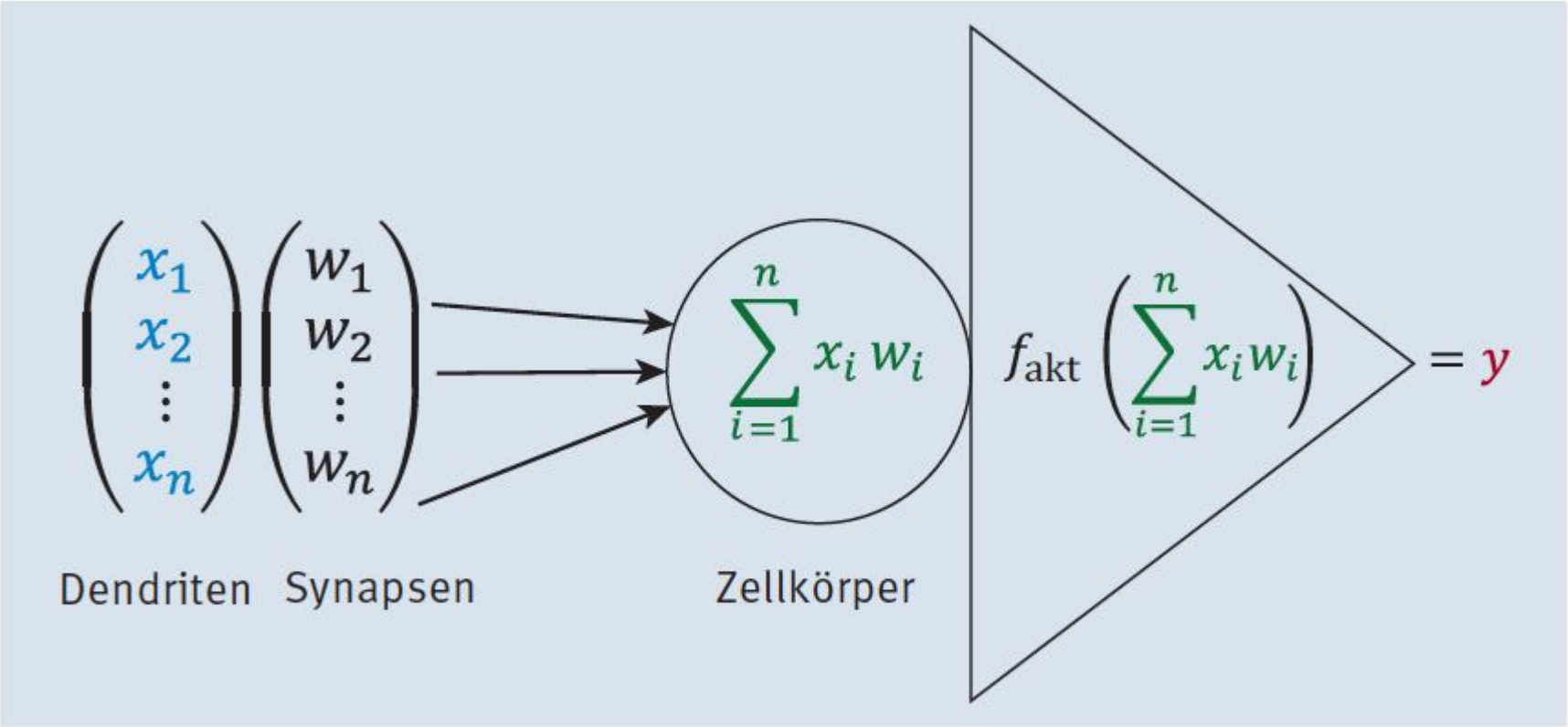


Abbildung 1.11 Das künstliche Neuron

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

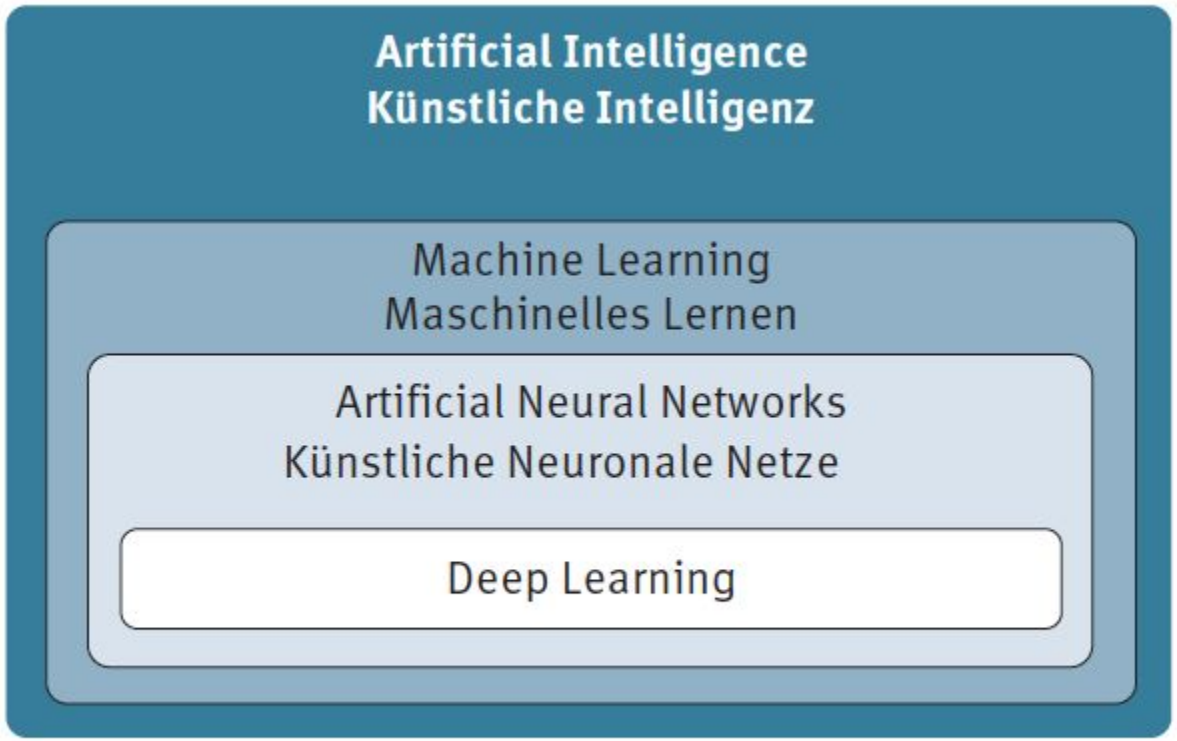


Abbildung 1.12 Begriffszwiebel für künstliche Intelligenz

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

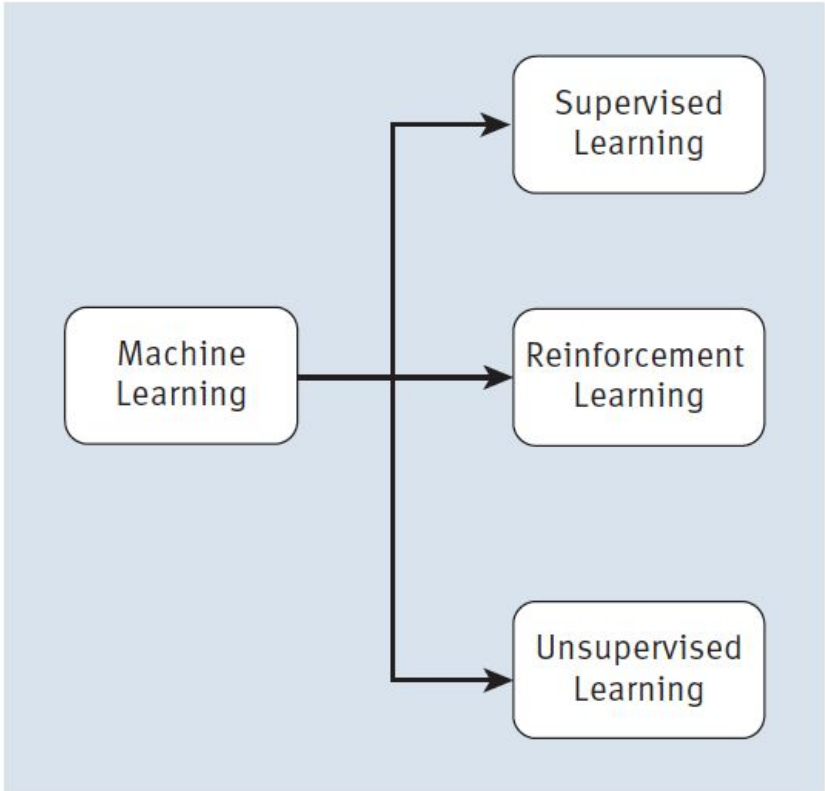


Abbildung 1.13 Lernstrategien im Machine Learning

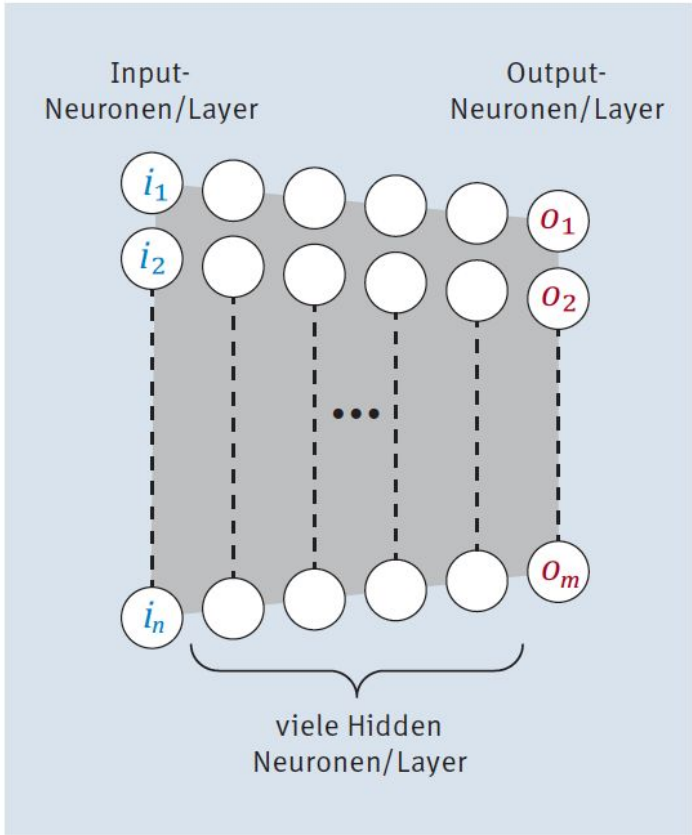


Abbildung 1.14 Deep-Neural-Network-Schema

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Starter Kit

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Ingrediences

- Anaconda
- Jupyter Notebook
- Python
  - NumPy
  - matplotlib
  - scikit-learn
  - pandas
  + TensorFlow 2
  + Keras

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

A Simple Neural Network

Frau Karotte	Herr Lauch	-->	Montag
nicht anwesend	nicht anwesend	-->	**nicht ok**
anwesend	nicht anwesend	-->	**ok**
nicht anwesend	anwesend	-->	**ok**
anwesend	anwesend	-->	**ok**

Tabelle 3.1 Erster Personalplan

Frau Karotte	Herr Lauch	-->	Montag
0	0	-->	0
1	0	-->	1
0	1	-->	1
1	1	-->	1

Tabelle 3.2 Zweiter Personalplan mit KNN-geeigneter Codierung

Frau Karotte	Berechnung	Herr Lauch	-->	Summe	Ergebnis
0	+	0	=	0	0
1	+	0	=	1	1
0	+	1	=	1	1
1	+	1	=	2	1

Tabelle 3.3 Errechneter Personalplan aus den Anwesenheiten

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

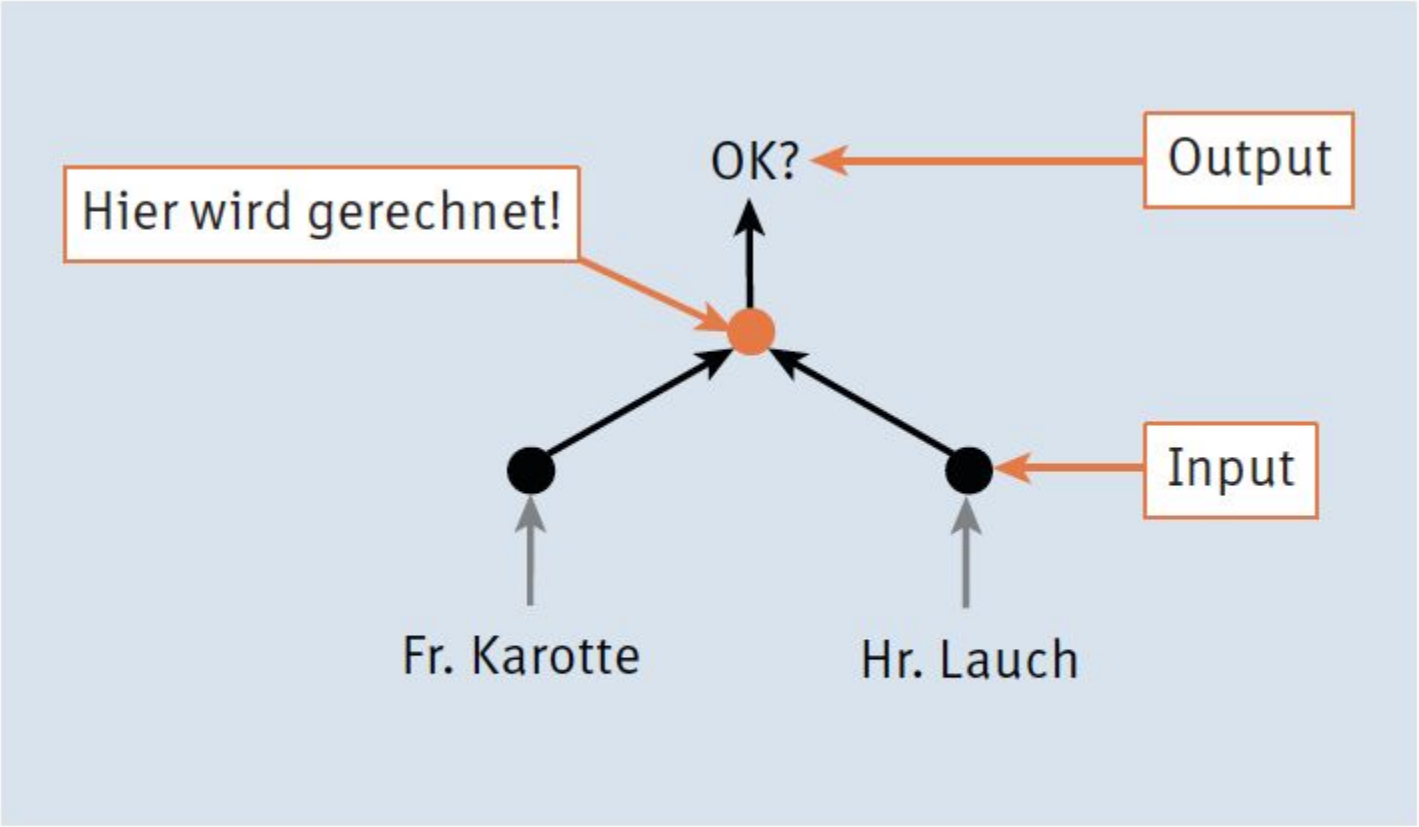


Abbildung 3.2 Die Entscheidung für Frau Karotte und Herrn Lauch berechnen

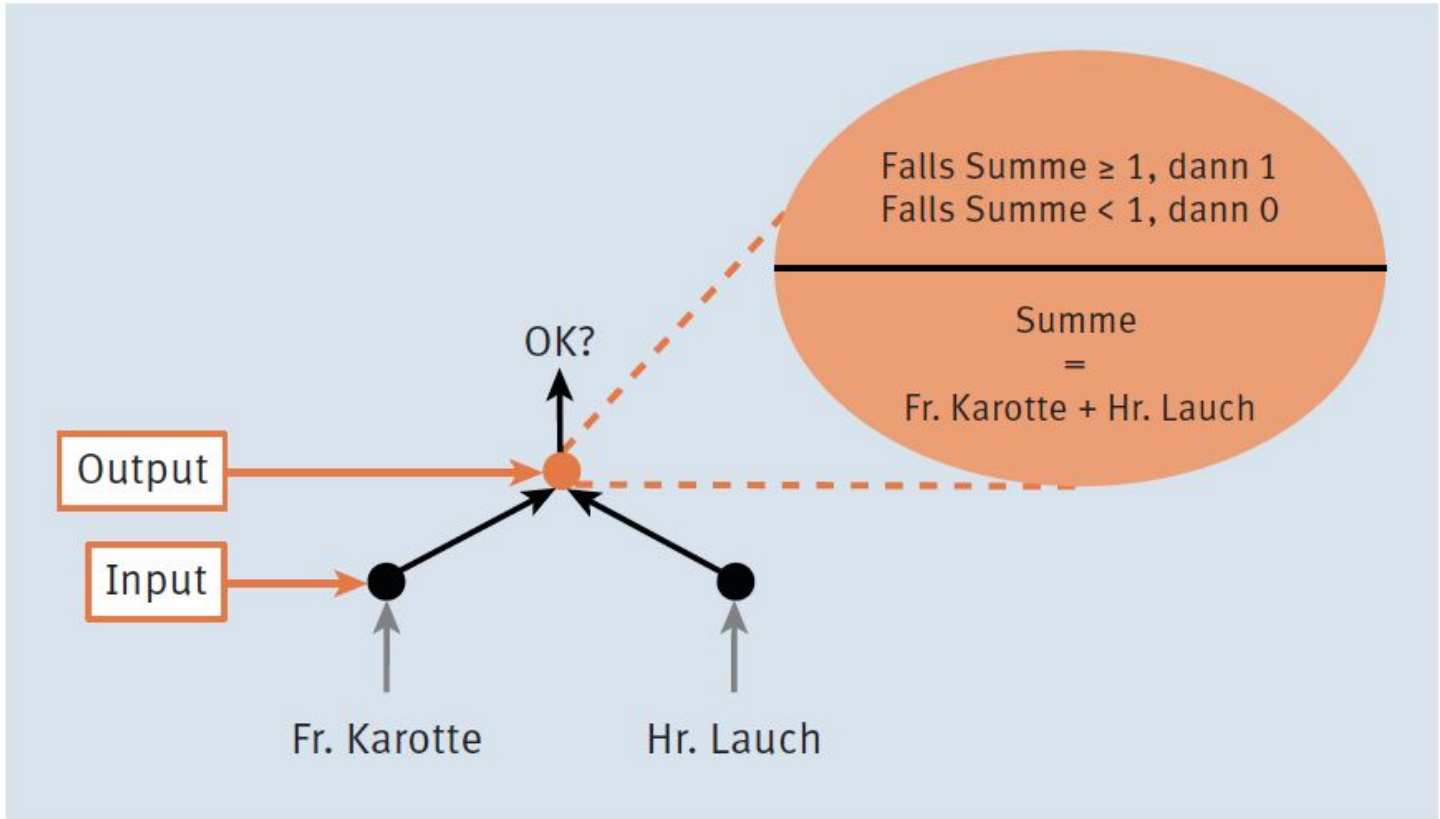


Abbildung 3.3 Ergebnisberechnung für Frau Karotte und Herrn Lauch im Detail

3.1

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

3.2 pyplot

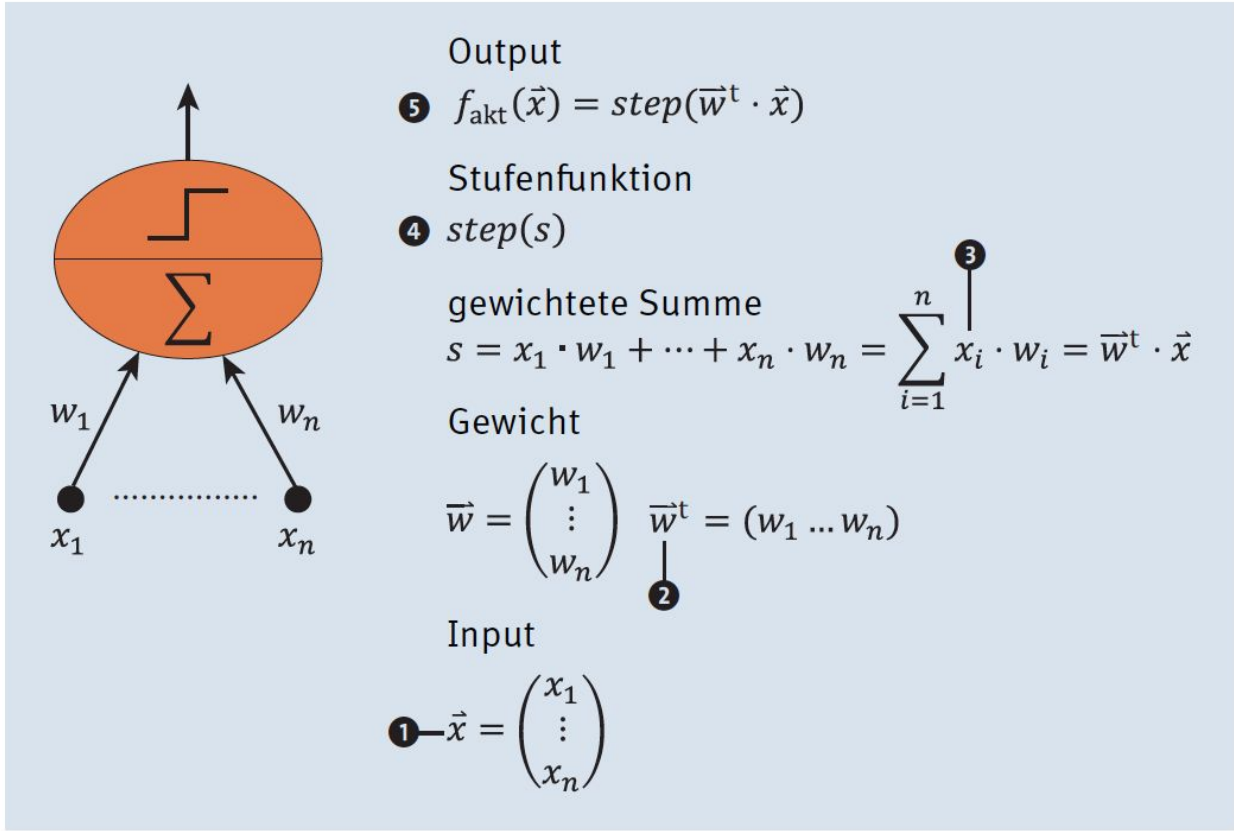


Abbildung 3.9 Perceptron – Bestandteile und Berechnungsbausteine

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

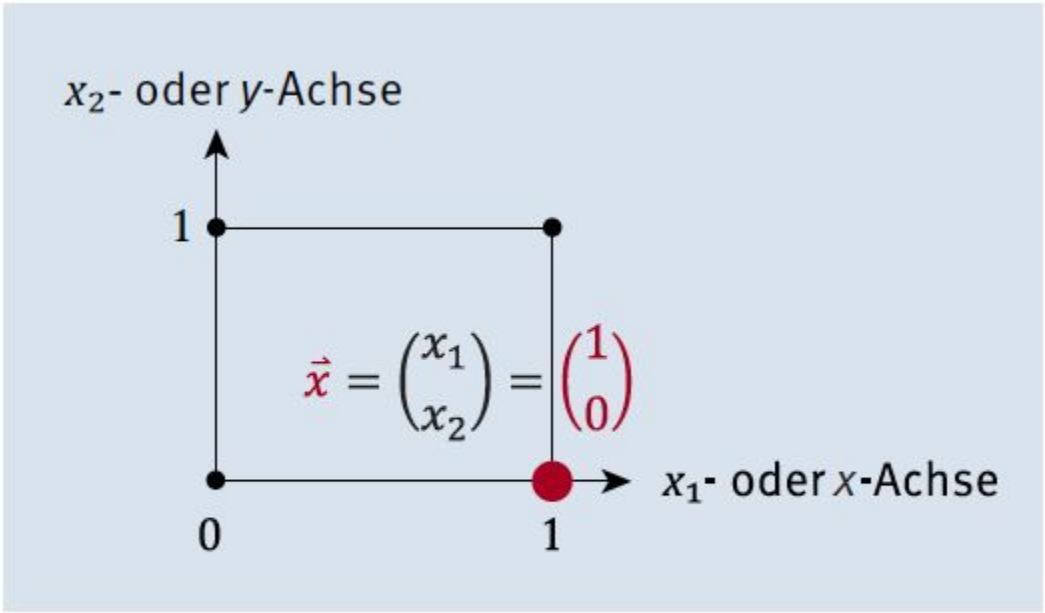


Abbildung 3.10 Punkt im kartesischen Koordinatensystem

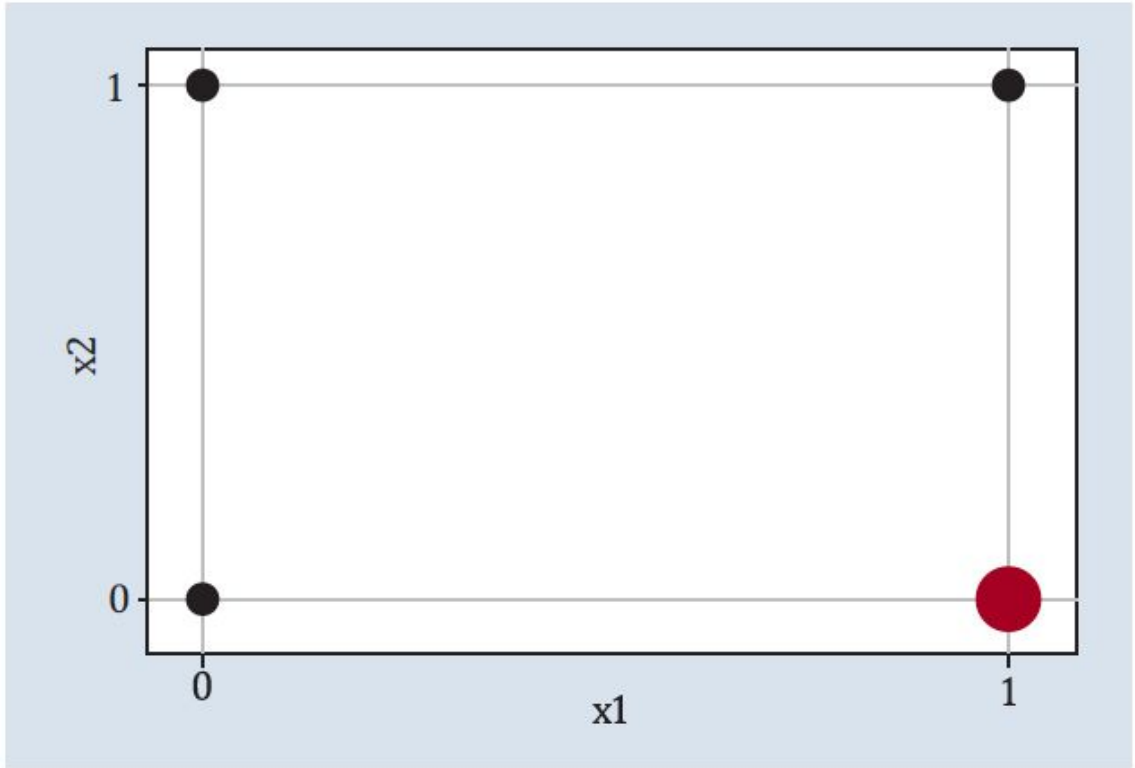


Abbildung 3.12 Scatter-Plot für die Planungspunkte

3.3 Scatter-Plot

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Iris Scatter Plot

Sepal Length	Sepal Width	Petal Length	Petal Width	Class
5.1	3.5	1.4	0.2	Iris-setosa
7.0	3.2	3.5	1.0	Iris-vesicolor
6.3	3.3	6.0	2.5	Iris-virginica
…	…	…	…	…

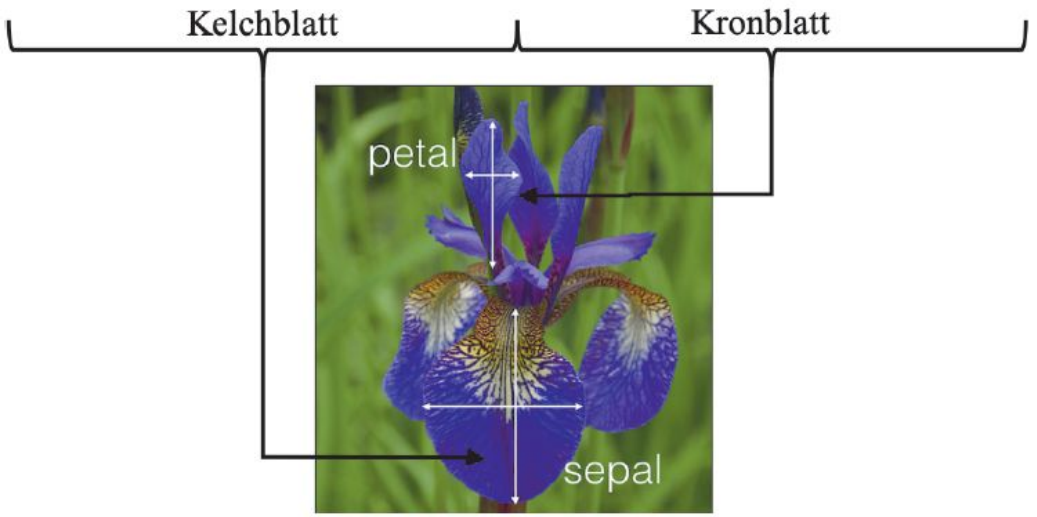


Abbildung 3.13 Blattmaße für Schwertlilien (© Kaggle)

3.4 Scatter-Plot

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

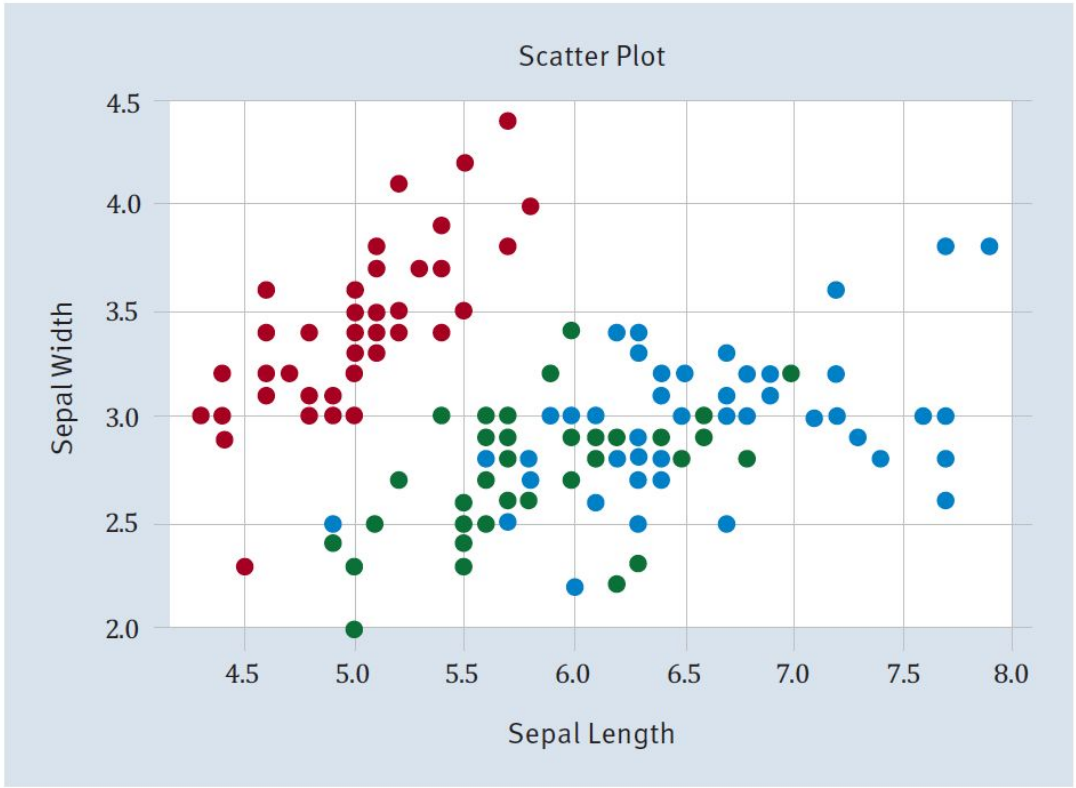


Abbildung 3.14 Die Lilien als Scatter-Plot mit den Koordinaten »Sepal Length« und »Sepal Width«

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Scalar Product

$$\vec{w}^{\mathrm{t}} \cdot \vec{x} = (w_1 w_2 \dots w_n) \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n$$

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

3.5 Scalar Product

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

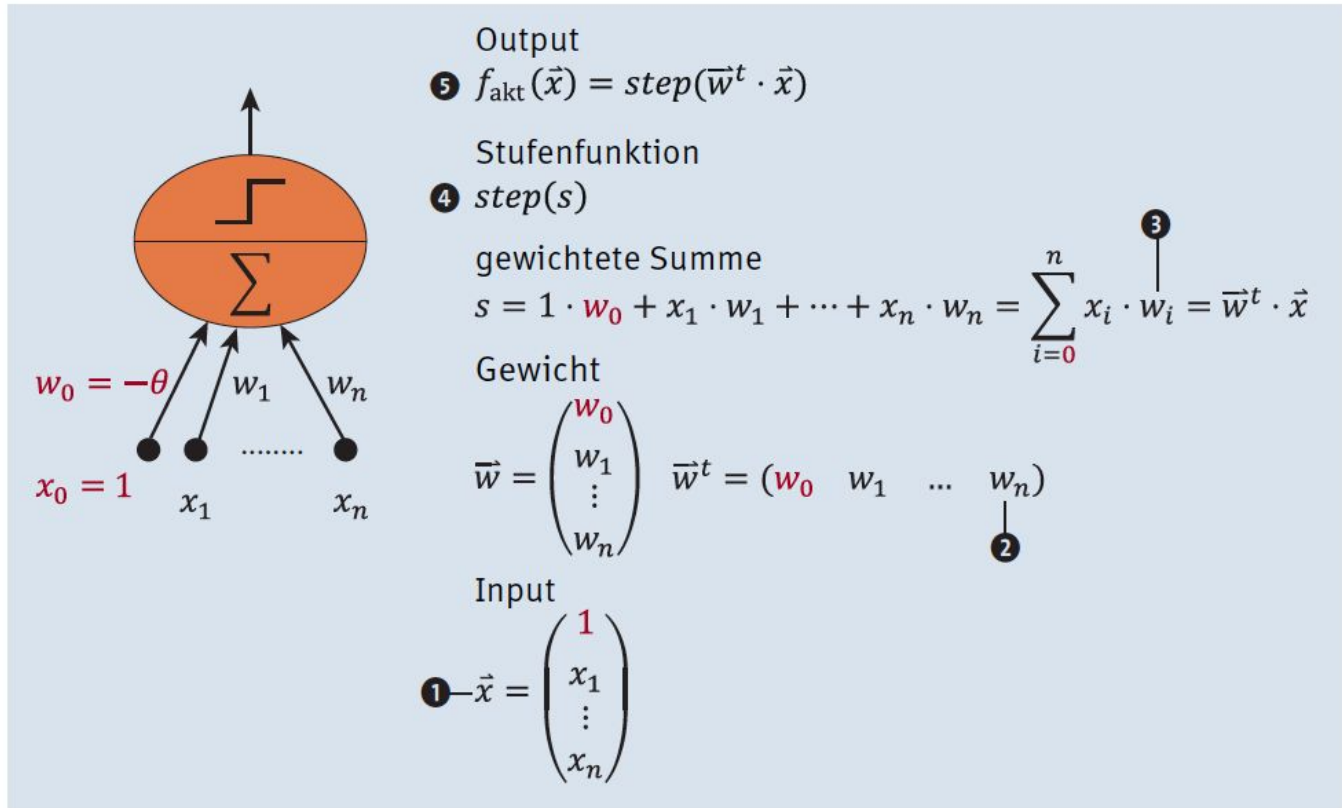


Abbildung 3.15 Bausteine des Perceptrons

Bias-Neuron	Frau Karotte	Herr Lauch	-->	Montag	Input-Vektor
1	0	0	-->	0	1
1	0	1	-->	1	2
1	1	0	-->	1	3
1	1	1	-->	1	4

Tabelle 3.4 Der erweiterte Input-Vektor für das Planungsproblem

3.6 Planning Problem

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Exercise (Solution 3.7)

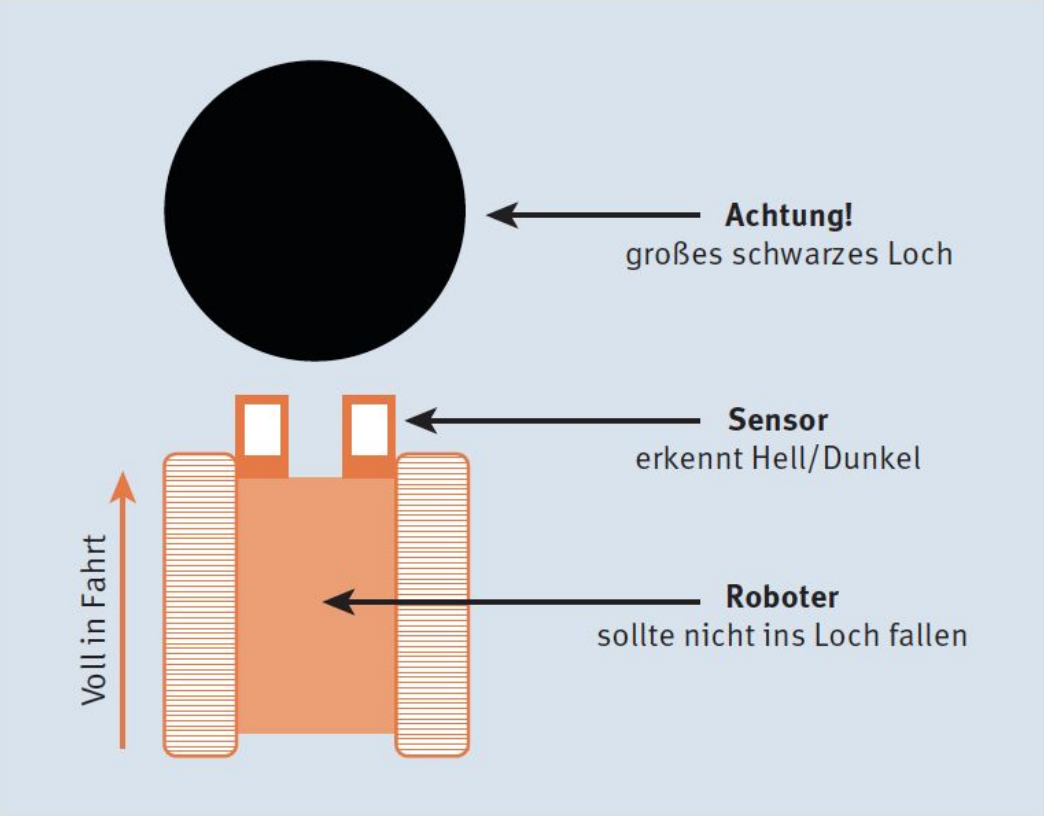


Abbildung 3.16 Einfache Robotersensorik

Sensor links	Sensor rechts	Loch
0	0	0
1	0	0
0	1	0
1	1	1

Tabelle 3.5 Locherkennung mithilfe der Sensorwerte

Learning for the Simple NN

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

P1	P2	-->	E
0	0	-->	0
0	1	-->	1
1	0	-->	1
1	1	-->	1

Tabelle 4.1 Das altbekannte Planungsproblem

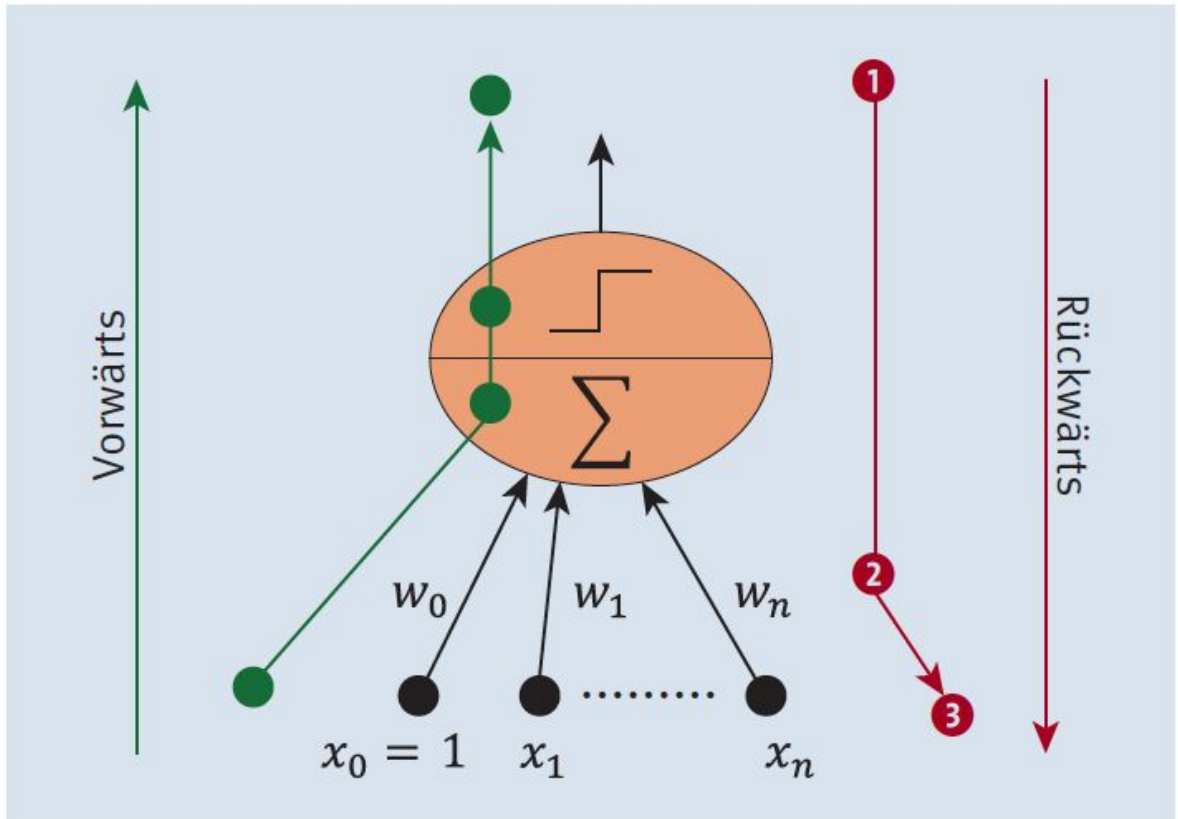


Abbildung 4.1 Vorwärts und rückwärts im Perceptron

$$s = \sum_{i=0}^{n} w_i \cdot x_i = \vec{w}^t \cdot \vec{x}$$

$s = 0.1 \cdot x_0 + 0.1 \cdot x_1 + 0.1 \cdot x_2$

Und für den Input $\vec{x} = (1.0, 0.0, 0.0)$ ergibt das:

$s = 0.1 \cdot 1.0 + 0.1 \cdot 0.0 + 0.1 \cdot 0.0 = 0.1$

Nun sind Sie an der Reihe!

Aufgabe

Wie sieht die Gleichung für die Gewichtswerte $\vec{w}^t = (-0.9, 0.1, 0.1)$ aus?

Wie sieht der Wert für $\vec{x} = (1.0, 0.0, 0.0)$ aus?

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Weight Change

$$w_i^{\text{neu}} = w_i^{\text{alt}} + \Delta w_i$$

wobei

$$\Delta w_i = (y - \hat{y}) \cdot x_i$$

w_i^{alt} bezeichnet das Gewicht mit dem aktuellen Wert, also zum Beispiel –0.3.

w_i^{neu} ist das Gewicht nach einer Änderung des alten Gewichts, also falls zum Beispiel 0.1 zum alten Gewicht dazugezählt wird. Das wäre dann bei unserem Beispiel –0.3 + 0.1 = –0.2.

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

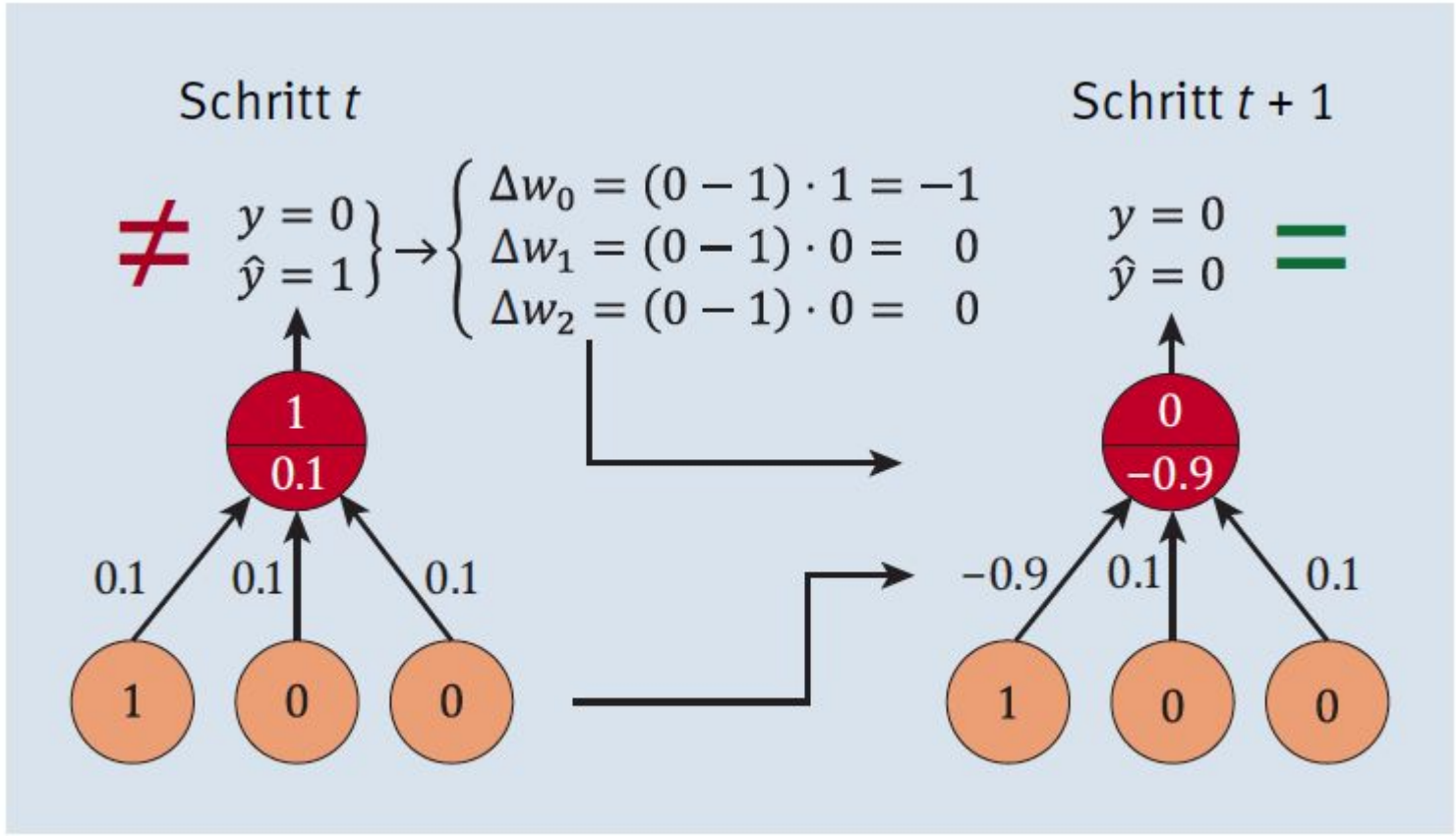


Abbildung 4.2 Ein Perceptron-Lernschritt

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

y_i	$\hat{y}_i$	$(y - \hat{y})$	$(y - \hat{y}) \cdot x_i$
0	0	0	0
0	1	–1	$-x_i$
1	0	1	x_i
1	1	0	0

Tabelle 4.2 Mögliche Fehler im Perceptron und die dazu passende Änderung des Gewichts

Mit diesen Erkenntnissen können Sie nun wieder in die Formel $w_i^{neu} = w_i^{alt} + \Delta w_i$ gehen und Ersetzungen durchführen:

- w_i^{neu} ist der Wert im Schritt t + 1, also können wir $w_i(t+1)$ schreiben.
- w_i^{alt} ist der Wert im Schritt t, also können wir $w_i(t)$ schreiben.

