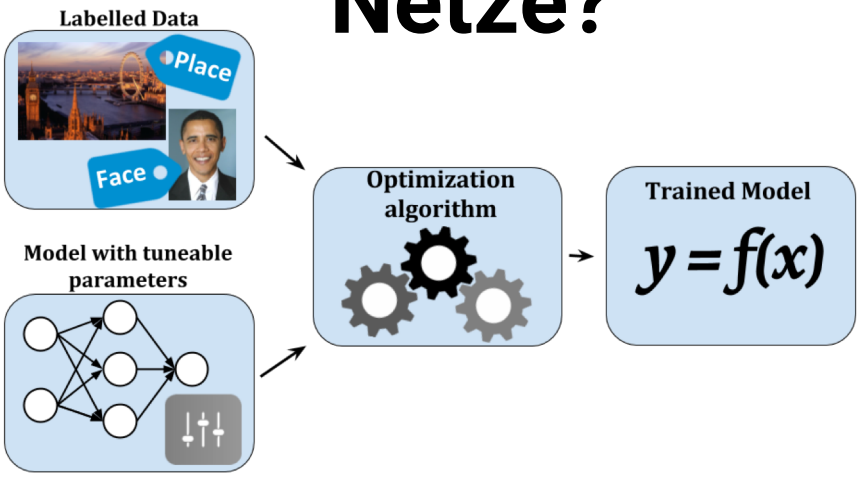
BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

# Machine Learning
# Neural Networks

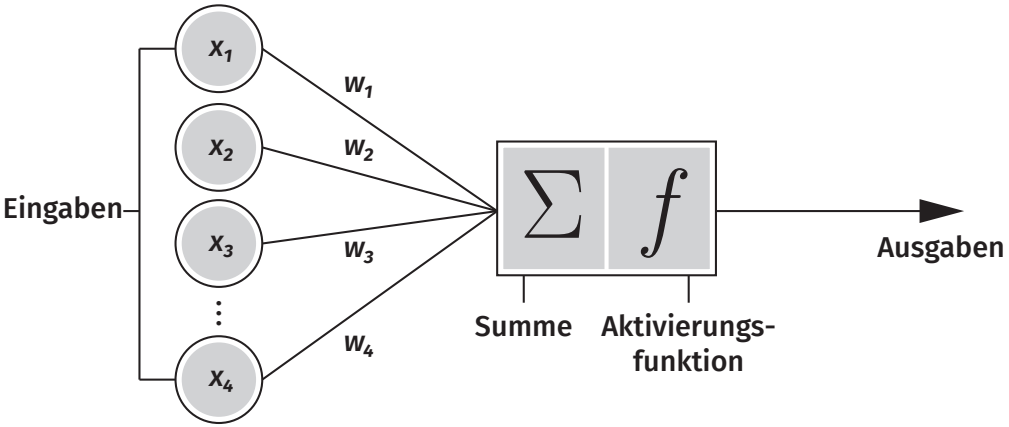
Bachelor Medieninformatik
Wintersemester 2019/20

