# Artificial Neural Networks

*Angelehnt an die Serie „ANN“ von 3Blue1Brown*

Heilbronn, den 24.05.2022

# Agenda

**01** Einführung ANN

**02** Feedforward Neural Network - Digit Recognition

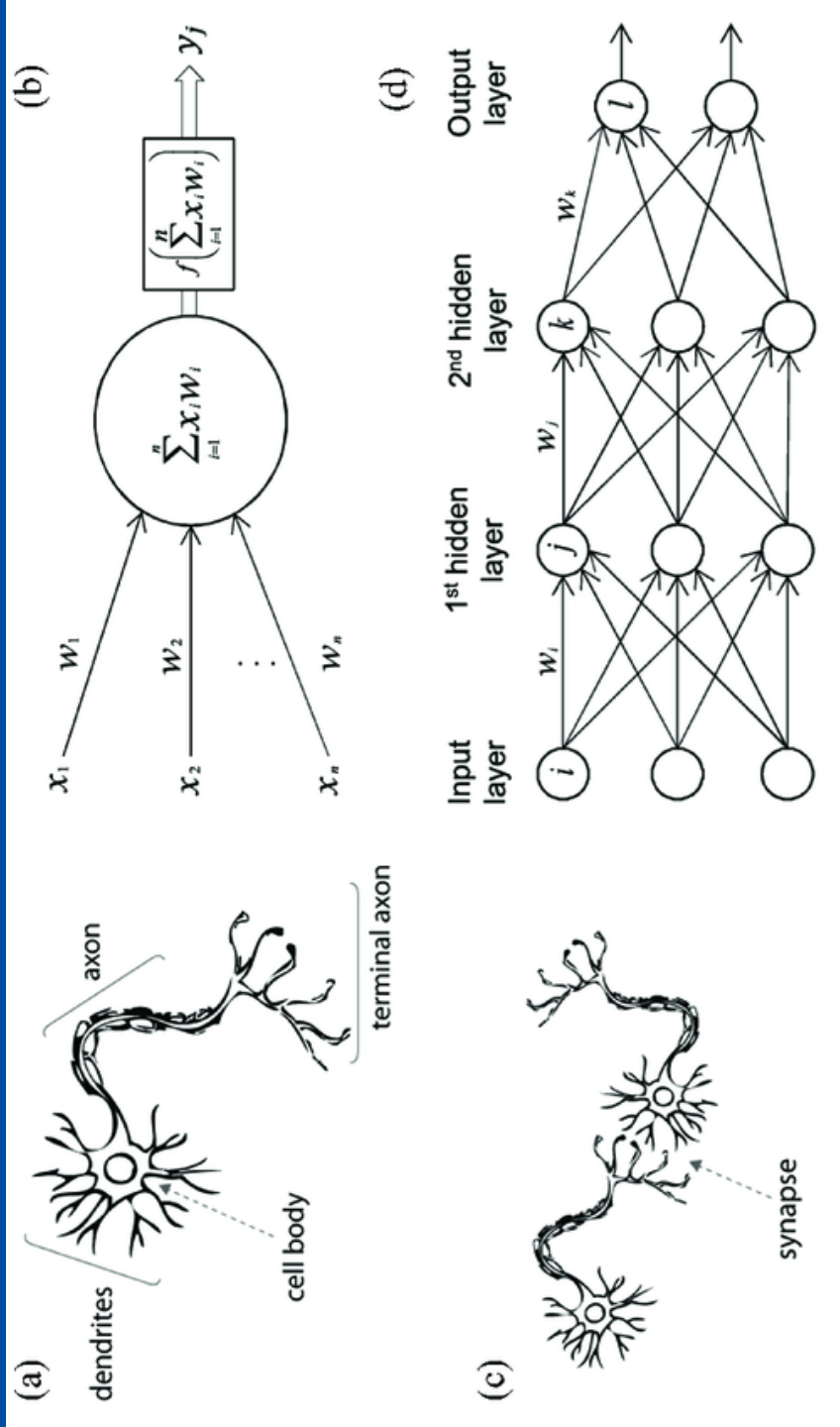
**03** Testing & Optimizing

**04** Auswertung unseres Modell

**05** Vergleich zu anderen NN

# 01 Einführung ANN

Artificial Neural Networks

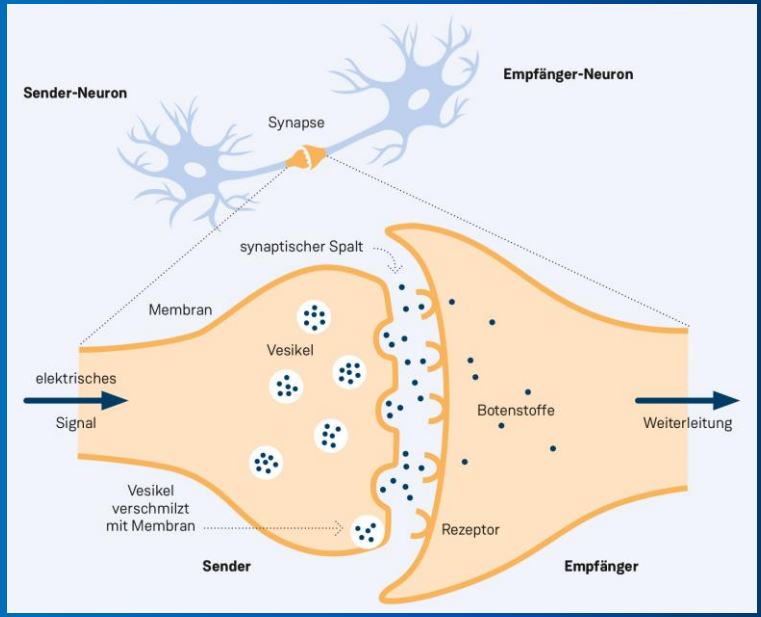


01

# Artificial Neural Networks

- Künstliche Neuronale Netze (KNN) sind inspiriert durch den Informationsfluss innerhalb des menschlichen Gehirns

- Neuronen, die mit anderen Neuronen verbunden sind und sich gegenseitig „Signale / Informationen“ senden

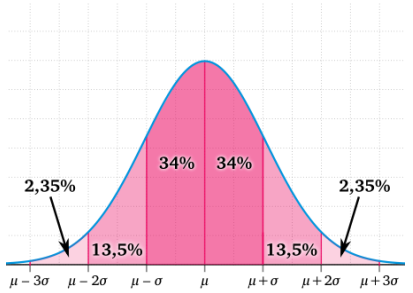



# Einsatzgebiete von Feedfordward NN‘s

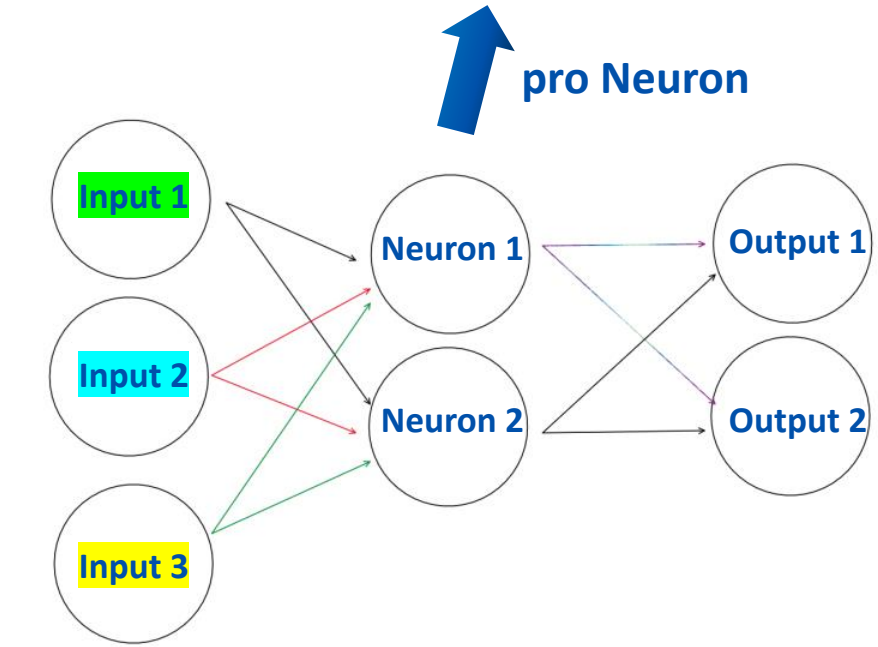
- **Pattern Recognition (unsupervised)**
  - Automatisches Erkennen von Mustern / Regelmäßigkeiten in Trainingsdaten
    - Clustering von Daten

- **Classification (supervised)**
  - Einteilung der Daten anhand der Merkmale in vordefinierte Klassen
    - Handschrifterkennung, Bildklassifikation

