



Deep Learning mit Python & TensorFlow

DR. CHIAO-YA CHANG
CHIAOYA.CHANG.TW@GMAIL.COM

Tag 1 – Grundlagen Python



Tag 1 10:00-10:30

Begrüßung & Einführung

- Kursziele
- Ablauf
- Erwartungsklärung
- Vorstellungsrunde

Kursziele:

Einen verständlichen & praxisnahen Einstieg in Deep Learning ermöglichen.

Am Ende des Kurses sollen die Teilnehmenden:

- die **Grundlagen der Programmiersprache Python** verstehen und einfache Programme lesen und anpassen können
- ein grundlegendes **Verständnis von Künstlicher Intelligenz und Deep Learning** entwickeln
- wissen, wie neuronale Netze aufgebaut sind
- erste Erfahrungen mit **TensorFlow** sammeln
- ein **einfaches neuronales Netz selbst erstellen, trainieren und auswerten**
- einschätzen können, **wie Deep Learning in der Praxis eingesetzt** wird und wie sie ihr Wissen weiter vertiefen können



Ablauf

	Tag 1 – Python & Datenanalyse		Tag 2 – Deep Learning	
10:00–10:30	Begrüßung & Einführung	Kursziele, Ablauf, Erwartungsklärung, Vorstellungsrund	Rückblick & Zielsetting	Wiederholung & Einführung in den zweiten Tag
10:30–11:30	Python-Grundlagen	Datentypen, Variablen, Schleifen, Funktionen	Verständliche Erklärung: Was ist Deep Learning und wie funktioniert es?	Neuronale Netze, TensorFlow Grundlagen
11:30–12:30	Arbeiten mit Daten (Pandas)	Daten laden, filtern, berechnen	Einführung in Deep Learning	
12:30–13:10	Mittagspause	—	Mittagspause	—
13:10–15:00	Datenvisualisierung	Matplotlib / Seaborn: Diagramme & Insights	Praktische Übung: Einfaches Modell	Modell bauen & trainieren mit TensorFlow
15:00–15:20	Pause	—	Pause	—
15:20–16:40	Mini-Projekt: Explorative Analyse mit Python	Gruppenarbeit mit realem Datensatz	Modellbewertung & Interpretation	Ergebnisse analysieren, Verbesserungen diskutieren
16:40–17:00	Ergebnisse & Diskussion	Präsentation, Reflexion, Q&A	Ergebnisse & Diskussion	Präsentation, Reflexion, Q&A



Erwartungsklärung:

Keine Vorkenntnisse in Programmierung
oder Python erforderlich

Die Teilnehmenden sollten jedoch:

- **Neugier und Interesse** an Künstlicher Intelligenz und neuen Technologien mitbringen
- bereit sein, **aktiv mitzuarbeiten und selbst auszuprobieren**
- grundlegende **PC-Kenntnisse** besitzen (z. B. Umgang mit Maus, Tastatur und Dateien)
- sich darauf einstellen, dass Programmieren ein **Lernprozess mit Übung und Geduld** ist

Der Kurs richtet sich an Anfänger*innen und legt Wert auf eine **ruhige, verständliche und unterstützende Lernatmosphäre**.



A photograph of a woman with long dark hair, wearing a yellow ribbed sweater, sitting at a desk and working on a computer. She is looking towards the right. The background shows an office environment with other desks and monitors. A graphic element consisting of two overlapping white ovals and a yellow circle is overlaid on the left side of the slide.

Vorstellungsrunde: Über mich

Dr. Chiao-Ya Chang

- **Promotion** in Agrarökonomie an der Universität Bonn
- Freiberufliche Data Spezialistin & Trainerin für Datenanalyse und Künstliche Intelligenz
- Supply Chain Analytics and Simulation Manager in der Crop Science Abteilung bei Bayer AG
- Commercial Data Scientist in der Pharmaceutical Abteilung bei Beyer AG
- Data Scientist bei Lidl Stiftung
- **Webseite:** www.drchiaoyachang.com

Vorstellungsrunde:

Wir lernen uns kennen

- **Name**
- **Erfahrungen mit Python und Deep Learning**
- **Motivation**
- **Erwartungen**
- **Sonstiges**





Tag 1 10:30-11:30

Python-Grundlagen

- Datentypen
- Variablen
- Schleifen
- Funktionen

Aufgabe: Python Installation

- Link: <https://www.python.org/>
- Jupyter Notebook file: file_name.**ipynb**
- Python file: file_name.**py**
- **Hausaufgaben:** Python installation mit privaten PC/Laptop



Tag 1 11:30-12:30

Arbeiten mit Daten (Pandas)

- Daten laden, filtern, berechnen

Was ist Pandas? Warum Pandas verwenden?

Pandas ist eine [Python-Bibliothek](#), die zur Arbeit mit **Datensätzen** verwendet wird.

Pandas bezieht sich sowohl auf „**Panel Data**“ als auch auf „**Python Data Analysis**“.

Funktionen zum:

- Analysieren, bereinigen und erkunden von Daten
- Verändern und Bearbeiten von Daten

Pandas ermöglicht es uns:

- große Datenmengen zu analysieren
- Schlussfolgerungen auf Basis statistischer Methoden zu ziehen



12:30-13:10 ☕ Mittagspause



Tag 1 13:10-15:00

Datenvisualisierung

- Matplotlib / Seaborn
- Diagramme & Insights

15:00–15:20  Pause



Tag 1 15:20-16:40

Mini-Projekt: Explorative Analyse mit Python

- Gruppenarbeit mit realem Datensatz



15:00–15:20

Tag 1 16:40–17:00

Ergebnisse & Diskussion

- Präsentation
- Reflexion
- Q&A

Tag 2 – Gemeinsames Erstellen eines Neuronalen Netzes



Tag 2 10:00-10:15

Rückblick & Zielsetzung

- Wiederholung
- Einführung in den zweiten Tag

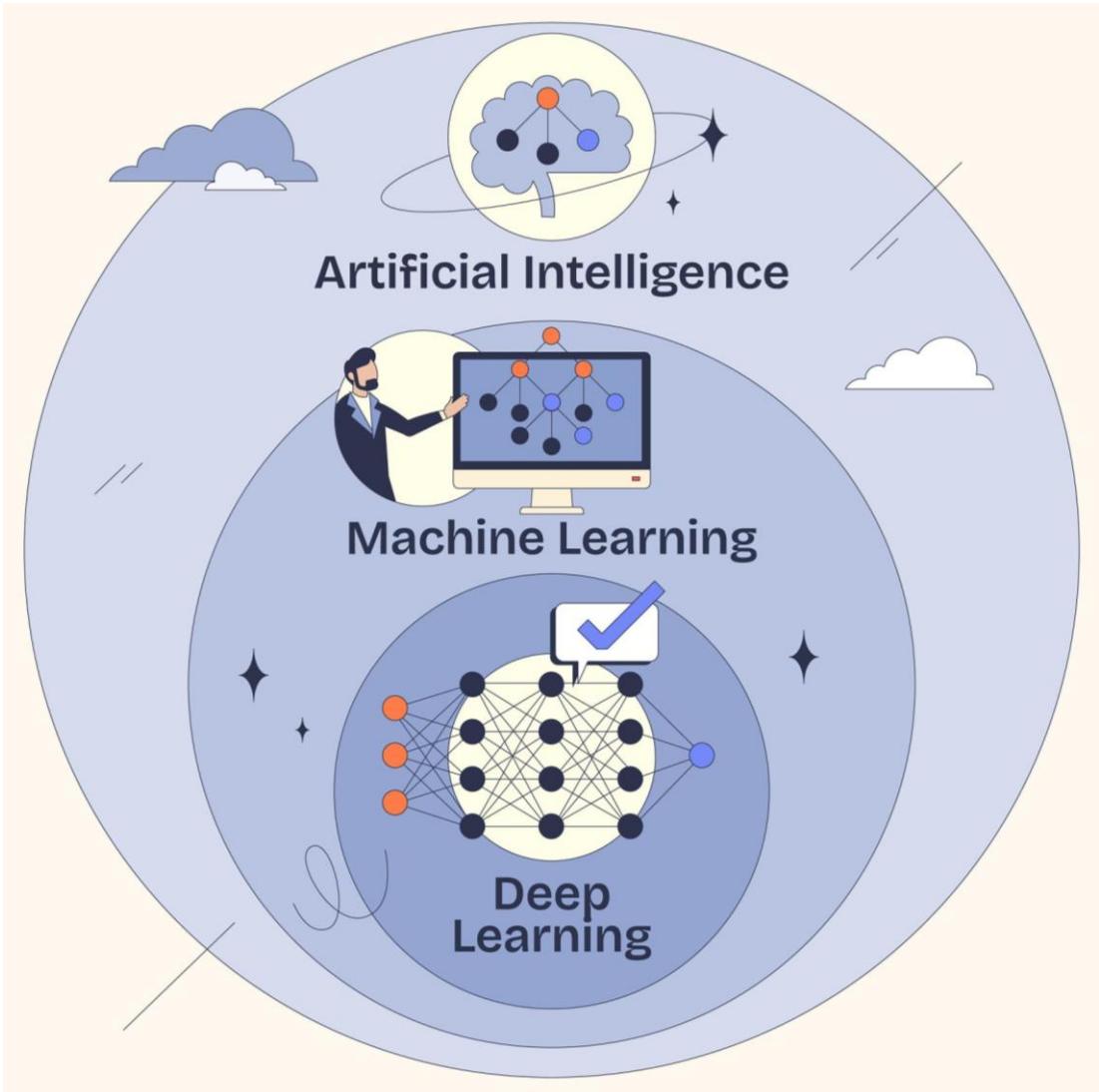


Tag 2 10:15-11:30

Verständliche Erklärung: Was ist Deep Learning und wie funktioniert es?

- Neuronale Netze
- TensorFlow Grundlagen

Künstliche Intelligenz (KI)

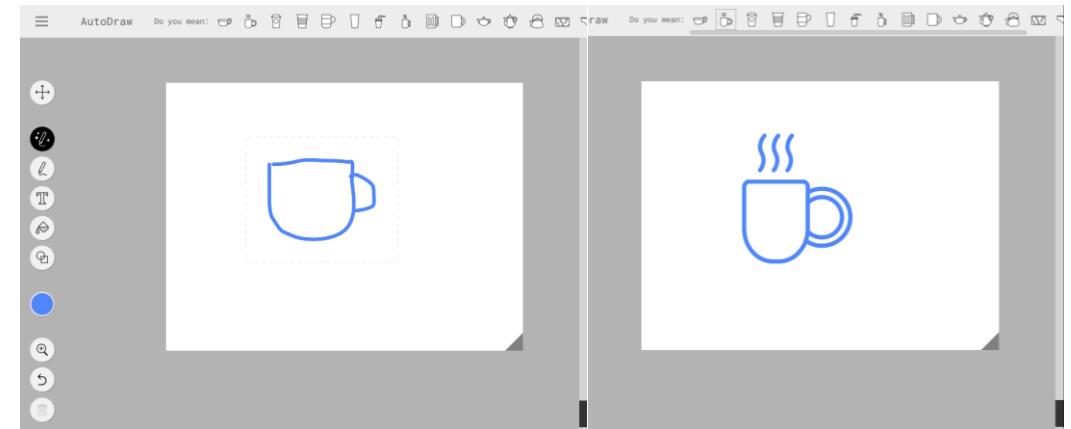


Deep-Learning-Anwendungen



Virtual Assistant

Chatbots



Google Übersetzer

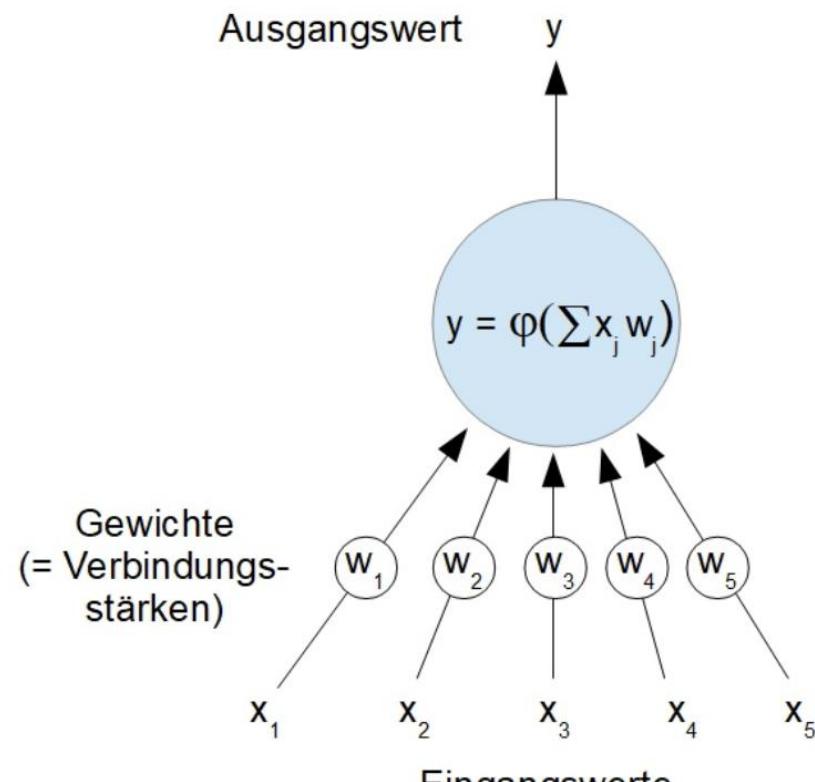
Text **Bilder** **Dokumente** **Websites**

