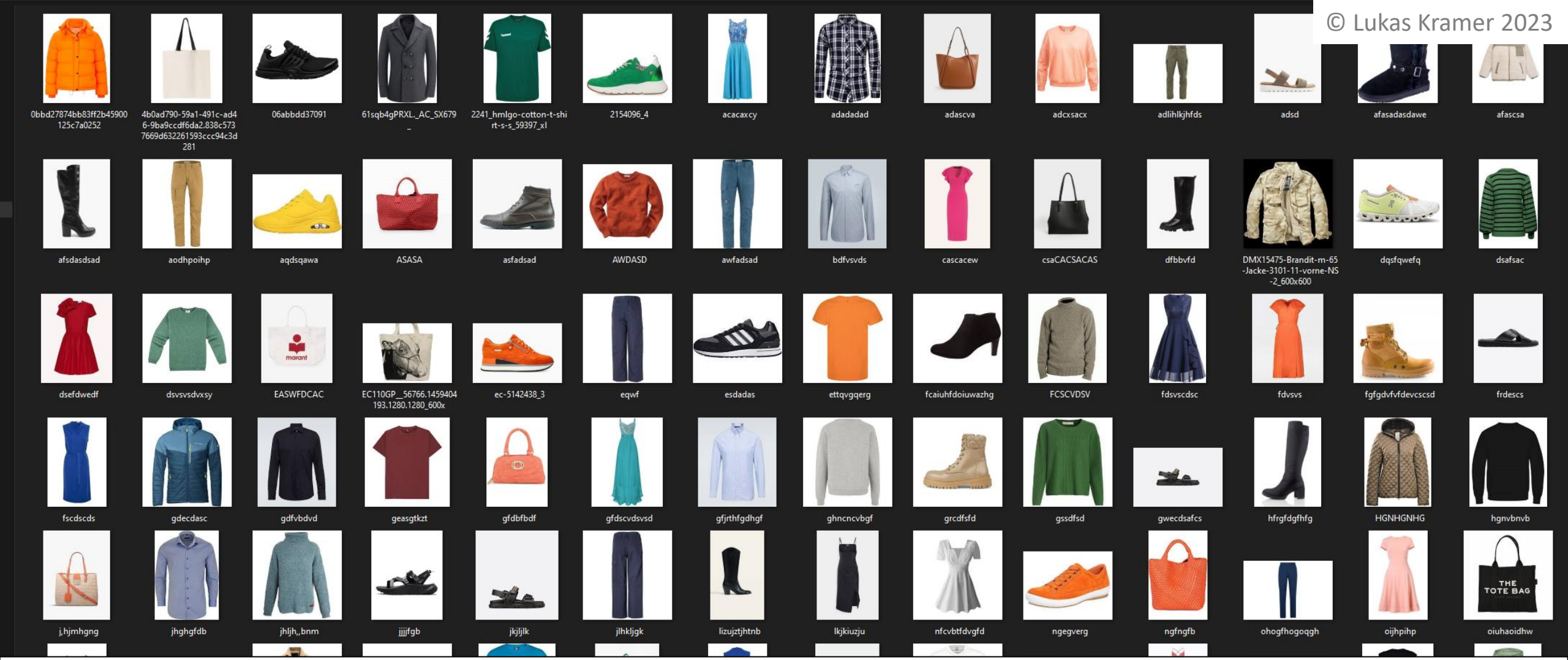
Das Problem

Modebilder sortieren



Kategorien / labels:

T-shirt / Top – Hosen (Trousers) – Pullover – Kleid (Dress) – Sandale – Jacke (Coat) – Hemd (Shirt) – Sneaker – Tasche (Bag) – Stiefel (Ankle Boot)

Neuronale Netze

(https://praxistipps.focus.de/kuenstliche-neuronale-netze-das-versteht-man-darunter_124885)

Gliederung

Was sind Neuronale Netze?

- Neuronale Netze als Teil von KI
  - Machine Learning und deep learning
- Einsatzgebiete von Neuronalen Netzen

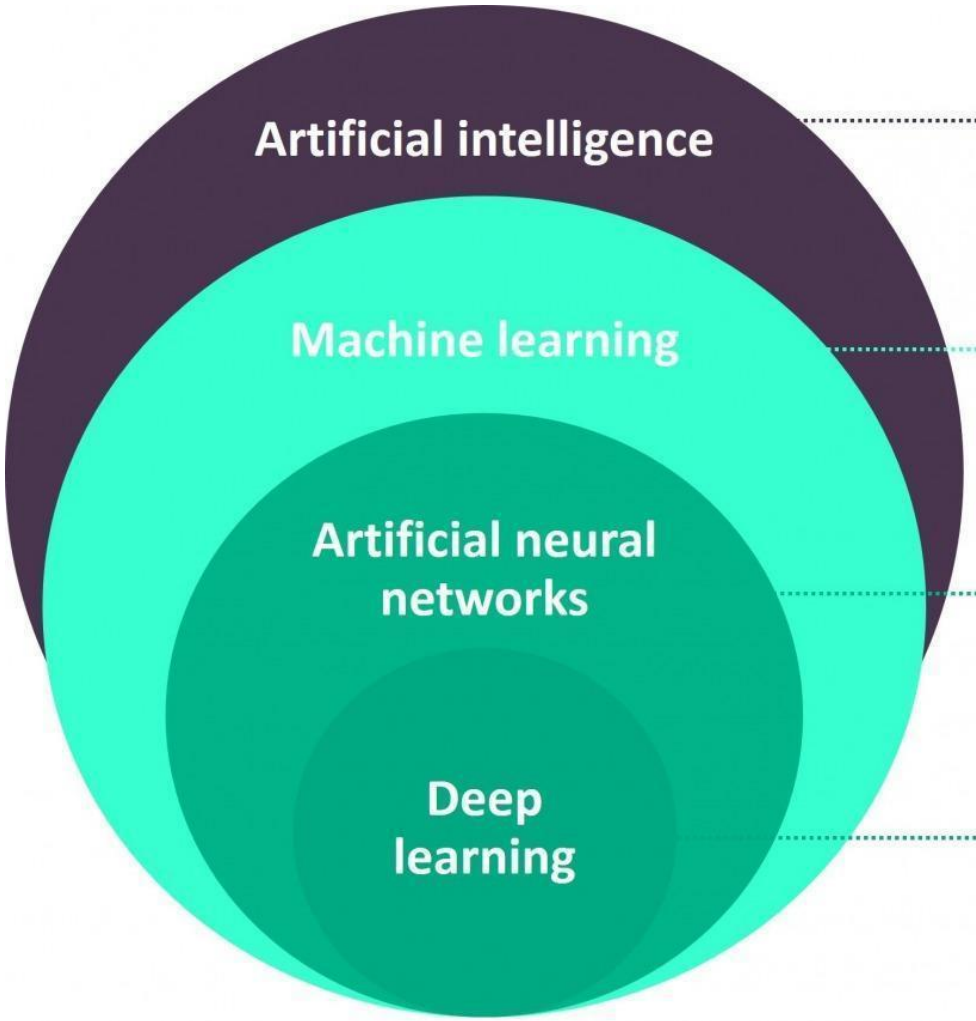
Wie funktionieren Neuronale Netze?

- Darstellung von Daten mithilfe von Vektoren und Matrizen
- Aufbau von Neuronalen Netzen (dense neural networks)
- Lineare Abbildungen -> Wie ein Neuronales Netz aus Daten Vorhersagen generiert
- Stochastic Gradient Descent -> Wie ein Neuronales Netz lernt
- -> Die Hyperparameter eines Neuronalen Netzes

Konkrete Lösungen für das Szenario

- Mehrere Modelle (bei denen verschiede Hyperparameter verändert werden) + deren Vorhersagen -> Auswirkung der veränderten Hyperparameter

Was sind Neuronale Netze?



Artificial intelligence (AI)

Machine learning (ML)

Artificial neural networks (ANN)

Deep learning (DL)

(https://www.bbntimes.com/images/AI_Complete_Graph.jpeg)

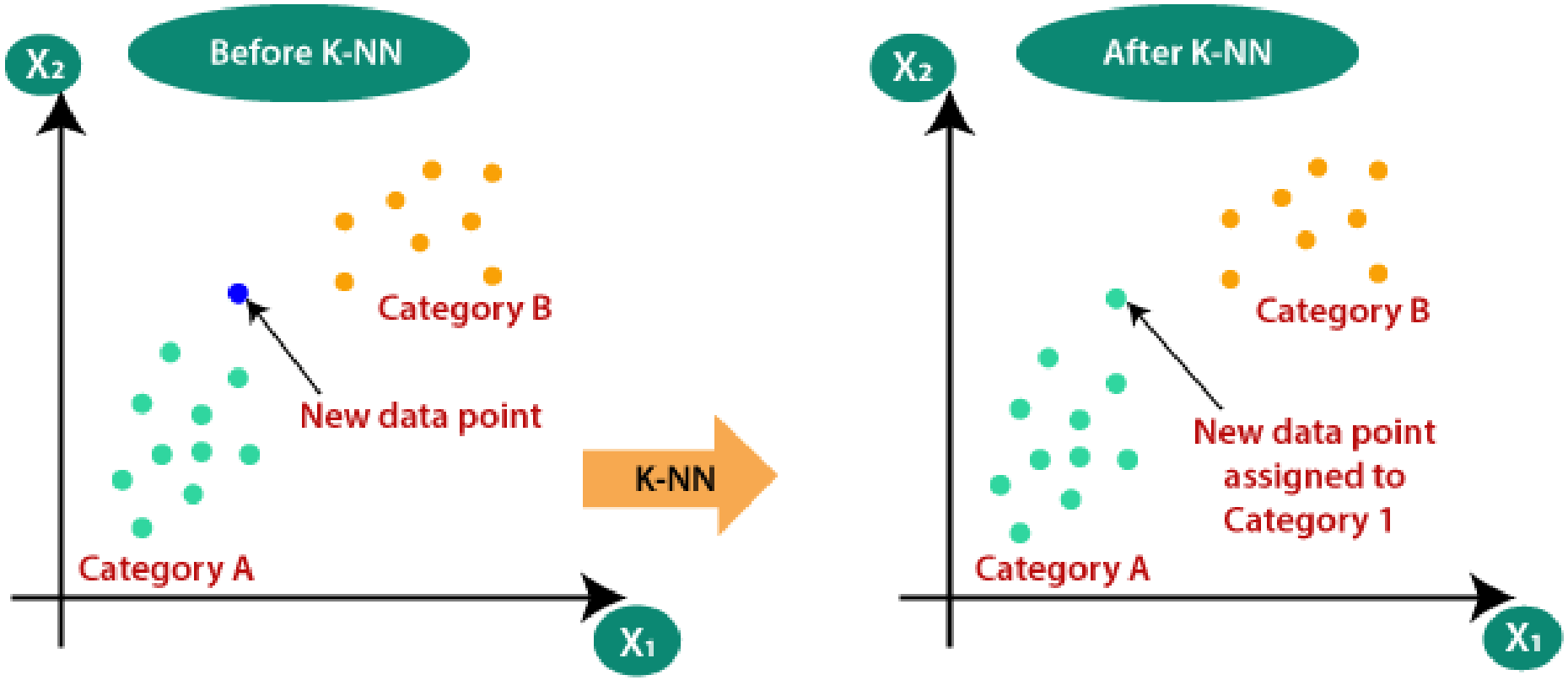
Beispiele – AI

Expertensysteme

(Elden Ring, FromSoftware)

