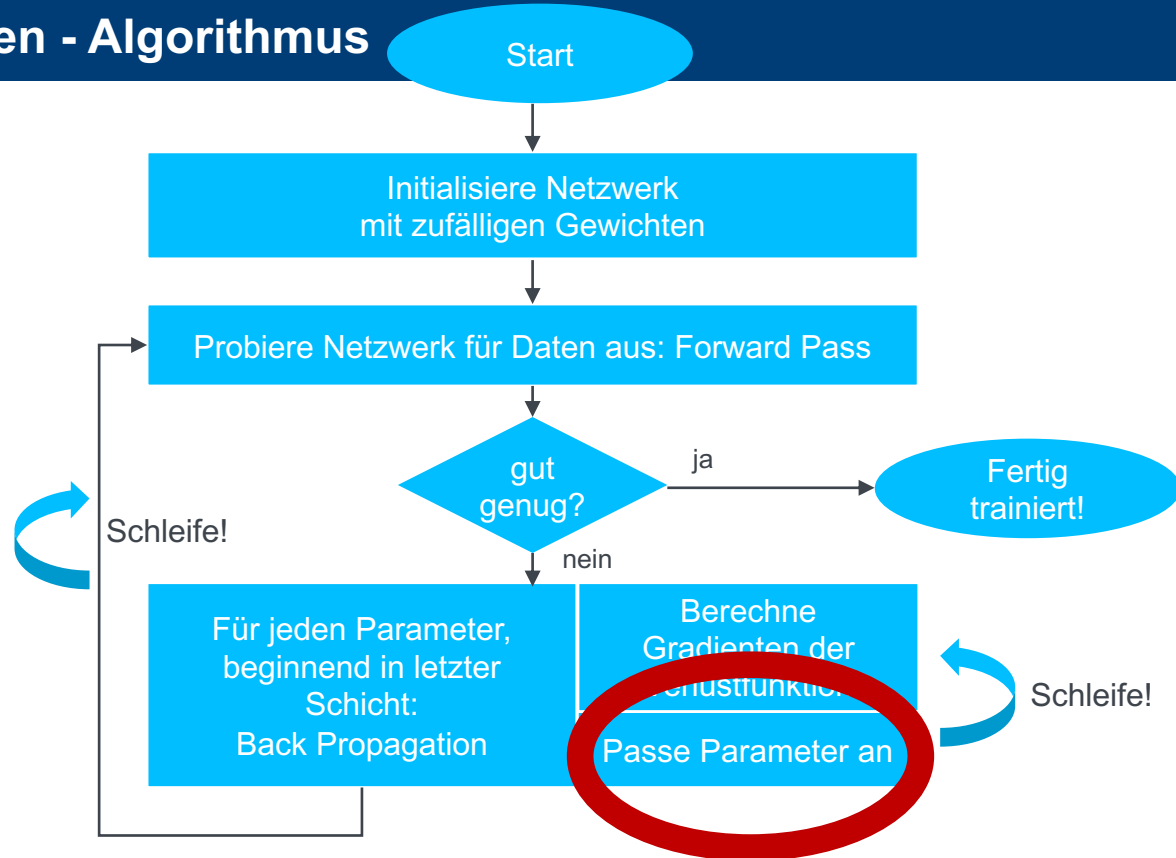
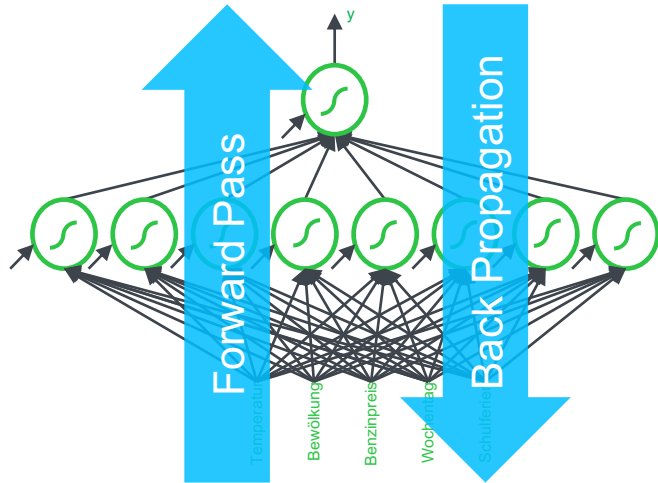


