

NEURONALE NETZE

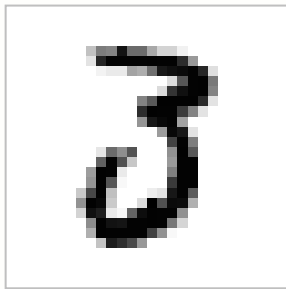
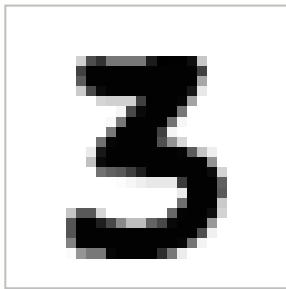
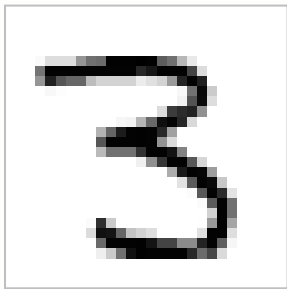
Handschriftliche Zahlen erkennen

Jasper Gude

3. Dezember 2023
Carl-Friedrich-Gauß-Gymnasium

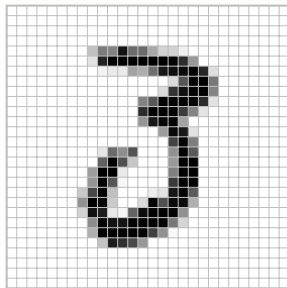
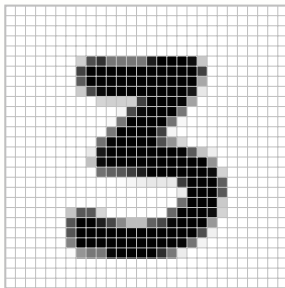
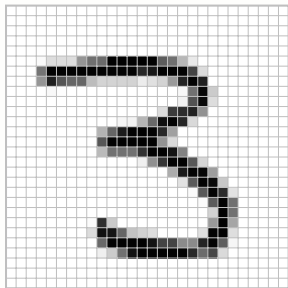
2.1