Prof. Dr.-Ing. Kristian Hildebrand
khildebrand@beuth-hochschule.de

# Was sind Neuronale Netze?

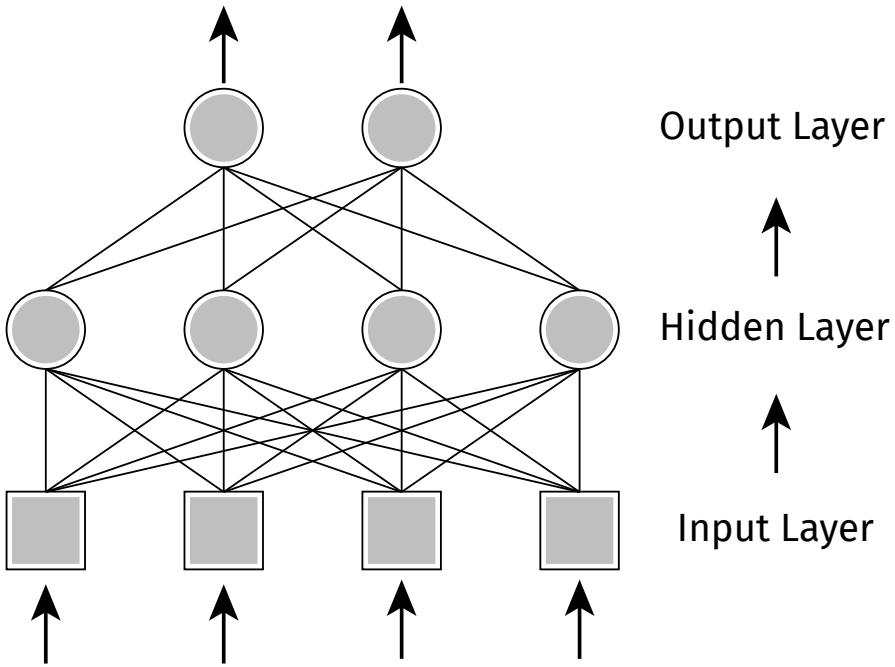


# Was sind Neuronale Netze?




**Ein Neuron**

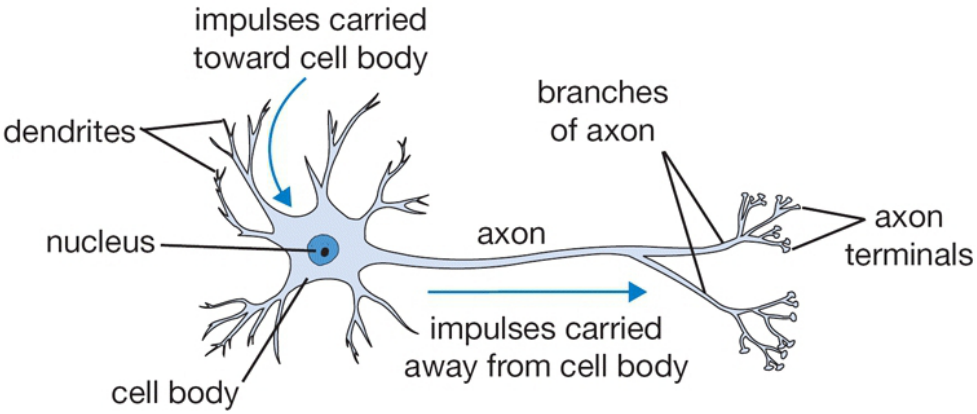
2-Layer-Netzwerk
Hausaufgabe



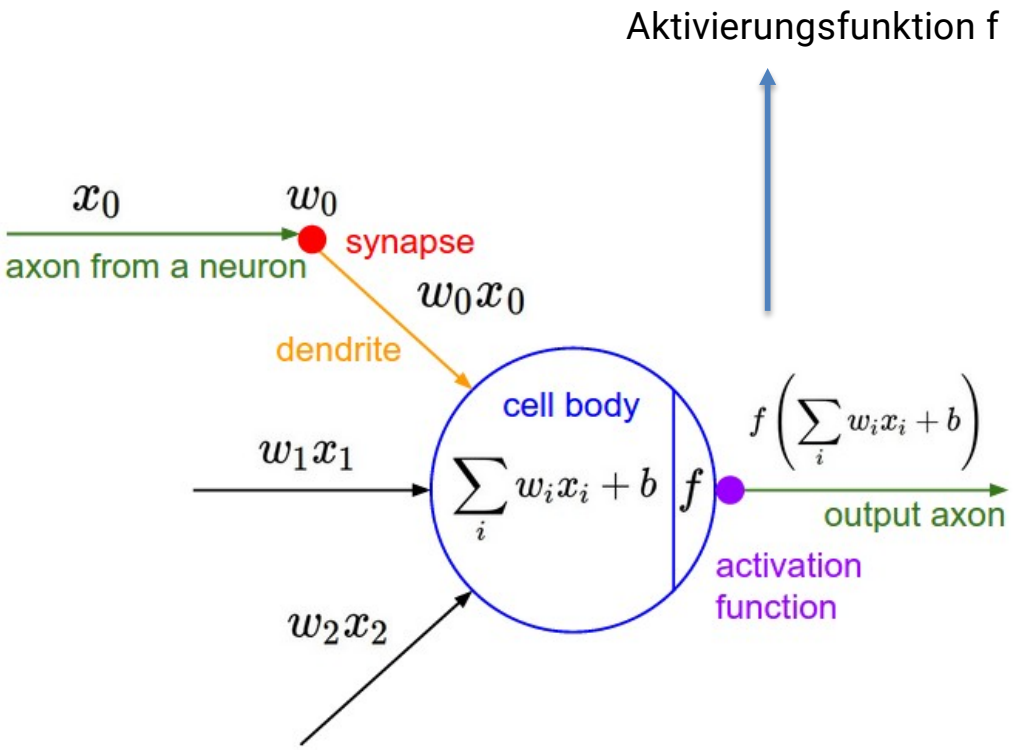

**Beliebig komplexes Netz**

**Hinweis:** Hidden Layer (weil nicht sichtbar aka Blackbox)

# Was sind Neuronale Netze?



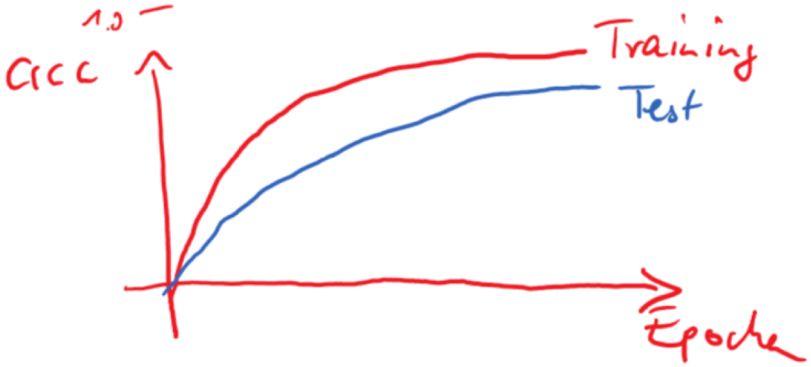

**Biologisches Neuron**

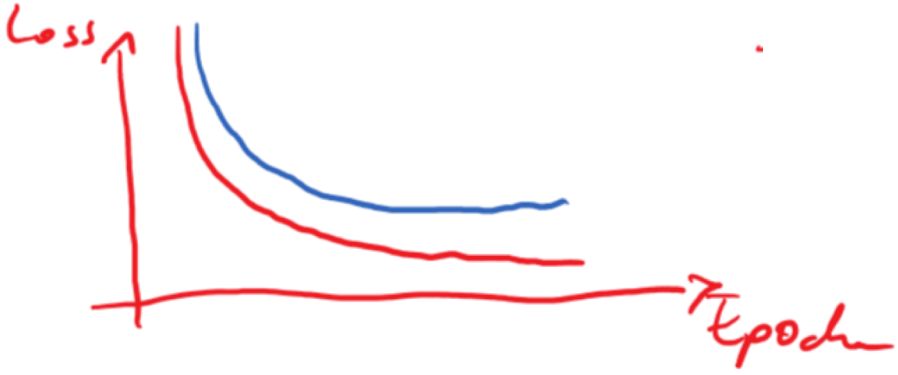



**Mathematisches Neuron**

# Fragen

- Was sind Trainings- bzw. Testdaten?