Somit ergeben sich die folgenden Berechnungsschritte für die Veränderung der Gewichte:

- $w_i(t+1) = w_i(t)$ bei korrekter Ausgabe
- $w_i(t+1) = w_i(t) + x_i$ bei Ausgabe 0 und gewünschter Ausgabe 1
- $w_i(t+1) = w_i(t) - x_i$ bei Ausgabe 1 und gewünschter Ausgabe 0

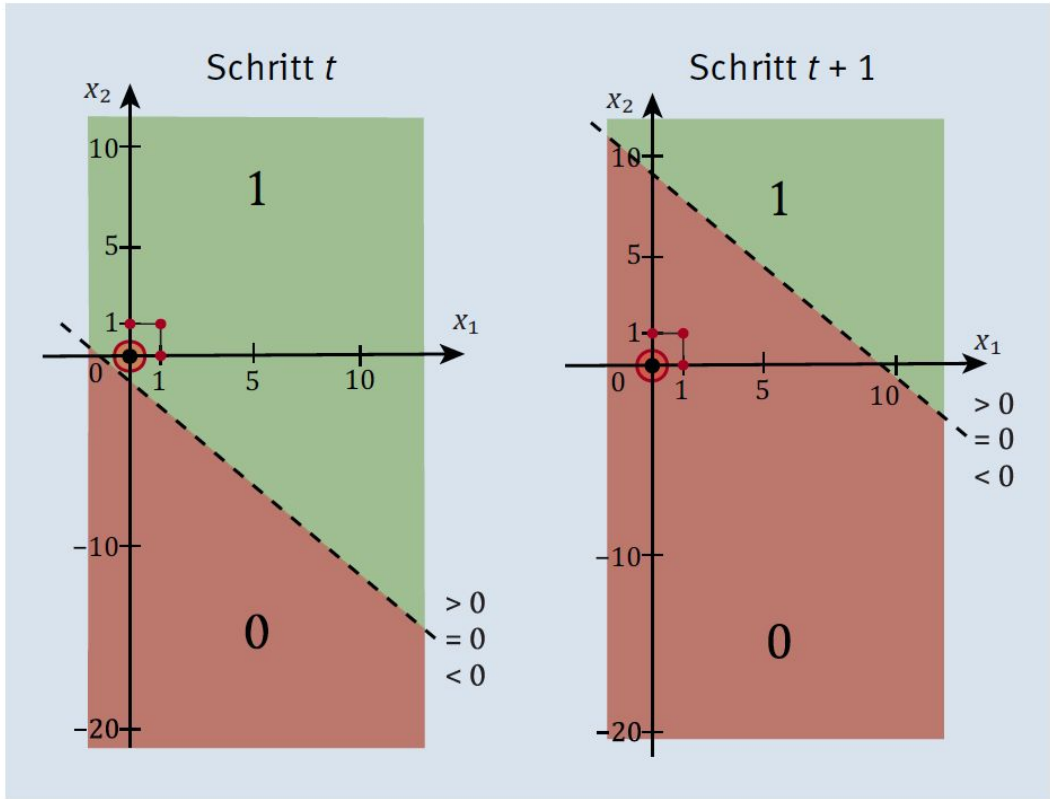


Abbildung 4.3 Die Veränderung der Lage der Trenngeraden durch einen Lernschritt, in dem die Gewichte angepasst werden

4.1 Perceptron Learning

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

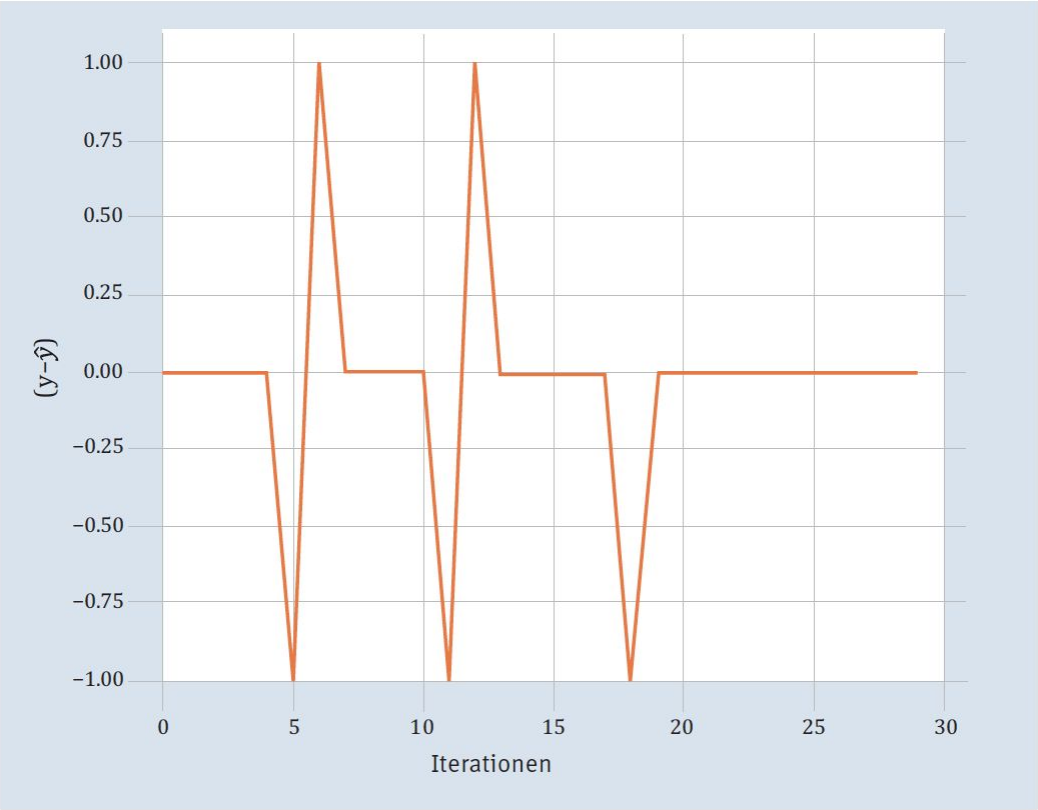


Abbildung 4.4 Die Differenz zwischen gewünschtem und errechnetem Output $(y - \hat{y})$ beim Lernen, pro zufällig gewähltem Trainingsbeispiel

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

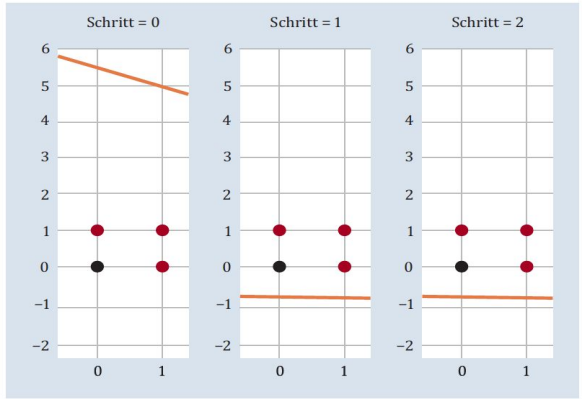


Abbildung 4.5 Schritt 0, Schritt 1 und Schritt 2

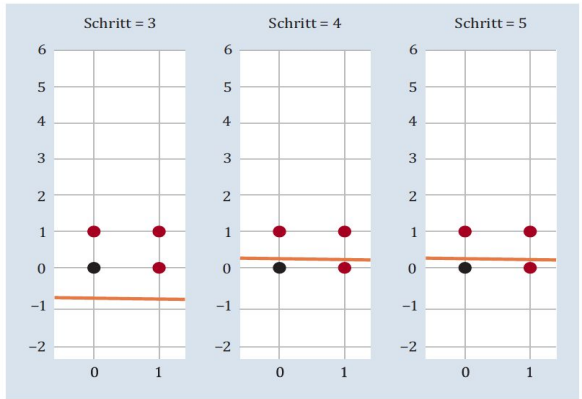


Abbildung 4.6 Schritt 3, Schritt 4 und Schritt 5

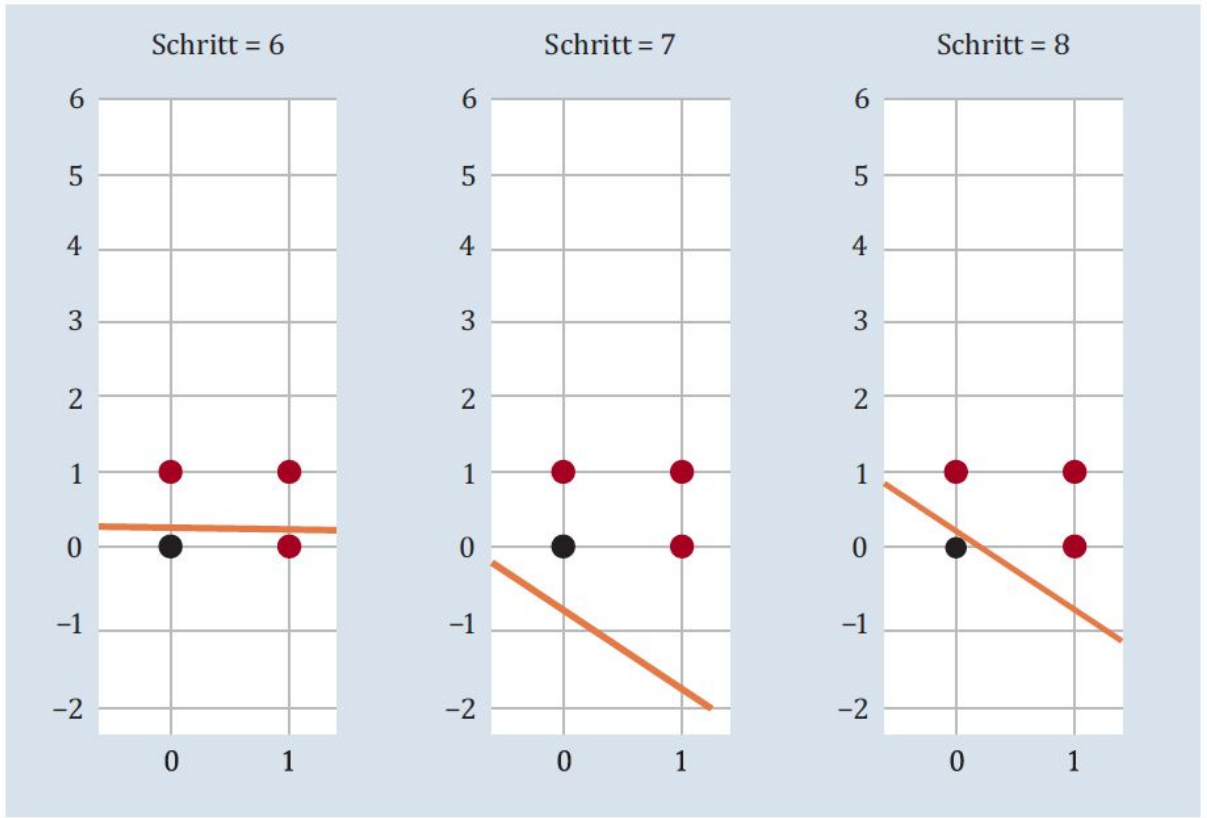


Abbildung 4.7 Schritt 6, Schritt 7 und Schritt 8

4.2,3,4,5,6,7,8,9 Scikit compatible Estimator

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

4.10 scikit-learn-Perceptron-Estimator

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

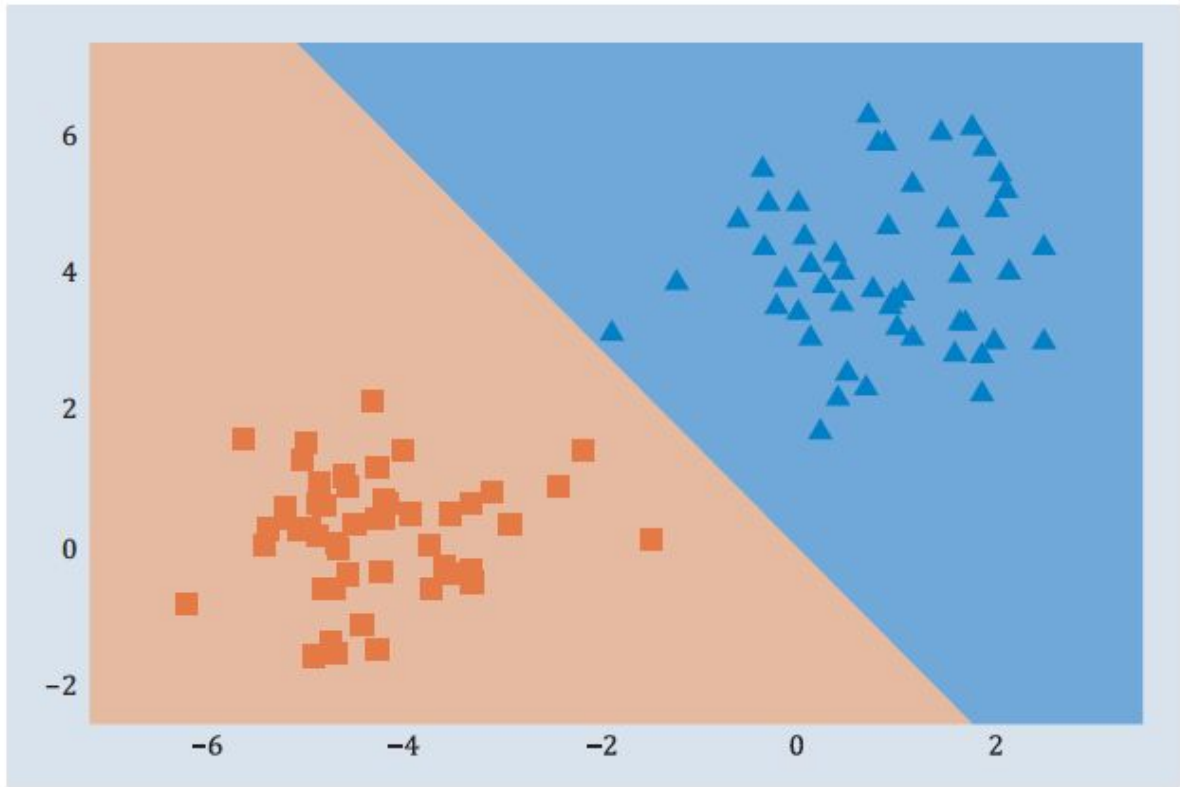


Abbildung 4.9 scikit-learn-Perceptron-Auswertung und Datenpunkte

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Adaline

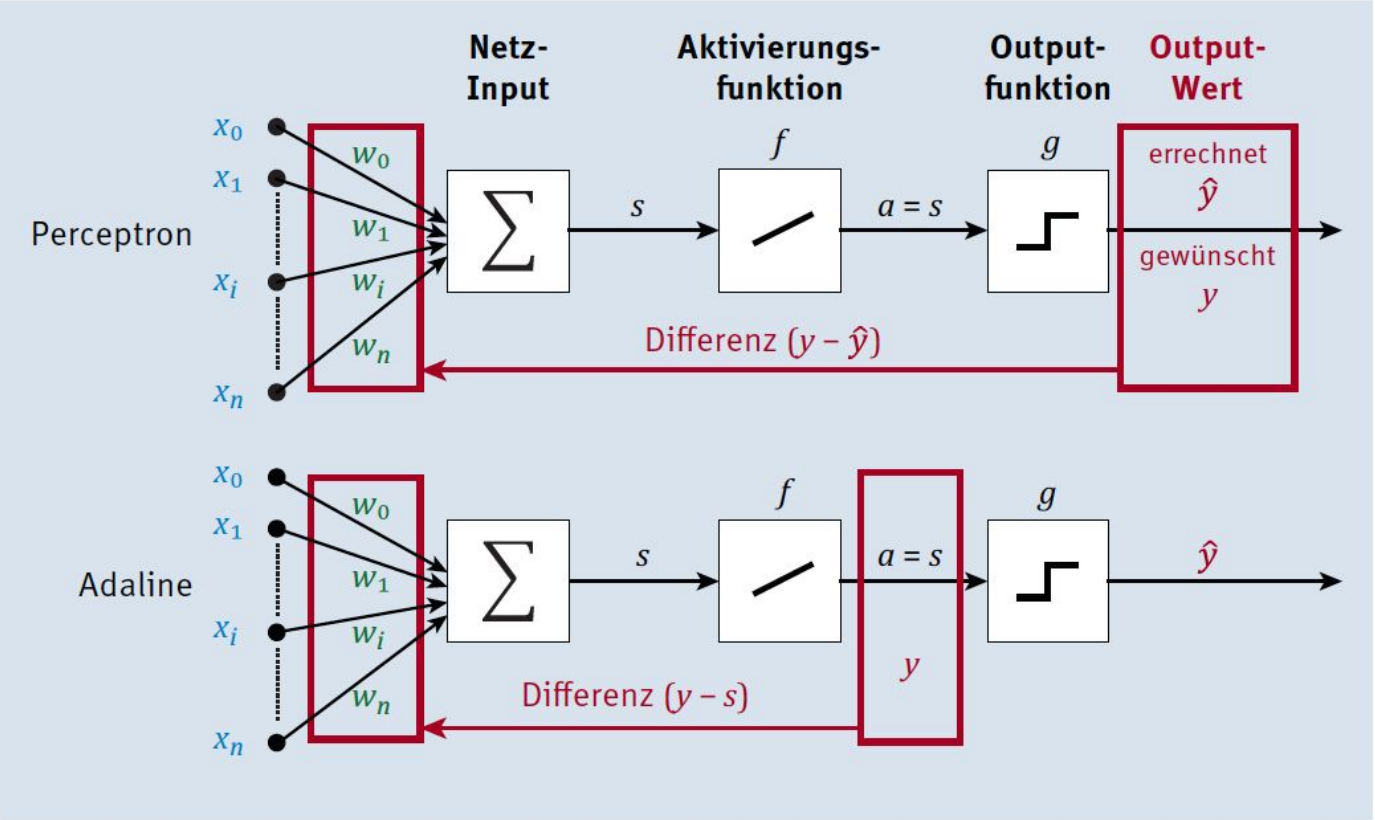


Abbildung 4.10 Differenzermittlung für das Perceptron und Adaline im Vergleich

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Learning Rate

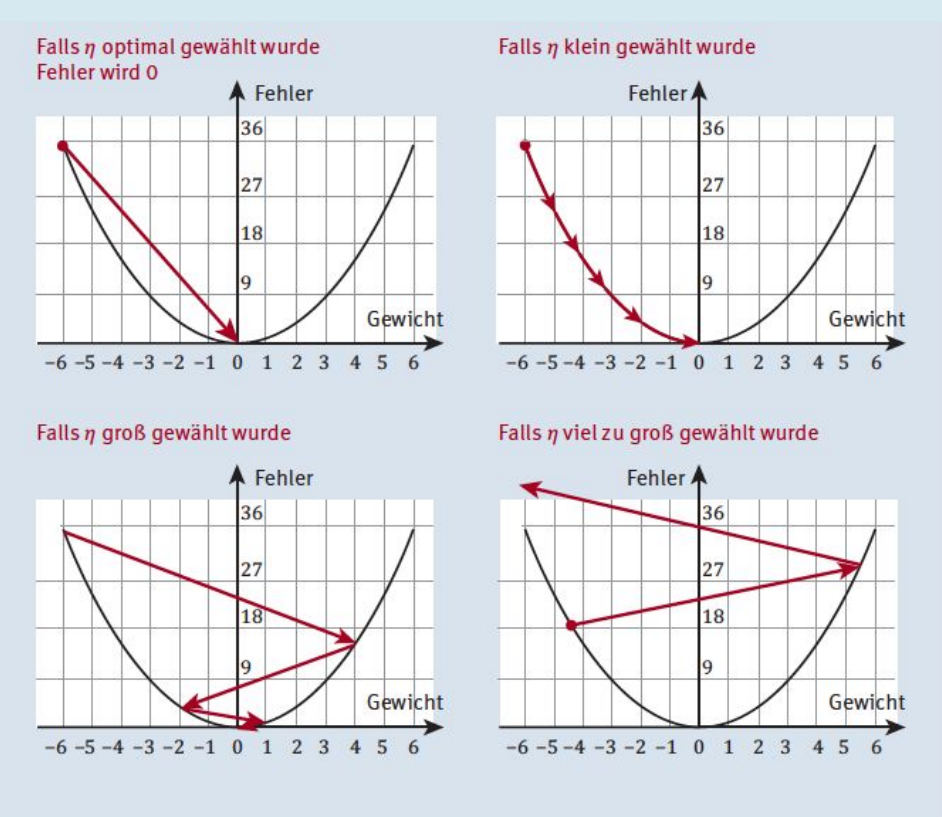


Abbildung 4.11 Auswirkung der Wahl der Lernrate auf die Fehlerminimierung

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

4.11,12,13,14,15 Adaline-Estimator

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

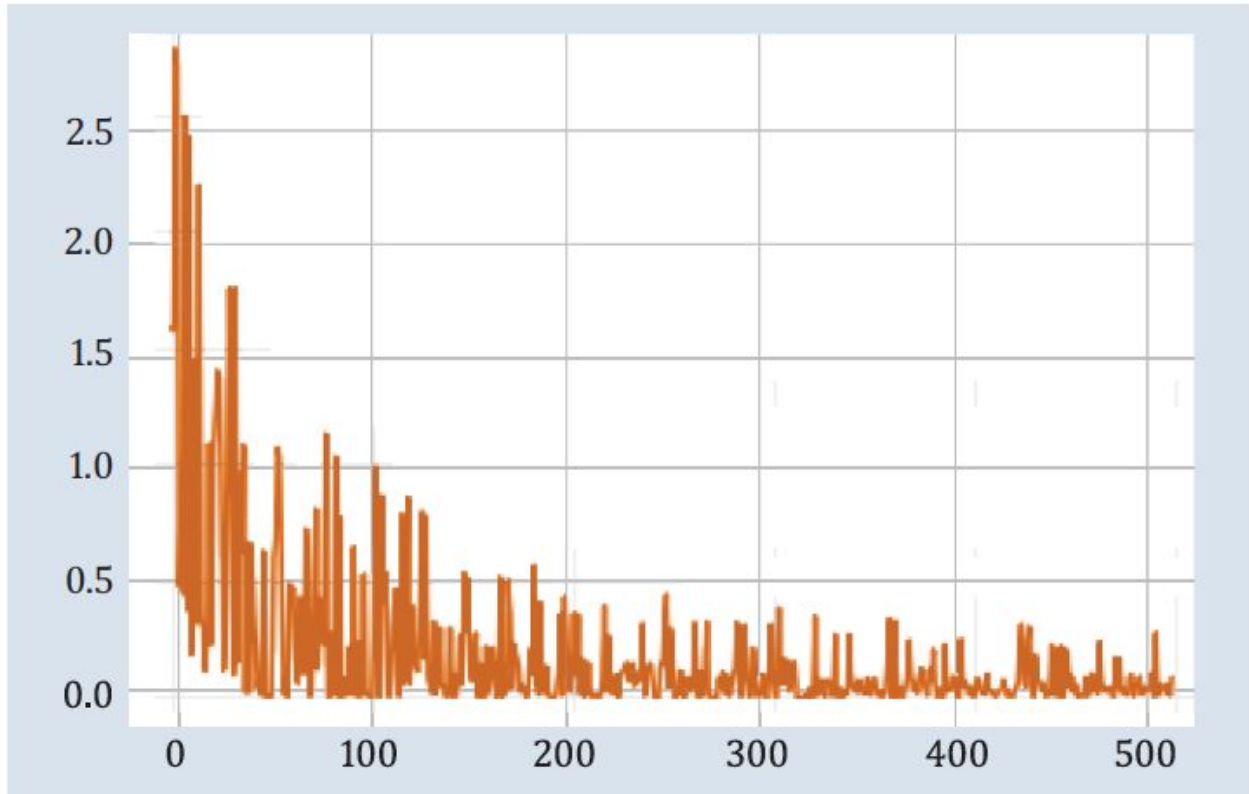


Abbildung 4.12 Adaline-Lernkurve

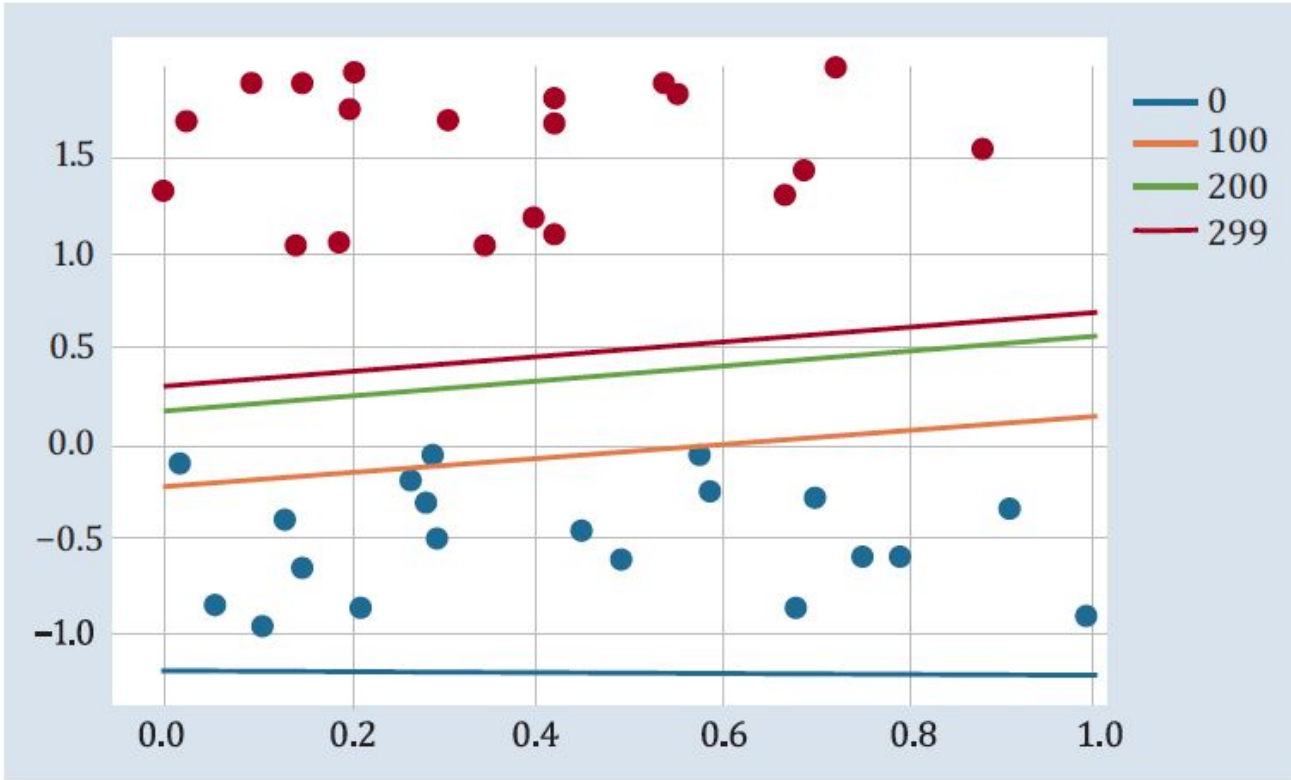


Abbildung 4.13 Adaline-Trenngeraden in unterschiedlichen Schritten

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

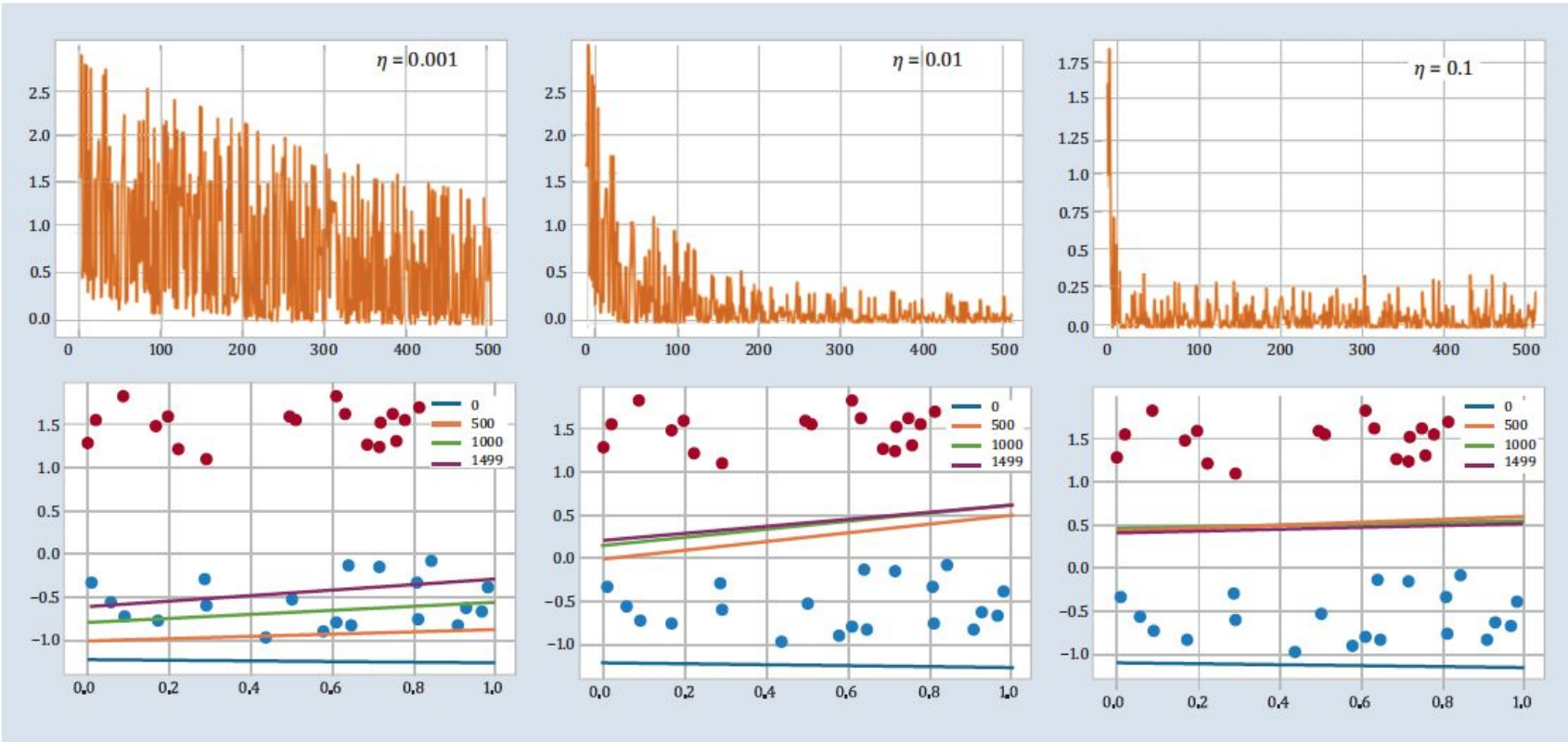


Abbildung 4.14 Lernratenvergleich

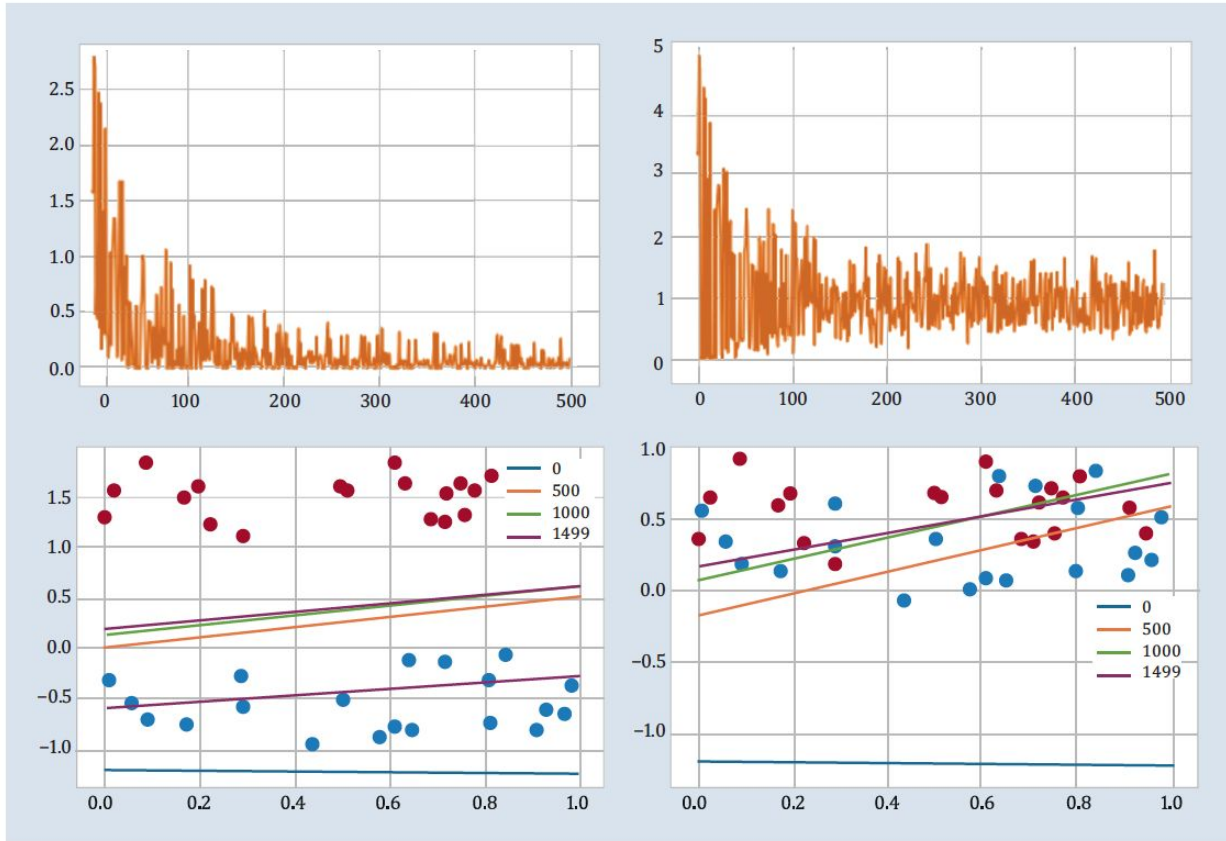


Abbildung 4.15 Überlappungsfaktor der zwei Klassen

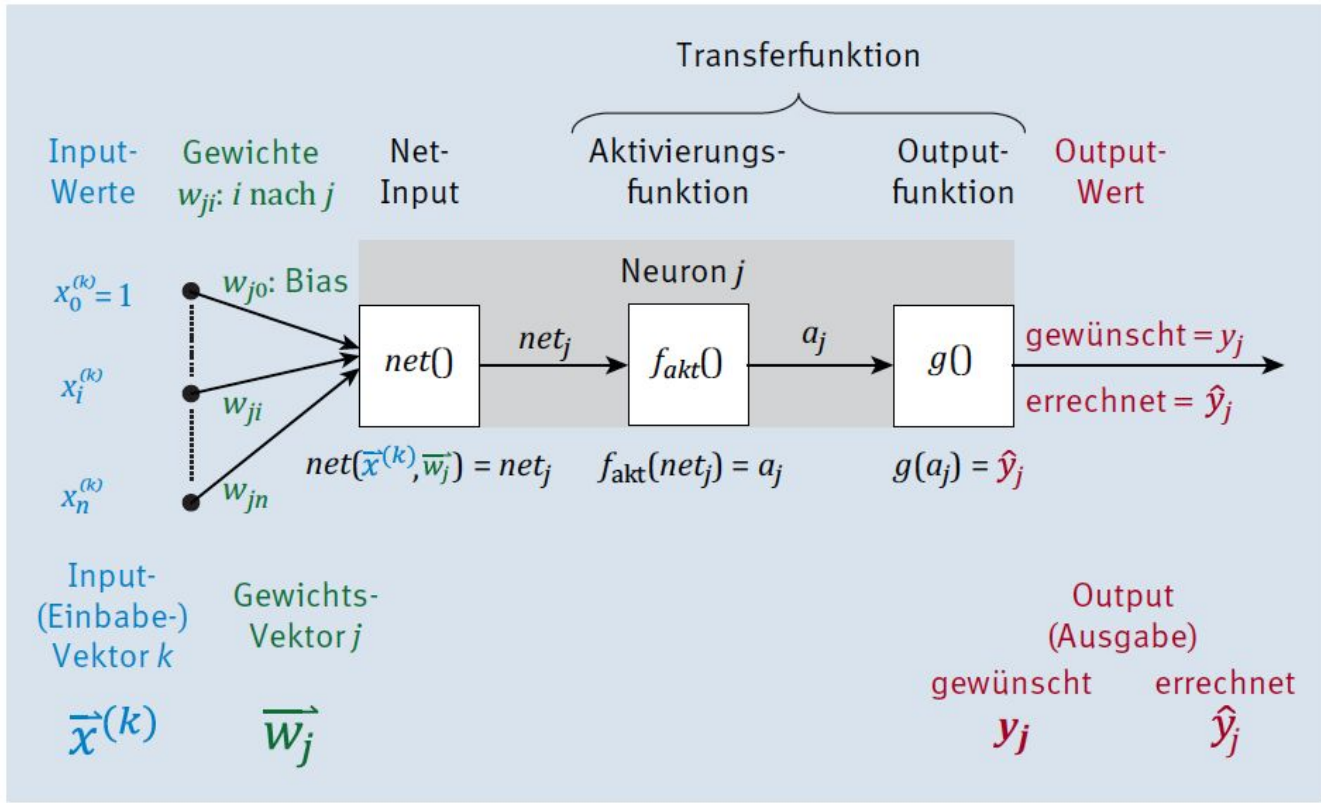


Abbildung 4.16 Verallgemeinertes Neuronenmodell mit Details aus diesem Kapitel

Rosenblatt:

$$\Delta w_i = \eta \cdot (y - \hat{y}) \cdot x_i \text{ , bzw.}$$

$$\Delta w_i = (y - \hat{y}) \cdot x_i \text{ , für } \eta = 1$$

Widrow-Hoff:

$$\Delta w_i = \eta \cdot (y - s) \cdot x_i$$

Multi Layer ANN

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

P1	P2	-->	E
0	0	-->	0
1	0	-->	1
0	1	-->	1
1	1	-->	**0**

Tabelle 5.1 Ein verstecktes XOR-Problem

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

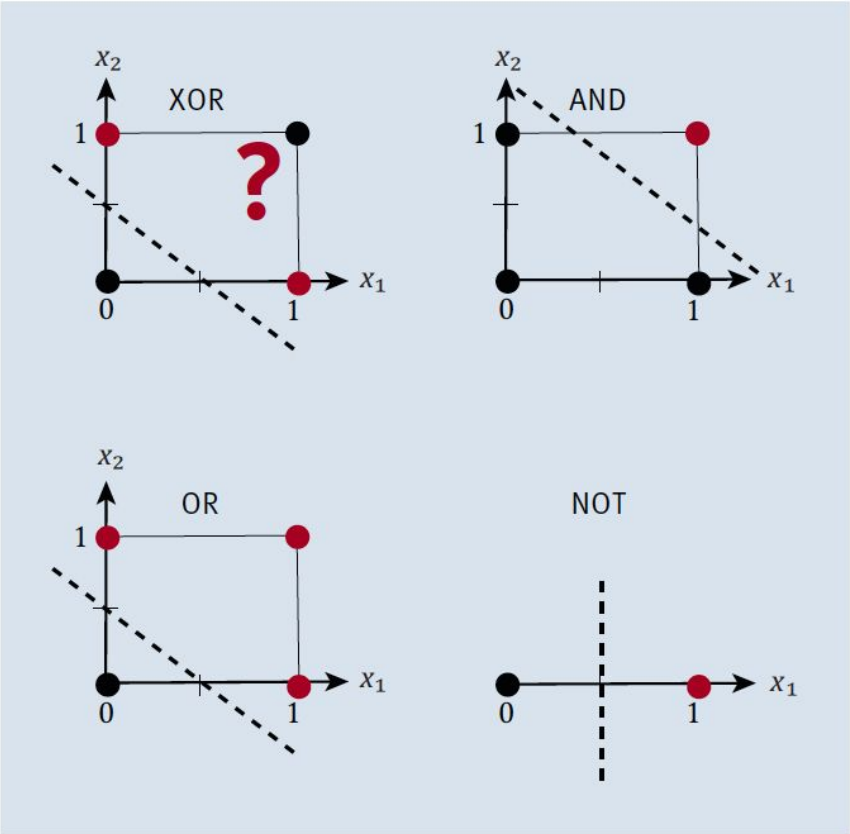


Abbildung 5.1 Wie soll man diese Planungsaufgabe lösen?

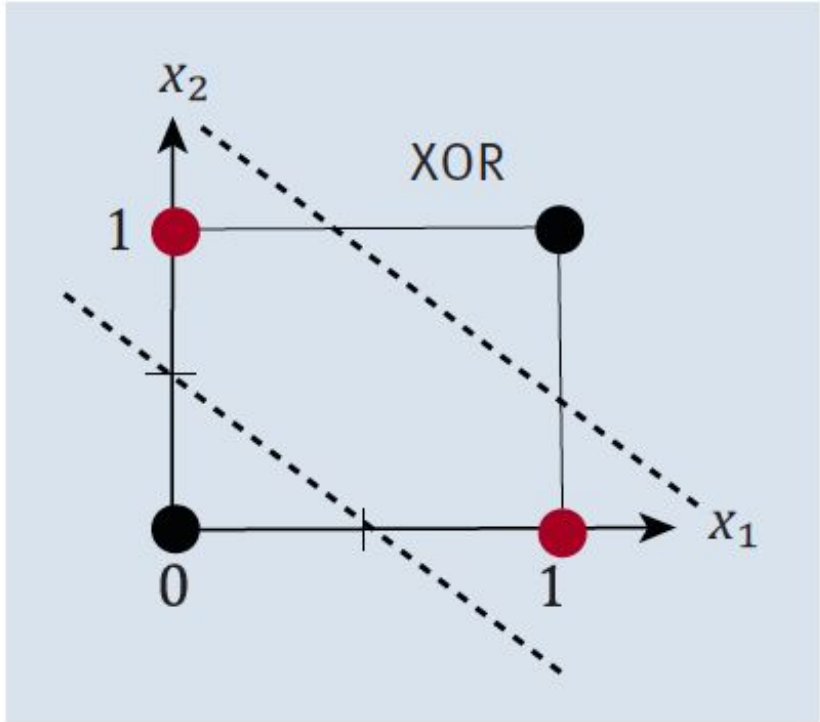


Abbildung 5.2 XOR gelöst

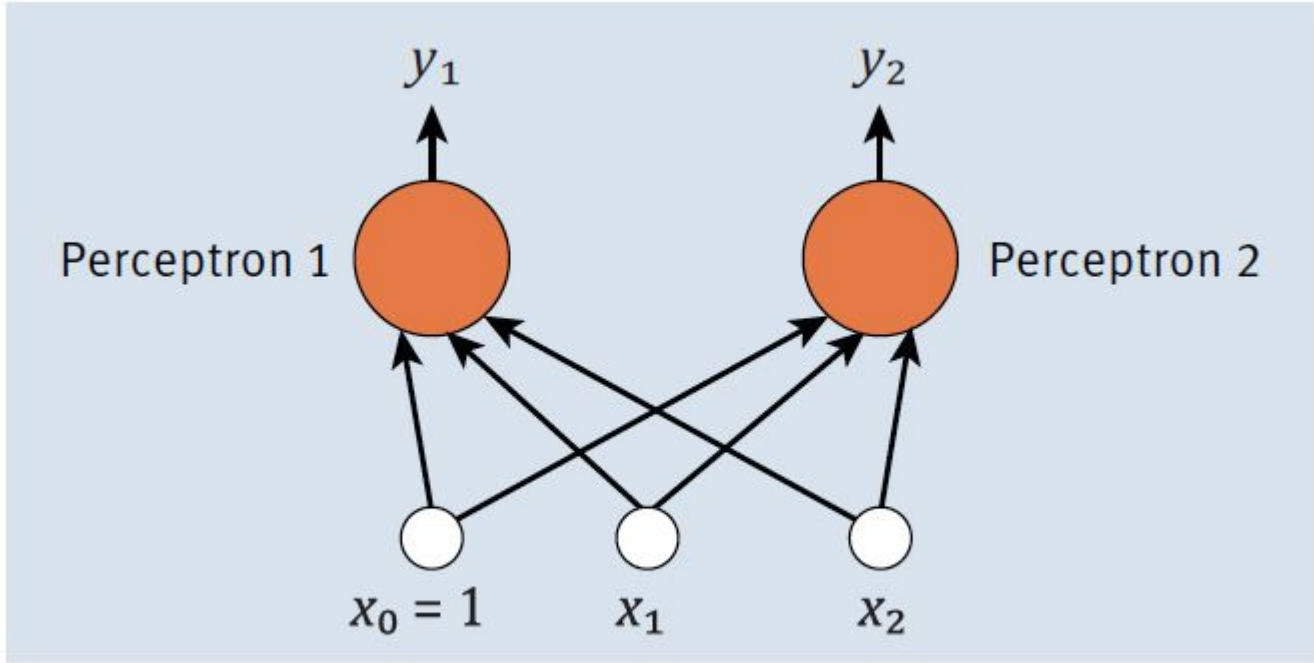


Abbildung 5.3 Zwei Perceptrons

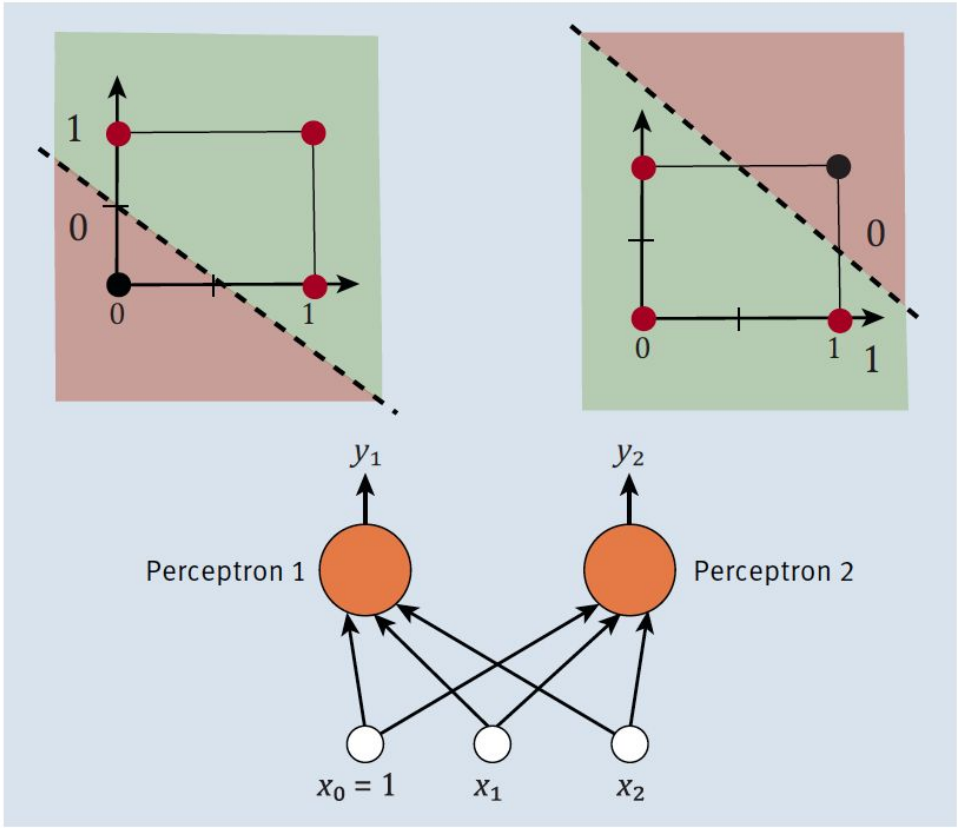


Abbildung 5.4 Lernaufgaben von Perceptron 1 und Perceptron 2

x_1	x_2	Perceptron 1 (y_1)	Perceptron 2 (y_2)
0	0	0	1
1	0	1	1
0	1	1	1
1	1	1	0

Tabelle 5.3 Perceptron 1 und Perceptron 2

x_1	x_2	Perceptron 1 (y_1)	Perceptron 2 (y_2)	Perceptron 3 ($\hat{y}$)
0	0	0	1	0
1	0	1	1	1
0	1	1	1	1
1	1	1	0	0

Tabelle 5.4 Perceptron 3 muss auch lernen.

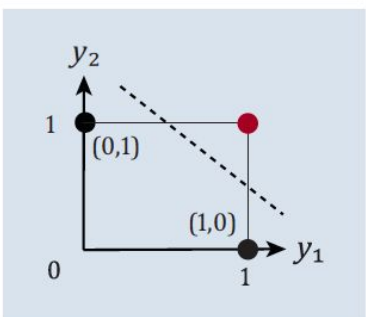


Abbildung 5.5 Perceptron 3 hat gelernt.

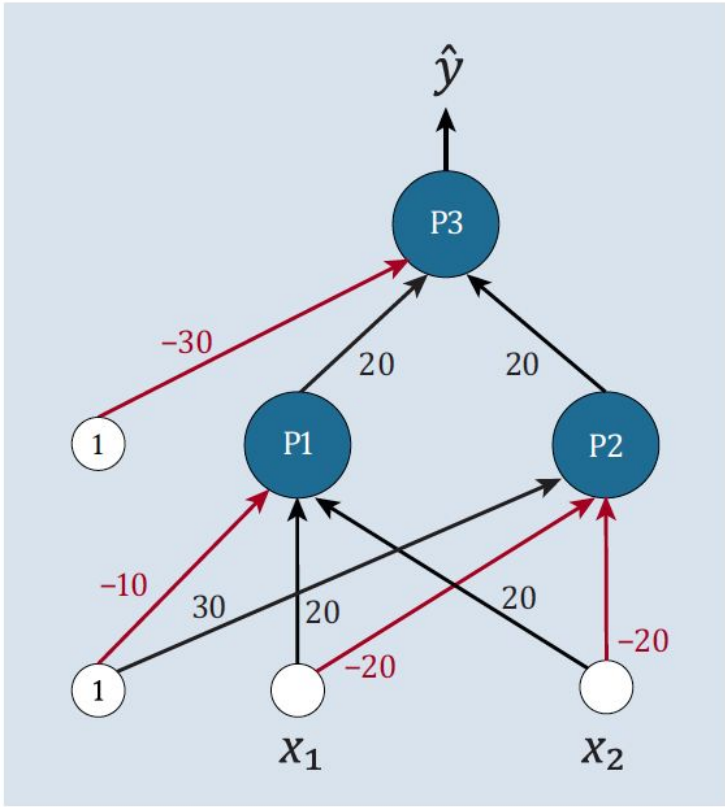


Abbildung 5.6 Das Netzwerk für das XOR-Problem

Input $(1, x_1, x_2)$	Output P1	Output P2	Output P3 $(\hat{y})$
(1,0,0)	$-10 \cdot 1 + 20 \cdot 0 + 20 \cdot 0 = -10$ $-> 0$	$30 \cdot 1 - 20 \cdot 0 - 20 \cdot 0 = 30$ $-> 1$	$-30 \cdot 1 + 20 \cdot 0 + 20 \cdot 1 = -10$ $-> 0$
(1,0,1)	$-10 \cdot 1 + 20 \cdot 0 + 20 \cdot 1 = 10$ $-> 1$	$30 \cdot 1 - 20 \cdot 0 - 20 \cdot 1 = 10$ $-> 1$	$-30 \cdot 1 + 20 \cdot 1 + 20 \cdot 1 = 10$ $-> 1$
(1,1,0)	$-10 \cdot 1 + 20 \cdot 1 + 20 \cdot 0 = 10$ $-> 1$	$30 \cdot 1 - 20 \cdot 1 - 20 \cdot 0 = 10$ $-> 1$	$-30 \cdot 1 + 20 \cdot 1 + 20 \cdot 1 = 10$ $-> 1$
(1,1,1)	$-10 \cdot 1 + 20 \cdot 1 + 20 \cdot 1 = 30$ $-> 1$	$30 \cdot 1 - 20 \cdot 1 - 20 \cdot 1 = 10$ $-> 0$	$-30 \cdot 1 + 20 \cdot 1 + 20 \cdot 0 = -10$ $-> 0$

Tabelle 5.5 Ermittlung der Outputs der unterschiedlichen Neuronen

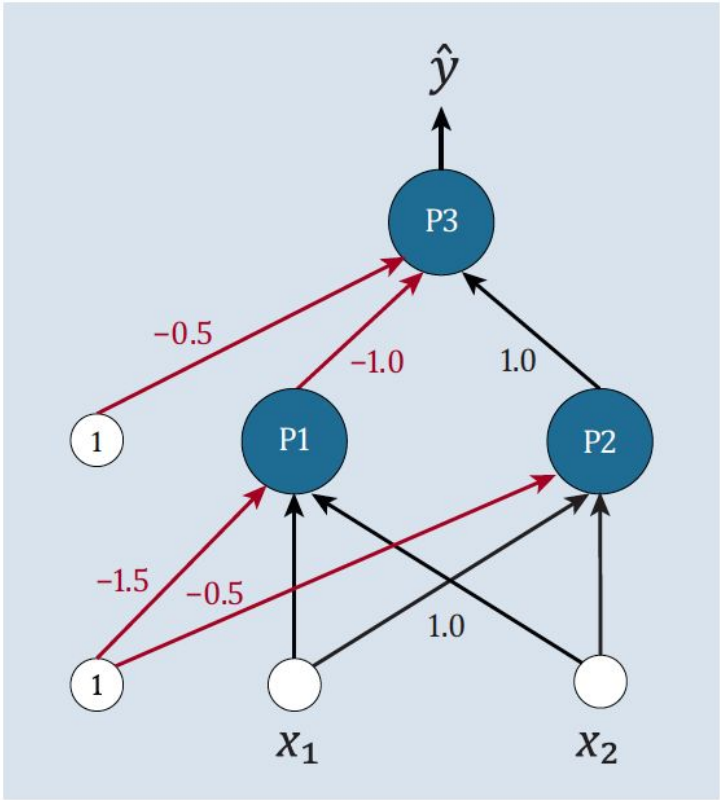


Abbildung 5.7 Lösungsvorschlag zu den Gewichten im MLP

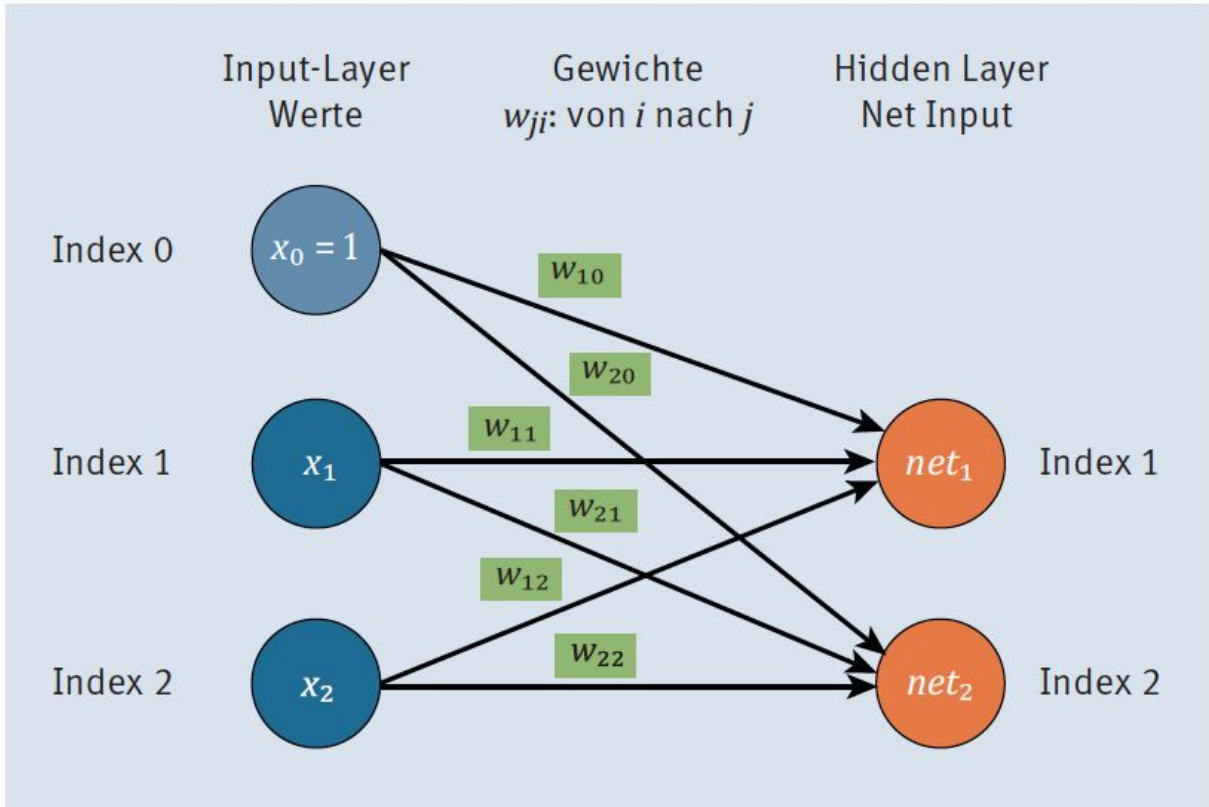


Abbildung 5.8 Die Berechnung im mehrschichtigen Netz

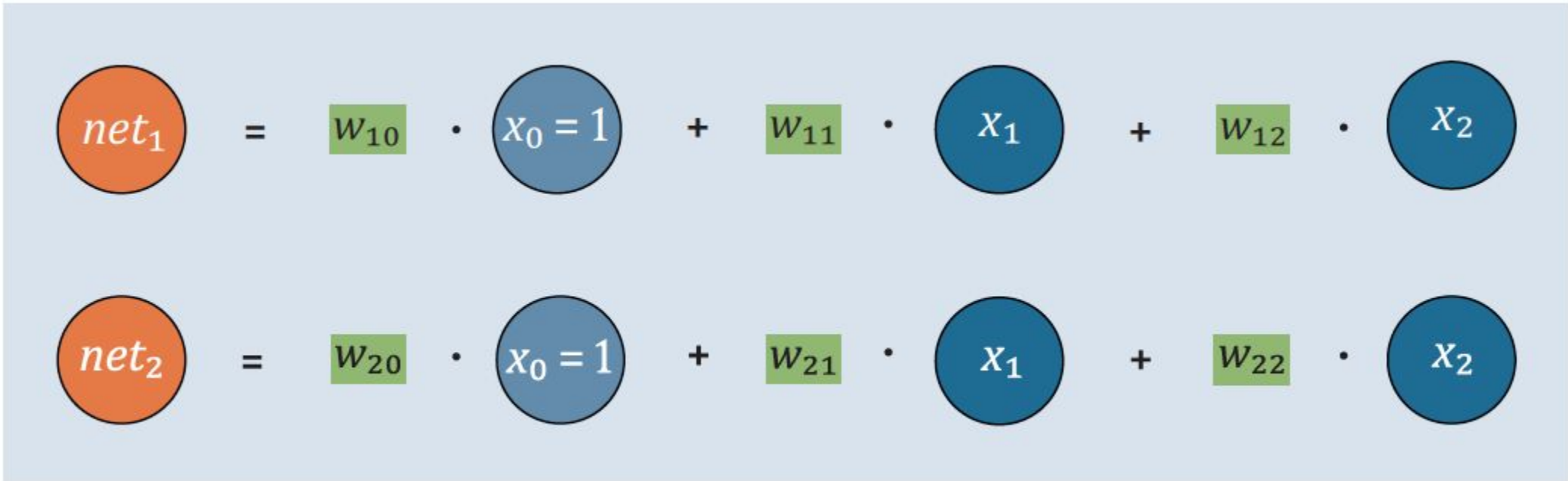


Abbildung 5.9 Auf dem Weg zur Matrixmultiplikation

Abbildung 5.10 Matrixmultiplikation, supergeeignet für »numpy«

I

x = o
$x_0 = 1$
$x_1 = o_1$
$x_2 = o_2$

W_IH

von / nach	0	1	2
0	dummy		
1	(1,0)	(1,1)	(1,2)
2	(2,0)	(2,1)	(2,2)

H

net	a	o
dummy		1
net_1	a_1	o_1
net_2	a_2	o_2

W_HO

von / nach	0	1	2
0	dummy		
1	(1,0)	(1,1)	(1,2)