Modellierung des Problems



2.2

Modellierung des Problems



Modellierung



Künstliche Neuronen

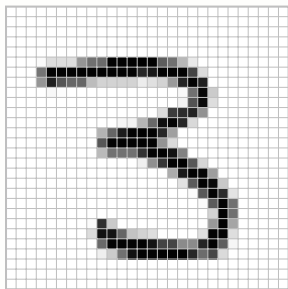


Training

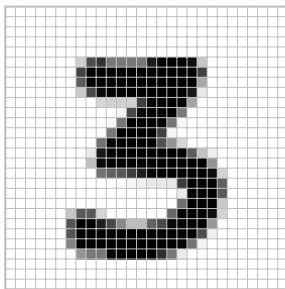


2.3

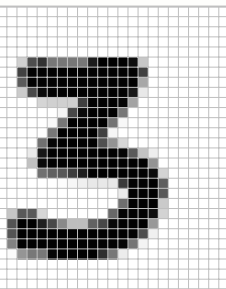
Modellierung des Problems



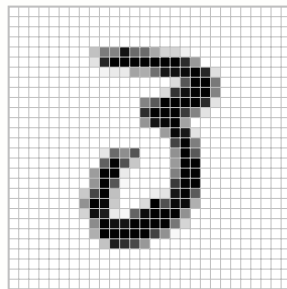
0,00



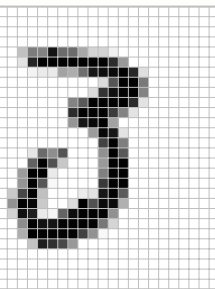
0,25



0,50



0,75



1,00



Modellierung



Künstliche Neuronen

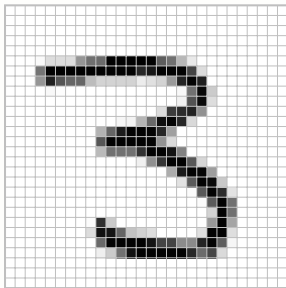


Training



3.1

Überführung auf eine Netzstruktur



28px × 28px



Modellierung



Künstliche Neuronen

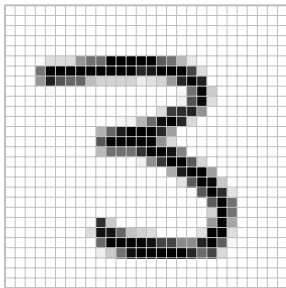


Training



3.2

Überführung auf eine Netzstruktur

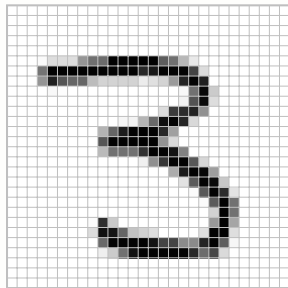


28px x 28px



3.3

Überführung auf eine Netzstruktur



28px x 28px

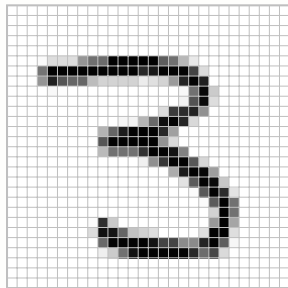


- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9

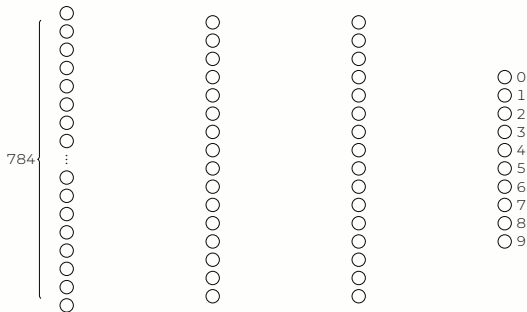


3.4

Überführung auf eine Netzstruktur



28px x 28px



Modellierung



Künstliche Neuronen

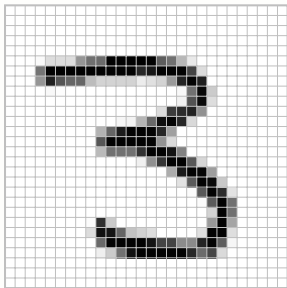


Training

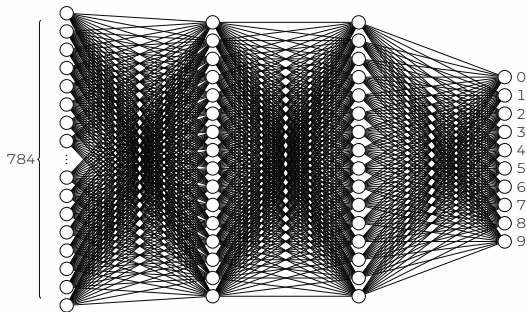


3.5

Überführung auf eine Netzstruktur



28px x 28px



Modellierung