- Batchsize
  - Größe eines Trainingsbatches für einen Forward/Backward Pass
- Epoche
  - Ein Forward / Backward Pass für alle Trainingsdaten
- Genauigkeit / Fehlerrate / Lernrate



# Fragen

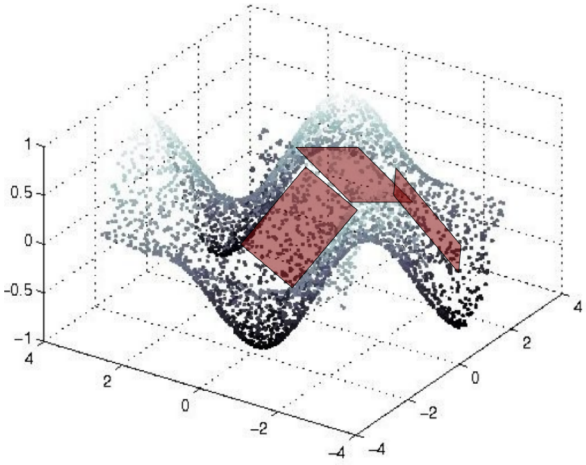
- Warum brauchen wir mehrere Layer?
  - Hierarchische Repräsentation
- Was machen die Hidden Layer?
  - Merkmalsdetektor
- Wieviele Layer braucht man, mit wieviele Neuronen?
  - Frage der Hyperparametersuche (muss herausgefunden werden)
- Wie werden Gewichte W gesetzt?
  - Werden gelernt und zufällig initialisiert.