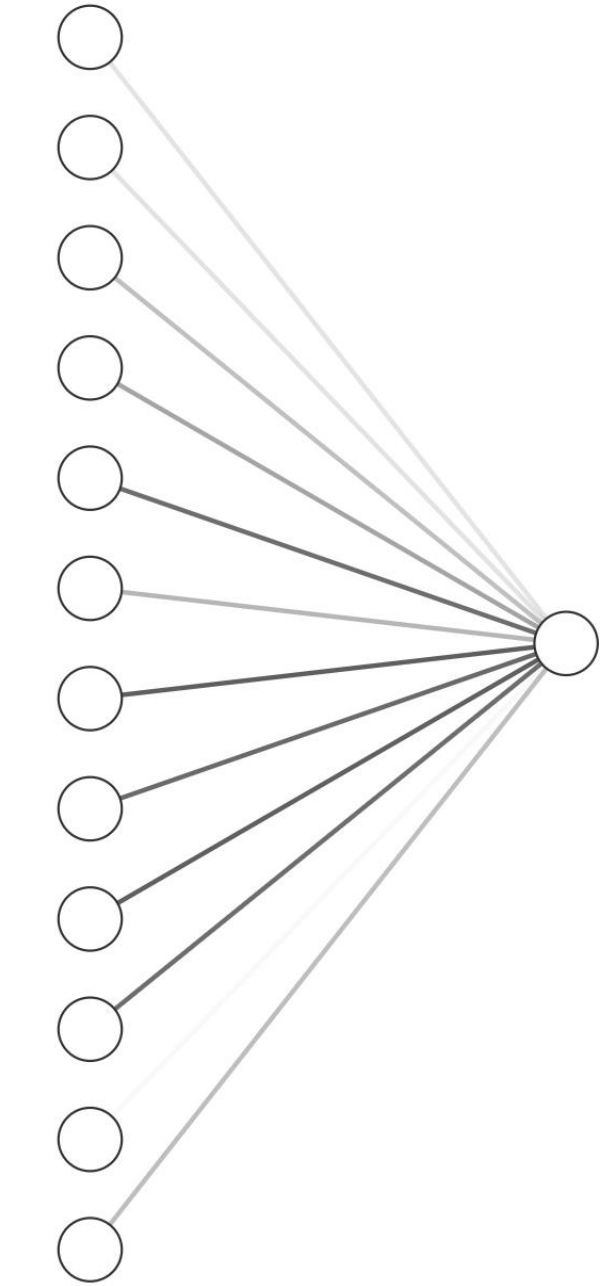
(https://www.iosb.fraunhofer.de/de/projekte-produkte/medizinische-expertensysteme.html)

Beispiele – ML



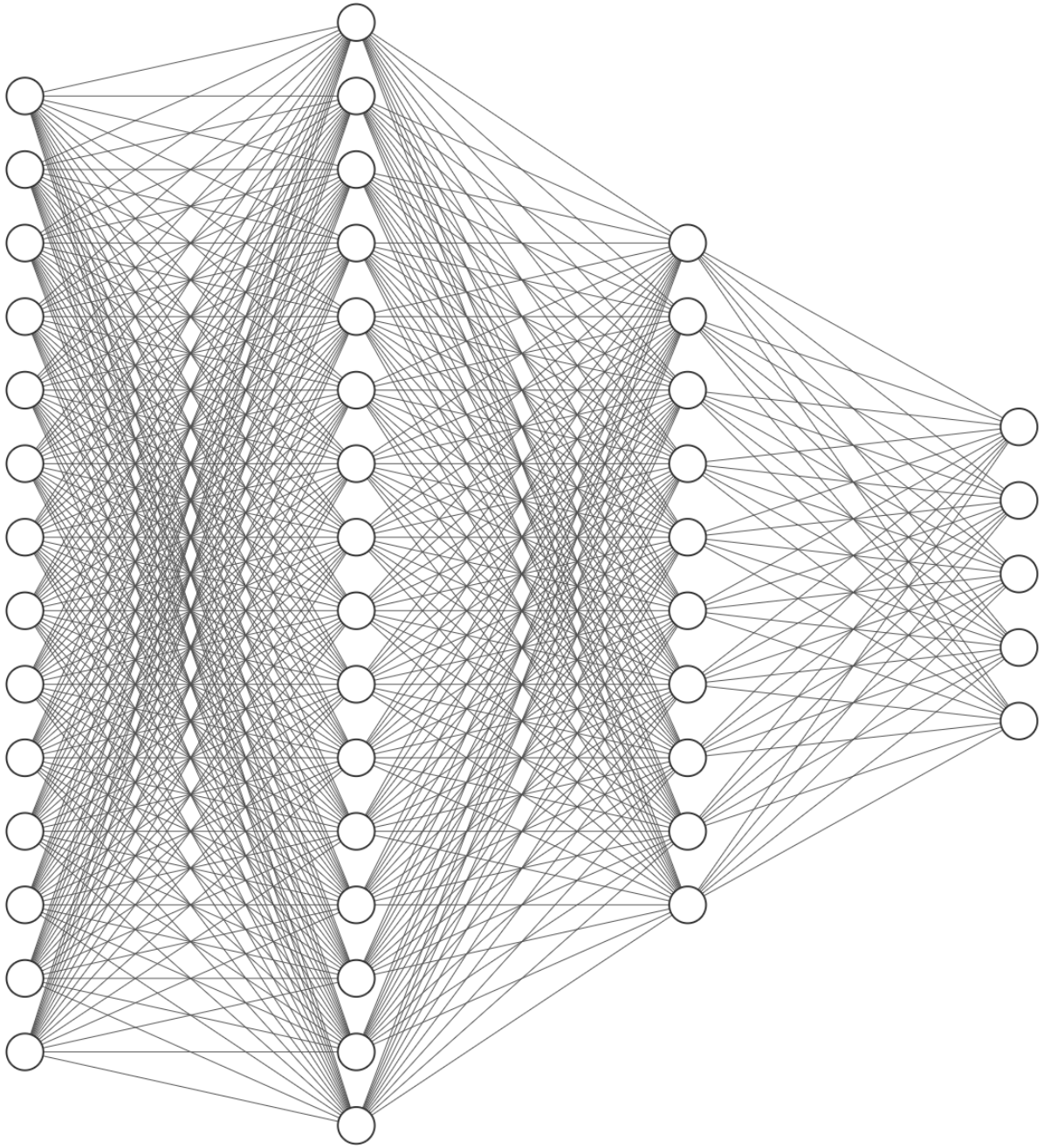
(https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning)

Beispiele – ANN



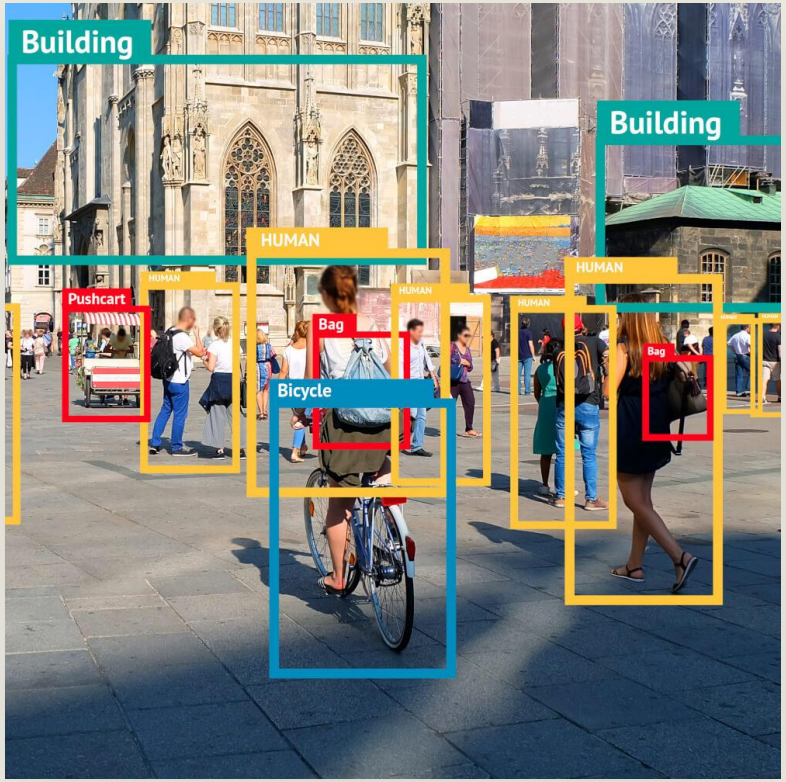
(erstellt mit https://alexlenail.me/NN-SVG/index.html)

Beispiele – DL



(erstellt mit https://alexlenail.me/NN-SVG/index.html)

Einsatzgebiete von Neuronalen Netzen (DL)



(https://www.netzpiloten.de/wie-funktioniert-automatische-bilderkennung/)

(https://www.spiegel.de)

(https://www.it-production.com/hardware-und-infrastruktur/sprechen-statt-tippen/)

(https://news.mit.edu/2022/ai-predicts-protein-docking-0201)

Wie funktionieren Neuronale Netze?*

*bei Klassifikationsproblemen

Grundfunktion

Training:

- Input: Features und Labels

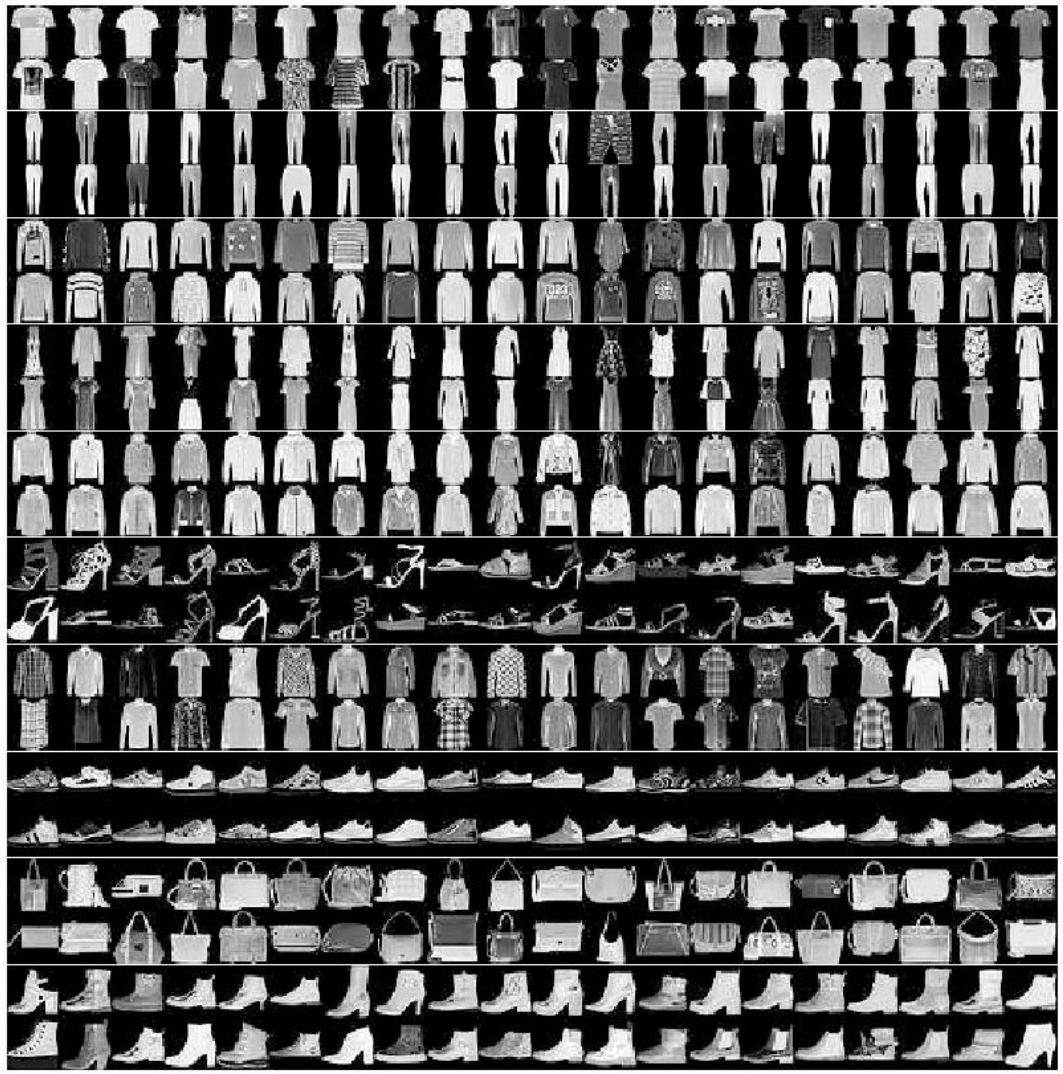
- Output: trainiertes Modell

Anwendung:

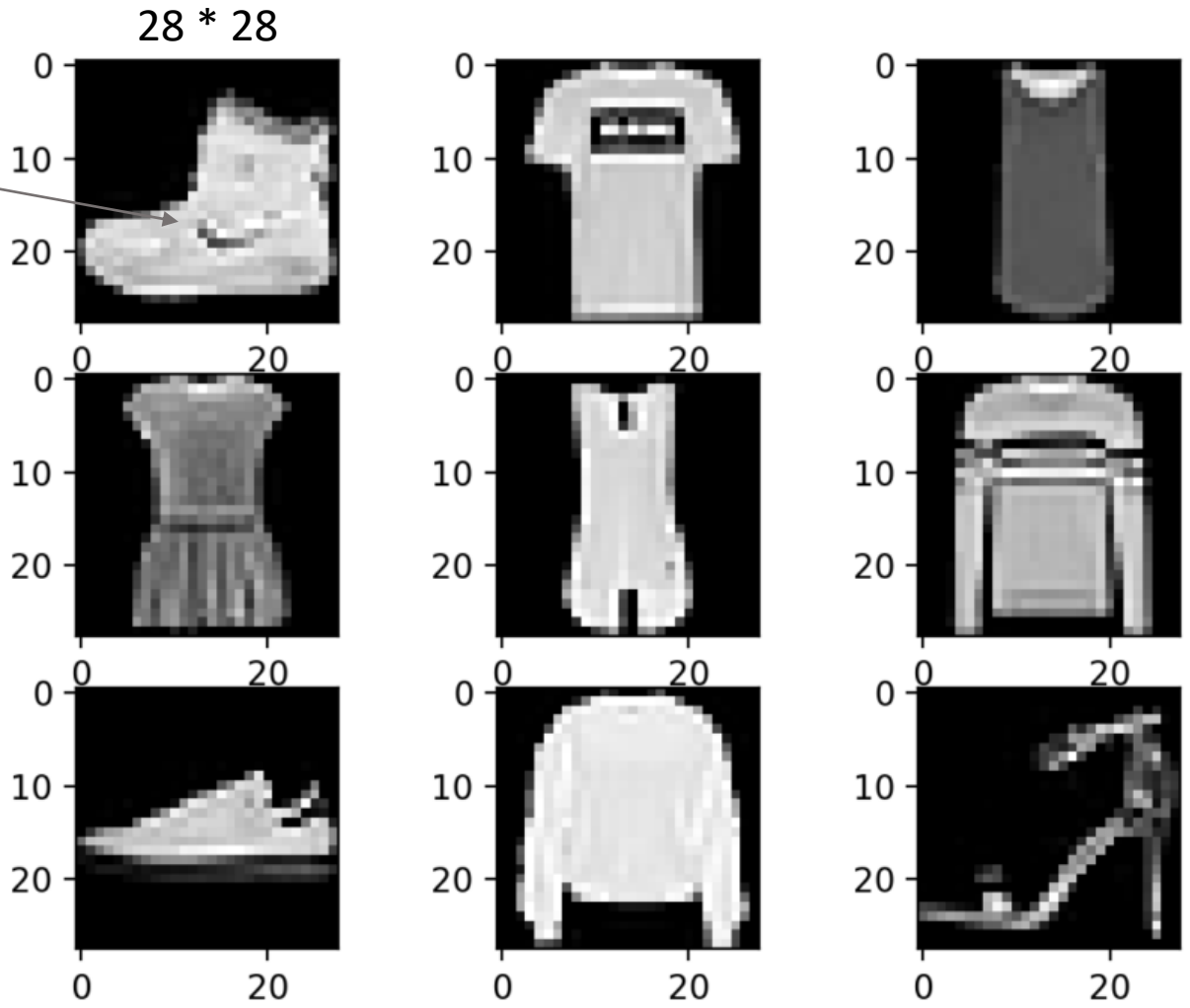
- Input: Features

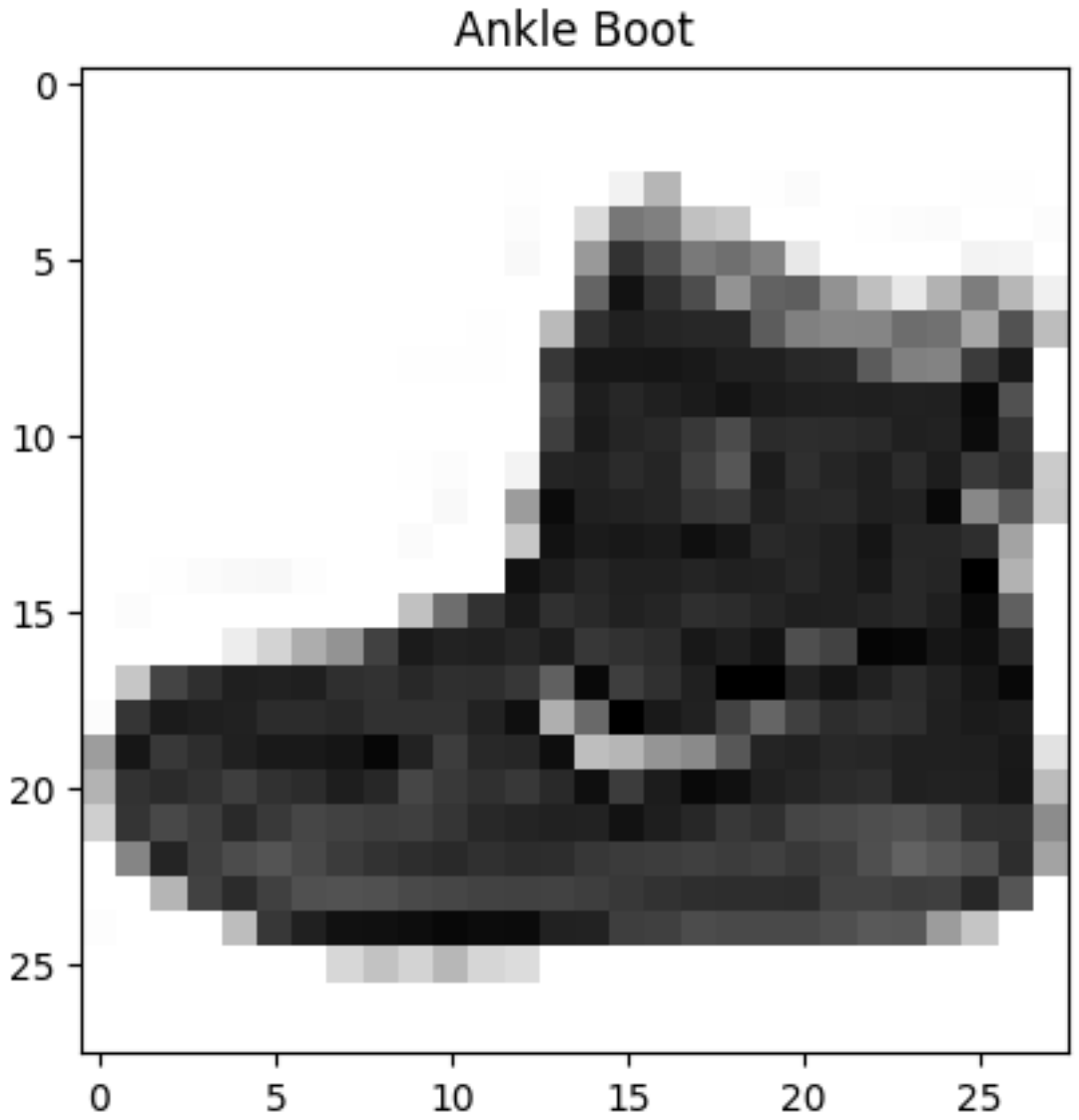
- Output: vorhergesagtes Label

Unsere Daten

Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

[137]

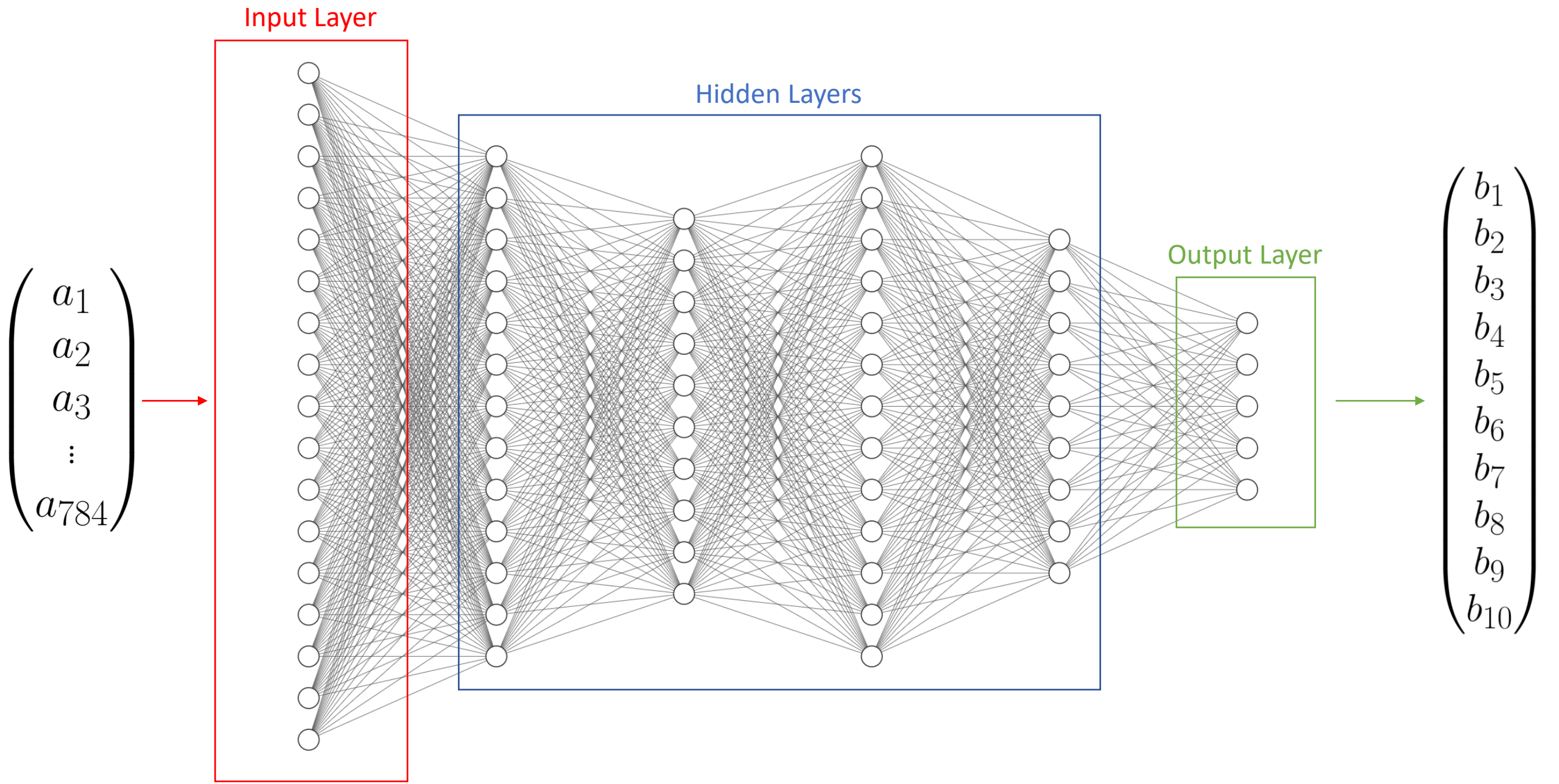




$$A_{28,28} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,28} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,28} \\ \vdots & \vdots & \ddots & \vdots \\ a_{28,1} & a_{m,2} & \cdots & a_{28,28} \end{pmatrix}$$