Künstliche Neuronen

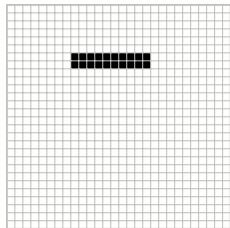


Training

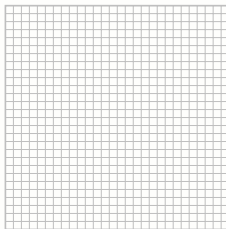


4.1

Gewichtungen setzen



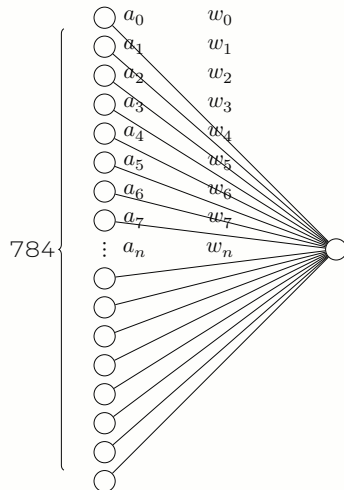
Inputs



Gewichte

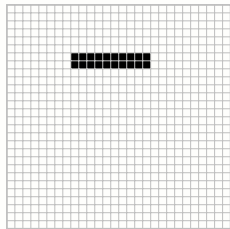
Linearkombination

$$w_0 a_0 + w_1 a_1 + \dots + w_n a_n$$

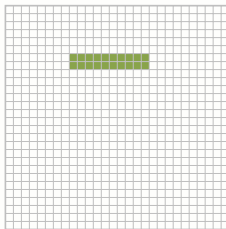


4.2

Gewichtungen setzen



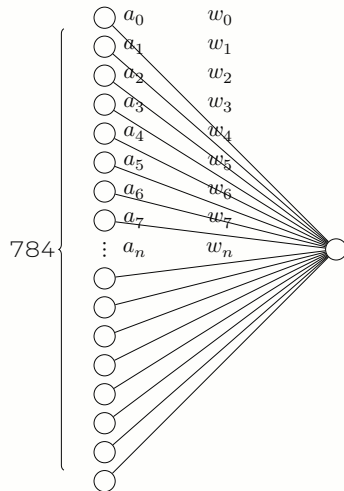
Inputs



Gewichte

Linearkombination

$$w_0 a_0 + w_1 a_1 + \dots + w_n a_n$$



Modellierung



Künstliche Neuronen

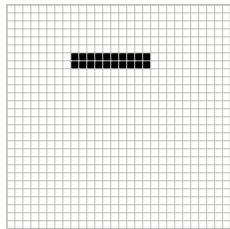


Training

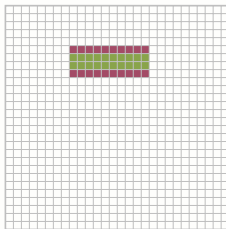


4.3

Gewichtungen setzen



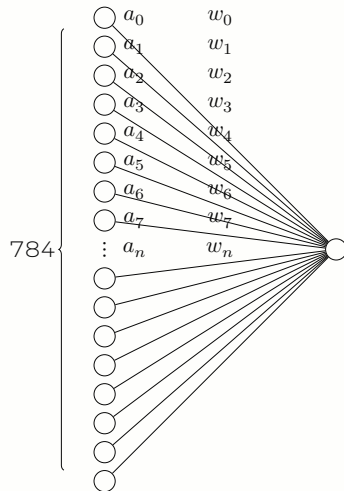
Inputs



Gewichte

Linearkombination

$$w_0 a_0 + w_1 a_1 + \dots + w_n a_n$$



Modellierung



Künstliche Neuronen

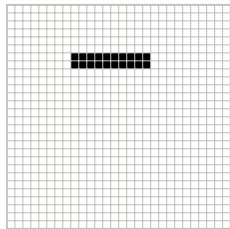


Training

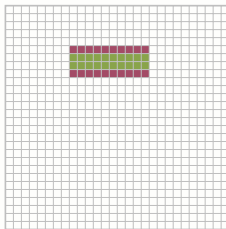


4.4

Gewichtungen setzen



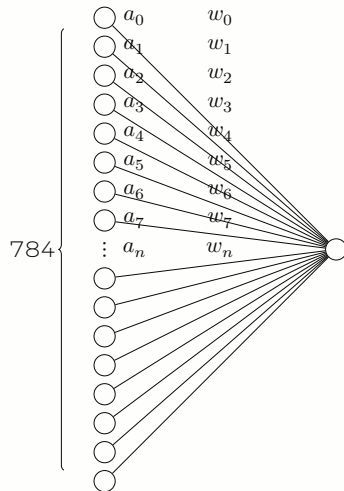
Inputs



Gewichte

Linearkombination

$$w_0 a_0 + w_1 a_1 + \dots + w_n a_n - b$$



Modellierung



Künstliche Neuronen



Training

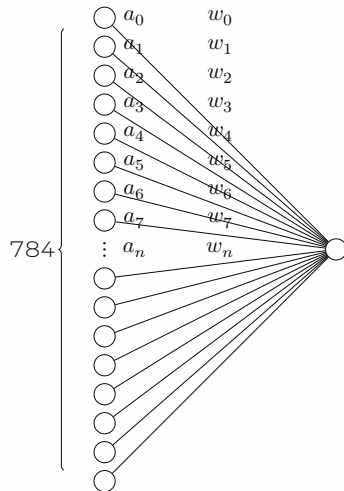
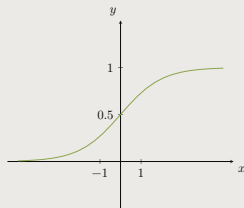