# Feedforward NN - Aufbau



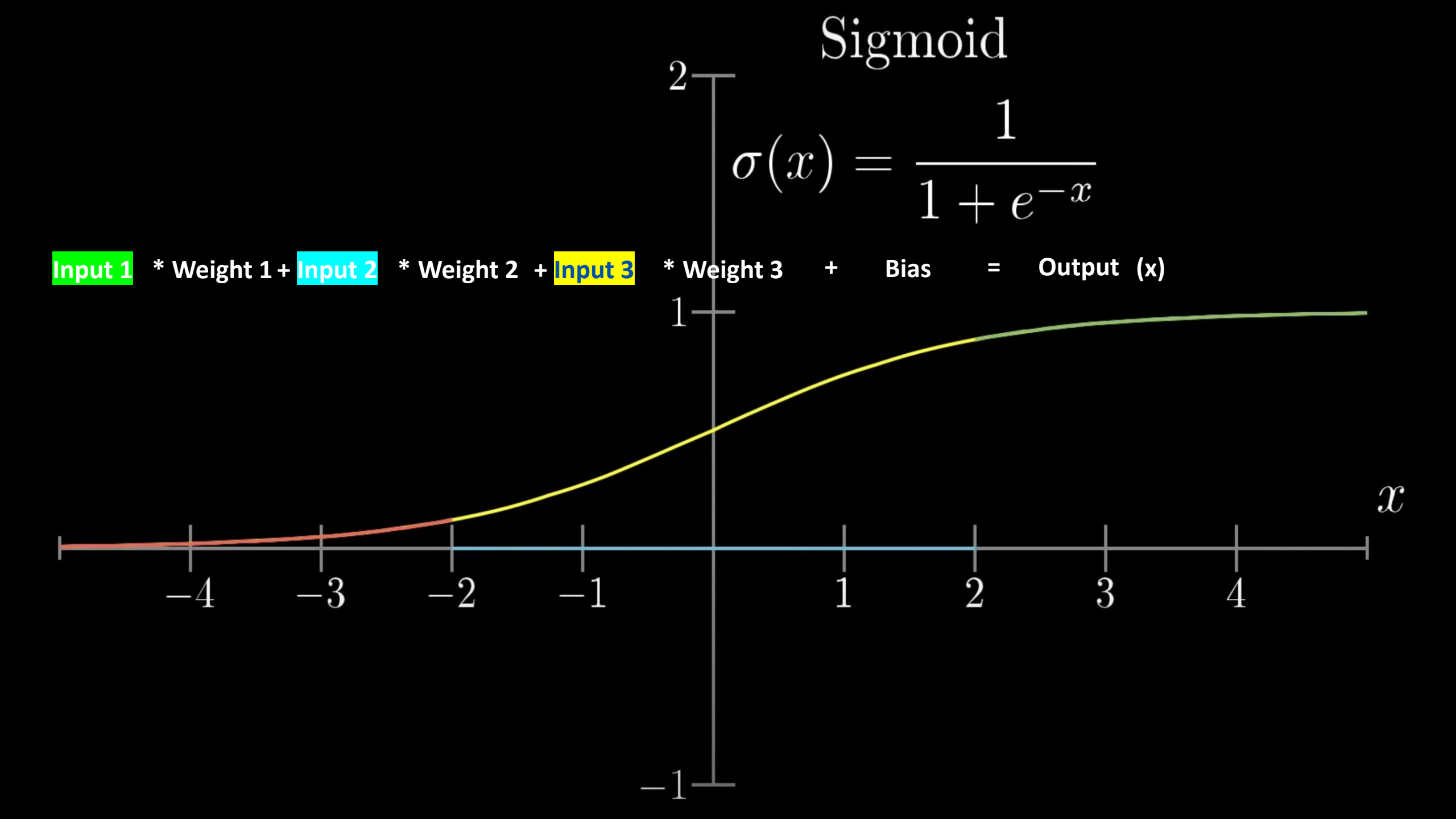

**Input 1 * Weight 1 + Input 2 * Weight 2 + Input 3 * Weight 3 + Bias = Output (x)**




*Sigmoid*

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

= Wert (zwischen 0 u. 1)


Sigmoid
$\sigma(x) = \frac{1}{1+e^{-x}}$
Input 1 * Weight 1 + Input 2 * Weight 2 + Input 3 * Weight 3 + Bias = Output (x)
2
1
−1
x
−4
−3
−2
−1
1
2
3
4

01

# Wie wird unser Netzwerk „schlauer“?

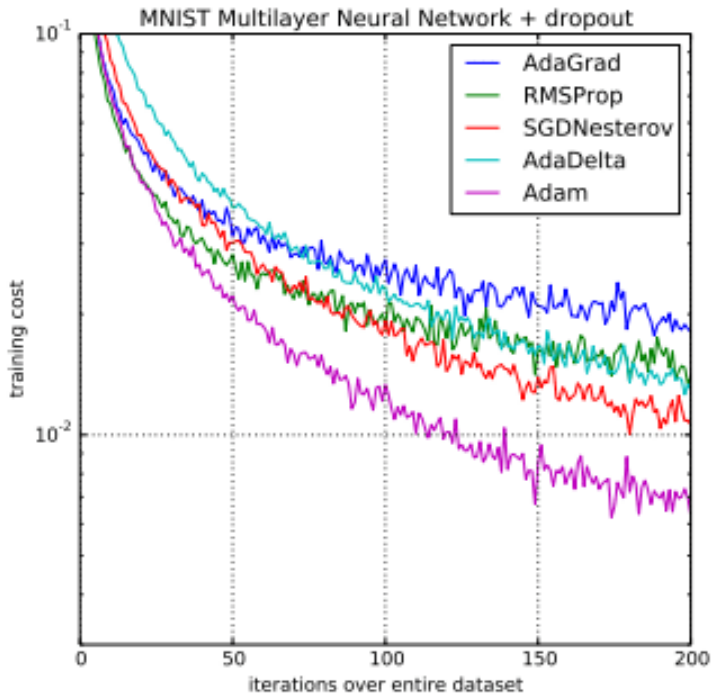
Supervised Learning / Überwachtes Lernen:

**Back Propagation**

- Algorithmus, der für den Lernprozess des Netzwerks zuständig ist
  - Cost des ANN: unterschied zwische output und dem erwarteten Wert
  - Cost wird genutzt um die weights und bias anzupassen
  - Durchschnitt von allen weights und bias bilden das verbesserte Model

→ Prozess wird so lange wiederholt bis die optimale Anpassung gefunden wurde

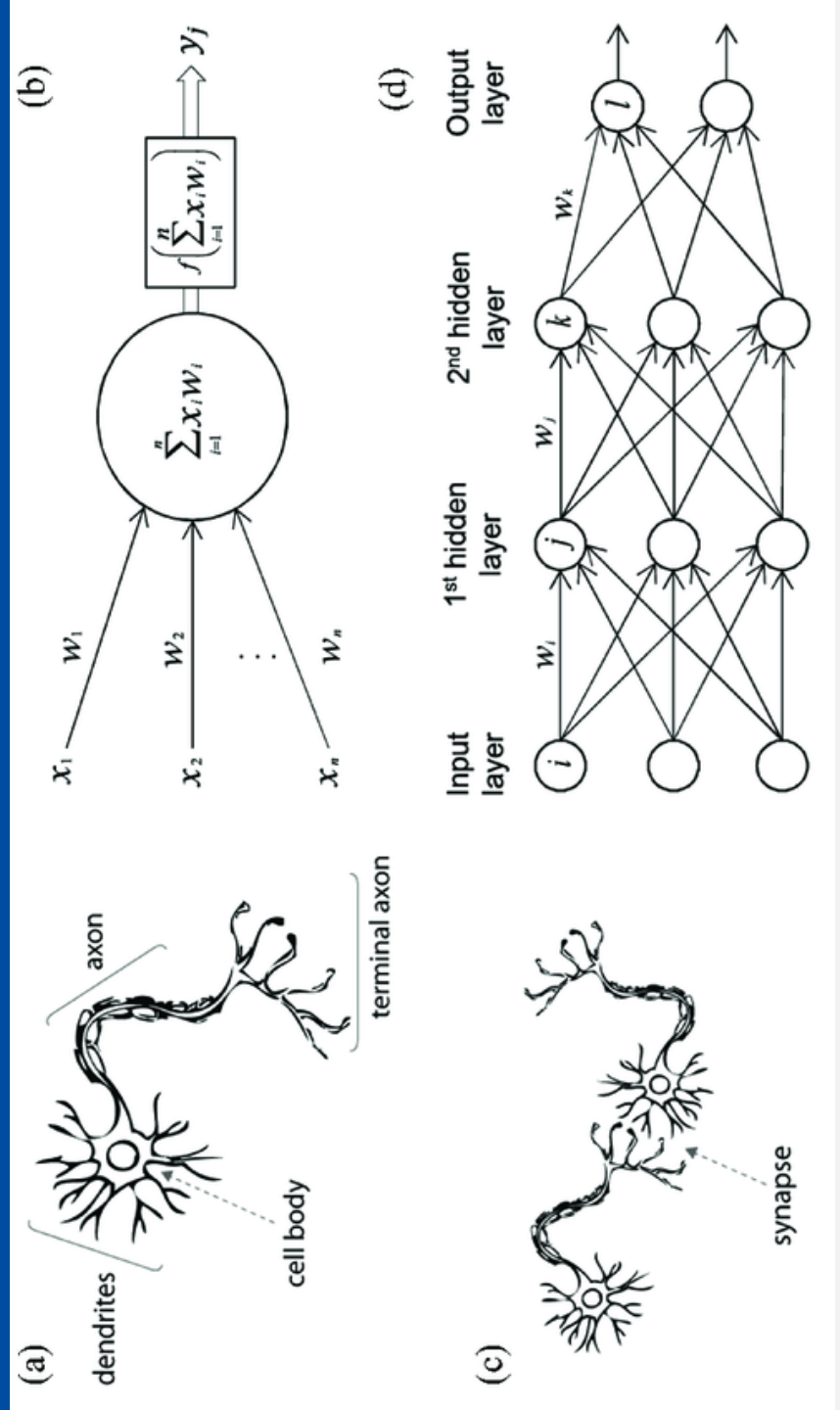
→ Kompromiss, um alle Klassen möglich genau zu erraten