Sprache erkennen Deutsch Englisch Französisch



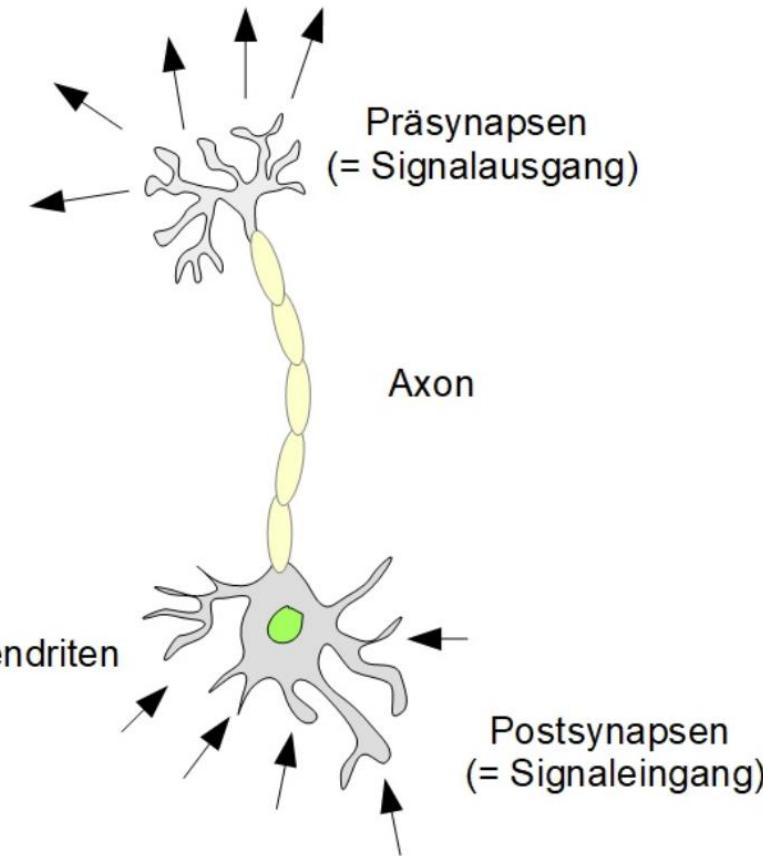
Neuron

Künstliches Neuron



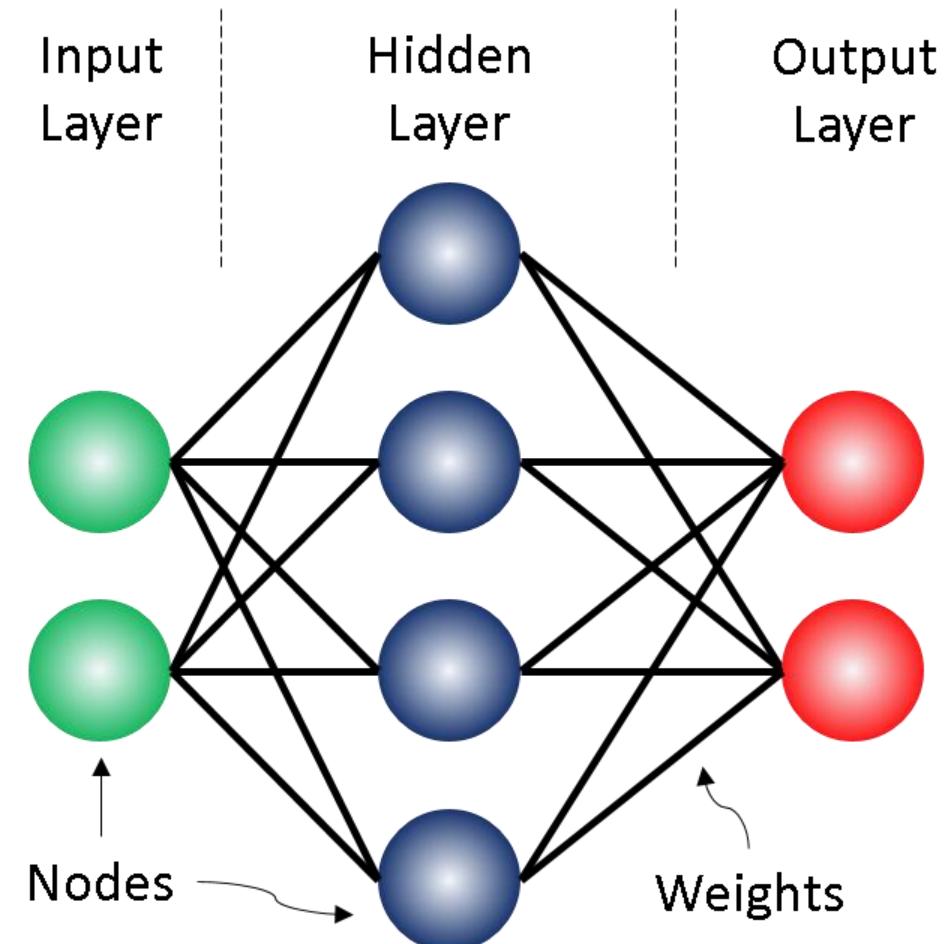
(Quelle: Helmut Linde)

Biologisches Neuron



Vom einzelnen Neuron zum neuronalen Netz

- **Was ist ein neuronales Netz?**
- **Schichten:**
 - Input Layer
 - Hidden Layer
 - Output Layer
- Warum „Deep“ = viele Schichten
- Intuition: Schichtweise Merkmalsextraktion



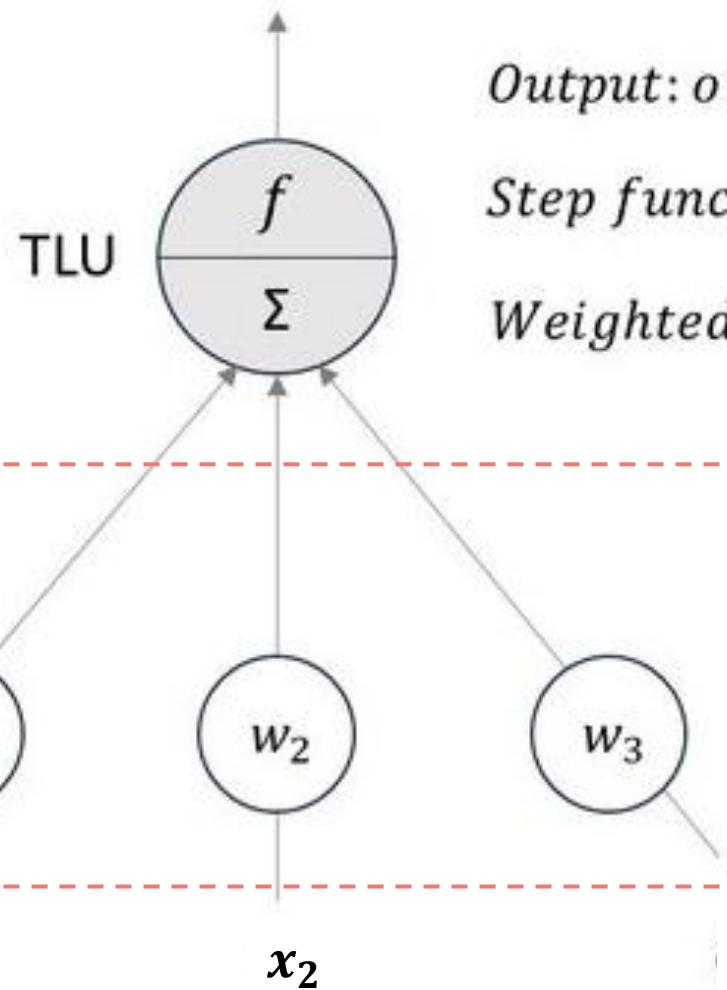
(Bildquelle)

Soll ein Alarm ausgelöst werden?

(TLU) Perzeptron

TLU Perzepron

1 Schicht
harte Schwelle (0/1)
nur linear trennbar
gut für Logik



Output: $o = f(z)$

Step function: $f(x)$

Weighted sum: $z = \sum_{k=1}^n x_k * w_k$

Schwellenfunktion (Threshold / Step Function)

Das ist das Herz der TLU:

$$y = \begin{cases} 1 & \text{wenn } z \geq 0 \\ 0 & \text{wenn } z < 0 \end{cases}$$

→ Nur Ja / Nein, keine Wahrscheinlichkeit.

Jede Eingabe wird **gewichtet**:

$$z = w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b$$

w_i : Wichtigkeit der Eingabe

b : Bias = Verschiebung der Entscheidungsschwelle

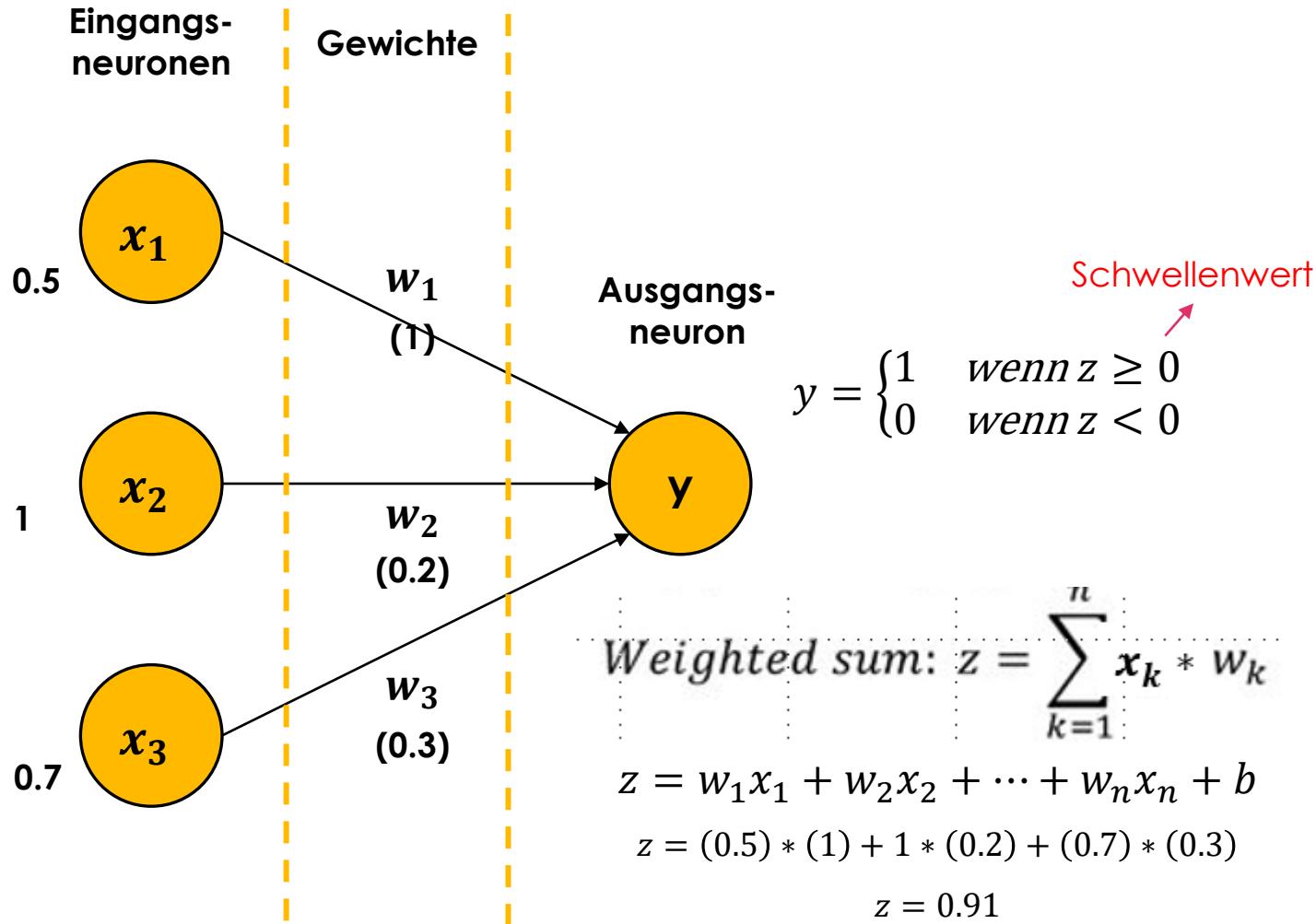
Weights: $w_1 .. w_n$

Inputs: $x_1 .. x_n$

Beispiele:

- x_1 : Bewegung erkannt? (0/1)
- x_2 : Fenster geschlossen? (0/1)
- x_3 : Nacht? (0/1)

(TLU) Perzeptron - Entscheidung



Frage:

Wenn der Schwellenwert = 1,
was ist y ?