5

Zahlenbereich begrenzen

Sigmoidfunktion

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



Modellierung



Künstliche Neuronen



Training

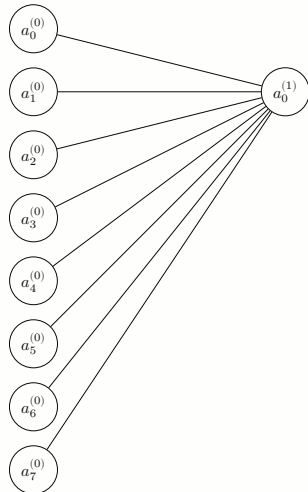


6.1

Alles zusammen setzen

Aktivierungsfunktion

$$a_0^{(1)} = \sigma(w_{0,0}a_0^{(0)} + w_{0,1}a_1^{(0)} + \dots + w_{0,n}a_n^{(0)} - b_0)$$



Modellierung

**Künstliche Neuronen**

Training



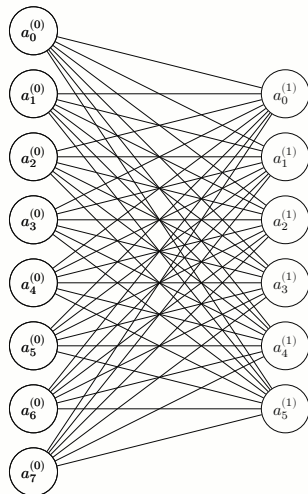
6.2

Alles zusammen setzen

Aktivierungsfunktion

$$a_0^{(1)} = \sigma(w_{0,0}a_0^{(0)} + w_{0,1}a_1^{(0)} + \dots + w_{0,n}a_n^{(0)} - b_0)$$

$$\begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ \vdots \\ a_k^{(1)} \end{bmatrix} = \sigma \left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \end{bmatrix} \right)$$



Modellierung



Künstliche Neuronen



Training



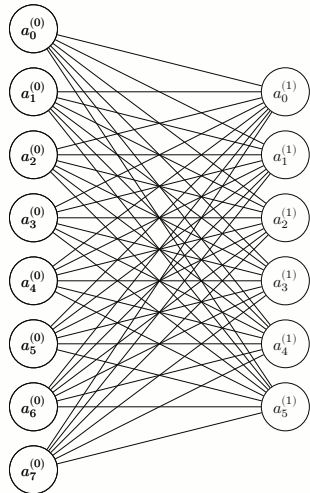
6.3

Alles zusammen setzen

Aktivierungsfunktion

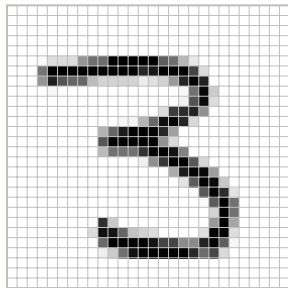
$$a^{(1)} = \sigma(Wa^{(0)} + \vec{b})$$

$$\begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ \vdots \\ a_k^{(1)} \end{bmatrix} = \sigma \left(\begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{k,0} & w_{k,1} & \cdots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_n^{(0)} \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_k \end{bmatrix} \right)$$