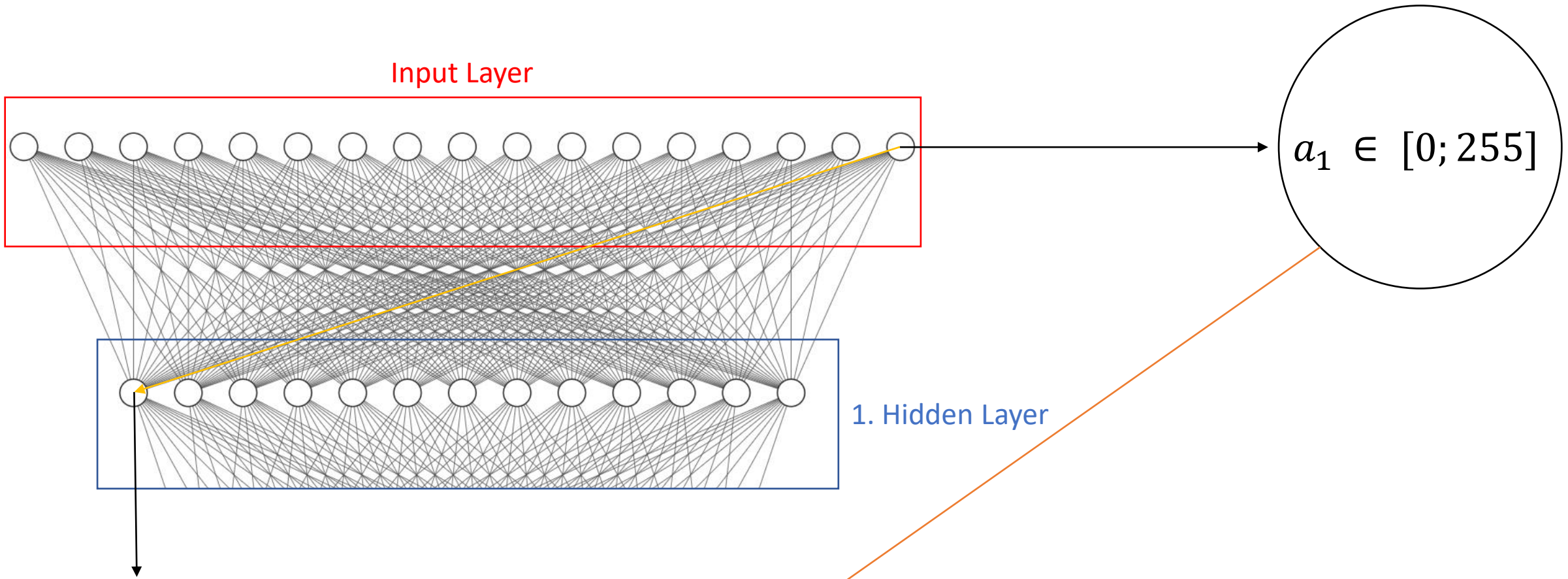
$$\vec{a}_{784} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{784} \end{pmatrix}$$

Der Aufbau



(erstellt mit https://alexlenail.me/NN-SVG/index.html)

Die Neuronen



$$h_1 = activation\_function(w_1 * a_1 + w_2 * a_2 + \cdots + w_{784} * a_{784} + bias)$$

$\in \mathbb{R}$ $\in \mathbb{R}$

(erstellt mit https://alexlenail.me/NN-SVG/index.html)

Die Aktivierungsfunktion

Bringt *Nicht-Linearität* in das Modell

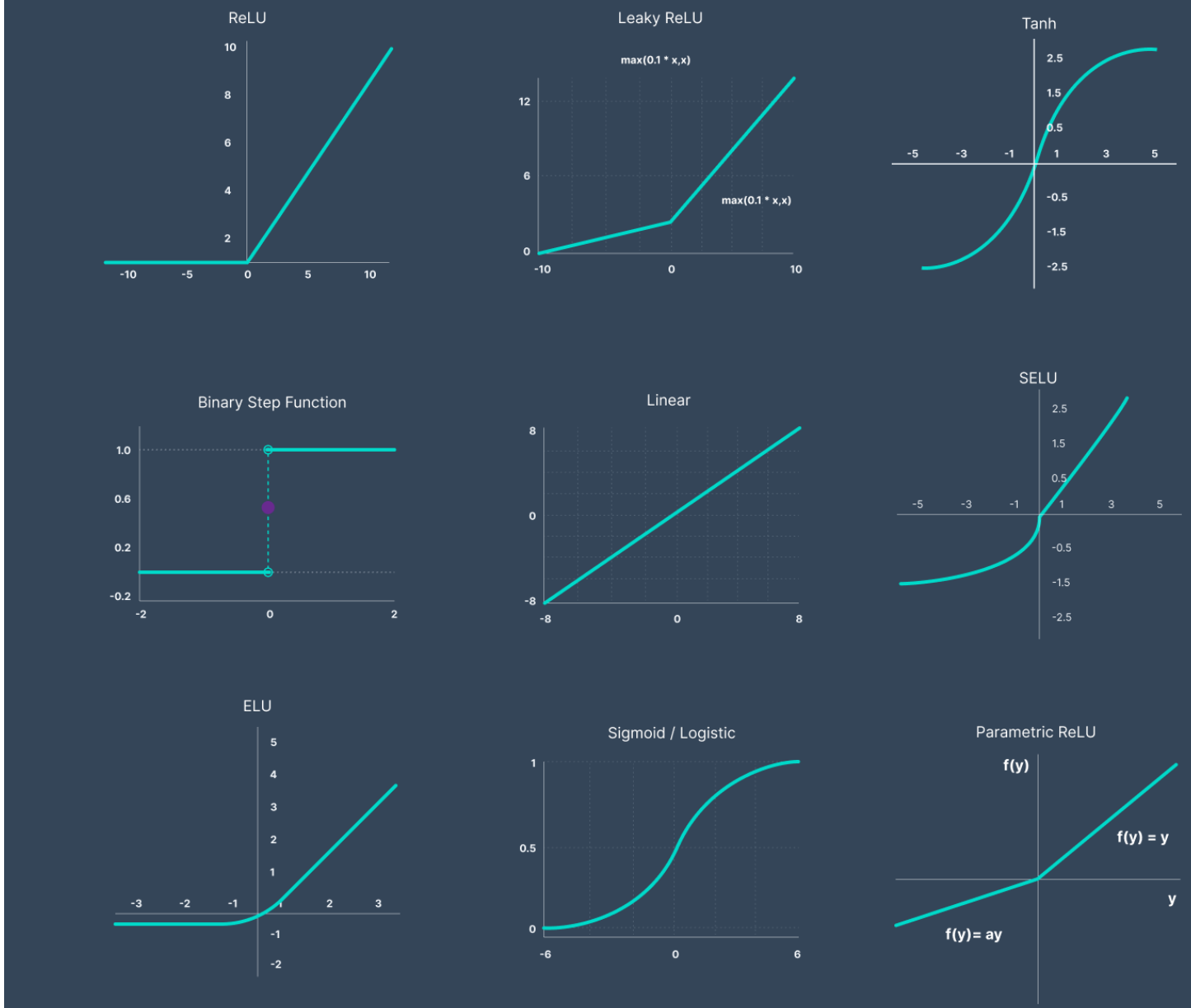
Am meisten verwendet:

- ReLU – max(0,x)
- Sigmoid

Beispiele:

$h_1 = ReLU\big((-23) + 15\big) = 0$

$h_2 = ReLU\big((56) + 24\big) = 80$

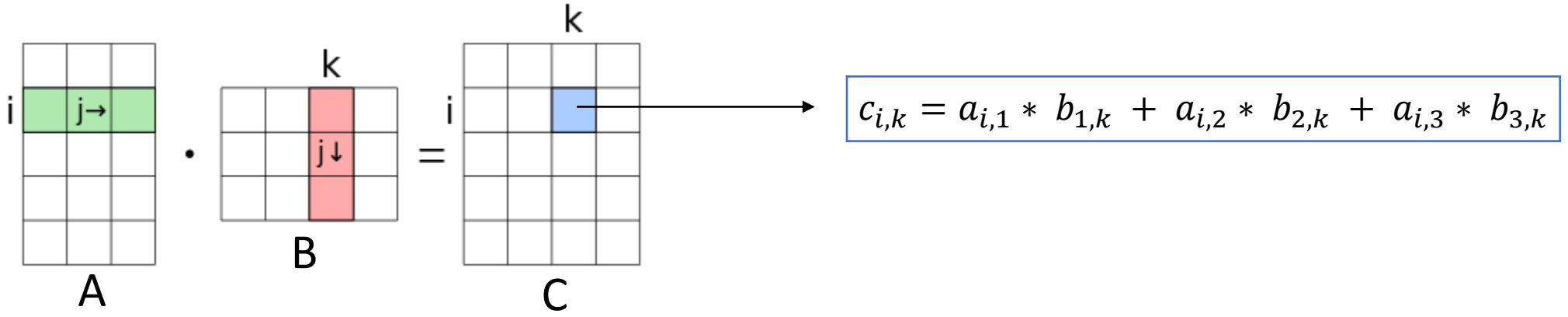


(https://www.v7labs.com/blog/neural-networks-activation-functions)

$$h_1 = ReLU(w_1 * a_1 + w_2 * a_2 + \cdots + w_{784} * a_{784} + bias)$$

$$H = ReLU(\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{pmatrix})$$

Mathematischer Exkurs: Matrizenmultiplikation und Lineare Abbildungen



$$c_{i,k} = a_{i,1} * b_{1,k} + a_{i,2} * b_{2,k} + a_{i,3} * b_{3,k}$$

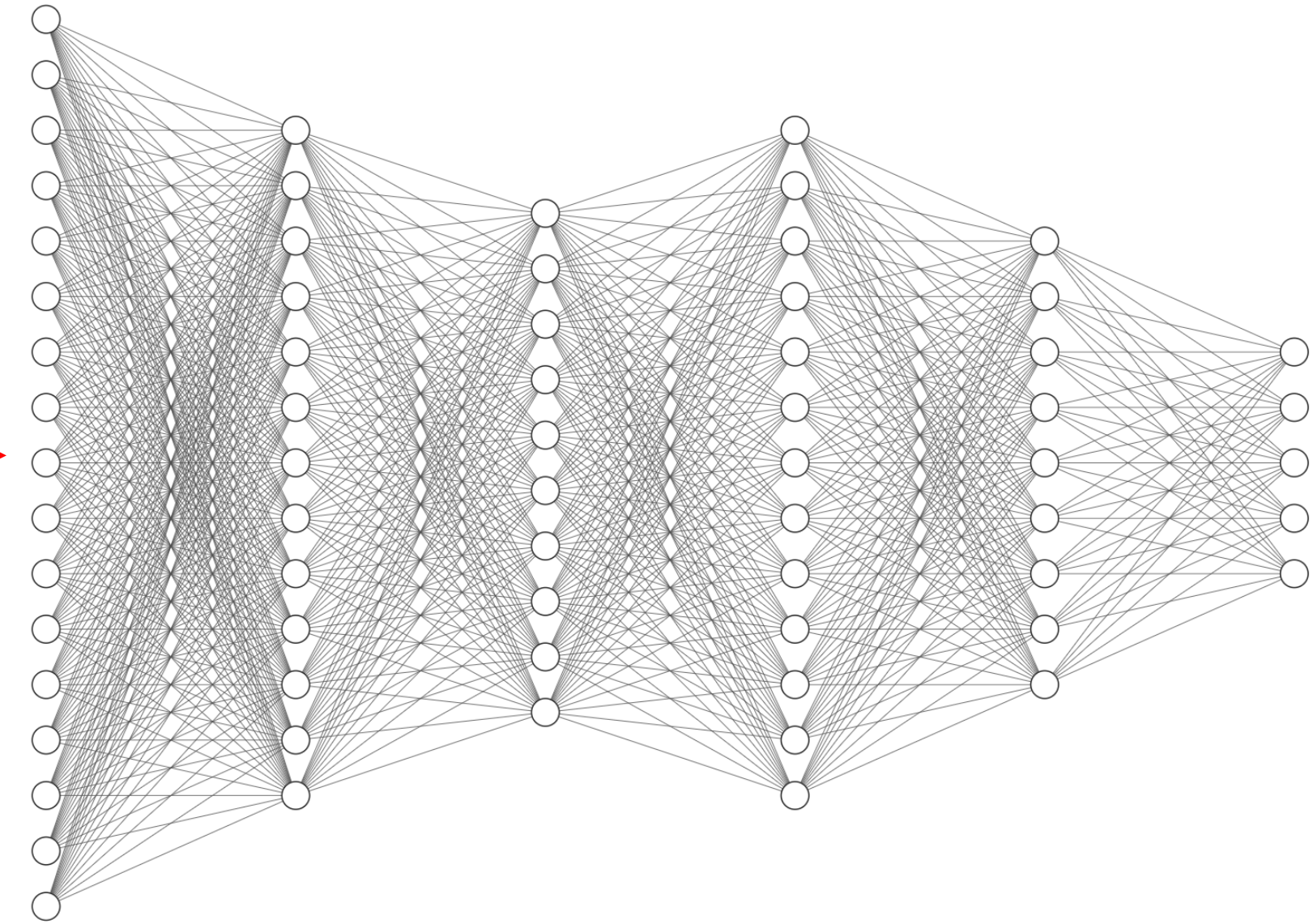
$$h_1 = ReLU(w_{1,1} * a_1 + w_{1,2} * a_2 + \cdots + w_{1,784} * a_{784} + bias)$$

$$H = ReLU\left(\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,n} \end{bmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ a_n \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \cdots \\ b_n \end{pmatrix}\right)$$

(https://en.wikipedia.org/wiki/Matrix_multiplication)