Hyperparameter beim Training von neuronalen Netzen

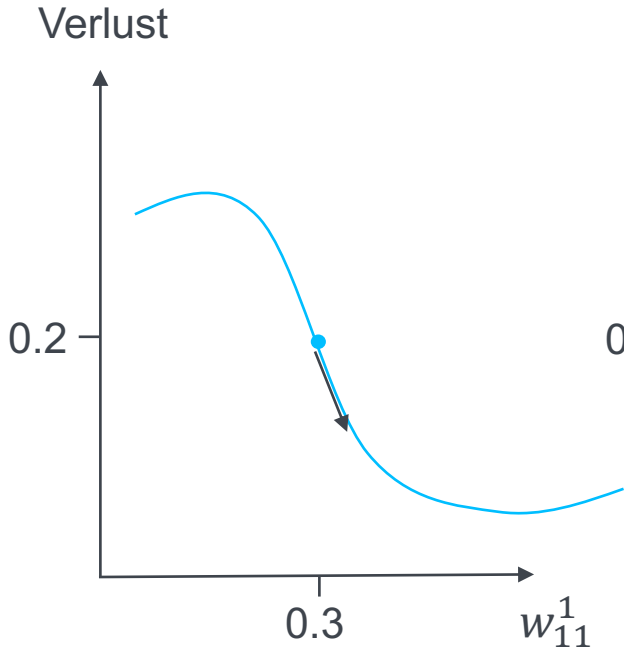
Effizienter üben

Training von neuronalen Netzen - Algorithmus

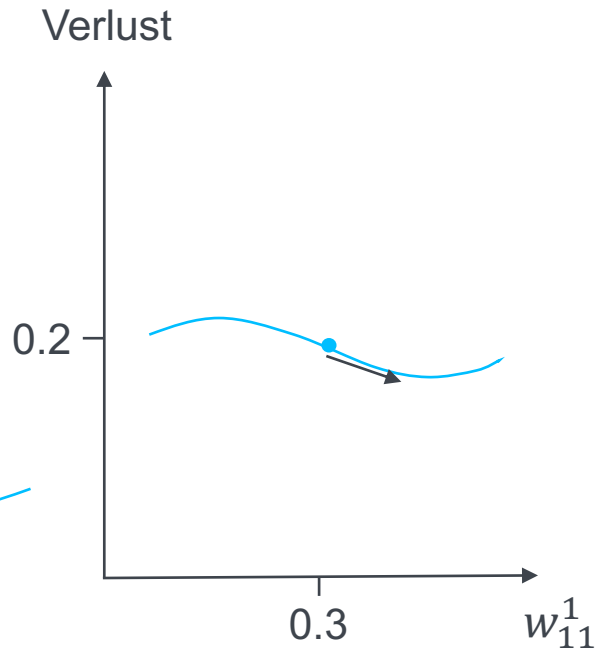


Gradientenabstiegsverfahren

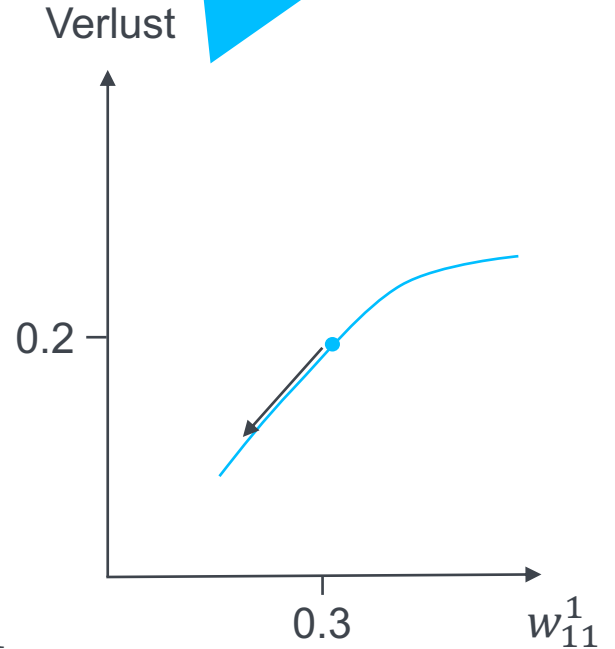
Gradient der Verlustfunktion für w_{ij}^k :
 ∂_{ij}^k