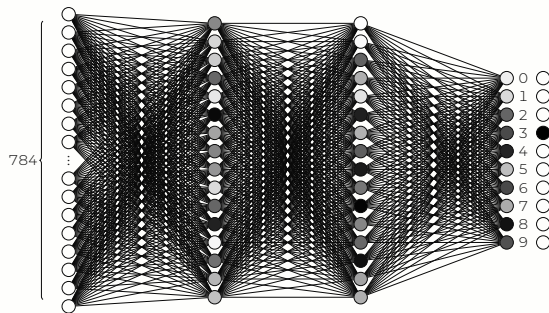


7.1

Fehler bestimmen



28px x 28px



Modellierung



Künstliche Neuronen

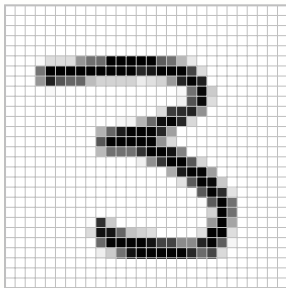


Training



7.2

Fehler bestimmen



28px x 28px

$C_3(\vec{W}) = \text{Summe}$

$\left(\begin{smallmatrix} 0.17 \end{smallmatrix} \right)_0$	—	$\left(\begin{smallmatrix} 0.00 \end{smallmatrix} \right)^2$
$\left(\begin{smallmatrix} 0.48 \end{smallmatrix} \right)_1$	—	$\left(\begin{smallmatrix} 0.00 \end{smallmatrix} \right)^2$
$\left(\begin{smallmatrix} 0.38 \end{smallmatrix} \right)_2$	—	$\left(\begin{smallmatrix} 0.00 \end{smallmatrix} \right)^2$
$\left(\begin{smallmatrix} 0.20 \end{smallmatrix} \right)_3$	—	$\left(\begin{smallmatrix} 1.00 \end{smallmatrix} \right)^2$
$\left(\begin{smallmatrix} 0.76 \end{smallmatrix} \right)_4$	—	$\left(\begin{smallmatrix} 0.00 \end{smallmatrix} \right)^2$
$\left(\begin{smallmatrix} 0.65 \end{smallmatrix} \right)_5$	—	$\left(\begin{smallmatrix} 0.00 \end{smallmatrix} \right)^2$
$\left(\begin{smallmatrix} 0.21 \end{smallmatrix} \right)_6$	—	$\left(\begin{smallmatrix} 0.00 \end{smallmatrix} \right)^2$
$\left(\begin{smallmatrix} 0.72 \end{smallmatrix} \right)_7$	—	$\left(\begin{smallmatrix} 0.00 \end{smallmatrix} \right)^2$
$\left(\begin{smallmatrix} 0.68 \end{smallmatrix} \right)_8$	—	$\left(\begin{smallmatrix} 0.00 \end{smallmatrix} \right)^2$
$\left(\begin{smallmatrix} 0.81 \end{smallmatrix} \right)_9$	—	$\left(\begin{smallmatrix} 0.00 \end{smallmatrix} \right)^2$



Modellierung



Künstliche Neuronen



Training



7.3

Fehler bestimmen

```

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
    
```

Dataset D

Fehlerfunktion: Mean Squared Error

m : Anzahl der Einträge in D

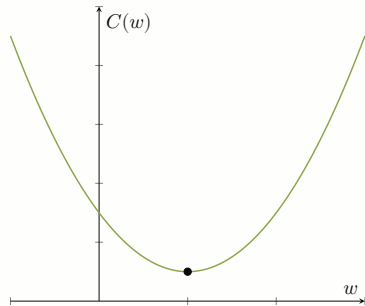
d : Eintrag (Pixelvektor) in D

$$C_D(\vec{W}) = \frac{1}{m} \sum_{d \in D} \left(\sum_{i=0}^9 (a_i^{(3,d)} - s_i^{(d)})^2 \right)$$

8.1

Fehler minimieren

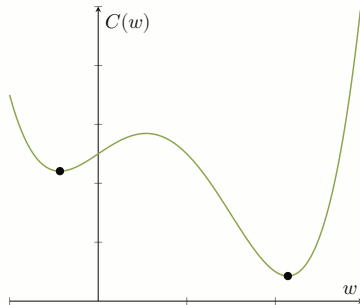
Globales Minimum durch Ableiten



8.2

Fehler minimieren

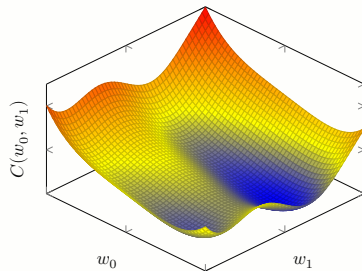
Zwei lokale Minima; globales Minimum durch Vergleich