O

net	a	o
dummy		
net_1	a_1	o_1

Abbildung 5.11 Die Bausteine des Netzwerkes

5.1,2,3,4,5,6 Adaline-Estimator

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

```
# Ausgabe:
Multi-Layer Perceptron - Netzwerkarchitektur
[[  1.000]
 [  0.000]
 [  0.000]]
----------v----------
[[  0.000   0.000   0.000]
 [-10.000  20.000  20.000]
 [ 30.000 -20.000 -20.000]]
----------v----------
[[  1.000   1.000   1.000]
 [  0.000   0.000   0.000]
 [  0.000   0.000   0.000]]
----------v----------
[[  0.000   0.000   0.000]
 [-30.000  20.000  20.000]]
----------v----------
[[  0.000   0.000   0.000]
 [  0.000   0.000   0.000]]
----------v----------
Predict:
[  1.000   1.000   1.000] 0.0 -> [  0.000]
[  1.000   0.000   1.000] 1.0 -> [  1.000]
[  1.000   1.000   0.000] 1.0 -> [  1.000]
[  1.000   0.000   0.000] 0.0 -> [  0.000]
```

Listing 5.7 Ausgabe der Architektur mit Gewichten und der Vorhersage

Learning im MLP

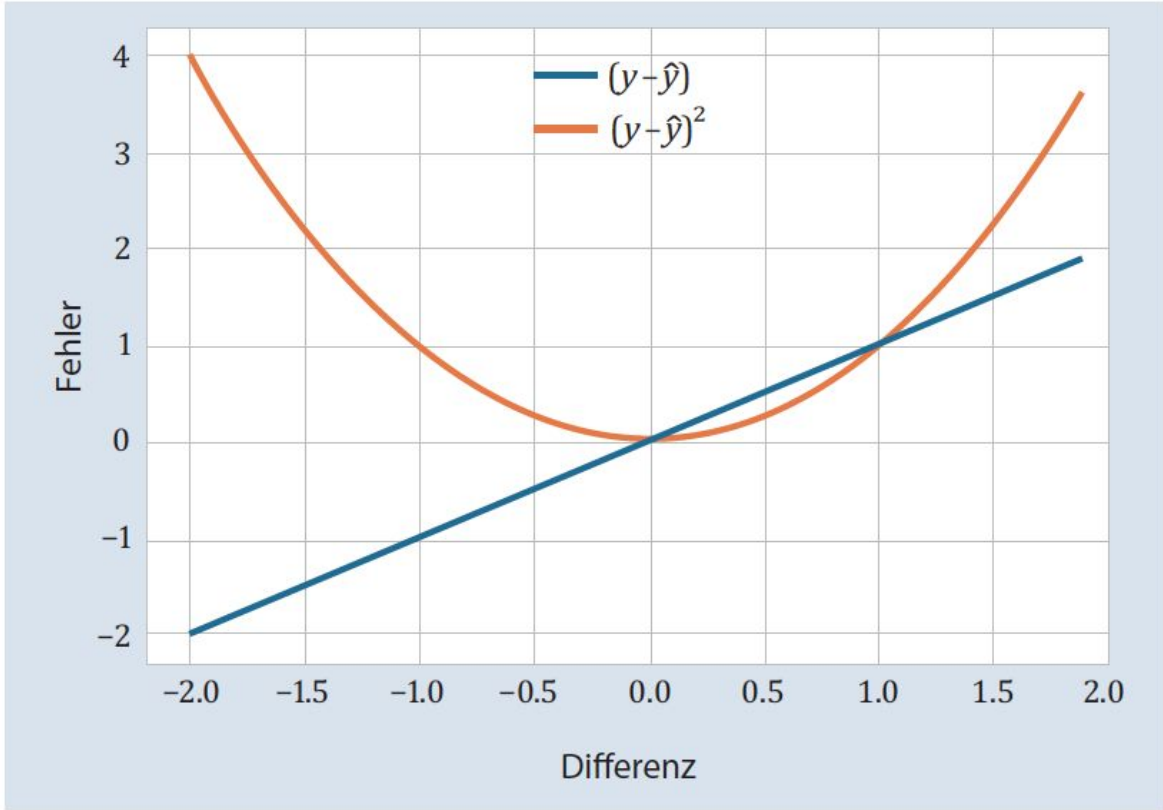


Abbildung 6.1 Fehlerkurven

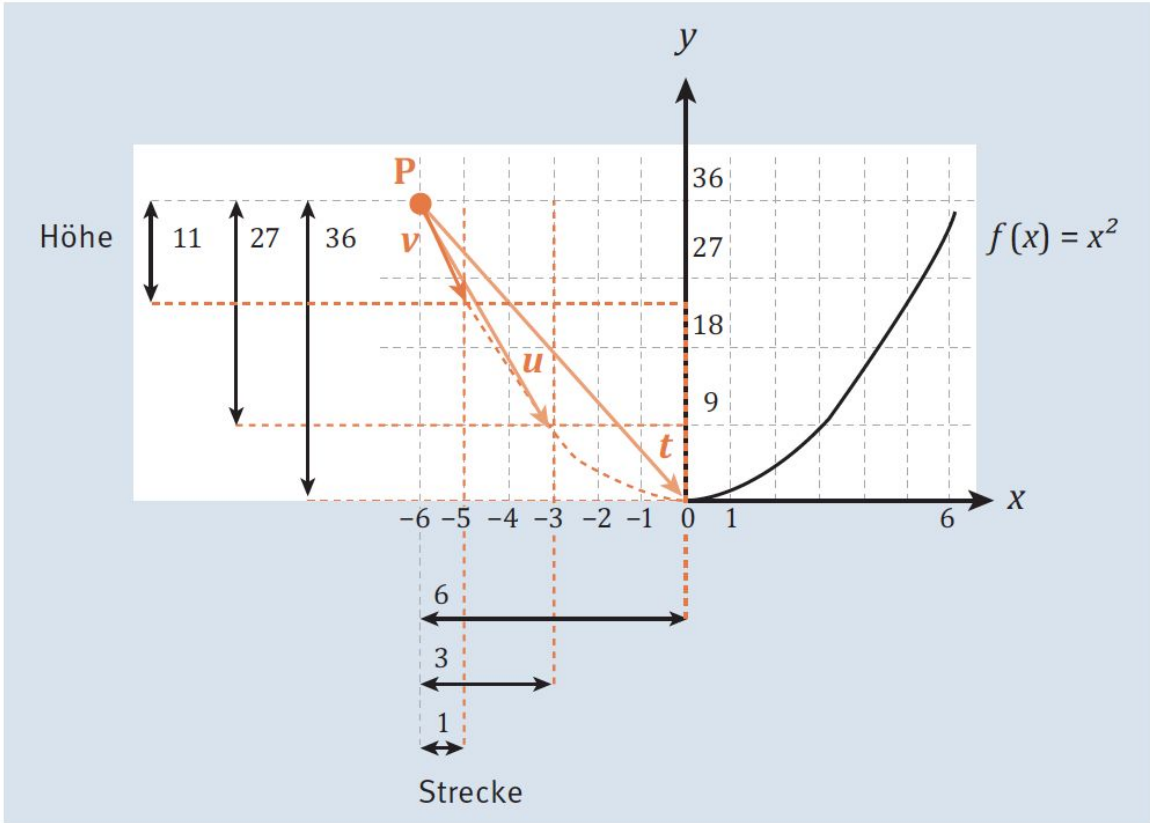


Abbildung 6.2 Der Gradient

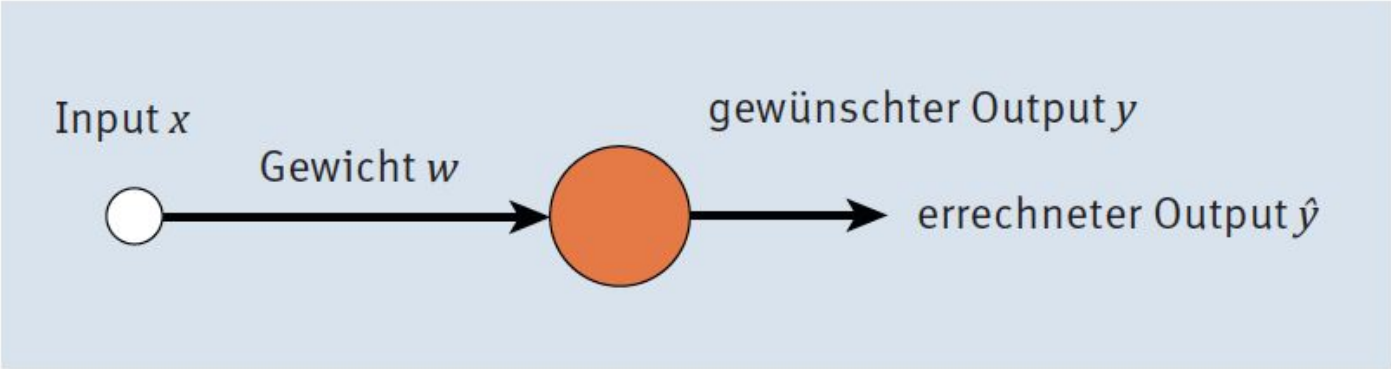


Abbildung 6.3 Einfaches KNN

$$E = \frac{1}{2} \cdot (y - \hat{y})^2 = \frac{1}{2} \cdot (y - w \cdot x)^2$$

lautet der Gradient:

$$\nabla E(w) = \frac{1}{2} \cdot 2 \cdot (y - w \cdot x) \cdot (-1) \cdot x = (-1) \cdot x \cdot (y - \hat{y})$$

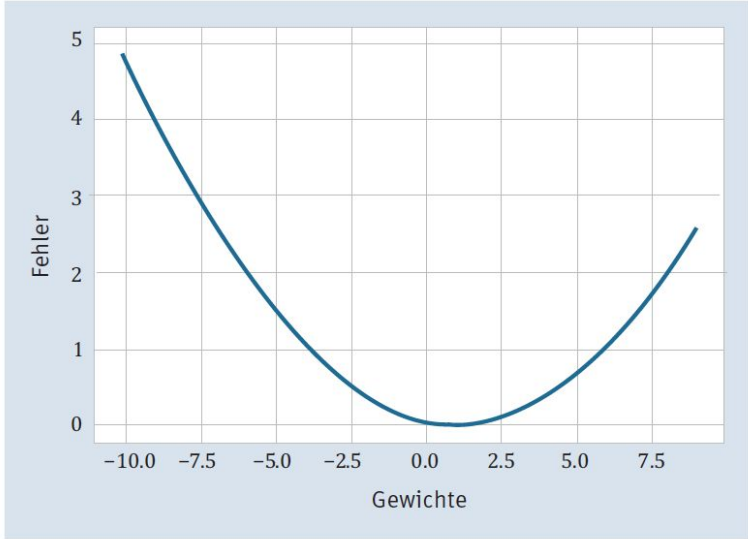


Abbildung 6.4 Fehler je nach Gewicht

6.1 Gradient descent

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Iteration	x	w	Net Input	a	y_hat	y	E	E'	w delta
0	0.2	−10.00	−2.00	−2.00	−2.00	0.20	2.42	−0.44	0.44
10	0.2	−6.31	−1.26	−1.26	−1.26	0.20	1.07	−0.29	0.29
20	0.2	−3.86	−0.77	−0.77	−0.77	0.20	0.47	−0.19	0.19
30	0.2	−2.23	−0.45	−0.45	−0.45	0.20	0.21	−0.13	0.13
40	0.2	−1.15	−0.23	−0.23	−0.23	0.20	0.09	−0.09	0.09
50	0.2	−0.43	−0.09	−0.09	−0.09	0.20	0.04	−0.06	0.06
60	0.2	0.05	0.01	0.01	0.01	0.20	0.02	−0.04	0.04
70	0.2	0.37	0.07	0.07	0.07	0.20	0.01	−0.03	0.03
80	0.2	0.58	0.12	0.12	0.12	0.20	0.00	−0.02	0.02
90	0.2	0.72	0.14	0.14	0.14	0.20	0.00	−0.01	0.01
100	0.2	0.81	0.16	0.16	0.16	0.20	0.00	−0.01	0.01
110	0.2	0.88	0.18	0.18	0.18	0.20	0.00	−0.00	0.00
120	0.2	0.92	0.18	0.18	0.18	0.20	0.00	-0.00	0.00

Tabelle 6.2 Gradientenabstieg für das einfache KNN

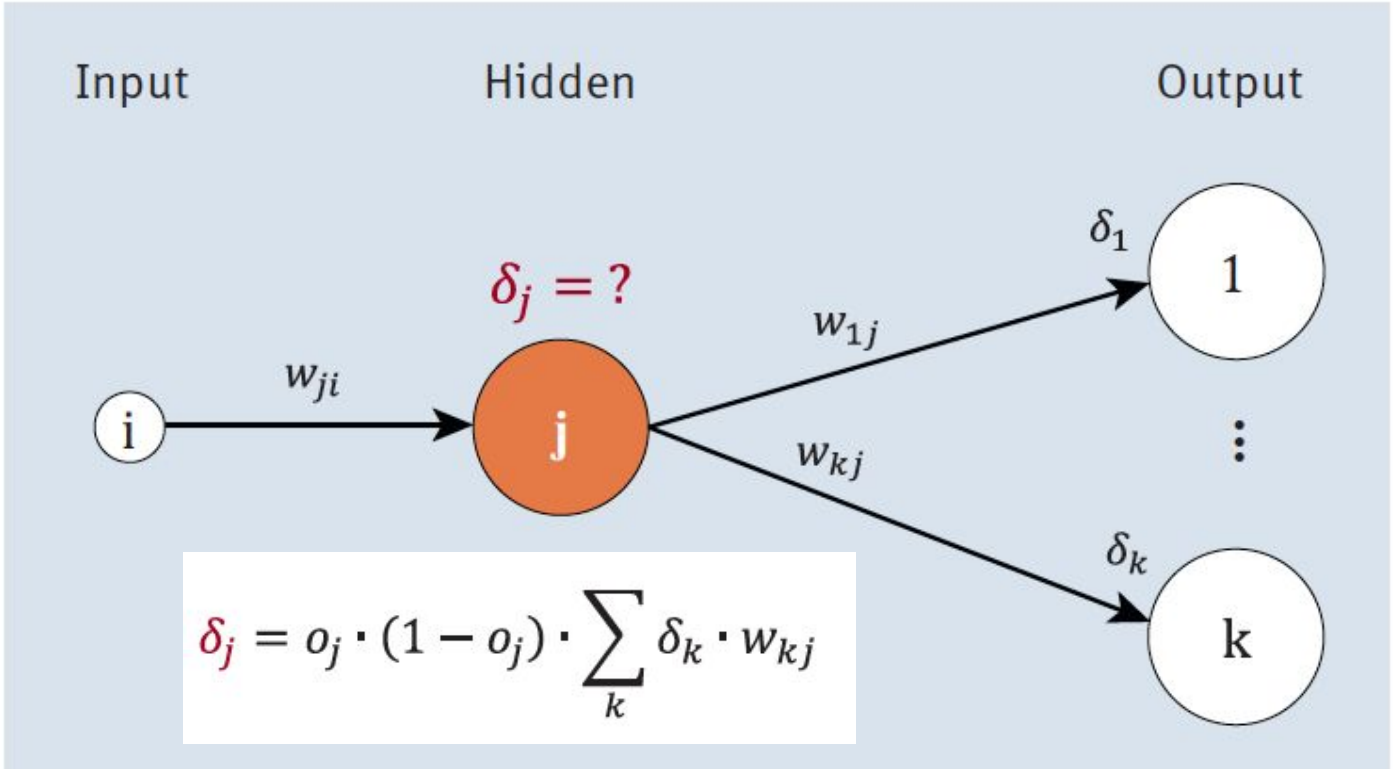


Abbildung 6.7 Vom Output zur versteckten Schicht

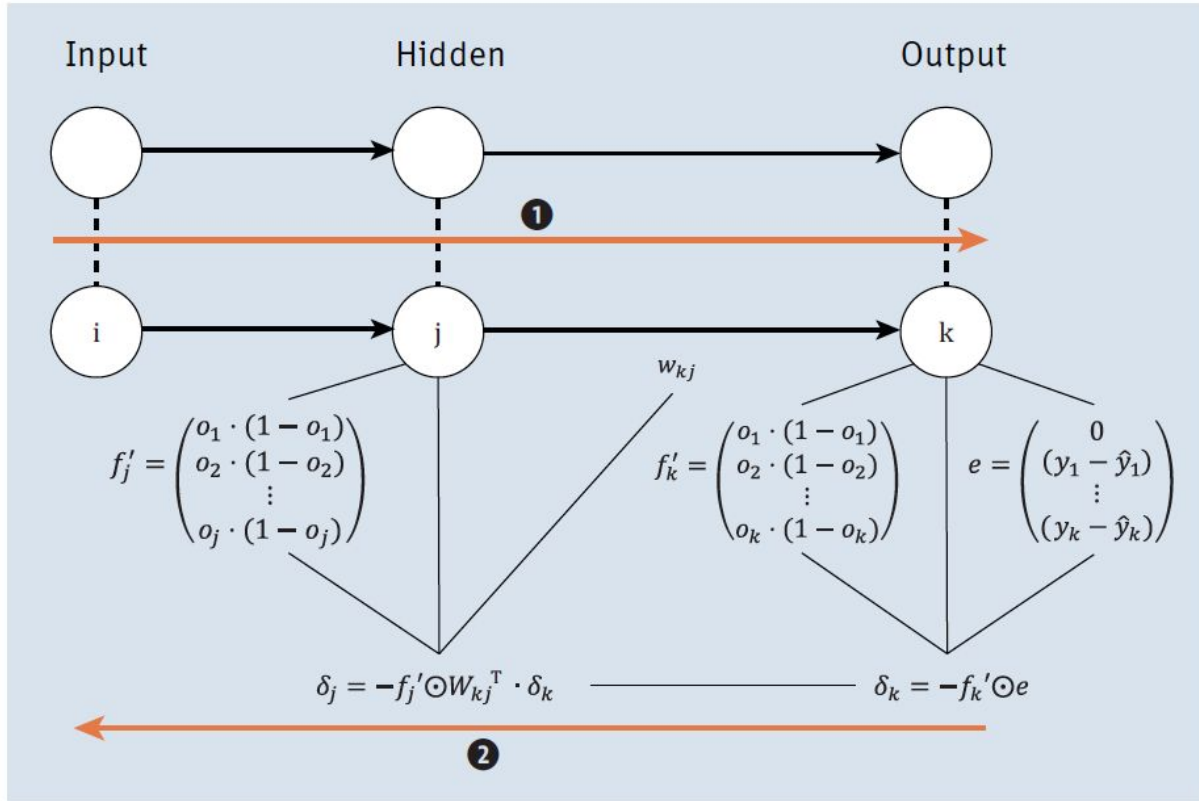


Abbildung 6.8 Vektor- und Matrixmultiplikationen für die Programmierung

6.2,3,4,5,6,7 BackProp

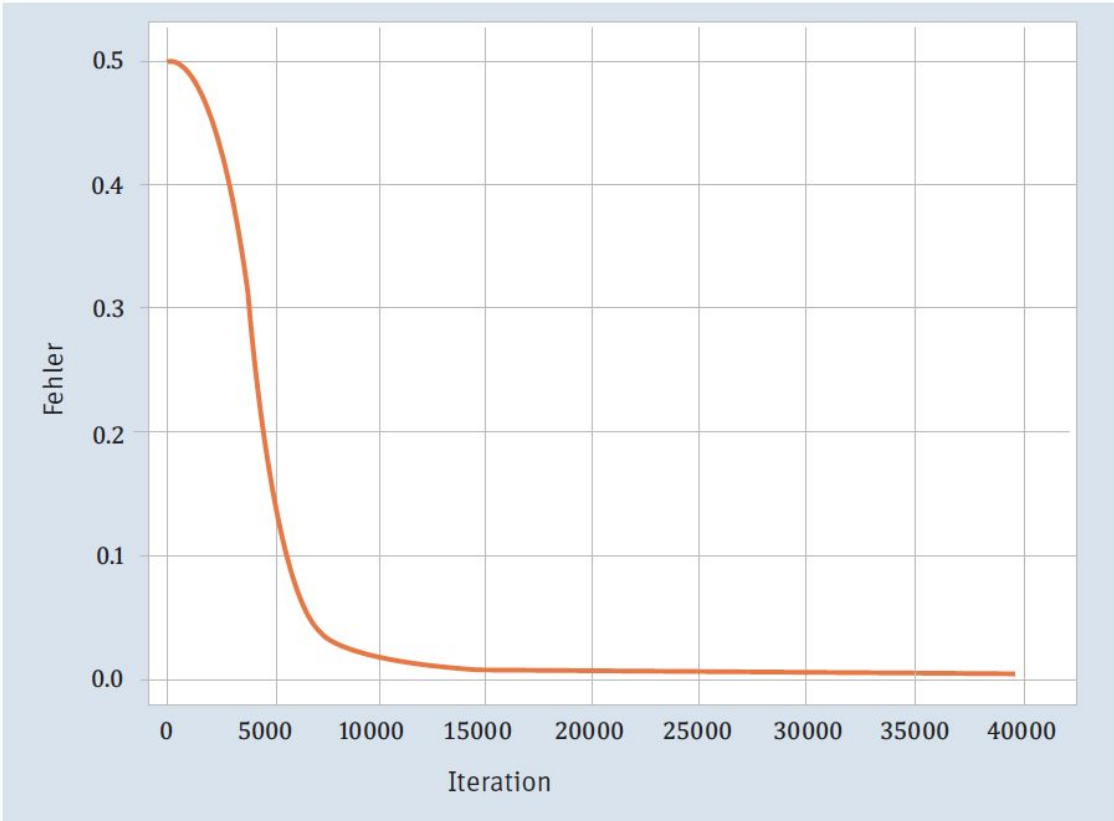
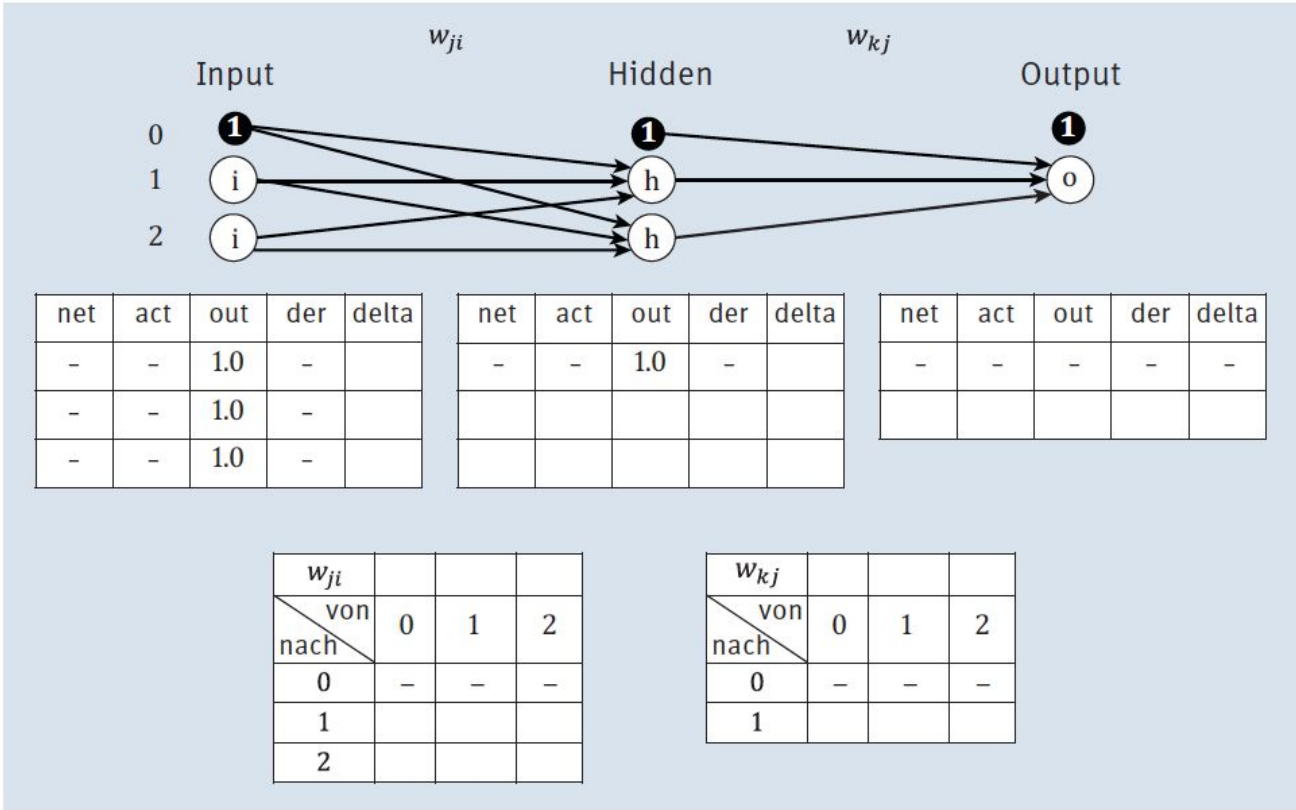


Abbildung 6.11 Was für eine schöne Fehlerkurve, wie aus dem Bilderbuch!

```
 # Ausgabe:
 Multi-Layer-Perceptron - Netzwerkarchitektur
 [[  1.000   1.000   1.000   1.000   1.000]

 [  0.000   0.000   0.000   0.000   0.000]
 [  0.000   0.000   0.000   0.000   0.000]]
----------v----------
[[  7.495  -5.248  -6.321]
 [  2.108  -5.657  -5.653]
 [ -6.578   4.224   4.223]]
----------v----------
[[  1.000   1.000   1.000   1.000  -0.005]
 [  2.108   0.892   0.892   0.097   0.001]
 [ -6.578   0.001   0.001   0.001   0.000]]
----------v----------
[[  0.416  -0.959   0.940]
 [  4.037  -8.164  -8.261]]
----------v----------
[[  0.000   0.000   0.000   0.000   0.000]
 [ -3.253   0.037   0.037   0.036  -0.001]]
----------v----------
Predict:
[  1.000   1.000   1.000] 0.0 -> [  0.042]
[  1.000   0.000   1.000] 1.0 -> [  0.957]
[  1.000   1.000   0.000] 1.0 -> [  0.957]
[  1.000   0.000   0.000] 0.0 -> [  0.037]
```

Listing 6.8 Output des Programms



net	act	out	der	delta
–	–	1.0	–	
–	–	1.0	–	
–	–	1.0	–	

net	act	out	der	delta
–	–	1.0	–	

net	act	out	der	delta
–	–	–	–	–

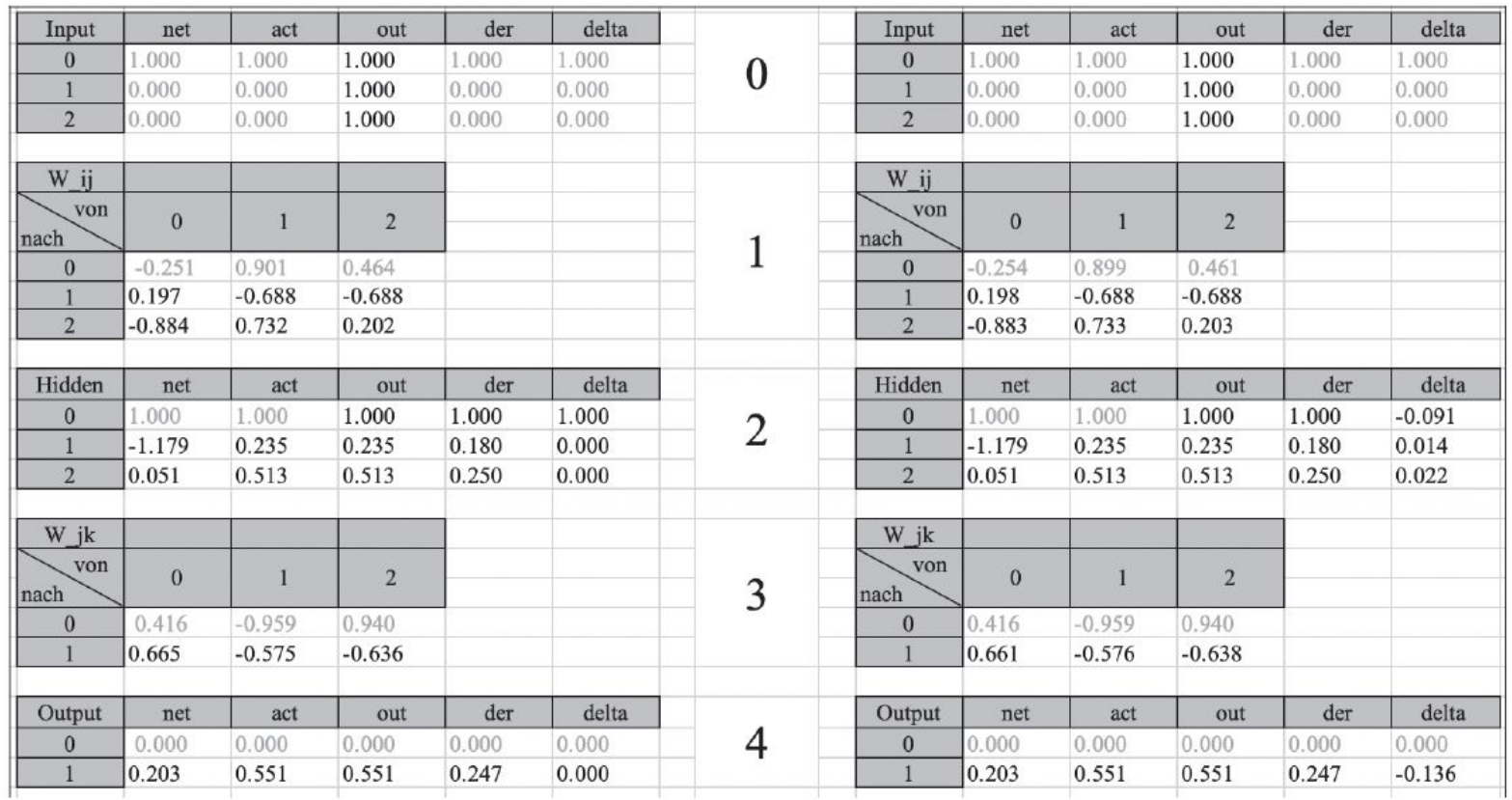
w_{ji}			
von / nach	0	1	2
0	–	–	–
1			
2			

w_{kj}			
von / nach	0	1	2
0	–	–	–
1			

Abbildung 6.12 Netzaufbau für die Iteration

6.9 Learnign Step

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127



Input	net	act	out	der	delta
0	1.000	1.000	1.000	1.000	1.000
1	0.000	0.000	1.000	0.000	0.000
2	0.000	0.000	1.000	0.000	0.000

0

W_ij			
von / nach	0	1	2
0	-0.251	0.901	0.464
1	0.197	-0.688	-0.688
2	-0.884	0.732	0.202

1

Hidden	net	act	out	der	delta
0	1.000	1.000	1.000	1.000	1.000
1	-1.179	0.235	0.235	0.180	0.000
2	0.051	0.513	0.513	0.250	0.000

2

W_jk			
von / nach	0	1	2
0	0.416	-0.959	0.940
1	0.665	-0.575	-0.636

3

Output	net	act	out	der	delta
0	0.000	0.000	0.000	0.000	0.000
1	0.203	0.551	0.551	0.247	0.000

4

Input	net	act	out	der	delta
0	1.000	1.000	1.000	1.000	1.000
1	0.000	0.000	1.000	0.000	0.000
2	0.000	0.000	1.000	0.000	0.000

W_ij			
von / nach	0	1	2
0	-0.254	0.899	0.461
1	0.198	-0.688	-0.688
2	-0.883	0.733	0.203

Hidden	net	act	out	der	delta
0	1.000	1.000	1.000	1.000	-0.091
1	-1.179	0.235	0.235	0.180	0.014
2	0.051	0.513	0.513	0.250	0.022

W_jk			
von / nach	0	1	2
0	0.416	-0.959	0.940
1	0.661	-0.576	-0.638

Output	net	act	out	der	delta
0	0.000	0.000	0.000	0.000	0.000
1	0.203	0.551	0.551	0.247	-0.136

Abbildung 6.13 Von alt zu neu, ein Iterationsschritt

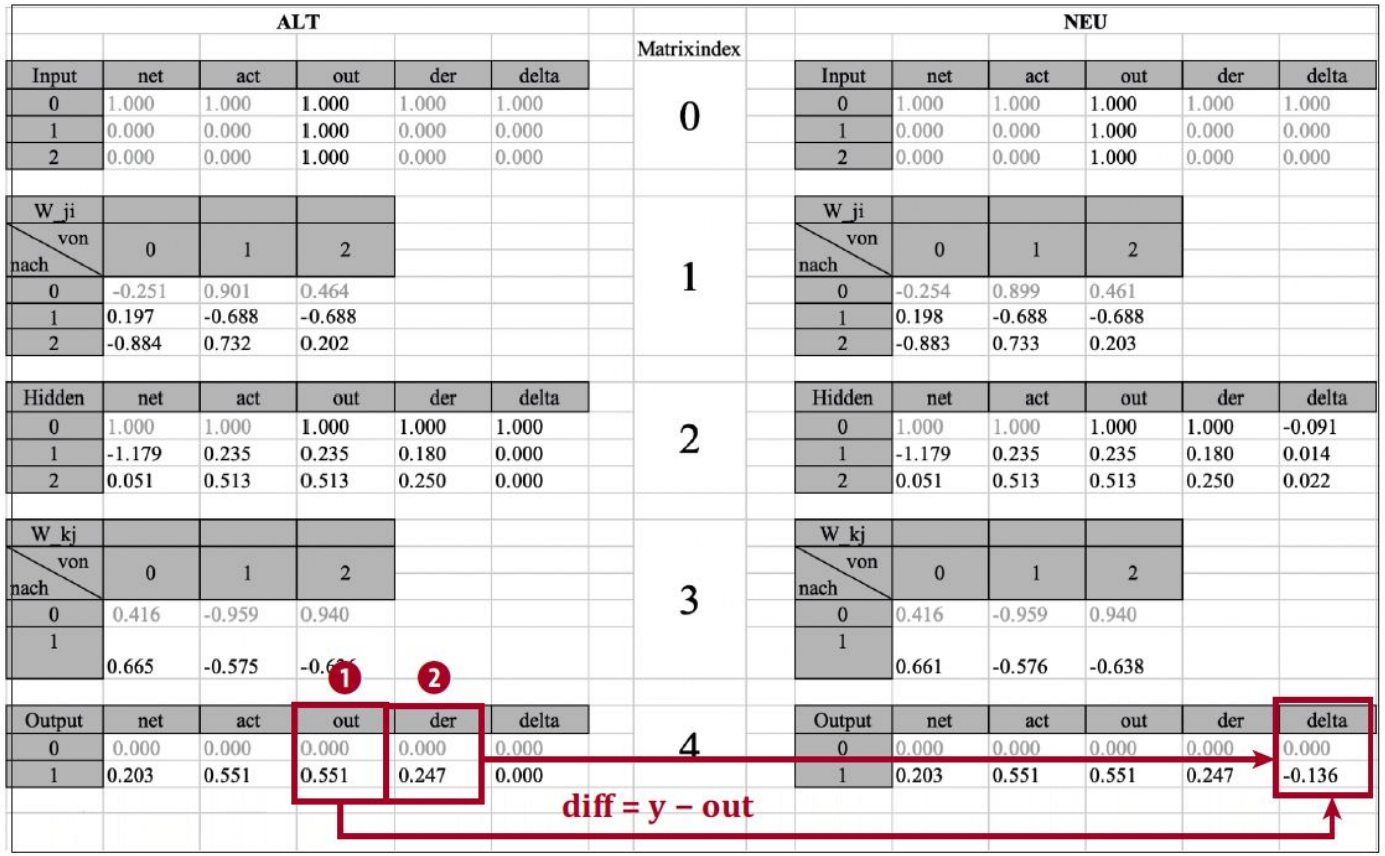


Abbildung 6.14 Berechnung der Output-Schicht

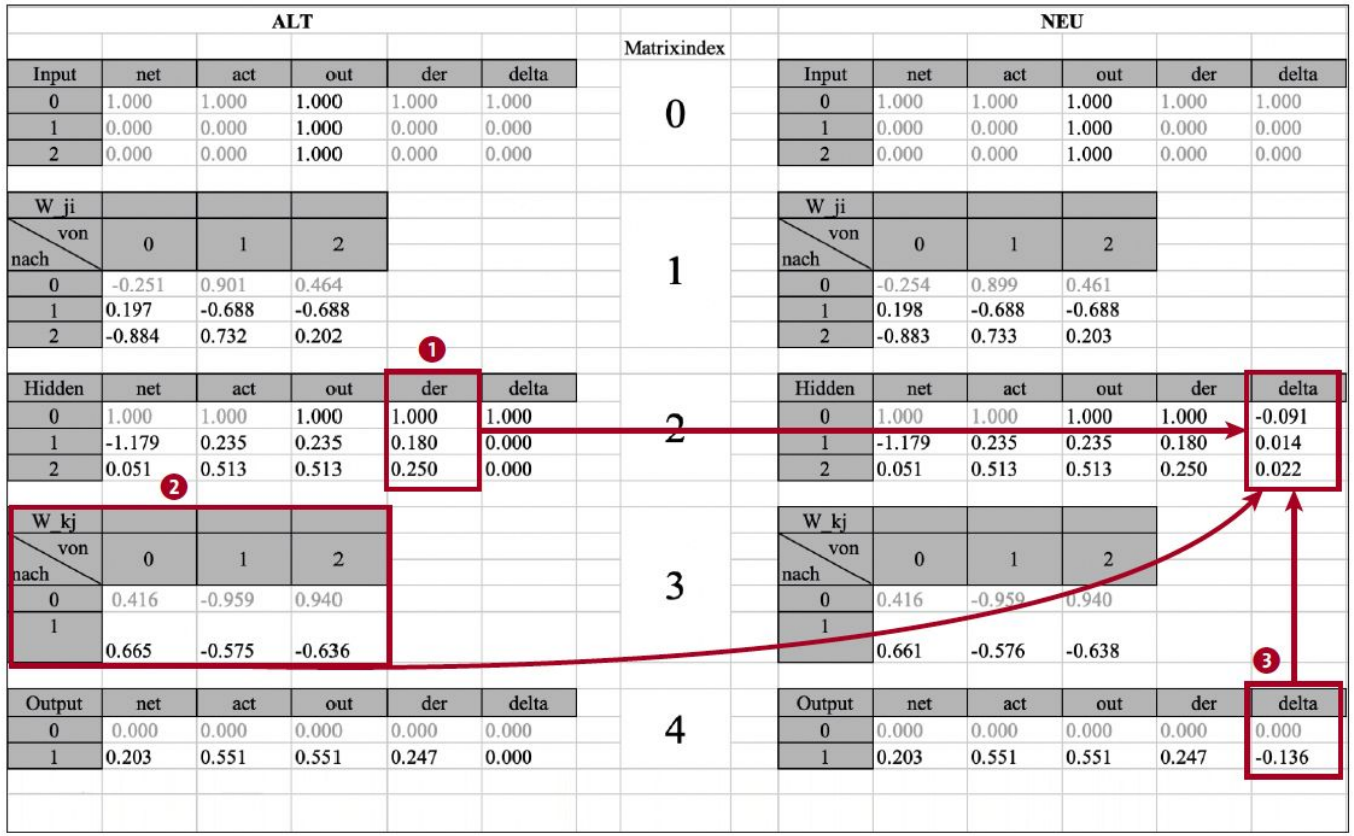
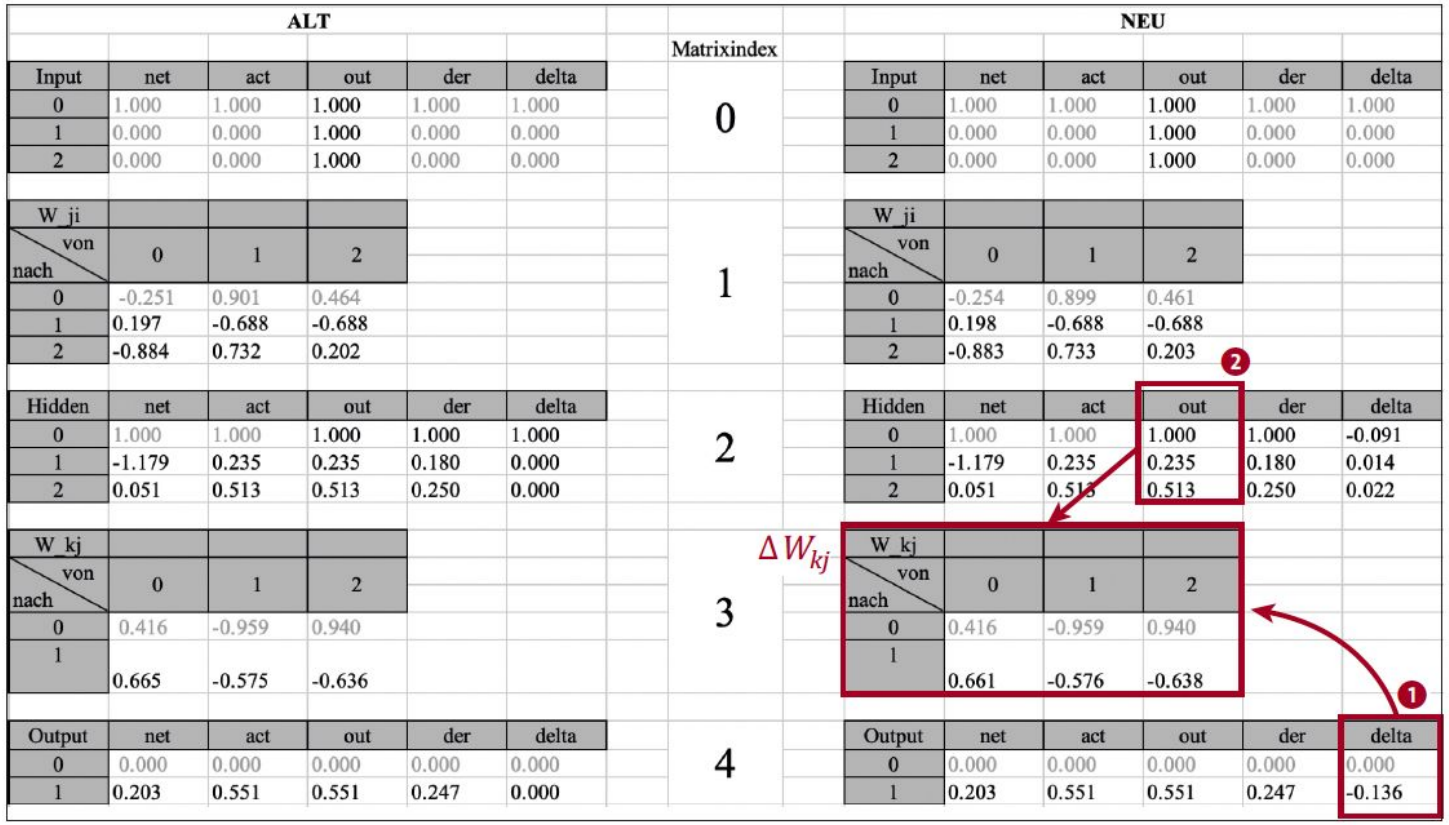


Abbildung 6.15 Delta-Berechnung in der Hidden-Schicht

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127



ALT

Matrixindex: 0

Input	net	act	out	der	delta
0	1.000	1.000	1.000	1.000	1.000
1	0.000	0.000	1.000	0.000	0.000
2	0.000	0.000	1.000	0.000	0.000

Matrixindex: 1

W_ji			
nach \ von	0	1	2
0	-0.251	0.901	0.464
1	0.197	-0.688	-0.688
2	-0.884	0.732	0.202

Matrixindex: 2

Hidden	net	act	out	der	delta
0	1.000	1.000	1.000	1.000	1.000
1	-1.179	0.235	0.235	0.180	0.000
2	0.051	0.513	0.513	0.250	0.000

Matrixindex: 3

W_kj			
nach \ von	0	1	2
0	0.416	-0.959	0.940
1	0.665	-0.575	-0.636

Matrixindex: 4

Output	net	act	out	der	delta
0	0.000	0.000	0.000	0.000	0.000
1	0.203	0.551	0.551	0.247	0.000

NEU

Input	net	act	out	der	delta
0	1.000	1.000	1.000	1.000	1.000
1	0.000	0.000	1.000	0.000	0.000
2	0.000	0.000	1.000	0.000	0.000

W_ji			
nach \ von	0	1	2
0	-0.254	0.899	0.461
1	0.198	-0.688	-0.688
2	-0.883	0.733	0.203

Hidden	net	act	out	der	delta
0	1.000	1.000	1.000	1.000	-0.091
1	-1.179	0.235	0.235	0.180	0.014
2	0.051	0.513	0.513	0.250	0.022

ΔW_{kj}

W_kj			
nach \ von	0	1	2
0	0.416	-0.959	0.940
1	0.661	-0.576	-0.638

Output	net	act	out	der	delta
0	0.000	0.000	0.000	0.000	0.000
1	0.203	0.551	0.551	0.247	-0.136

Abbildung 6.16 Berechnung des Gewichtsdeltas für Wkj

Convolutional Neural Networks

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

7.1 Aufbau eines CNN

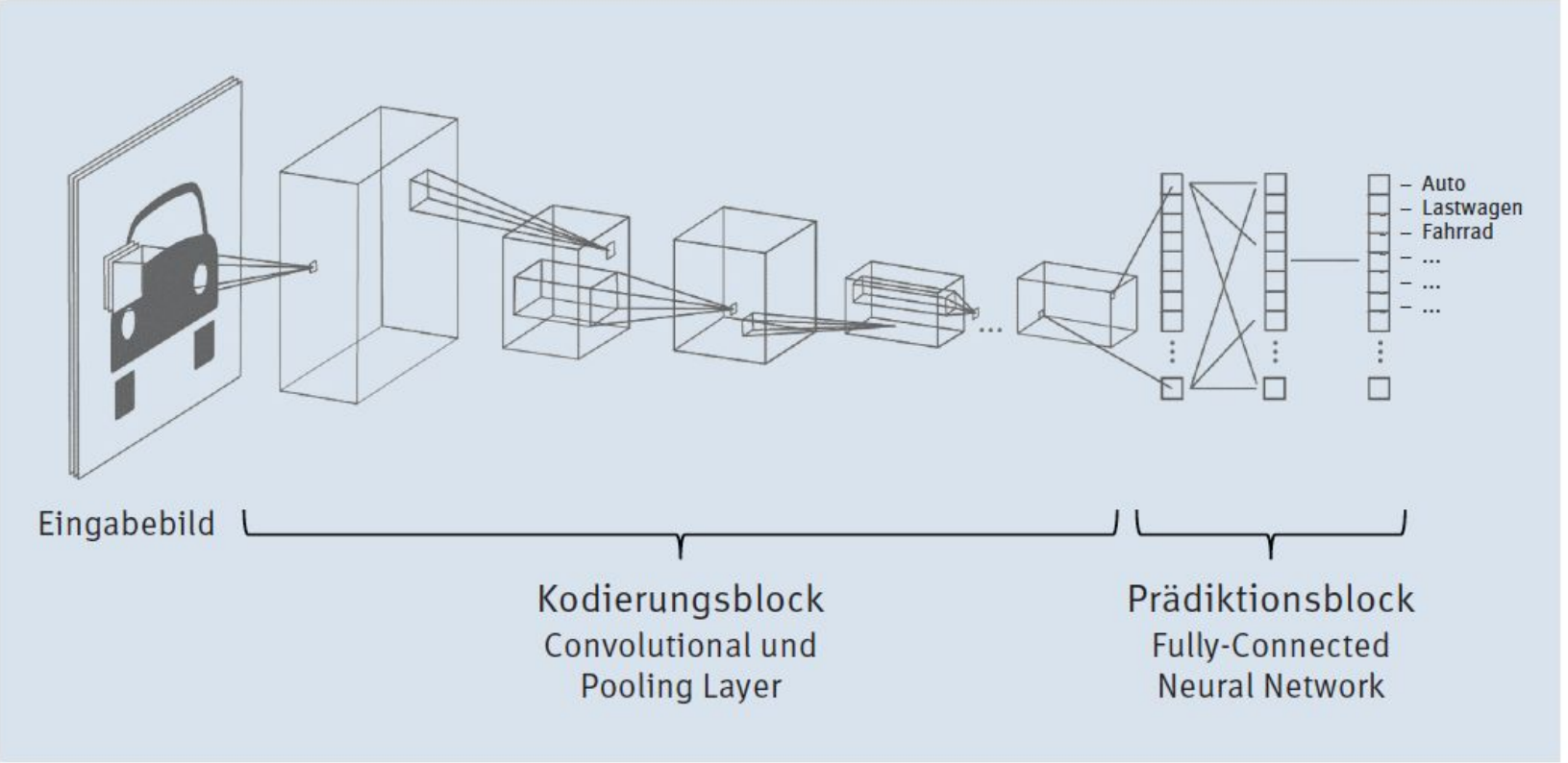


Abbildung 7.2 Die Struktur eines CNN mit Kodierungs- und Prädiktionsblock

Convolution?

Was bedeutet *Convolution*? Die deutsche Übersetzung wäre der mathematische Begriff der Konvolution oder Faltung, in unserem Fall genauer die diskrete Faltung. Mathematisch kann die Faltung als Produkt von zwei Funktionen verstanden werden. In der Bildverarbeitung bedeutet die diskrete Faltung das »Filtern« eines Bildes mit einer 3×3- oder 5×5-Matrix (siehe Abbildung 7.3), wobei es unterschiedliche Filtertypen gibt (Linienfilter, Kantenfilter, Weichzeichner etc.). Beim Convolutional Neural Network passiert in den ersten Ebenen genau das – das Eingangsbild wird »gefiltert«, um gewisse Merkmale zu betonen (Linien, Kanten, Punkte, Ecken etc.).