Lineare Abbildung

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{784} \end{pmatrix}$$



$$\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \\ b_8 \\ b_9 \\ b_{10} \end{pmatrix}$$

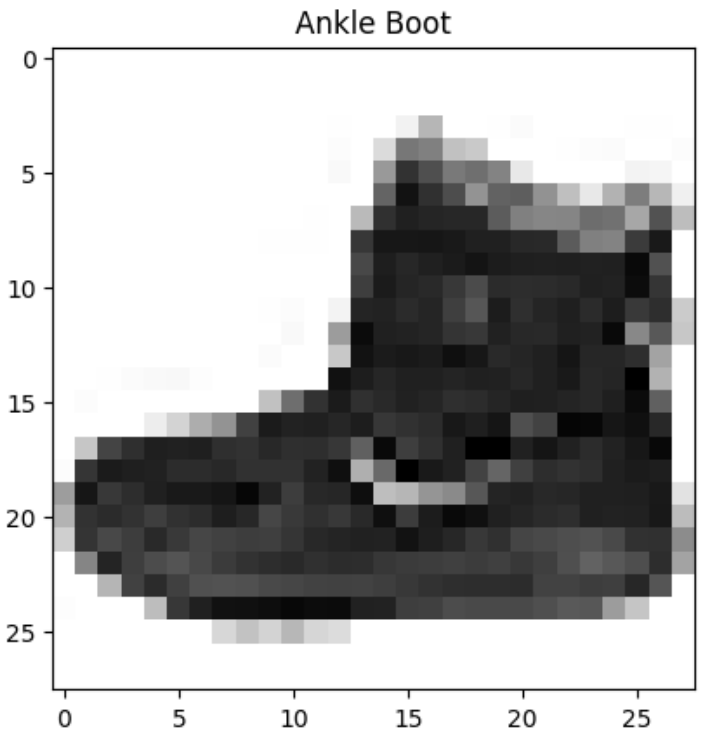
(erstellt mit https://alexlenail.me/NN-SVG/index.html)

Wie lernt das Neuronale Netz?

1. Der Loss wird über Loss Funktion berechnet
2. Der *Gradient* wird bestimmt
3. Über den Backpropagation Algorithmus werden die Parameter des Modells angepasst
4. 1 – 3 Wiederholen

⇒ (Stochastic) Gradient Descent

1. Loss



→ Modell →

y_pred

$$\begin{pmatrix} 0.5 \\ 0 \\ 0 \\ 0.05 \\ 0.05 \\ 0.3 \\ 0 \\ 0 \\ 0 \\ 0.1 \end{pmatrix}$$

Loss ≙ Differenz

= 1,155*

y_true

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

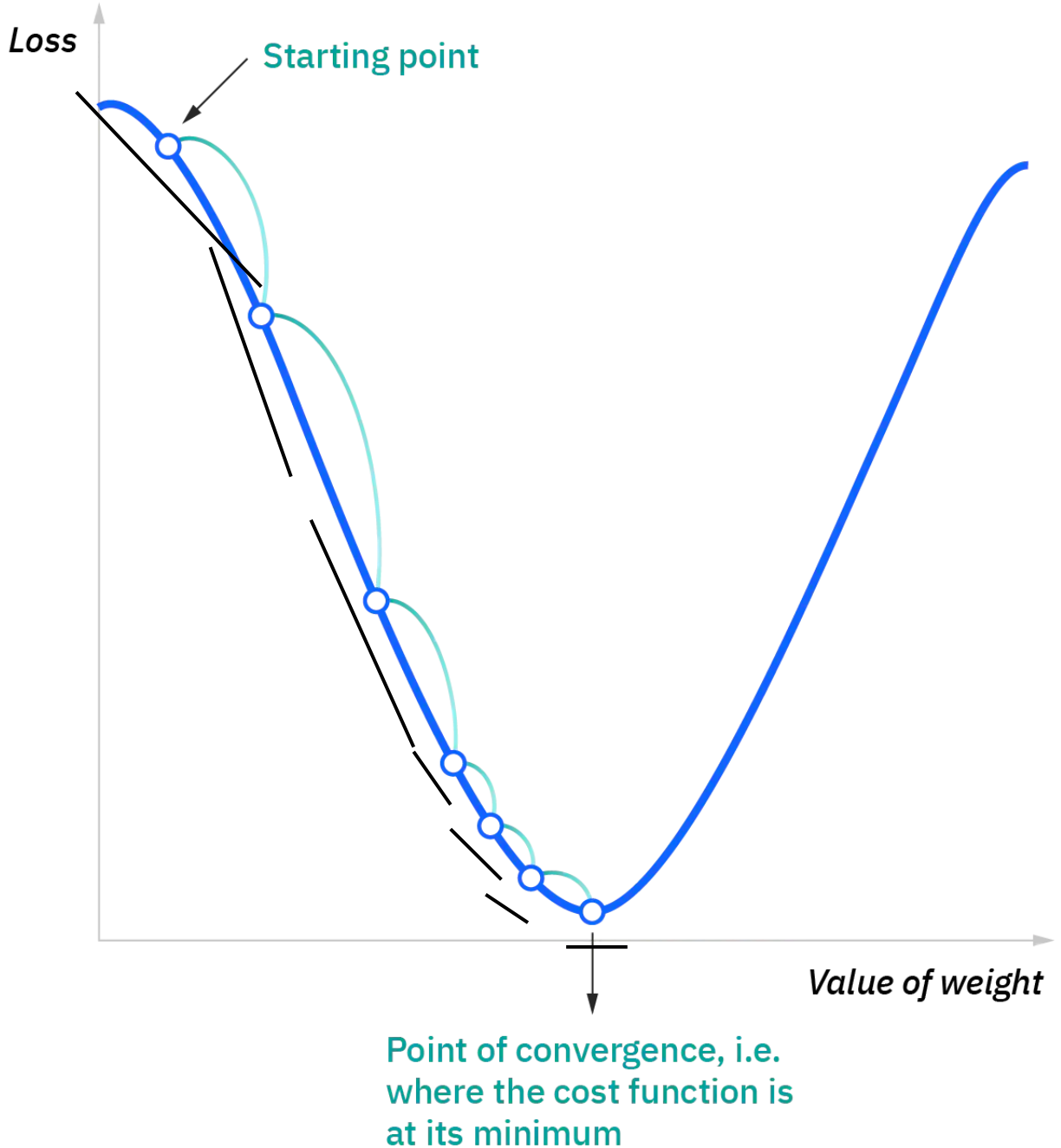
⇒ Der Durchschnitt der Losses aller Bilder ergibt den Loss für diese konkreten Parameter