Gradient stark negativ



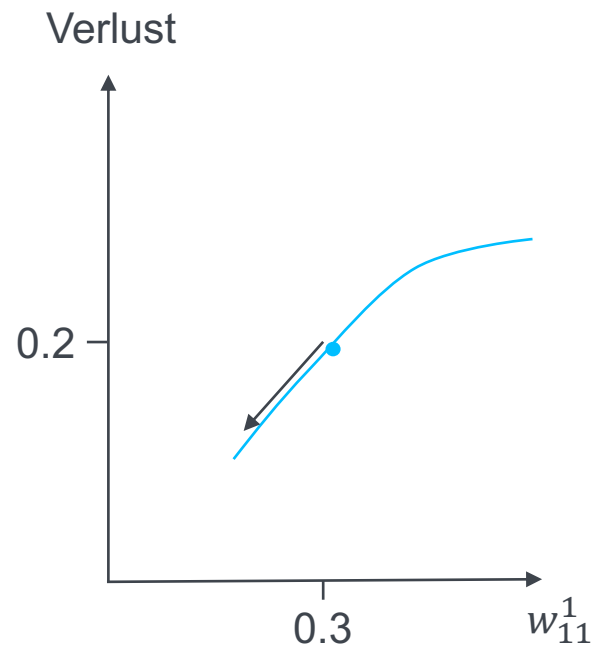
Gradient schwach negativ



Gradient positiv

Anpassung der Parameter beim Gradientenabstiegsverfahren

- Anpassung der Gewichte:
 - $w_{ij}^k = w_{ij}^k - \partial_{ij}^k$
 - Rechts: Steigung ca. 1, also:
 - $w_{ij}^k = 0.3 - 1$
 - Bewirkt bei kleinen Gewichten ggf. sehr starke Veränderung
- Meistens daher:
 - $w_{ij}^k = w_{ij}^k - \eta * \partial_{ij}^k$
 - η Lernrate, z.B. 0.001



Die Lernrate (englisch: Learning Rate) bewirkt eine vorsichtigeren Anpassung der Gewichte. Dabei wird der Gradient mit einem Faktor multipliziert. Der Faktor ist normalerweise nur wenig größer als Null.

Die Lernrate muss vor dem Training festgelegt werden. Es handelt sich um einen sogenannten Hyperparameter.

Hyperparameter sind Parameter, die im Training nicht gelernt werden, sondern vorher festgelegt werden müssen.

Hyperparameter bisher

Entscheidungsbäume

- Baumtiefe
- Maximale Zahl von Blättern
- Mindestmenge von Datenpunkte für Aufteilung/für Blätter
- Pruning-Parameter

