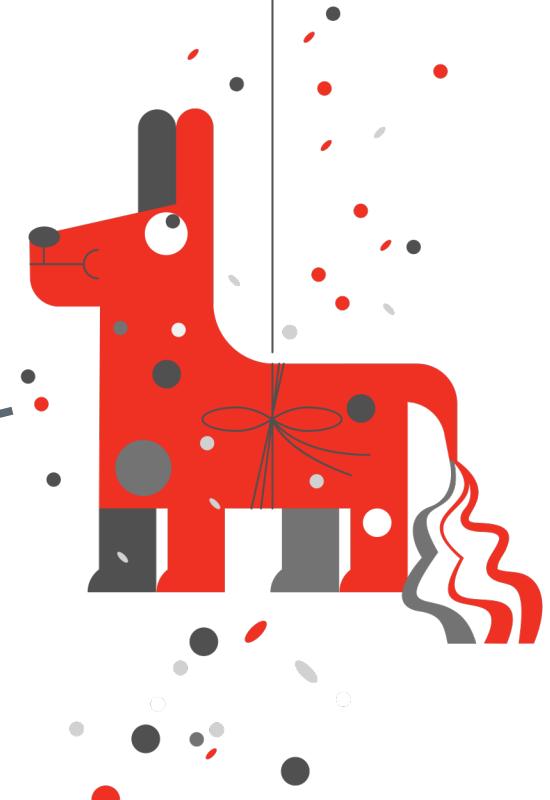
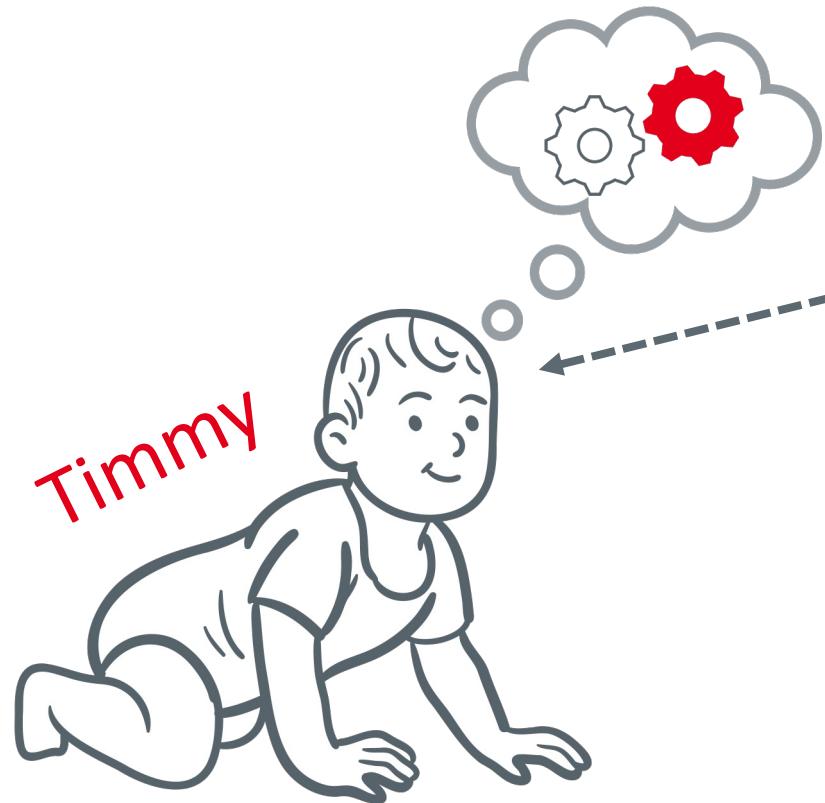


Neuronale Netze

eMail: neuronale.netze@dhbw-ravensburg.de
Name: <https://www.dhbw-ravensburg.de/Personal/NeuronaleNetze>



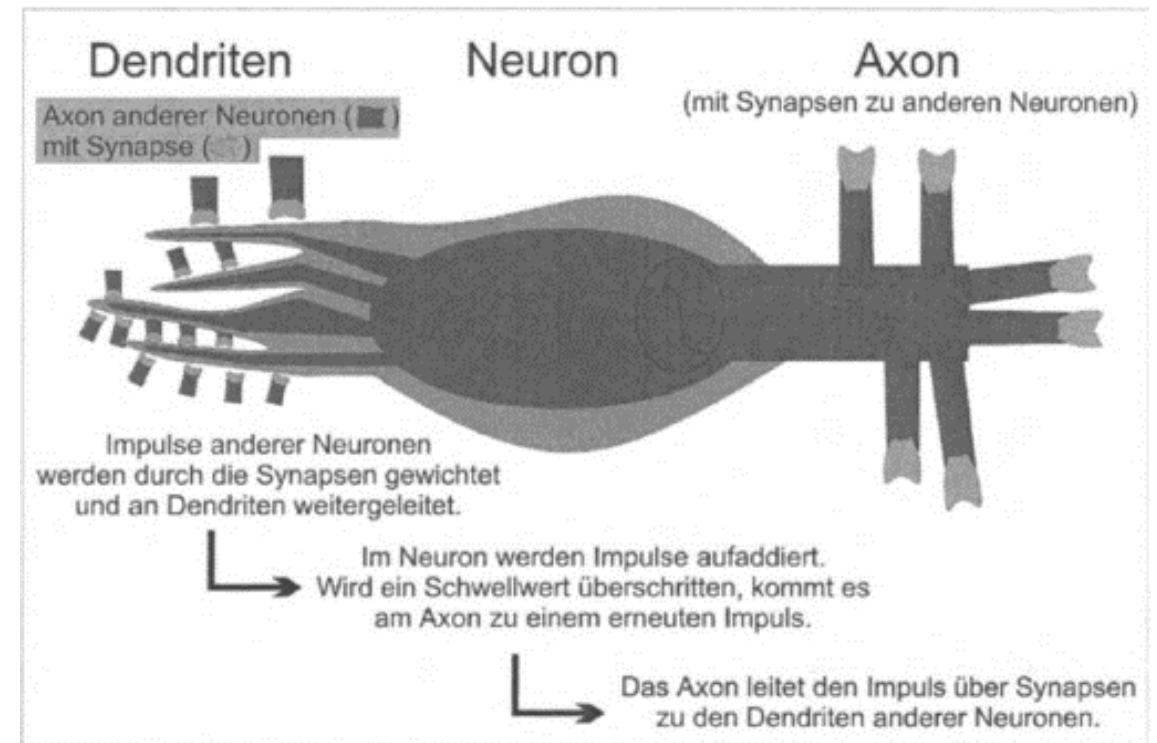
Biologie – Beispiel



Biologie

Pyramidenzelle

- Pyramidenzelle:
 - Synapsen: Übertragung und Gewichtung
 - Dendriten: Signal Aufnahme
 - Neuron: Signal Aufaddierung und Aktivierung
 - Axon: Sendet Signale an andere Neuronen
- 100 Milliarden Neuronen
Jeweils 1.000 bis 10.000 Synapsen



Geschichte



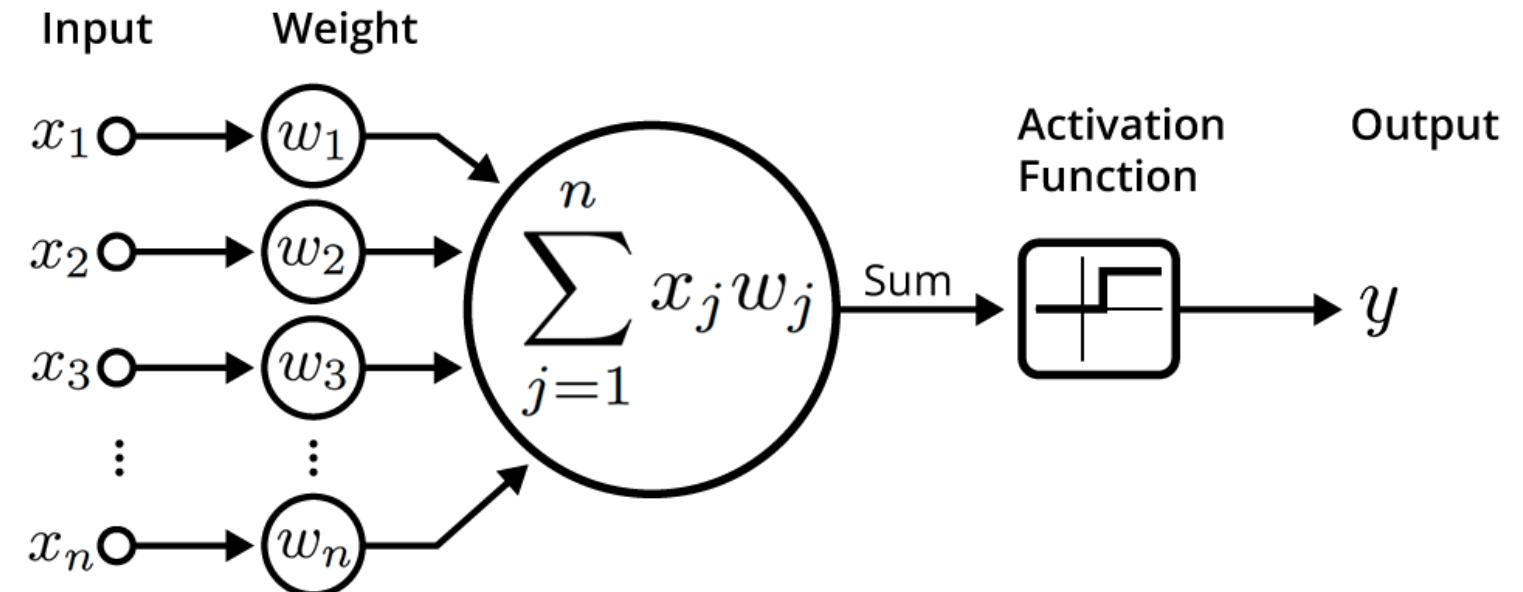
Geschichte



Perceptron

Von Frank Rosenblatt

- klassische künstliche Neuroneneinheit
- gerichteter Graph
- Schwellenwertfunktion
- binäre Ausgabe