⇒ Loss Funktion: C(alle Parameter) = X $\in \mathbb{R}$

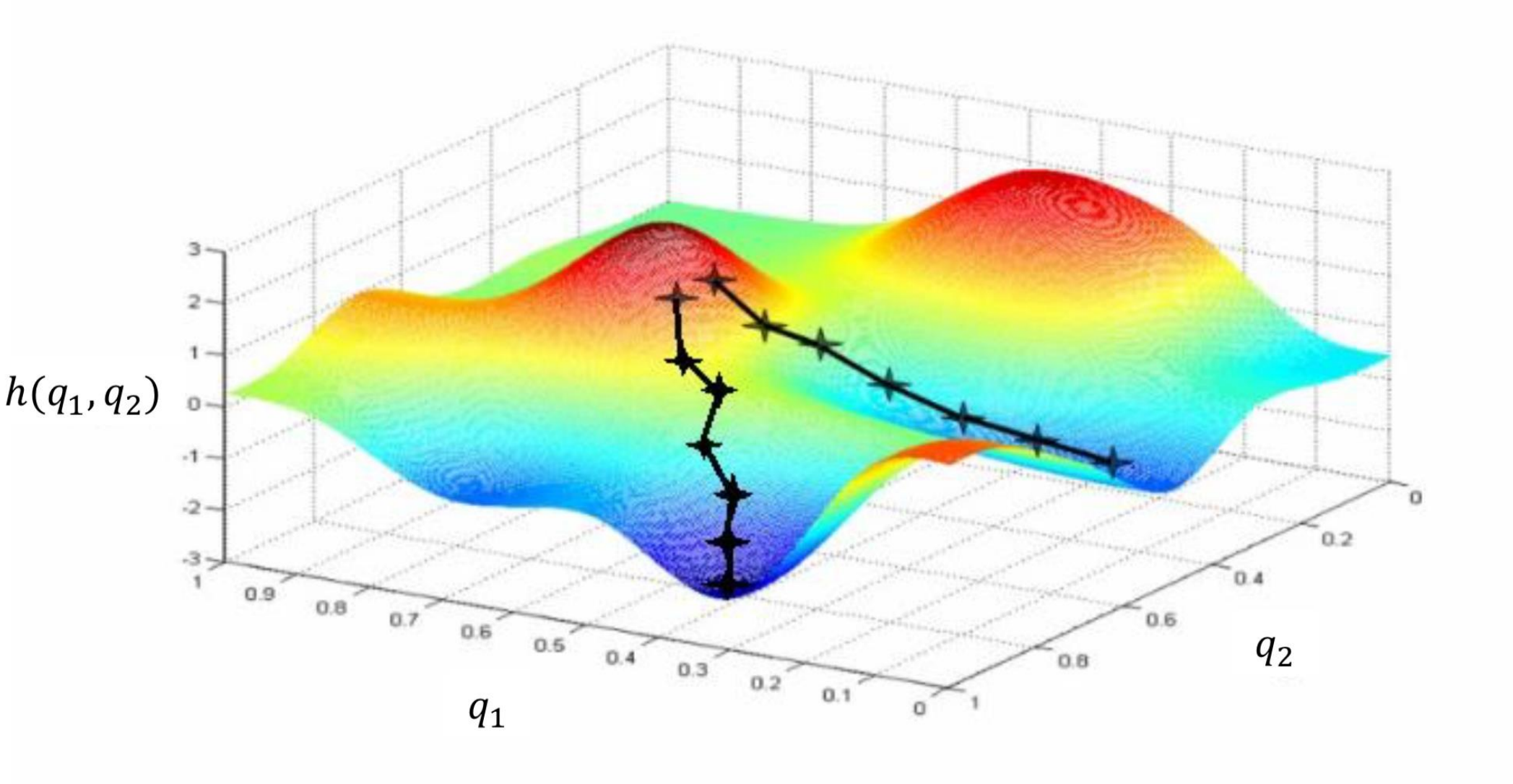
*mit RSS

2. Gradient

Der Gradient bestimmt die Richtung der größten Steigung

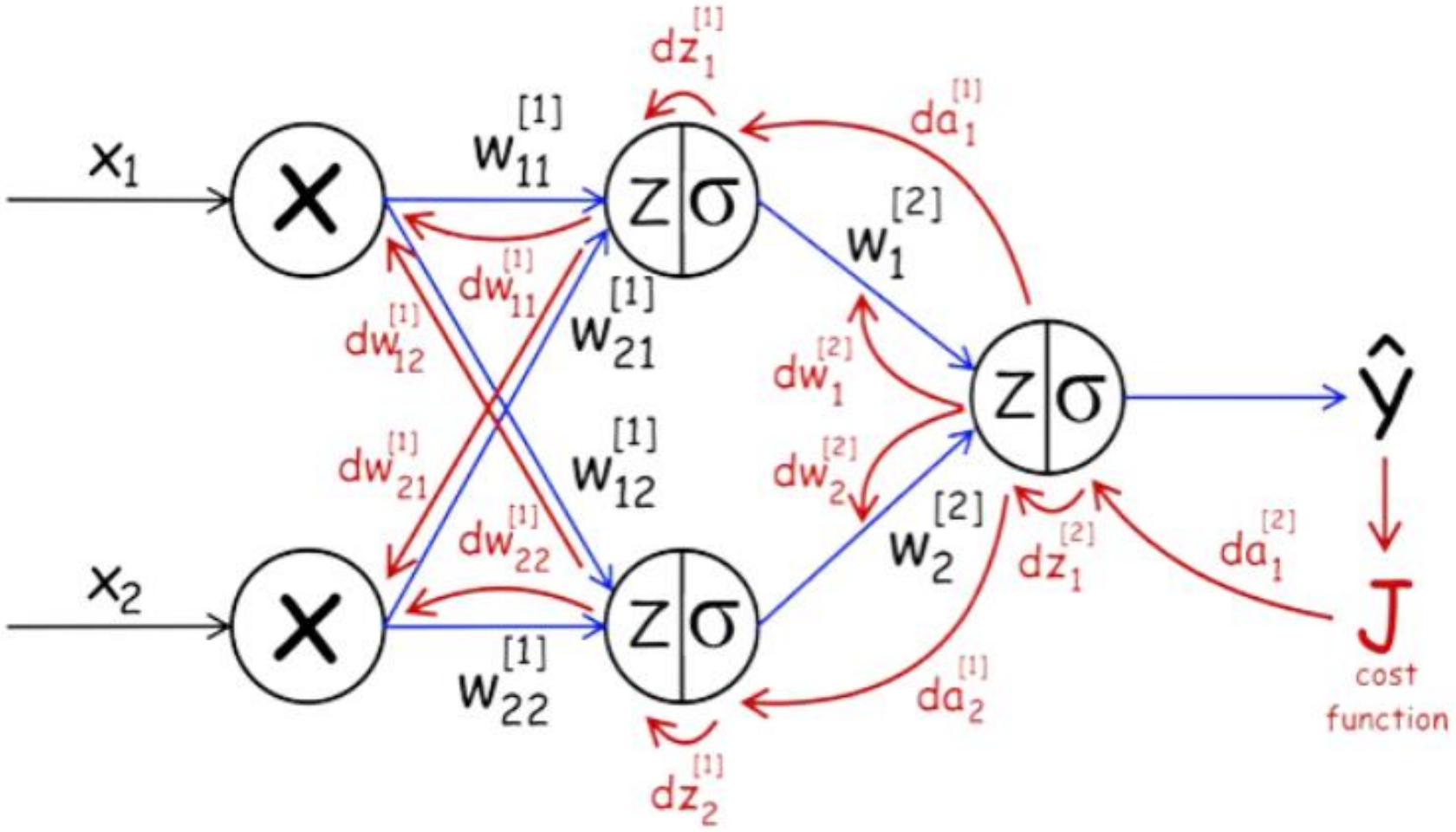


(https://www.kaggle.com/code/bhatnagardaksh/gradient-descent-from-scratch)



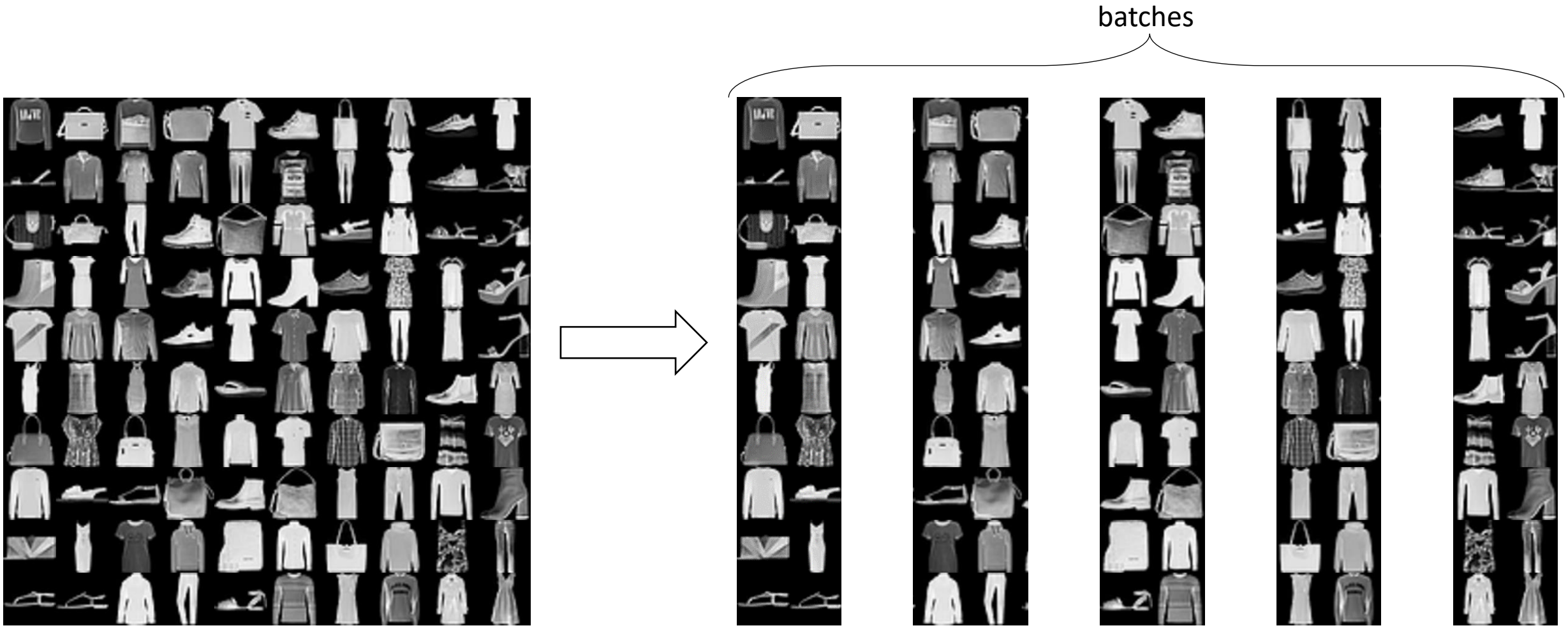
(https://shashank-ojha.github.io/ParallelGradientDescent/)

3. Backpropagation



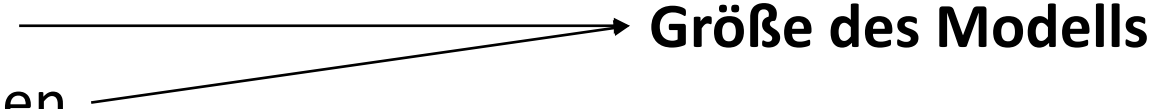
(https://analyticsarora.com/8-unique-machine-learning-interview-questions-on-backpropagation/)

Stochastic Gradient Descent



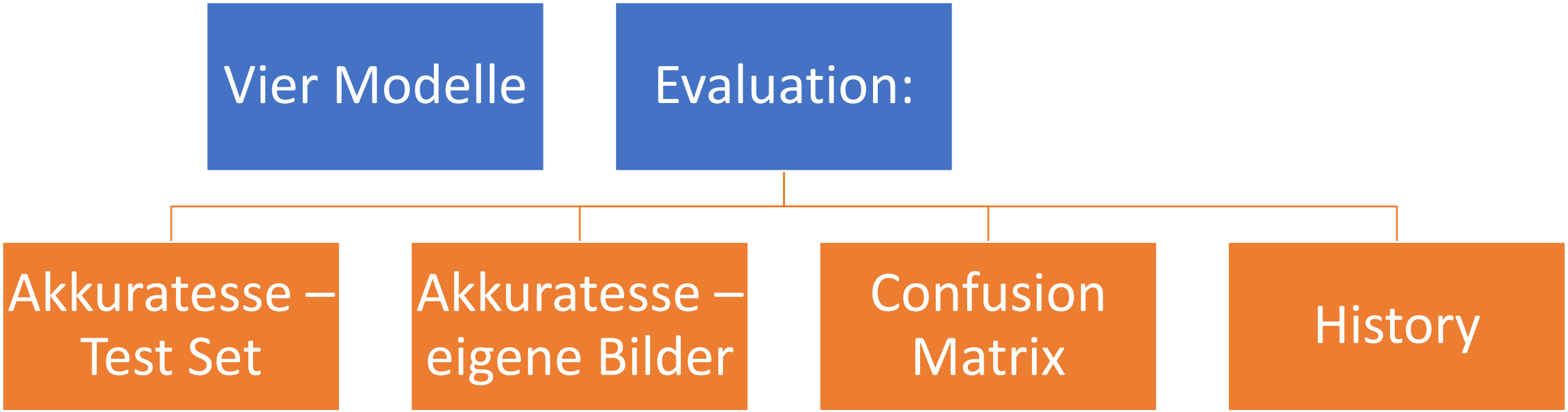
Hyperparameter – Stellschrauben des Modells

- Modell:
  - Anzahl Layers
  - Anzahl Neuronen
  - Aktivierungsfunktionen



- Training:
  - **Anzahl Epochs (Durchläufe)**
  - Optimizer **–> Learning Rate**
  - Loss Funktion
  - **Daten –> Menge, Diversität**
  - Größe der Batches (batch size)