Beispiel

Soll ein Alarm ausgelöst werden?

Eingaben

$x_1 = 1 \rightarrow$ Bewegung erkannt
 $x_2 = 0 \rightarrow$ Fenster geschlossen
 $x_3 = 1 \rightarrow$ Nacht

Gewichte

$w_1 = 2$ (Bewegung ist sehr wichtig)
 $w_2 = 1$
 $w_3 = 1.5$
 Bias: $b = -2$

Berechnung

$$z = 2 \cdot 1 + 1 \cdot 0 + 1.5 \cdot 1 - 2 = 1.5$$

→ $z \geq 0 \Rightarrow y = 1$

⚠️ Alarm an

(TLU) Perzeptron – Übungen & Diskussion

 **Situation:** Stell dir vor, du möchtest entscheiden, ob du heute ins Fitnessstudio gehst oder zu Hause bleibst?

1. Inputs (Faktoren)

Das Modell bekommt Informationen: (0 oder 1)

- x_1 : Habe ich genug Energie?
- x_2 : Habe ich Zeit?
- x_3 : Wie stark ist meine Motivation?

2. Gewichtung (Wichtigkeit)

Nicht jeder Faktor ist gleich wichtig.

- Energie → sehr wichtig (Gewicht 5)
- Zeit → mittel wichtig (Gewicht 3)
- Motivation → wichtig (Gewicht 4)

3. Schwellenwert (Threshold)

Du sagst dir:

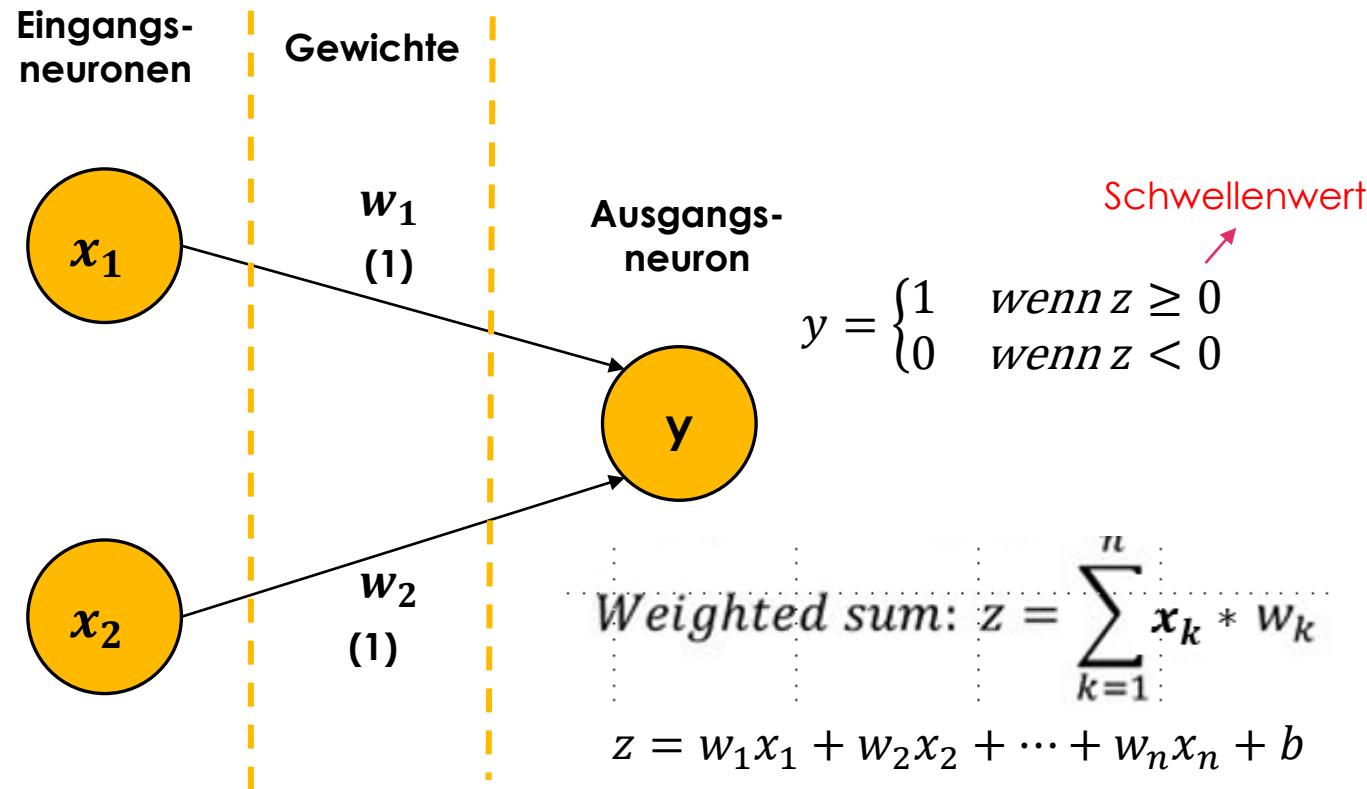
„Wenn meine Gesamtmotivation über 8 liegt, gehe ich.“

4. Entscheidung

Modell rechnet: **Faktoren × Gewichte → Summe**

Wenn **Summe ≥ Schwelle** → JA; Sonst → NEIN

(TLU) Perzeptron – Logik AND, OR, XOR



x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

1) AND ($b=-1.5$)

Nur wenn **beide** Eingänge aktiv sind, wird Output = 1 gefeuert

→ Große Schwellenwert

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

2. OR ($b=-0.5$)

Wenn **mindestens einer** der Eingänge aktiv ist, wird Output = 1 gefeuert

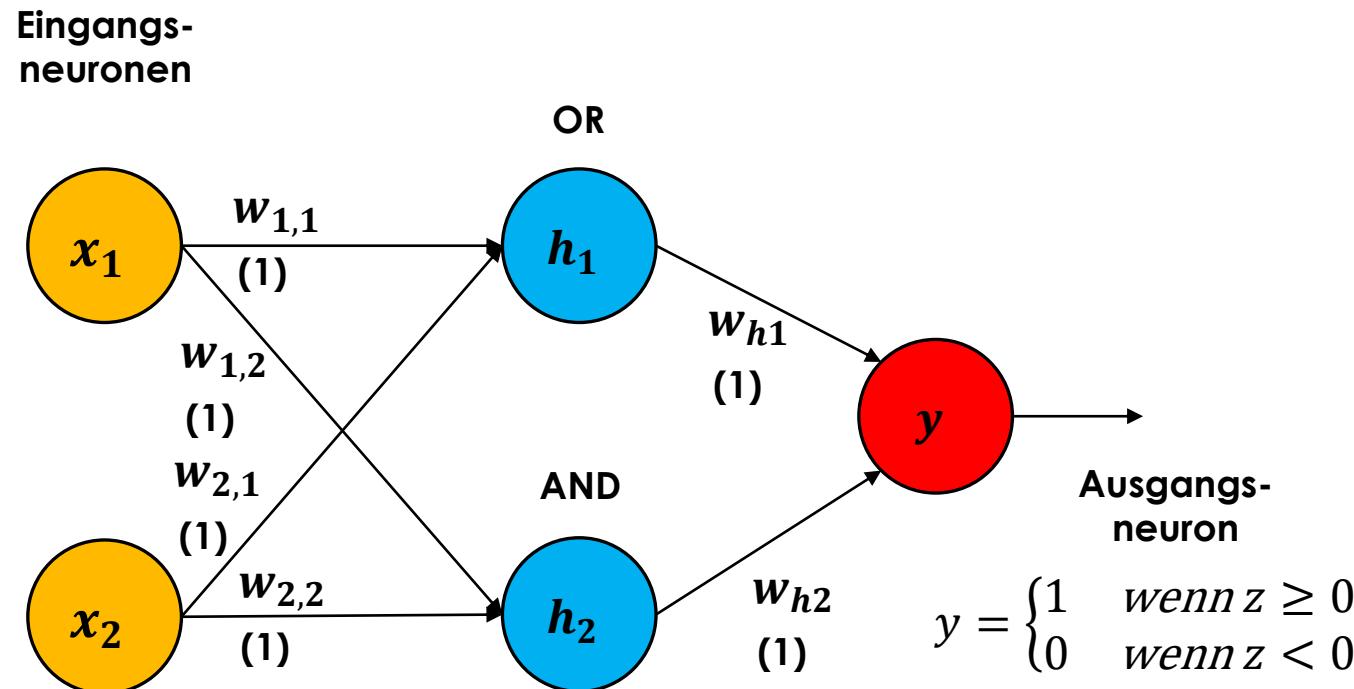
→ Kleiner Schwellenwert

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

2. XOR

Wenn die beiden Eingaben unterschiedlich sind, wird Output = 1 gefeuert

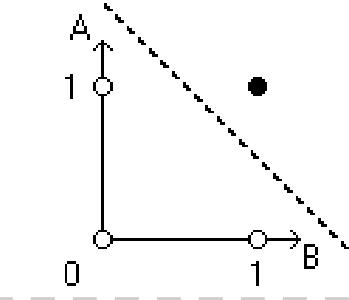
Warum brauchen wir ein Hiddenschicht?



DEEP LEARNING MIT PYTHON UND TENSORFLOW

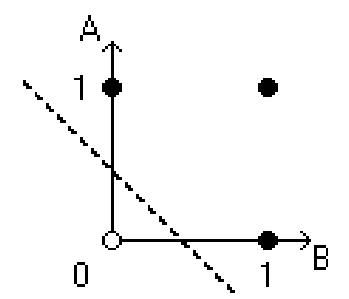
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

1) AND ($b=-1.5$)



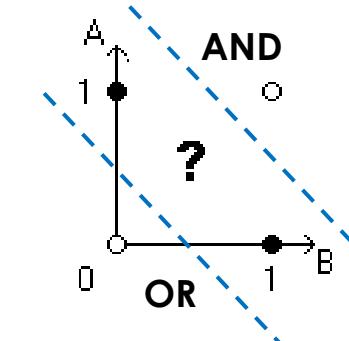
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