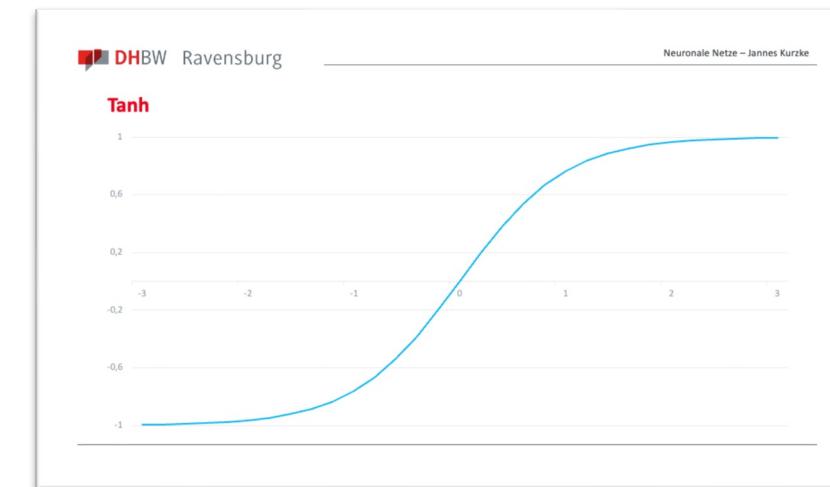
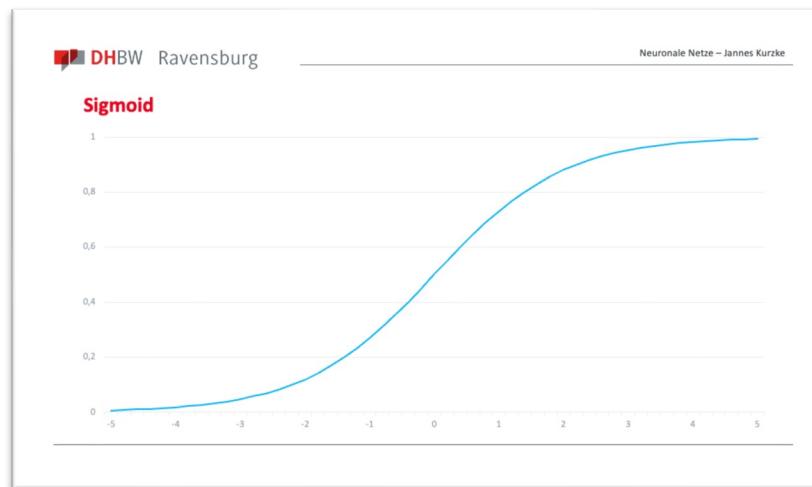
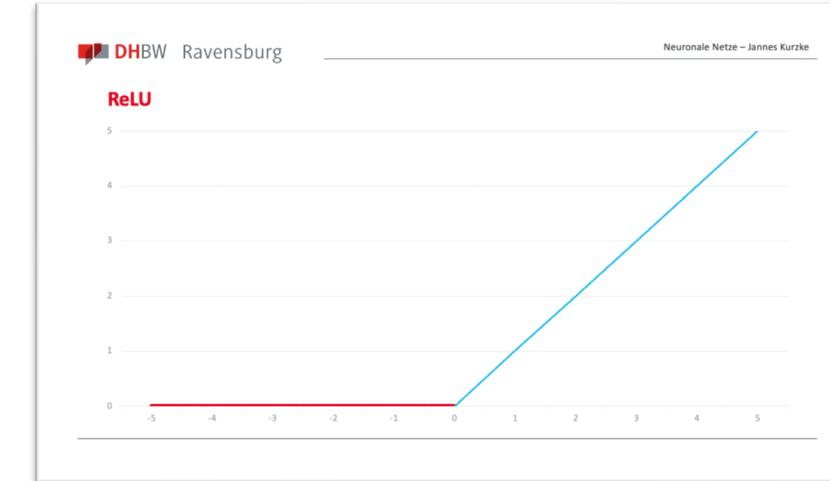
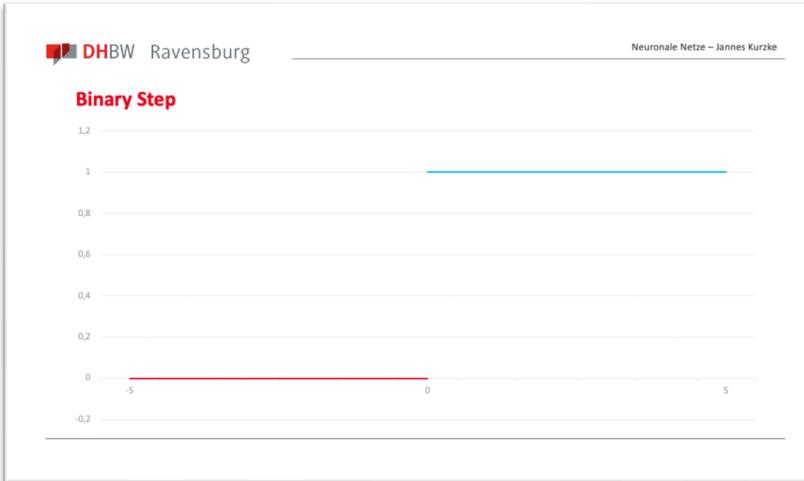
klassisch:

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

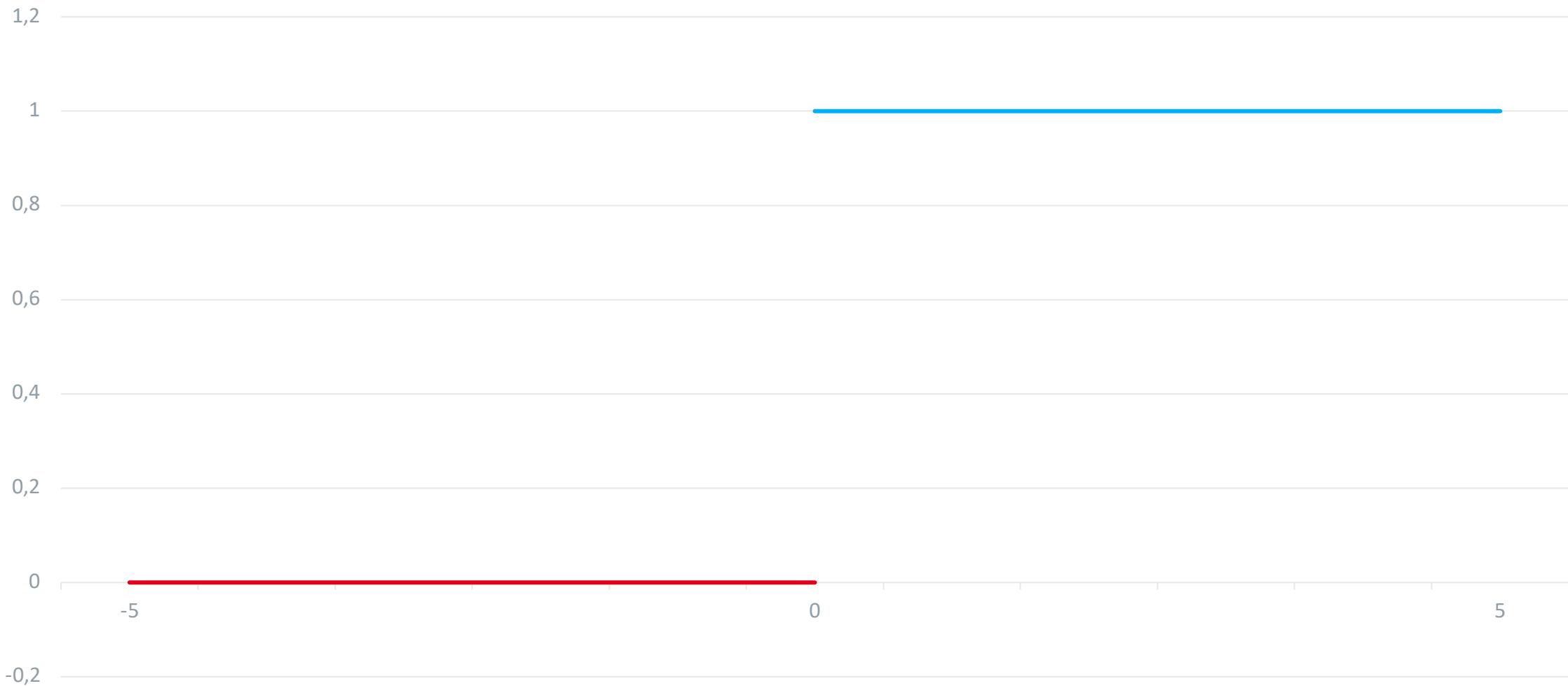
mit bias:

$$\text{output} = \begin{cases} 0 & \text{if } \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases}$$