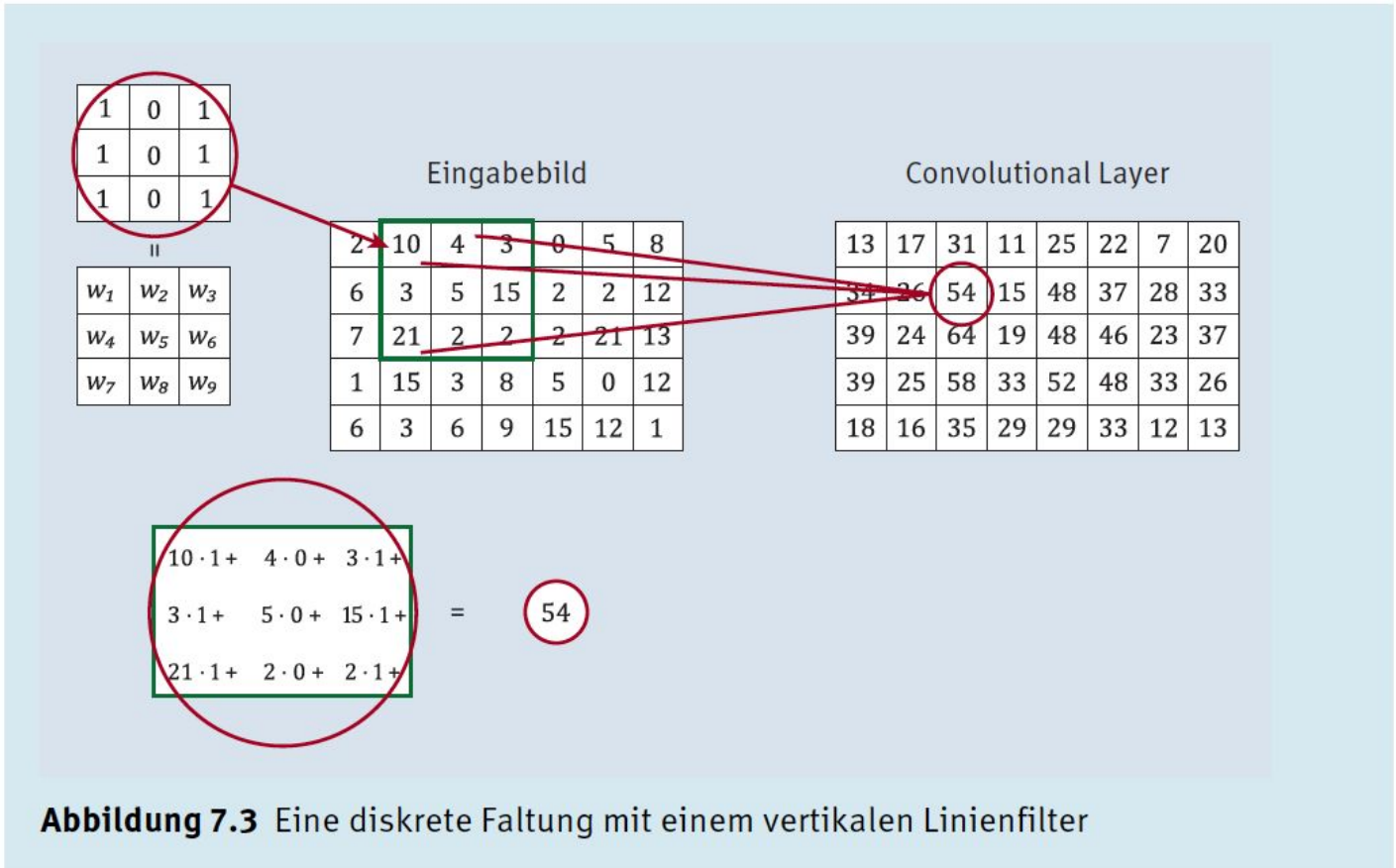


Abbildung 7.3 Eine diskrete Faltung mit einem vertikalen Linienfilter

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

7.2 Der Kodierungsblock
7.2.1 Convolutional Layer

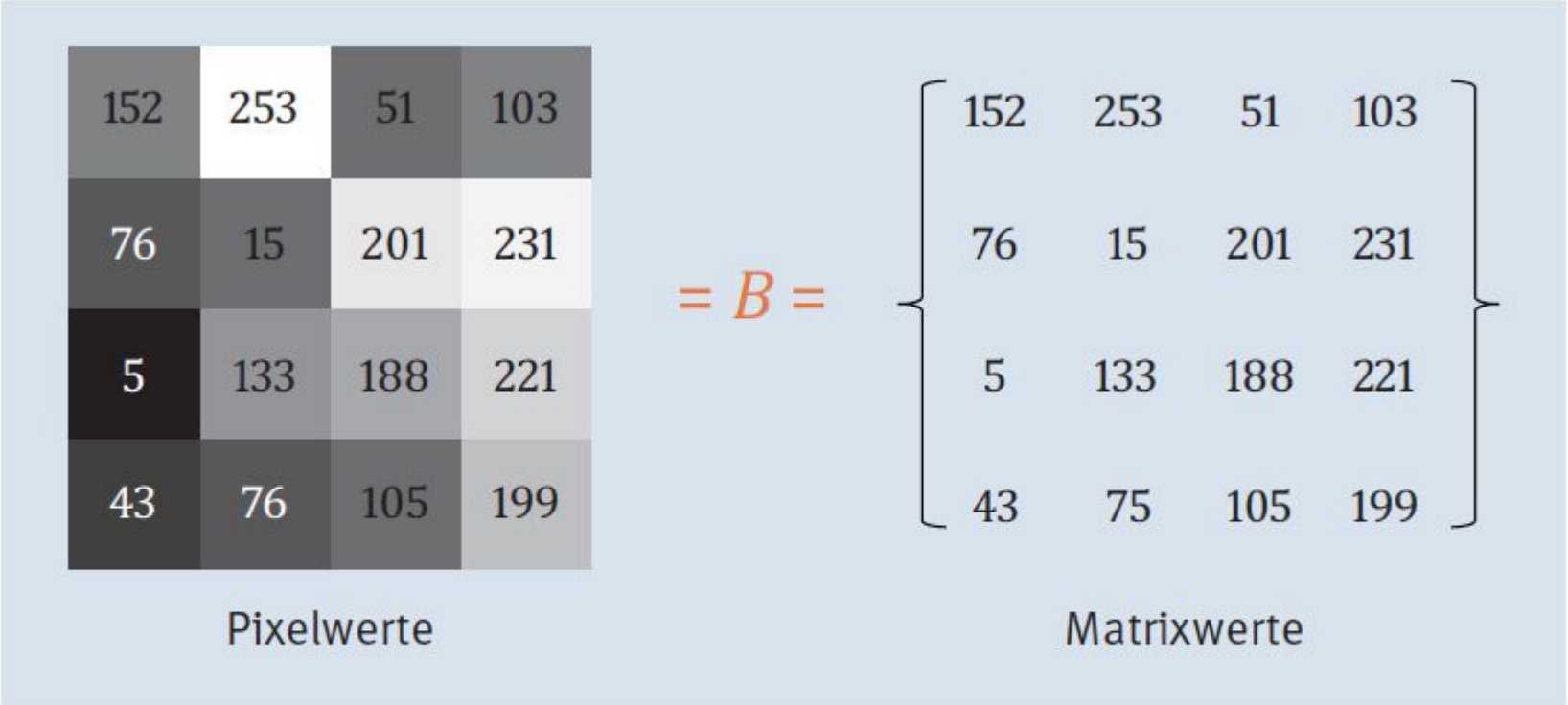


Abbildung 7.4 Vom Bild (Pixelwerte) zur Matrix

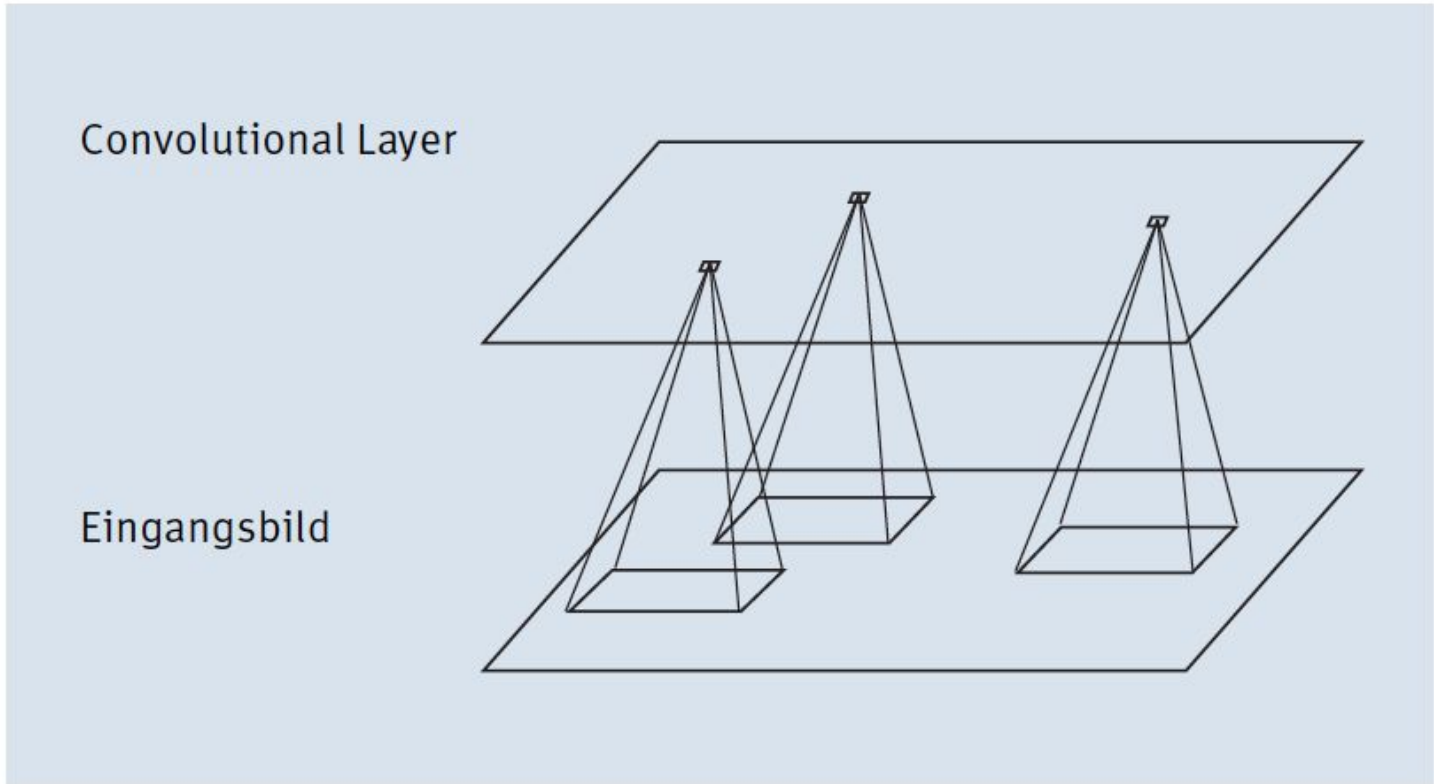


Abbildung 7.5 Der erste Convolutional Layer

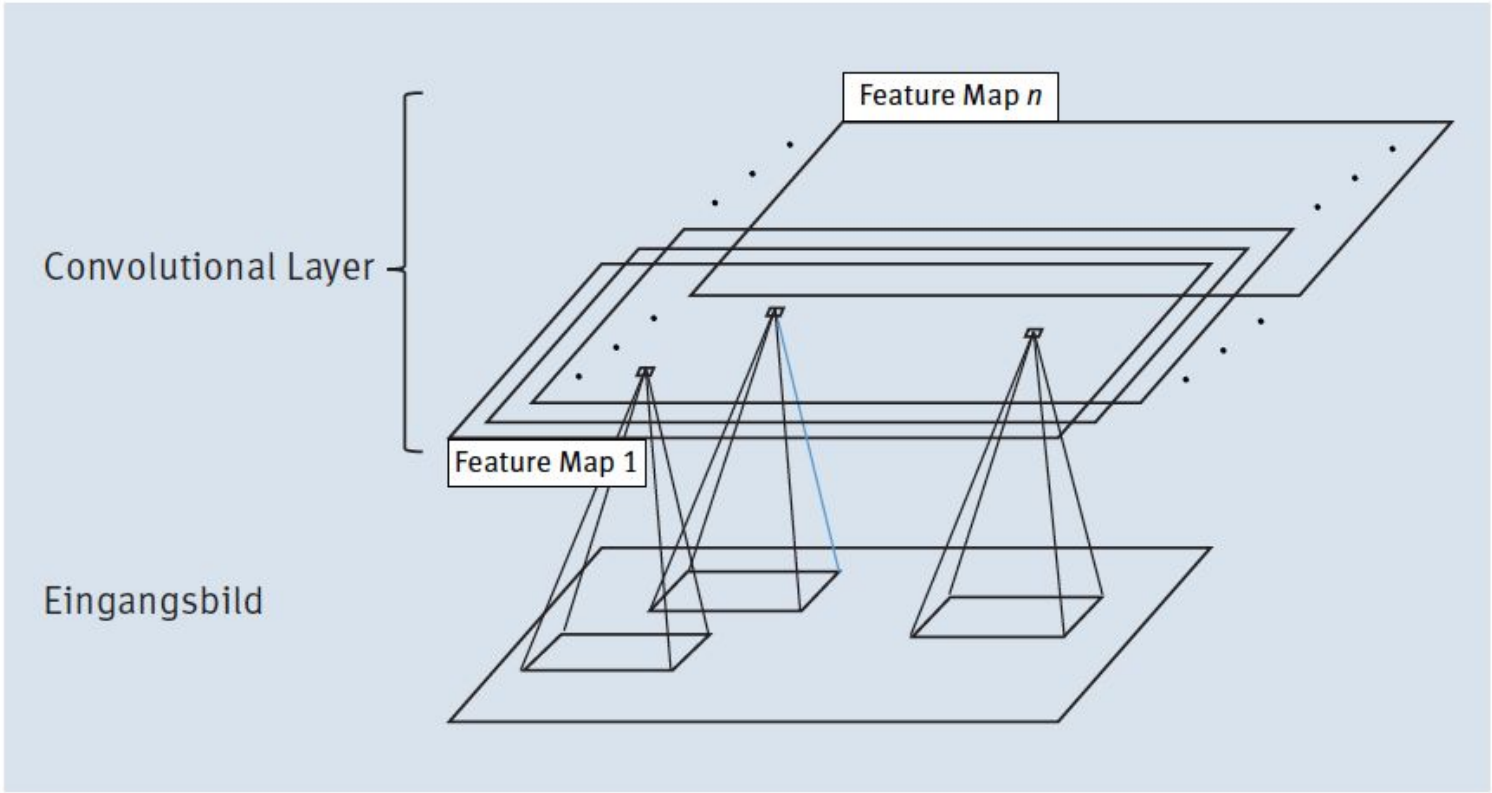


Abbildung 7.6 Ein Convolutional Layer besteht aus mehreren Feature Maps.

Wie viele Feature Maps?

Woher weiß ich, wie viele Feature Maps ich brauche und auf welches Muster im Bild eine Feature Map reagiert? Nun, die richtige Anzahl zu finden, ist nicht so einfach und eine Sache der Erfahrung bzw. des Experimentierens. Die gute Nachricht ist, dass die Muster (also die Gewichte, die durch den 3×3-Filter repräsentiert werden) durch das CNN selbstständig gelernt werden! Allerdings bedarf es dazu einer großen Anzahl an Bildern in der Trainingsphase – aber dazu später.

7.2.2 Activation Function

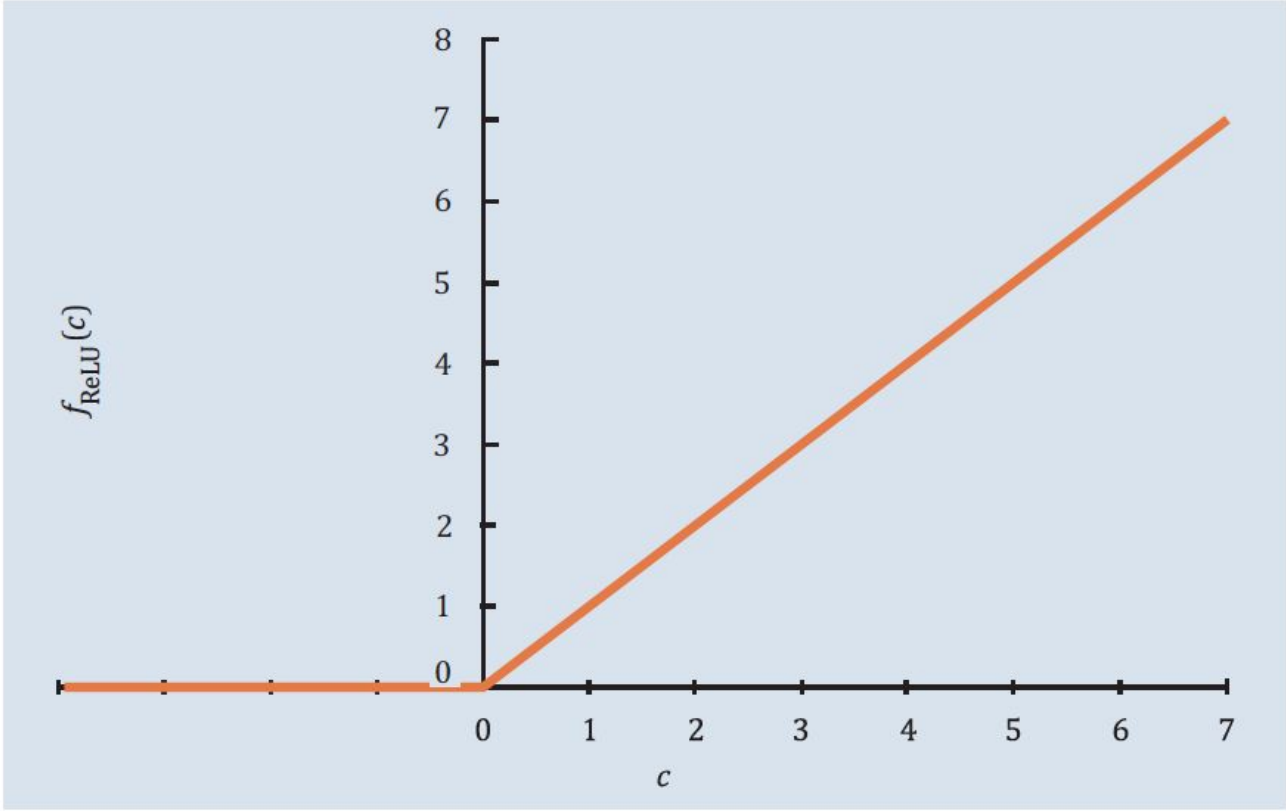


Abbildung 7.7 Darstellung der ReLU-Aktivierungsfunktion

Was ist ein Hyperparameter?

Jeder Algorithmus braucht gewisse Einstellungen – das können Parameter zur Definition der Aktivierungsfunktion sein oder überhaupt die Festlegung, welche Aktivierungsfunktion verwendet wird; auch die Bestimmung, welche Fehlerfunktion verwendet wird, oder ganz spezifische Parameter, die vom Algorithmus abhängen. Man kann sie mit Schaltern, Rädchen, Schräubchen vergleichen, die vor dem Start eingestellt werden müssen, aber das Ergebnis wesentlich beeinflussen können. Solche Parameter nennen wir *Hyperparameter*.

Es gibt natürlich Verfahren, die versuchen, selbstständig die optimalen Einstellungen zu finden. Die sind allerdings sehr aufwendig, da für jede Änderung eines Hyperparameters ein neues Training durchgeführt werden muss, und eignen sich daher nur für Aufgaben mit nicht allzu großer Datenmenge oder wenn die entsprechenden Rechenkapazitäten zur Verfügung stehen.

7.2.3 Pooling Layer

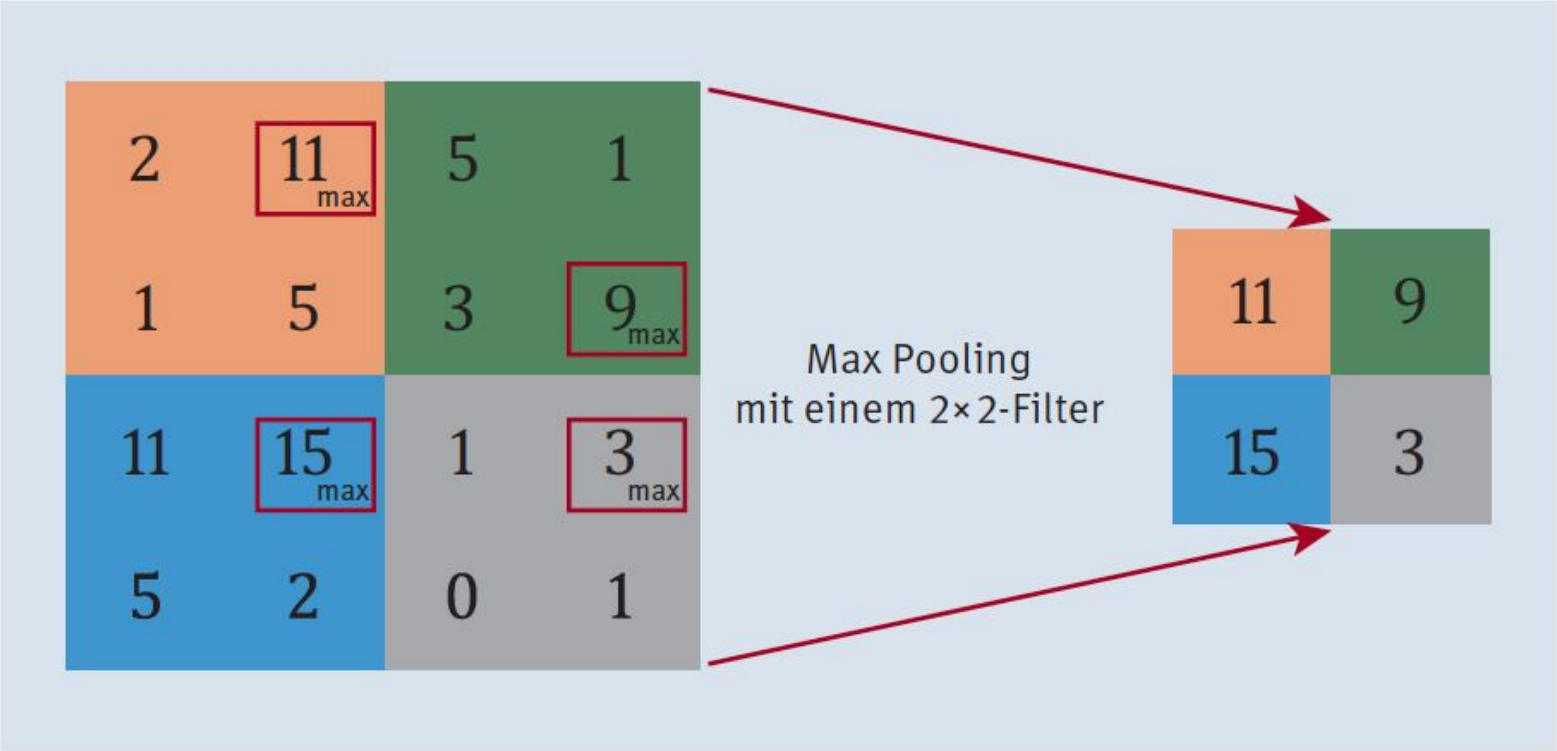


Abbildung 7.8 Beispiel für ein Max Pooling mit einem 2×2-Filter

7.2.4 Überlappen, ausfüllen und Schrittlänge

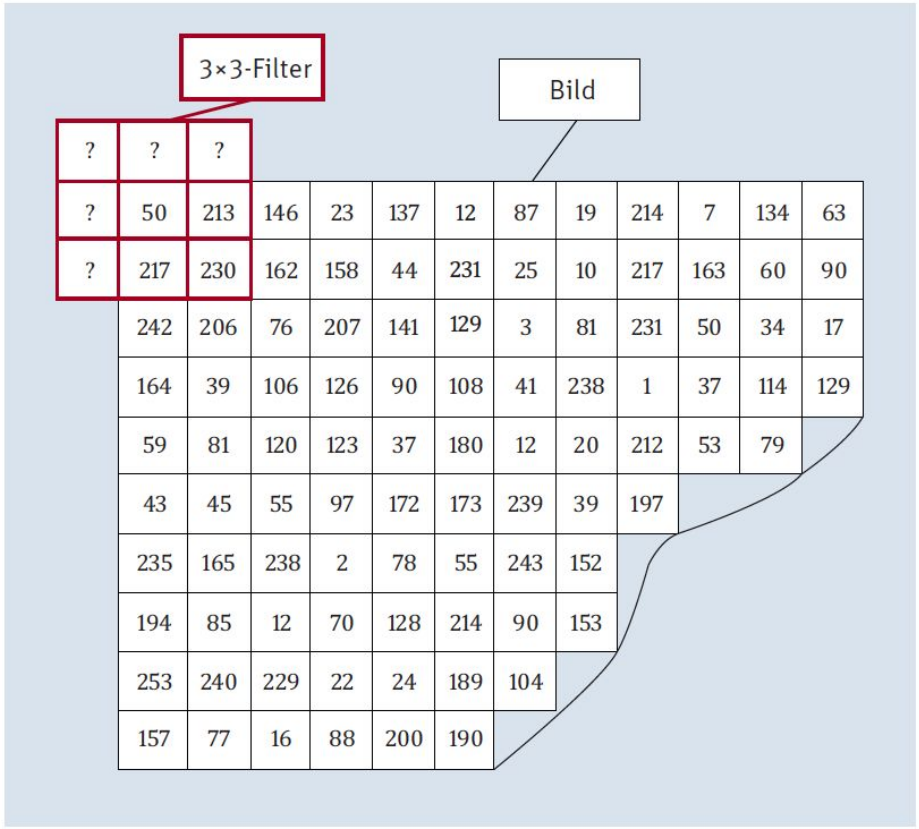


Abbildung 7.9 Filter – was tun mit Randpixeln?

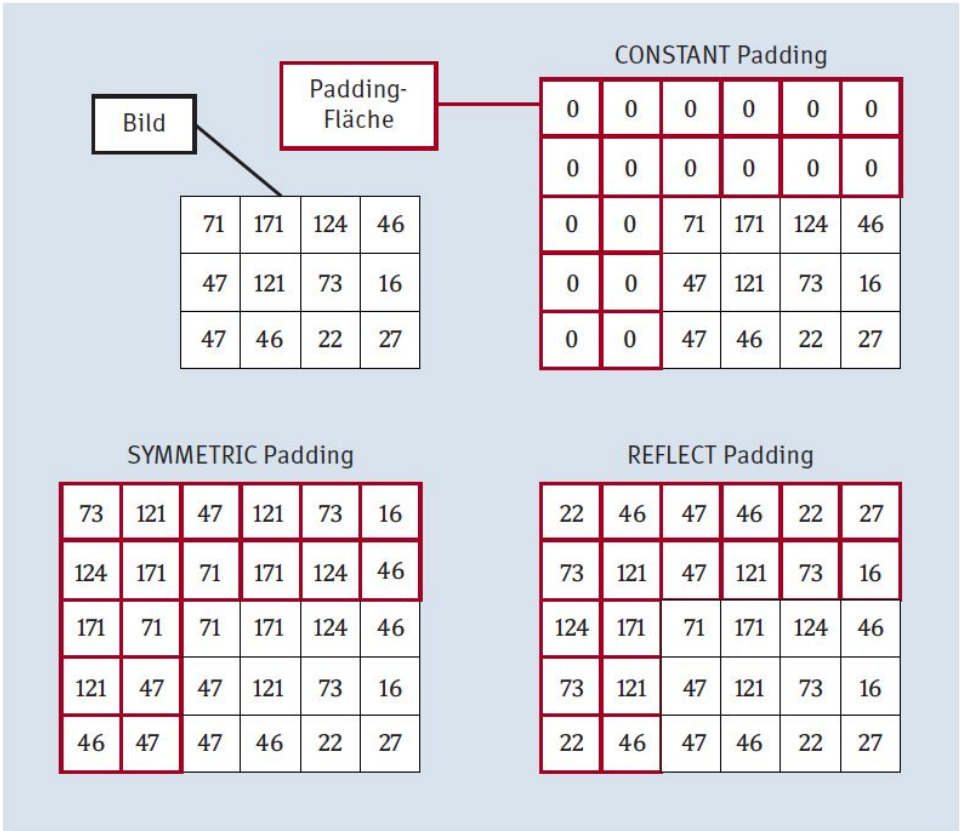


Abbildung 7.10 Padding-Beispiele für einen 5×5-Filter – TensorFlow

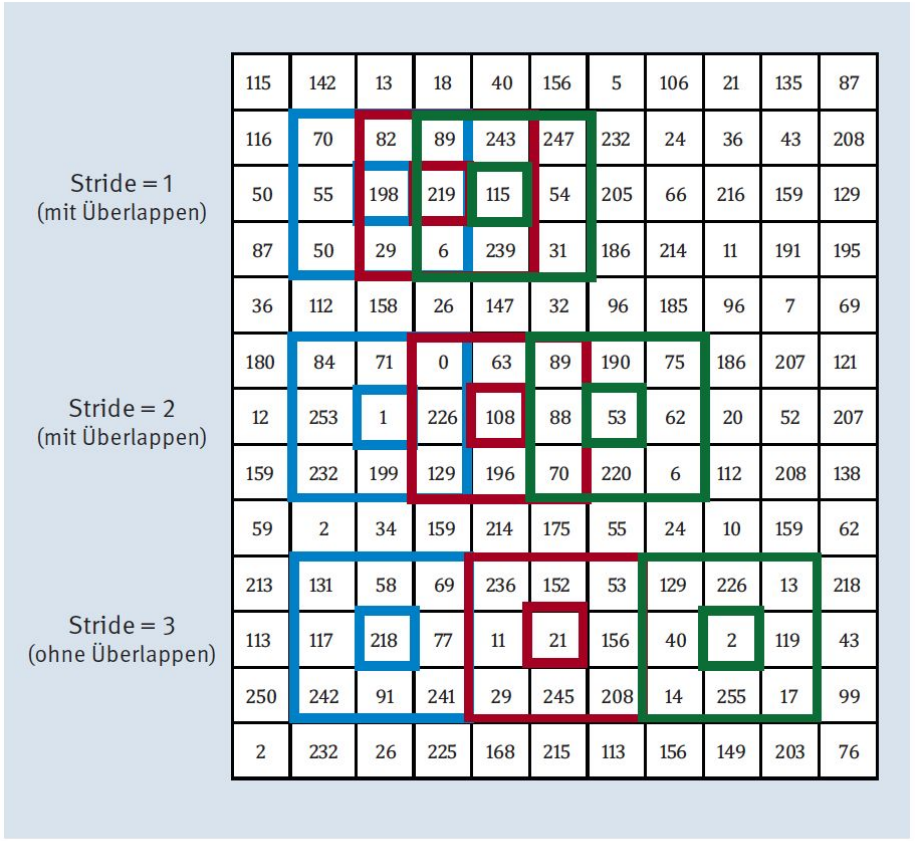


Abbildung 7.11 Die Schrittlänge = Stride eines Filters

7.3 Der Prädiktionsblock
7.3.1 Flatten

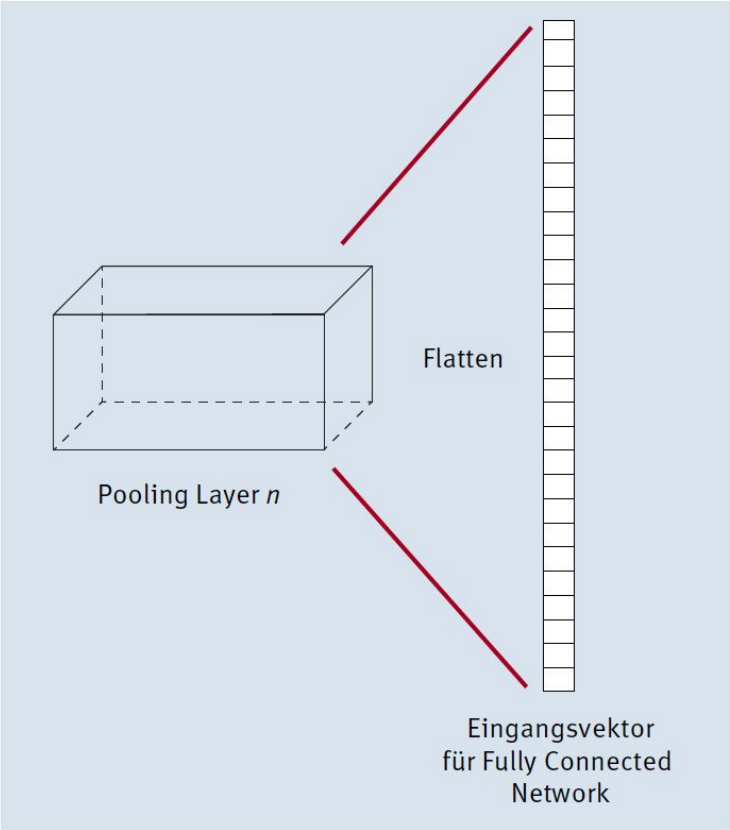


Abbildung 7.12 Flattening des letzten Blocks im Kodierungsblock

7.3.2 Softmax

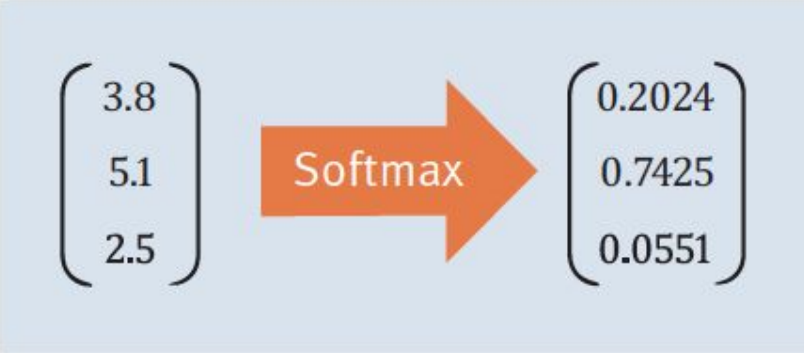


Abbildung 7.13 Beispiel für Softmax für drei Klassen

Softmax kann auch als Aktivierungsfunktion gesehen werden und wird bei k Klassen und $i = 1, \dots, k$ für die Klasse c_j folgendermaßen definiert:

$$\text{Softmax: } f_{\text{akt}}(c_j) = f_{\text{softmax}}(c_j) = \frac{e^{c_j}}{\sum_{i=1}^{k} e^{c_i}}$$

7.4 Trainieren von Convolutional Neural Networks

Name des Convolutional Neural Networks	Anzahl Parameter
LeNet5 (1998)	~ 60.000
AlexNet (2012)	~ 60.000.000
VGG (2014)	~ 138.000.000
GoogleNet (2014)	~ 4.000.000
ResNet50 (2015)	~ 2.400.000

Tabelle 7.1 Vergleich von CNN anhand der Anzahl der Parameter

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

7.4.1 Das Problem der explodierenden/verschwindenden Gradienten Initialisierung

Aktivierungsfunktion	Gleichverteilung $(-v, v)$	Normalverteilung
Sigmoidfunktion	$v = \sqrt{\frac{6}{n_{\text{ein}} + n_{\text{aus}}}}$	$\sigma = \sqrt{\frac{2}{n_{\text{ein}} + n_{\text{aus}}}}$
Tangens hyperbolicus	$v = 4\sqrt{\frac{6}{n_{\text{ein}} + n_{\text{aus}}}}$	$\sigma = 4\sqrt{\frac{2}{n_{\text{ein}} + n_{\text{aus}}}}$
ReLU	$v = \sqrt{2}\sqrt{\frac{6}{n_{\text{ein}} + n_{\text{aus}}}}$	$\sigma = \sqrt{2}\sqrt{\frac{2}{n_{\text{ein}} + n_{\text{aus}}}}$

Tabelle 7.2 Gleichverteilung und Normalverteilung für die zufällige Initialisierung der Gewichte

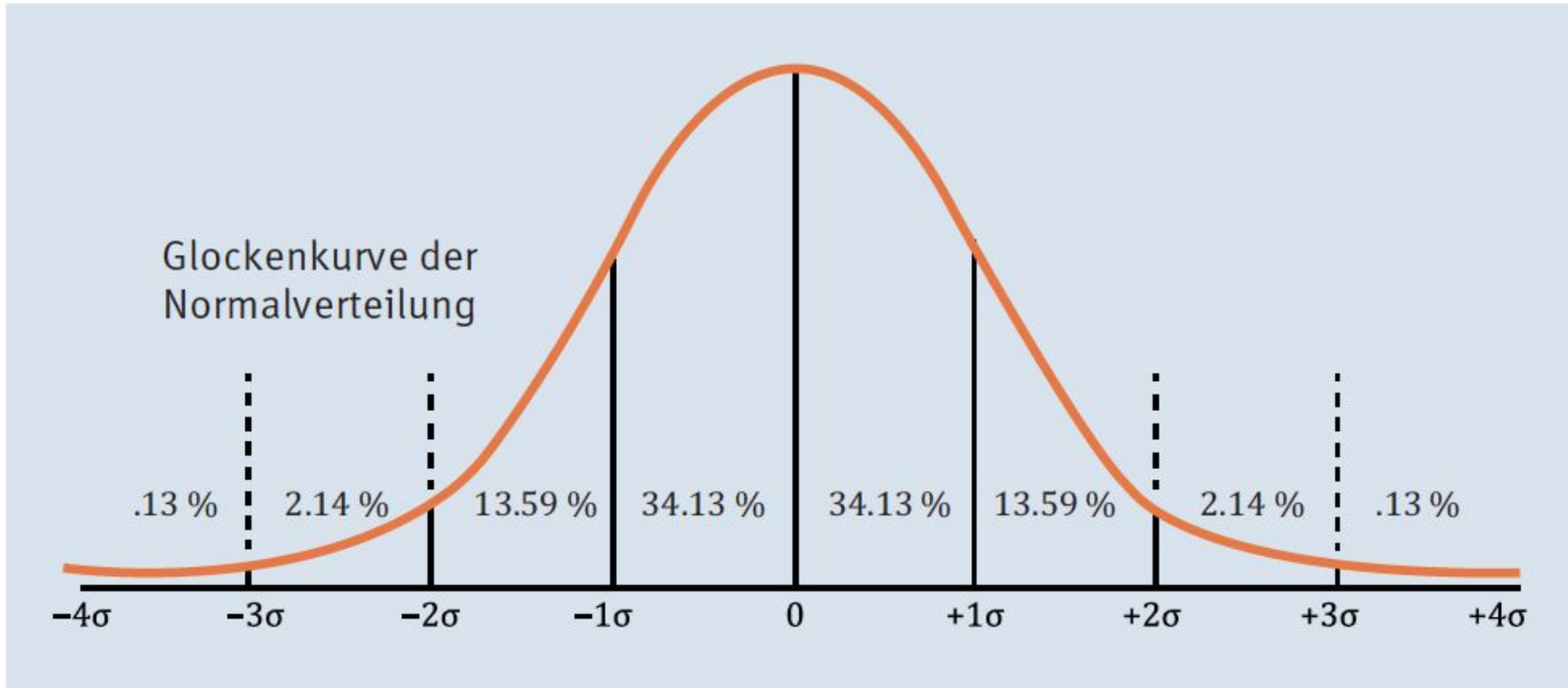


Abbildung 7.14 Typische Normalverteilung

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Aktivierungsfunktion

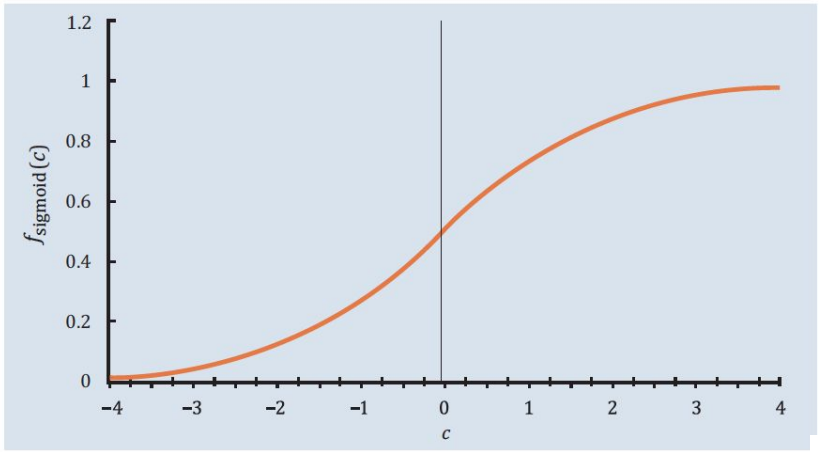


Abbildung 7.15 Sigmoidfunktion

lReLU: $f_{\text{akt}}(x) = f_{\text{lReLU}}(x) = \max(\alpha x, x)$, wo $\alpha = 0.01$ (sehr klein)

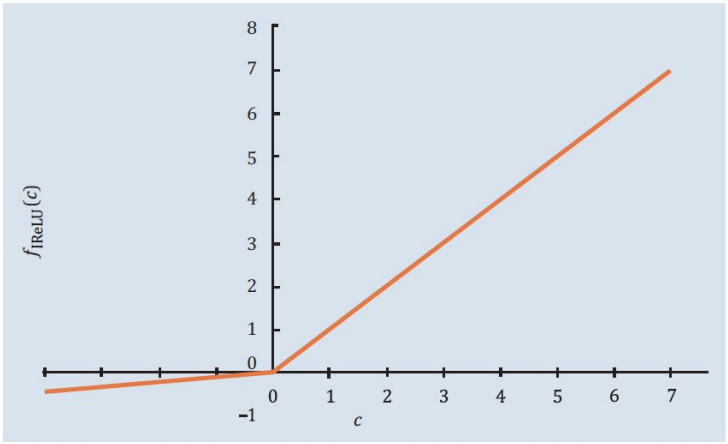


Abbildung 7.16 Die Leaky-ReLU-Aktivierungsfunktion

ELU: $f_{\text{akt}}(x) = f_{\text{ELU}}(x) = \begin{cases} \alpha(e^x - 1), & \text{falls } x < 0 \\ x, & \text{falls } x \geq 0 \end{cases}$

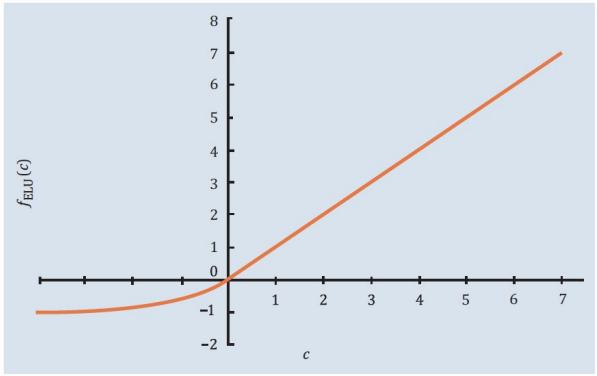


Abbildung 7.17 Die ELU-Aktivierungsfunktion

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

7.4.2 Das Optimierungsverfahren Momentum-Optimierung Adam-Optimierung

$$\nabla_W C(W) = \begin{pmatrix} \frac{\partial}{\partial W_0} C(W) \\ \frac{\partial}{\partial W_1} C(W) \\ \vdots \\ \frac{\partial}{\partial W_n} C(W) \end{pmatrix}$$

$$\vec{h}_{\text{neu}} = \eta_{\text{momentum}} \vec{h}_{\text{alt}} + \eta \nabla_W C(W)$$

$$W_{\text{neu}} = W_{\text{alt}} - \eta_{\text{Adam}}(h_{\text{alt}}, h_{\text{alt}}^2)$$

Das bedeutet also, vom Gewichtsvektor wird ein Veränderungsvektor $\vec{h}$ abgezogen. Dieser Veränderungsvektor sieht beim Standard-Gradientenverfahren folgendermaßen aus:

$$\vec{h} = \eta \nabla_W C(W)$$

Dabei ist η unsere schon bekannte Lernrate. Die neuen Gewichtsvektoren erhalten wir also durch Subtraktion des Veränderungsvektors:

$$W^{\text{neu}} = W^{alt} - \vec{h}$$

Die folgenden beiden Beispiele von Optimierungsvarianten modifizieren diesen Veränderungsvektor, um das Training schneller konvergieren zu lassen.

7.4.3 Verhindern von Overfitting Early Stopping

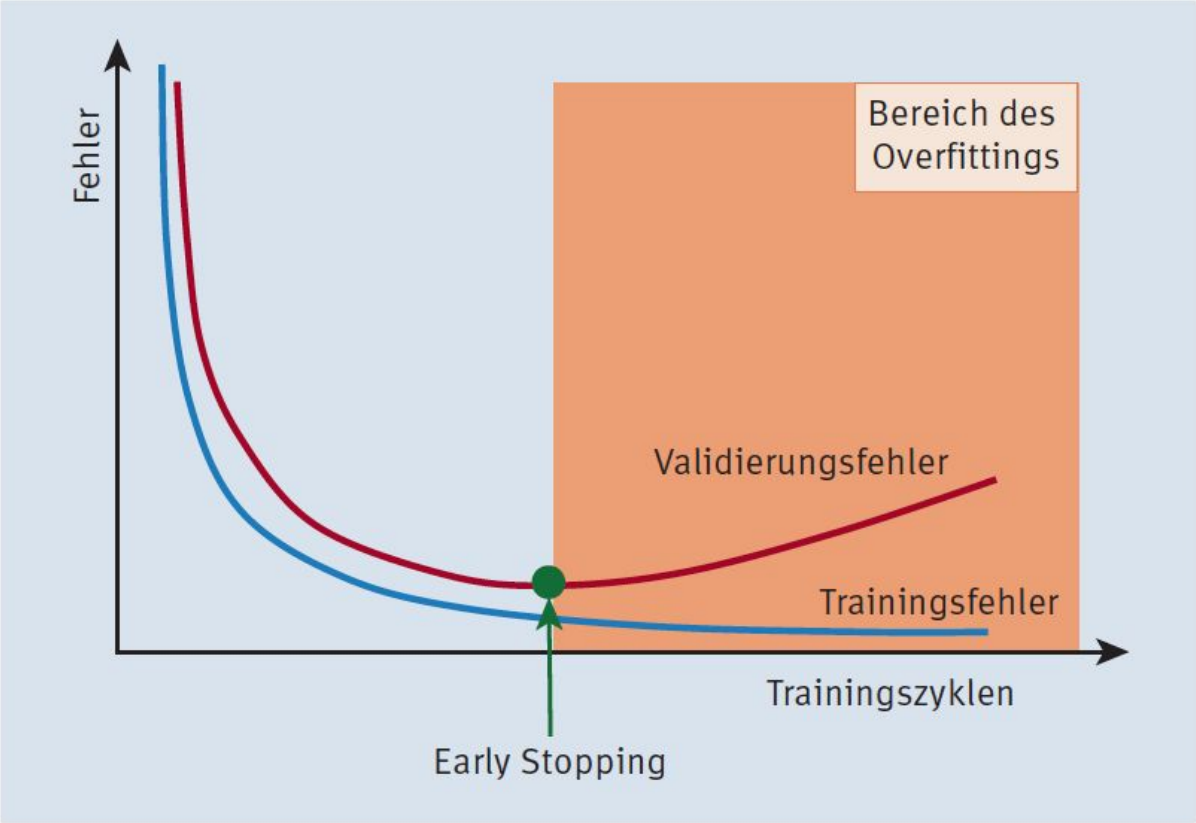


Abbildung 7.18 Fehlerkurve bei Early Stopping

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Dropout

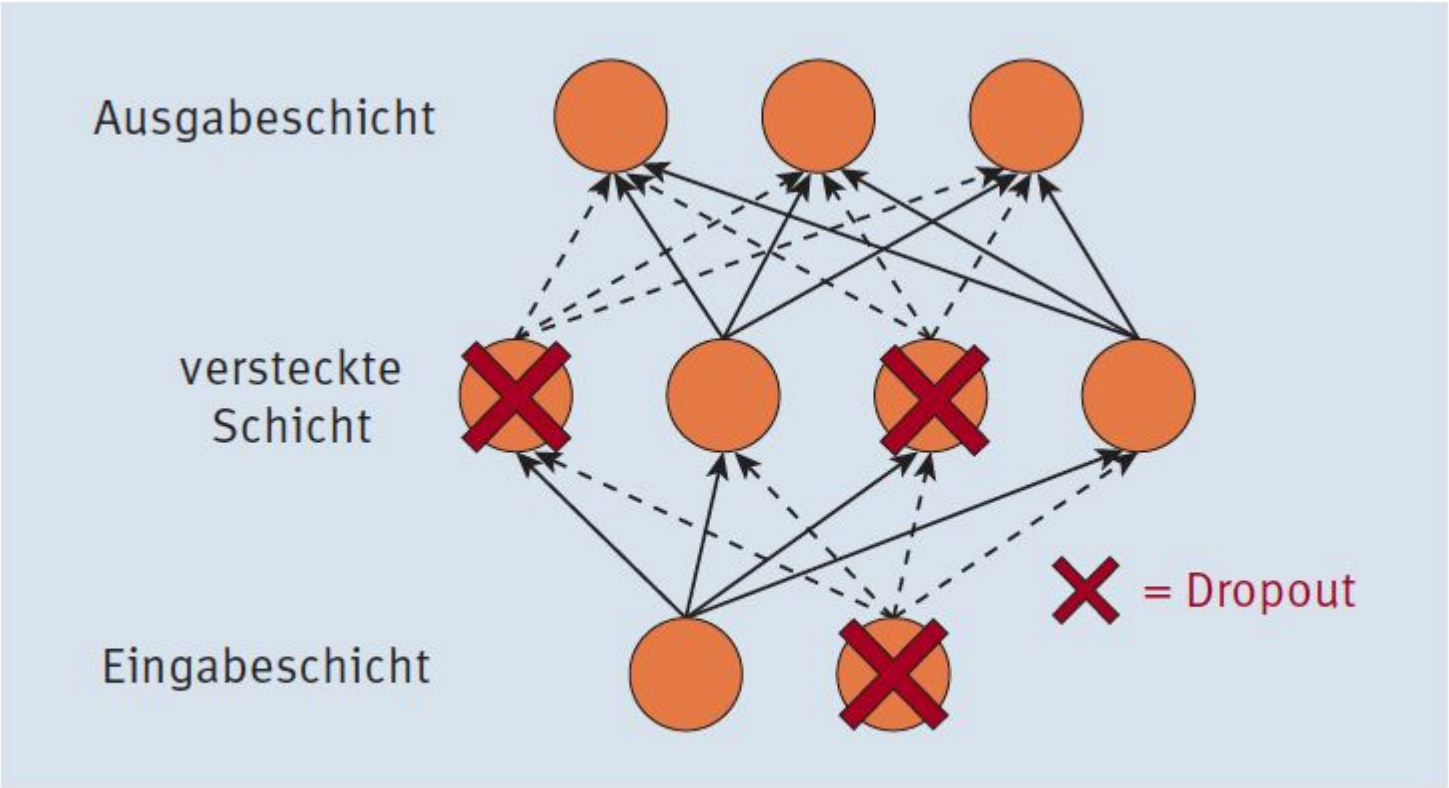


Abbildung 7.19 Netz mit Dropout-Neuronen (Dropout-Rate p = 50 %)

Programmierung von Convolutional Neural Networks mit TensorFlow 2.0

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

8.1 Convolutional Networks zur Handschriftenerkennung
8.1.1 Der Datensatz

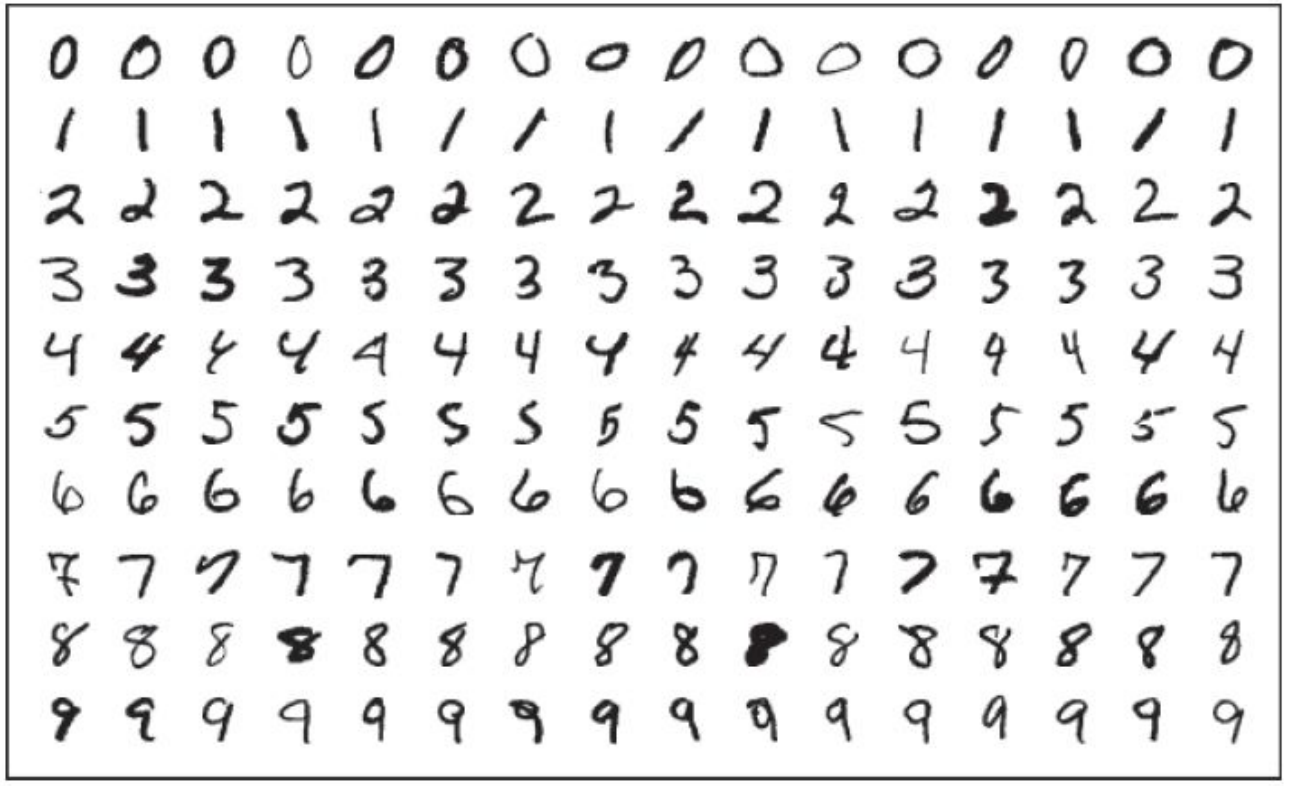


Abbildung 8.1 Ausschnitt aus dem MNIST-Datensatz

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Keras

Die Keras-Bibliothek verdient eine besondere Erwähnung. Sie wurde von François Chollet entwickelt. Keras ist eine Python-Bibliothek, die eine einheitliche Schnittstelle für verschiedene Bibliotheken für neuronale Netzwerke bietet, wie TensorFlow, CNTK oder Theano. Das Ziel von Keras ist, die Anwendung dieser Bibliotheken so einsteiger- und benutzerfreundlich wie möglich zu machen. Obwohl Keras als eigenständige Bibliothek erhalten bleibt, wurde sie ab der TensorFlow-Version 1.4 integriert und mit der Version 2.0 noch mehr an TensorFlow angepasst und erweitert.