2. OR ($b=-0.5$)



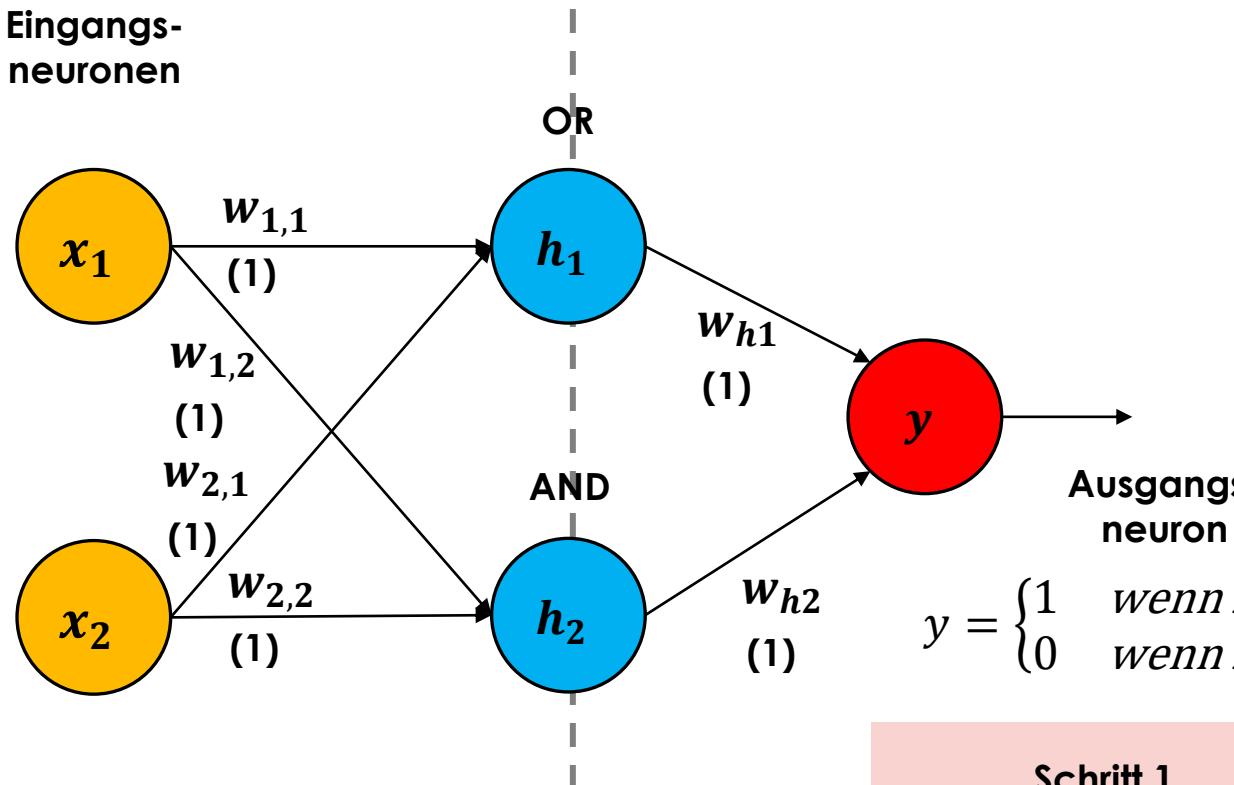
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

2. XOR



Aktivierungsfunktion

Eingangs-neuronen



$$y = \begin{cases} 1 & \text{wenn } z \geq 0 \\ 0 & \text{wenn } z < 0 \end{cases}$$

Schritt 1

$$z = w_1 x_1 + w_2 x_2 + b$$

Schritt 2

$$f(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

Was passiert ohne Aktivierungsfunktion?

Der Output ist einfach: $y = z$

Wenn wir mehrere Schichten hintereinander haben:

👉 Also immer noch eine lineare Funktion.

Was macht die Aktivierungsfunktion?

- entscheidet, ob ein Neuron „aktiv“ wird
- führt eine **Nichtlinearität** ein
- ermöglicht komplexe Entscheidungsgrenzen

Beispiel:

Step-Funktion:

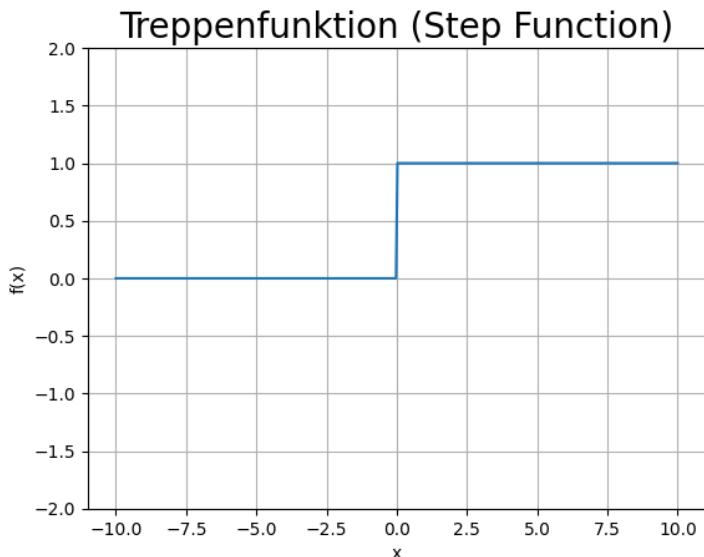
$$f(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$

Oder moderner:

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

Diese Funktionen sind **nicht linear**.

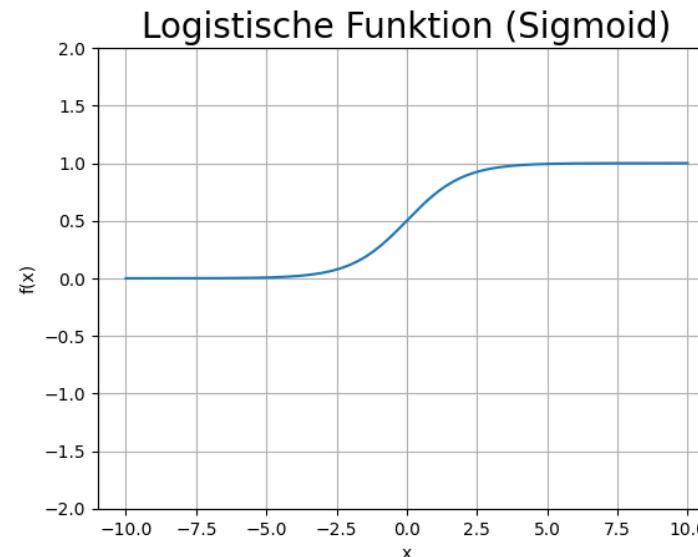
Aktivierungsfunktion



$$f(x) = \begin{cases} 0 & \text{für } x < 0 \\ 1 & \text{für } x \geq 0 \end{cases}$$

Die Schrittfunktion gibt nur zwei mögliche Werte aus:

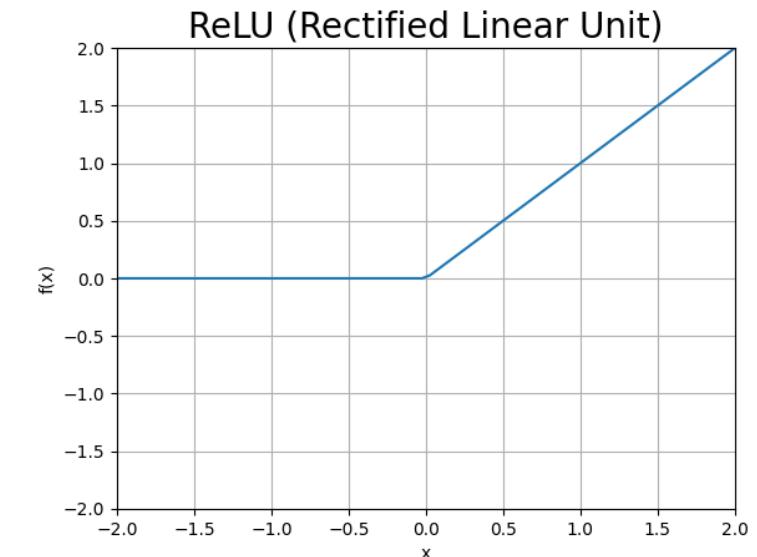
- 0 für negative Eingaben
- 1 für positive oder null Eingaben



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

Die Funktion transformiert jede reelle Zahl in einen Wert zwischen 0 und 1.

- Sehr negative Werte \rightarrow nahe 0
- Sehr positive Werte \rightarrow nahe 1
- Bei $x = 0 \rightarrow 0,5$



$$f(x) = \begin{cases} 0 & \text{für } x < 0 \\ x & \text{für } x \geq 0 \end{cases}$$

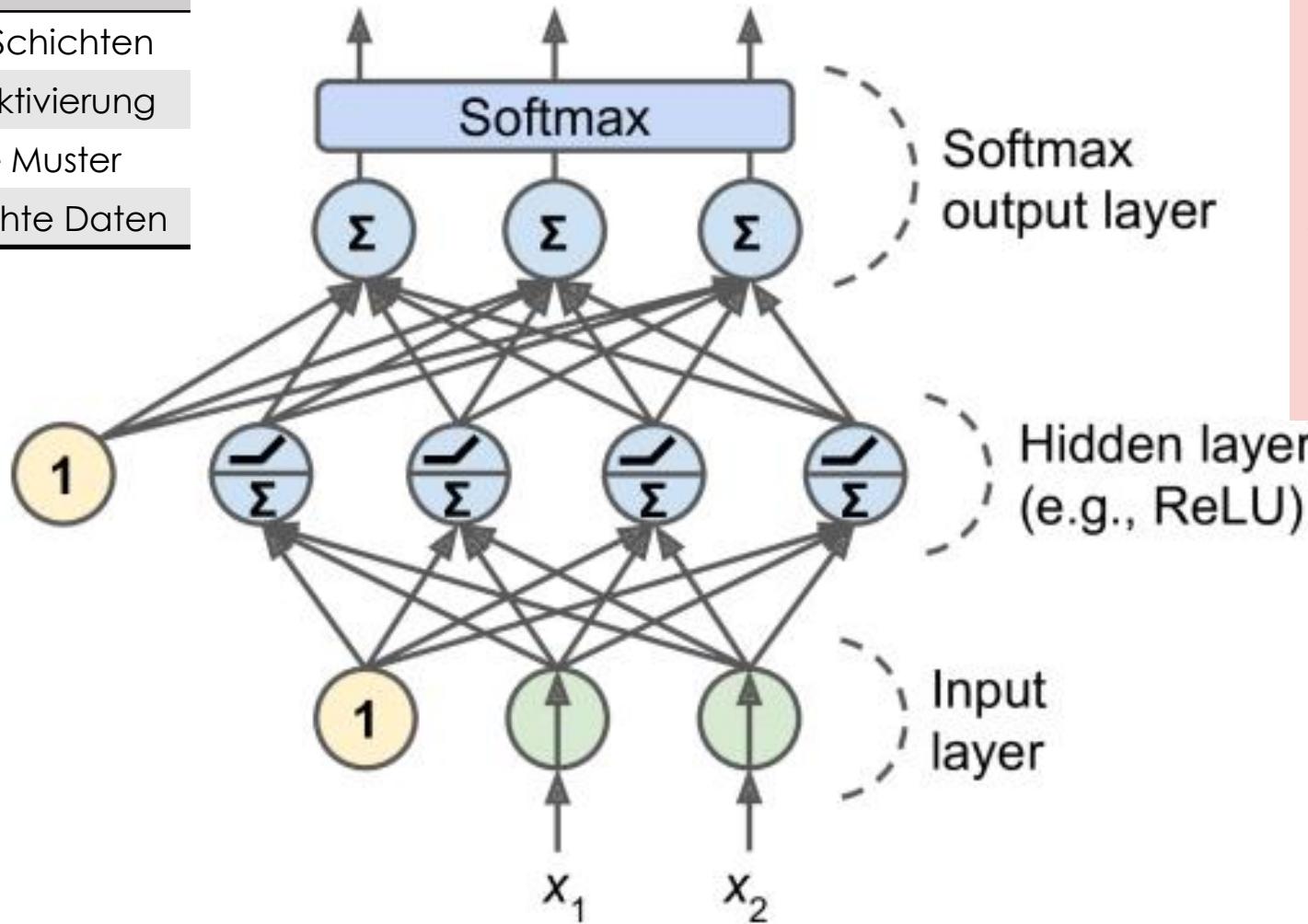
ReLU schneidet alle negativen Werte ab.

- Negative Eingaben $\rightarrow 0$
- Positive Eingaben \rightarrow unverändert
- Halb linear, halb abgeschnitten.