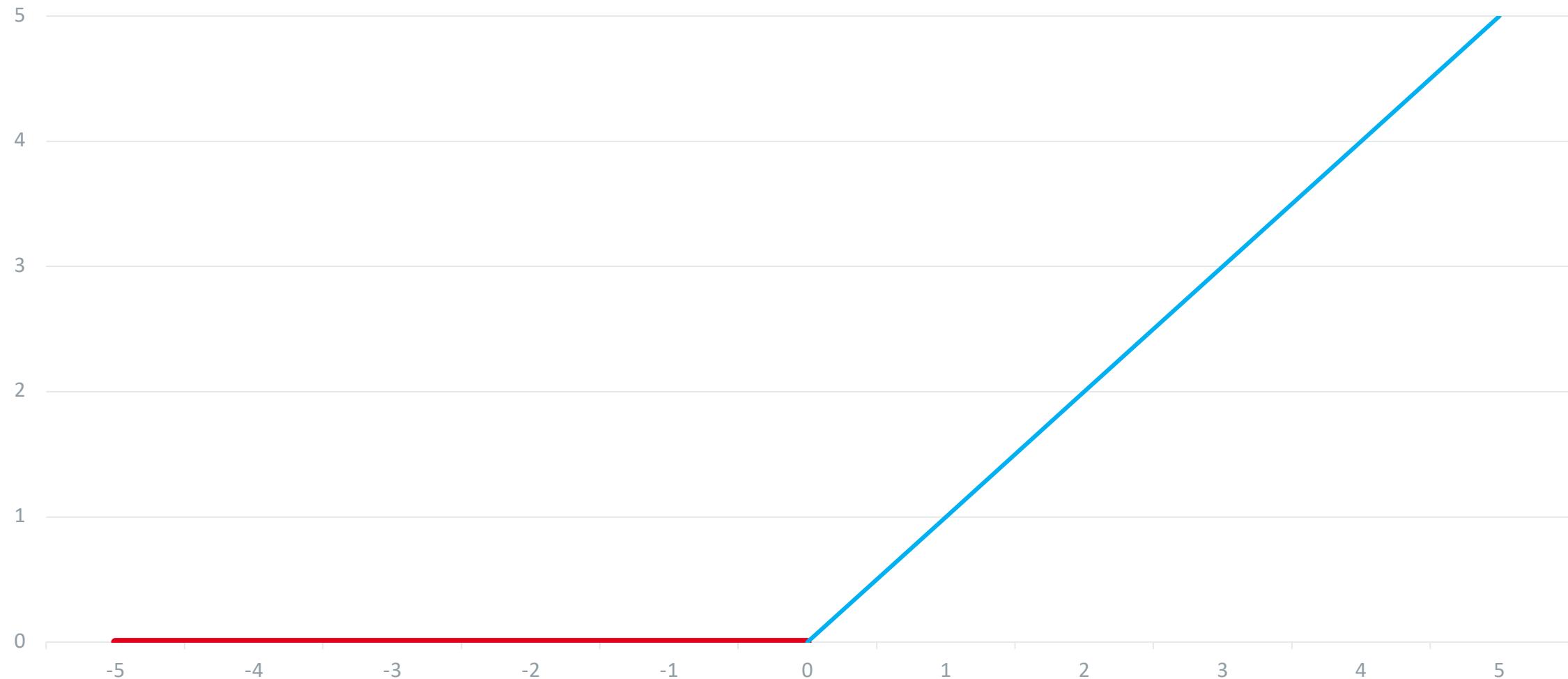
Aktivierungs-Funktionen



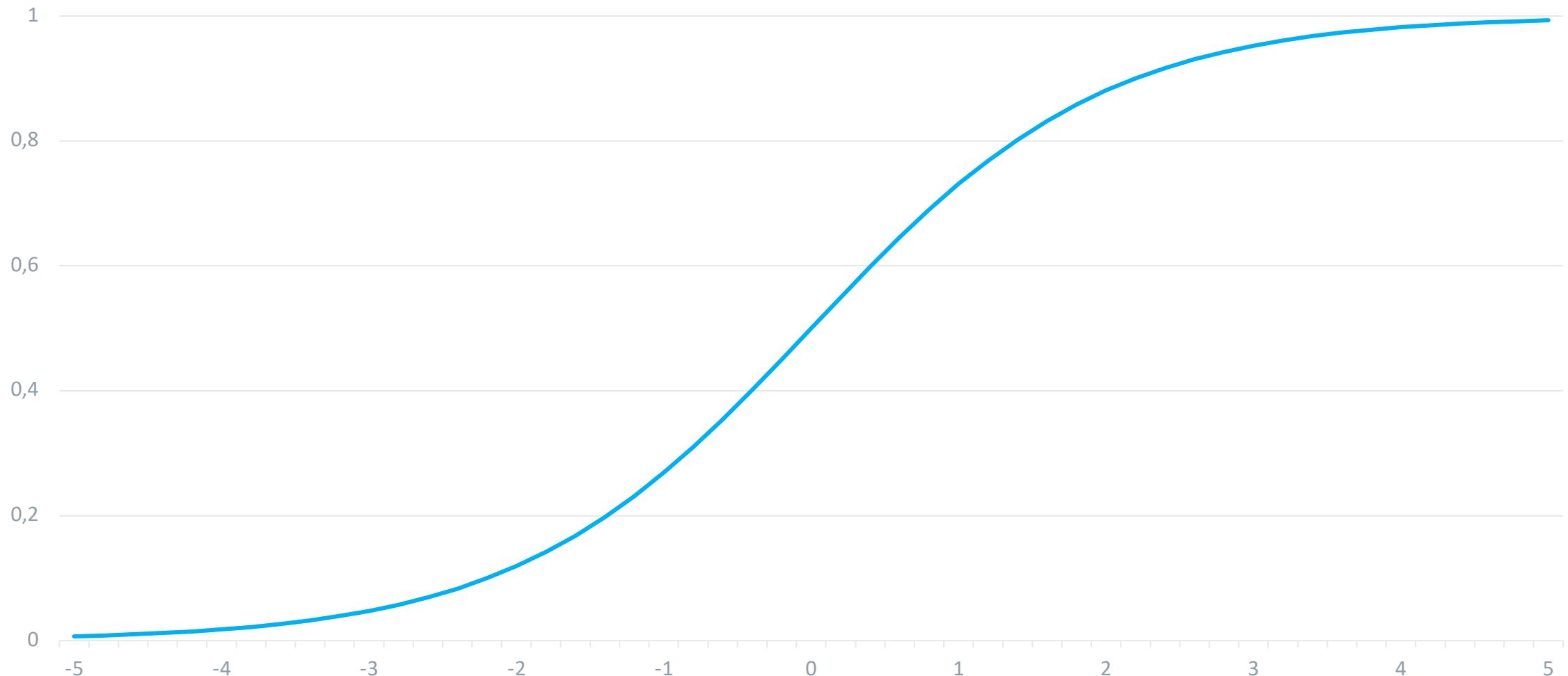
Binary Step



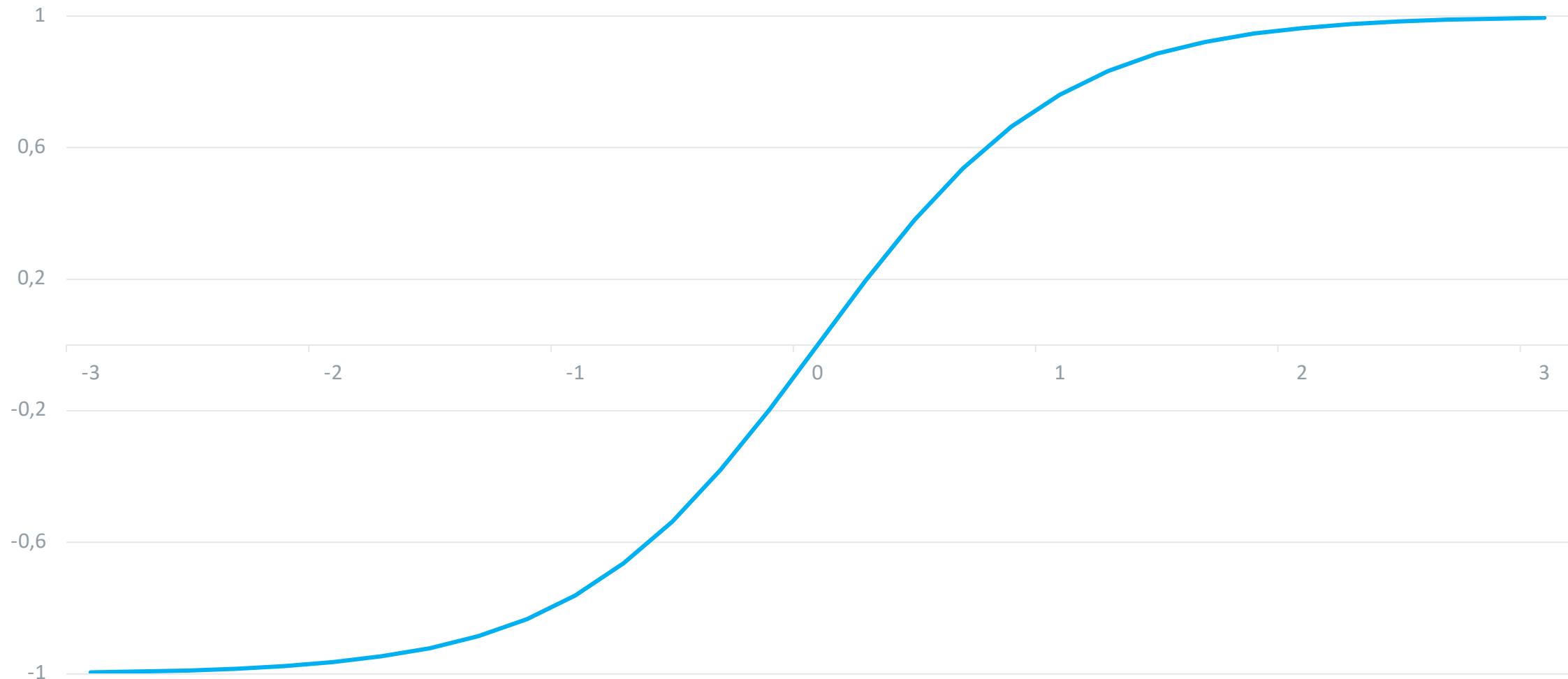
ReLU



Sigmoid



Tanh



Trainings Parameter Neuronaler Netze

Training Loss



Trainings Daten

Abweichung Erwartung gegen Ergebnis

Niedrigerer Wert -> Besser

Validation Loss



Validation Daten

Abweichung Erwartung gegen Ergebnis

Niedrigerer Wert -> Besser

Differenz Training Loss beachten

Accuracy



Alle Daten

Hoher Wert -> Besser

Prozentsatz einer korrekten Vorhersagung

Lernmethoden

Supervised Learning



Datenpaare mit Ergebnis

Vorhandene Daten werden gesplittet

Regression oder Klassifikation

Unsupervised Learning



Datenset ohne „Ergebnisse“

Modell trainiert mit gesamten Daten

Clusterung, Assoziation, Anomalie
Erkennung

Reinforcement Learning



Training in Simulationsumgebung

Trial and Error

Entscheidungsprozesse

Fehlerbehandlung?