Müssen vor dem Training festgelegt werden – werden nicht gelernt

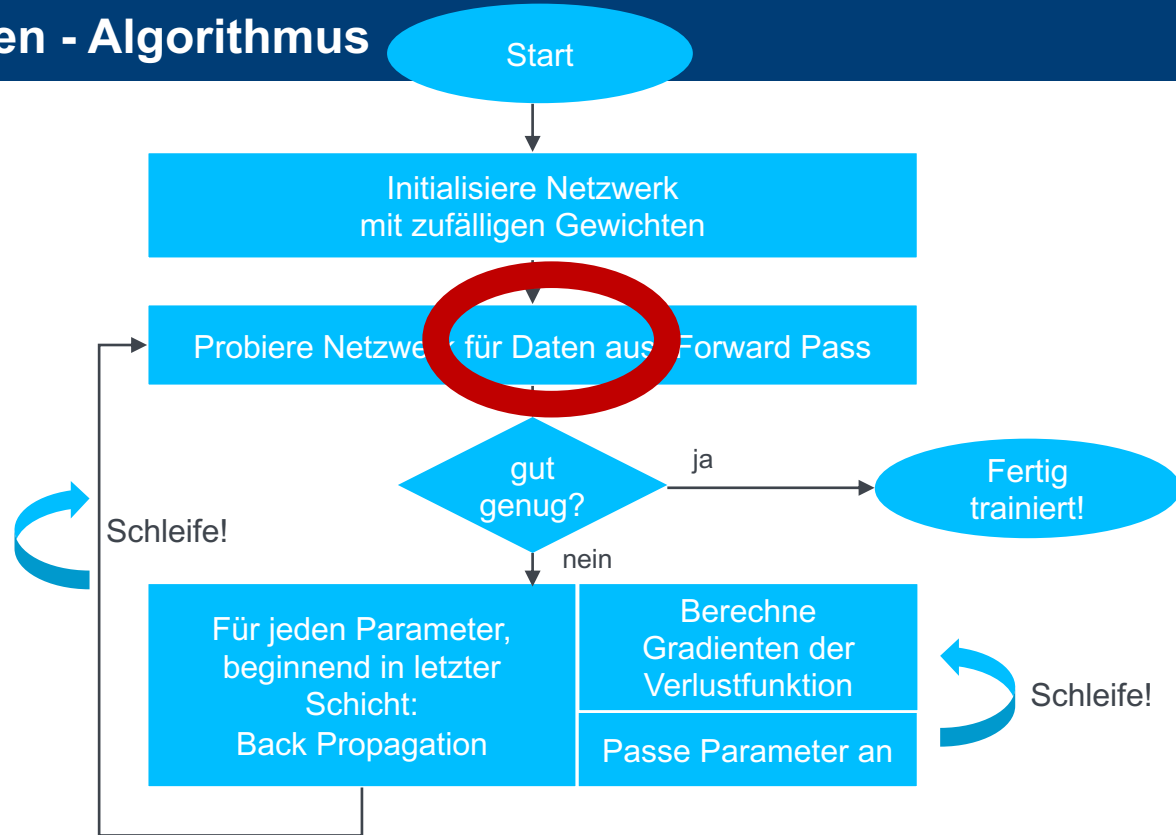
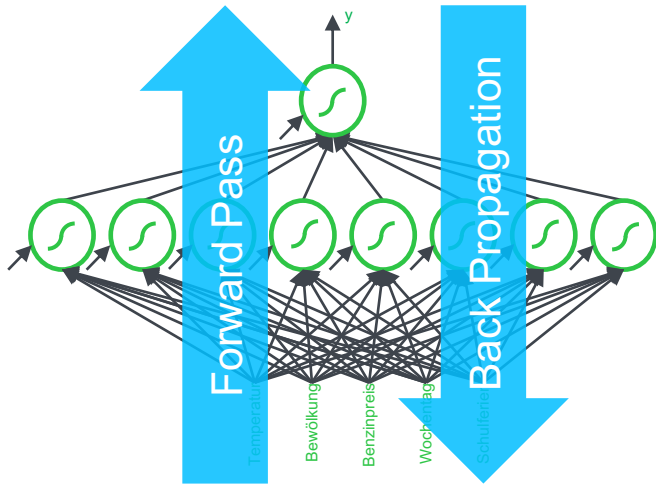
Neuronale Netze

- Anzahl der Schichten
- Anzahl der Neuronen in jeder Schicht
- Wahl der Aktivierungsfunktion

Zu den Ihnen bereits bekannten Hyperparametern für Neuronale Netze gehören die Anzahl der Schichten, die Anzahl der Neuronen in jeder Schicht, und die Wahl der Aktivierungsfunktion.

Auch die Parameter beim Bau von Entscheidungsbäumen sind Hyperparameter.

Training von neuronalen Netzen - Algorithmus



Evaluation des Netzwerks während des Trainings

- Beispiel im letzten Video: Forward Pass und Fehler für nur einen Datenpunkt auf einmal
- In der Praxis:
 - Batches mit mehreren Datenpunkten
 - Verlustfunktion dann: durchschnittlicher Verlust für alle Punkte im Batch
- Hyperparameter: Batchgröße



Training

Modell trainieren

Erweitert ^

Epochen: ?

Batchgröße: ?

Lernrate: ?

Auf Standardwerte zurücksetzen 🕒

Details 📄

Teachable Machine:

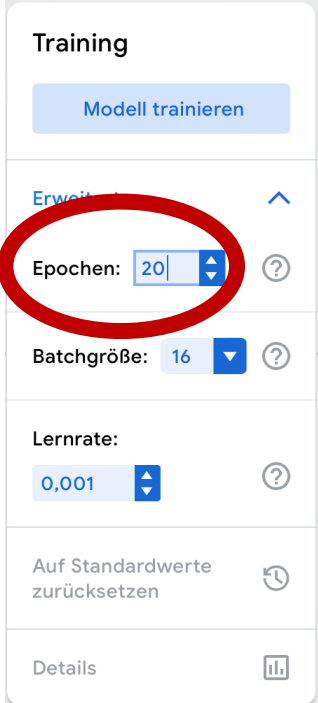
16, 32, ... oder 512

Batchgröße (engl. Batch Size) ist ein Hyperparameter.