# Fragen

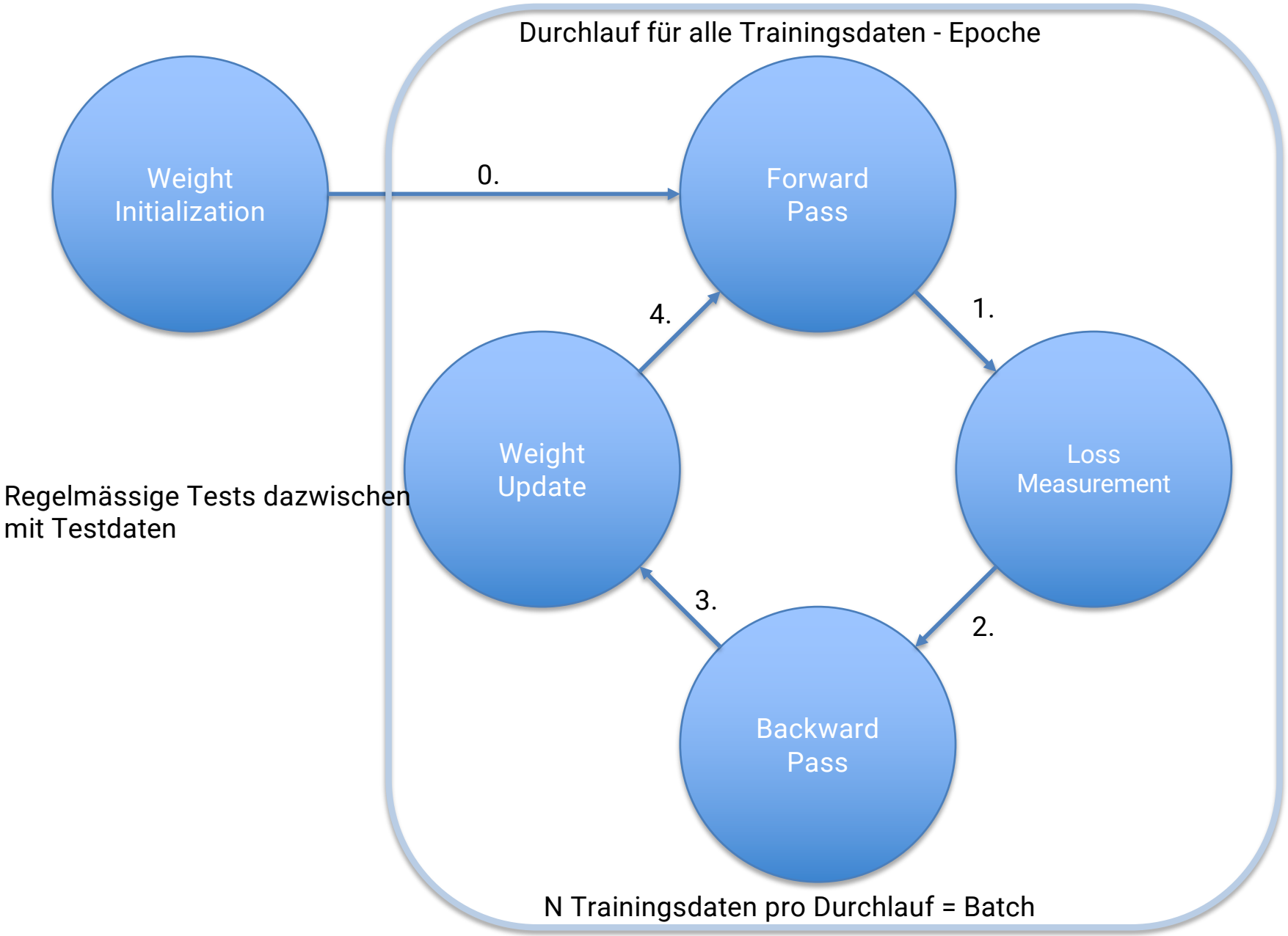
- Welche Funktion(en) sollte man nutzen, um von den Inputdaten auf die Outputklassen zu mappen?
  - Komposition von einfachen Funktionen