Underfitting



Schlecht bereinigte Daten / Noise



Biased Datenset

Zu simples Modell



Neue Features generieren

Dropout reduzieren

Overfitting



Schlecht bereinigte Daten / Noise



Hohe Varianz in Daten

Zu komplexes Modell



Mehr Daten hinzufügen (Diversität steigern)

Dropout erhöhen

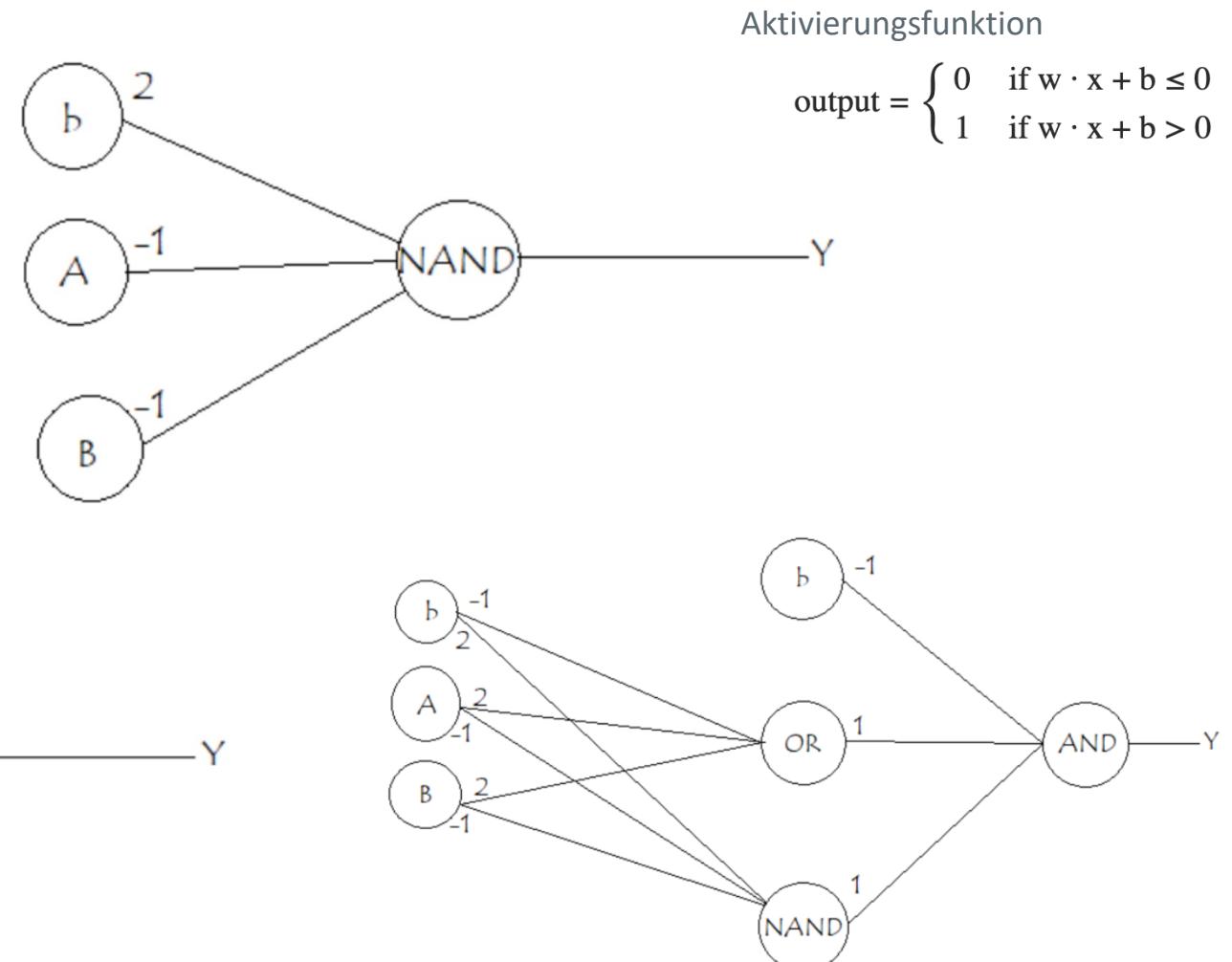
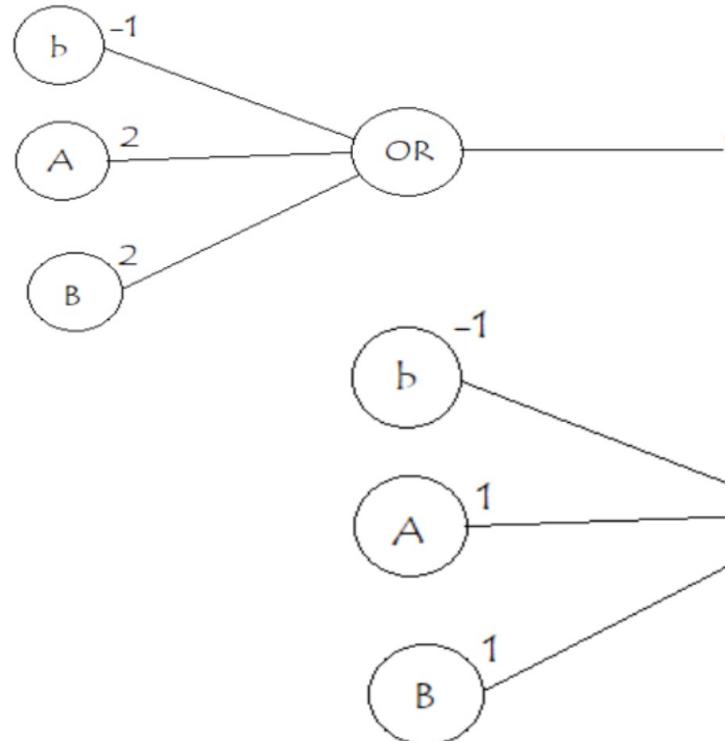
02