Er bestimmt, wie viele Datenpunkte auf einmal zur Evaluierung der Parameter verwendet werden.

Evaluation des Netzwerks während des Trainings

- Beispiel im letzten Video: Forward Pass und Fehler für nur einen Datenpunkt auf einmal
- In der Praxis:
 - Batches mit mehreren Datenpunkten
 - Verlustfunktion dann: durchschnittlicher Verlust für alle Punkte im Batch
- Hyperparameter: Batchgröße



Teachable Machine

Default: 50

16, 32, ... oder 512

Default: 0.001

Batches beim Training



Bei n Trainingsdaten und k Datenpunkten pro Batch braucht man n/k Batches, um eine Epoche abzuschließen

Zum Beispiel: 10 000 Trainingsdatenpunkte, 500 pro Batch

⇒ $10\,000/500 = 20$ Batches

Wenn alle Trainingsdaten einmal zum Optimieren der Parameter verwendet wurden, ist eine Epoche abgeschlossen.

Bei n Trainingsdatenpunkten und Batchgröße k besteht eine Epoche also aus der Verarbeitung von n/k Batches.



Single Choice: Lernrate



**Drag and Drop: Welcher
Hyperparameter gehört zu
welcher Art von Training?**

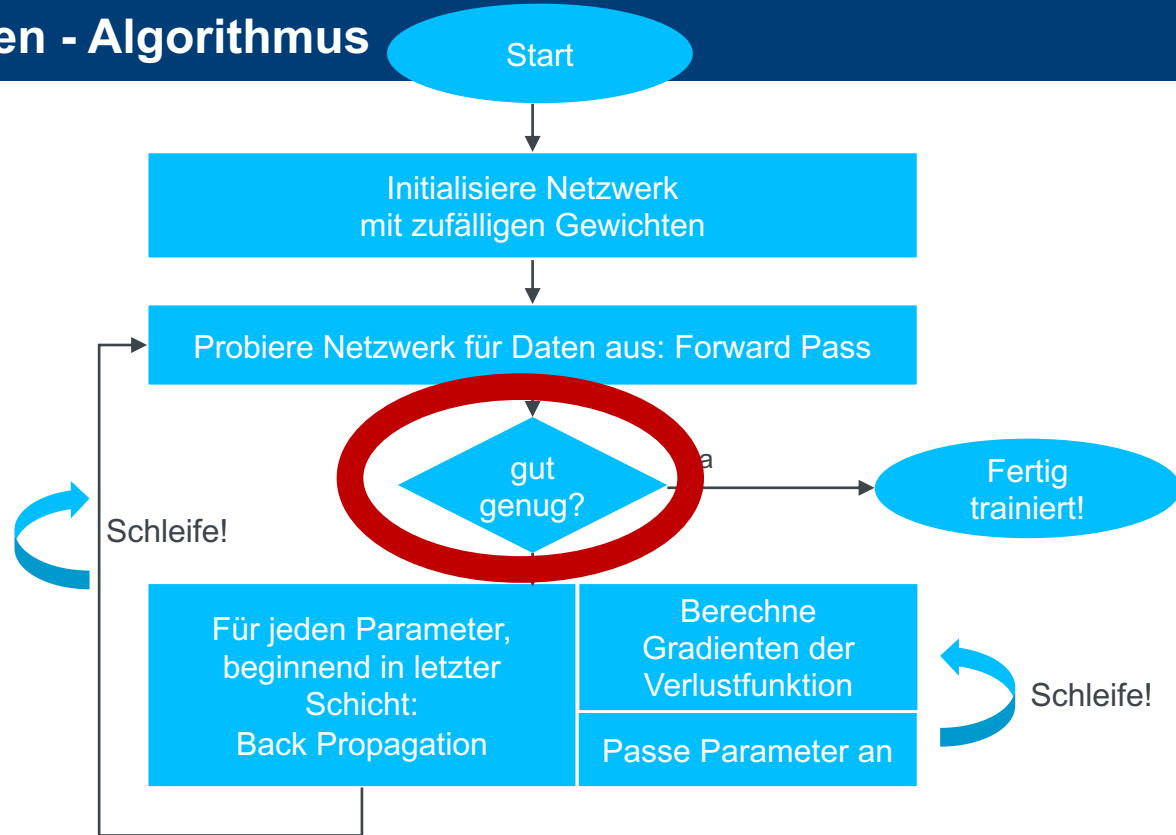
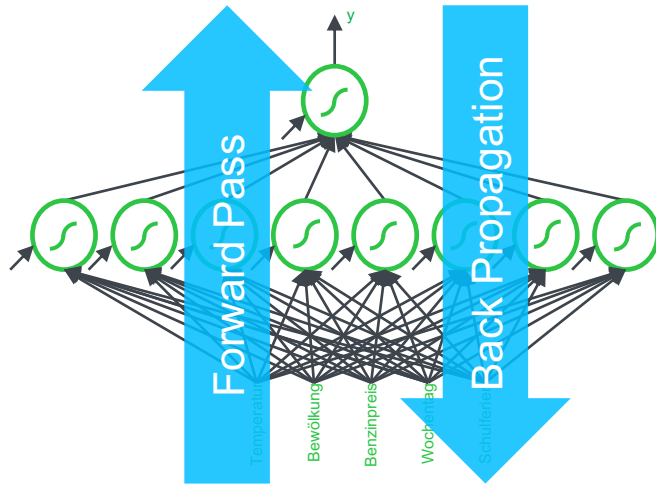