Konkrete Lösungen

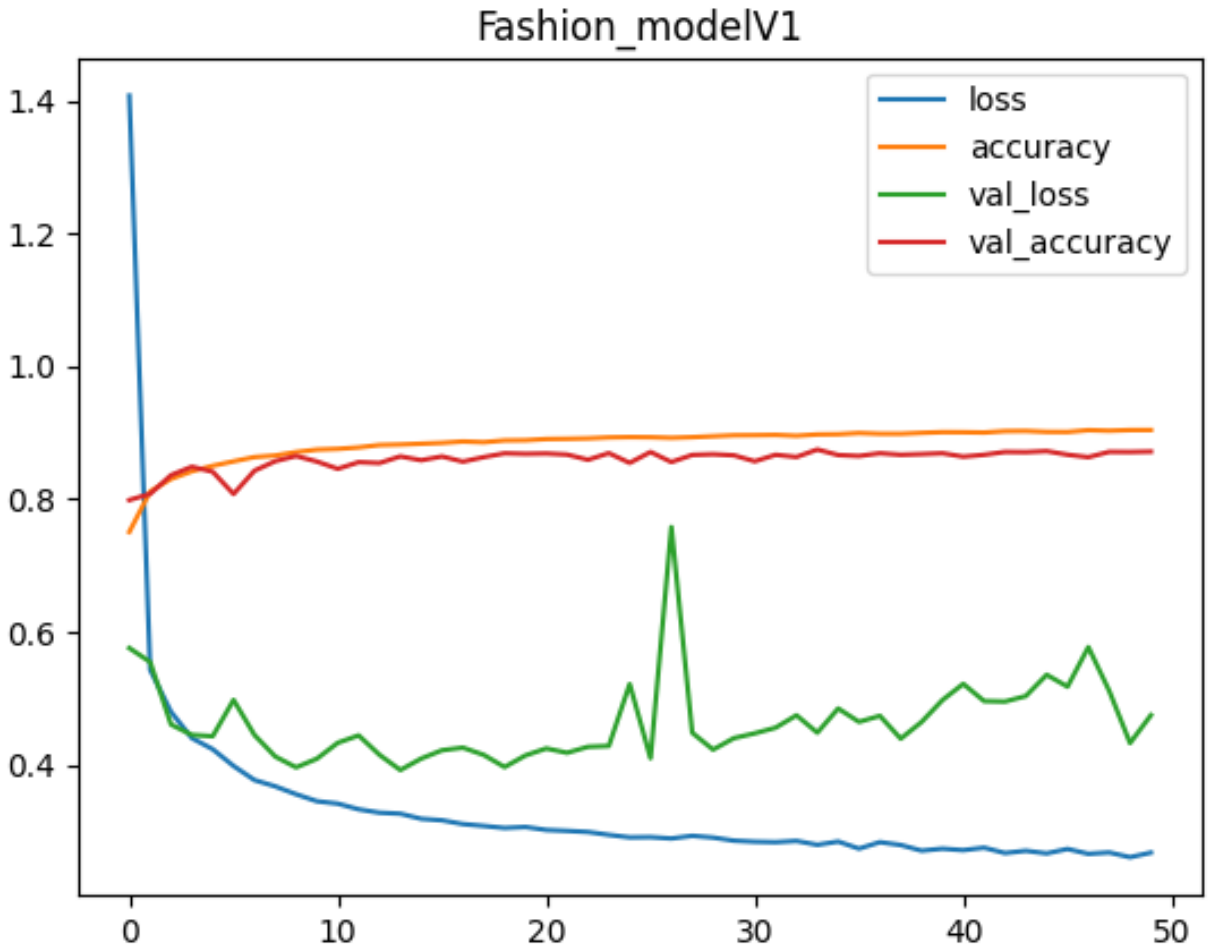


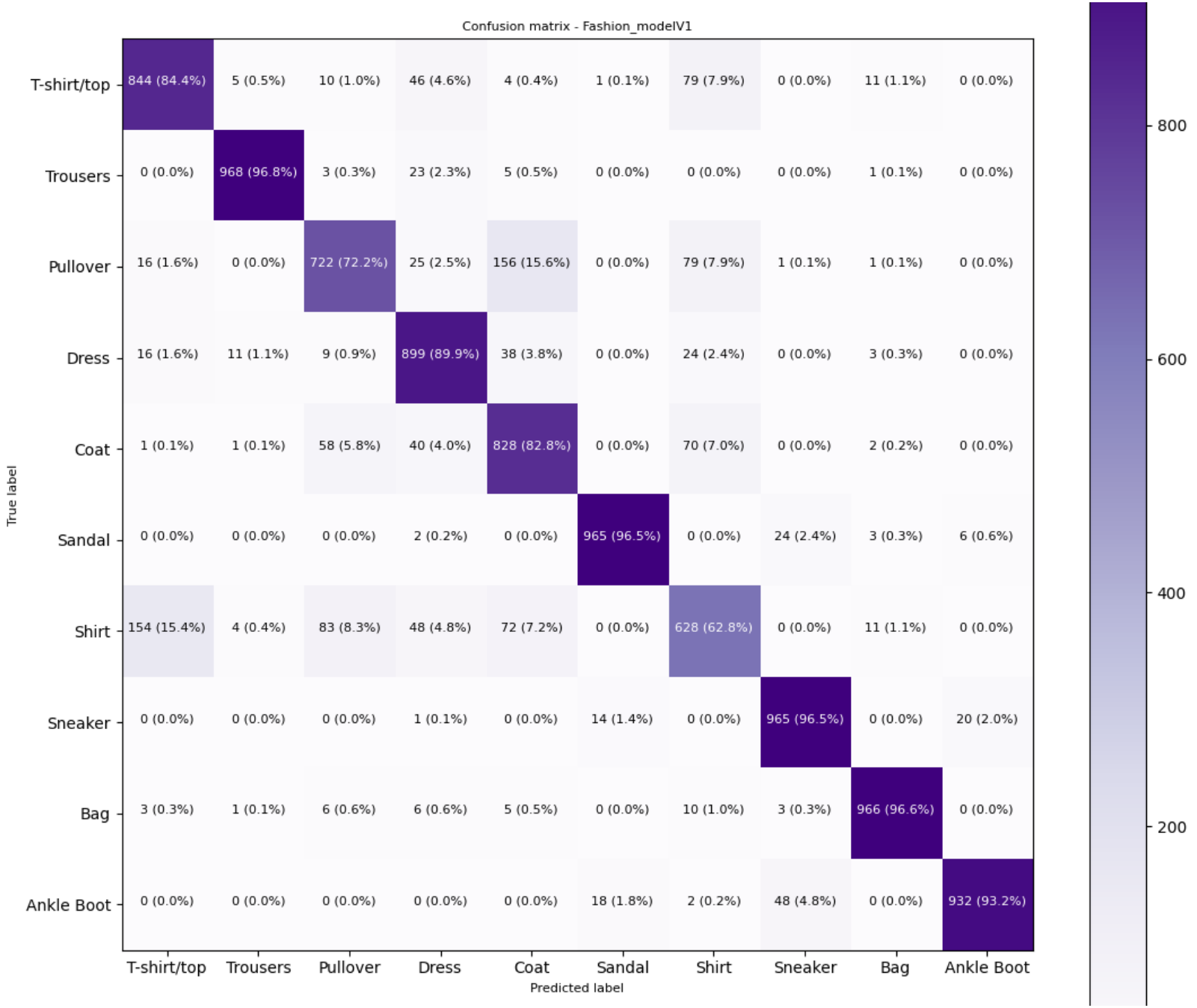
Modell 1

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(100, activation="relu"),
    tf.keras.layers.Dense(100, activation="relu"),
    tf.keras.layers.Dense(50, activation="relu"),
    tf.keras.layers.Dense(10, activation="softmax")
])
model.compile(loss="sparse_categorical_crossentropy",
              optimizer=tf.keras.optimizers.Adam(),
              metrics=["accuracy"])
history = model.fit(train_data, train_labels,
                    epochs=50,
                    validation_data=(test_data, test_labels),
                    )
```

Evaluation

- Test Set : 87,17% akkurat
- Eigene Bilder : 55,5% akkurat



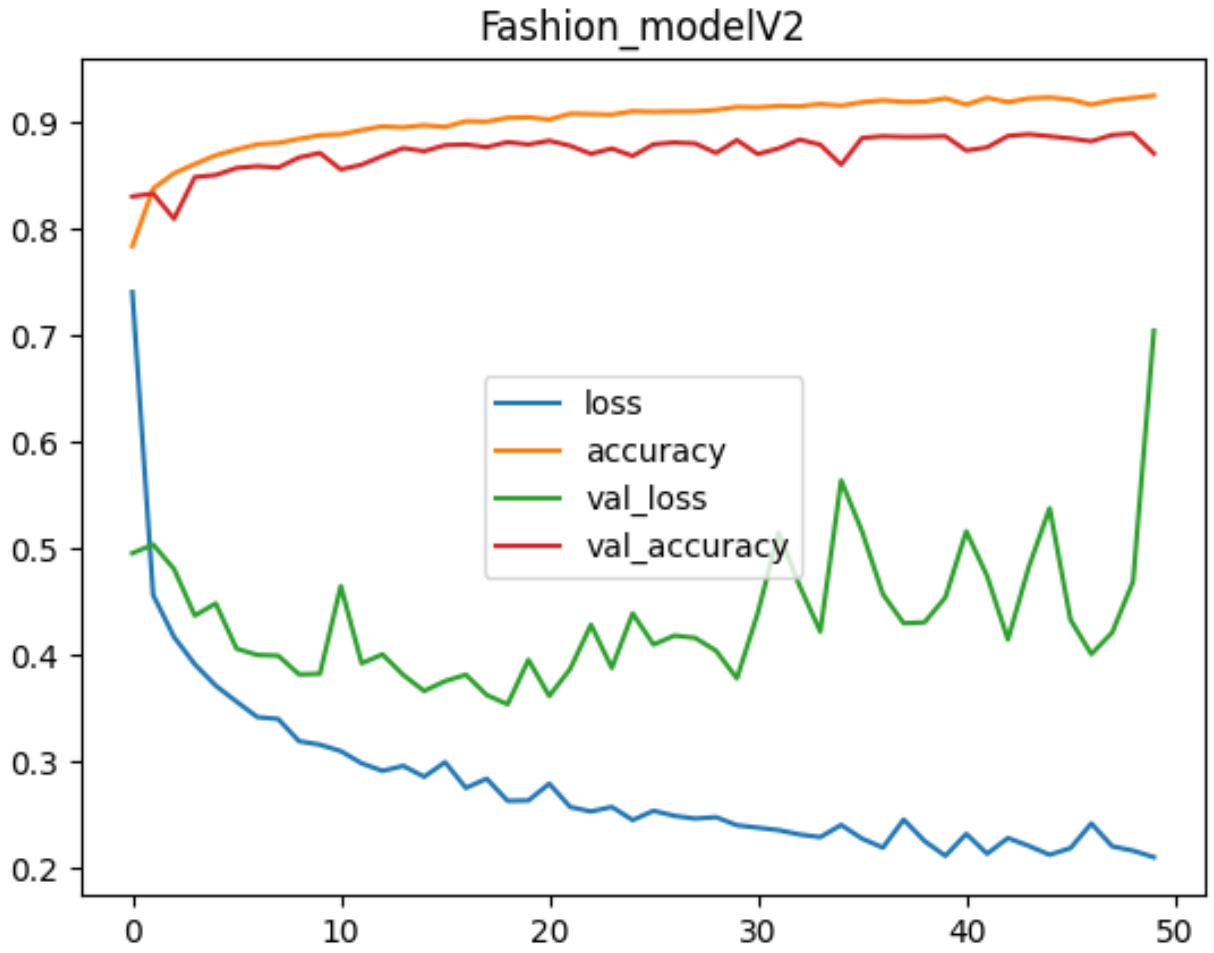


Modell 2

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28,28)),
    tf.keras.layers.Dense(250, activation="relu"),
    tf.keras.layers.Dense(150, activation="relu"),
    tf.keras.layers.Dense(250, activation="relu"),
    tf.keras.layers.Dense(150, activation="relu"),
    tf.keras.layers.Dense(300, activation="relu"),
    tf.keras.layers.Dense(10, activation="softmax")
])
model.compile(loss="sparse_categorical_crossentropy",
              optimizer=tf.keras.optimizers.Adam(),
              metrics=["accuracy"])
bigger_history = model.fit(train_data,train_labels,
                           epochs=50,
                           validation_data=(test_data, test_labels))
```

Evaluation

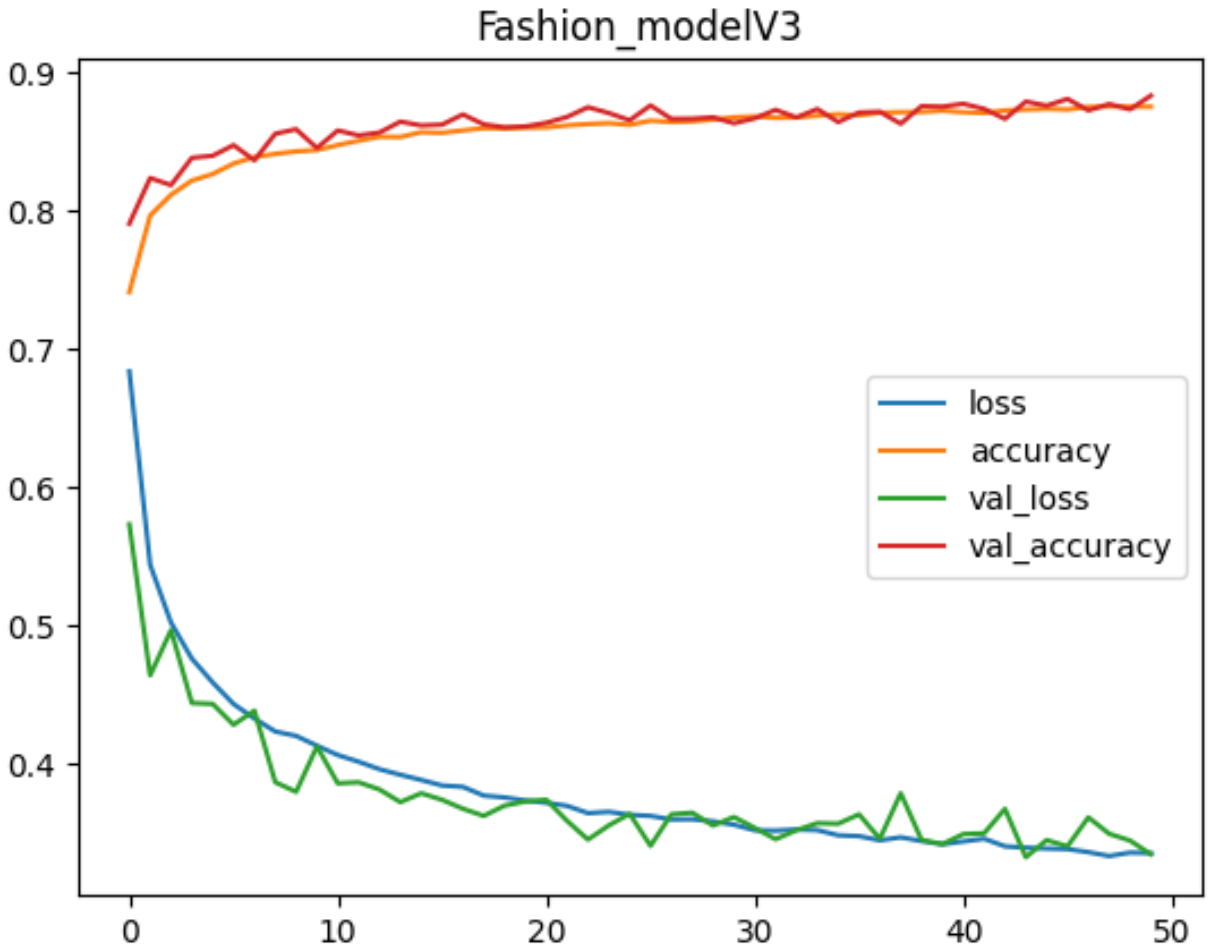
- Test Set : 88,9% akkurat
- Eigene Bilder : 61,9% akkurat