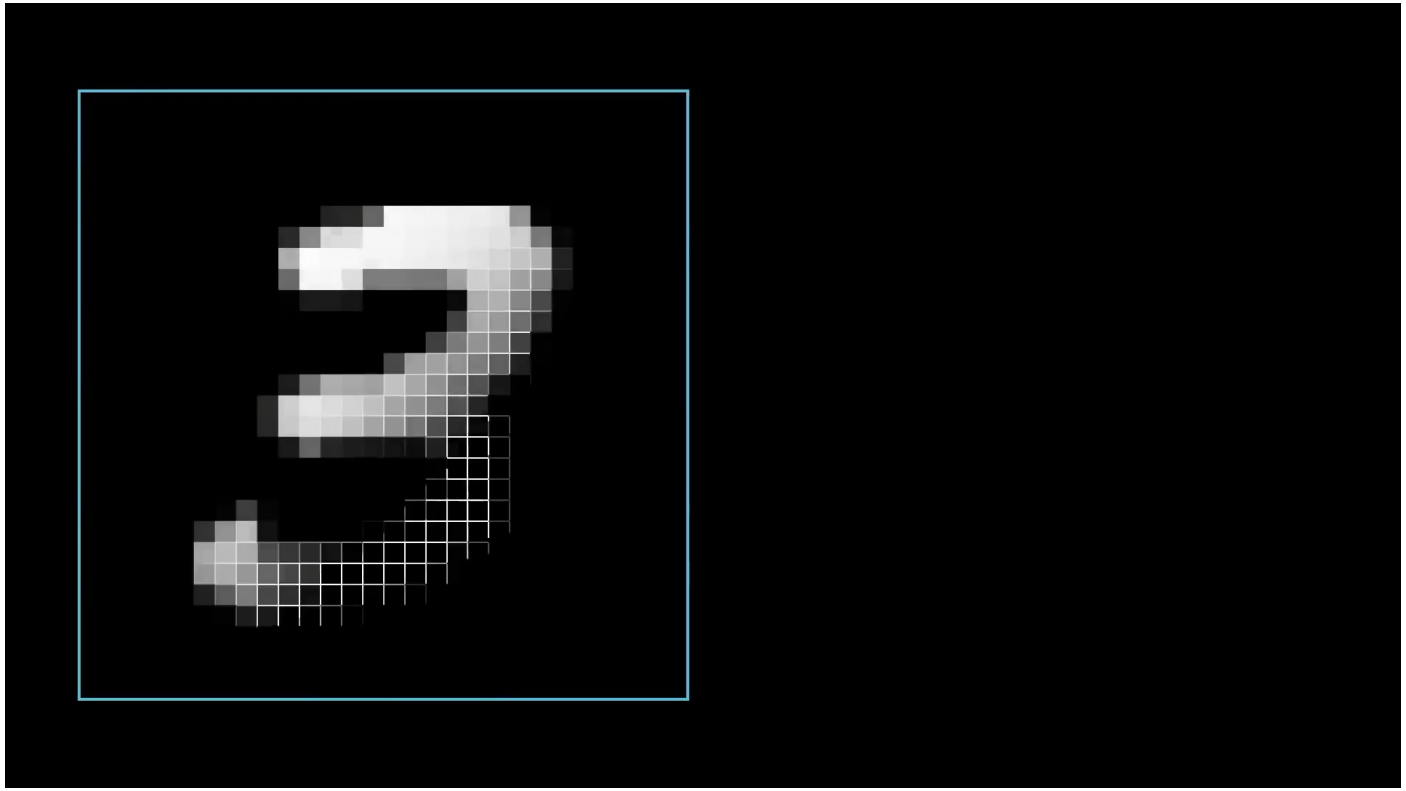
# 02

# Feedforward NN - Digit Recognition

Artificial Neural Networks



# Einführung – Was ist unser Ziel?

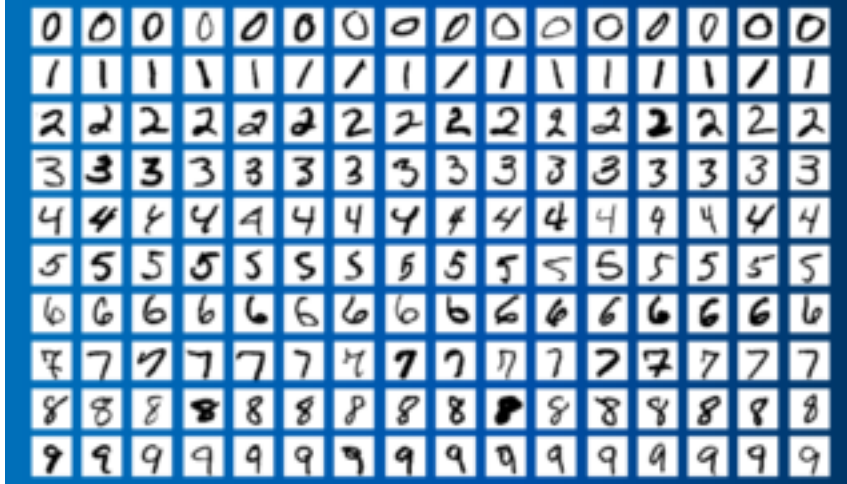


**Unser Neuronales Netz soll anhand eines Bildes eine handgeschriebene Zahl identifizieren**

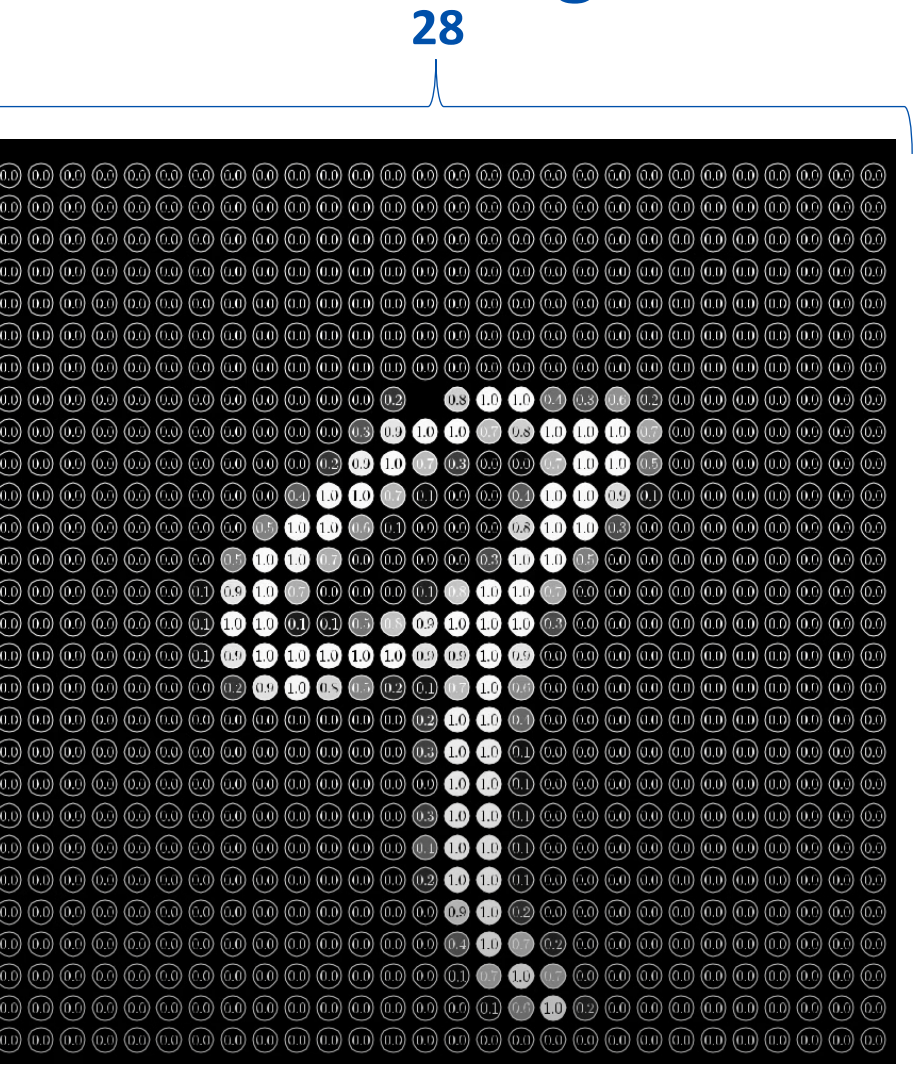
# MNIST- Datensatz (Input)

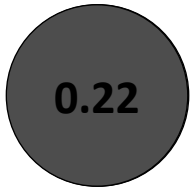
- **Mixed National Institute of Standards and Technology** (Öffentliche Datenbank für handgeschriebene Zahlen)
- Training Datensatz aus 60.000 Bildern
- Testing Datensatz aus 10.000 Bildern

- Zahlen aus Training- und Test-Datensatz von unterschiedlichen Personen geschrieben
  - SD-3 (Mitarbeiter von Census Bureau)
  - SD-1 (Schüler)



# Input - Verarbeitung



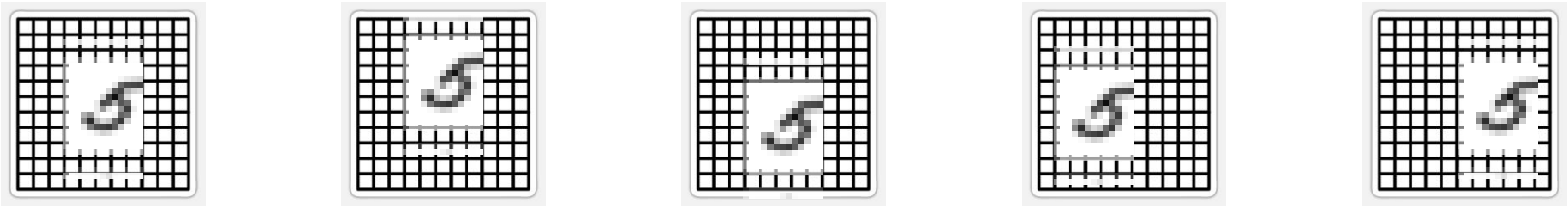

- Neuronenfeld **28x28** = 784 Pixel
- Jedes **Neuron** nimmt Wert zwischen **0 u. 1** an