Drag the Words: Batches



Fill in the Blanks: Batchgröße

Training von neuronalen Netzen - Algorithmus



Was ist “gut genug“?

Training

Modell trainieren

Erweitert ^

Epochen: ?

Batchgröße: ?

Lernrate: ?

Auf Standardwerte zurücksetzen ↻

Details ☰

- Konfigurationsdialog aus Teachable Machine
- Hier: mögliche Parameter für das Training von Apfel- vs. Zitrusfrüchten (siehe Abschnitt Algorithmischer Bias)

Was ist “gut genug“?

Genauigkeit pro Klasse

CLASS	ACCURACY	# SAMPLES
Class 1	1.00	4
Class 2	1.00	3



Validierungsdaten

- Detail-Ansicht aus Teachable Machine-Projekt
- Trainingsdaten:
 - 22 Objekte in Class 1
 - 17 Objekte in Class 2
- Teachable Machine hält davon 7 (4+3) zurück und trainiert auf den restlichen
- Ziel: Beurteilung der Qualität des Modells schon während des Trainings

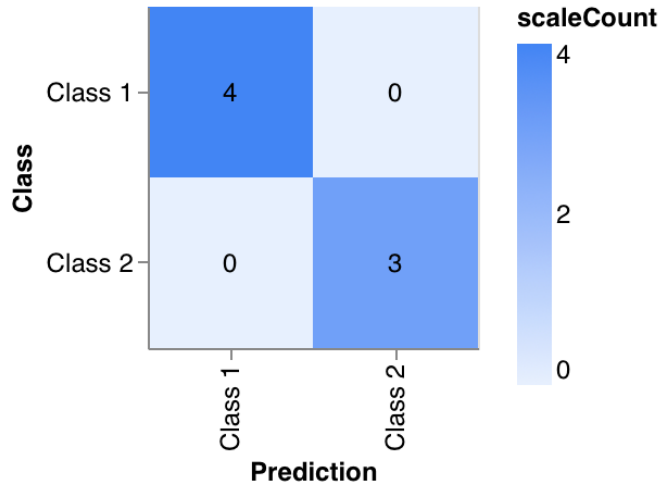
Die Validierungsdaten dienen zur Bewertung des Modells während des Trainings.

Bei den Validierungsdaten handelt es sich NICHT um die Testdaten. Statt dessen wird meist ein Teil der Trainingsdaten als Validierungsdaten zurückgehalten.

Die Testdaten werden erst NACH Abschluss des Trainings zur Evaluation verwendet.