Künstliche Neuronale Netze



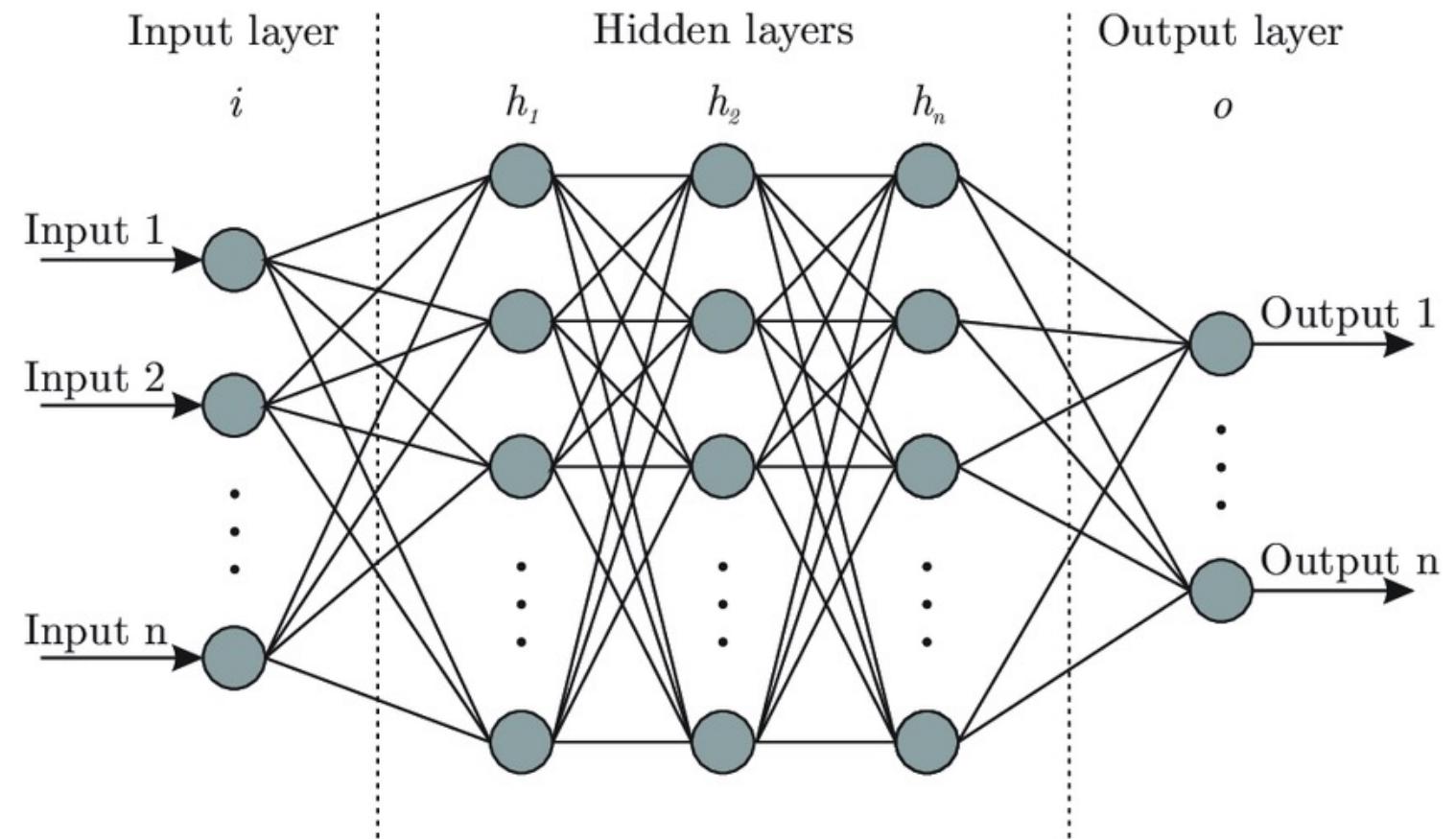
Beispiele

Grenzen des Perceptrons

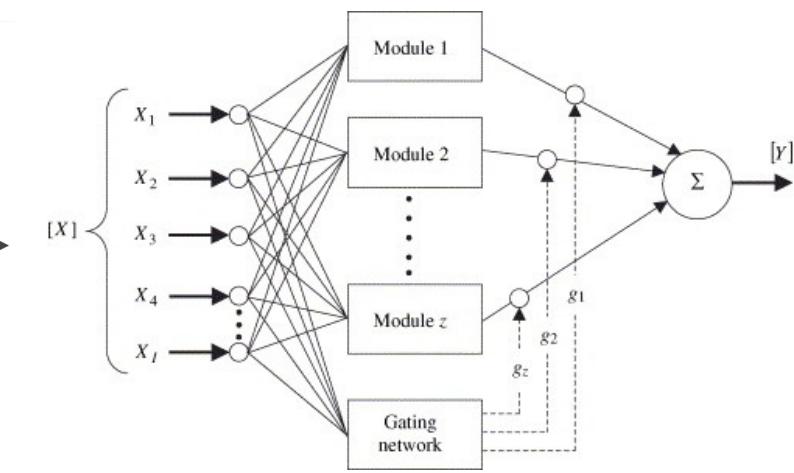
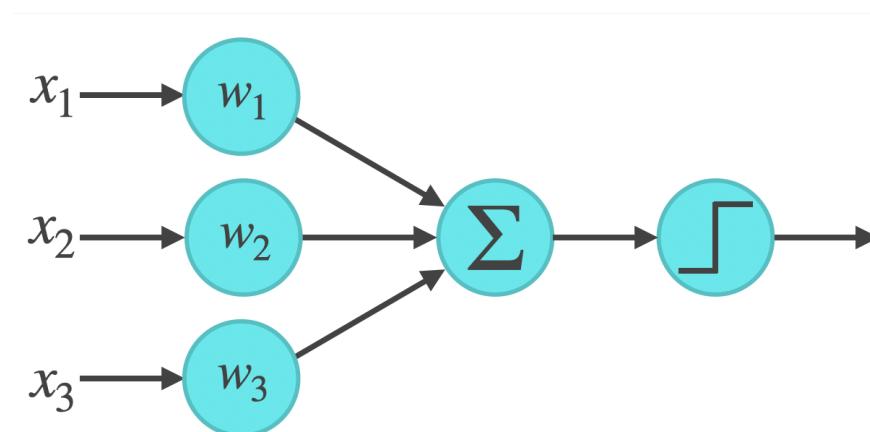
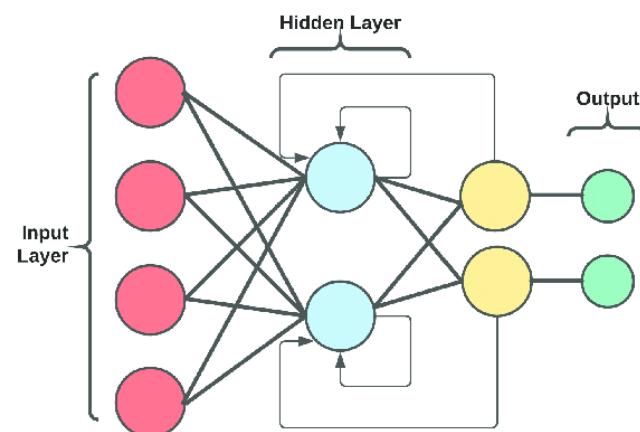
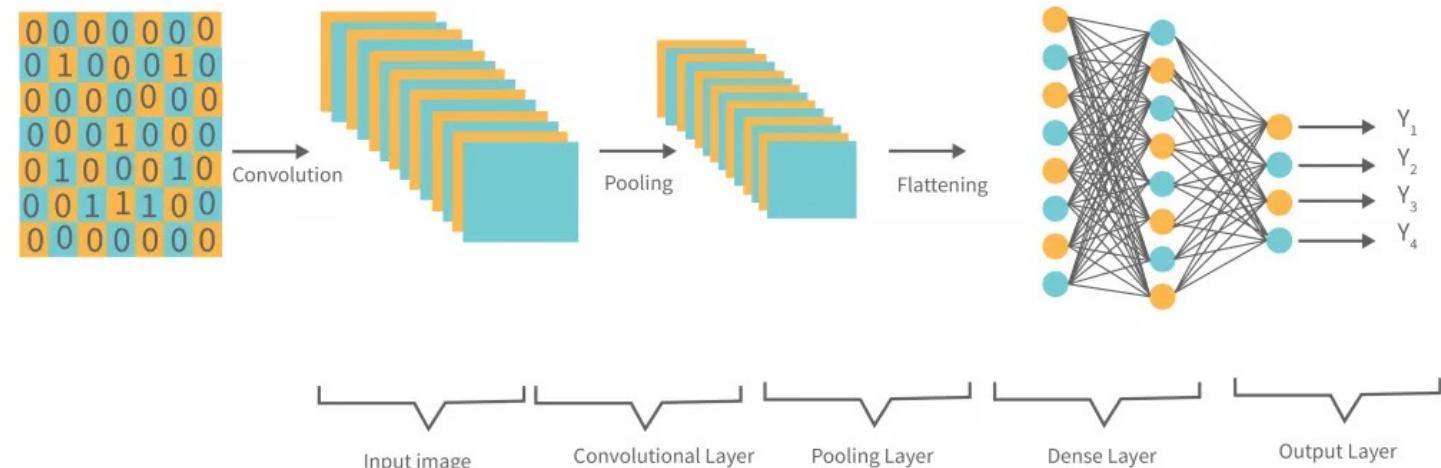
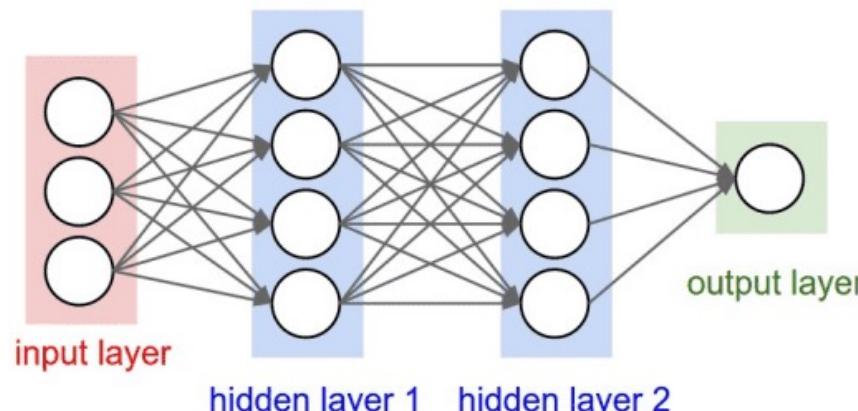


Basic Neural Network

- hier multi-layer perceptrons (MLPs)
 - Input Layer, mind. 1 hidden Layer, Output Layer



Erweiterungen



Feedforward-Netzwerk

- Informationsfluss nur in eine Richtung
- Single Layer Perceptron – simples feedforward neural network, keine hidden layers
- Multi Layer Perceptron (MLP) – beinhaltet mind. 1 hidden layer

- meist nichtlineare Aktivierungsfunktionen (Sigmoid, ReLU, tanh)
- fully connected/ sparse connected
- >1 hidden Layers: Deep Network
- oft Verbunden mit Backpropagation

Feedforward-Netzwerk

Was ist der momentane Output?

1. total net input h1:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$net_{h1} = 0.15 * 0.05 + 0.2 * 0.1 + 0.35 * 1 = 0.3775$$

2. output h1 nach sigmoid function:

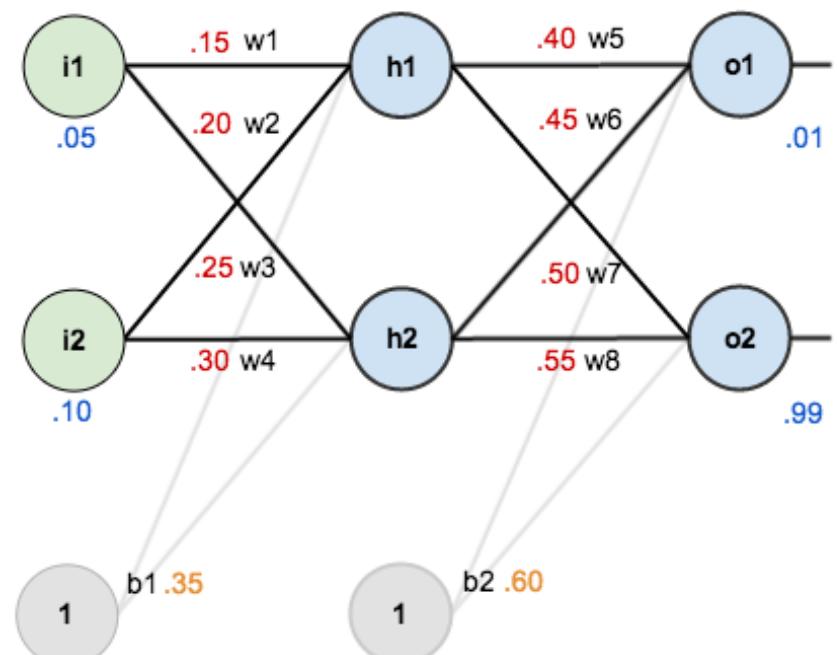
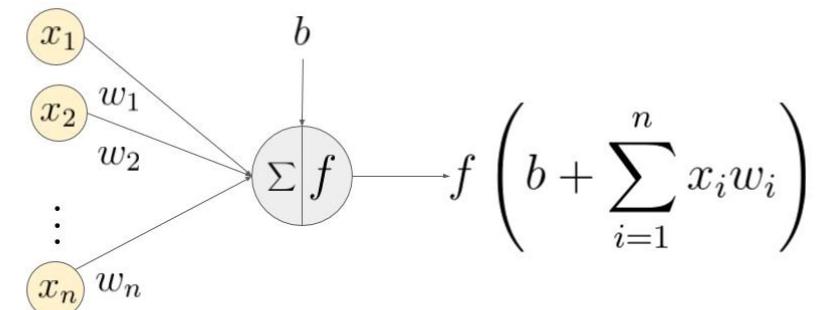
$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}} \approx 0.593269992$$

$$out_{h2} \approx 0.596884378$$

3. wiederholen für o₁ and o₂, (mit out_{h1} und out_{h2} als Inputs)

$$out_{o1} \approx 0.75136507, out_{o2} \approx 0.772928465$$

wie konfigurieren für 0.05 -> 0.1, 0.1 -> 0.99?