Je heller, desto höher der Wert (näher an 1)

Je dunkler, desto kleiner der Wert (näher an 0)

→ Jedes Neuron im Input-Layer ist für ein Pixel zuständig

# Erweiterung des Datensatz



**oben** **unten** **links** **rechts**

**Data without expandation**: 70.000 Bilder

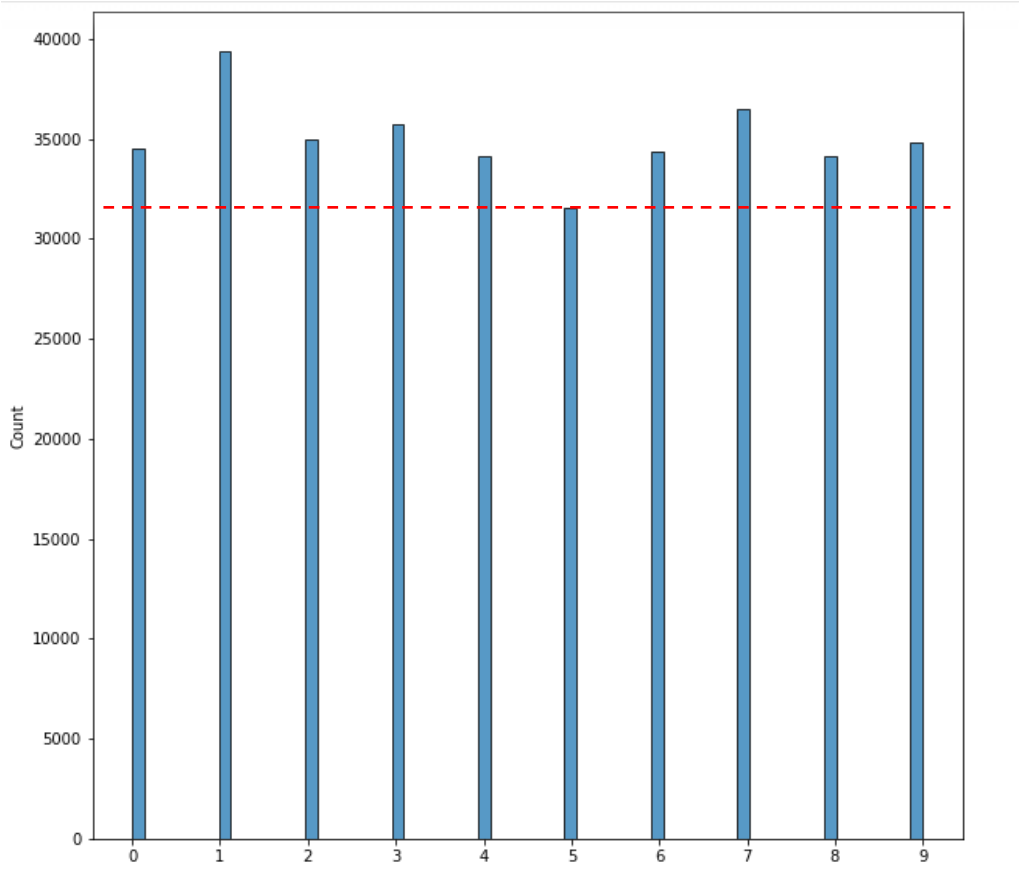
50.000 (training_data) + 10.000 (validation_data) + 10.000 (test_data)

**Expanded Data**: 350.000 Bilder

50.000 Training Data + 4 * 50.000 (Bilder nach oben, unten, links, recht) +
10.000 Test Data + 4* 10.000 (Bilder nach oben, unten, links, recht) +
10.000 Validation Data + 4 * 10.000 (Bilder nach oben, unten, links, recht)

→ Zusammen gebündelt in **training_data**

# Undersampling



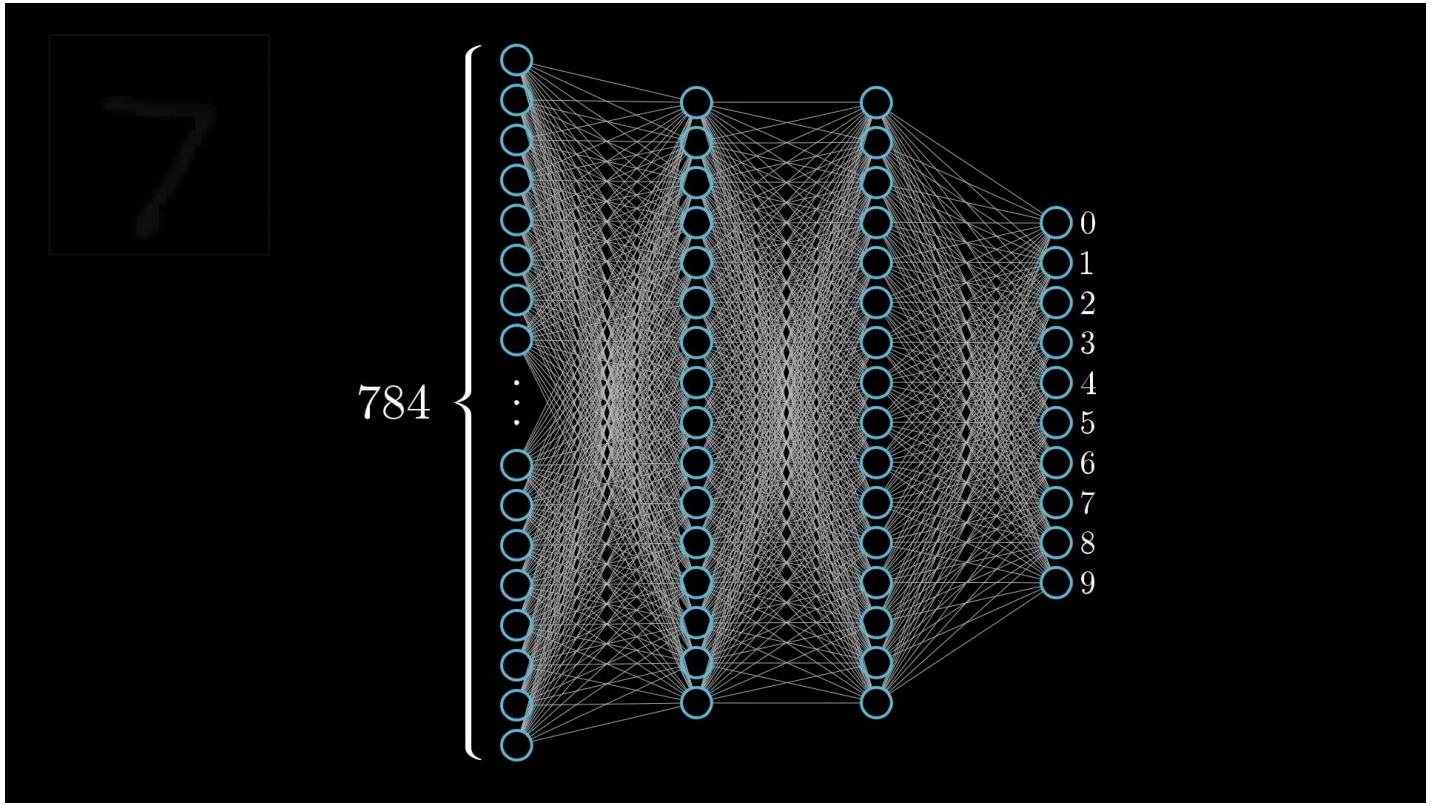

**31.565 Fünfen in unserem Datensatz**

**Alle Zahlen mit mehr als 31.565 Bildern sollen auf diese Anzahl begrenzt werden**

**→ Jede Zahl hat die selbe Anzahl an Trainingsdaten**

# Unser erstes / simples Modell

**Input 1 * Weight 1 + Input 2 * Weight 2 + Input 3 * Weight 3 (…) + Bias = Output (x)**




Jedes Neuron im Hidden-Layer bekommt einen Input von allen Neuronen der Schicht davor