Visualisierung der Qualität des Modells

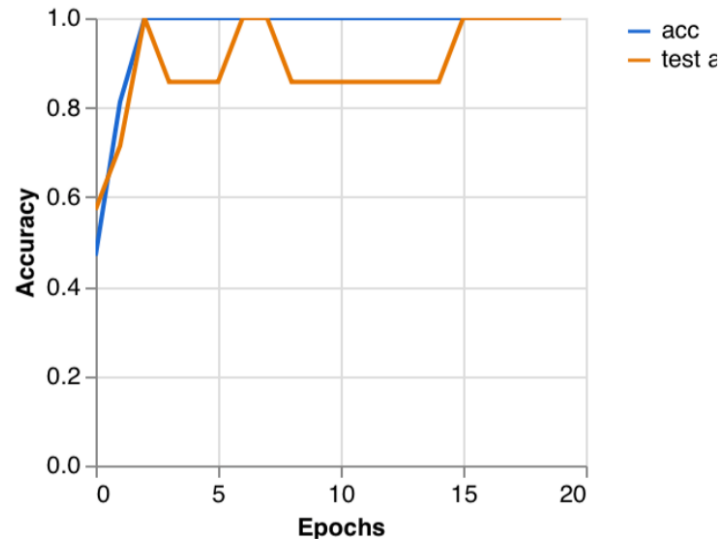
Wahrheitsmatrix



- Teachable Machine bietet nach Abschluss des Trainings die finale Konfusionsmatrix für die Validierungsdaten an
- Hier: nach dem Training wurden alle Validierungsdaten korrekt klassifiziert

Entwicklung der Accuracy im Lauf des Trainings

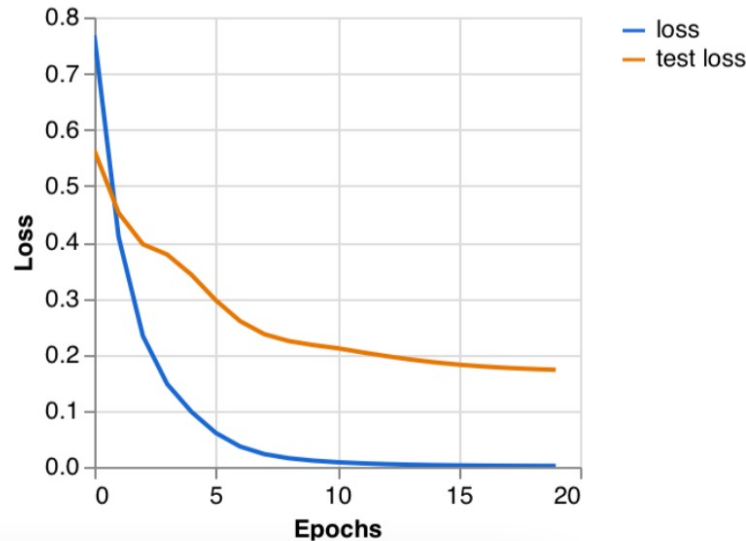
Genauigkeit pro Epoche



- Blau: Accuracy auf den Trainingsdaten
- Erreicht nach wenigen Epochen 100%
- Orange: Accuracy auf den Validierungsdaten (von Teachable Machine unglücklich als “test acc“ bezeichnet)
- Nicht ganz so konsistent bei 100%, erst ab Epoche 15
- Gut sichtbar: das Modell verbessert sich im Lauf des Trainings

Entwicklung des Fehlers (Verlust) im Lauf des Trainings

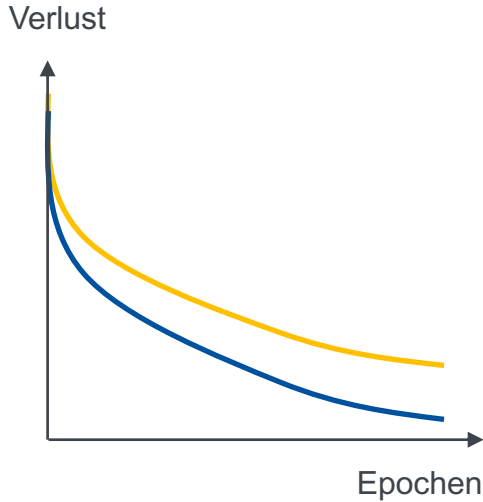
Verlust pro Epoche



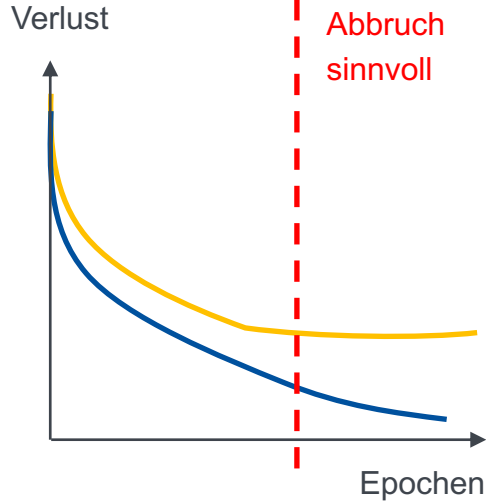
- Gebräuchlicher als Accuracy zur Bewertung des Trainingsverlaufs
- Blau: Verlust auf den Trainingsdaten
- Orange: Verlust auf Validierungsdaten
- Wird nie so klein wie der Verlust auf den Trainingsdaten (auch wenn die Klassifizierung am Ende für die Validierungsdaten 100% ist, wie eben gesehen)
- Leichtes Overfitting!