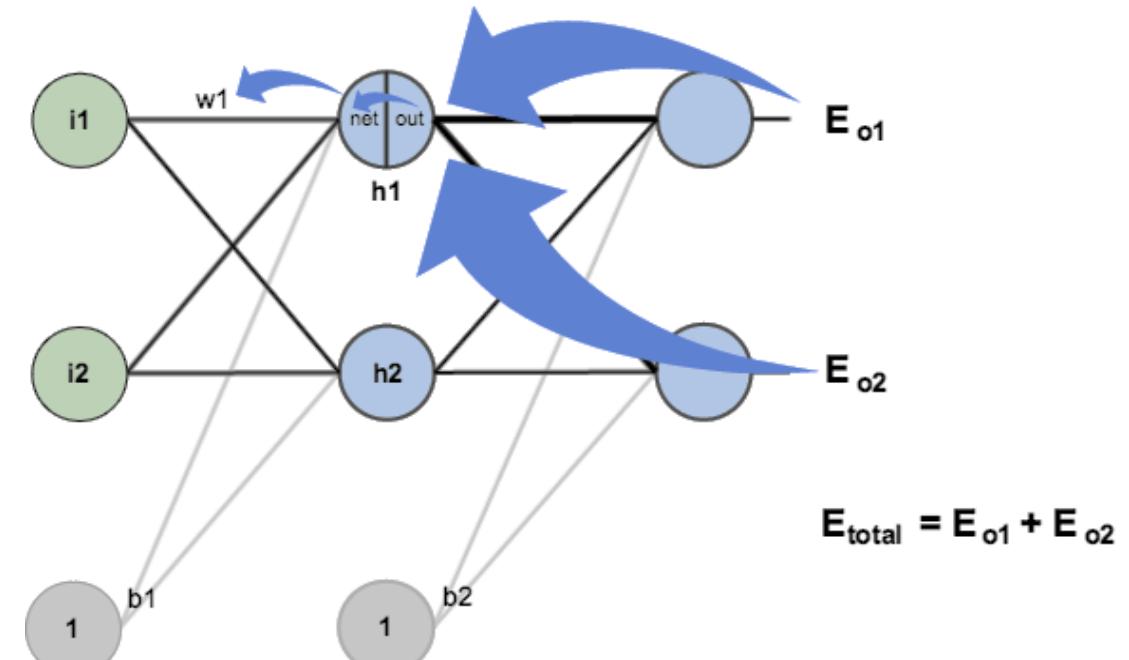
Backpropagation

Ziel: 0.05 -> 0.01, 0.1 -> 0.99?

- Rückwärtsdurchlauf
 - Gradienten der ersten Schicht berechnen
 - $\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$
 - Gradienten der zweiten Schicht berechnen
 - $\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$
- Gewichtsaktualisierung
 - $w_5^* = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} \approx 0.3589$
 - $w_1^* = w_1 - \eta * \frac{\partial E_{total}}{\partial w_1} \approx 0.1498$
- Wiederholung
 - ohne Back Fehler: 0.2983, 1. Durchlauf Fehler: 0.2910

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

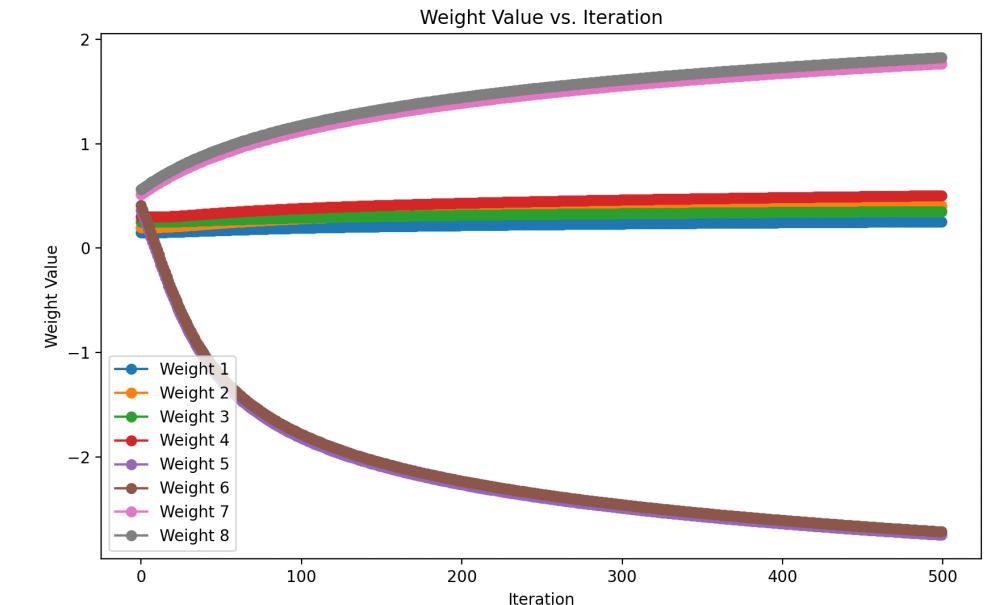
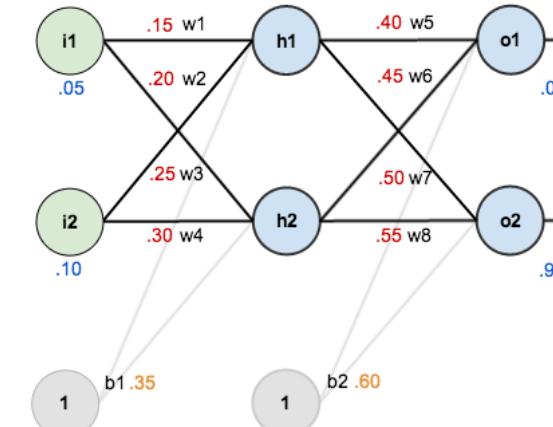
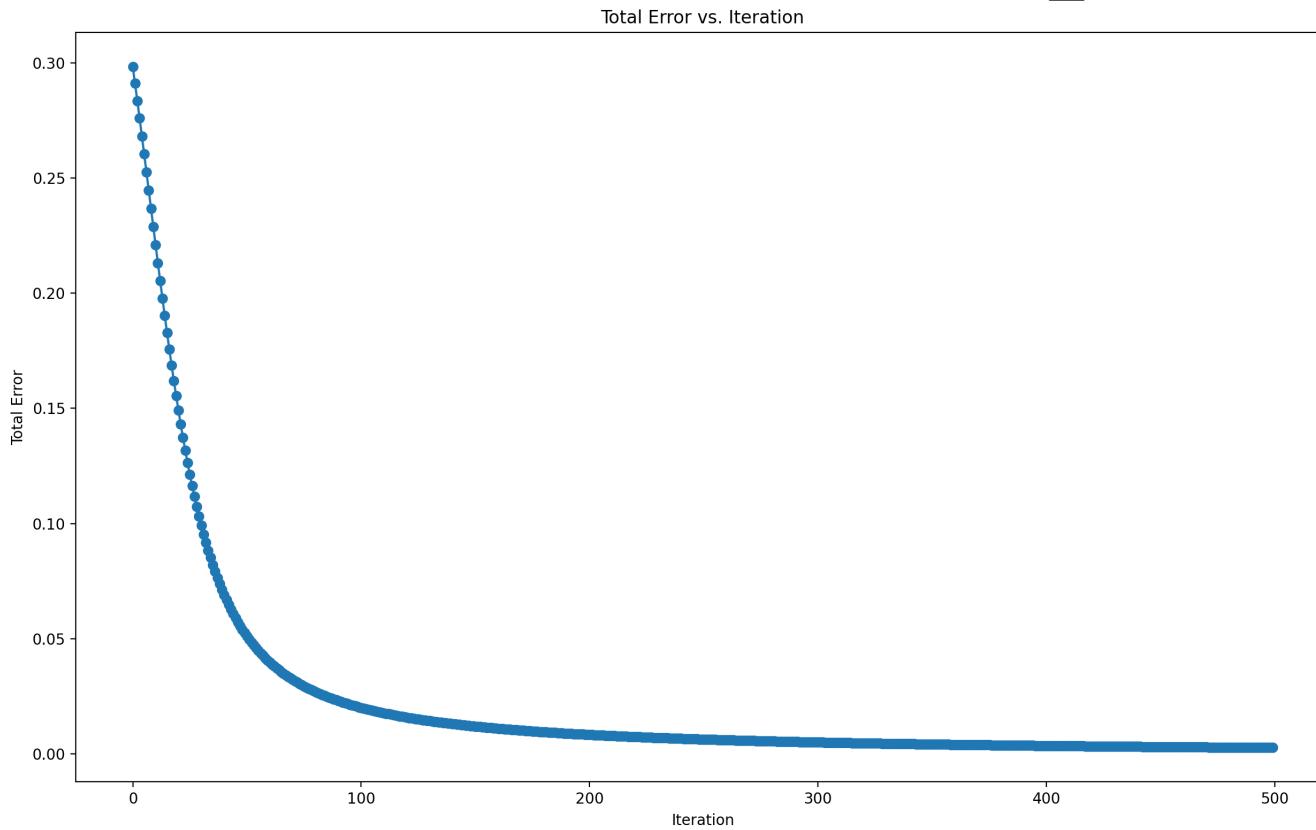
$$\begin{aligned} E_{total} &\approx \frac{1}{2} * (0.01 - 0.751)^2 + \frac{1}{2} * (0.99 - 0.772)^2 \\ E_{total} &\approx 0.298 \end{aligned}$$