8.3

Fehler minimieren

Finden lokaler Minima durch
Ableiten nicht möglich



8.4

Fehler minimieren

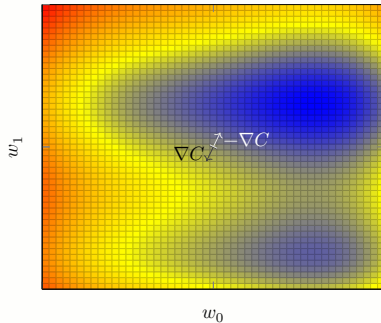
Gradient Descent

$\nabla C(\vec{W})$: Gradient der Fehlerfunktion C an Stelle \vec{W}
Vektor in Richtung des größten Anstiegs

η : Lernrate

Springende Werte verhindern

$$\vec{W}_{neu} = \vec{W} - \eta \nabla C(\vec{W})$$



8.5

Fehler minimieren

Gradient Descent

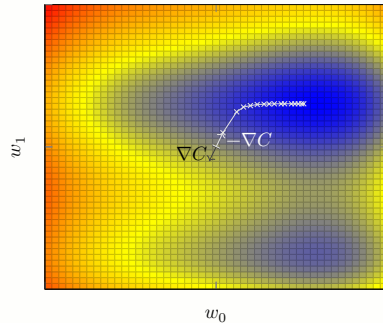
$\nabla C(\vec{W})$: Gradient der Fehlerfunktion C an Stelle \vec{W}

Vektor in Richtung des größten Anstiegs

η : Lernrate

Springende Werte verhindern

$$\vec{W}_{neu} = \vec{W} - \eta \nabla C(\vec{W})$$

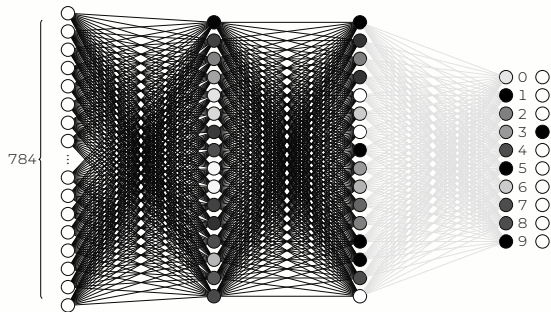


9.1

Gradienten bestimmen

Backpropagation

Bias verändern



Modellierung



Künstliche Neuronen



Training



9.2

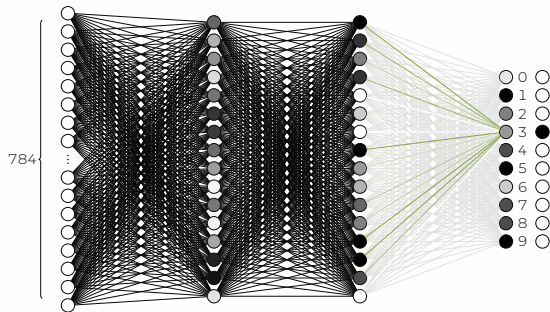
Gradienten bestimmen

Backpropagation

Bias verändern

Gewichte verändern

*Proportional zu Neuronen aus
vorheriger Schicht*



Modellierung



Künstliche Neuronen



Training



9.3

Gradienten bestimmen

Backpropagation

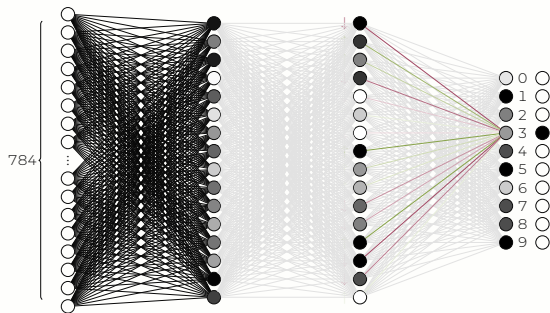
Bias verändern

Gewichte verändern

*Proportional zu Neuronen aus
vorheriger Schicht*

Neuronen verändern

*Gewichte aus vorheriger Schicht
verändern*



Modellierung



Künstliche Neuronen



Training



- [Mös17] Cedric Mössner. *Neuronale Netze*. 2017
- [San17] Grant Sanderson. *Neural Networks. The basics of neural networks, and the math behind how they learn*. 2017
- [Wik23] Wikipedia. *Künstliches neuronales Netz*. 2023

Jasper Gude

Hockenheim, 3. Dezember
2023