8.1 Laden des MNIST-Datensatzes aus der »tensorflow.keras«-Bibliothek

8.2 Laden und Vorverarbeitung des MNIST-Datensatzes

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Listing 8.3 Format des modifizierten Trainings- und Testdatensatzes

Listing 8.4 Darstellung eines zufällig ausgewählten MNIST-Bildes

8.1.2 Ein einfaches CNN

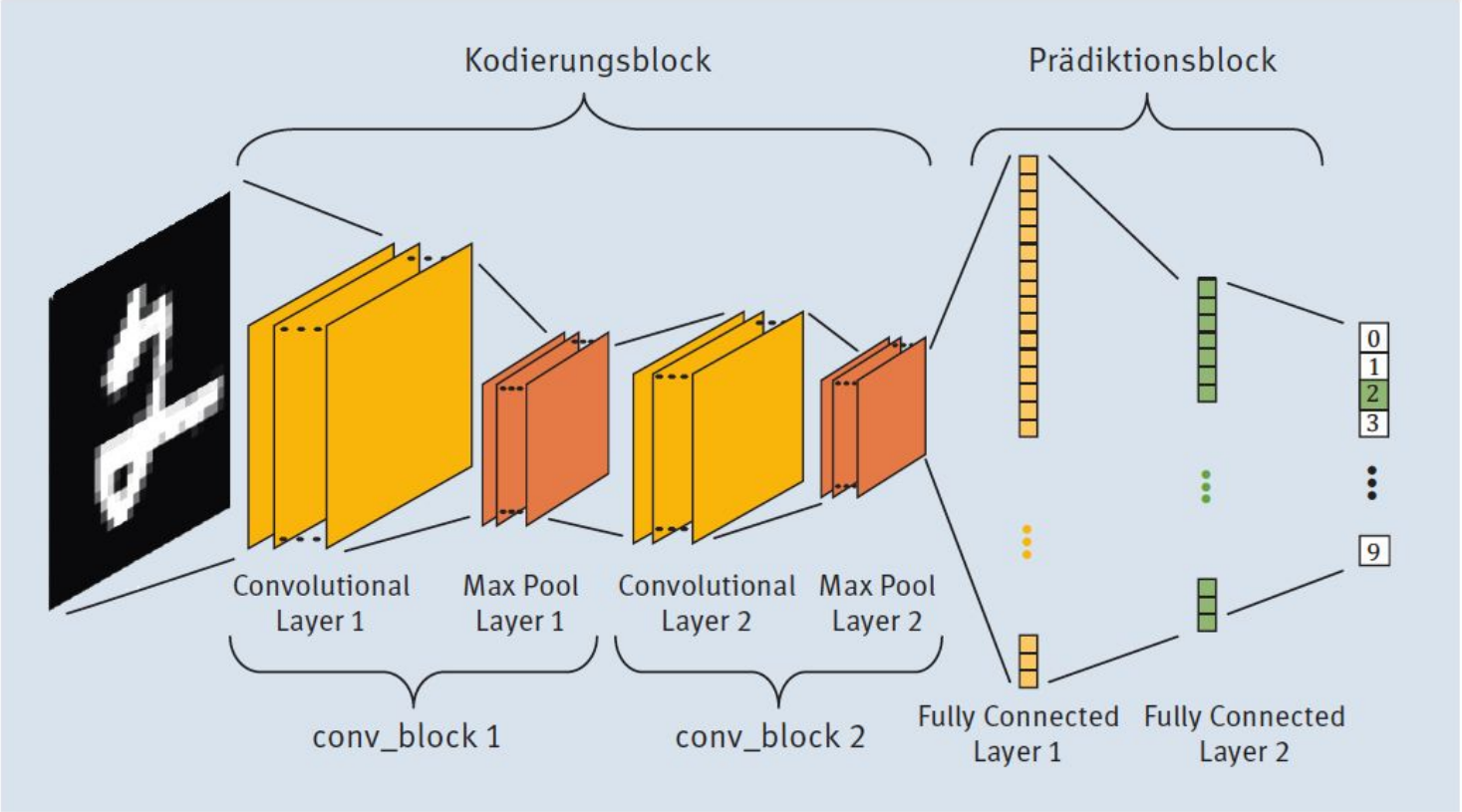


Abbildung 8.3 Die Struktur unseres CNN für den MNIST-Datensatz

Listing 8.5 Import der Bibliotheken

Das Modell
Listing 8.6 Codeabschnitt für die Netzwerkarchitektur

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 24, 24, 32)        832
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 12, 12, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 8, 8, 64)          51264
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
dropout_8 (Dropout)          (None, 4, 4, 64)          0
_________________________________________________________________
flatten_4 (Flatten)          (None, 1024)              0
_________________________________________________________________
features (Dense)             (None, 128)               131200
_________________________________________________________________
dropout_9 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_7 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_8 (Dense)              (None, 10)                650
=================================================================
Total params: 192,202
Trainable params: 192,202
Non-trainable params: 0

_________________________________________________________________
```

Abbildung 8.4 Ausgabe der Funktion »model.summary()«

Listing 8.7 Festlegung Verlustfunktion und Optimierung

Das Training
Listing 8.8 Training des Modells mit »model.fit()«

8.1.3 Die Ergebnisse

Tensorboard

Um die Ergebnisse selbst mittels TensorBoard nachzuvollziehen, können Sie es in einem Terminal (am besten aus Anaconda Navigator in der entsprechenden Environment) selbst starten mit dem Befehl:

```
tensorboard -logdir "logs"
```

Achtung! Sie müssen im Terminal zuerst in das Verzeichnis wechseln, in dem sich auch das Jupyter Notebook befindet. Statt "logs" müssen Sie das Verzeichnis nehmen, dass Sie in Listing 8.8 für die Variable LOG_DIR gewählt haben.

Danach öffnen Sie einen Webbrowser und starten die TensorBoard-Webapplikation mit der URL *http://localhost:6006/* (oder, wenn das nicht funktioniert, mit *http://rechner-name:6006*)

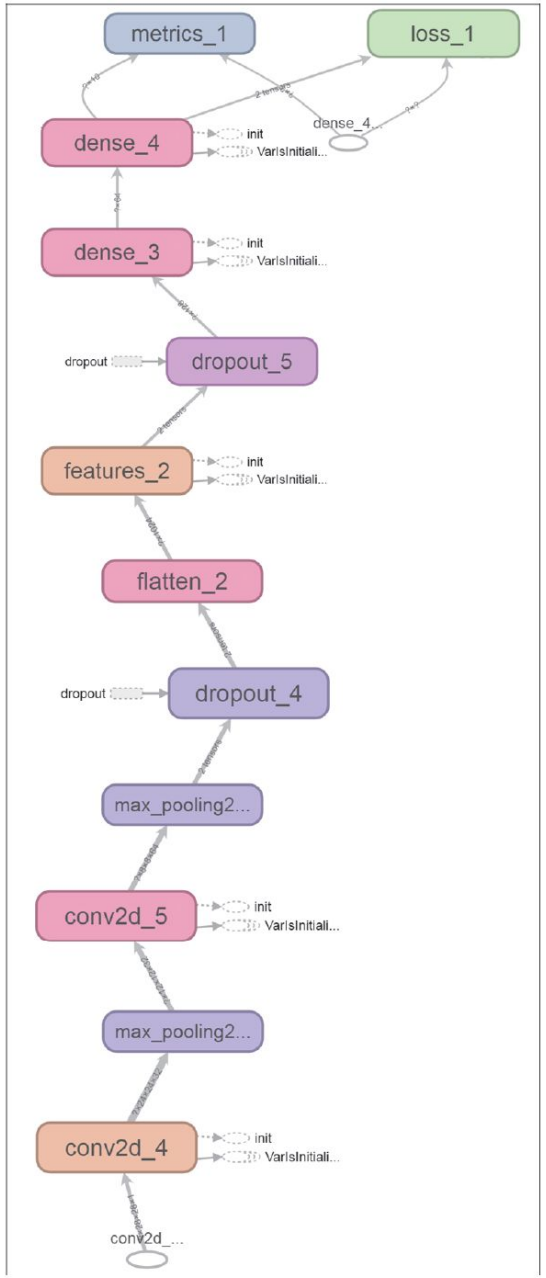


Abbildung 8.5 Unser Convolutional Neuronal Network im TensorBoard (Ausschnitt)

Listing 8.9 Verlust und Genauigkeit unseres trainierten CNN

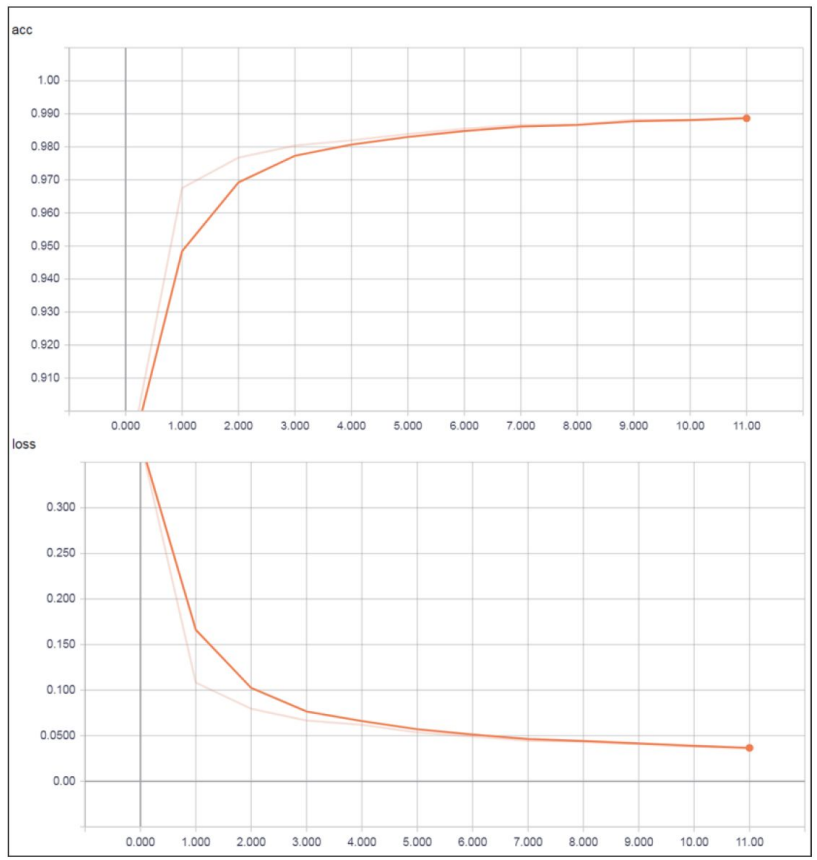


Abbildung 8.6 Verlauf Genauigkeit (»acc«) und Verlust (»loss«) der Trainingsdaten

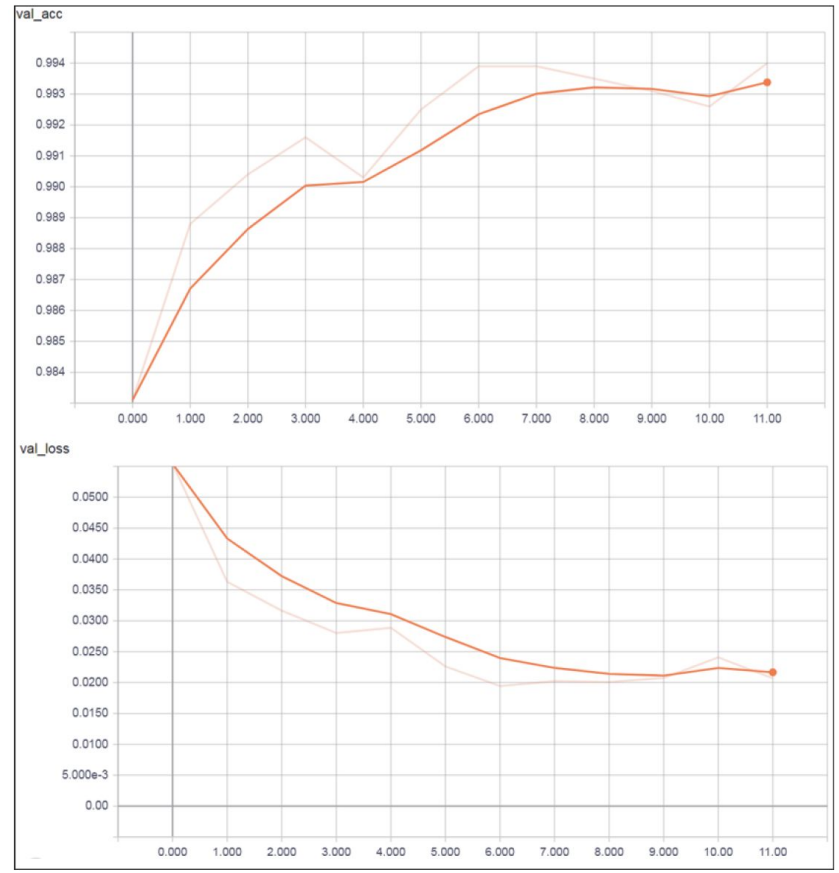


Abbildung 8.7 Verlauf Genauigkeit (»acc«) und Verlust (»loss«) der Testdaten

Listing 8.10 Speichern des Modells und der Gewichte (h5-format)

Listing 8.11 Speichern des Modells und der Gewichte (pb-format)

Listing 8.12 Laden und Verwenden des vortrainierten Modells

Listing 8.13 Vergleich der Ergebnisse und Prüfung des trainierten Netzes

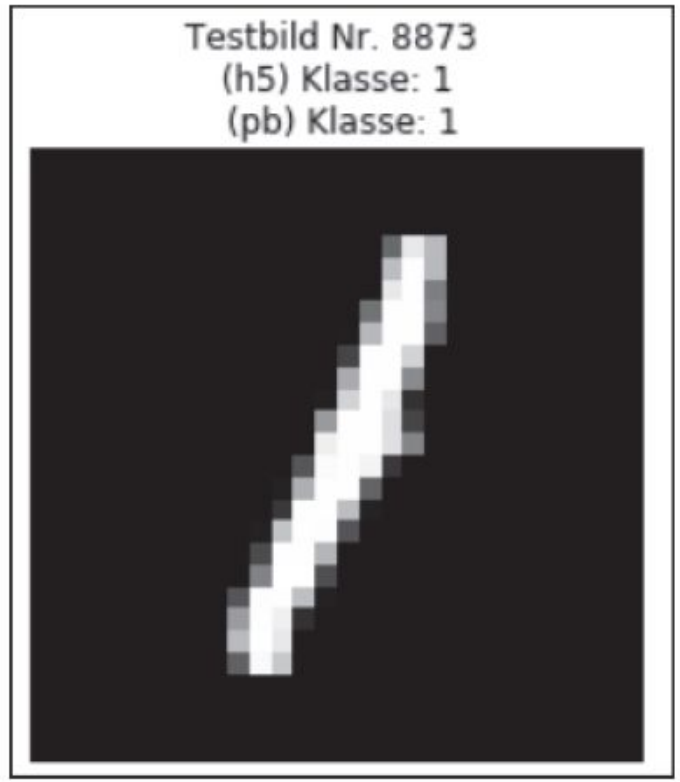


Abbildung 8.8 Klassifikationsergebnis eines Testbildes

8.2 Transfer Learning mit Convolutional Neural Networks

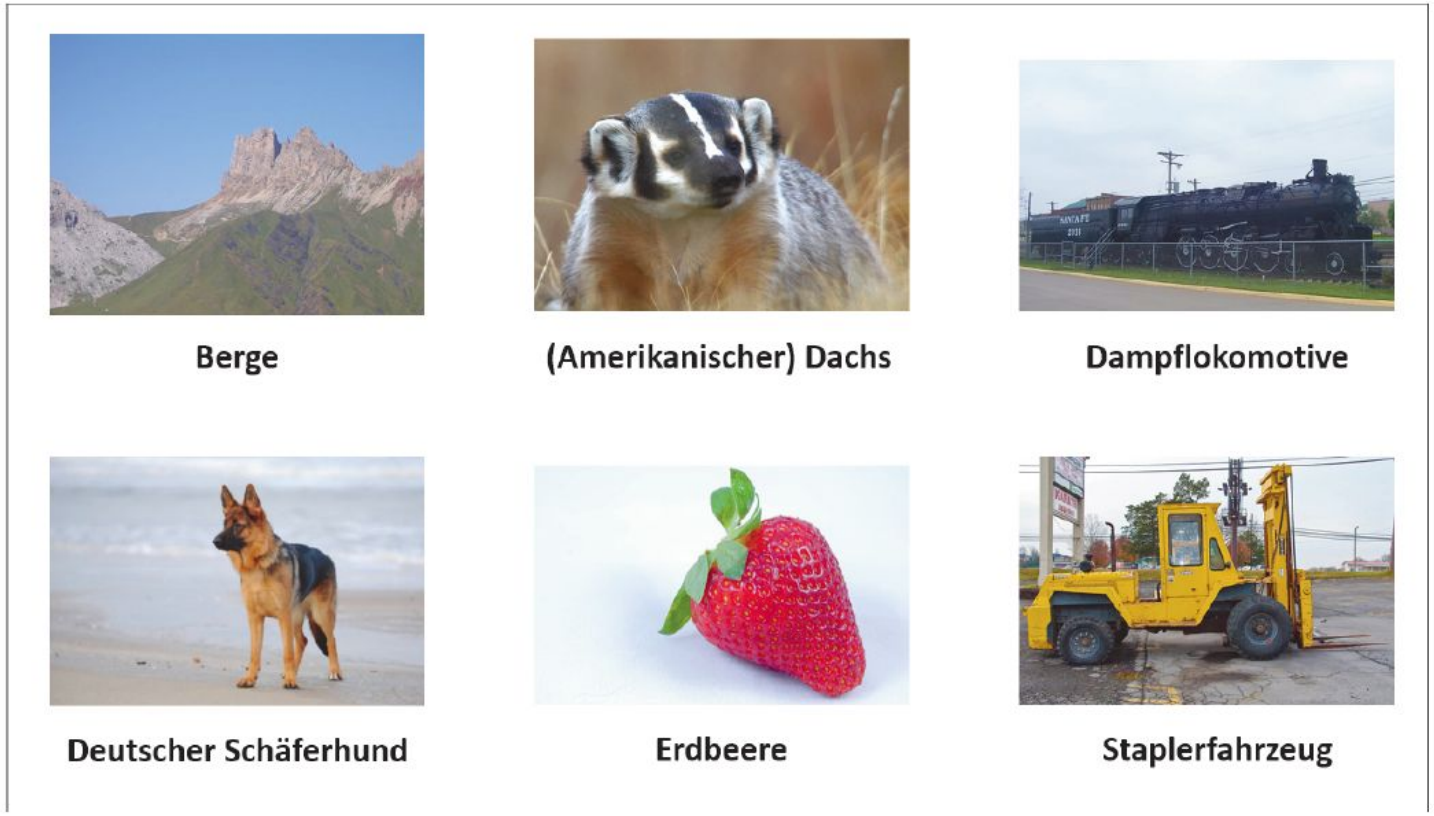


Abbildung 8.9 Beispielbilder für unsere Aufgabe

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

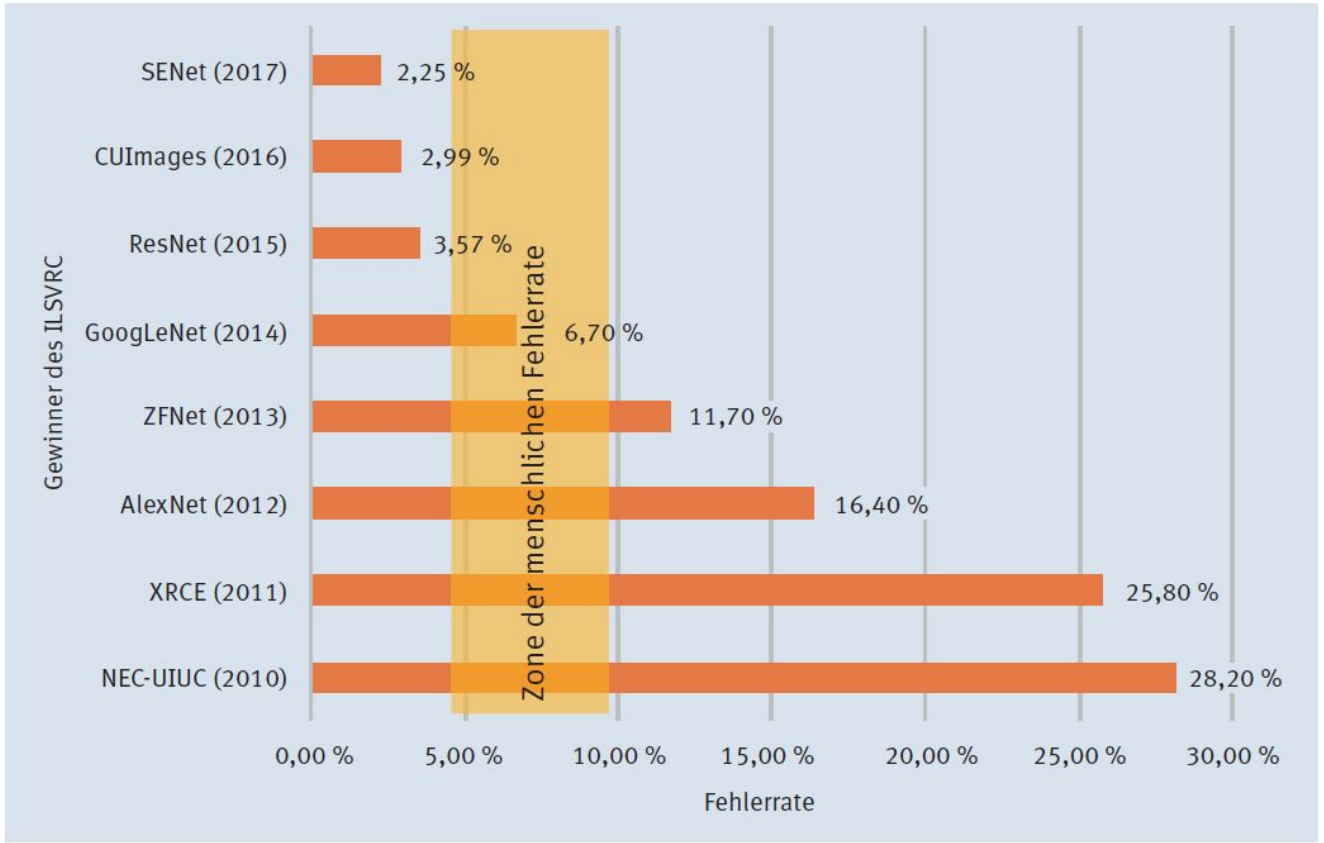


Abbildung 8.10 Gewinner des ILSVRC seit 2010 mit Angabe der Fehlerrate

8.2.1 Das vortrainierte Netzwerk

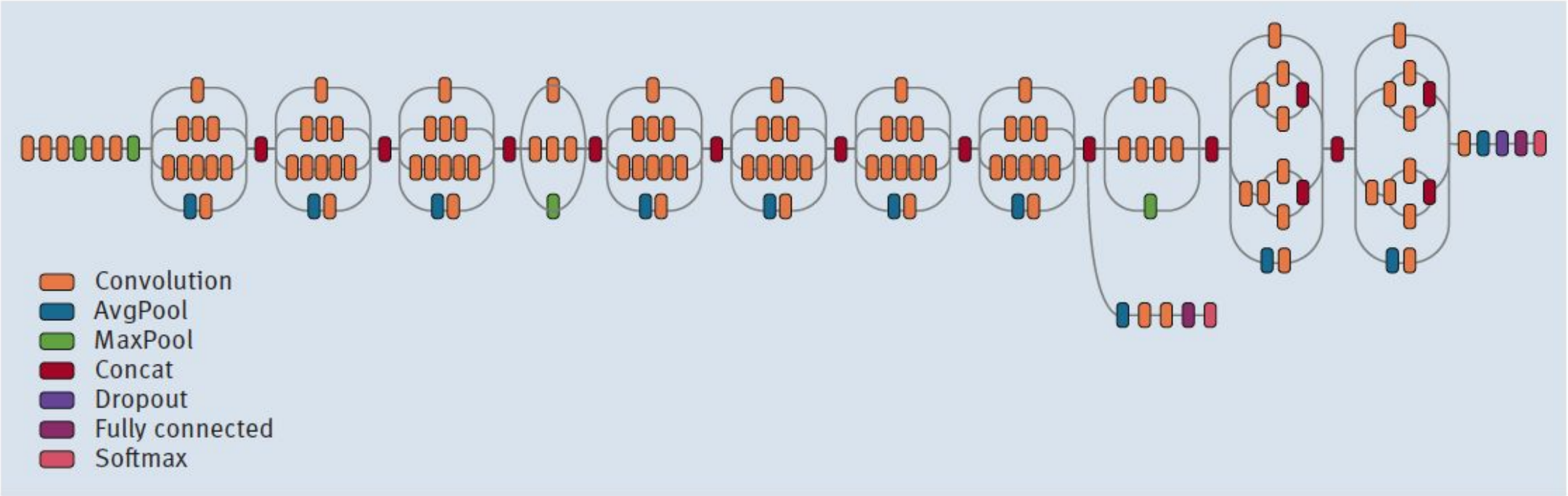


Abbildung 8.11 Die Netzwerkstruktur von Inception-v3

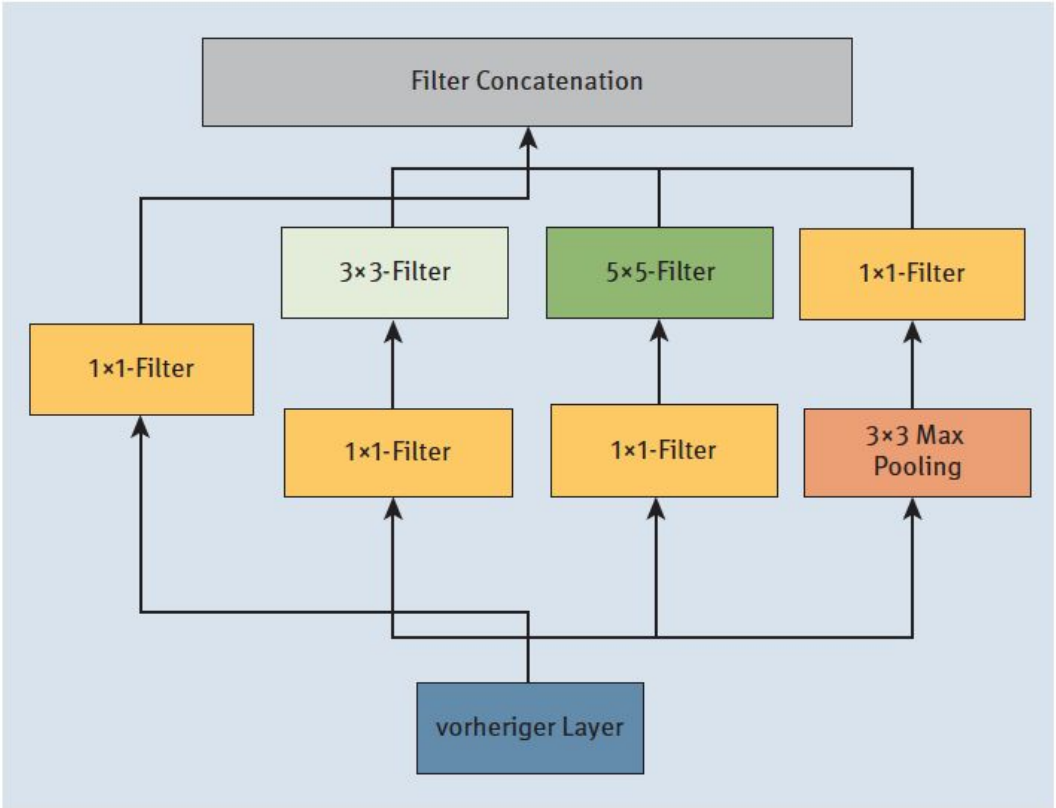
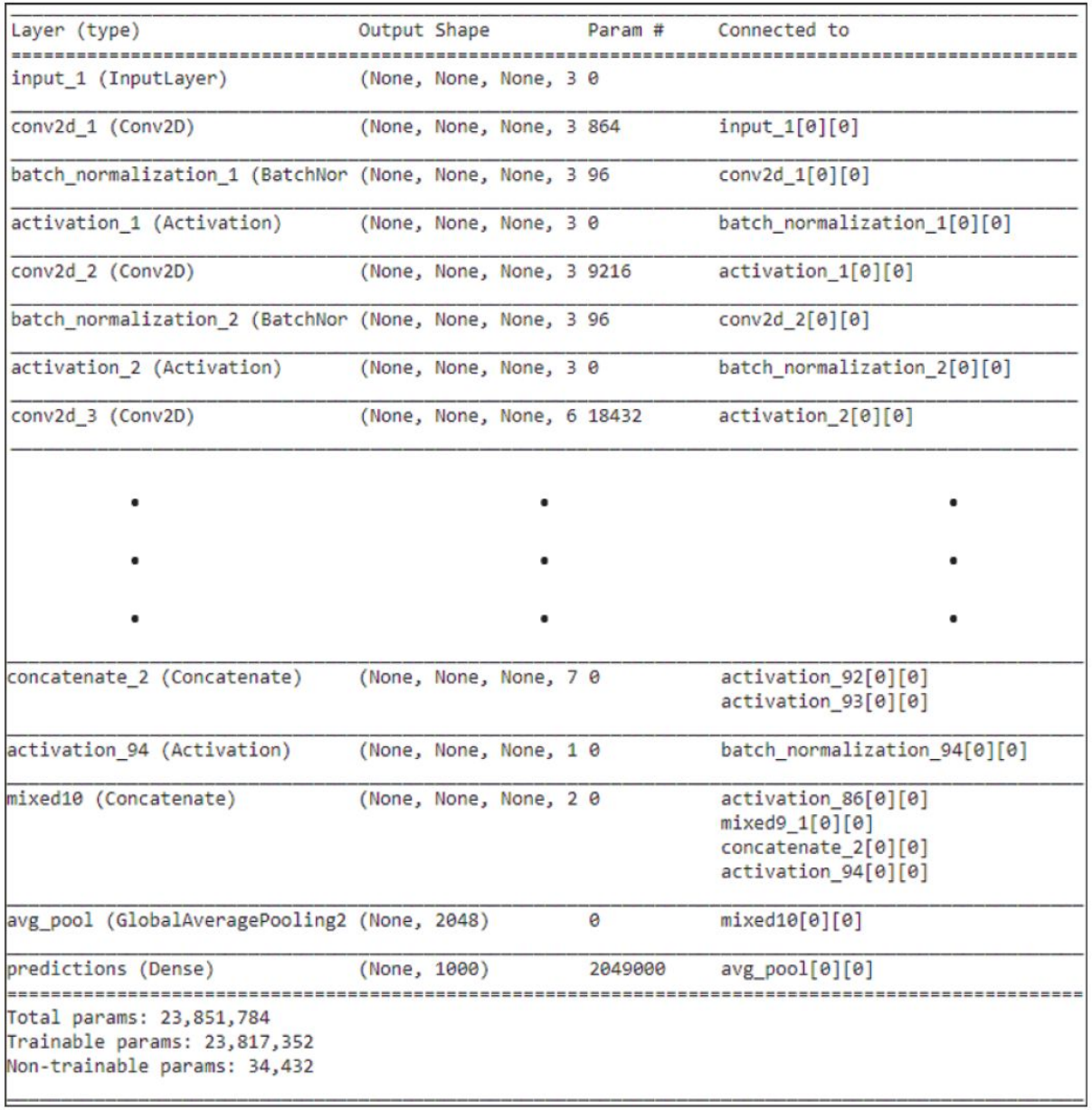


Abbildung 8.13 Inception-Modul, aus dem Inception-v3 aufgebaut ist

8.2.2 Datenvorbereitung
Listing 8.14 Import der Bibliotheken für neuronale Netze

Listing 8.15 Wichtige Helferlein für die Bildbearbeitung

8.2.3 Das vortrainierte Netz
Listing 8.16 Laden und Anzeigen des Modells Inception-v3



Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, None, None, 3	0	
conv2d_1 (Conv2D)	(None, None, None, 3	864	input_1[0][0]
batch_normalization_1 (BatchNor	(None, None, None, 3	96	conv2d_1[0][0]
activation_1 (Activation)	(None, None, None, 3	0	batch_normalization_1[0][0]
conv2d_2 (Conv2D)	(None, None, None, 3	9216	activation_1[0][0]
batch_normalization_2 (BatchNor	(None, None, None, 3	96	conv2d_2[0][0]
activation_2 (Activation)	(None, None, None, 3	0	batch_normalization_2[0][0]
conv2d_3 (Conv2D)	(None, None, None, 6	18432	activation_2[0][0]
•	•	•	•
•	•	•	•
•	•	•	•
concatenate_2 (Concatenate)	(None, None, None, 7	0	activation_92[0][0] activation_93[0][0]
activation_94 (Activation)	(None, None, None, 1	0	batch_normalization_94[0][0]
mixed10 (Concatenate)	(None, None, None, 2	0	activation_86[0][0] mixed9_1[0][0] concatenate_2[0][0] activation_94[0][0]
avg_pool (GlobalAveragePooling2	(None, 2048)	0	mixed10[0][0]
predictions (Dense)	(None, 1000)	2049000	avg_pool[0][0]

Total params: 23,851,784
Trainable params: 23,817,352
Non-trainable params: 34,432

Abbildung 8.14 Ausschnitt der Inception-v3-Netzwerkstruktur

Listing 8.17 Laden des zu klassifizierenden Bilds

Abbildung 8.15 Bild eines Deutschen Schäferhunds (Budinho, CC BY-SA 3.0, https://commons.wikimedia.org/wiki/File:Grauer_Deutscher_Schäferhund_Standbild.jpg)

Listing 8.18 Weitere Bildvorverarbeitung und Ausgabe der Form und der Pixelwerte

8.2.4 Die Ergebnisse
Listing 8.19 Klassifikation des Bildes

Listing 8.20 Das klassifizierte Bild mit Klassenlegende

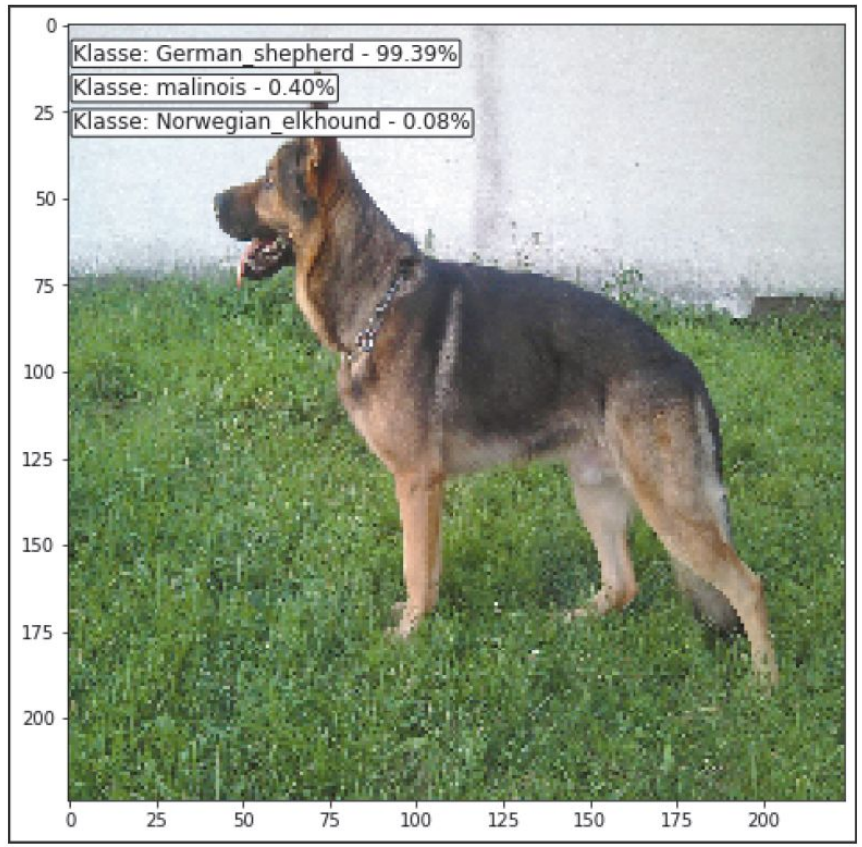


Abbildung 8.16 Klassifikationsergebnis

Kapitel 10
Die Evolution der neuronalen Netze

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

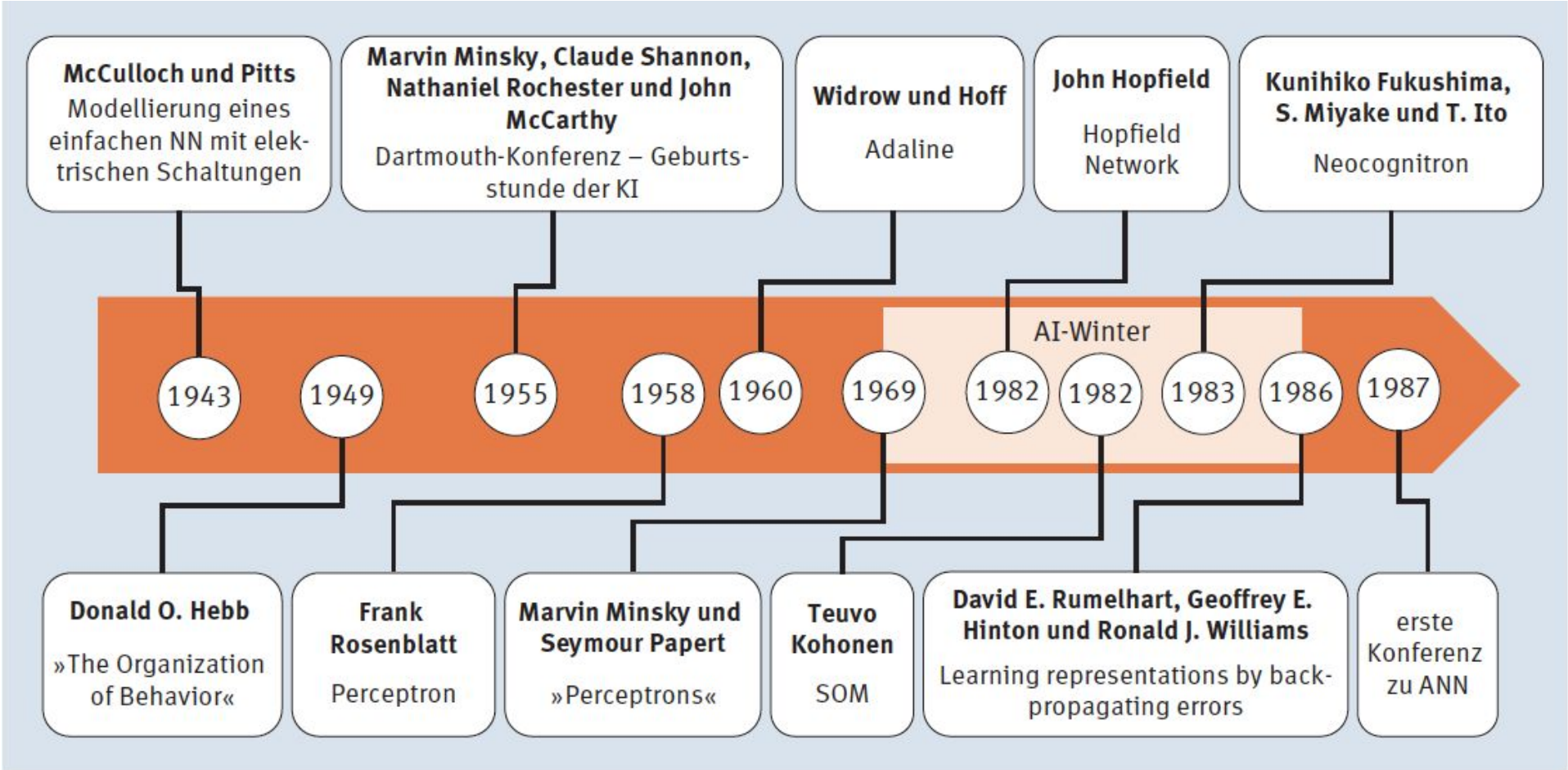


Abbildung 10.1 KNN-Geschichte, Teil 1

10.1.1 1943: McCulloch-Pitts Neurons

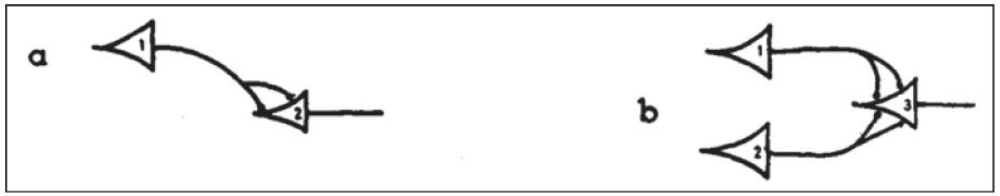


Abbildung 10.2 Grafische Darstellung (Ausschnitt) von McCulloch-Pitts-Netzen aus deren Originalpaper von 1943: »A logical calculus of the ideas immanent in nervous activity«

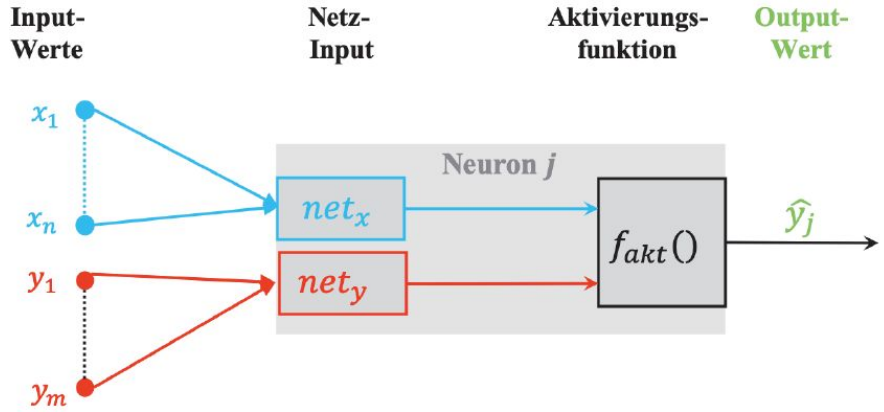


Abbildung 10.3 Darstellung eines McCulloch-Pitts-Neurons

Wenn man nun eher der formale Typ ist, dann wird man sich eine kompakte Darstellung wünschen, die wir, farblich angereichert, gerne liefern:

$$net_x = \sum_{i=1}^{n} x_i$$

$$net_y = \sum_{i=1}^{m} y_i$$

$$f_{\text{akt}}(net_x, net_y, \theta) = \hat{y}_j = \begin{cases} 1, & \text{falls } net_x \geq \theta \text{ und } net_y = 0 \\ 0, & \text{sonst} \end{cases}$$

10.2.4 1959: Bernard Widrow und Marcian Hoff – Adaline und Madaline

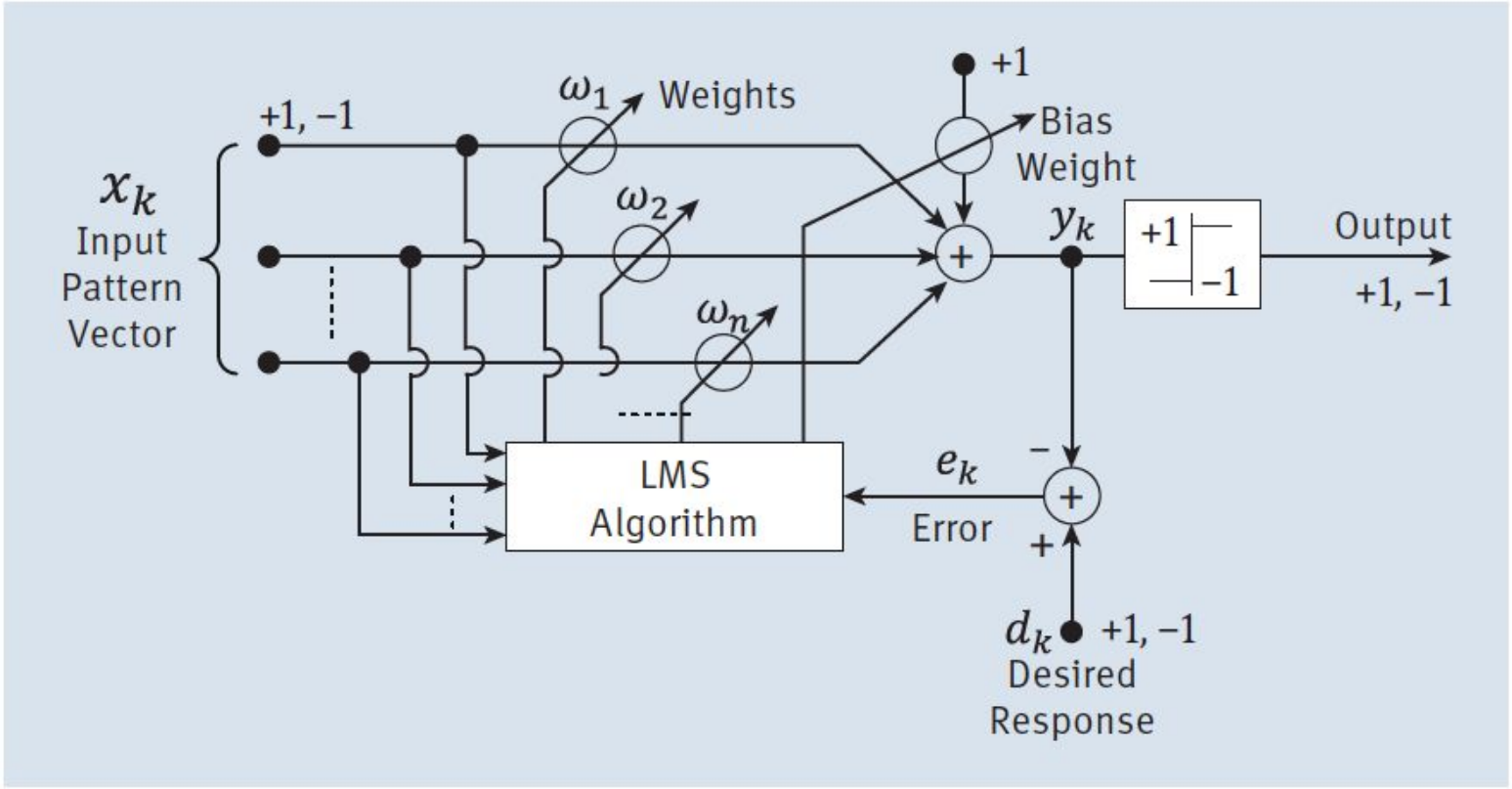


Abbildung 10.4 Adaline-Schaltdiagramm von Widrow und Hoff[1]

10.5.1 1980: Fukushimas Neocognitron

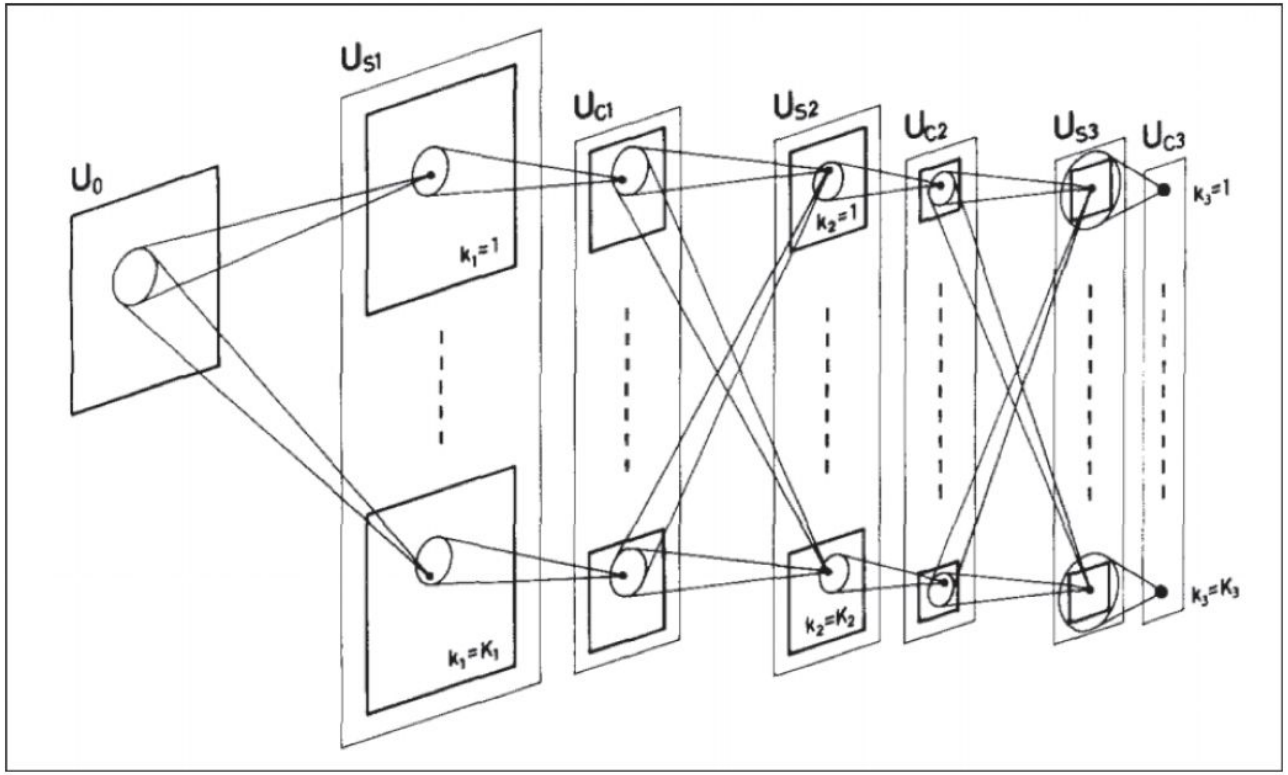


Abbildung 10.5 Schematisches Bild der Verschaltungen im Neocognitron von Fukushima (Quelle: https://www.rctn.org/bruno/public/papers/Fukushima1980.pdf)

10.5.1 1980: Fukushimas N

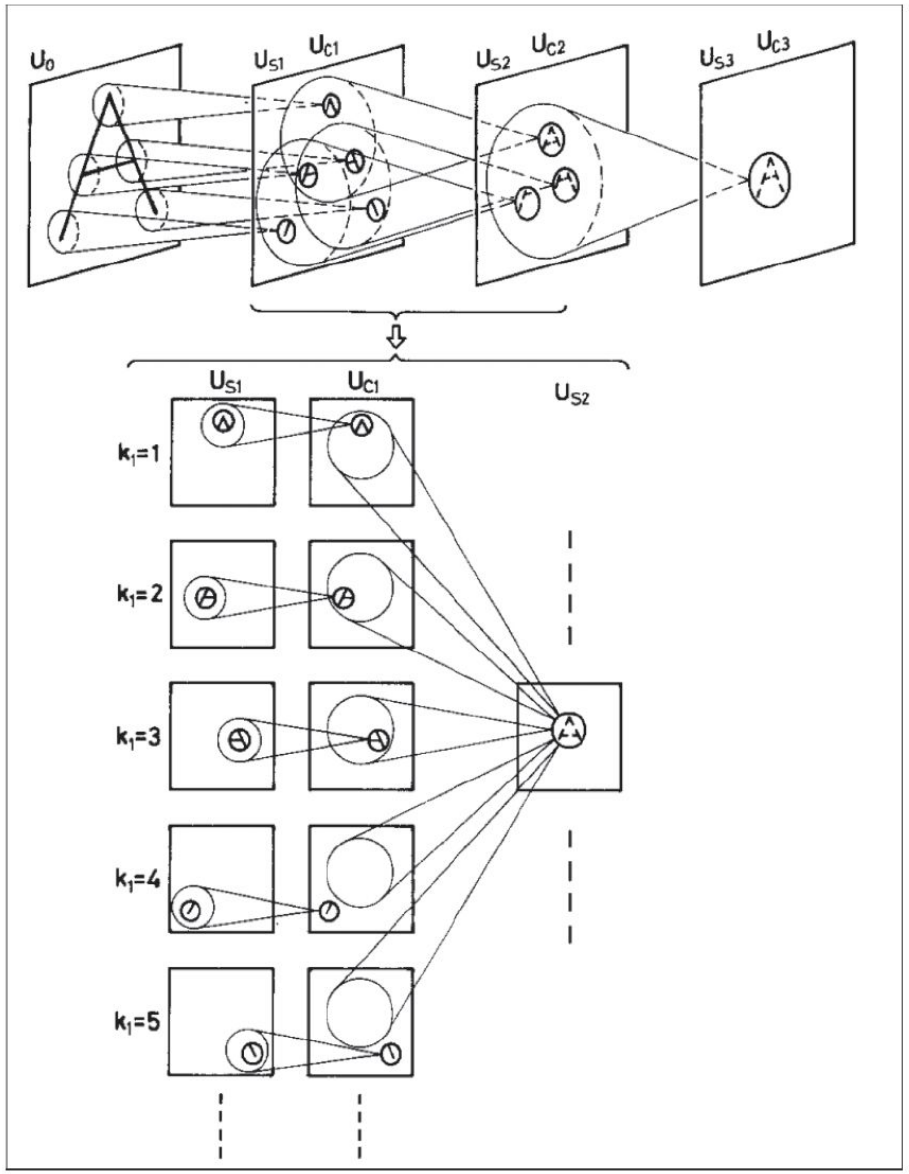


Abbildung 10.6 Neocognitron-Beispiel für die Extraktion von Features[2]

10.5.1 1980: Fukushimas Neocognitron

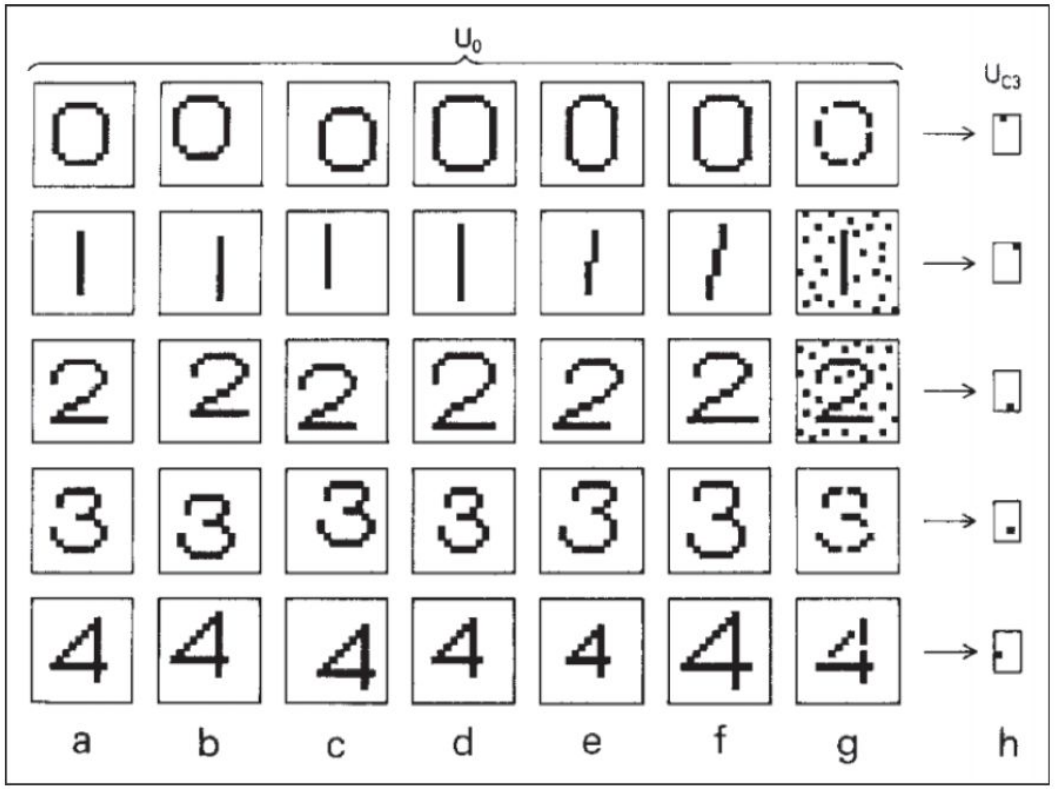


Abbildung 10.7 Neocognitron-Beispiel von unterschiedlichen Variationen und die Antwort im höchsten Layer[3]

10.5.2 1982: John Hopfield

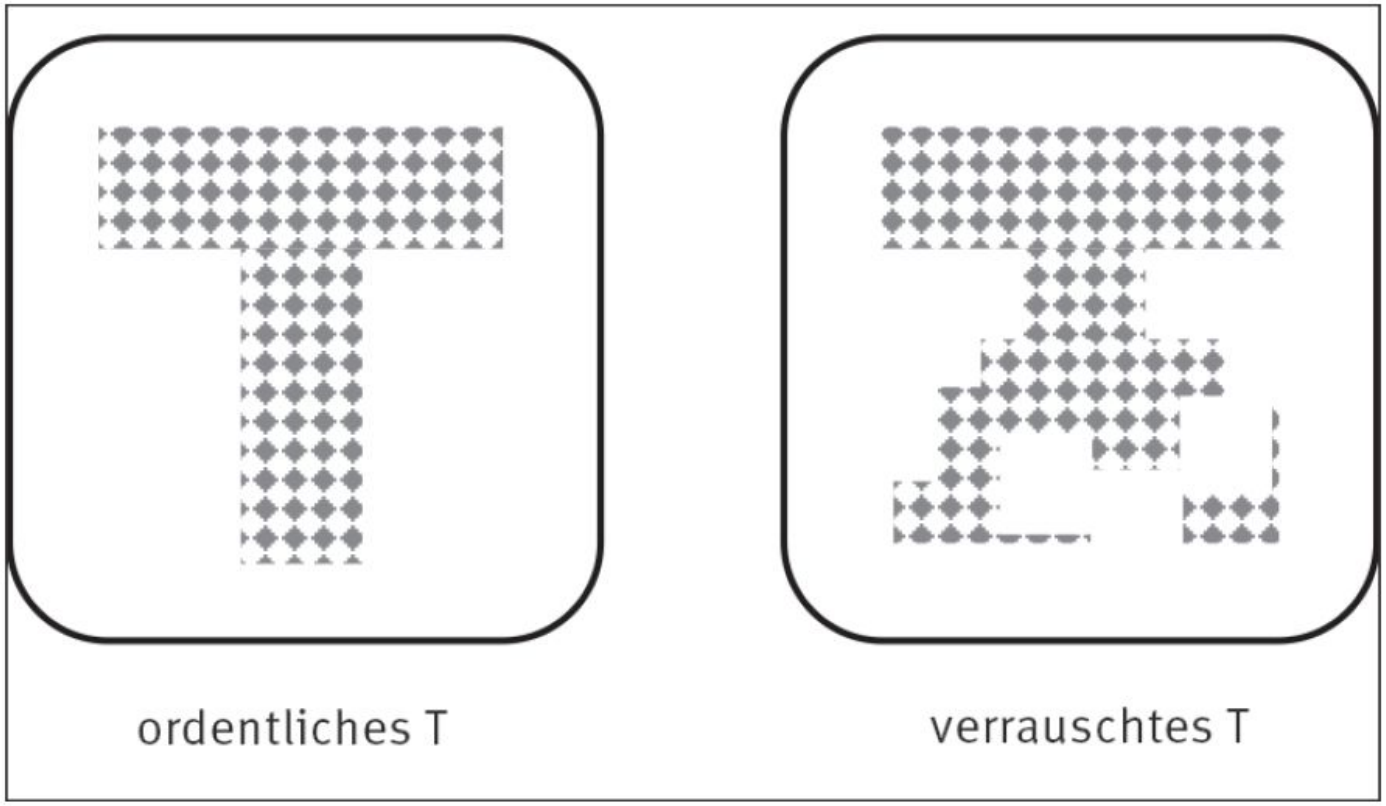


Abbildung 10.8 Assoziativer Speicher

10.5.2 1982: John Hopfield

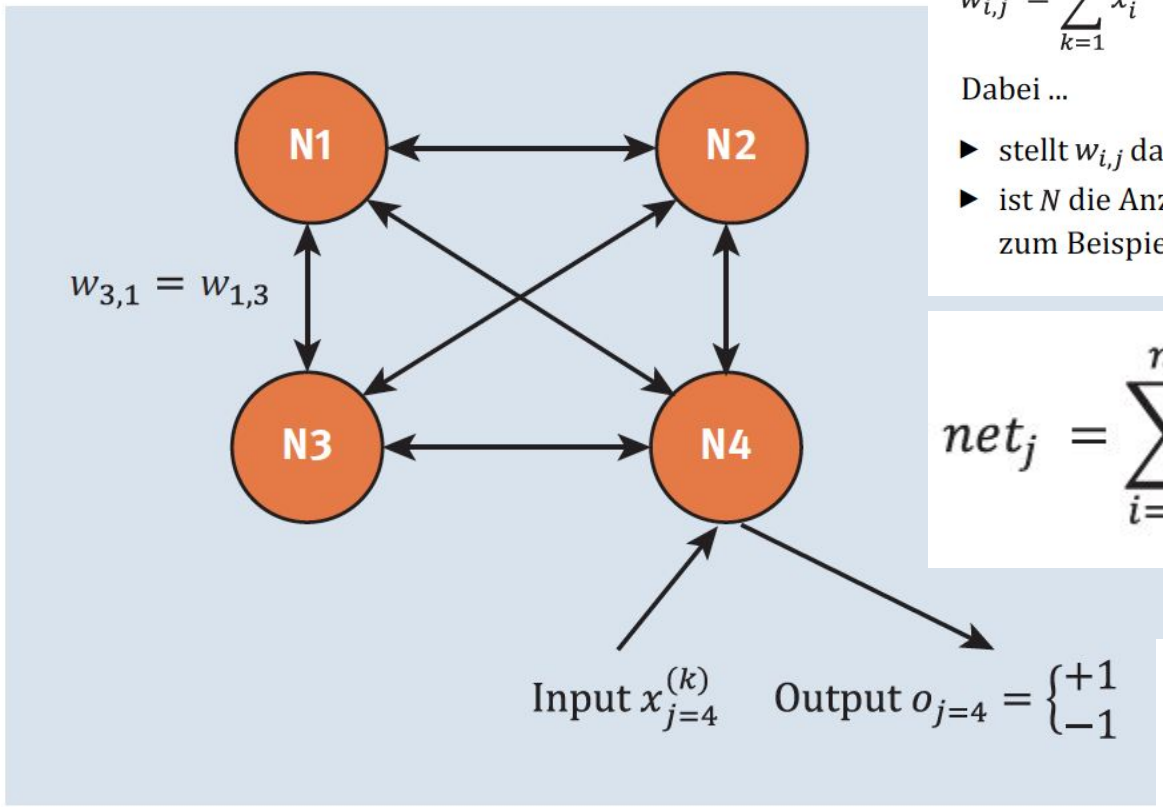


Abbildung 10.9 Beispiel eines binären Hopfield-Netzwerks

$$w_{i,j} = \sum_{k=1}^{N} x_i^{(k)} \cdot x_j^{(k)} w_{i,j}$$

Dabei ...

- stellt $w_{i,j}$ das Gewicht der Verbindung von Neuron i nach Neuron j dar,
- ist N die Anzahl der Inputs, die wie bereits gewohnt als Vektor repräsentiert werden, zum Beispiel der k-te Input $\vec{x}^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, \ldots\right)$.

$$net_j = \sum_{i=1}^{n} w_{i,j} \cdot o_i$$

$$f_{\text{step}}(net_j) = o_j = \begin{cases} +1, & \text{falls } net_j \geq \theta_j \\ -1, & \text{falls } net_j < \theta_j \end{cases}$$

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

10.5.2 1982: John Hopfield Beispiel

$$\vec{x}^{(\mathrm{T})} = (+1, +1, +1, -1, +1, -1, -1, +1, -1)$$

und das U durch den Vektor

$$\vec{x}^{(\mathrm{U})} = (+1, -1, +1, +1, -1, +1, +1, +1, +1)$$

repräsentiert.

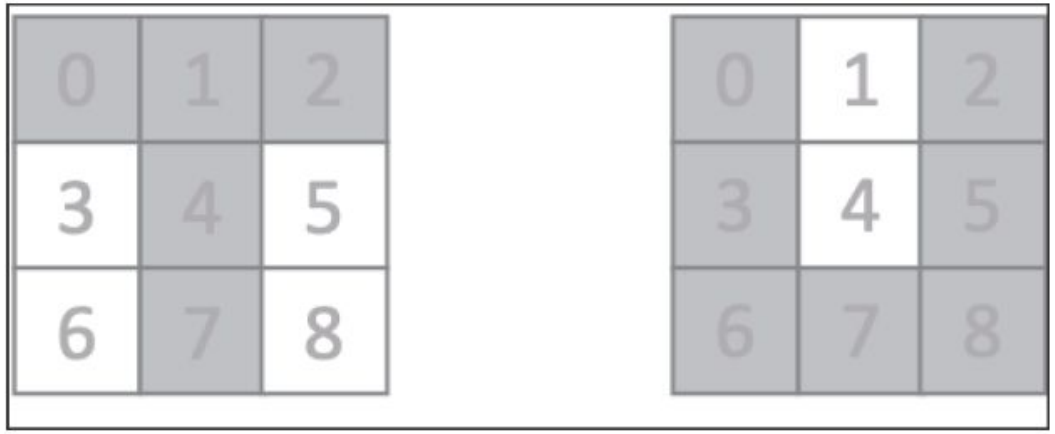


Abbildung 10.10 Links ein T und rechts ein U mit den Indizes für den Vektor

10.5.2 1982: John Hopfield Beispiel

$$W_{\mathrm{T}} = \begin{pmatrix} 0 & 1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 0 & -1 & 1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 0 & -1 & 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 0 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 0 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 & -1 & 1 & 0 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 0 \end{pmatrix}$$

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Listing 10.1/2/3 Die Multiplikation des transponierten T/U-Vektors mit sich selbst ergibt eine Matrix und W = WT + WU

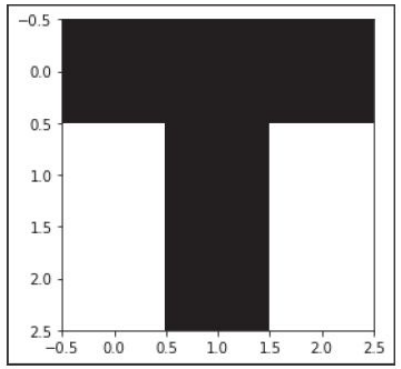
Listing 10.4 Lernen und Auswertung mit Hopfield-Netzen

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

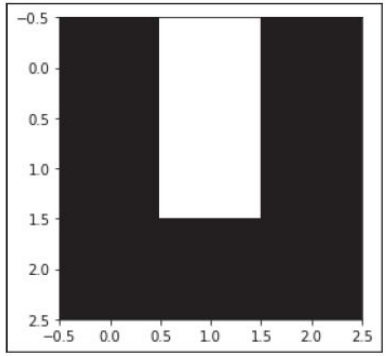
10.5.2 1982: John Hop

Die Ausgabe sieht so aus:

T - Muster



U - Muster



T - Matrix minus Einheitsmatrix