Backpropagation

Python Code Output

$$E_{total} = \sum \frac{1}{2} (target - output)^2$$

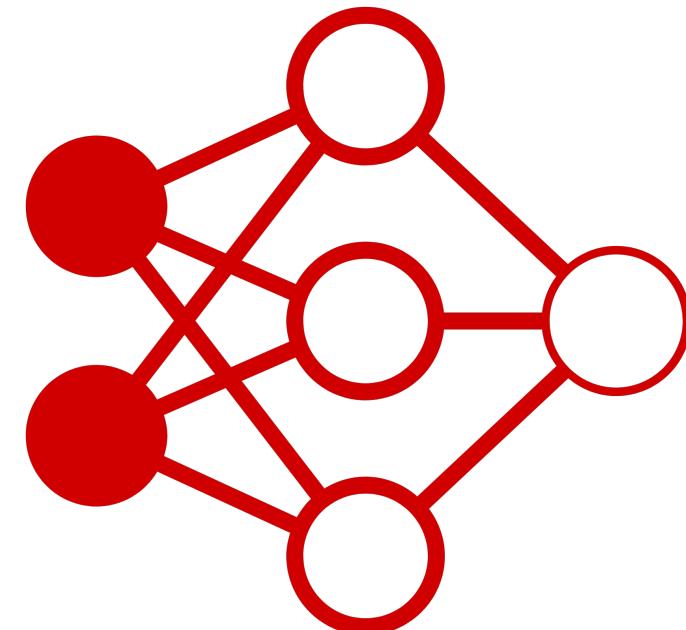


-> Welche Anwendung hat nun die Theorie?

Anwendungsbeispiele

Von Neuronalen Netzen

- Bilderkennung & Autonomes Fahren
- Vorhersagen & Betrugserkennung
- Large Language Models (LLM's)



03

Check GPT



Large Language Models (LLM's)

Eigenschaften & Merkmale

- Große, Neuronale Netze (oft Milliarden von Parametern)
- Trainiert auf riesigen Textdatensätzen
- Lernen Beziehungen zwischen Wörtern und Sätzen
- Können Texte generieren, Fragen beantworten, ...

Chat GPT

Einordnung der Begriffe

- **Chat**
- **Generative**
- **Pre-trained**
- **Transformer**



Attention Is All You Need

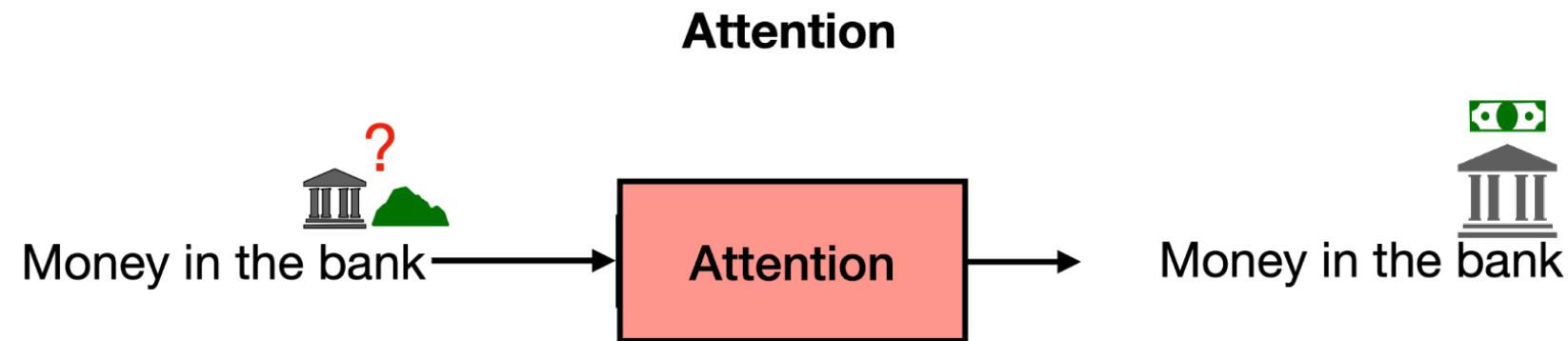
Einführung in die Transformer Architektur