2 Hidden Layer mit 16 Neuronen

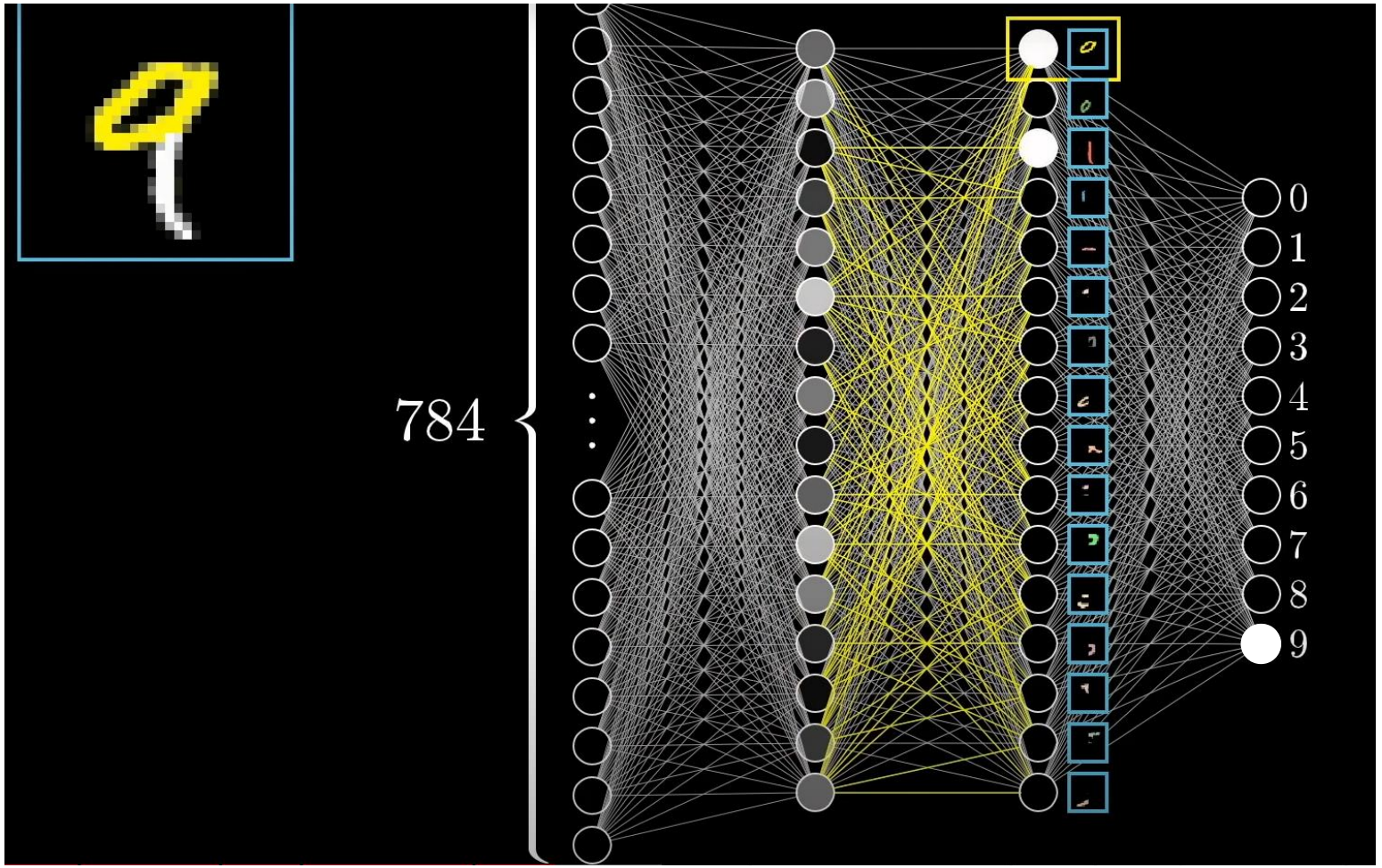
Jedes Neuron besitzt eigene Weights

**784×16+16×16+16×10 weights**

**16 +16+10 biases**

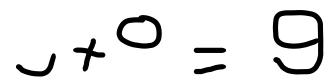
**→ 13,002 Optimiermöglichkeiten**

# Mögliche Verarbeitung (Hidden Layer)




**Jede Zahl besteht aus einzelnen Komponenten**

**Die Zahl 9 besteht z.B aus**



**In den Hidden Layer werden die bestandteile einer Zahl ausgewertet und kombiniert**

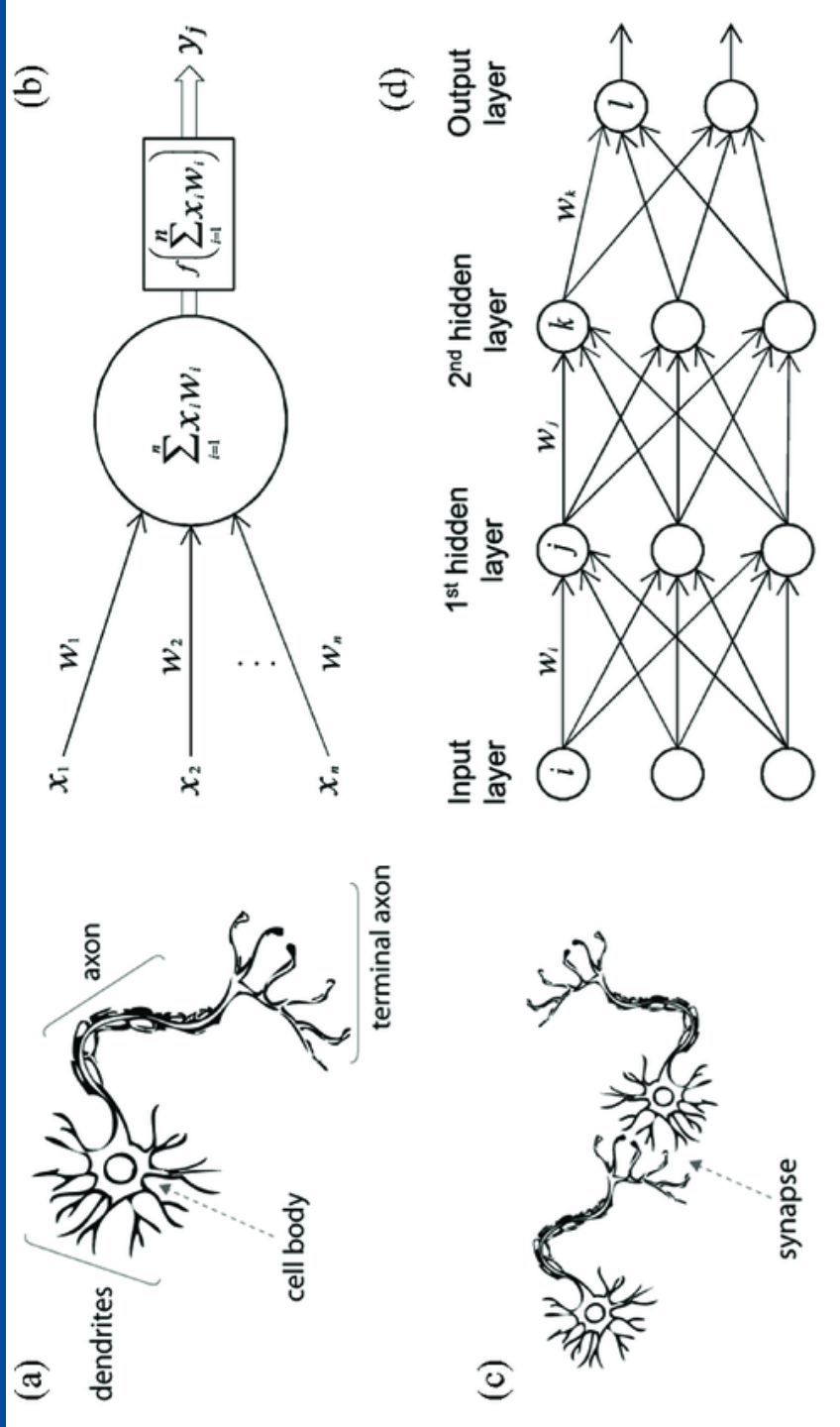
**=> Output**

**In dem dritten Layer wird entschieden, inwiefern der Input (Bild), dem Pixelausschnitt ähnelt**

# 03

# Testing & Optimizing

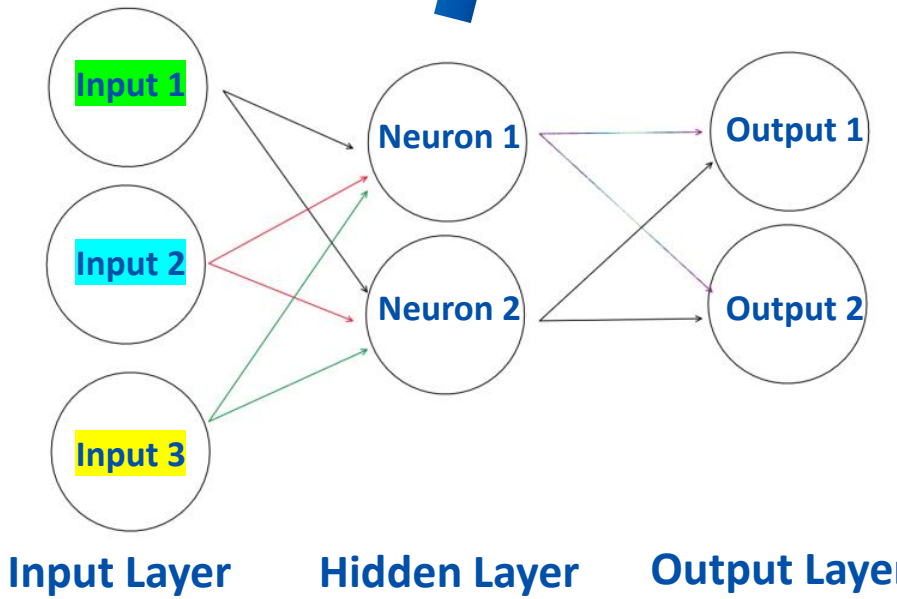
Artificial Neural Networks



# Recap Feedforward NN - Aufbau

Input 1 * Weight 1 + Input 2 * Weight 2 + Input 3 * Weight 3 + Bias = Output (x)

pro Neuron



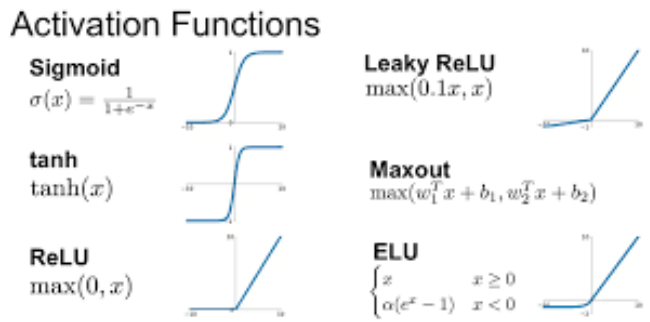

Sigmoid

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

= Wert (zwischen 0 u. 1)