(MLP) Perzeptron

MLP

mehrere Schichten
weiche Aktivierung
komplexe Muster
gut für echte Daten



② Softmax – Aktivierungsfunktion in der Output Layer

Schritt 3: Output-Scores (Logits)

Für jede Klasse einen Score:

$z_{out} = [2.1, -1.3, 0.2, 3.8, -0.5, -2.0, 1.1, 0.3, 0.9, 2.5]$

⚠ Das sind **keine Wahrscheinlichkeiten**:

- können negativ sein
- Summe ist beliebig

Schritt 4: Softmax anwenden

$[0.08, 0.01, 0.03, 0.42, 0.02, 0.01, 0.07, 0.03, 0.06, 0.27]$

✓ Alle Werte zwischen 0 und 1

✓ **Summe = 1**

① ReLU – Aktivierungsfunktion in der Hidden Layer

Schritt 1: Lineare Berechnung

$z = [-2.3, 0.7, 4.1, -0.5]$

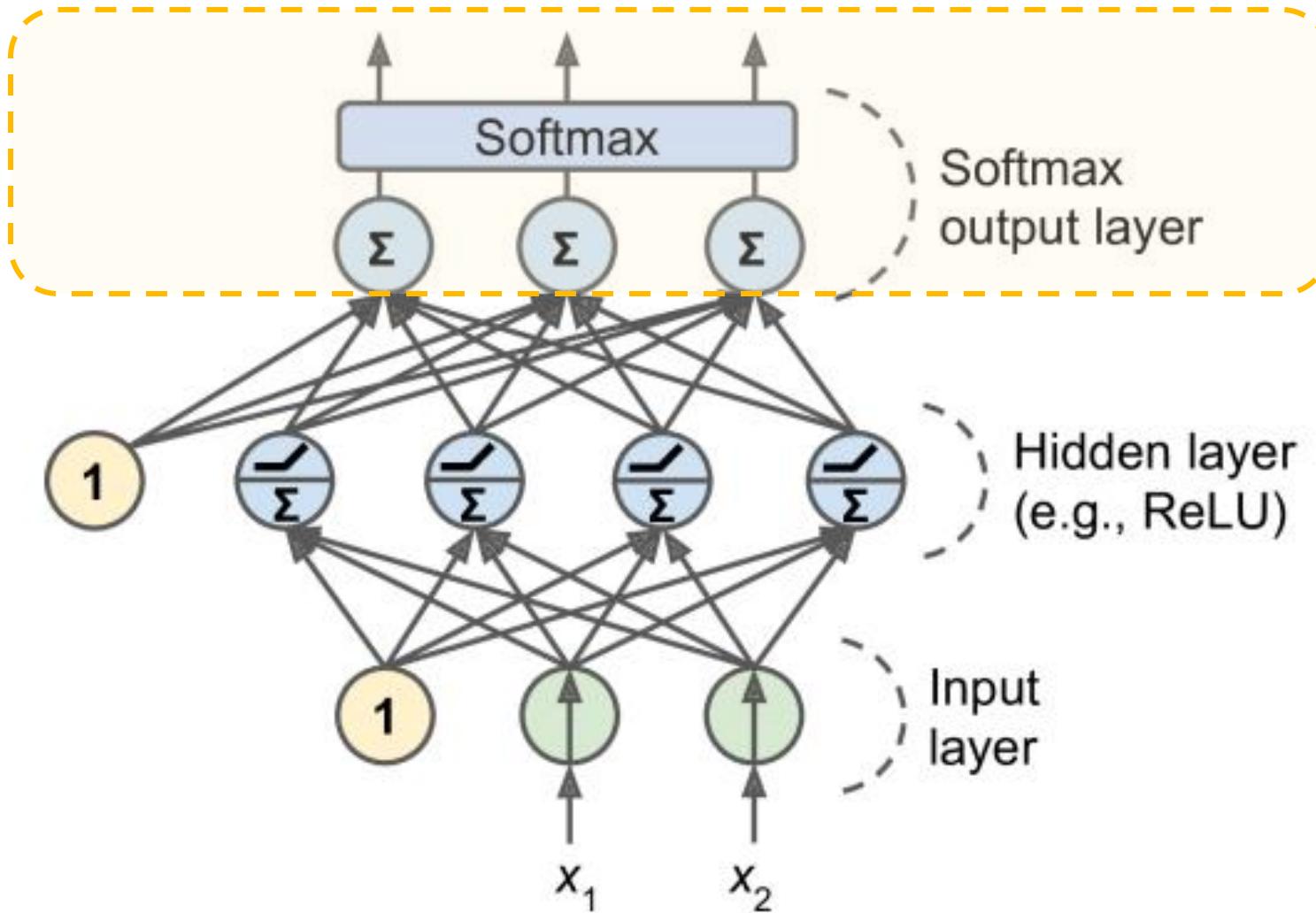
Schritt 2: ReLU anwenden

$h = [0, 0.7, 4.1, 0]$

Was macht ReLU?

- **Negativen Werte: 0**
- Positive Werte bleiben erhalten
- **Keine Wahrscheinlichkeiten**
- **Summe ist egal**

(MLP) Perzeptron



Softmax

Was passiert, wenn wir **mehr als zwei Klassen** haben?

Sigmoid: zwei Klassen

Softmax: drei Klassen oder mehr

Die letzte Schicht gibt 10 Zahlen aus:

[2.1, 0.3, 5.4, -1.2, ...]

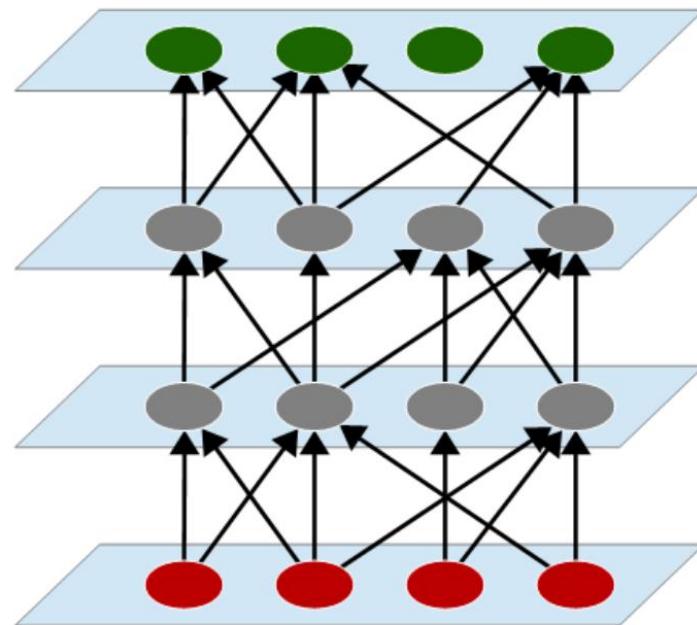
Softmax macht **Wahrscheinlichkeiten** daraus:

[0.01, 0.002, 0.85, 0.00001, ...]

- Nur positive Werte
- Alle Werte zwischen 0 und 1
- Die Summe ist genau 1

Deep Learning

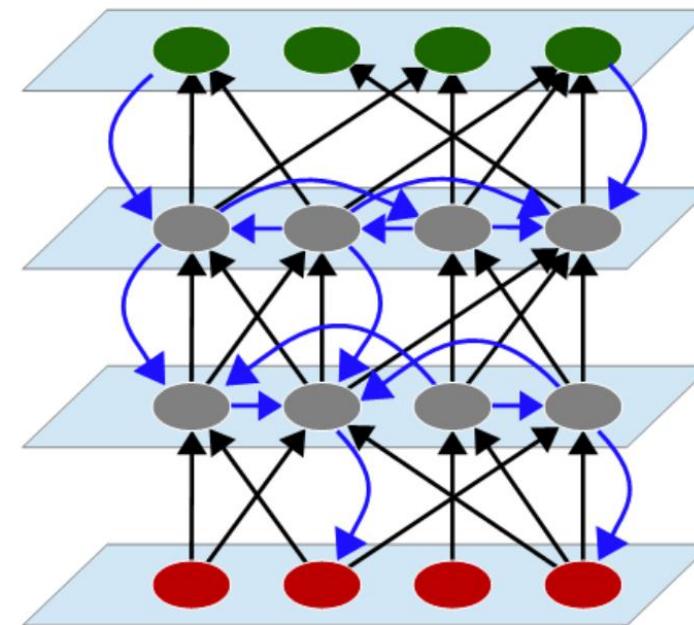
Deep Learning



Vorwärtsgerichtete Architektur

(Quelle: Helmut Linde)

Gehirn



Rekurrente Architektur

Grundlagen TensorFlow

Get started with TensorFlow

TensorFlow makes it easy to create ML models that can run in any environment. Learn how to use the intuitive APIs through interactive code samples.

[View tutorials >](#)

Website: <https://www.tensorflow.org/>

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

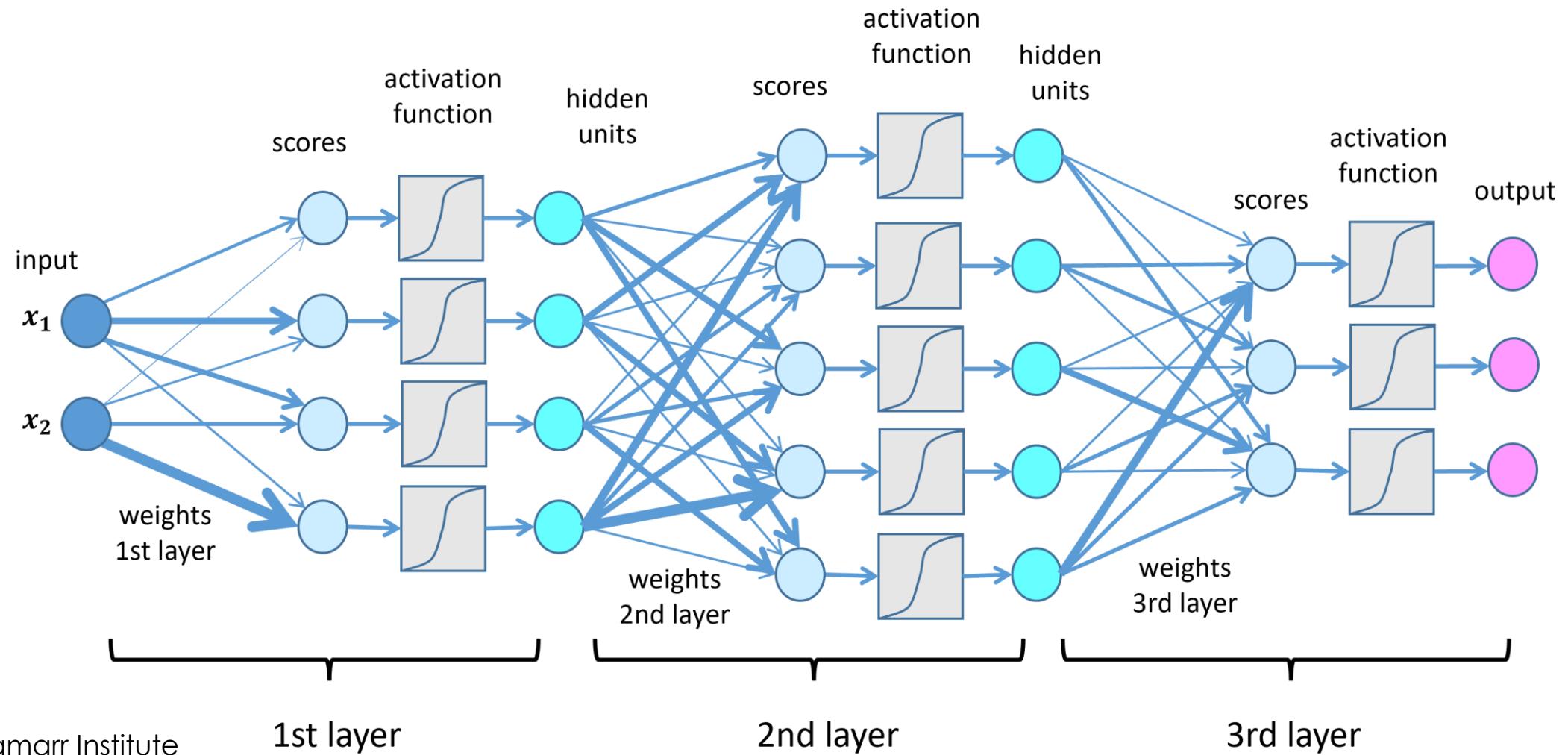
Run quickstart 



Tag 2 11:30-12:15

Einführung in Deep Learning

Deep Learning – Vom Datensatz zum Modell



Daten

Label

The figure displays a 10x10 grid of 100 grayscale images representing different clothing items. Each row corresponds to a specific item category, labeled on the left side:

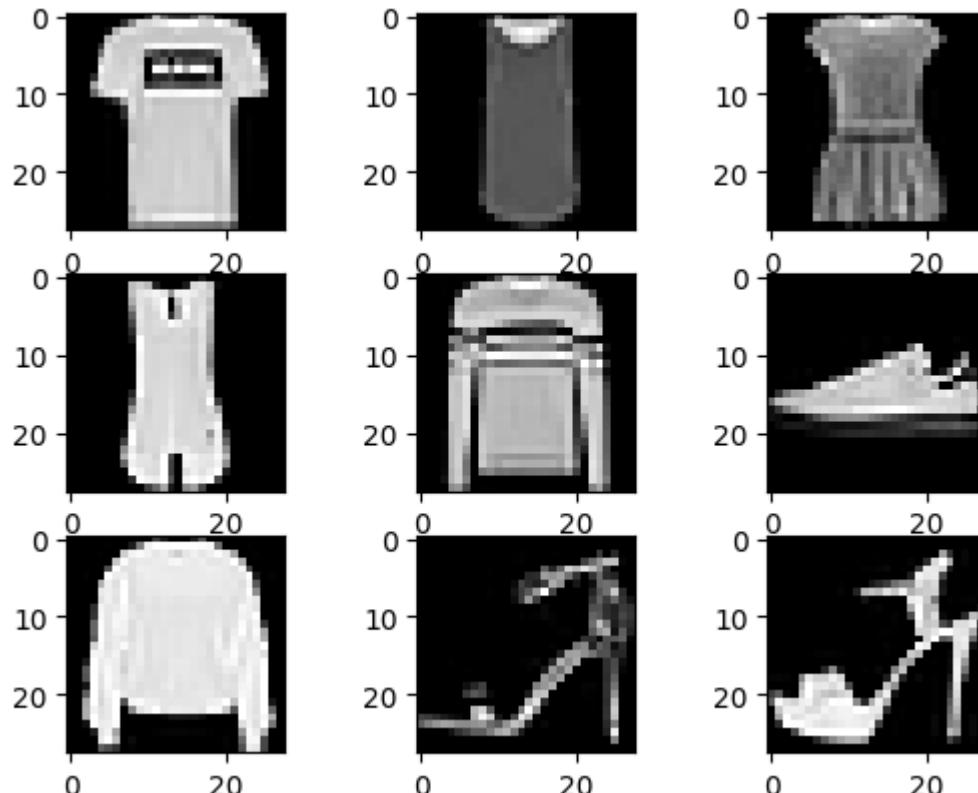
- Row 0: T_shirt
- Row 1: Trouser
- Row 2: Pullover
- Row 3: Dress
- Row 4: Coat
- Row 5: Sandal
- Row 6: Shirt
- Row 7: Sneaker
- Row 8: Bag
- Row 9: Ankle boot

A vertical dashed orange line runs through the grid, separating the first 7 columns from the last 3 columns. Orange text labels indicate the split:

- An orange box with the text "Trainingdaten" covers the area from approximately column 4 to column 7.
- An orange box with the text "Testdaten" covers the area from approximately column 8 to column 10.

Daten

Fashion-MNIST ist ein Datensatz mit Bildern von Kleidungsstücken, der häufig verwendet wird, um Deep Learning und neuronale Netze zu erklären und zu üben.



Welche Daten enthält Fashion-MNIST?

Fashion-MNIST besteht aus vielen kleinen Bildern, und jedes Bild zeigt ein einzelnes Kleidungsstück.

Umfang des Datensatzes

60.000 Trainingsbilder (zum Lernen) + 10.000 Testbilder (zum Testen)



Die Bilder

Größe: 28 × 28 Pixel

Graustufen: Jeder Pixel hat einen Wert von 0 bis 255

0 = schwarz ; 255 = weiß

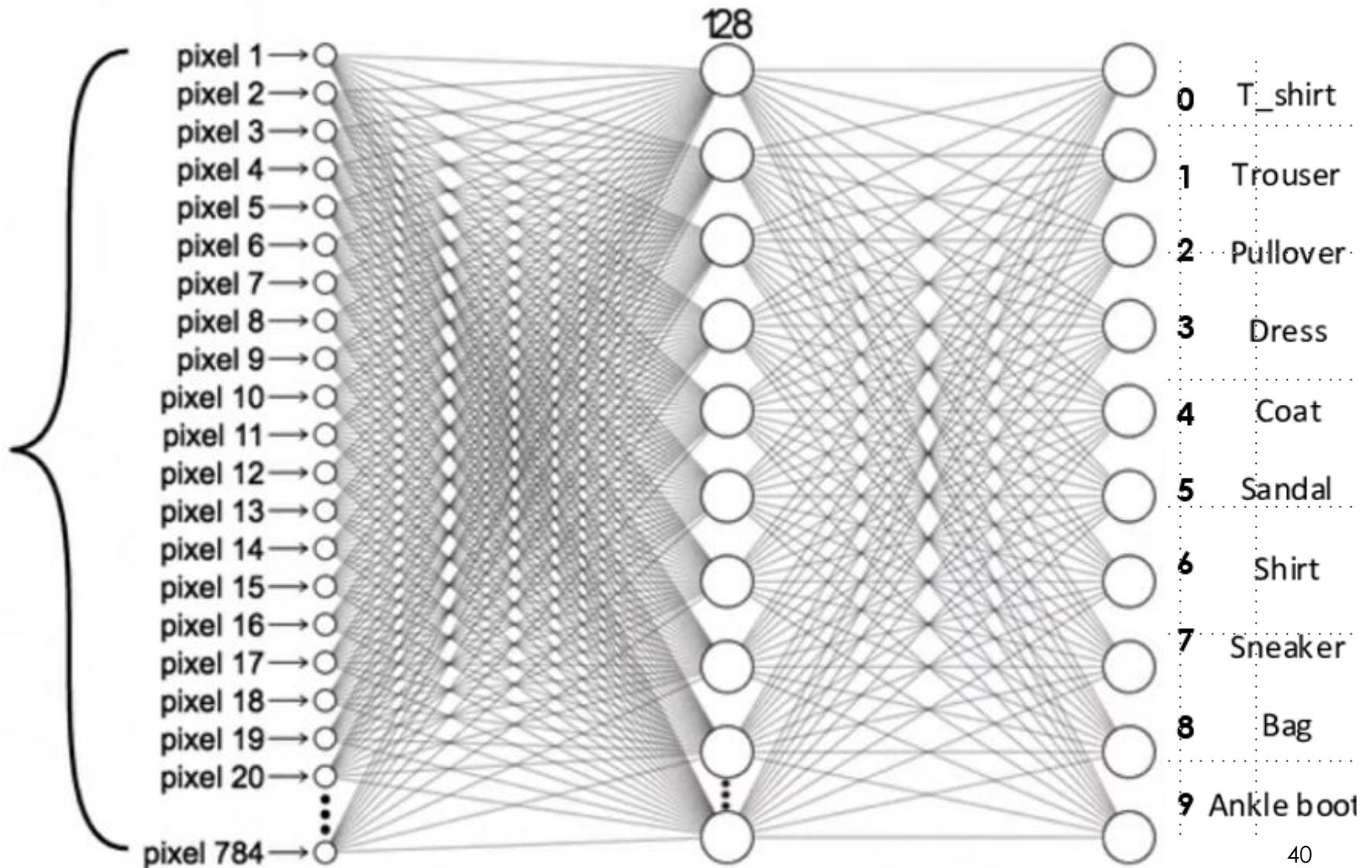
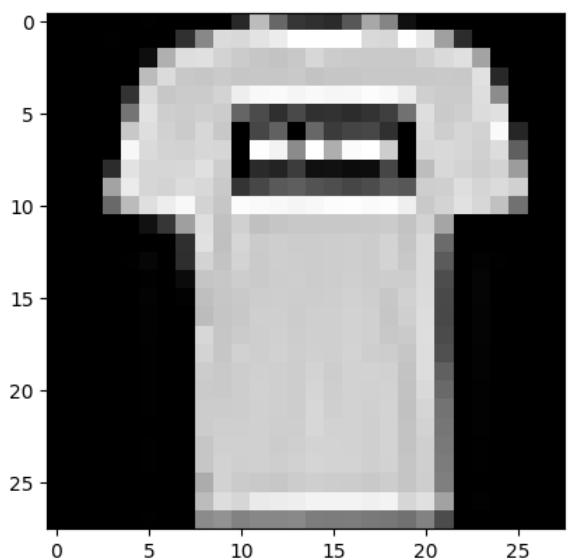
👉 Das 28×28 -Bild wird in eine lange Zahlenliste mit 784 Werten umgewandelt.

🏷️ Die Klassen (Labels)

Das neuronale Netz soll:

👉 lernen, das richtige Kleidungsstück auf einem Bild zu erkennen

Daten



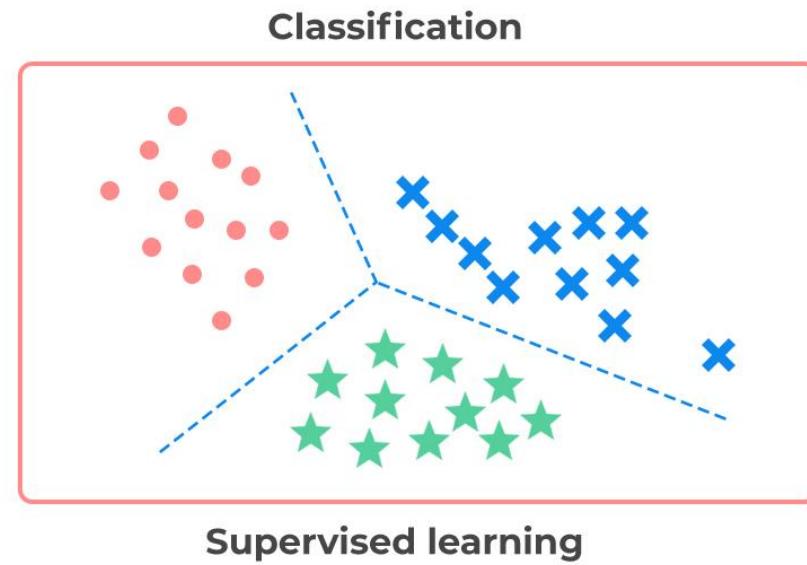
Lernmethoden und wichtige Modelle (Überblick)

1. Supervised Learning (Überwachtes Lernen)

- **Trainingsdaten haben Labels**
- Modell lernt Eingabe → Zielwert
- Fehler wird berechnet (Loss Function)

Beispiele:

- Bildklassifikation
- Spam-Erkennung

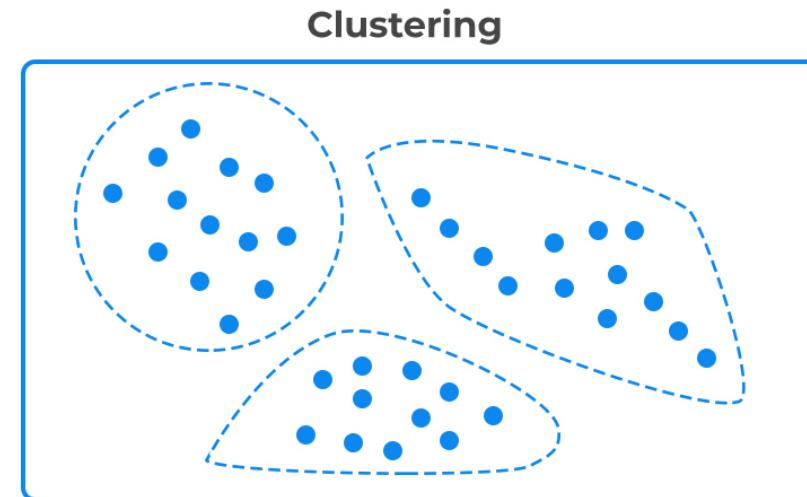


2. Unsupervised Learning (Unüberwachtes Lernen)

- **Keine Labels**
- Modell sucht selbst Strukturen
- Muster- oder Clustererkennung

Beispiele:

- Kundensegmentierung



(Bildquelle)

Wichtige Modelle - FNN

Feedforward Neural Network (FNN)

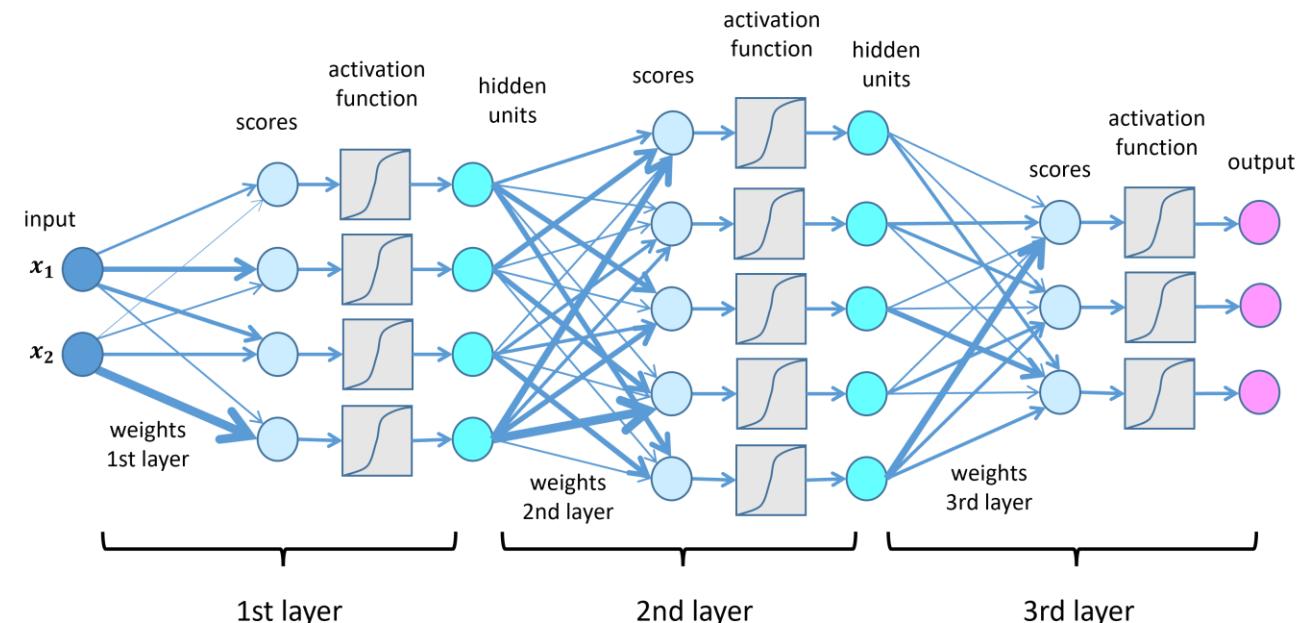
- Daten fließen von links nach rechts – vom **Input** über verdeckte Schichten zum **Output**.