- **Chat**
- **Generative**
- **Pre-trained**
- **Transformer**



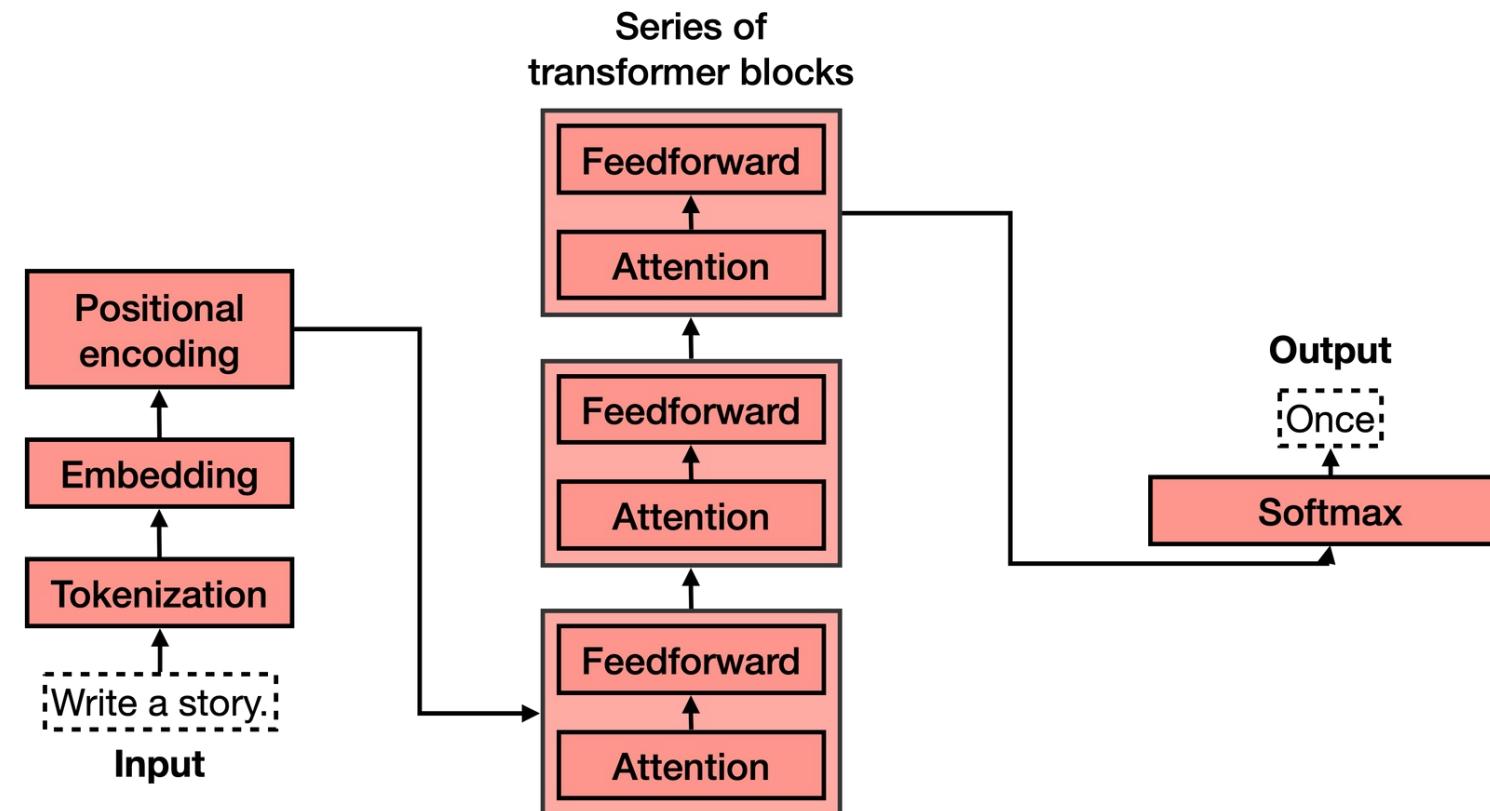
Attention Is All You Need

Einführung in die Transformer Architektur



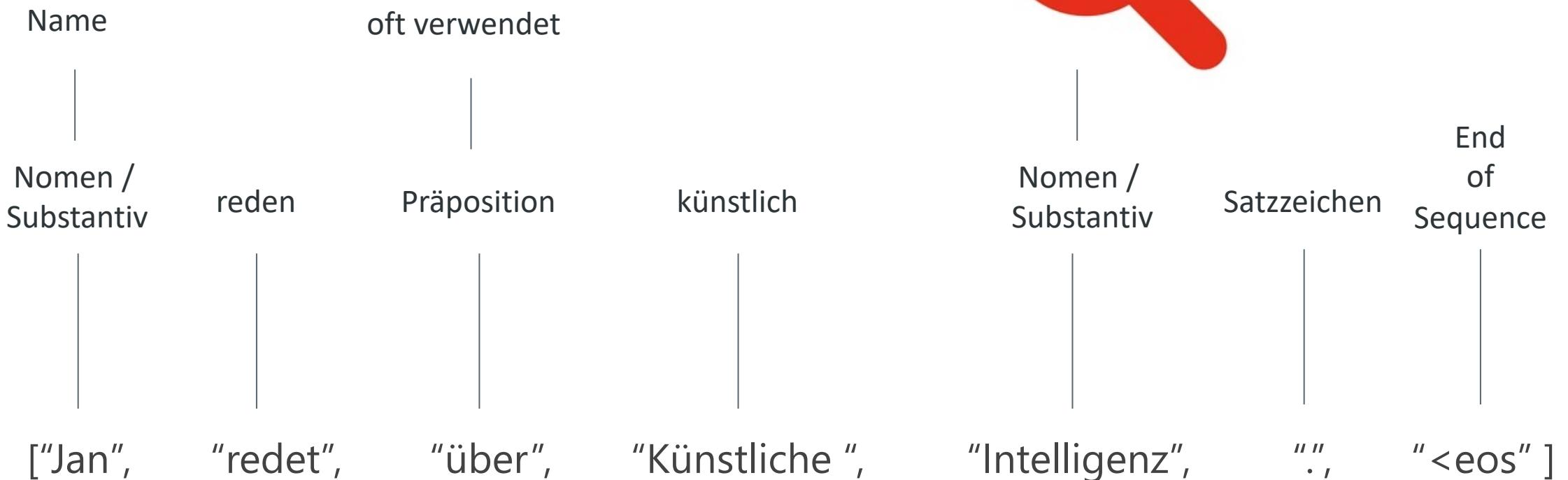
Attention Is All You Need

Einführung in die Transformer Architektur



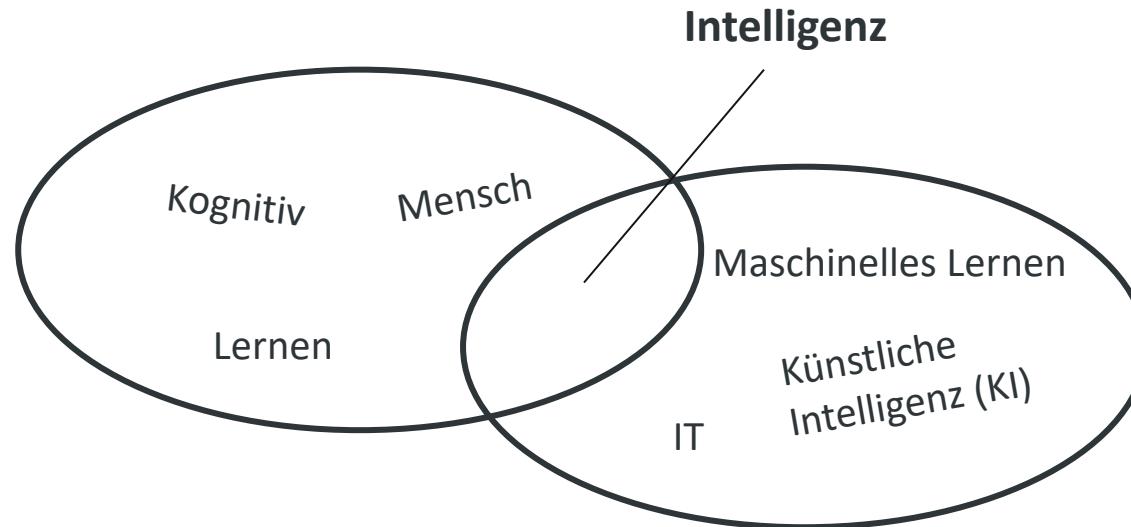
Attention Is All You Need

Vom Text zur Eingabe



Attention Is All You Need

Kontext & Wortfelder



[“Jan”, “redet”, “über”, “Künstliche”, “Intelligenz”, “.”, “<eos”]

Attention Is All You Need

Vom Text zu Parametern

$$\begin{array}{cccccc} \mathbf{10010} & \mathbf{10100} & \mathbf{01010} & \mathbf{01001} & \mathbf{11011} & = \begin{pmatrix} 10010 \\ 10100 \\ 01010 \\ 01001 \\ 11011 \end{pmatrix} \\ | & | & | & | & | & \\ ["Jan", & "redet", & "über", & "Künstliche ", & "Intelligenz".] & \end{array}$$

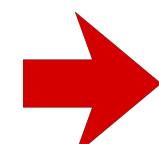
Attention Is All You Need

Parameter & Gewichte



170 Milliarden Parameter

über 100 Billionen Parameter

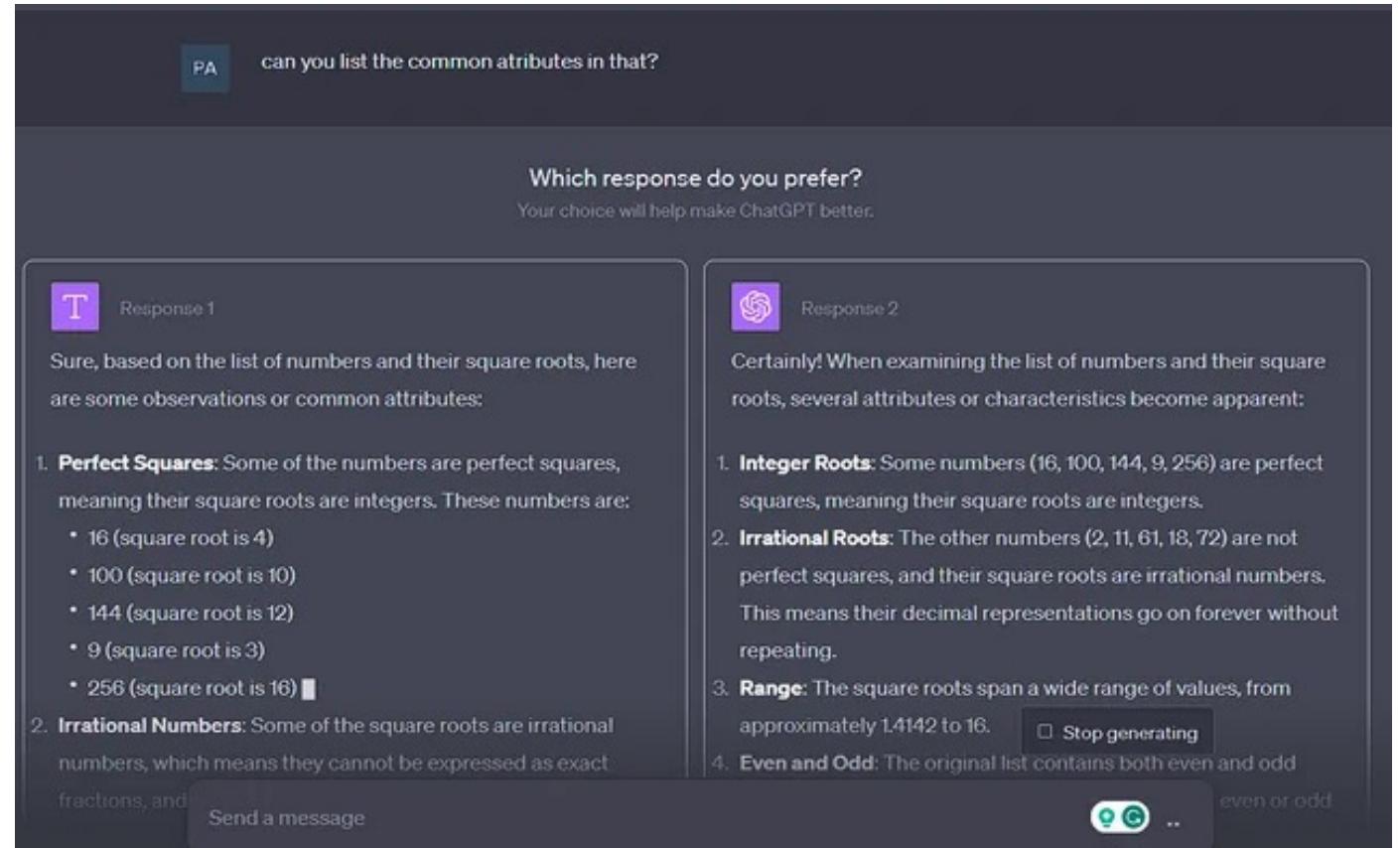


Anzahl Parameter ist ein maßgeblicher Indikator, wie gut ein Neuronales Netz arbeitet

Chat GPT

Wie erreichen LLM's einen optimalen Output?

- Supervised Learning
- Reinforcement Learning from Human Feedback (RHLF)



PA can you list the common attributes in that?

Which response do you prefer?
Your choice will help make ChatGPT better.

T Response 1

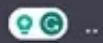
Sure, based on the list of numbers and their square roots, here are some observations or common attributes:

1. **Perfect Squares:** Some of the numbers are perfect squares, meaning their square roots are integers. These numbers are:
 - 16 (square root is 4)
 - 100 (square root is 10)
 - 144 (square root is 12)
 - 9 (square root is 3)
 - 256 (square root is 16) ■
2. **Irrational Numbers:** Some of the square roots are irrational numbers, which means they cannot be expressed as exact fractions, and

G Response 2

Certainly! When examining the list of numbers and their square roots, several attributes or characteristics become apparent:

1. **Integer Roots:** Some numbers (16, 100, 144, 9, 256) are perfect squares, meaning their square roots are integers.
2. **Irrational Roots:** The other numbers (2, 11, 61, 18, 72) are not perfect squares, and their square roots are irrational numbers. This means their decimal representations go on forever without repeating.
3. **Range:** The square roots span a wide range of values, from approximately 1.4142 to 16.
4. **Even and Odd:** The original list contains both even and odd

Send a message even or odd 

Schwächen

Von Neuronalen Netzen

- Lange Trainingszeiten, Bedarf an riesigen Datenmengen
- Antworten werden nicht begründet, fehlende Quellen (Blackbox)
- Generalisierungsfähigkeit nicht garantiert (Overfitting)
- Halluzination

Check GPT

Resümee

“

“Our intelligence is what
makes us human, and AI is an
extension of this quality.”

Yann LeCun

- Chief AI Scientist at Meta

Vielen Dank

Für Eure Aufmerksamkeit!