# Adjusting Hyper-Parameter

- **Number of Hidden Layers, Neurons**(mehr Neurons dauert länger)
- **Activation** (sigmoid, ReLU, tanh etc.)
- **Solver** (Verschiedene Ansätze, um die Weights anzupassen)




- **Batch Size** (Legt fest, wie viele Bilder werden bearbeitet  bis die weights angepasst werden)
- **Learning Rate** (Bestimmung der Start-rate / Schrittgröße, Veränderung der Rate im Lern-Prozess , Stop festlegen)

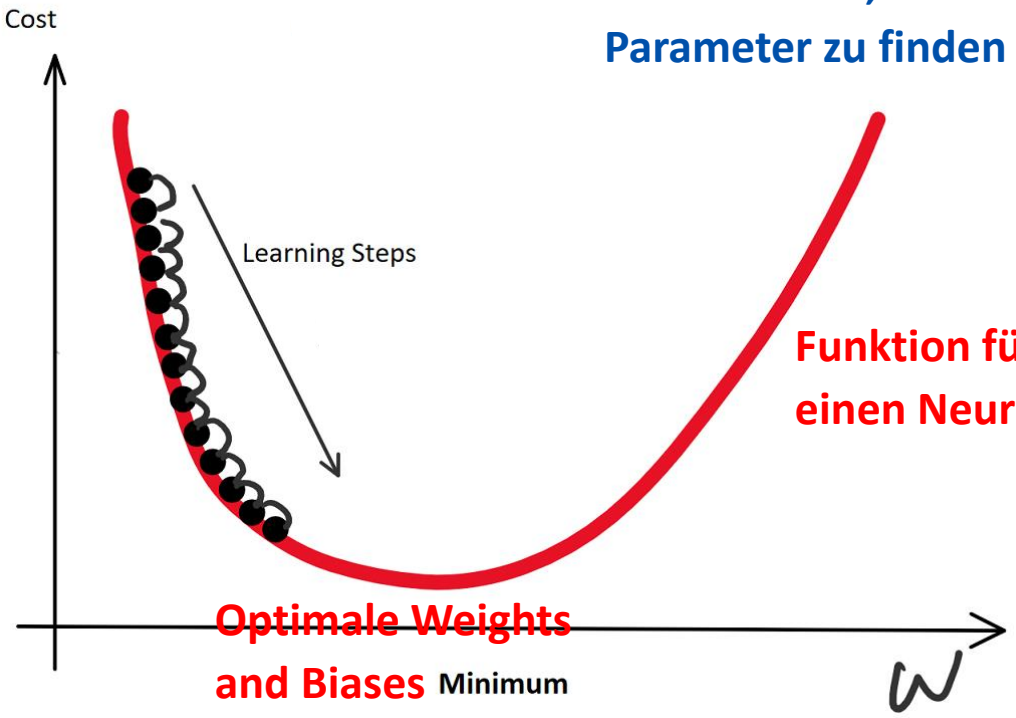
# Learning Rate ($\eta$) – Kostenfunktion

**Low $\eta$**

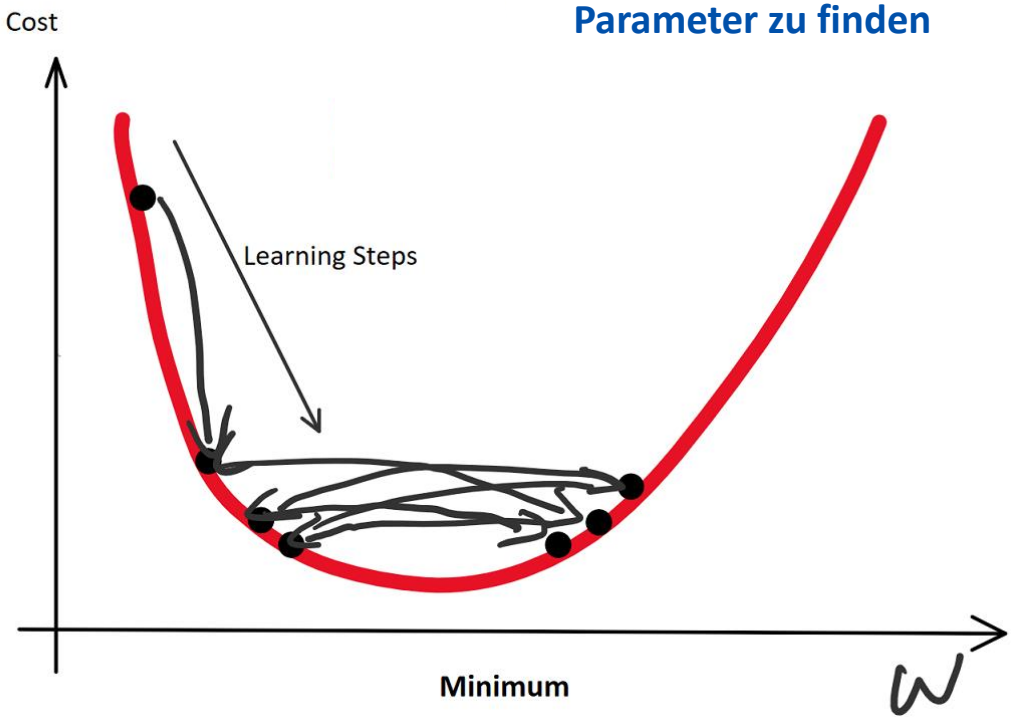
- Niedrige Annäherung pro Update
- Hohe Dauer
- Hohe Chance, den richtigen Parameter zu finden

**High $\eta$**

- Schnelle Annäherung pro Update
- Niedrige Dauer
- Niedrige Chance, den richtigen Parameter zu finden