```
[[ 0.  1.  1. -1.  1. -1. -1.  1. -1.]
 [ 1.  0.  1. -1.  1. -1. -1.  1. -1.]
 [ 1.  1.  0. -1.  1. -1. -1.  1. -1.]
 [-1. -1. -1.  0. -1.  1.  1. -1.  1.]
 [ 1.  1.  1. -1.  0. -1. -1.  1. -1.]
 [-1. -1. -1.  1. -1.  0.  1. -1.  1.]
 [-1. -1. -1.  1. -1.  1.  0. -1.  1.]
 [ 1.  1.  1. -1.  1. -1. -1.  0. -1.]
 [-1. -1. -1.  1. -1.  1.  1. -1.  0.]]
```

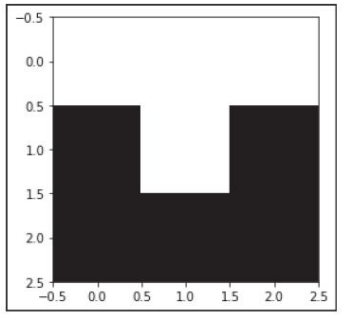
U - Matrix minus Einheitsmatrix

```
[[ 0. -1.  1.  1. -1.  1.  1.  1.  1.]
 [-1.  0. -1. -1.  1. -1. -1. -1. -1.]
 [ 1. -1.  0.  1. -1.  1.  1.  1.  1.]
 [ 1. -1.  1.  0. -1.  1.  1.  1.  1.]
 [-1.  1. -1. -1.  0. -1. -1. -1. -1.]
 [ 1. -1.  1.  1. -1.  0.  1.  1.  1.]
 [ 1. -1.  1.  1. -1.  1.  0.  1.  1.]
 [ 1. -1.  1.  1. -1.  1.  1.  0.  1.]
 [ 1. -1.  1.  1. -1.  1.  1.  1.  0.]]
```

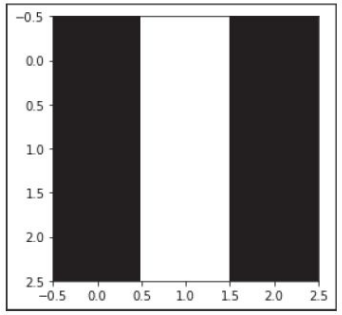
T + U - Matrix