Was passiert?

- Jede Schicht verarbeitet Informationen
- Gewichte werden angepasst
- Am Ende entsteht eine Vorhersage

Typische Anwendungen

- Klassifikation (z. B. Spam oder kein Spam)
- Regression (z. B. Preisvorhersaged)



© Lamarr Institute

Wichtige Modelle - CNN

Convolutional Neural Network (CNN)

- Das Modell erkennt **Muster in Bildern**.

Was macht es besonders?

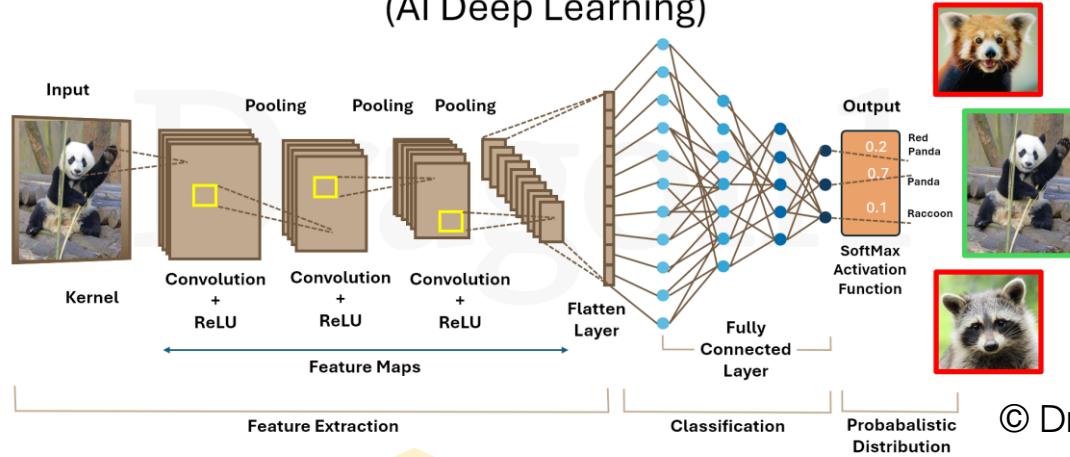
- Arbeitet mit Filtern
- Erkennt Kanten, Formen, Texturen
- Baut schrittweise komplexe Strukturen auf

Typische Anwendungen

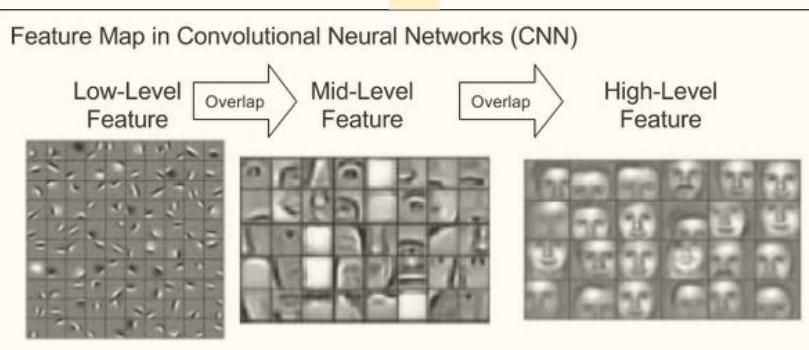
- Gesichtserkennung
- Medizinische Bildanalyse
- Autonomes Fahren

Convolutional Neural Networks

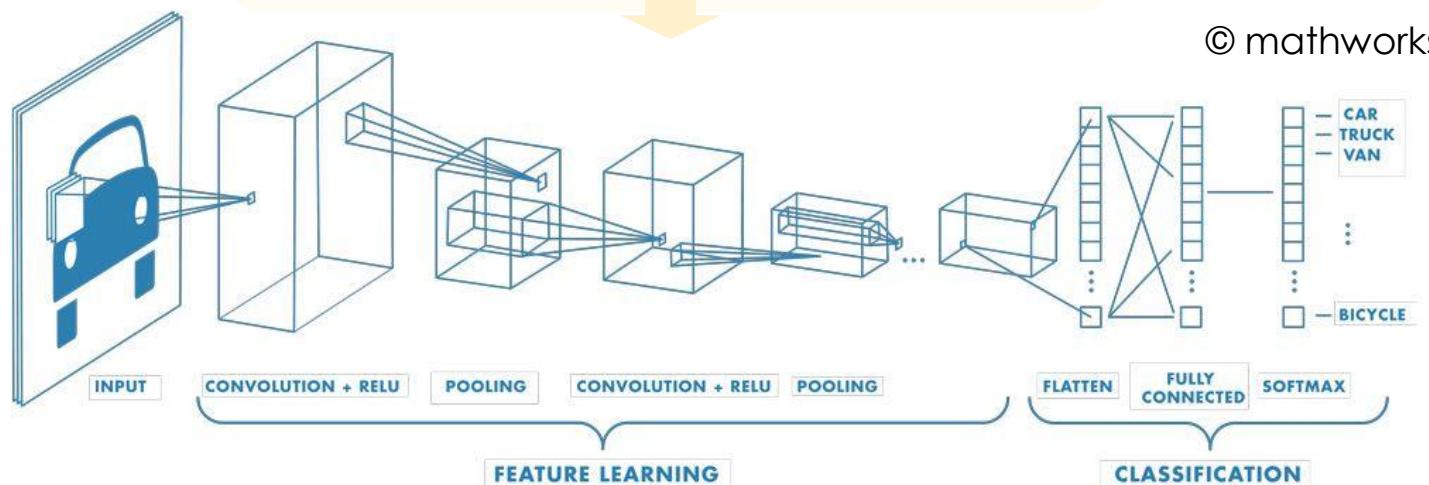
(AI Deep Learning)



© Dragon1



© mathworks



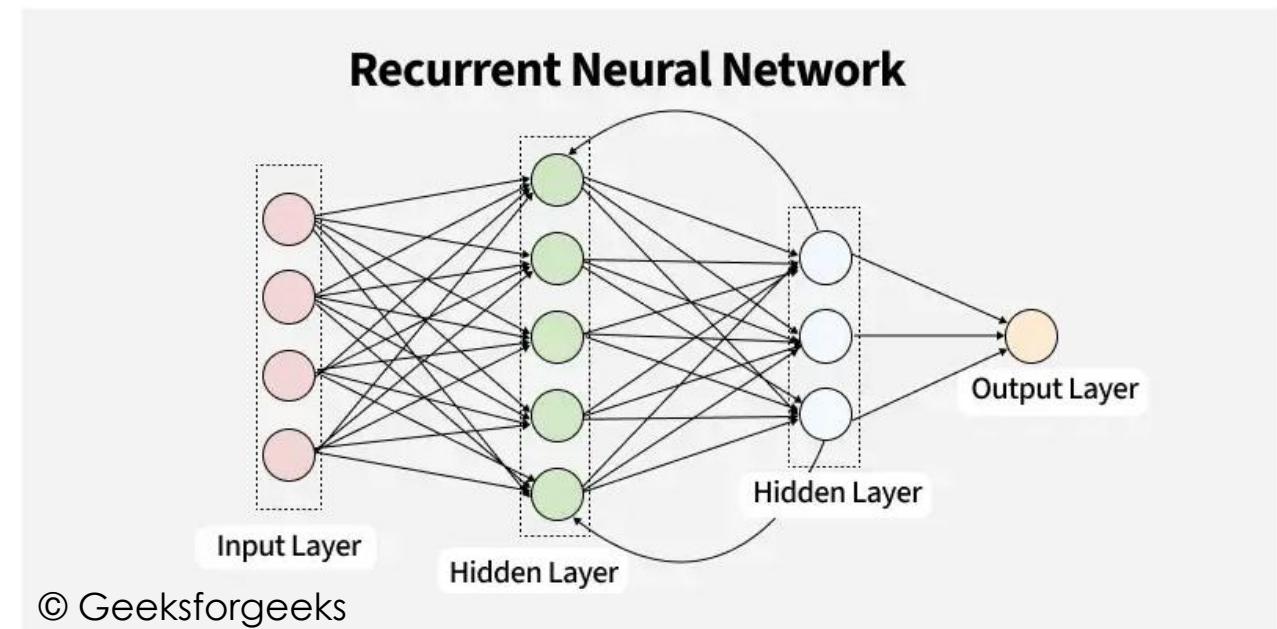
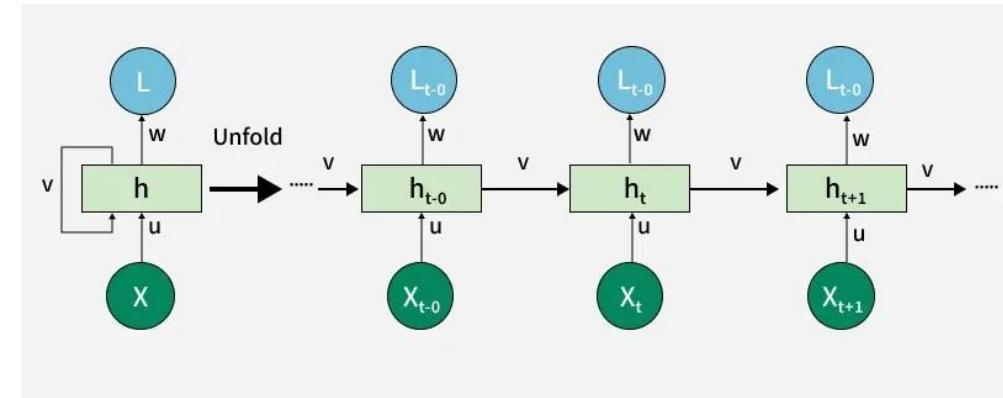
Wichtige Modelle - RNN

Recurrent Neural Network (RNN)

- Das Modell verarbeitet **Informationen in Reihenfolge**.
- Was ist besonders?**
- Es „merkt sich“ vorherige Eingaben
- Frühere Informationen beeinflussen spätere Entscheidungen

Typische Anwendungen

- Sprache
- Zeitreihen (z. B. Aktienkurse)
- Textanalyse



Lernmethoden und wichtige Modelle (Überblick)

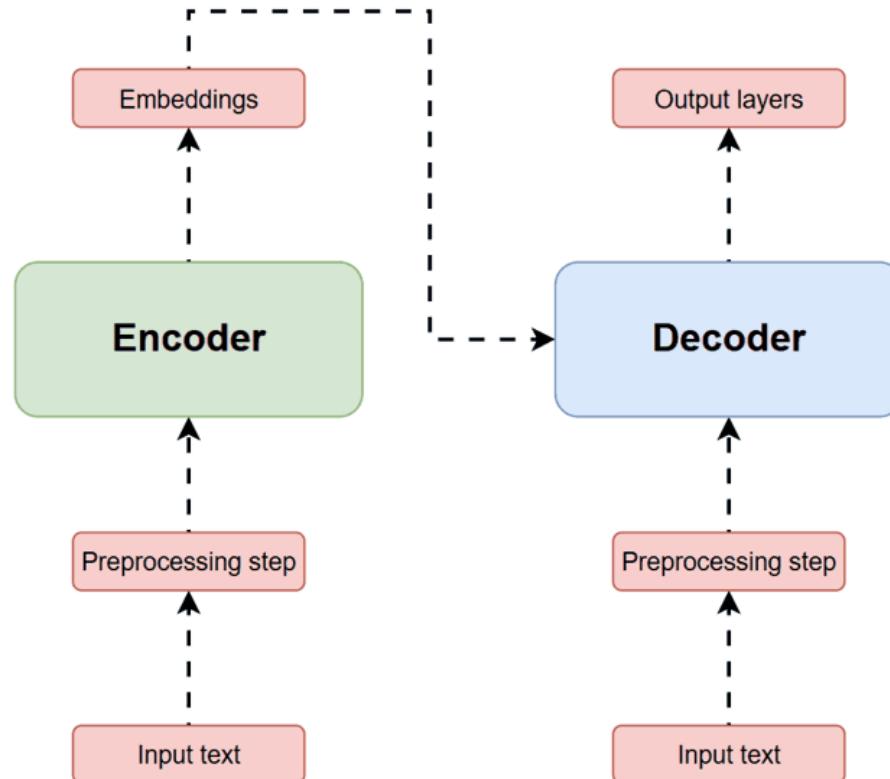
Transformer – Moderne Sprachmodelle

- Alle Wörter werden gleichzeitig betrachtet.
- ★ Was macht ihn so stark?**
- Attention-Mechanismus
 - Erkennt Zusammenhänge über große Distanzen
 - Sehr effizient bei großen Datenmengen

🎯 Typische Anwendungen

- Chatbots
- Übersetzung
- Textzusammenfassung
- Large Language Models

© compute.hivenet



Lernen aus Daten: Wie trainiert ein Modell?