- Warum macht es keinen Sinn das Mapping zwischen Layern linear zu machen?
  - Komposition von linearen Funktionen ist wieder eine lineare Funktion

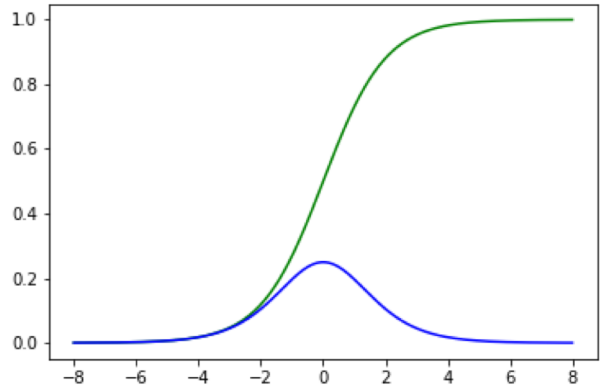
- Was macht die ReLU Aktivierungsfunktion
  - Pise-wise linear tiling



# Trainings- und Testzyklus



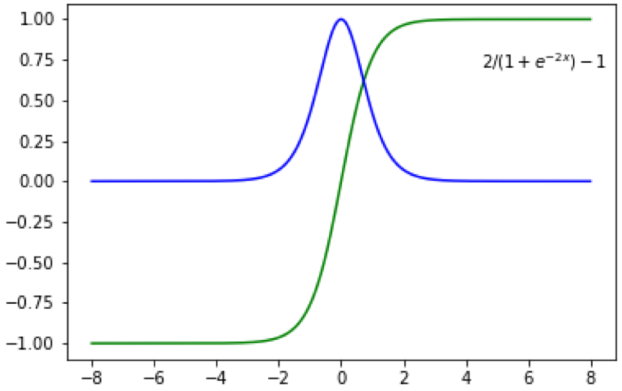
# Aktivierungsfunktionen




**Sigmoid-Aktivierungsfunktion:**

1. Skaliert alle Werte zwischen [0,1]
2. Heute wenig genutzt, weil
    1. Gradienten verloren gehen
    2. Werte nicht um 0 zentriert sind