```
[[ 0.  0.  2.  0.  0.  0.  0.  2.  0.]
 [ 0.  0.  0. -2.  2. -2. -2.  0. -2.]
 [ 2.  0.  0.  0.  0.  0.  0.  2.  0.]
 [ 0. -2.  0.  0. -2.  2.  2.  0.  2.]
 [ 0.  2.  0. -2.  0. -2. -2.  0. -2.]
 [ 0. -2.  0.  2. -2.  0.  2.  0.  2.]
 [ 0. -2.  0.  2. -2.  2.  0.  0.  2.]
 [ 2.  0.  2.  0.  0.  0.  0.  0.  0.]
 [ 0. -2.  0.  2. -2.  2.  2.  0.  0.]]
```

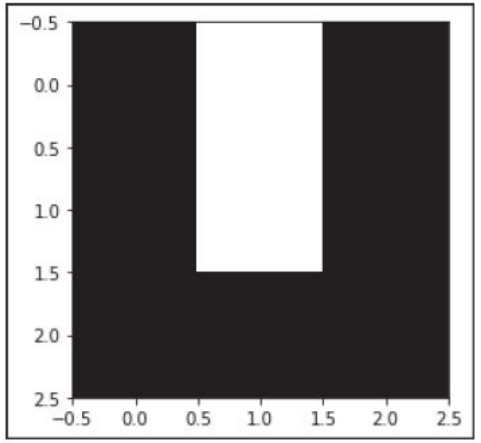
Inputvektor



Schritt 0



Schritt 1



www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

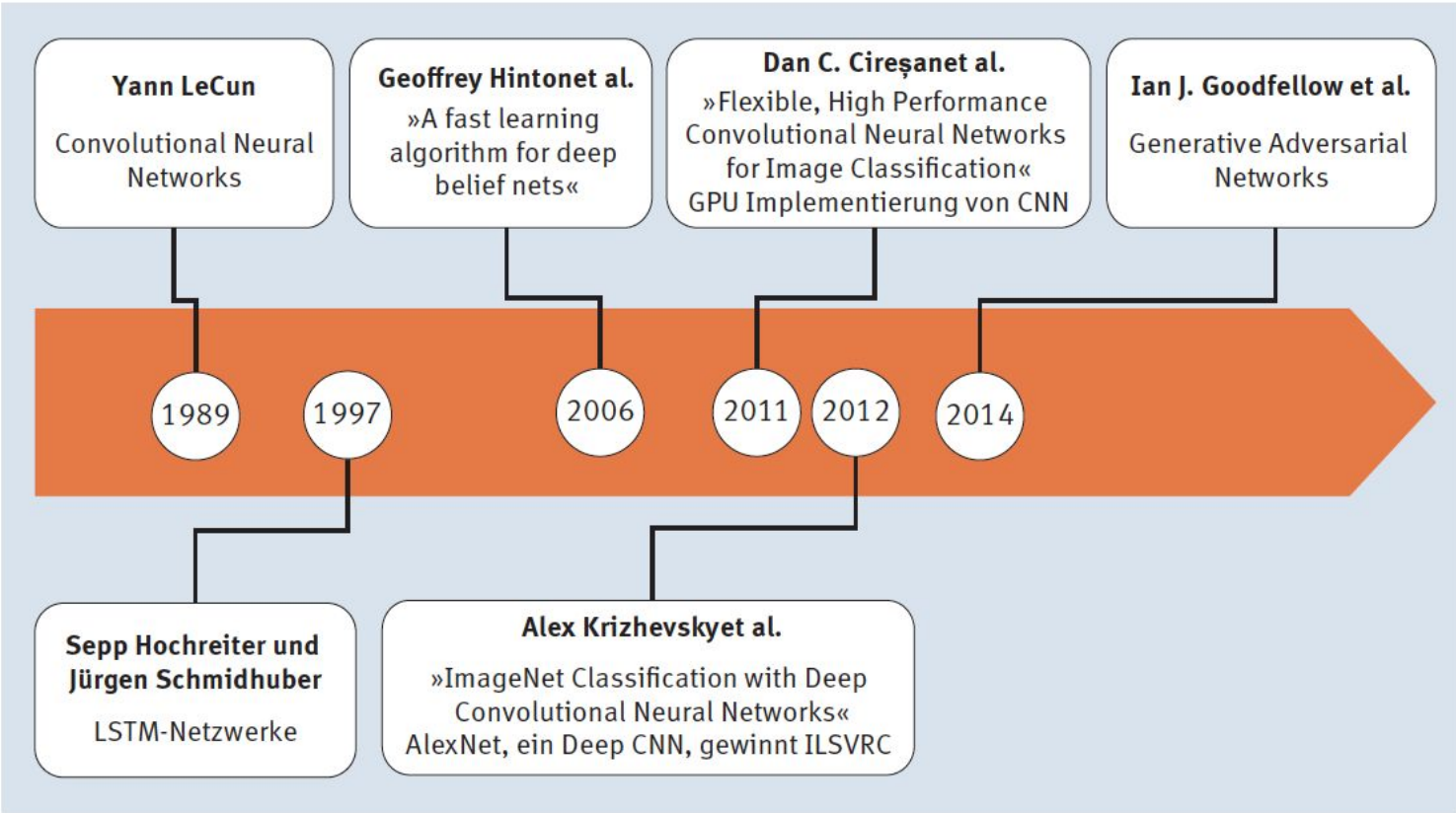


Abbildung 10.11 KNN-Geschichte, Teil 2

10.8.1 2014: Ian J. Goodfellow et al. – Generative Adversarial Networks (GAN)

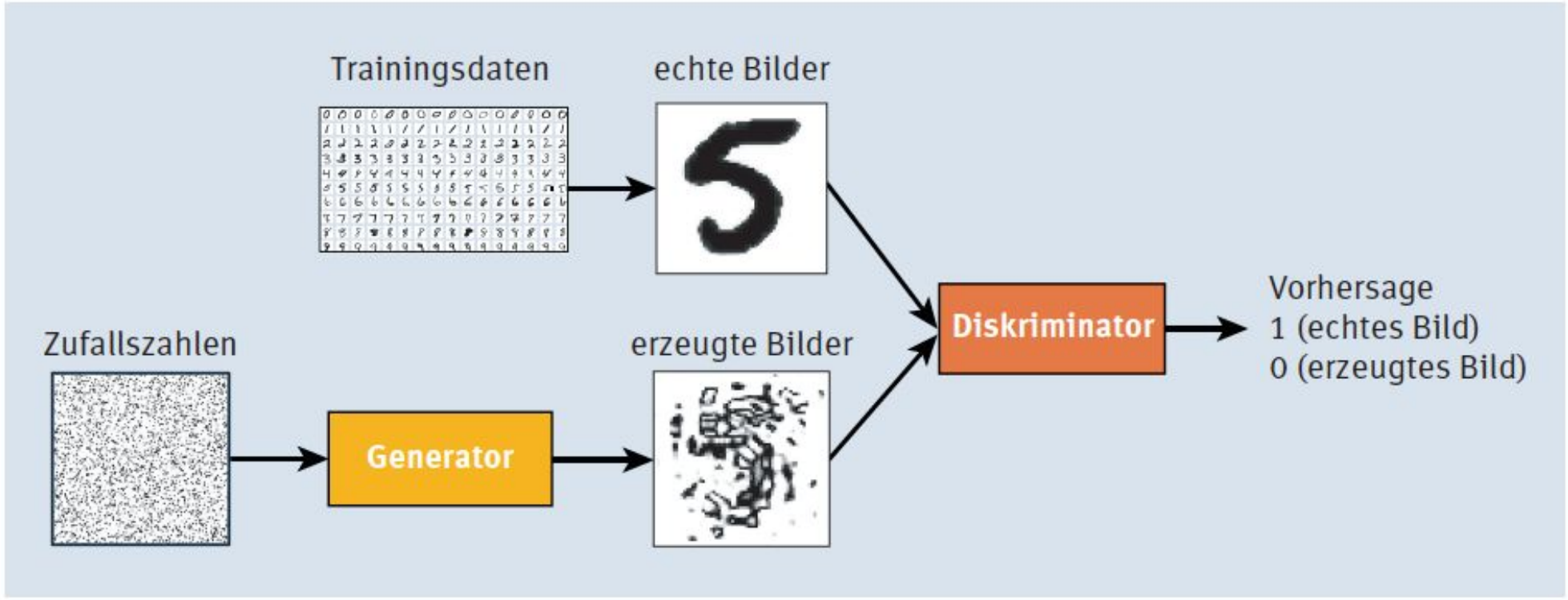


Abbildung 10.12 GAN – prinzipieller Aufbau

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

10.8.1 2014: Ian J. Goodfellow et al. – Generative Adversarial Networks (GAN)

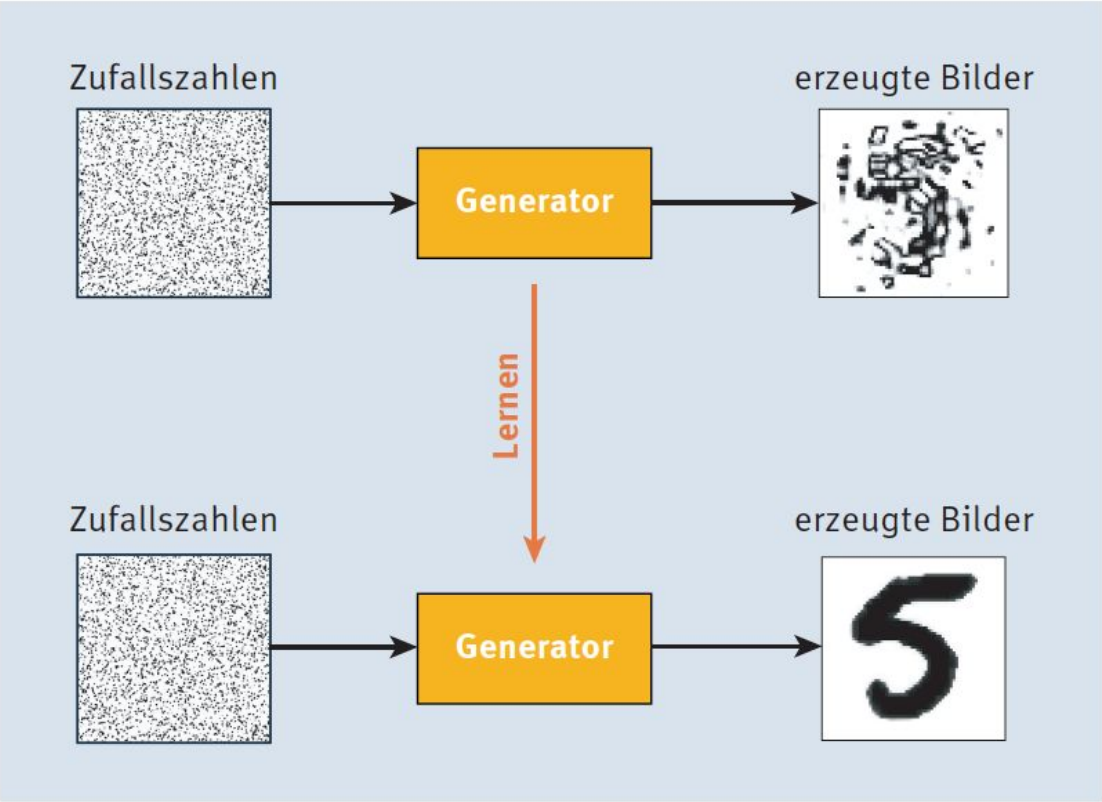


Abbildung 10.13 Der Generator lernt, Bilder zu erzeugen.

10.8.1 2014: Ian J. Goodfellow et al. – Generative Adversarial Networks (GAN)

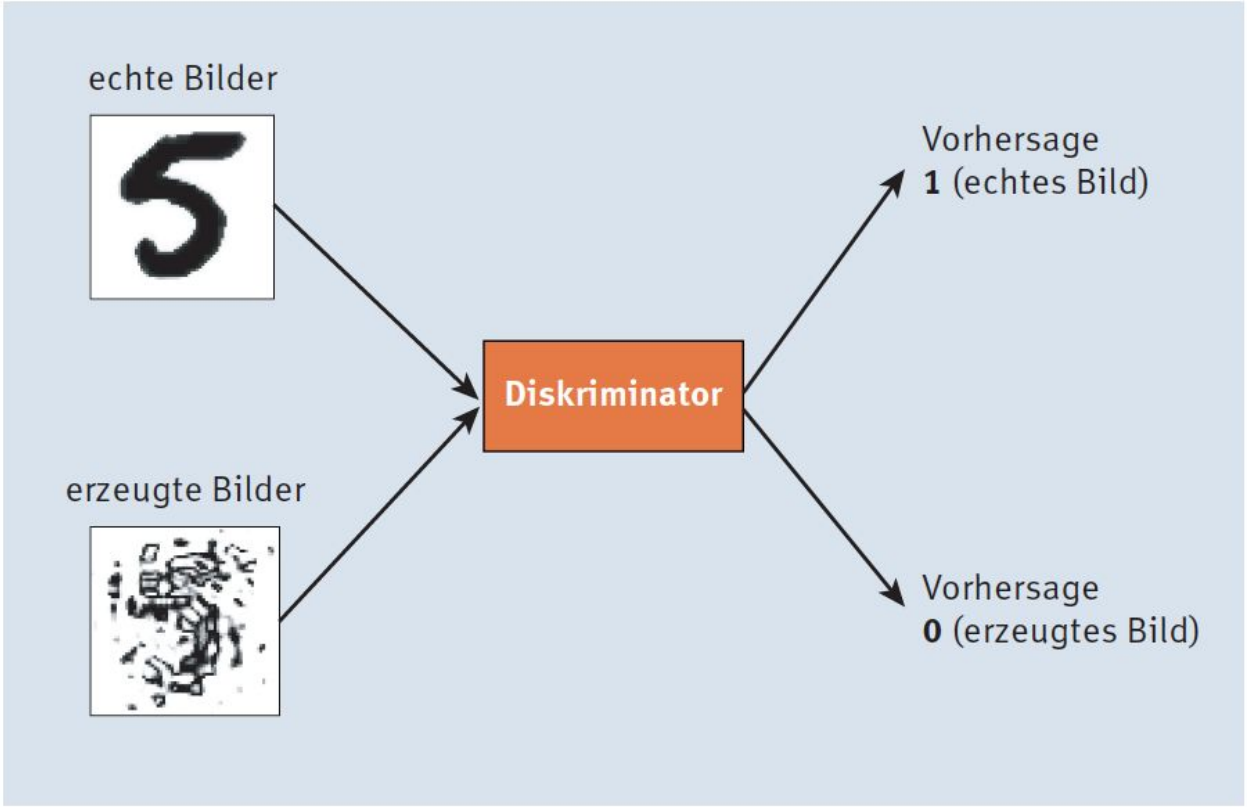


Abbildung 10.14 Der Diskriminator lernt, Bilder zu unterscheiden.

Kapitel 11
Der Machine-Learning-Prozess

11.1 Das CRISP-DM-Modell

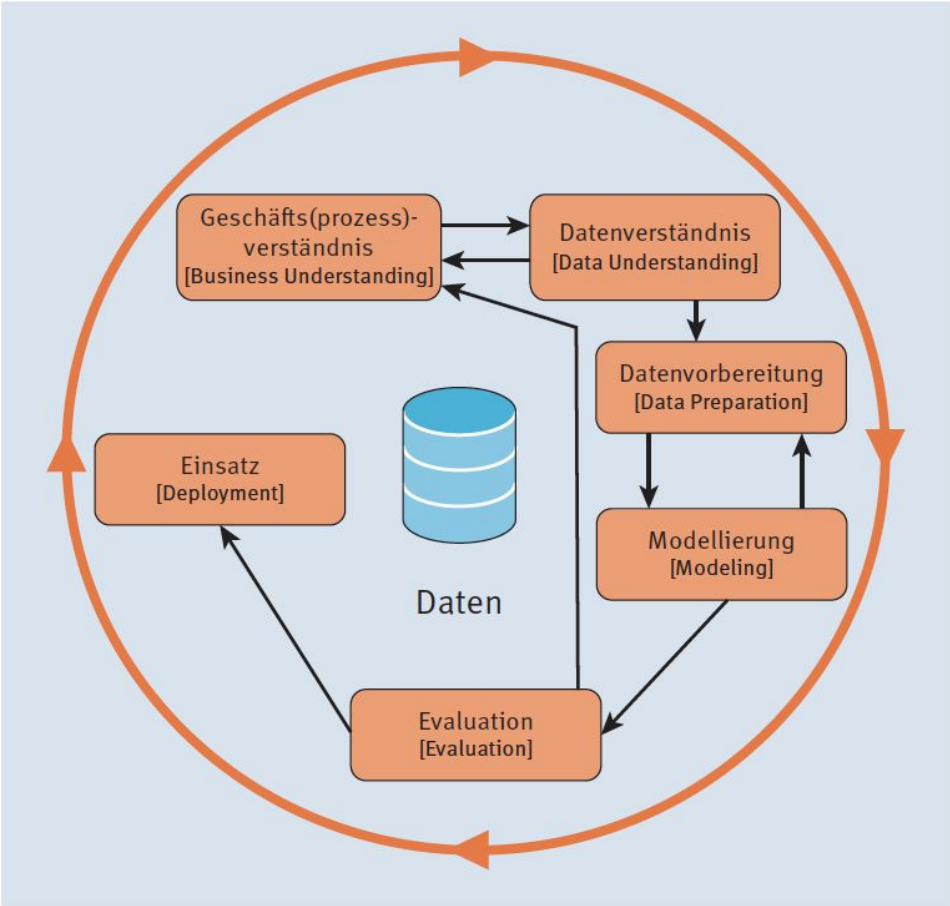


Abbildung 11.1 Das CRISP-DM-Modell (Quelle: Wikimedia Commons)

11.2 Feature Engineering

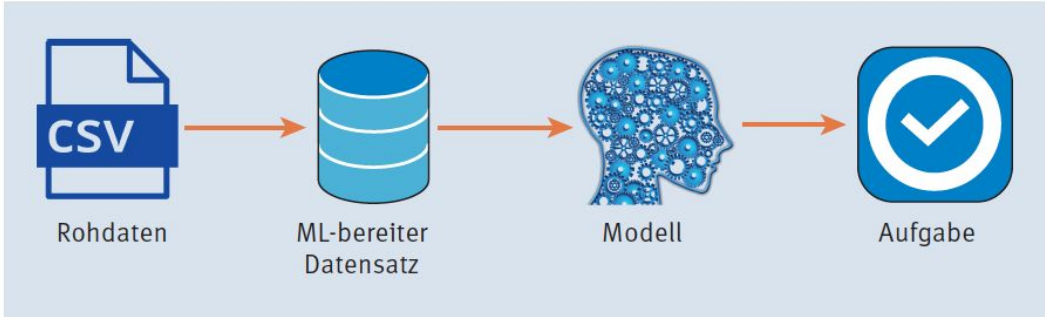


Abbildung 11.2 Gewünschter Ablauf eines Machine-Learning-Projekts

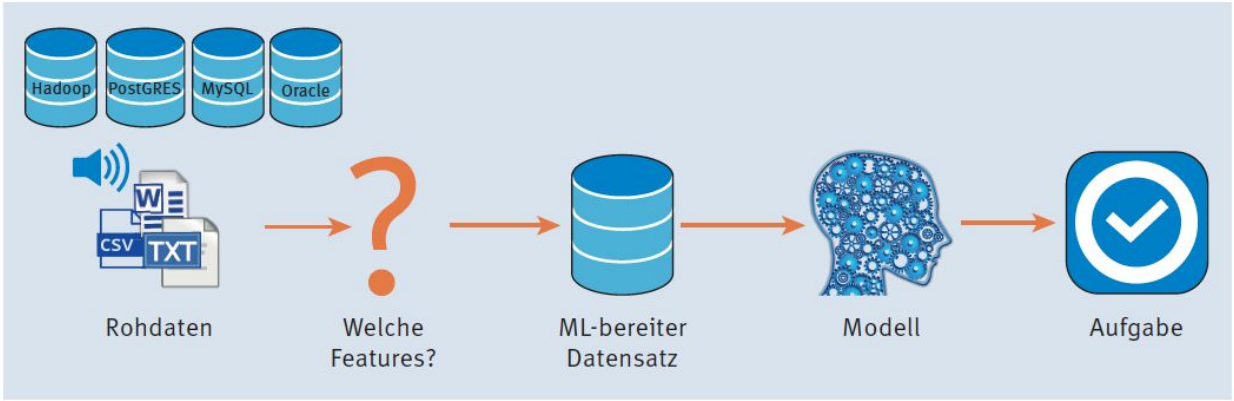


Abbildung 11.3 Realität eines Machine-Learning-Projekts

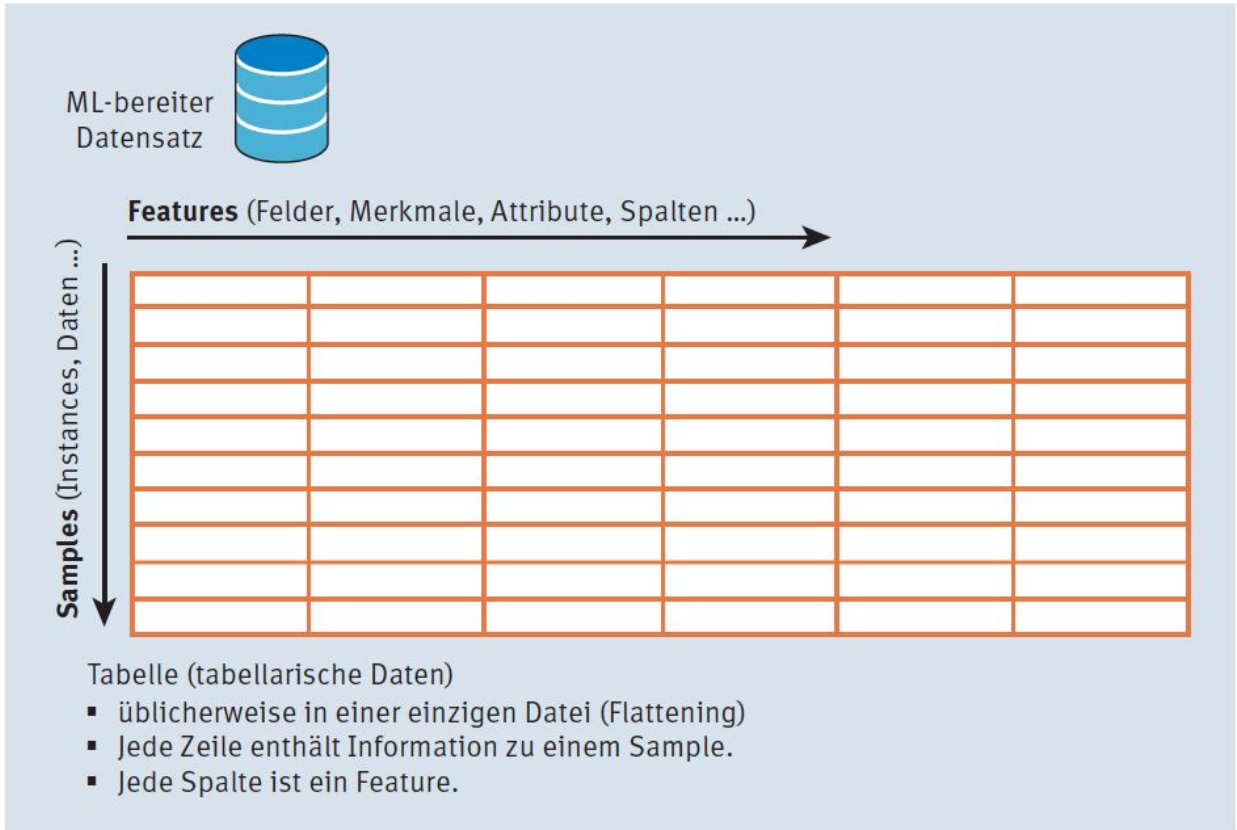


Abbildung 11.4 Für ML (Machine Learning) bereiter Datensatz

11.2.1 Feature-Kodierung

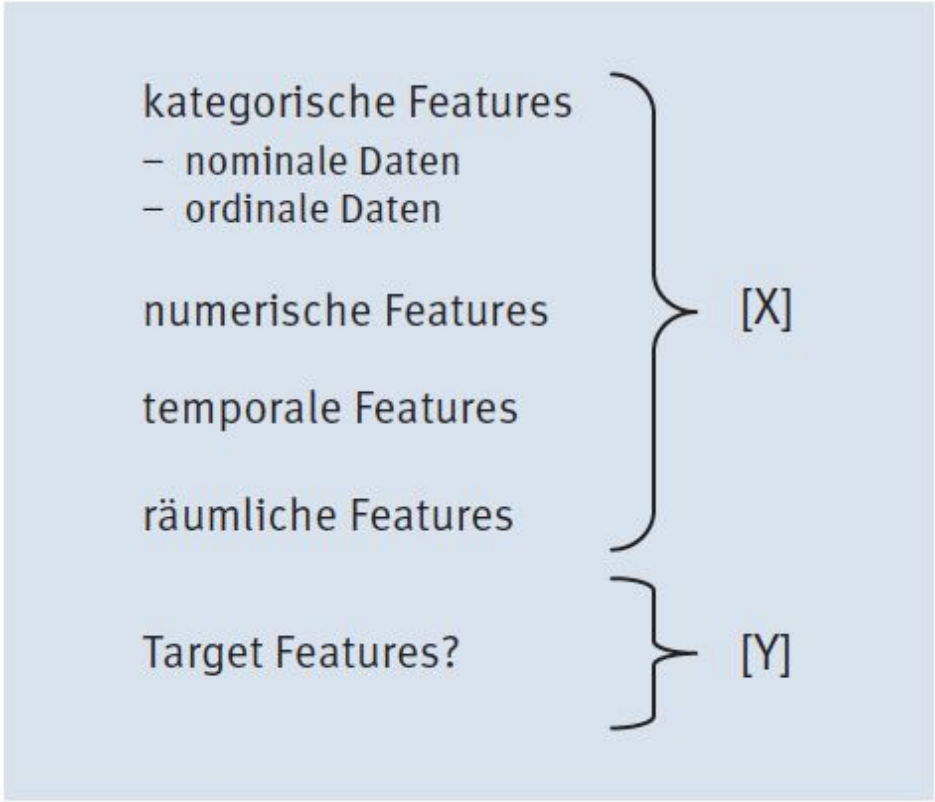


Abbildung 11.5 Einteilung des Formats der Rohdaten

Kategorisches Feature
Label Encoding

sehr gut	1
gut	2
befriedigend	3
genügend	4
nicht genügend	5

Tabelle 11.1 Beispiel für Label Encoding (Schulnoten in Österreich)

Kategorisches Feature
One-Hot-Encoding

Listing 11.1 Beispiel für Label Encoding der Haarfarbe der Schüler durch »sklearn «

Listing 11.2 One-Hot-Encoding mit der scikit-learn-Bibliothek

Listing 11.3 Vorbereitung der Daten für » category_encoders«

11.2.2 Feature-Extrakti

- Datenbanken
- Textdateien
- Bilddaten
- Audiodaten

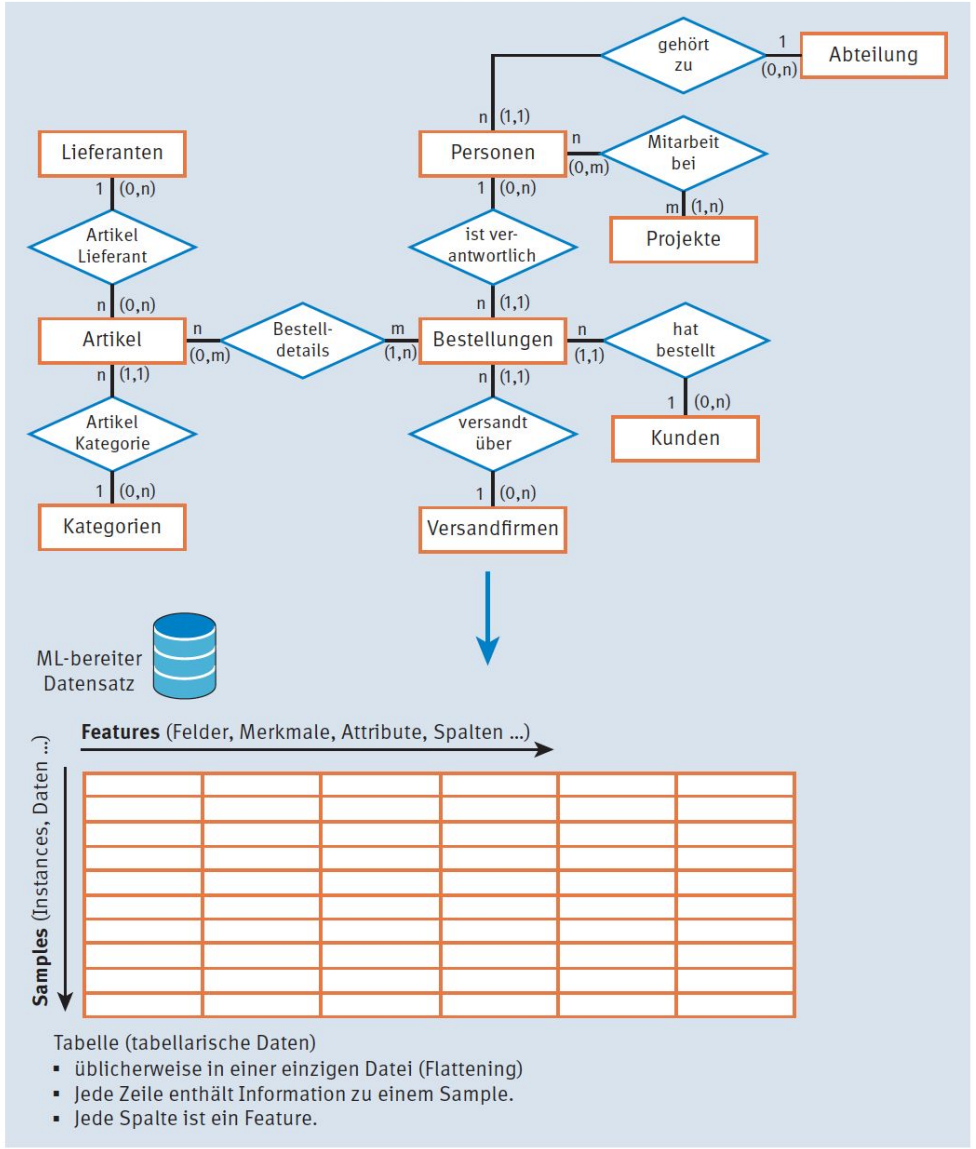


Abbildung 11.12 Datenverflachung einer Datenbank

Lernverfahren

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

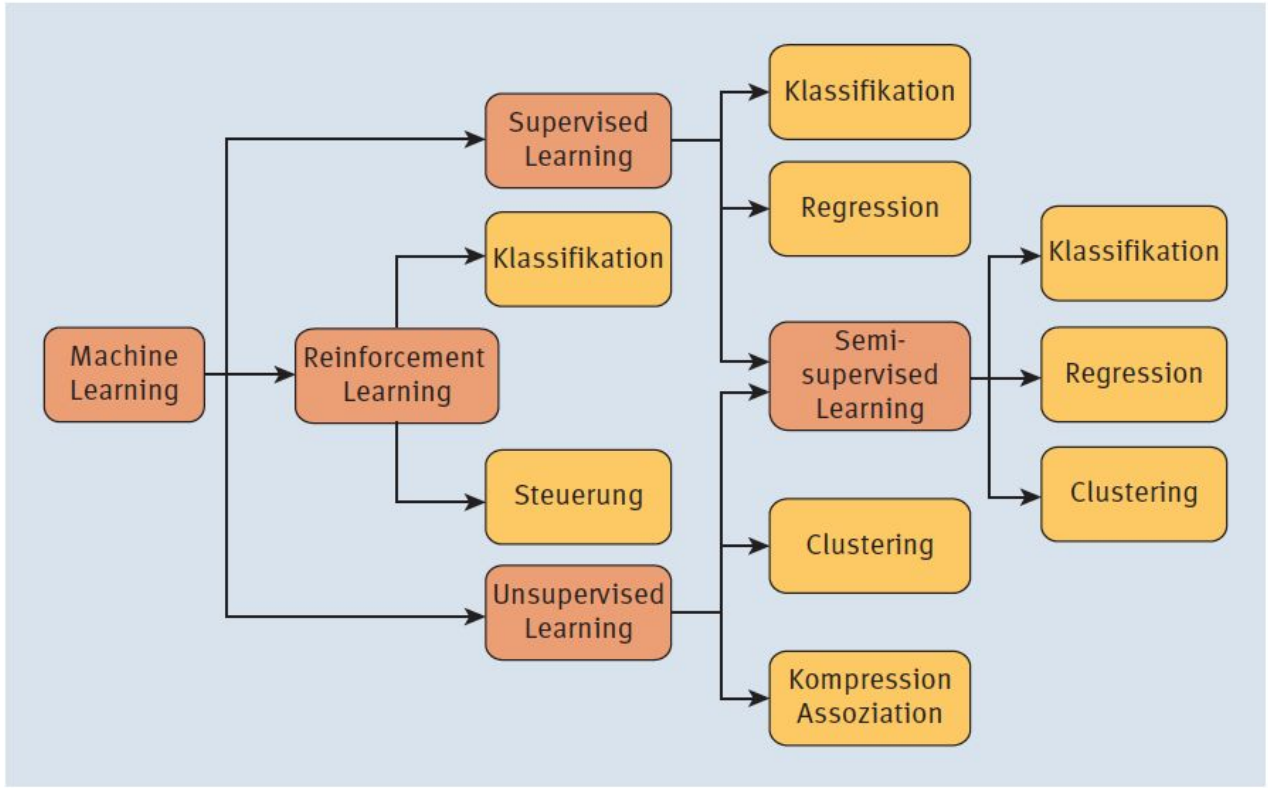


Abbildung 12.1 Lernstrategien und ihre möglichen Anwendungen

Supervised learning

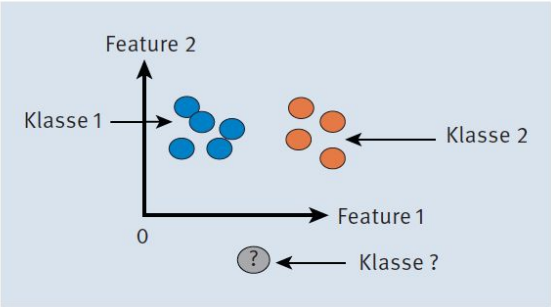


Abbildung 12.2 Klassifikation

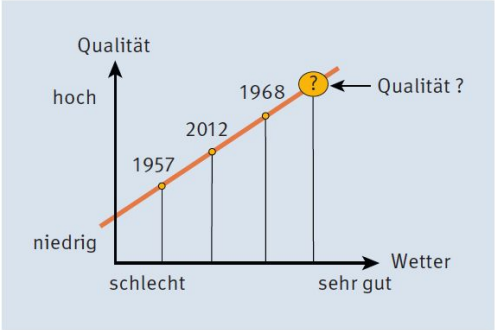


Abbildung 12.3 Wein-Qualitäts-Regression

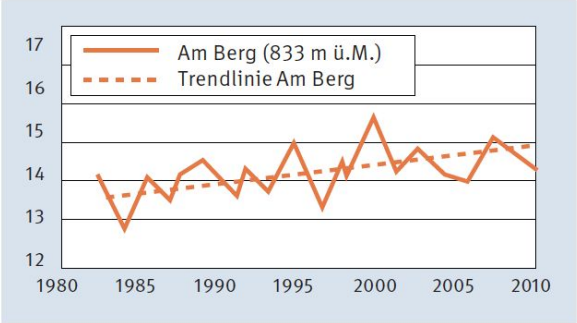


Abbildung 12.4 Schematische Darstellung der Trendanalyse des Mittelwertes der Temperatur für den Ort Am Berg

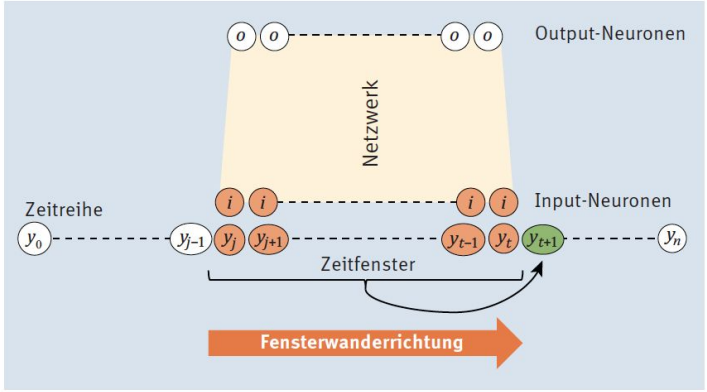


Abbildung 12.5 Zeitreihe mit Netzwerk

Unsupervised learning

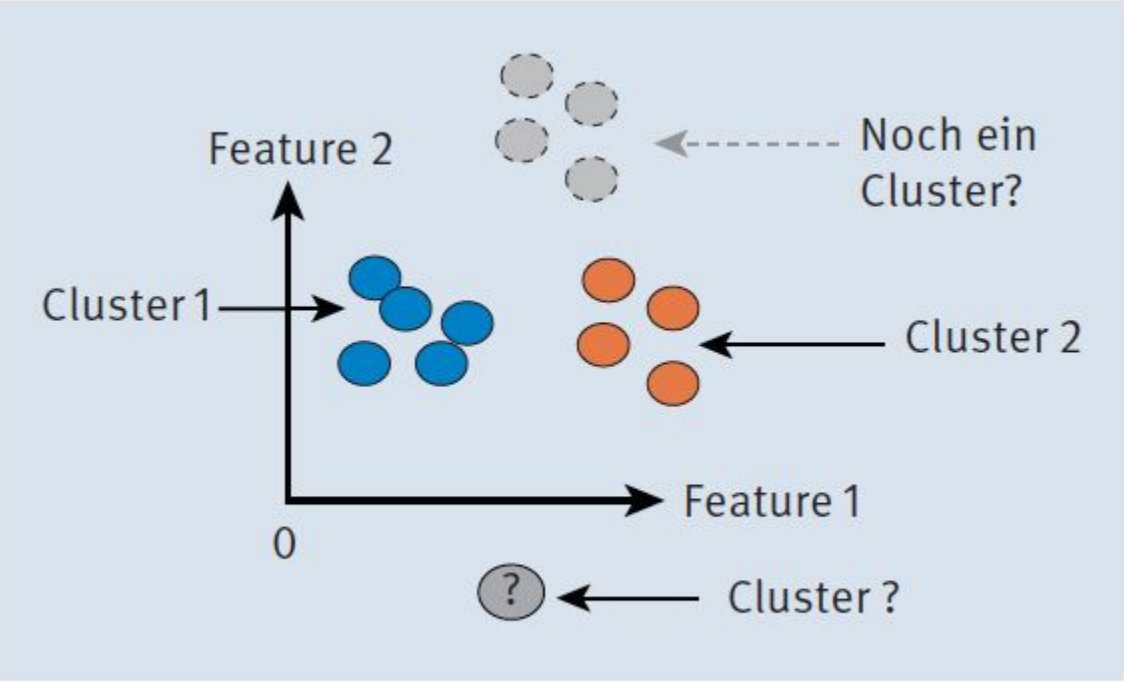


Abbildung 12.6 Cluster

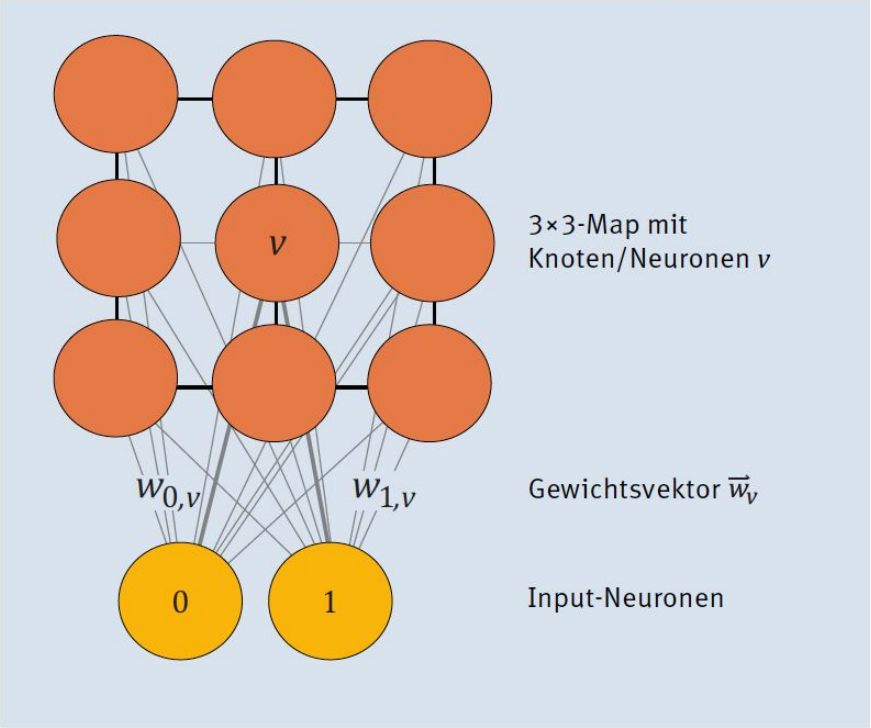


Abbildung 12.7 SOM mit zwei Input-Neuronen und einer 3×3-Map

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Algorithmus

1. Initialisiere die Gewichte der Gewichtsvektoren $\vec{w}_v$, zu jedem Knoten v mit kleinen zufälligen Werten aus dem Intervall [0,1].
2. Wiederhole für eine bestimmte Anzahl an Iterationen I:
   - Wähle einen Input-Vektor $\vec{x}(s)$ aus der Menge der Trainingsbeispiele X zufällig aus, wobei s der Iterationsschritt ist.
   - Berechne die Distanz d vom Input-Vektor zu den Gewichtsvektoren aller Knoten mit $d(\vec{x}(s), \vec{w}(s))$.
   - Bestimme den Knoten u mit der kleinsten Distanz; dieser wird auch als *Best Matching Unit* (*BMU*) bezeichnet.
   - Passe den BMU-Gewichtsvektor $\vec{w}_u(s)$ und den der Knotennachbarn in Richtung des Input-Vektors an. Die Anpassung der Gewichte erfolgt anhand folgender Formel:

     $$\vec{w}_v(s+1) = \vec{w}_v(s) + \theta(u, v, s) \cdot \alpha(s) \cdot \left(\vec{x}(s) - \vec{w}_v(s)\right)$$

Dabei haben die unterschiedlichen Elemente der Formel folgende Bedeutung:

- $\vec{w}_v(s)$ – der Gewichtsvektor für Knoten mit Index v zum Iterationsschritt s
- s – der Iterationsschritt
- u – der Index für die BMU bezüglich des Input-Vektors $\vec{x}(s)$
- $\vec{x}(s)$ – der Input-Vektor
- $\alpha(s)$ – eine Lernrate, die im Laufe der Zeit abnimmt
- $\theta(u, v, s)$ – die Nachbarschaftsfunktion zur Ermittlung der Nachbarn mit Index v für BMU mit Index u zum Iterationsschritt s (Abbildung 12.10). Mit zunehmendem Iterationsschritt s schrumpft die Nachbarschaft und bekommt Werte im Intervall [0,1].

$$d(\vec{x},\vec{w}) = \|\vec{x}-\vec{w}\|$$
$$= \sqrt{(x_1-w_1)^2+(x_2-w_2)^2+\ \ldots} = \sqrt{\sum_i (x_i-w_i)^2}$$

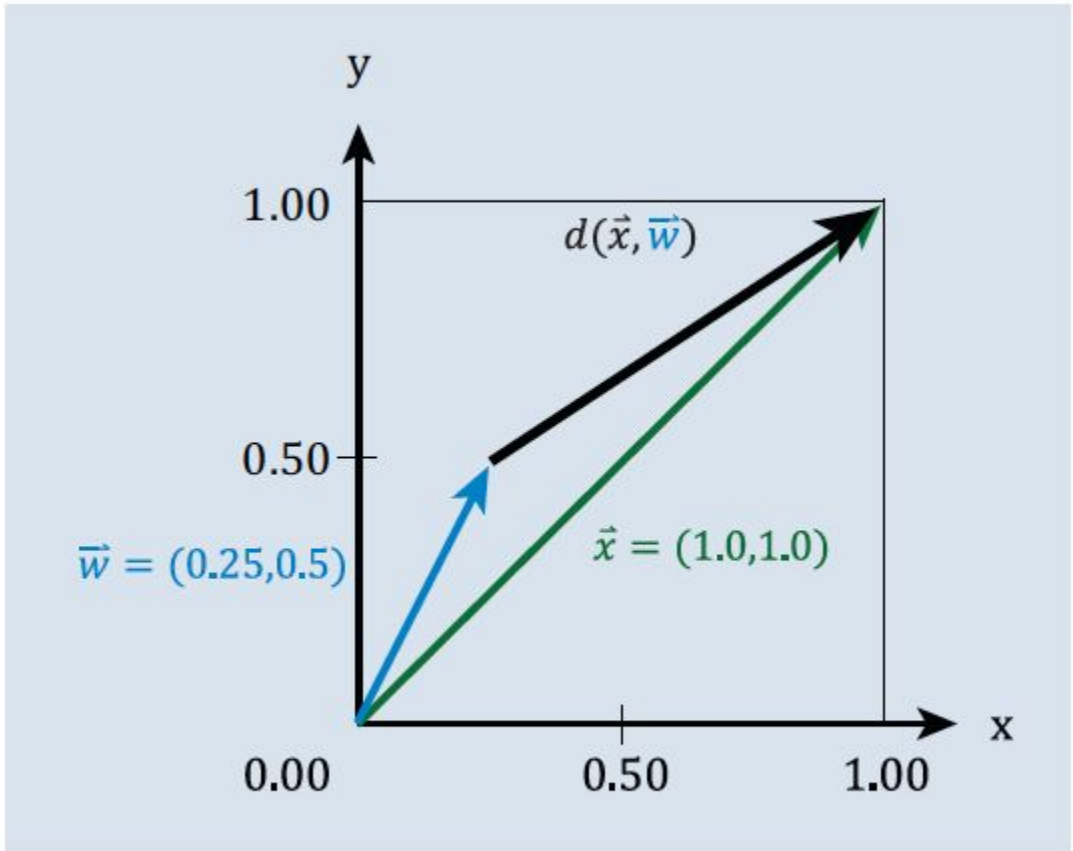


Abbildung 12.8 Euklidische Distanz von zwei Vektoren

$$\alpha(s) = \alpha(0) \cdot \frac{1}{s}$$

$$\alpha(0) = 0.80$$

$$\alpha(s, I) = \alpha(0) \cdot (1 - \frac{s}{I})$$

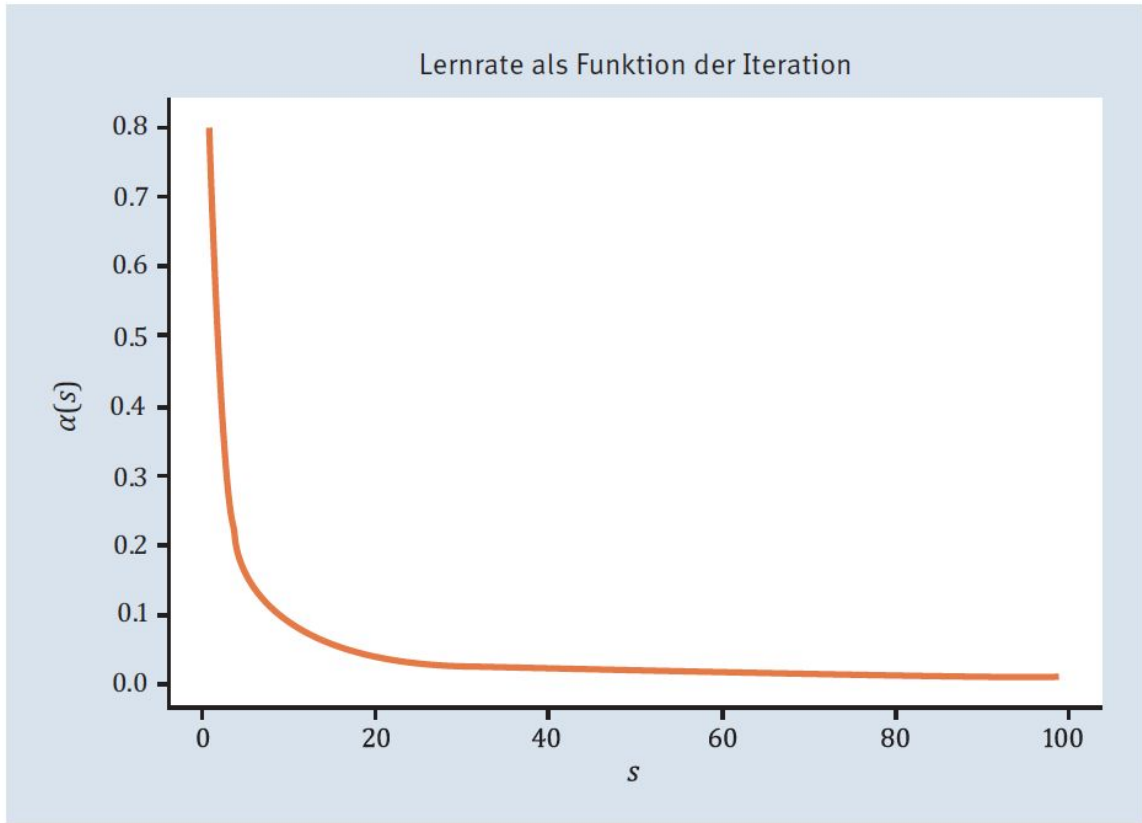


Abbildung 12.9 Abfall der Lernrate im Laufe der Zunahme der Iterationen

$$\theta(u,v,s) = \begin{cases} h_{u,v}(s), & \text{falls } d(u,v) \leq r(s,I) \\ 0, & \text{sonst} \end{cases}$$

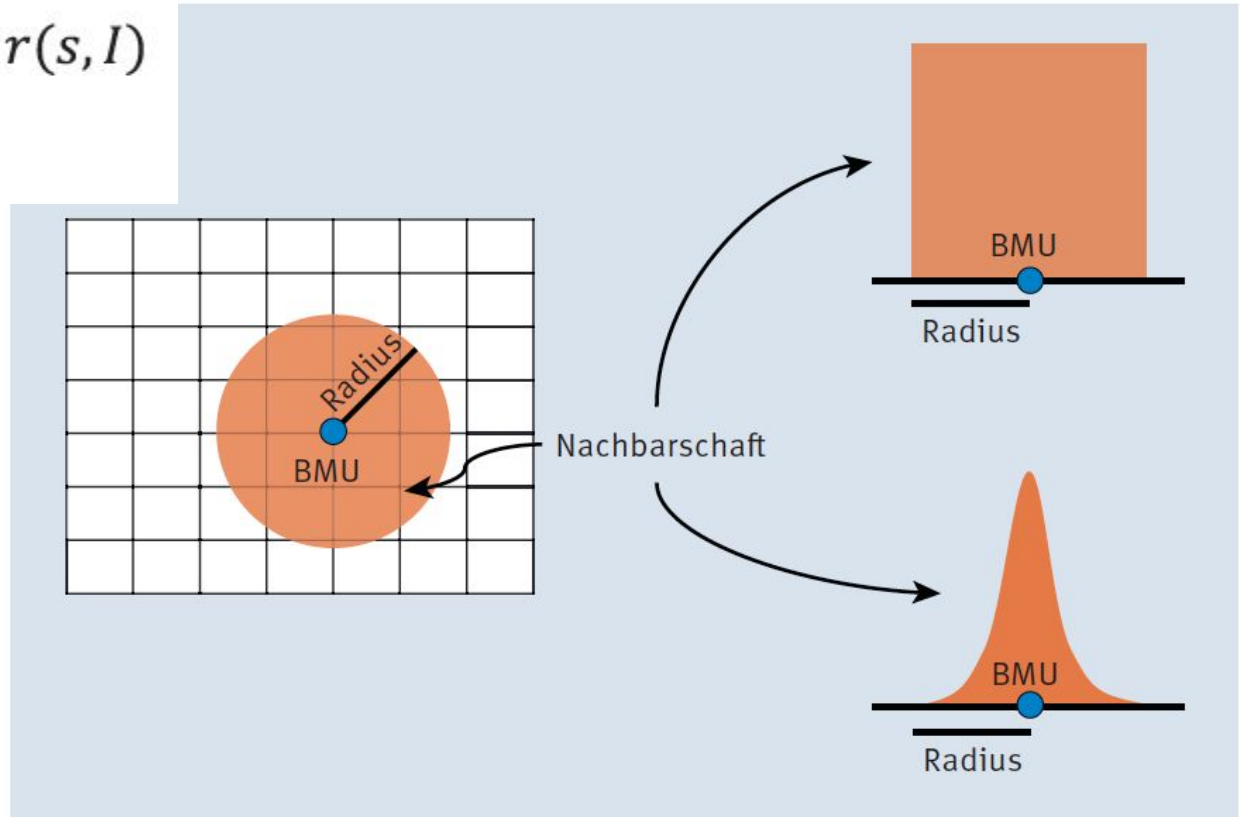


Abbildung 12.10 Unterschiedliche Nachbarschaftsfunktionen

$$r(s, I) = r(0) \cdot e^{\left(\frac{s}{I}\right)}$$

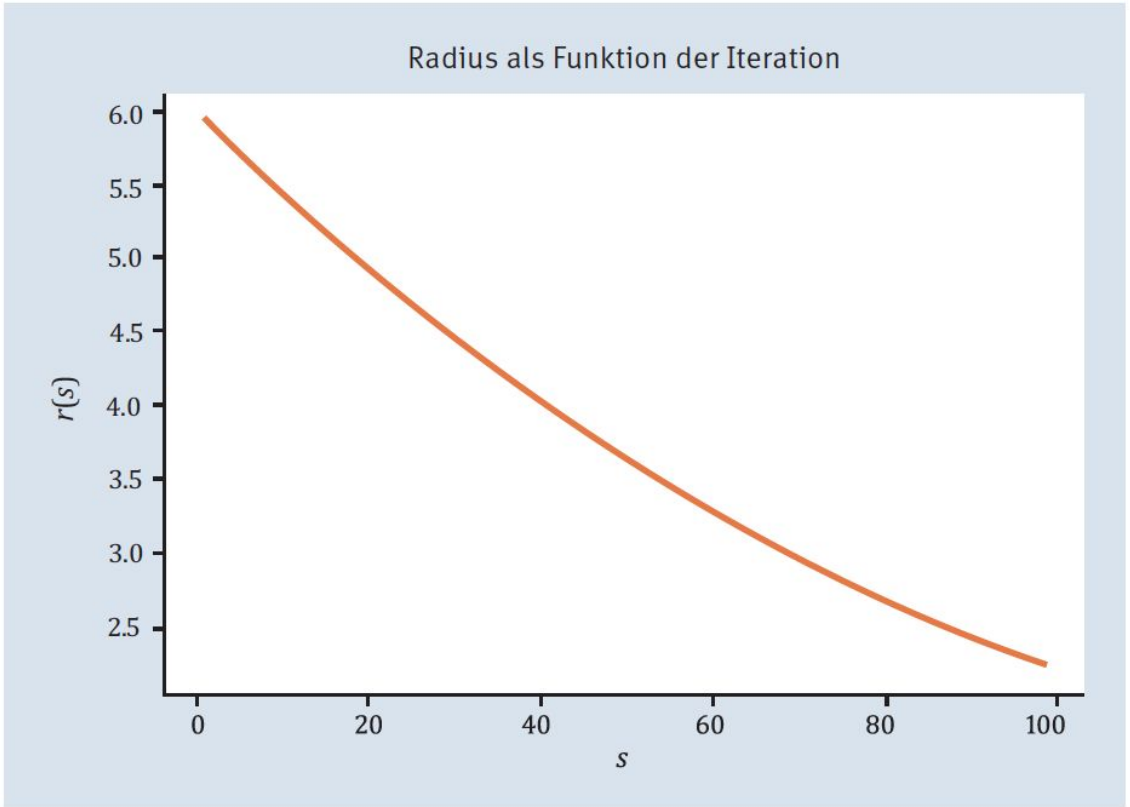


Abbildung 12.11 Radius als Funktion der Iteration

Learn

$$\vec{w}_v(s+1) = \vec{w}_v(s) + \underbrace{\theta(u,v,s)}_{\text{kleiner 1}} \cdot \underbrace{\alpha(s)}_{\text{kleiner 1}} \cdot \underbrace{\left(\vec{x}(s) - \vec{w}_v(s)\right)}_{\text{Vektor zur BMU}}$$

Program

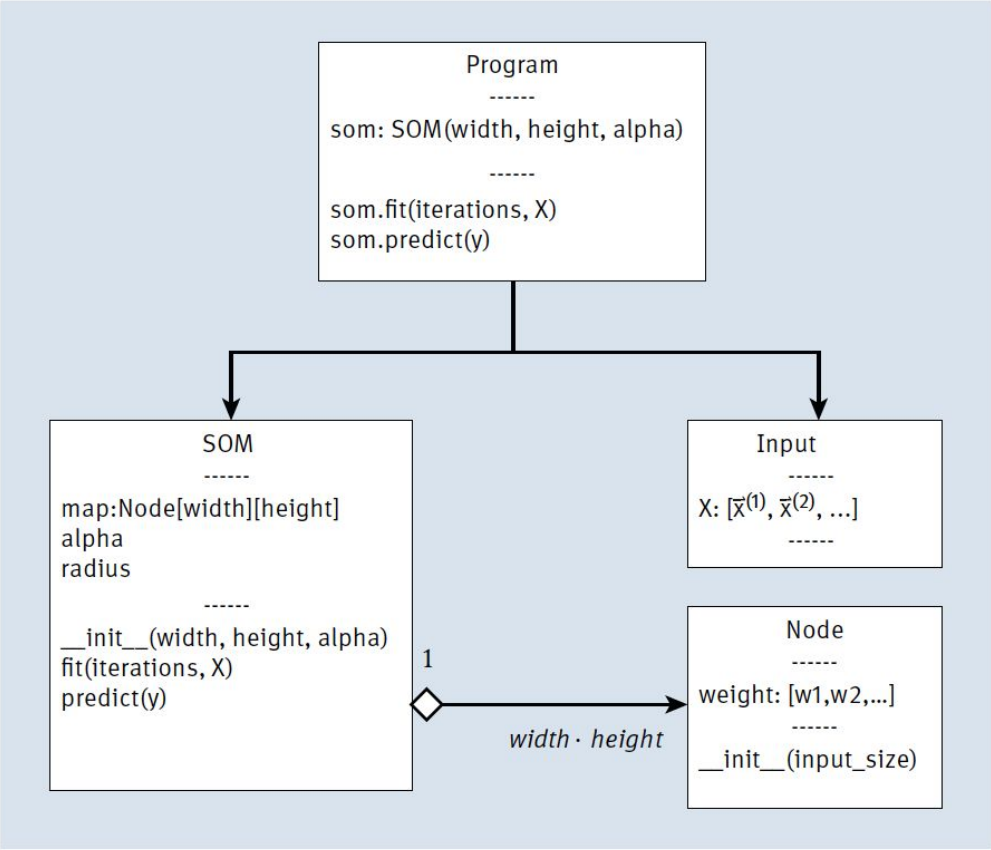


Abbildung 12.12 Die Bausteine zur SOM und ihre Zusammenhänge

Listing 12.2 SOM-Implementierung

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

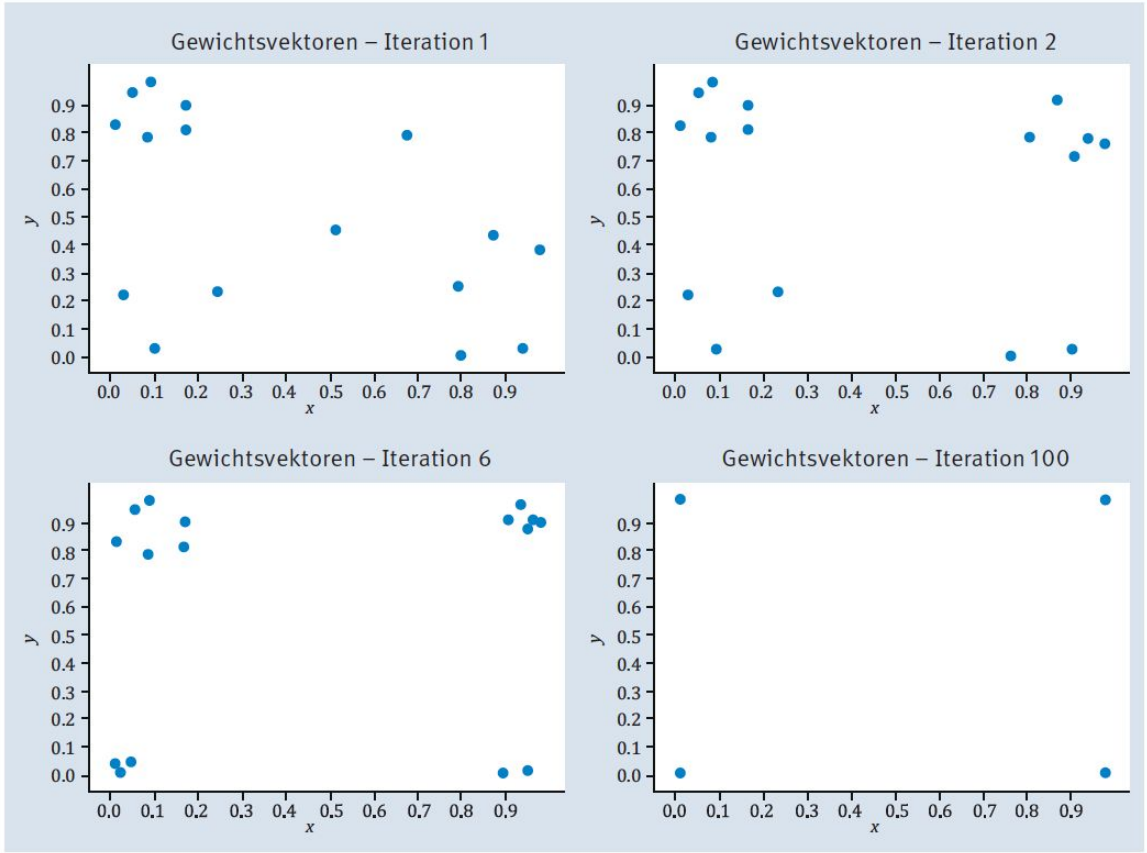
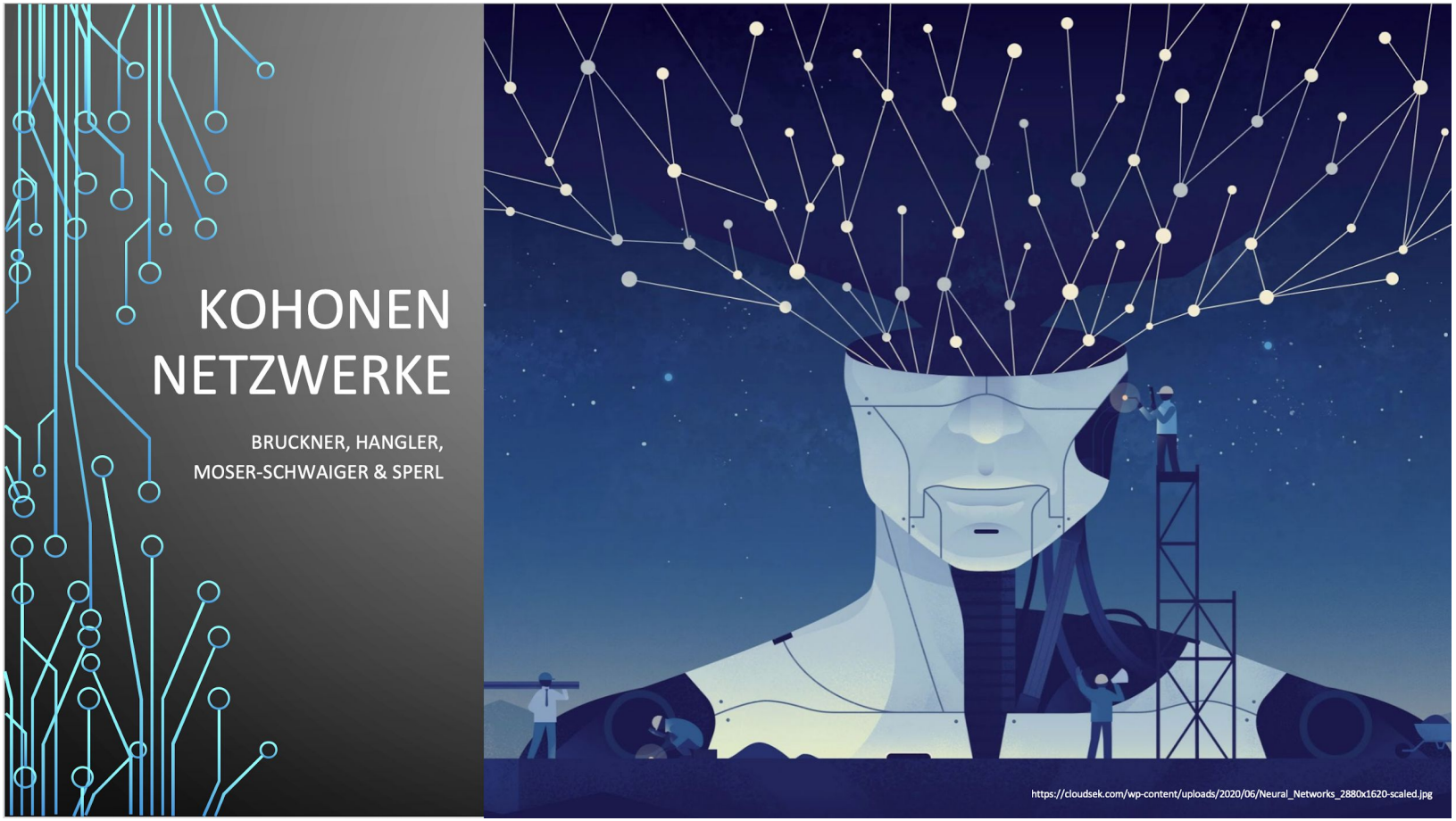


Abbildung 12.13 SOM und die Clusterbildung während des Lernvorgangs

Kohonen
COWI presentation



www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

IEC
Institute of Electronics and Computer

Journal of Artificial Intelligence and Systems, 2020, 2, 1-13
https://iecscience.org/journals/AIS
ISSN Online: 2642-2859

EEG signal based Modified Kohonen Neural Networks for Classification of Human Mental Emotions

D. Jude Hemanth[1,*]

[1] Department of ECE, Karunya Institute of Technology and Sciences, Coimbatore, India

*Corresponding Author: D. Jude Hemanth, Email: judehemanth@karunya.edu

Computers and Electrical Engineering 68 (2018) 170–180

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Brain signal based human emotion analysis by circular back propagation and Deep Kohonen Neural Networks☆

D. Jude Hemanth[a], J. Anitha[a], Le Hoang Son[b,c,*]

[a] Department of ECE, Karunya University, Coimbatore, India
[b] Division of Data Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam
[c] Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam

Reinforcement

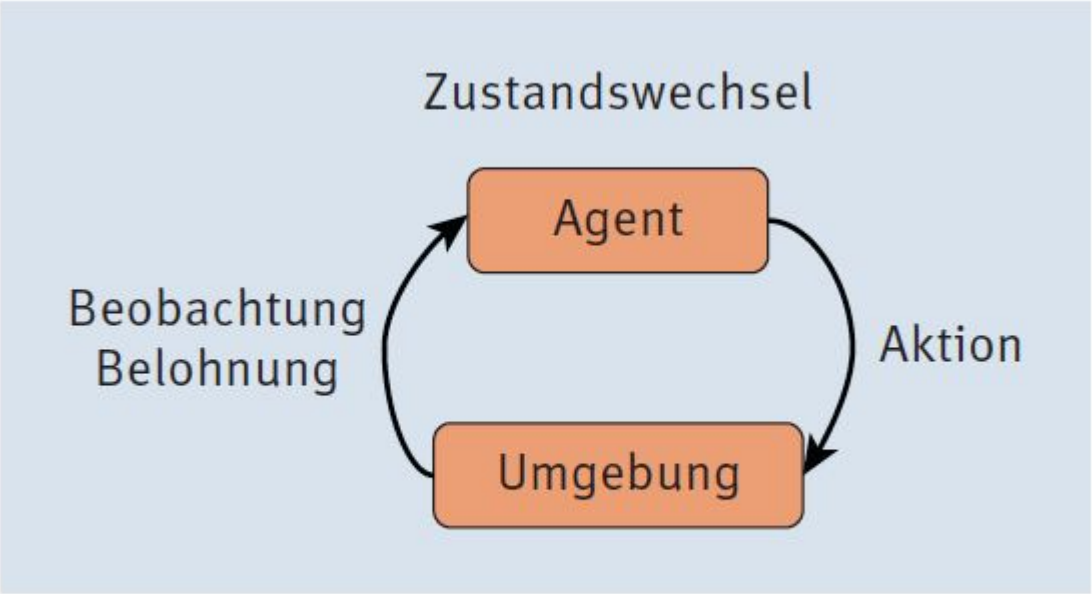


Abbildung 12.14 Reinforcement-System

Abschlagsfaktor γ (»Gamma«)

oder auch *Diskontierungsfaktor* genannt.

$$G_t = \sum_{k=1}^{T} \gamma^k \cdot r_{t+k} \text{ mit } 0 \leq \gamma < 1$$

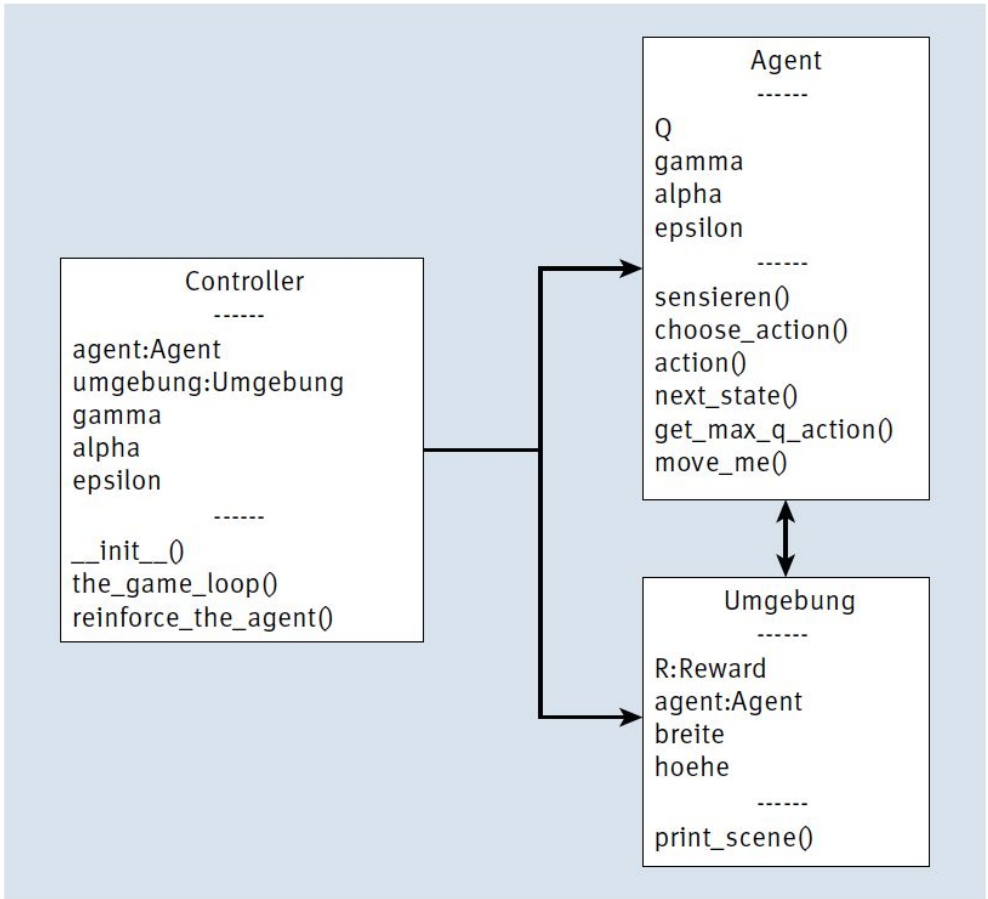


Abbildung 12.15 Klassendiagramm zum Q-Learning

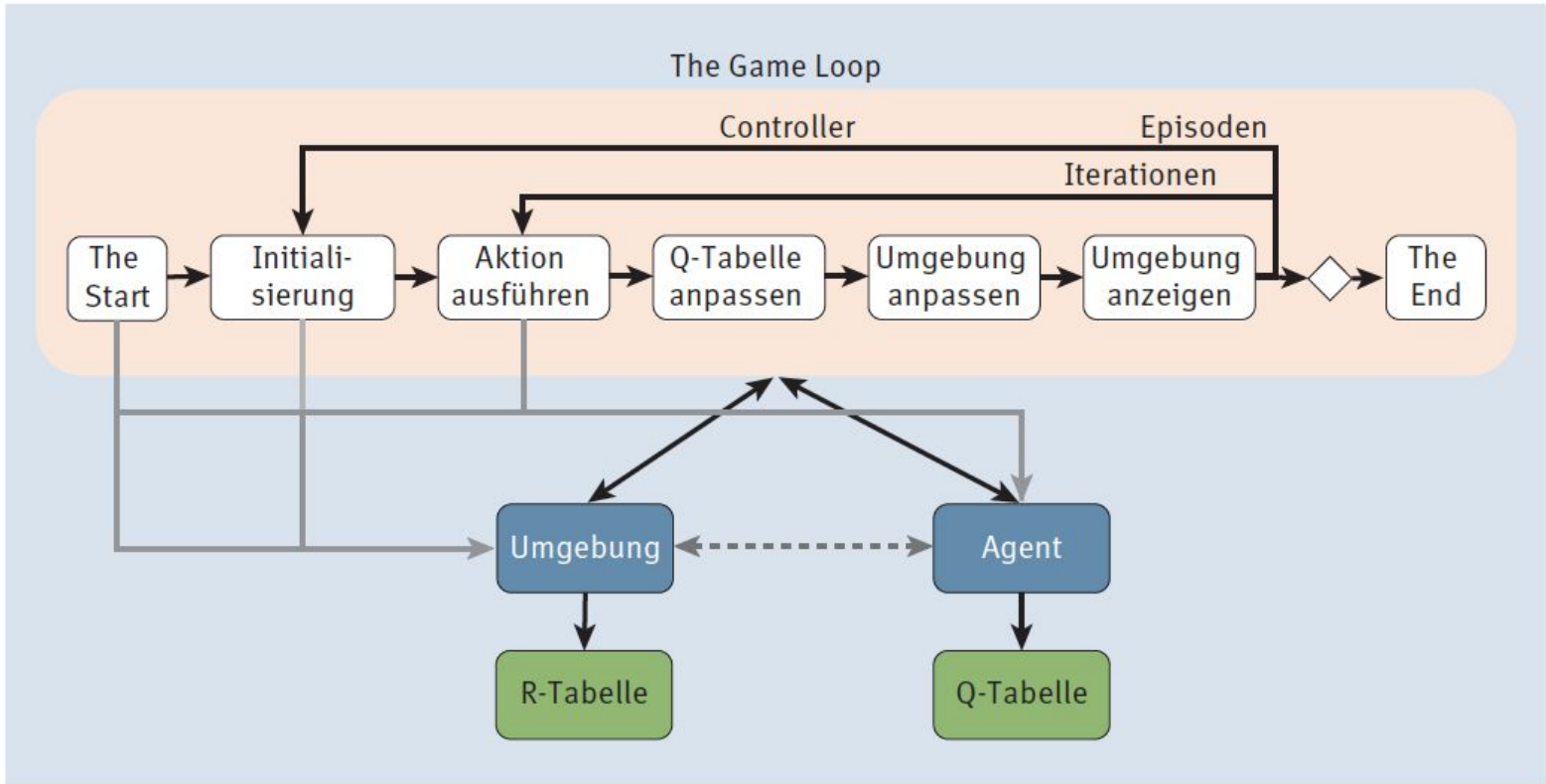


Abbildung 12.16 Game-Loop für das Q-Learning

Algo

Der Game-Loop sieht übersetzt in einen Algorithmus dann folgendermaßen aus:

Q-Learning-Algorithmus

1. Initialisiere den *Gamma-Wert* γ, der den Abschlag für zukünftige Belohnungen festlegt, die *Lernrate* und die *Belohnungstabelle* R, die für Zustände und Aktionen die resultierende Belohnung enthält.
2. Initialisiere die Q-Tabelle mit beliebigen Werten, zum Beispiel 0.
3. Für jede Episode:
   - Wähle einen zufälligen Initialzustand.
   - Wiederhole, bis der Zielzustand erreicht ist:
     - Wähle eine Aktion aus der Menge der möglichen Aktionen für den derzeitigen Zustand aus.
     - Berechne den neuen Q-Wert.
     - Ermittle die Belohnung zum aktuellen Zustand und zur Aktion.
     - Ermittle den maximalen Q-Wert für alle möglichen Aktionen im nächsten Zustand.
     - Setze den nächsten Zustand als den aktuellen Zustand.

$$Q^{\text{neu}}(s_t, a_t) \leftarrow Q^{\text{alt}}(s_t, a_t) + \underbrace{\alpha}_{\text{LR}} \cdot \left(r_t + \underbrace{\gamma}_{\text{DF}} \cdot \underbrace{\max_{a \in A} Q(s_{t+1}, a)}_{\text{ZW}} - Q^{\text{alt}}(s_t, a_t) \right)$$

Eine alternative Formulierung der Formel ist die folgende:

$$Q^{\text{neu}}(s_t, a_t) \leftarrow (1 - \alpha) \cdot Q^{\text{alt}}(s_t, a_t) + \alpha \cdot \left(r_t + \underbrace{\gamma}_{\text{DF}} \cdot \underbrace{\max_{a \in A} Q(s_{t+1}, a)}_{\text{ZW}} \right)$$

Auswertung

Sind die Epochen absolviert, kann der Agent mittels der Q-Tabelle auf sein Wissen zugreifen und es für jede Startsituation nutzen. Dabei folgt der Agent seinem Gedächtnis von seinem aktuellen Zustand zum Zielzustand, und das könnte in algorithmischer Form folgendermaßen lauten:

1. Setze den aktuellen Zustand als Startzustand.
2. Wiederhole, bis der Zielzustand erreicht ist:
   - Suche in der Q-Tabelle in der Zeile mit dem aktuellen Zustand die Aktion mit dem maximalen Q-Wert.
   - Ermittle aus dem aktuellen Zustand und der Aktion den neuen Zustand und setze diesen als aktuellen Zustand.

Listing 12.3/4/5/6 Q-Learning

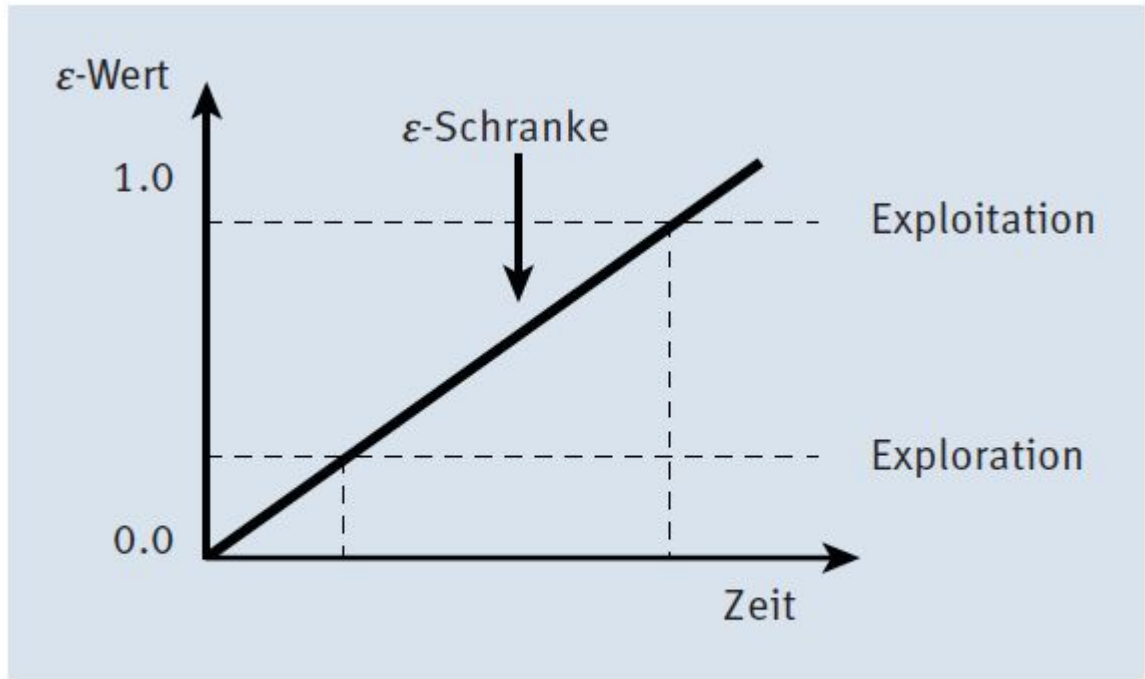


Abbildung 12.17 Exploration versus Exploitation

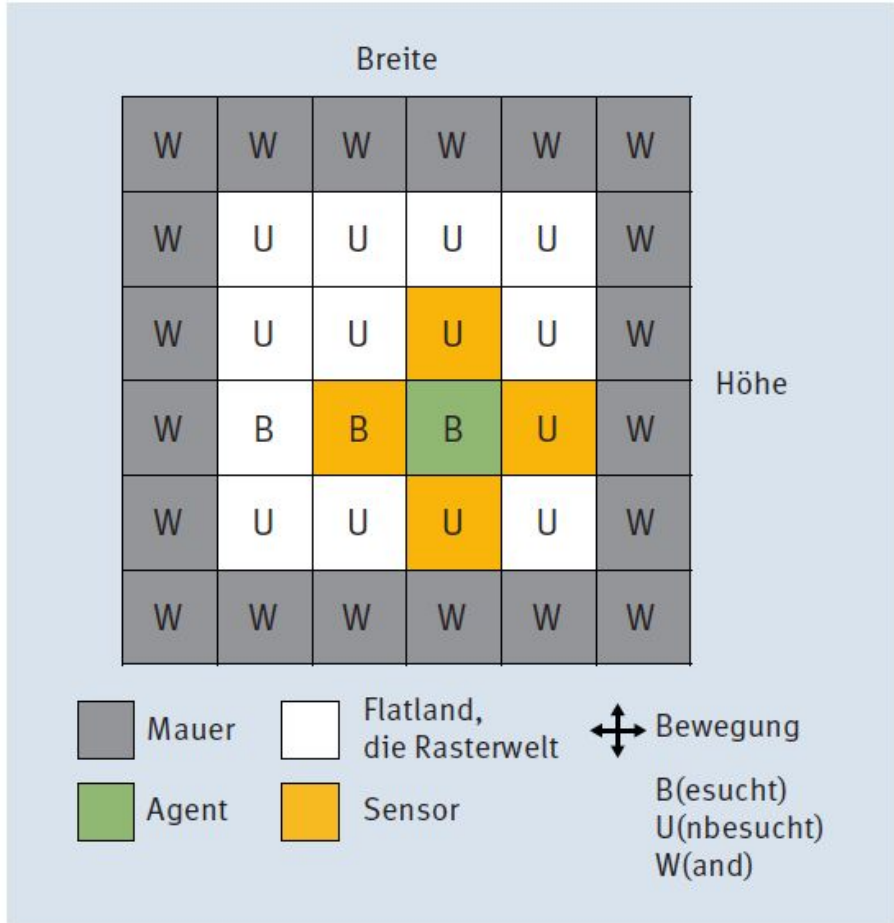


Abbildung 12.18 Flatland

R-Tabelle				
Zustände S	**Aktionen**			
(R,U,L,O)	rechts (0)	unten (1)	links (2)	oben (3)
(W,W,W,W)	–10.0	–10.0	–10.0	–10.0
...				
(U,U,W,W)	+5.0	+5.0	–10.0	–10.0
...				
(B,B,B,B)	–3.0	–3.0	–3.0	–3.0

Tabelle 12.3 Initialisierung der R-Tabelle; für Zustand (U,U,W,W) und Aktion »rechts« gibt es die Belohnung 5.0.

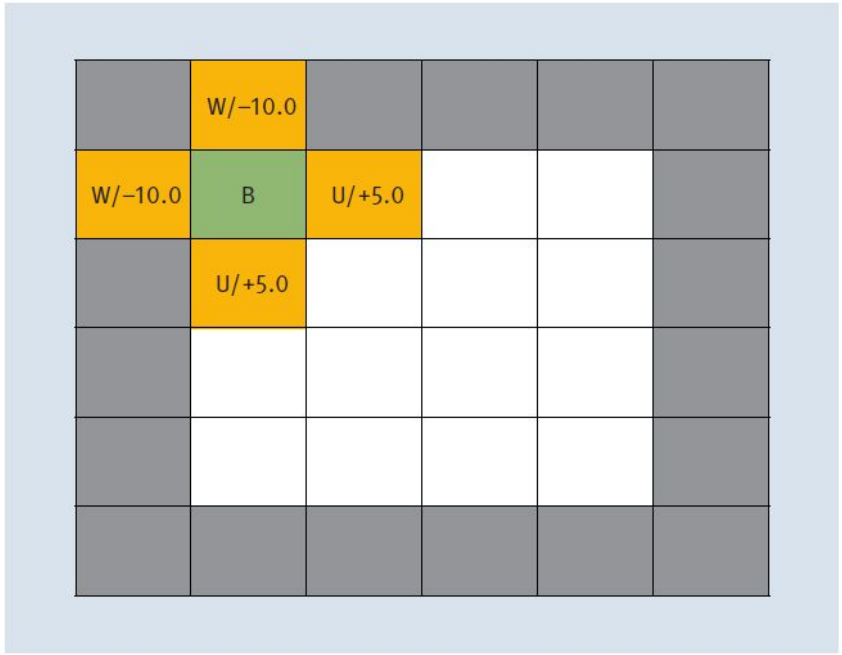


Abbildung 12.19 Initialisierung (Schritt t = 0), mit Zustand (U,U,W,W)

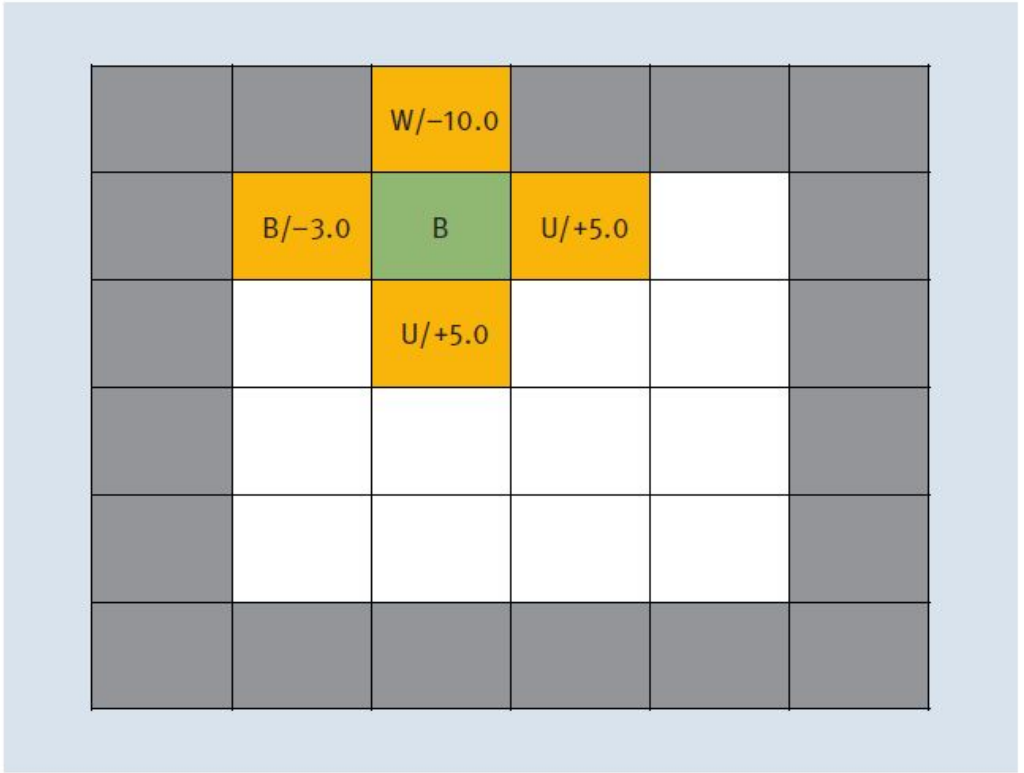


Abbildung 12.20 Schritt t = 1, mit Zustand (U,U,B,W)

Q-Tabelle				
Zustände	**Aktionen**			
(R,U,L,O)	rechts (0)	unten (1)	links (2)	oben (3)
…				
(U,U,W,W)	0.1	0.2	0.3	0.4
…				
(U,U,B,W)	0.1	0.2	0.3	0.4
…				

Tabelle 12.4 Initialisierung der Q-Tabelle

Q-Tabelle				
Zustände	**Aktionen**			
(R,U,L,O)	rechts (0)	unten (1)	links (2)	oben (3)
…				
(U,U,W,W)	5.32	0.2	0.3	0.4
…				
(U,U,B,W)	0.1	0.2	0.3	0.4
…				

Tabelle 12.5 Ermittlung der neuen Q-Werte für den ersten Schritt

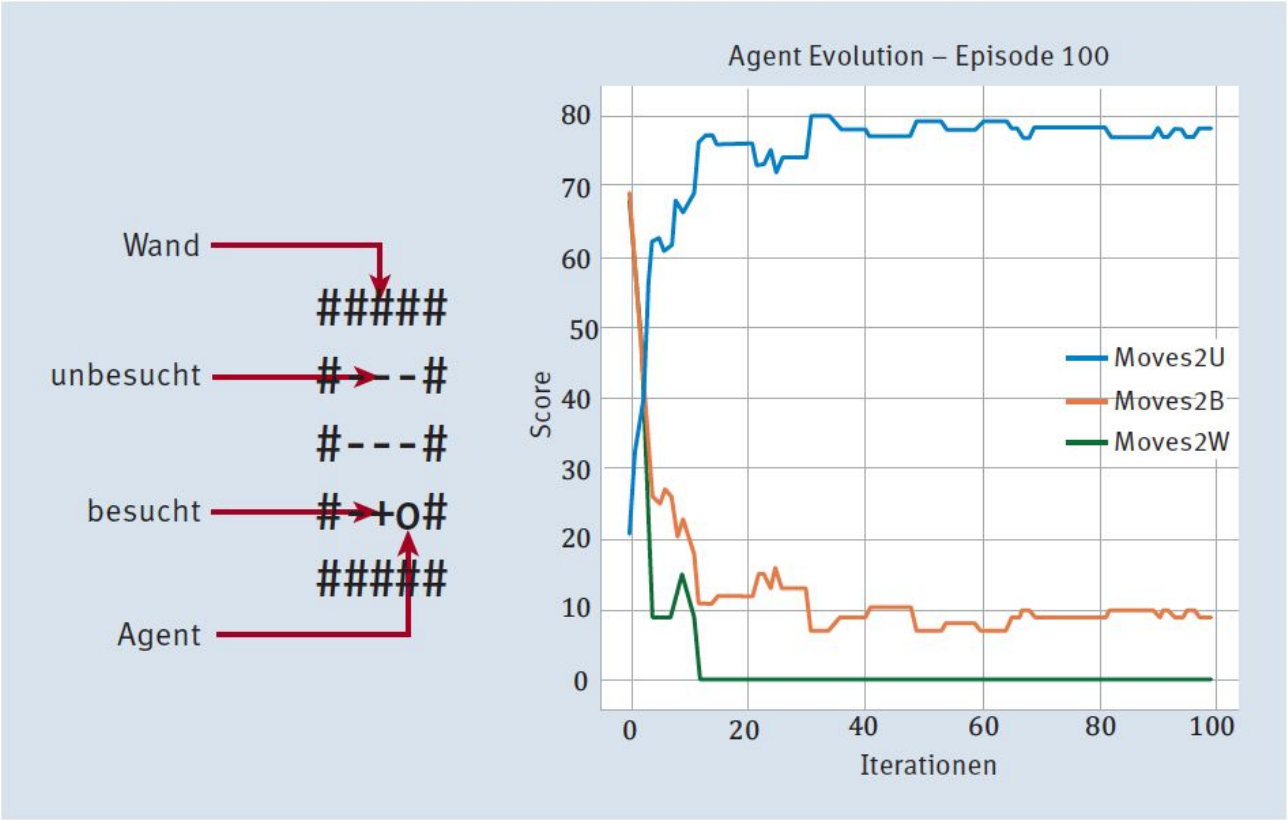


Abbildung 12.21 Der gelernte Agent in Episode 100 durchwandert die Umgebung.

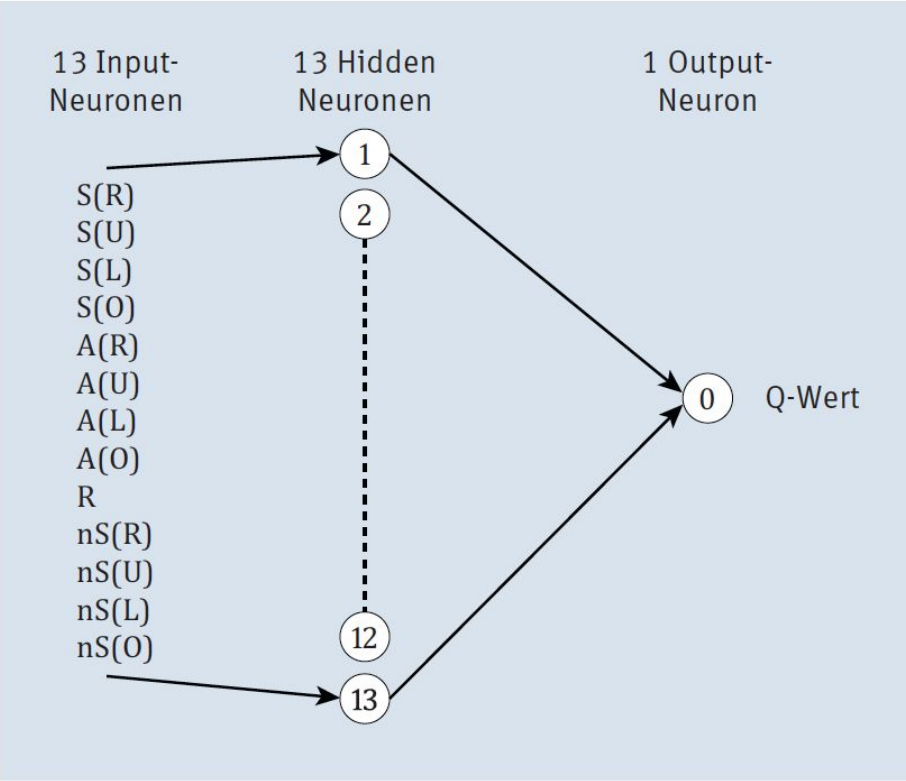


Abbildung 12.22 Q-NN-Architektur

- Die **Anzahl** haben wir in Klammern geschrieben.
- **One-Hot**(-Encoding) 4 bedeutet, dass vier Inputs verwendet werden und jeder den Wert 0 oder 1 annehmen kann, wobei immer nur genau einer der Inputs den Wert 1 annehmen kann.
- **Max 4** bedeutet, dass der maximale Wert gewinnt.
- Nun zu den Parametern:
- Input-Layer = aktueller State (4) + Aktion (One-Hot, 4) + Belohnung + nächster State (4)
- Hidden Layer = gleiche Dimension wie Input-Layer
- Output-Layer = entweder Q-Werte pro Aktion (maximal 4) oder einen Q-Wert
- Transferfunktion: ReLU
- Lernen: MSE
- Lernalgorithmus: Adam

wird diese Art von Speicher als *Ringspeicher* bezeichnet. Den Speicheraufbau, der gleichbedeutend mit dem Input für das KNN ist, haben wir folgendermaßen definiert:

1. Zustand/Sensor (4)
2. Aktion (4)
3. Belohnung
4. neuer Zustand (4)

Die zu optimierende Funktion, das heißt, die Fehlerfunktion, die durch Lernen zu minimieren ist, ist abgeleitet von der Q-Funktion:

$$MSE = \frac{1}{2} \cdot \sum \left(\underbrace{r_t + \gamma \cdot \max_{a \in A} Q(s_{t+1}, a)}_{\text{Zielwert}} - \underbrace{Q^{\text{alt}}(s_t, a_t)}_{\text{Vorhersage}} \right)^2$$

Semi-supervised

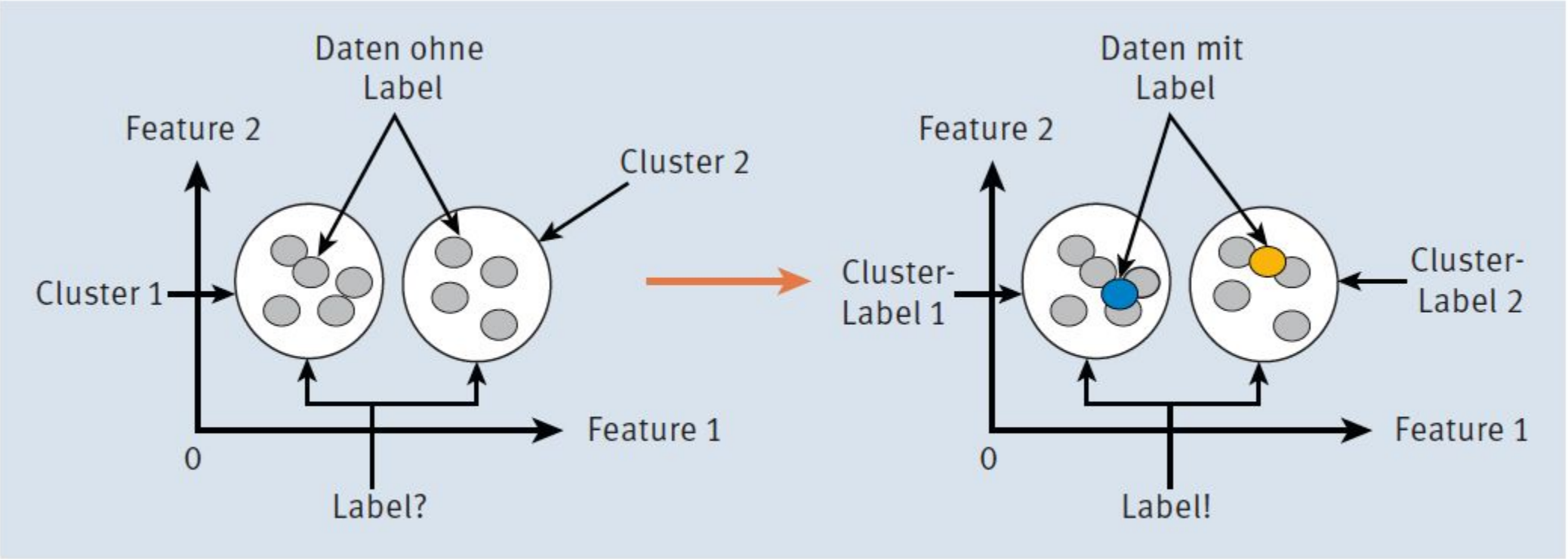


Abbildung 12.23 Semi-supervised Learning

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

		tatsächliche Klasse		
		A	**nicht A**	**Summe**
vorhergesagte Klasse	**A**	True Positive (TP)	False Positive (FP)	$TP + FP$
	nicht A	False Negative (FN)	True Negative (TN)	$FN + TN$
	Summe	$TP + FN$	$FP + TN$	

Tabelle 12.6 Konfusionsmatrix

		tatsächliche Klasse		
		A	**nicht A**	**Summe**
vorhergesagte Klasse	**A**	9 (TP)	3 (FP)	12 ($TP + FP$)
	nicht A	6 (FN)	9 (TN)	15 ($FN + TN$)
	Summe	15 ($TP + FN$)	12 ($FP + TN$)	27

Tabelle 12.7 Konfusionsmatrix für Stoppschild-Klassifikator

Die *Genauigkeit* als erstes einfaches Beispiel einer Kennzahl ergibt den Anteil der als korrekt klassifizierten Ereignisse (TP) zu der Gesamtheit der als Klasse »A« klassifizierten ($TP + FP$), d. h.

$$Genauigkeit = \frac{TP}{TP + FP}$$

Dadurch ergibt sich die $Genauigkeit = \frac{9}{12} = 0.75$.

Die *Sensitivität* (Empfindlichkeit) ist der Anteil der als korrekt klassifizierten Ereignisse (TP) zu der Gesamtheit aller als Klasse »A« zu klassifizierenden bzw. tatsächlich der Klasse »A« zugehörigen ($TP + FN$), d. h.

$$Sensitivität = \frac{TP}{TP + FN}$$

Dadurch ergibt sich die $Sensitivität = \frac{9}{15} = 0.6$.

Die *Spezifität* ist der Anteil der korrekt klassifizierten Nicht-Stoppschilder (TN) zu der Gesamtheit aller als Klasse »nicht A« klassifizierten ($FP + TN$), d. h.

$$Spezifität = \frac{TN}{FP + TN}$$

Dann ergibt sich dadurch die $Spezifität = \frac{9}{12} = 0.75$.

Schwellenwert	1-Spezifität	Sensitivität
0.00	0.00	0.00
0.10	0.00	0.13
0.20	0.01	0.37
...	...	...
0.50	0.25	0.60

Tabelle 12.8 Spezifität und Sensitivität für die Ermittlung der ROC-Kurve

Schwellenwert	1-Spezifität	Sensitivität
...		
1.00	1.00	1.00

Tabelle 12.8 Spezifität und Sensitivität für die Ermittlung der ROC-Kurve (Forts.)

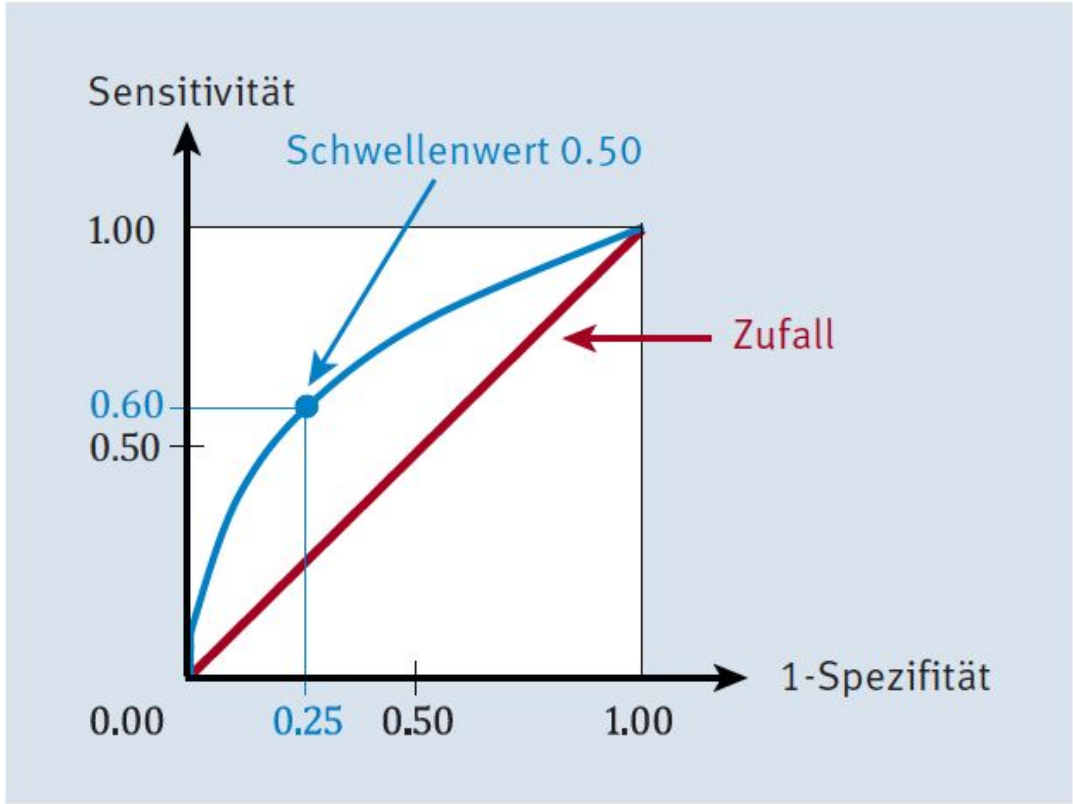


Abbildung 12.24 ROC-Kurve

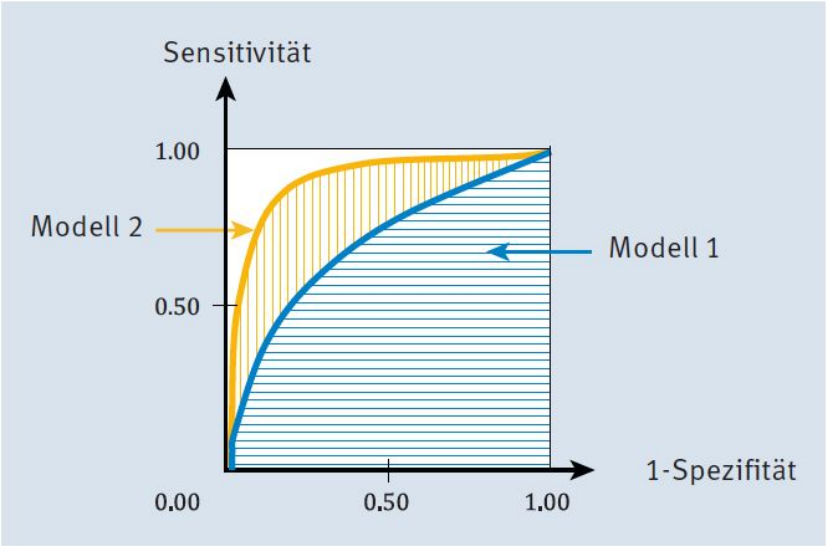


Abbildung 12.26 Zwei ROC-Kurven und ihre AUROC

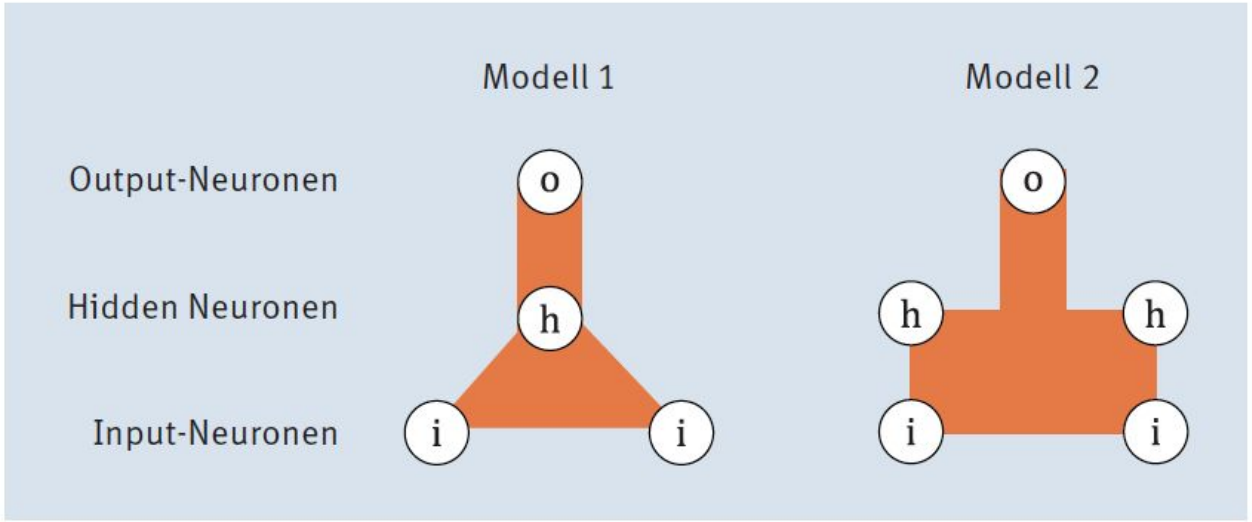


Abbildung 12.25 Modellvergleich

Anwendungsbereiche und Praxisbeispiele

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Listing 13.1/2/3/4/5 TensorFlow

Abbildung 13.2 Beispiel für Bildklassifikation

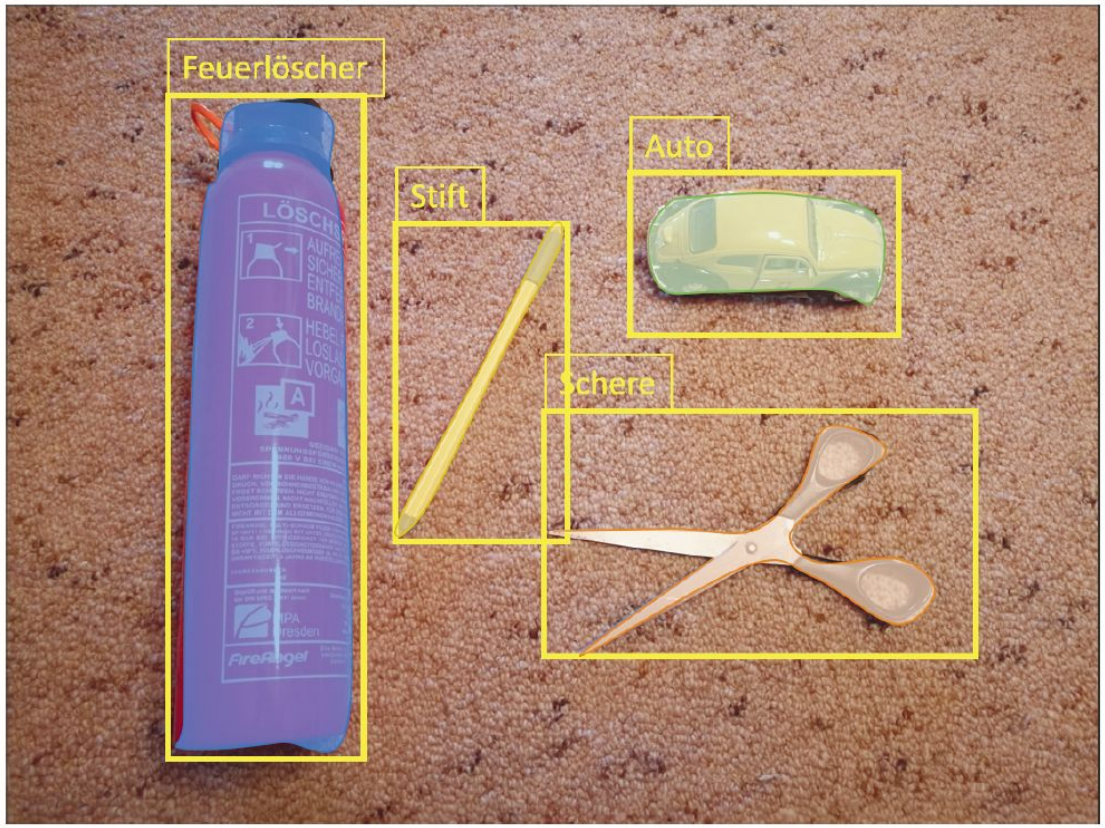


Abbildung 13.4 Beispiel für Objektsegmentierung

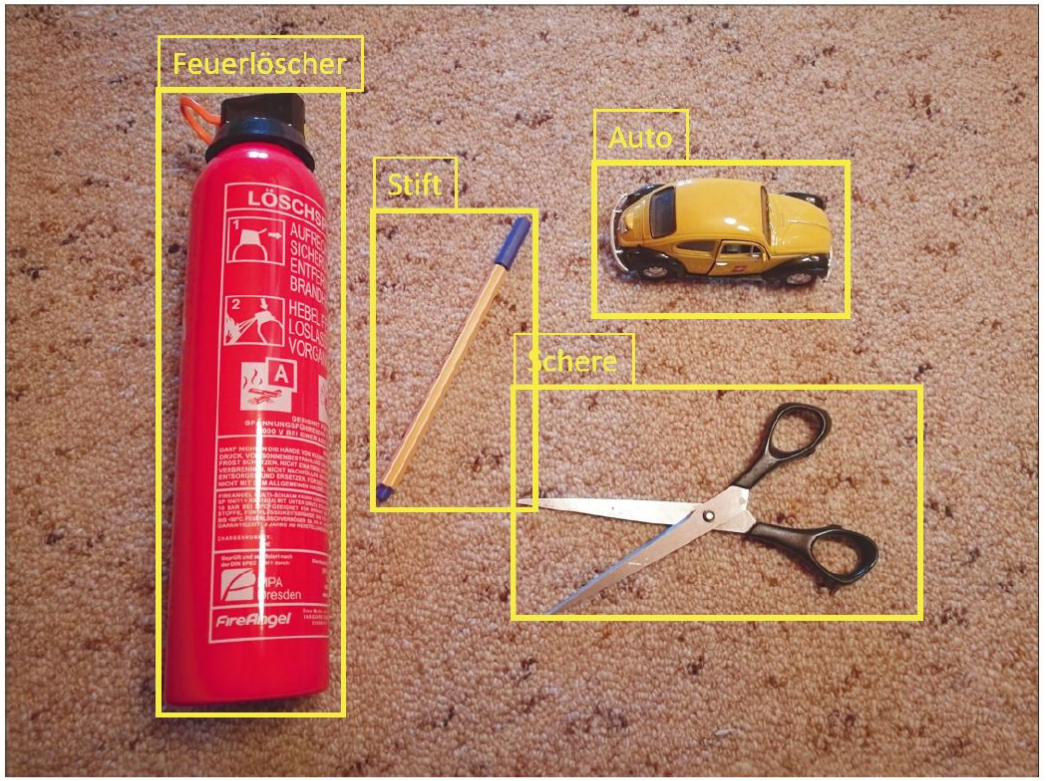


Abbildung 13.3 Beispiel für Objektidentifikation

Listing 13.7 Biene & Hummel

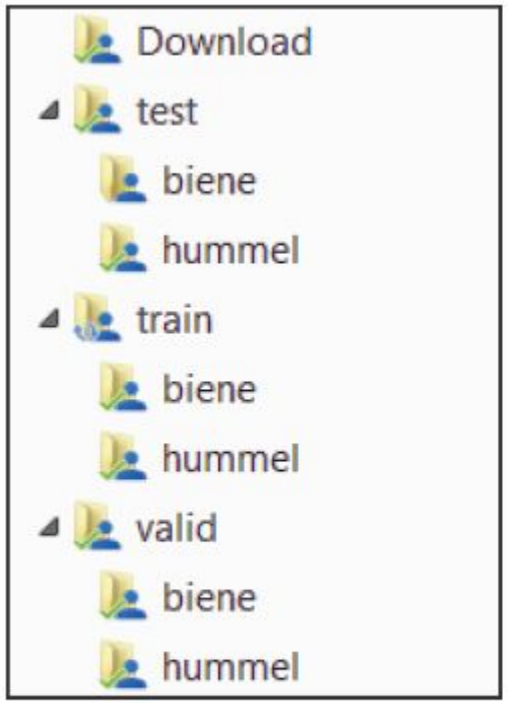


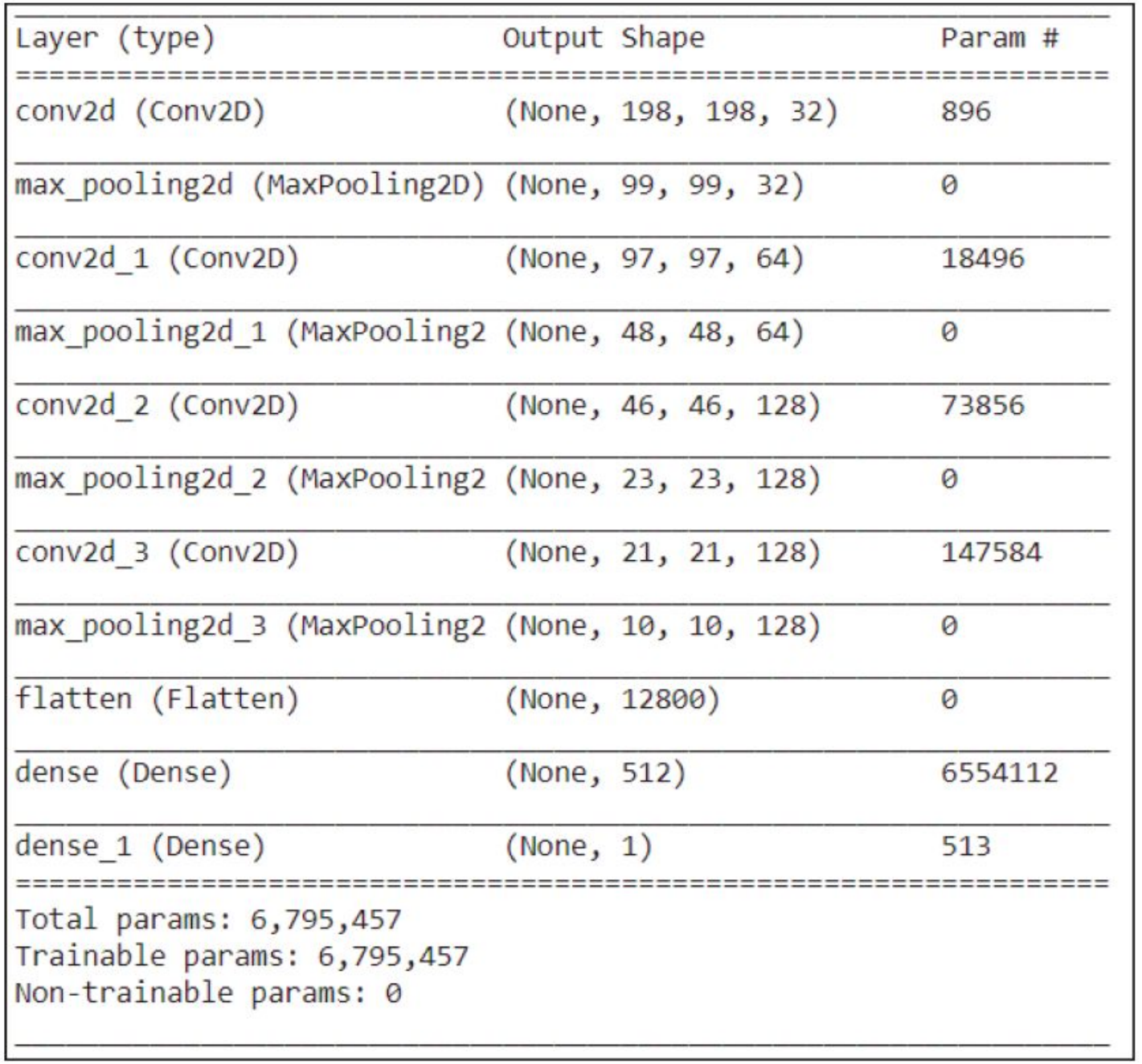
Abbildung 13.5 Verzeichnisstruktur für den Naive-Bees-Datensatz

Listing 13.8 Download der Datei » train_labels.csv«

Listing 13.9/10/11/12 Übertragen der Datei in einen DataFrame

Abbildung 13.6 Bild Nummer 8 (Biene oder Hummel? Nicht nur wegen der Unschärfe schwer zu entscheiden)

Listing 13.13/14 Verteilung der Bilddaten auf die Netzwerkstruktur



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 198, 198, 32)	896
max_pooling2d (MaxPooling2D)	(None, 99, 99, 32)	0
conv2d_1 (Conv2D)	(None, 97, 97, 64)	18496
max_pooling2d_1 (MaxPooling2	(None, 48, 48, 64)	0
conv2d_2 (Conv2D)	(None, 46, 46, 128)	73856
max_pooling2d_2 (MaxPooling2	(None, 23, 23, 128)	0
conv2d_3 (Conv2D)	(None, 21, 21, 128)	147584
max_pooling2d_3 (MaxPooling2	(None, 10, 10, 128)	0
flatten (Flatten)	(None, 12800)	0
dense (Dense)	(None, 512)	6554112
dense_1 (Dense)	(None, 1)	513

Total params: 6,795,457
Trainable params: 6,795,457
Non-trainable params: 0

Abbildung 13.7 Schichten des Modells

Listing 13.15/16/17/18 Erzeugung der Trainings-, Test- und Validierungsdaten aus den Bilddaten

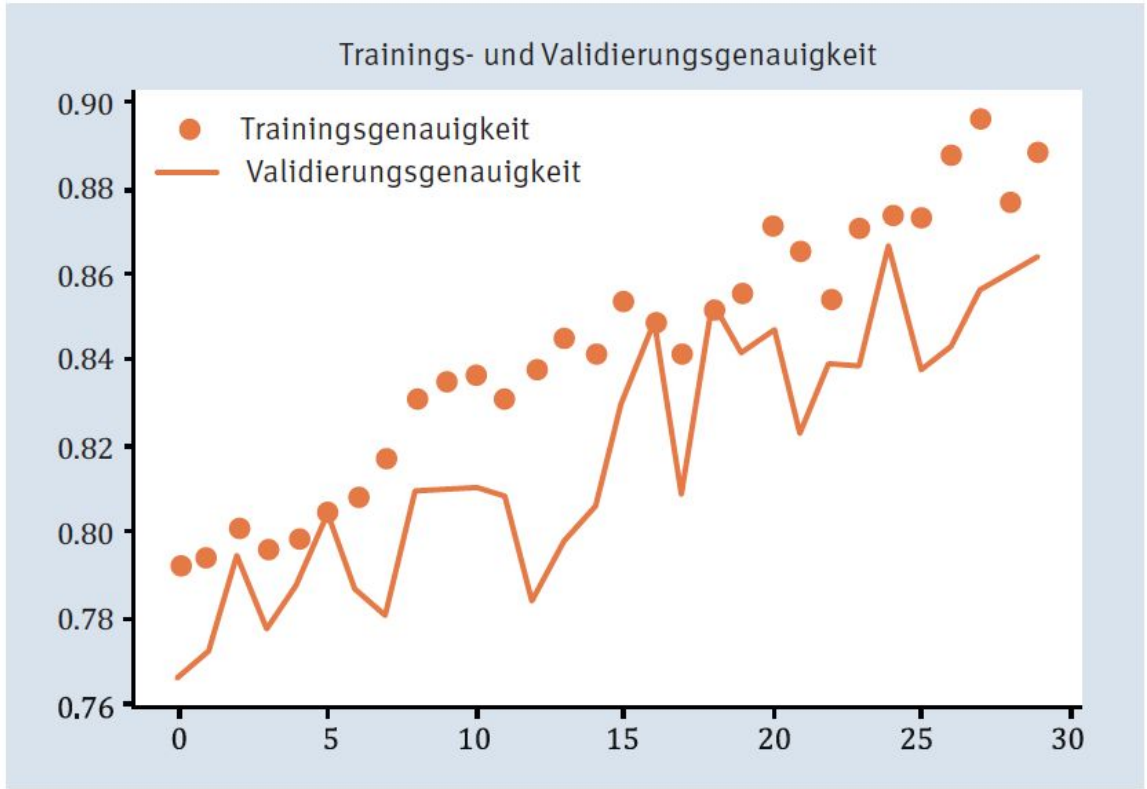


Abbildung 13.8 Verlaufskurve Trainings- und Validierungsgenauigkeit

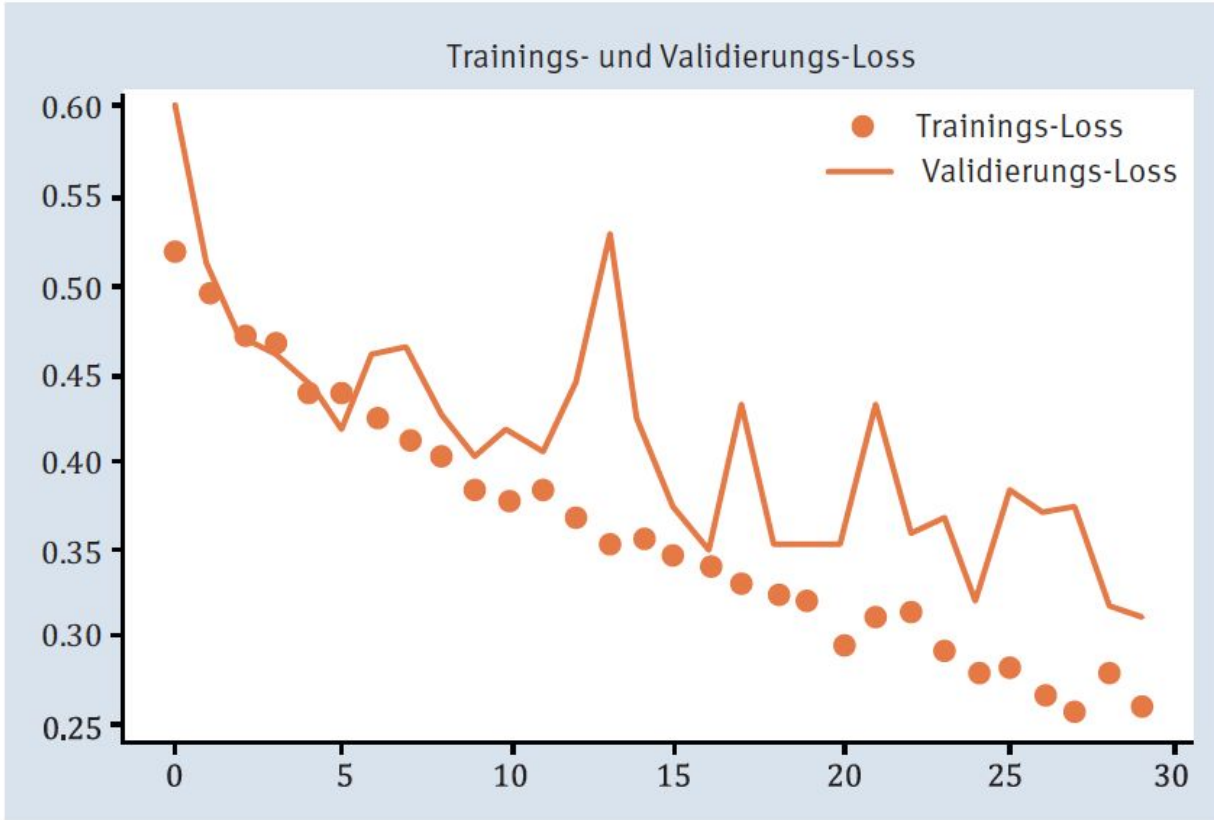


Abbildung 13.9 Verlaufskurve Trainings- und Validierungs-Loss

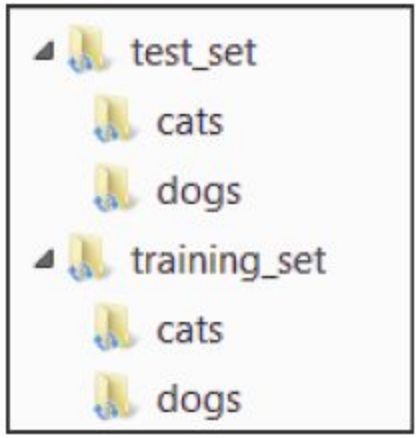


Abbildung 13.10 Verzeichnisstruktur des Katzen- und Hundedatensatzes

Listing 13.19/20/21 Import der Bibliotheken

Abbildung 13.11 Auszug aus dem Testdatensatz für Katzen

Abbildung 13.12 Auszug aus dem Testdatensatz für Hunde

Listing 13.22/23 Import und Aufbau des vortrainierten CNN

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, 4, 4, 2048)        21802784
_________________________________________________________________
flatten (Flatten)            (None, 32768)             0
_________________________________________________________________
dense (Dense)                (None, 256)               8388864
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 513
=================================================================
Total params: 30,323,745
Trainable params: 8,520,961
Non-trainable params: 21,802,784
_________________________________________________________________
```