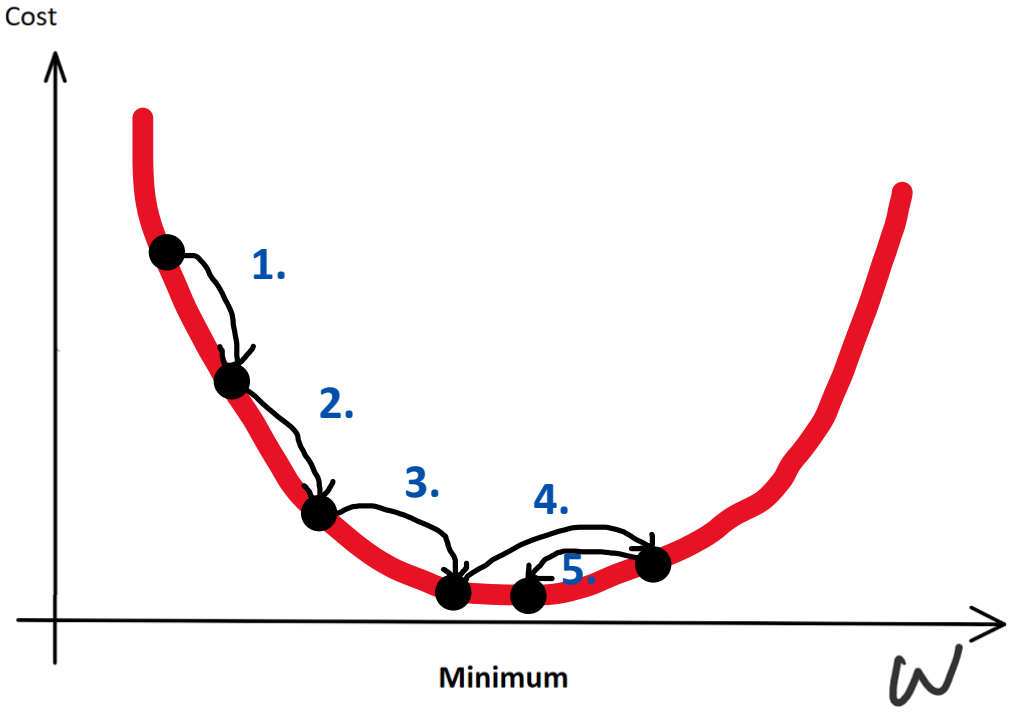



Funktion für einen Neuron



# Annealing the Learning Rate → adaptive

learning_rate{*'constant', 'invscaling', 'adaptive'*}



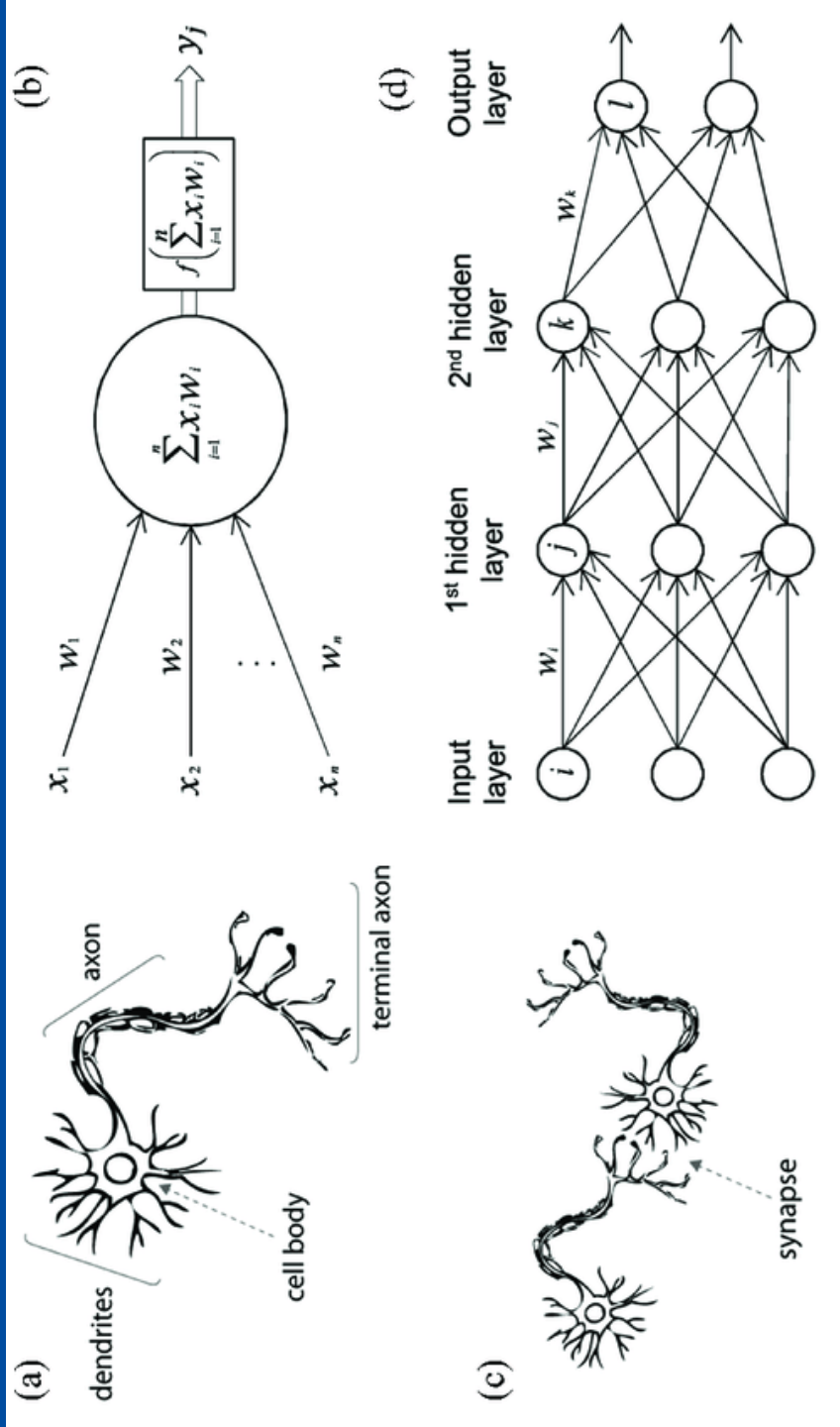

**Loss rate = Fehlerrate im Training des NN**

1. **Loss rate wird geringer, $\eta$ bleibt gleich**
2. **Loss rate wird geringer, $\eta$ bleibt gleich**
3. **Loss rate wird geringer, $\eta$ bleibt gleich**
4. **Loss rate wird höher, $\eta$ wird verkleinert**
5. **Loss rate wird geringer**

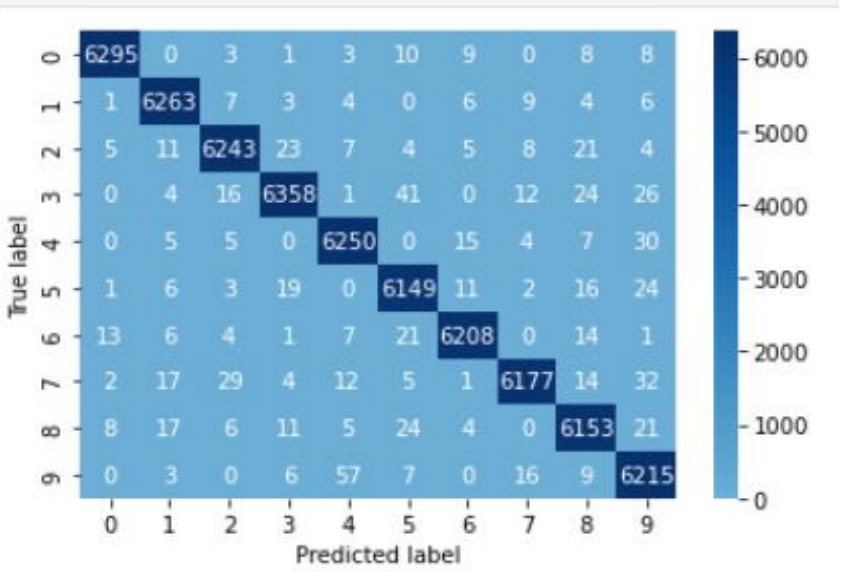
**→ Optimum/Minimum gefunden**

# 04 Auswertung unseres Modells

Artificial Neural Networks



# Evaluation of our model (Confusion matrix)




| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.99 | 0.99 | 0.99 | 6337 |
| 1 | 0.99 | 1.00 | 0.99 | 6303 |
| 2 | 0.99 | 0.99 | 0.99 | 6331 |
| 3 | 0.98 | 0.99 | 0.99 | 6482 |
| 4 | 0.99 | 0.99 | 0.99 | 6316 |
| 5 | 0.99 | 0.99 | 0.99 | 6231 |
| 6 | 0.99 | 0.99 | 0.99 | 6275 |
| 7 | 0.99 | 0.98 | 0.99 | 6293 |
| 8 | 0.99 | 0.98 | 0.99 | 6249 |
| 9 | 0.99 | 0.98 | 0.98 | 6313 |
| accuracy | | | 0.99 | 63130 |
| macro avg | 0.99 | 0.99 | 0.99 | 63130 |
| weighted avg | 0.99 | 0.99 | 0.99 | 63130 |

Unser Modell identifiziert eine handgeschriebene Zahl in Form eines Bildes mit einer **Genauigkeit von 0.988**

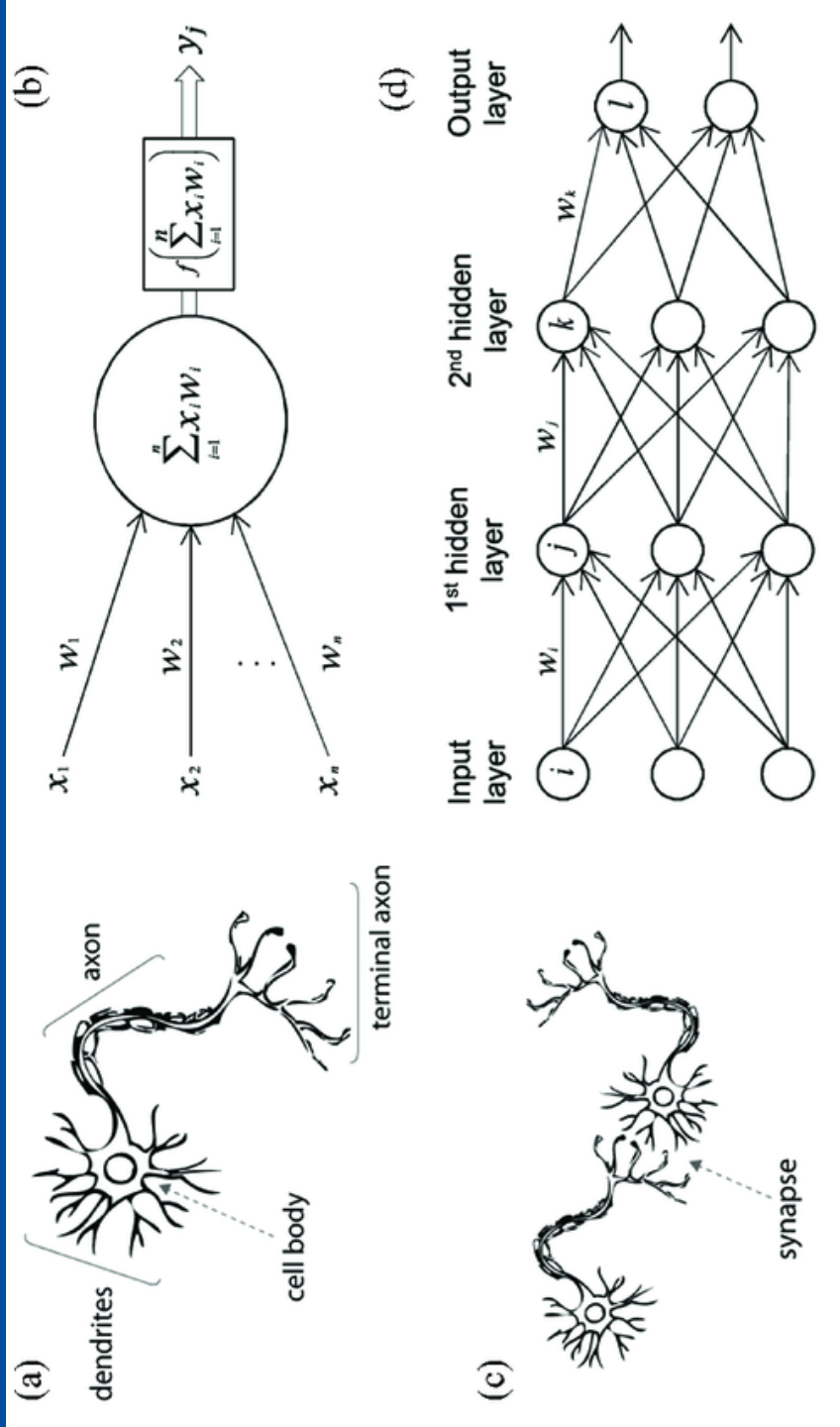
- Niedrige Loss Rate
- „Niedrige Anzahl an falsch erratenen Zahlen (im Verhältnis zur Menge der Daten (63130 Test Daten))
- Hohe precision, recall, f1-score, support