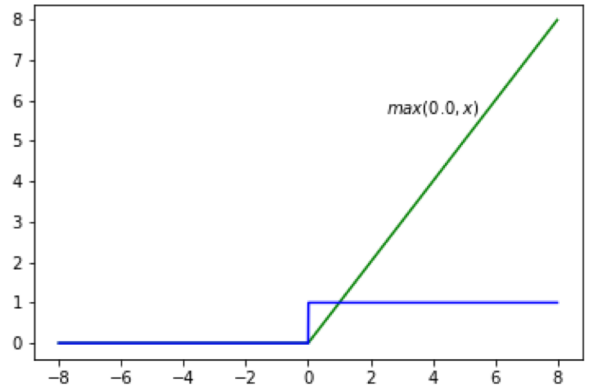
$$\sigma(x) = 1/(1 + e^{-x})$$




**Tanh-Aktivierungsfunktion:**

1. Skaliert alle Werte zwischen [-1,1]
2. Besser als Sigmoid, weil Werte um 0 zentriert

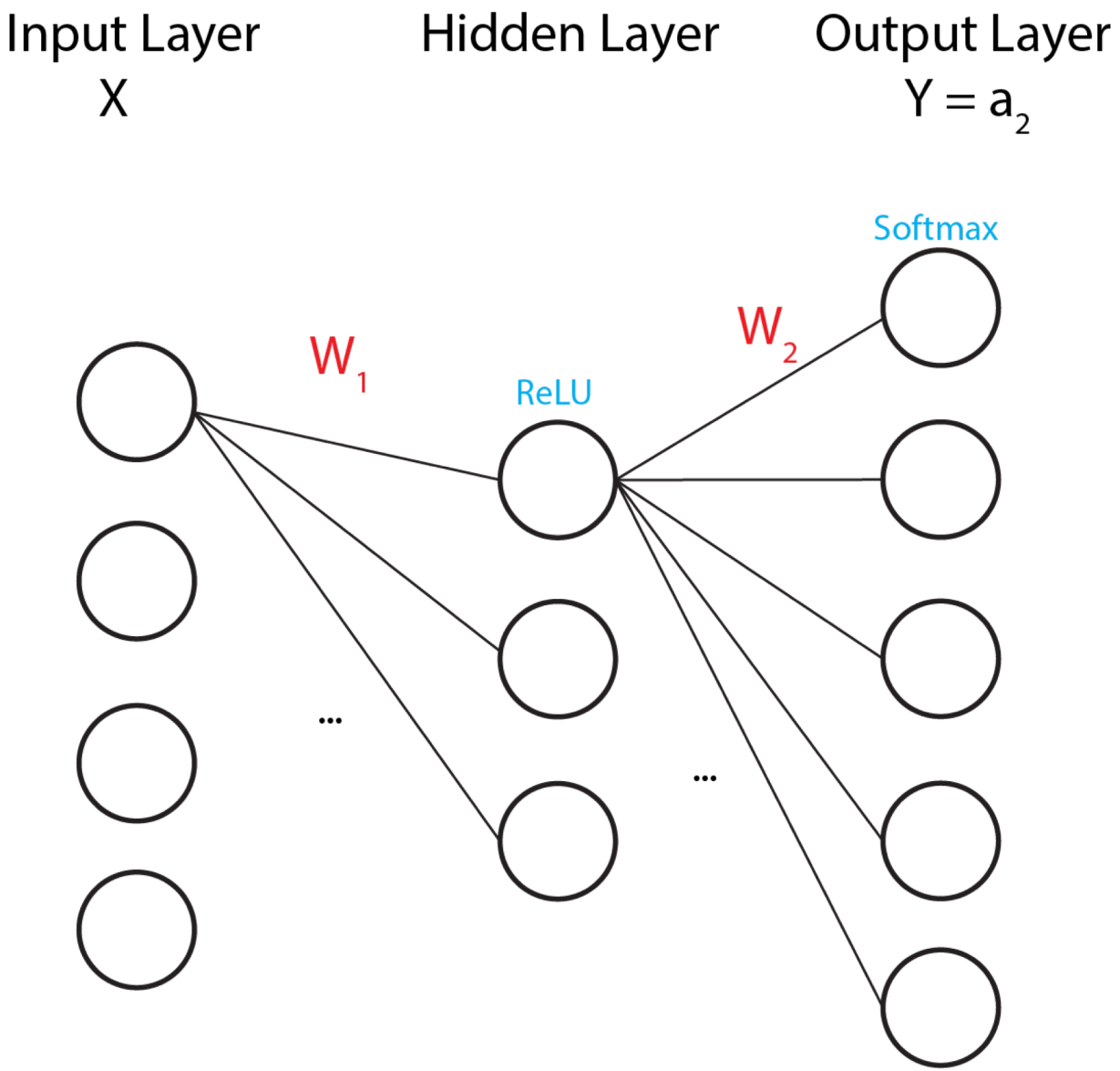
$$\tanh = 2\sigma(2x) - 1$$

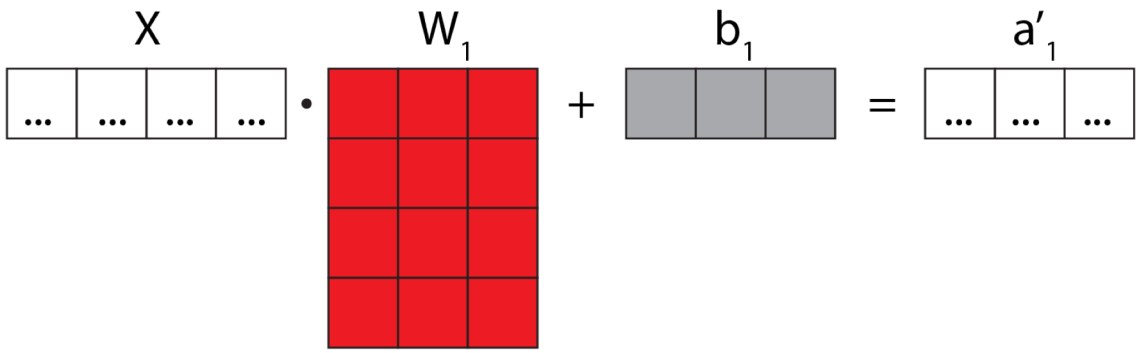


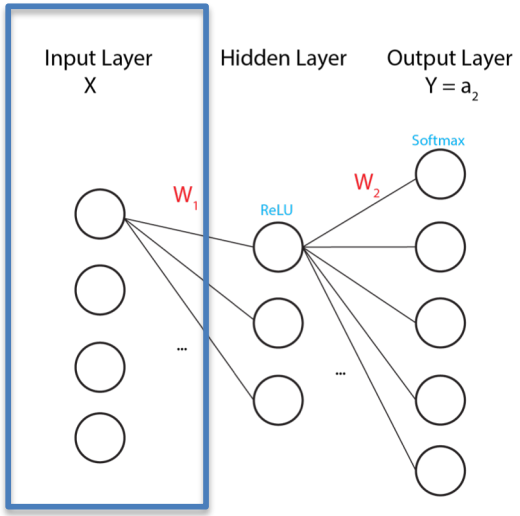