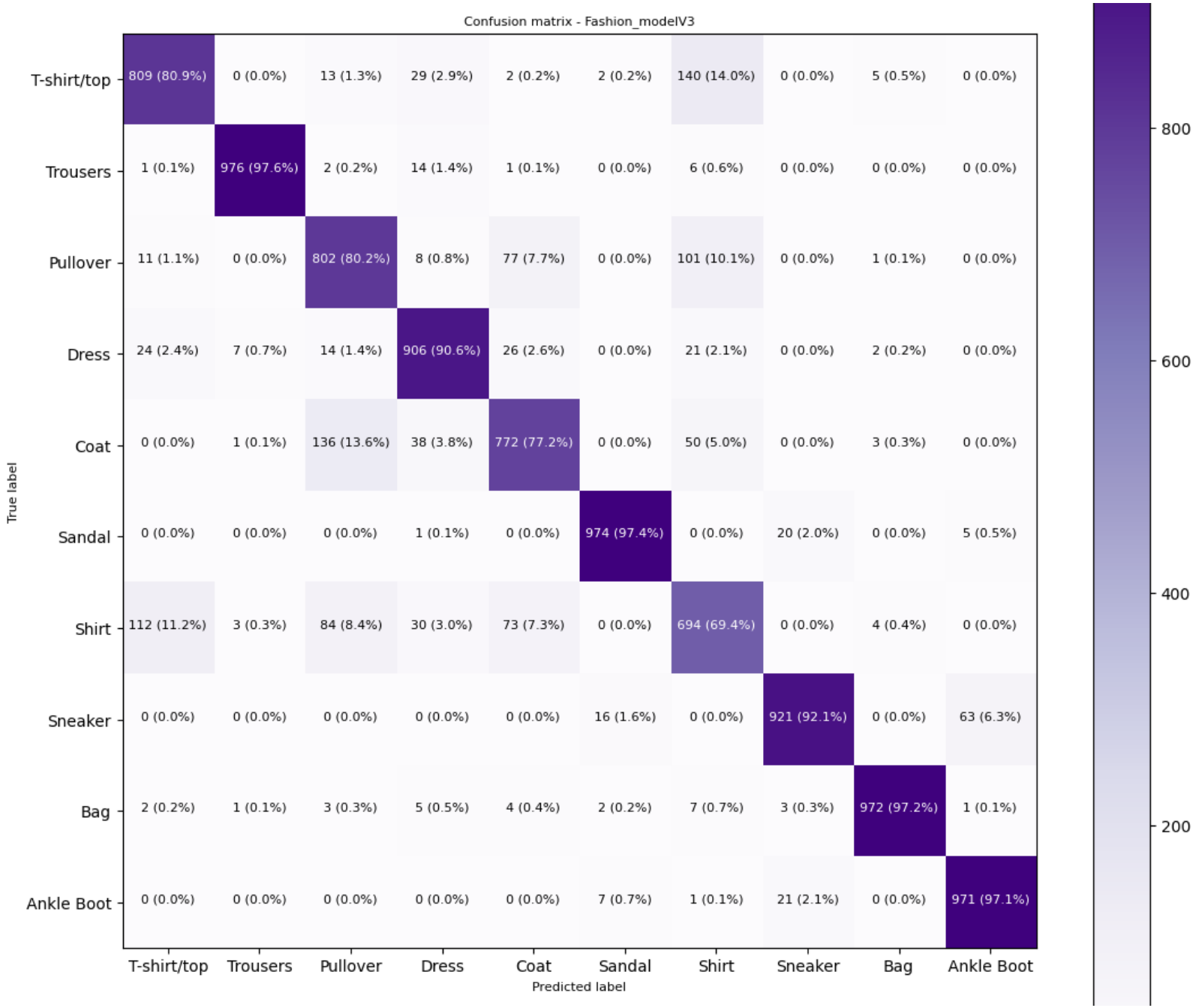


Modell 3

Evaluation

- Test Set : 88,35% akkurat
- Eigene Bilder : 72% akkurat

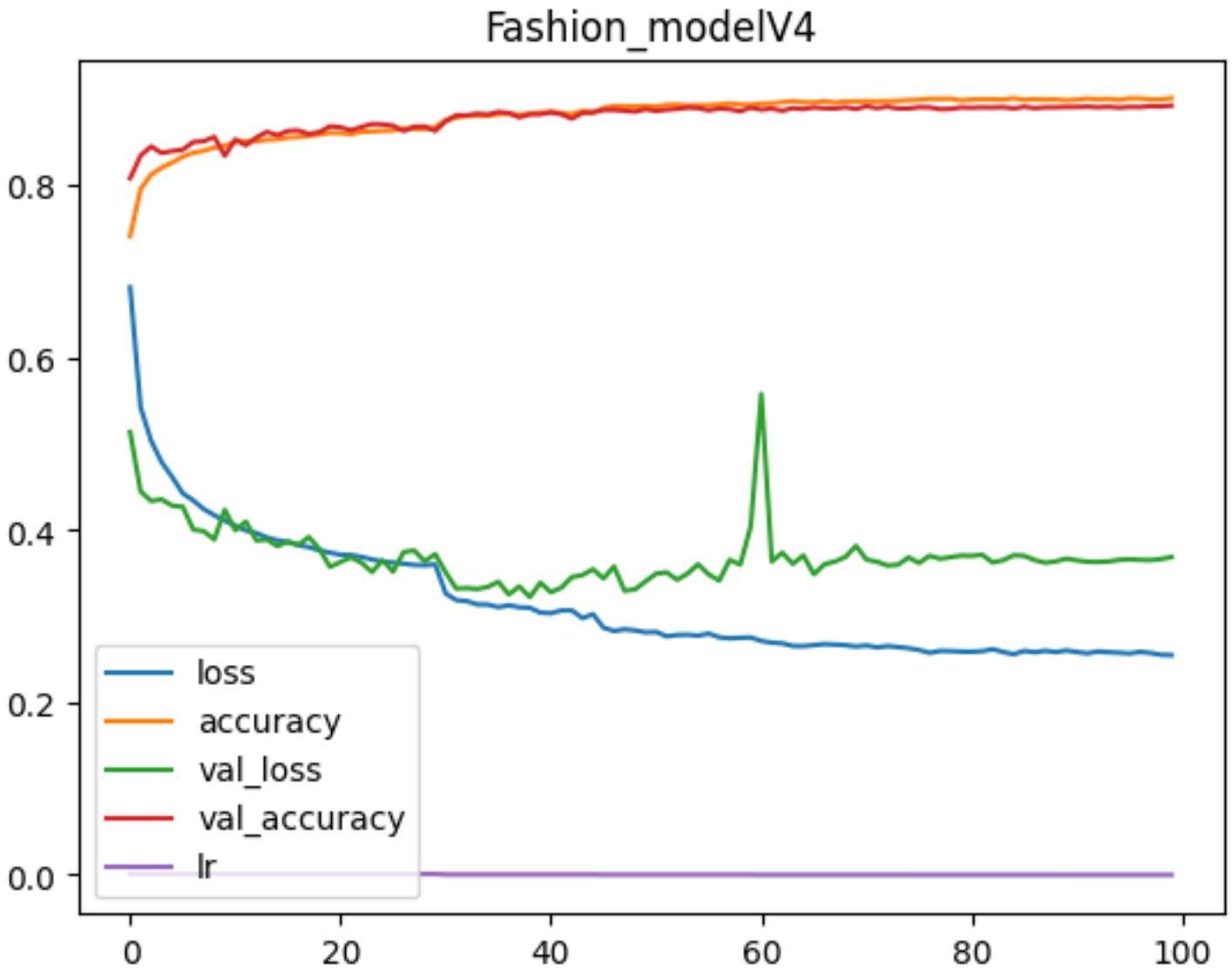


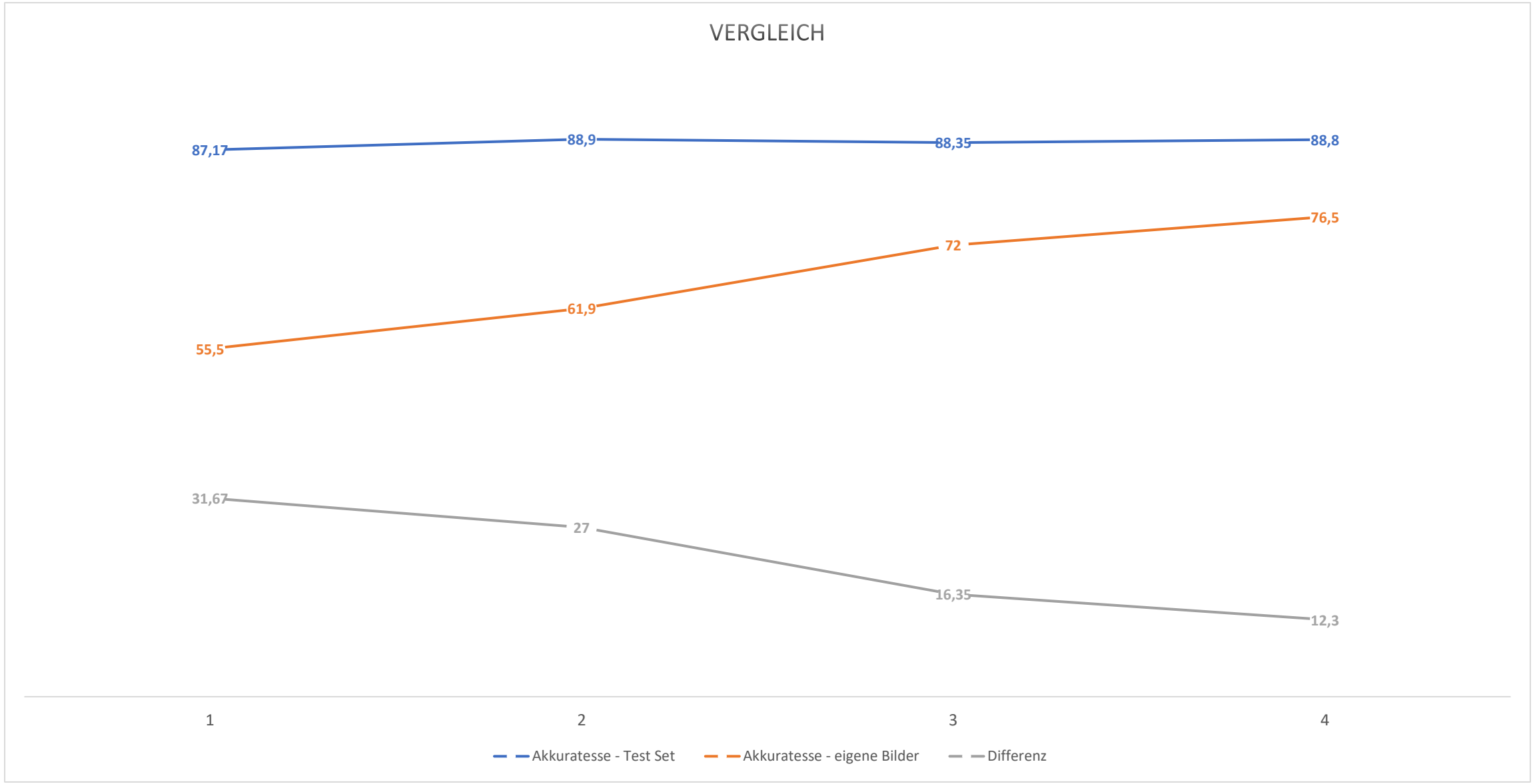


Modell 4

Evaluation

- Test Set : 88,8% akkurat
- Eigene Bilder : 76,5% akkurat





Quellen

- Grant Sanderson - 3Blue1Brown, **Neural Networks – The basics of neural networks, and the math behind how they learn**, 05.11.2017, https://www.3blue1brown.com/lessons/neural-networks (abgerufen am 11.05.2023)
- Daniel Bourke, **Learn TensorFlow and Deep Learning fundamentals with Python (code-first introduction) Part 1/2**, 16.03.2021, https://www.youtube.com/watch?v=tpCFfeUEGs8&t=19131s (abgerufen am 13.04.2023)
- Daniel Bourke, **Learn TensorFlow and Deep Learning fundamentals with Python (code-first introduction) Part 2/2**, 17.03.2021, https://www.youtube.com/watch?v=ZUKz4125WNI&list=RDLVtpCFfeUEGs8&index=2 (abgerufen am 16.04.2023)

Quellen

- Pragati Baheti, **Activation Functions in Neural Networks [12 Types & Use Cases]**, 27.05.2021, https://www.v7labs.com/blog/neural-networks-activation-functions (abgerufen am 16.04.2023)
- Daksh Bhatnagar, **Gradient Descent from scratch**, 13.11.2022, https://www.kaggle.com/code/bhatnagardaksh/gradient-descent-from-scratch (abgerufen am 20.05.2023)