04

# Vergleich der Modelle

# 05 Vergleich zu anderen Neuronalen Netzen

Artificial Neural Networks



# Arten von Neuronalen Netzwerken

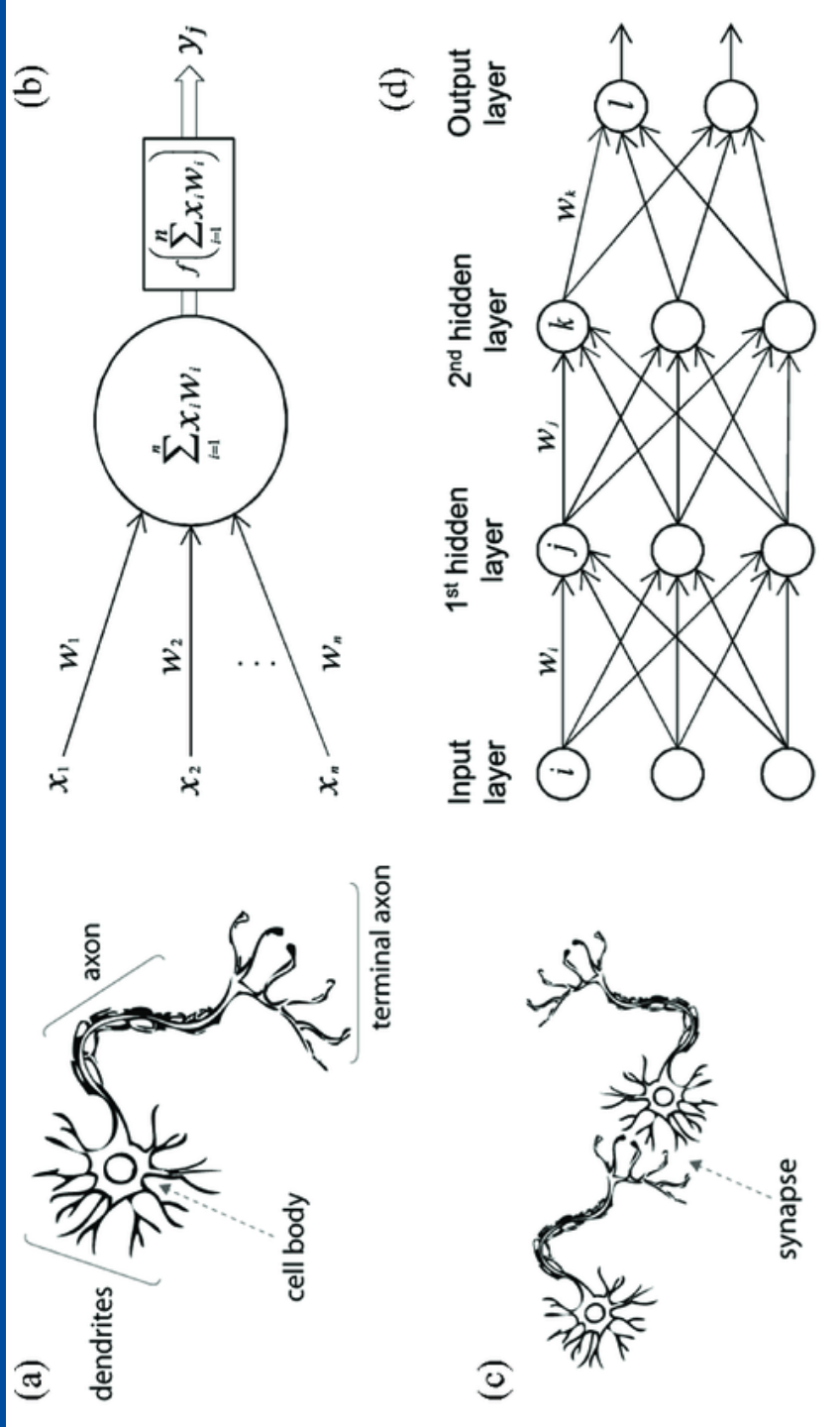
| **Feedforward Neural Network** | **Convolutional (CNN)** | **Recurrent (RNN)** | **Generative Adversarial Networks (GANs)** |
|---|---|---|---|
| Informationsfluss ausschließlich vorwärtsgerichtet | Convolutional Layer - Pooling Layer - Fully-Connected Layer. | Output wird wieder als Input verwendet<br>→ Rückkopplung | Erzeugen aus Input neue Daten<br><br>Generatoren und Diskriminatoren |
| → Klassifikation | → Bilderkennung | → Spracherkennung | → Erzeugen eines fiktiven Bild |

05

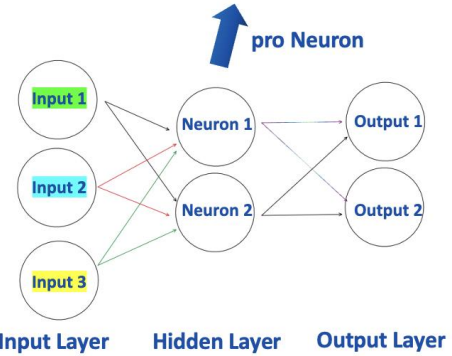
# Einsatzgebiete


# Das Wichtigste auf einen Blick

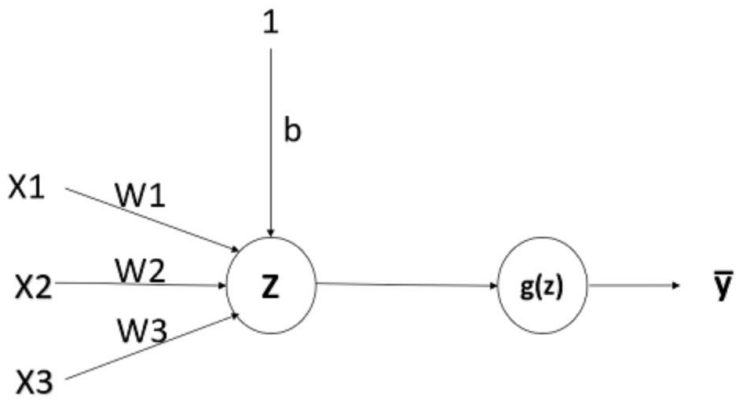
Artificial Neural Networks



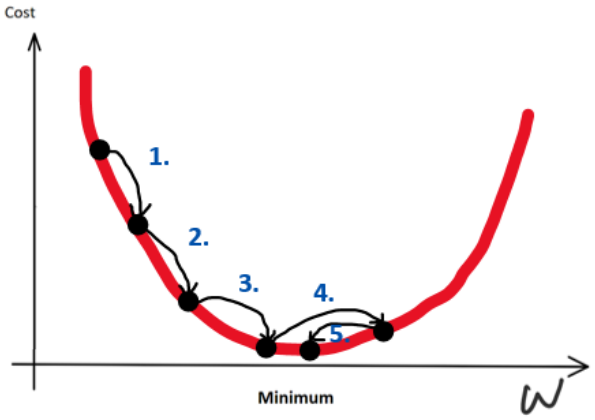
Input 1 * Weight 1 + Input 2 * Weight 2 + Input 3 * Weight 3 + Bias = Output (x)




Sigmoid $f(x) = \frac{1}{1+e^{-x}}$ = Wert (zwischen 0 u. 1)

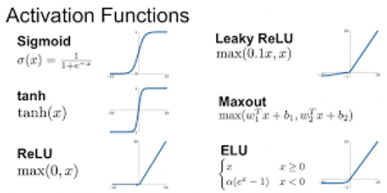



learning_rate{'constant', 'invscaling', 'adaptive'}




# Adjusting Hyper-Parameter

- **Number of Hidden Layers, Neurons** → braucht sehr viel Rechenleistung
- **Activation** (sigmoid, ReLU, tanh etc.)
- **Solver** (Verschiedene Ansätze, um die Weights anzupassen)
- **Batch Size** (Legt fest, wie viele Bilder werden bearbeitet bis die weights angepasst werden)
- **Learning Rate** (Bestimmung der Start-rate / Schrittgröße, Veränderung der Rate im Lern-Prozess , Stop festlegen)




# Quellen


- https://www.youtube.com/watch?v=aircAruvnKk&list=LL&index=2&ab_channel=3Blue1Brown
- https://www.youtube.com/watch?v=IHZwWFHWa-w&list=LL&index=1&ab_channel=3Blue1Brown
- https://www.marketing-boerse.de/fachartikel/details/2049-pattern-matching---muster-fuer-marketing-nutzen/173145 Letzter Zugriff: 12.05.22
- https://www.bigdata-insider.de/die-wichtigsten-typen-neuronaler-netze-fuer-deep-learning-a-1101586/ Letzter Zugriff: 16.05.22
- https://data-science-blog.com/blog/2019/01/13/training-eines-neurons-mit-dem-gradientenverfahren/ Letzter Zugriff: 20.05.22
- https://machine-learning.paperspace.com/wiki/weights-and-biases#:~:text=In%20an%20ANN%2C%20each%20neuron,inputs%20along%20with%20the%20bias.&text=Weights%20control%20the%20signal%20(or,the%20connection)%20between%20two%20neurons. Letzter Zugriff: 20.05.22
- https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/ Letzter Zugriff: 22.05.22
- https://github.com/MichalDanielDobrzanski/DeepLearningPython Letzter Zugriff: 23.05.22
- Powerpoint Ruben Nuredini

# Vielen Dank für eure Aufmerksamkeit!

**Bei späteren Fragen könnt Ihr euch gerne bei uns melden.**


**Moritz Theis**

moritz.theis@gmail.com

**Máté Benedek Jordán**

mate.jordan.b@gmail.com


LIDL