**ReLU-Aktivierungsfunktion:**

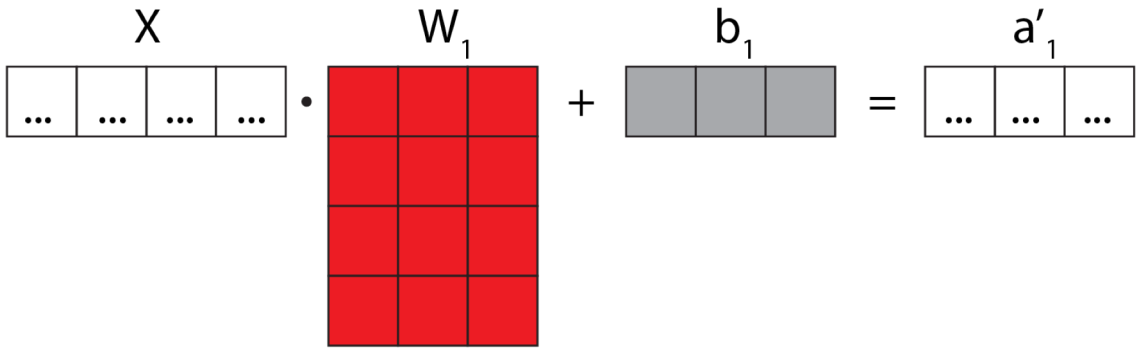
1. Rectified Linear Unit
2. Populär, weil schnell berechenbar und konvergiert schneller, weil Werte saturieren nicht gegen 1

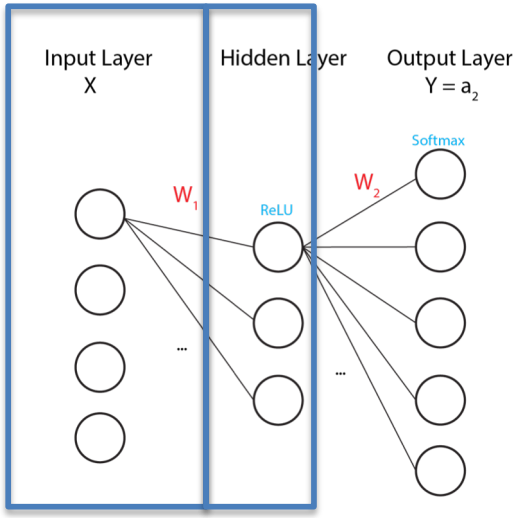
$$\max(0, x)$$


Input Layer
X
Hidden Layer
Output Layer
Y = a2
Softmax
W1
W2
ReLU
...
...

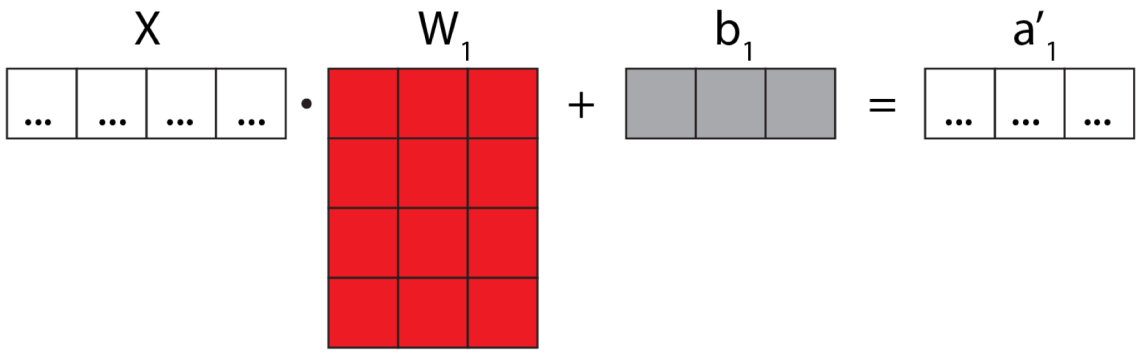

X
W1
b1
a'1
...
...
...
...
...
...
...