Wie wird aus Daten ein „intelligentes“ Modell?

Trainingsdaten vs. Testdaten

Trainingsdaten

- Werden zum Lernen verwendet
- Modell passt seine Gewichte an
- Kennt die richtigen Antworten

Testdaten

- Werden nicht zum Lernen benutzt
- Prüfen die Generalisierungsfähigkeit
- Simulieren neue, unbekannte Daten

Die Grundidee des Lernens Lernprinzip

1. Modell macht eine Vorhersage
2. Vorhersage wird mit der Wahrheit verglichen
3. Modell bekommt Feedback
4. Modell verbessert sich

Datenqualität ist wichtiger als Modellgröße

Ein großes Modell hilft nicht, wenn:

- Daten falsch sind
- Daten unvollständig sind
- Klassen unausgeglichen sind

Modellbewertung – Wie gut ist ein Modell?

Wie wissen wir, ob ein Modell gut ist?

Ein Modell ist nicht gut, nur weil es trainiert wurde.

Wir müssen es bewerten.

Ein Modell wird:

- Mit Trainingsdaten gelernt
- Mit Testdaten bewertet

Überanpassung (Overfitting)

Ein Modell kann:

- Auf Trainingsdaten sehr gut sein
- Aber auf Testdaten schlecht

Das nennt man Overfitting.

Wichtige Bewertungsmetriken (Confusion Metriken für Klassifikation)

- **Accuracy:** Anteil korrekt vorhergesagter Beispiele.
Stell dir vor: Wir erkennen Spam-Mails.
- **Precision:** Wie viele der als Spam erkannten Mails sind wirklich Spam?
- **Recall:** Wie viele der tatsächlichen Spam-Mails wurden erkannt?
- **F1-Score:** Kombination aus Precision und Recall.

Modellgröße ≠ Qualität

 Ein größeres Modell ist nicht automatisch besser.

Entscheidend sind:

- Datenqualität
- Ausgewogene Klassen
- Saubere Labels

12:30-13:10 ☕ Mittagspause



Tag 2 13:10-15:00

Praktische Übung: Einfaches Modell

- Modell bauen & trainieren mit TensorFlow

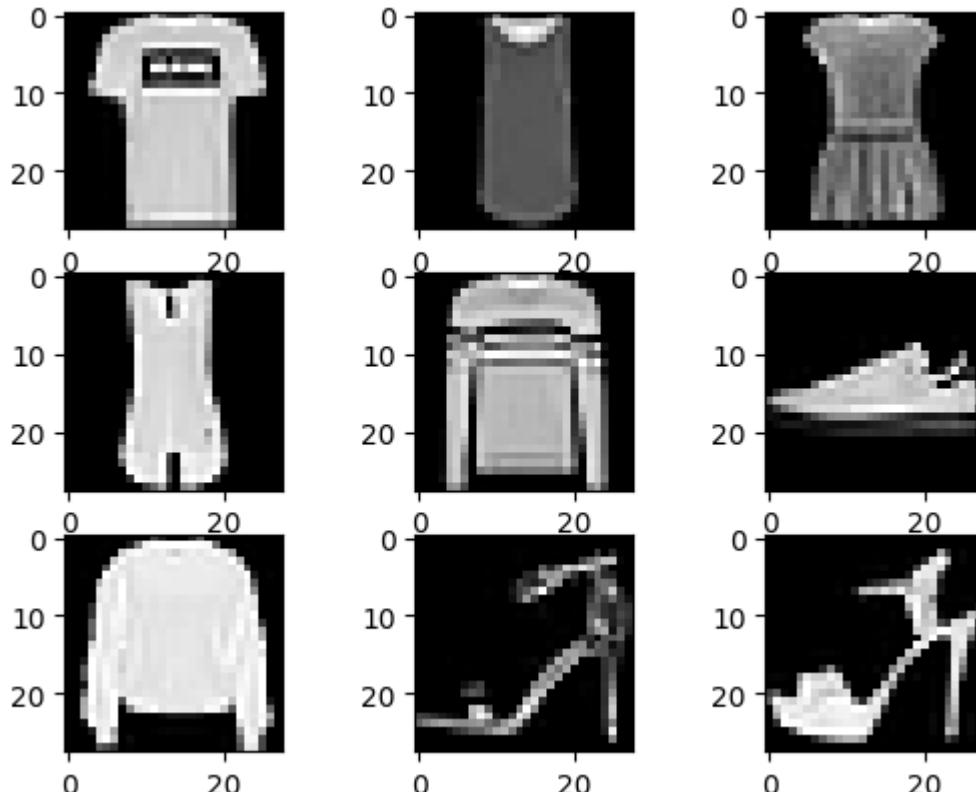
Daten: Fashion MNIST

Label

Label	T_shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
0 T_shirt										
1 Trouser										
2 Pullover										
3 Dress										
4 Coat										
5 Sandal										
6 Shirt										
7 Sneaker										
8 Bag										
9 Ankle boot										

Daten: Fashion MNIST

Fashion-MNIST ist ein Datensatz mit Bildern von Kleidungsstücken, der häufig verwendet wird, um Deep Learning und neuronale Netze zu erklären und zu üben.



Welche Daten enthält Fashion-MNIST?

Fashion-MNIST besteht aus vielen kleinen Bildern, und jedes Bild zeigt ein einzelnes Kleidungsstück.

Umfang des Datensatzes

60.000 Trainingsbilder (zum Lernen) + 10.000 Testbilder (zum Testen)

Die Bilder

Größe: 28×28 Pixel

Graustufen: Jeder Pixel hat einen Wert von 0 bis 255

0 = schwarz ; 255 = weiß

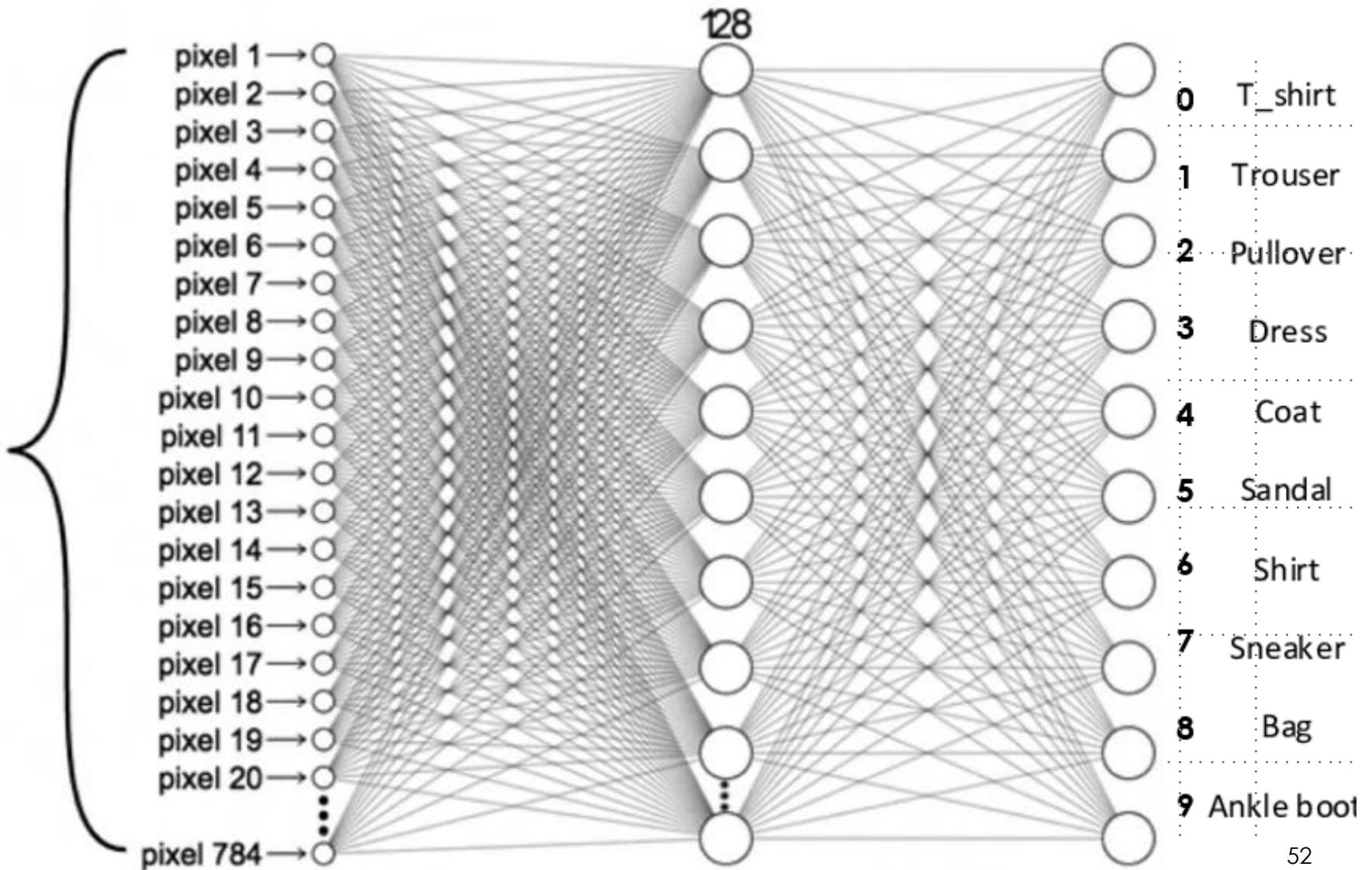
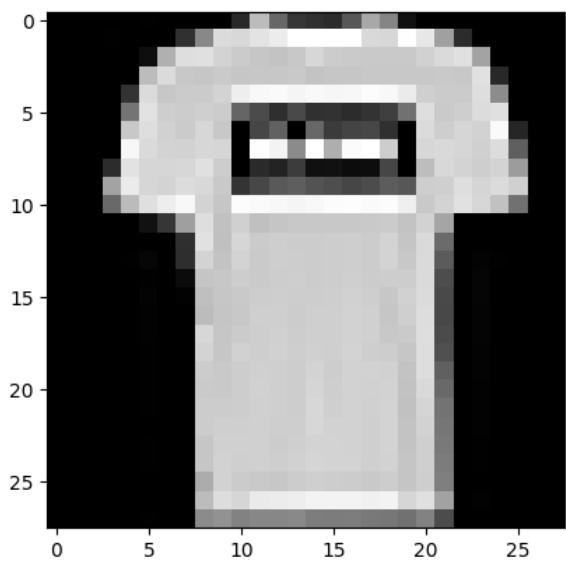
👉 Das 28×28 -Bild wird in eine lange Zahlenliste mit 784 Werten umgewandelt.

Die Klassen (Labels)

Das neuronale Netz soll:

👉 lernen, das richtige Kleidungsstück auf einem Bild zu erkennen

Daten: Fashion MNIST



15:00–15:20  Pause



Tag 2 15:20-16:40

Modellbewertung & Interpretation

- Ergebnisse analysieren
- Verbesserungen diskutieren



Tag 2 16:40-17:00

Ergebnisse & Diskussion

- Präsentation
- Reflexion
- Q&A