Abbildung 13.13 Ausgabe von »model.summary()« unseres Netzes

Listing 13.24/25/26 Import und Aufbau des vortrainierten CNN

www.nor.gmbh office@nor.gmbh NÖR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

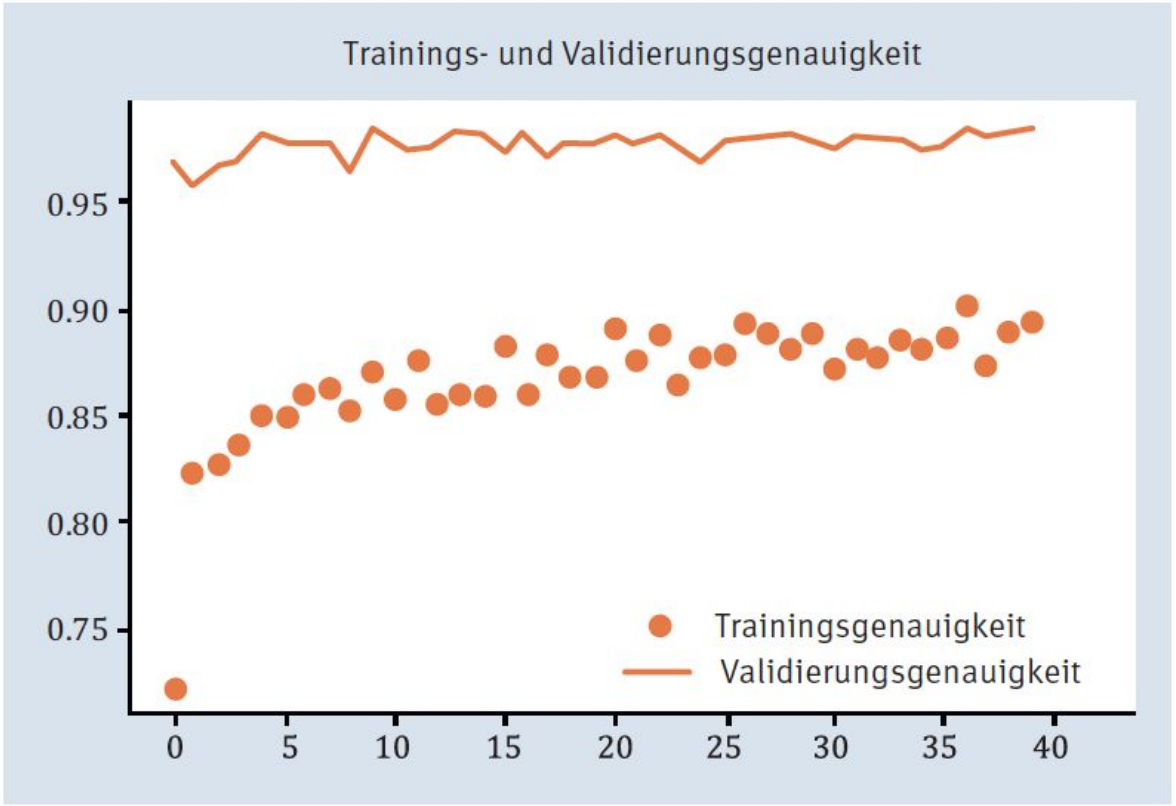


Abbildung 13.14 Genauigkeitsentwicklung über 40 Trainingsepochen

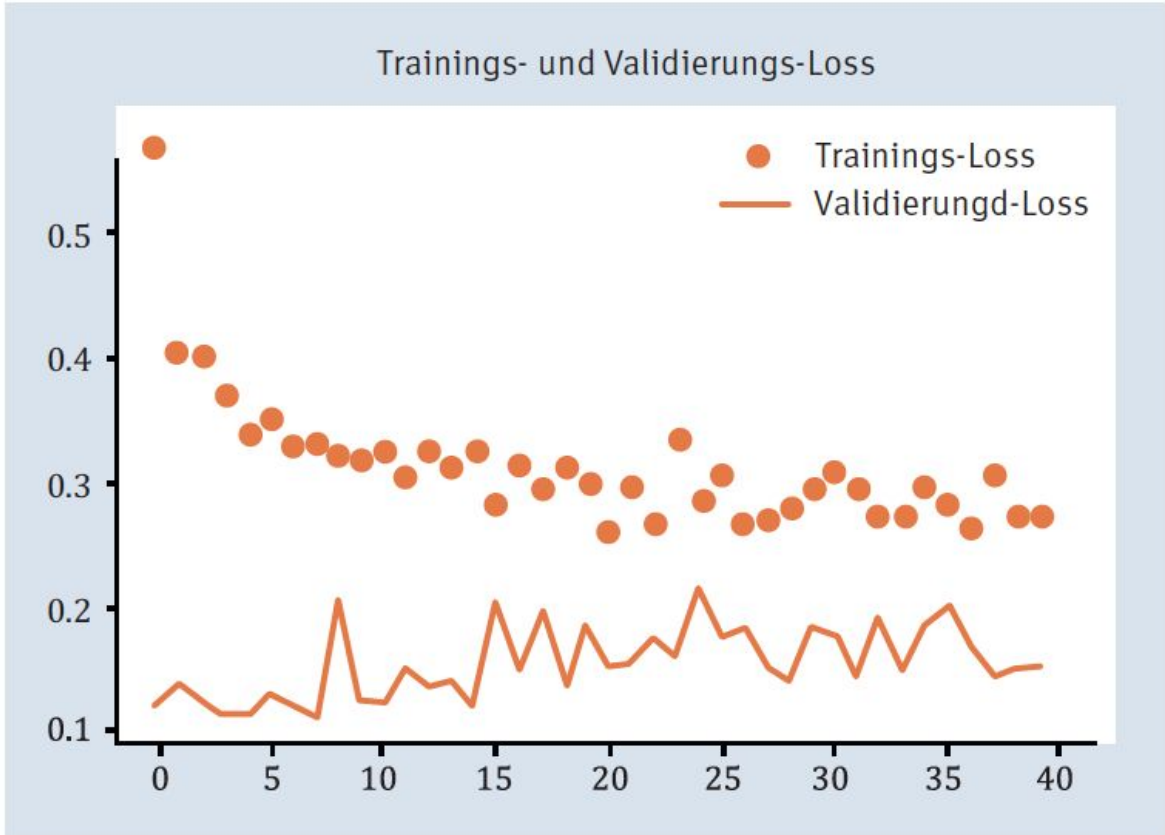


Abbildung 13.15 Verlustentwicklung über 40 Trainingsepochen

Abbildung 13.16 Bild einer Herbstlandschaft und die Traumvariante (Quelle: Wikimedia Commons)

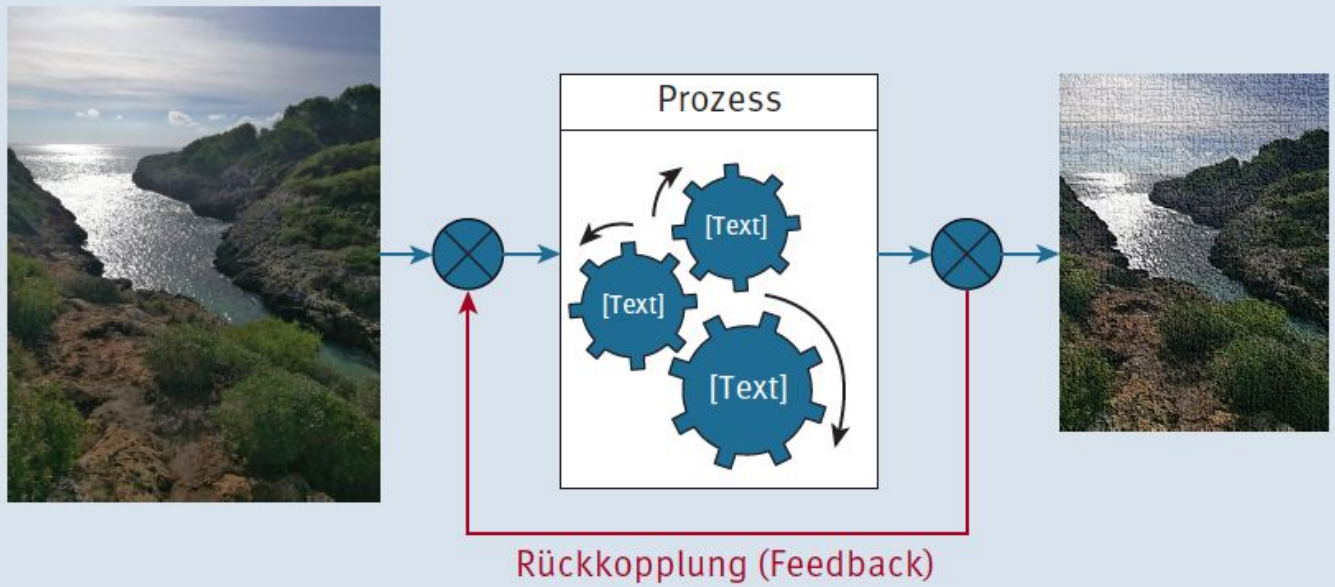


Abbildung 13.17 Diagramm einer Rückkopplungsschleife

Abbildung 13.18 Fünfstufige Bildpyramide

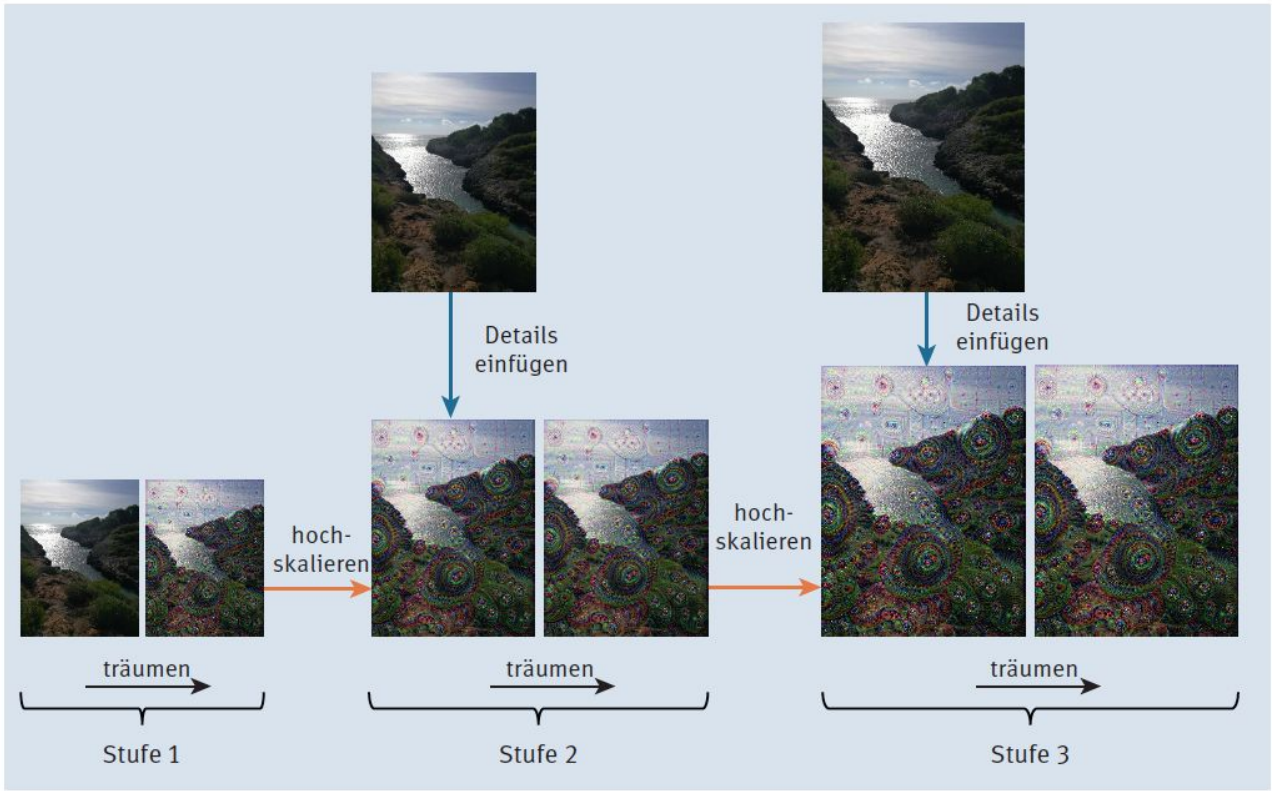
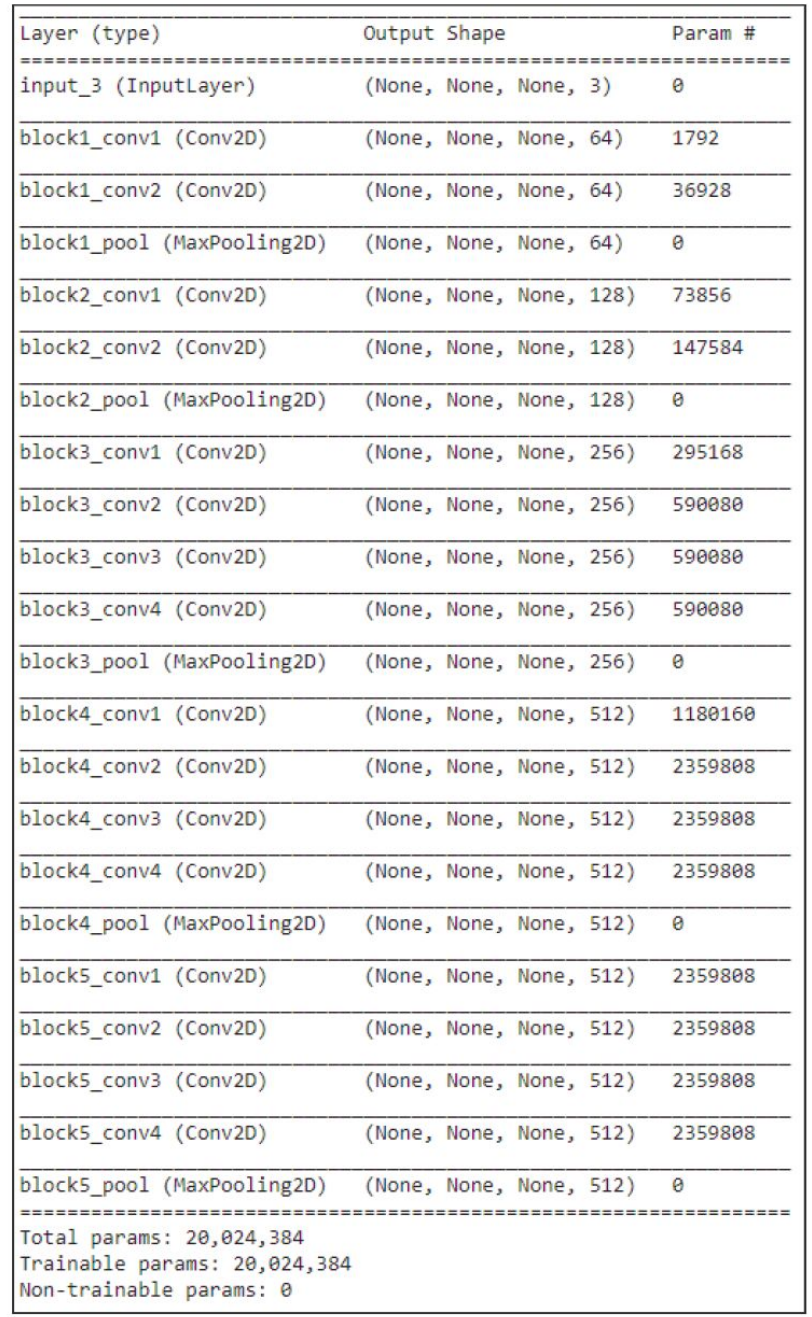


Abbildung 13.19 Bildhafte Darstellung des stufenweisen DeepDream-Algorithmus

Listing 13.28/29/30/31/32 Import der Bibliotheken und -Funktionen



Layer (type)	Output Shape	Param #
input_3 (InputLayer)	(None, None, None, 3)	0
block1_conv1 (Conv2D)	(None, None, None, 64)	1792
block1_conv2 (Conv2D)	(None, None, None, 64)	36928
block1_pool (MaxPooling2D)	(None, None, None, 64)	0
block2_conv1 (Conv2D)	(None, None, None, 128)	73856
block2_conv2 (Conv2D)	(None, None, None, 128)	147584
block2_pool (MaxPooling2D)	(None, None, None, 128)	0
block3_conv1 (Conv2D)	(None, None, None, 256)	295168
block3_conv2 (Conv2D)	(None, None, None, 256)	590080
block3_conv3 (Conv2D)	(None, None, None, 256)	590080
block3_conv4 (Conv2D)	(None, None, None, 256)	590080
block3_pool (MaxPooling2D)	(None, None, None, 256)	0
block4_conv1 (Conv2D)	(None, None, None, 512)	1180160
block4_conv2 (Conv2D)	(None, None, None, 512)	2359808
block4_conv3 (Conv2D)	(None, None, None, 512)	2359808
block4_conv4 (Conv2D)	(None, None, None, 512)	2359808
block4_pool (MaxPooling2D)	(None, None, None, 512)	0
block5_conv1 (Conv2D)	(None, None, None, 512)	2359808
block5_conv2 (Conv2D)	(None, None, None, 512)	2359808
block5_conv3 (Conv2D)	(None, None, None, 512)	2359808
block5_conv4 (Conv2D)	(None, None, None, 512)	2359808
block5_pool (MaxPooling2D)	(None, None, None, 512)	0

Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0

Abbildung 13.20 Ein Ausschnitt der Netzwerkstruktur von Inception_v3 (Ausgabe von »model.summary()«)

Listing 13.33/34/35/36/37 Loss- und Gradientenfunktion für den Optimierungsprozess

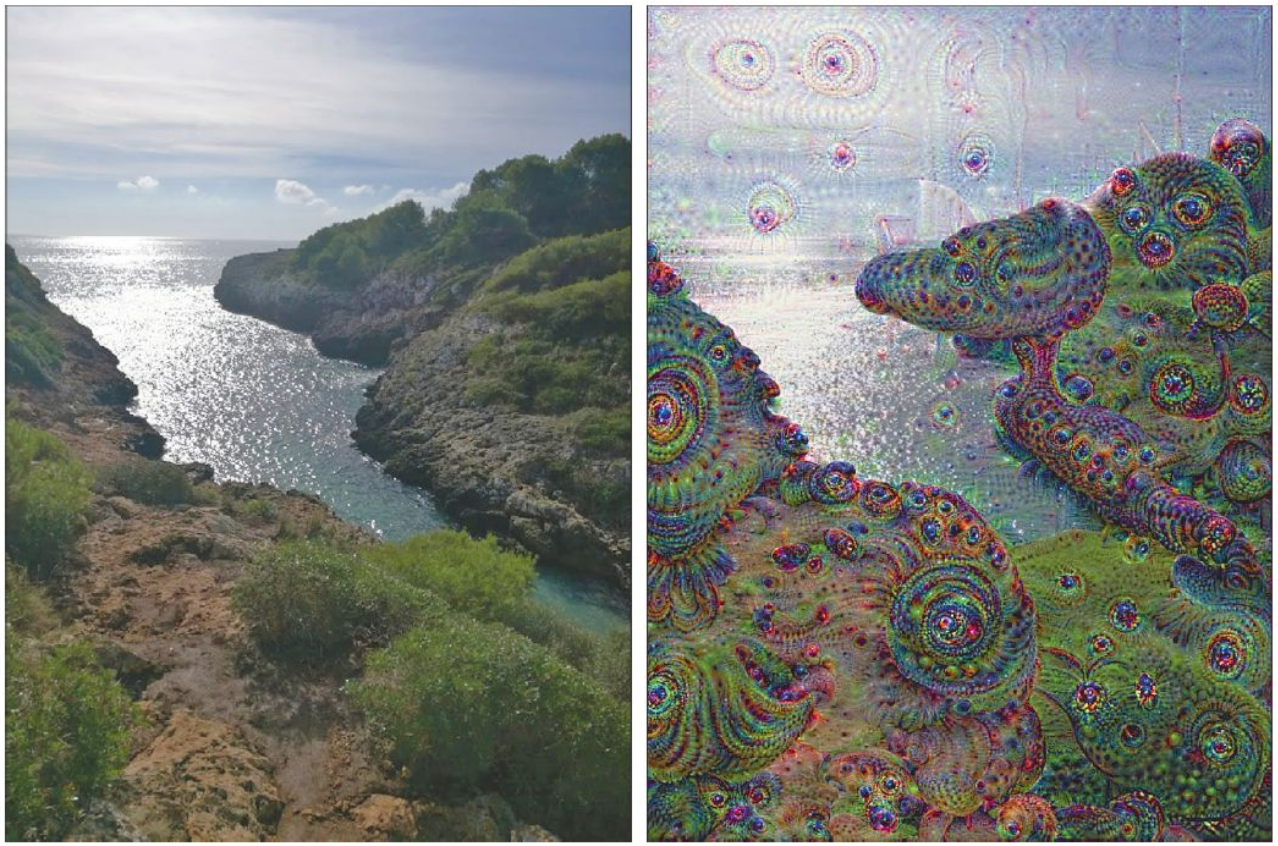


Abbildung 13.21 Original und Traumbild auf Basis von Inception_v3

TensorFlow 2 und Keras

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

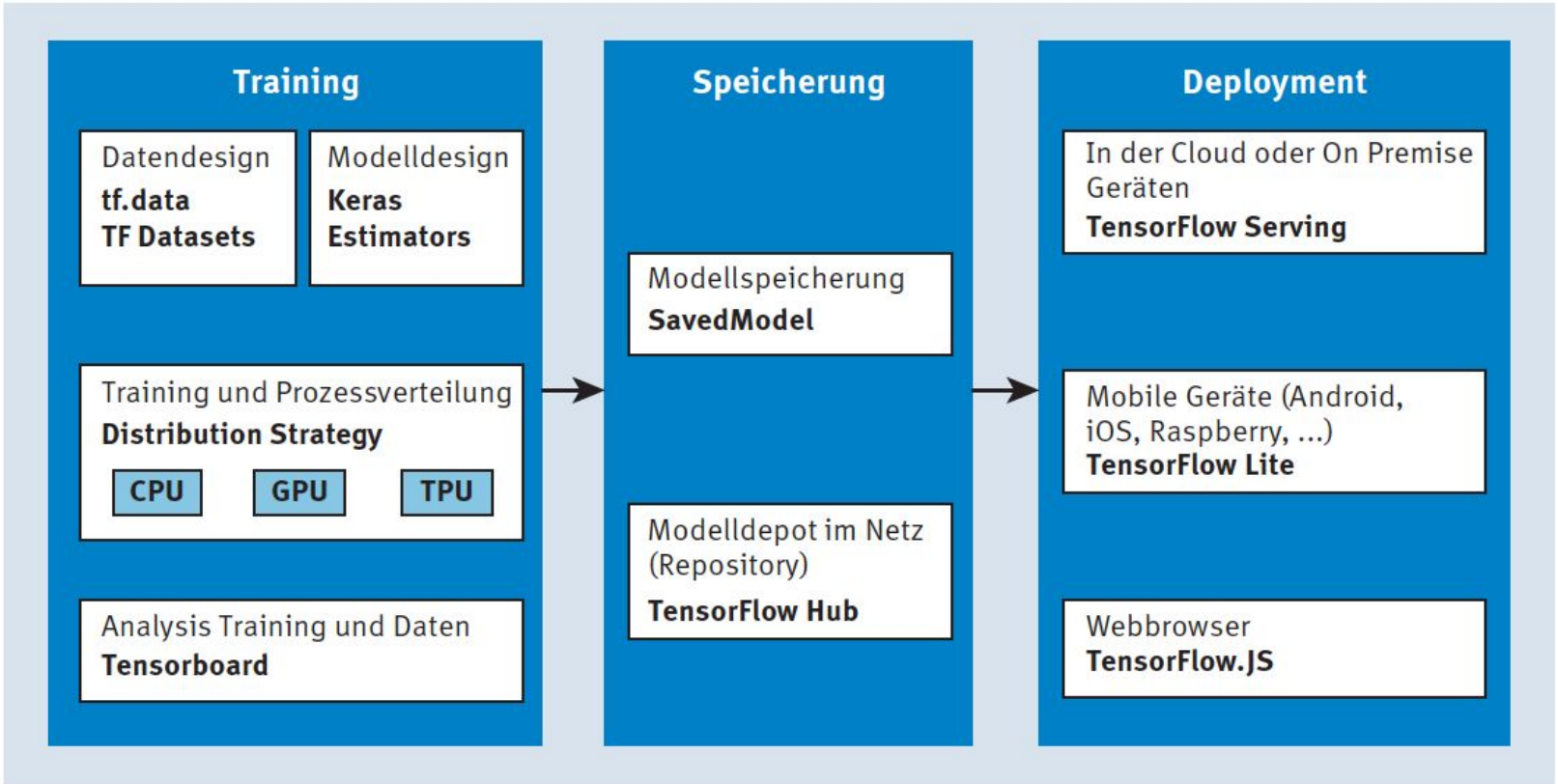


Abbildung C.1 Das TensorFlow-Ökosystem

Listing C.6 Einfaches neuronales Netz nach dem sequenziellen Modell (Keras)

Listing C.7 Neuronales Netz, mit funktionalem Paradigma erstellt (Keras)

Wie geht es weiter?

www.nor.gmbh office@nor.gmbh NoR GmbH - Hüttfeldstrasse 17 – A-6345 Kössen FN: 475603t UID: ATU72533127

Anhang