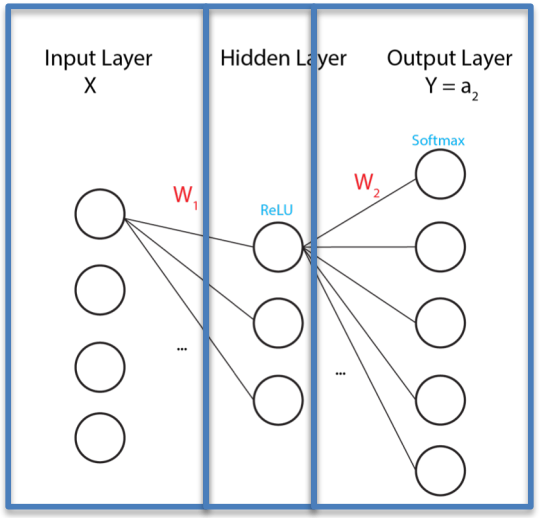

Input Layer
X
Hidden Layer
Output Layer
Y = a2
W1
ReLU
W2
Softmax


X
W1
b1
a'1
...
...
...
...
...
...
...
...
...

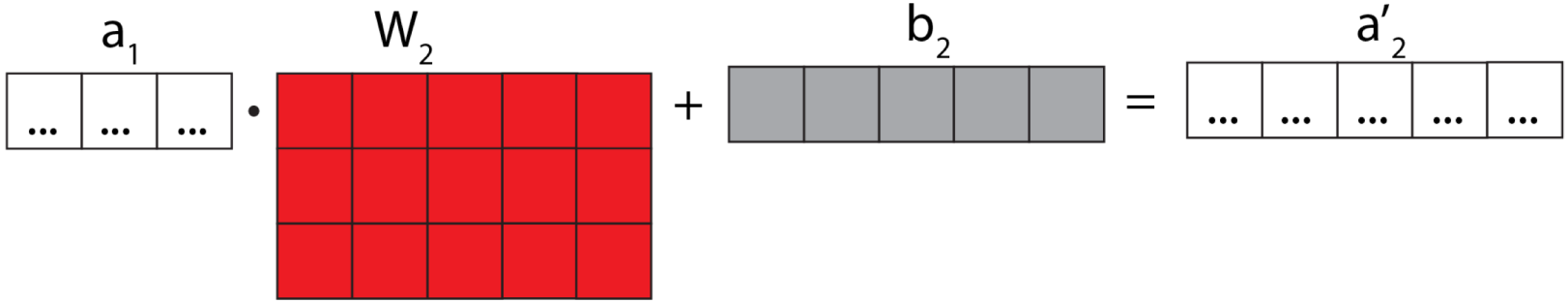

Input Layer
X
Hidden Layer
Output Layer
Y = a2
W1
ReLU
W2
Softmax
...
...

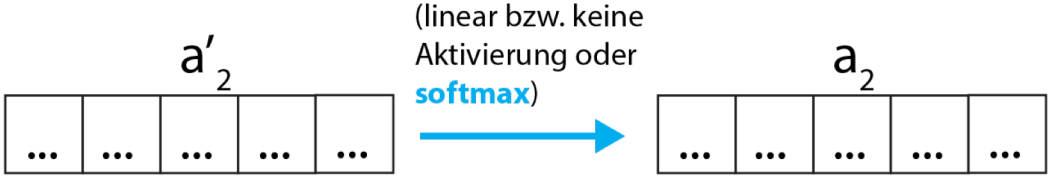

a'1
ReLU
a1
...
...
...
...
...
...


X
W1
b1
a'1
...


Input Layer
X
Hidden Layer
Output Layer
Y = a2
W1
ReLU
W2
Softmax


a'1
ReLU
a1


a1
W2
b2
a'2


a'2
(linear bzw. keine
Aktivierung oder
softmax)
a2