Early
Stopping

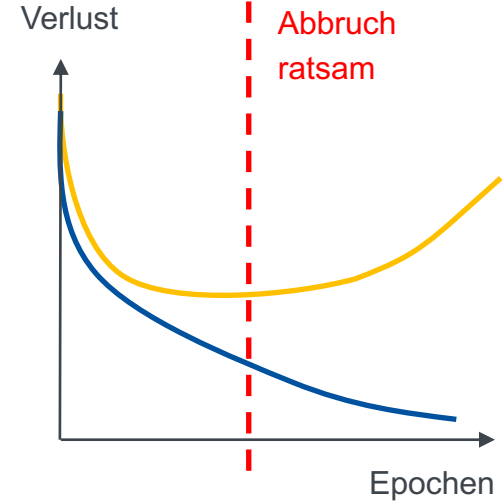
Grad an Overfitting



Modell lernt noch
Brauchbares
(Verbesserung auf
Validierungsdaten)



Modell lernt nichts mehr

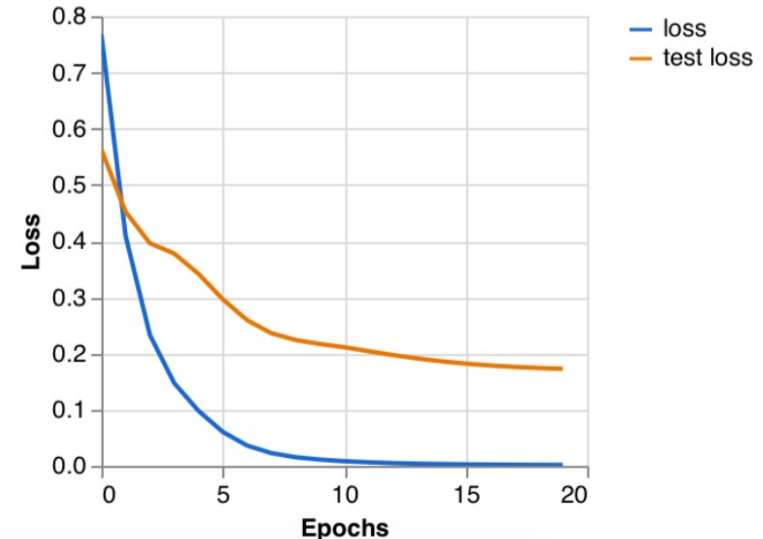


Modell wird immer
schlechter!

Early Stopping

- In vielen Fällen wird nicht blind eine vorgegebene Anzahl von Epochen lang trainiert
- Statt dessen Abbruch, wenn “gut genug“
- Typische Kriterien für „gut genug“:
 - Der Verlust auf den Validierungsdaten bleibt über mehre Epochen unverändert
 - ... wird wieder größer
 - ... verringert sich nur noch sehr langsam

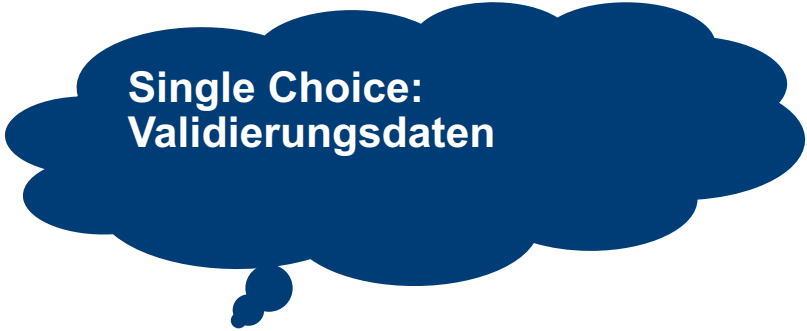
Verlust pro Epoche



Early Stopping ist eine Technik, um Overfitting zu reduzieren.

Beim Early Stopping wird das Training beendet, wenn der Verlust auf den Validierungsdaten nahe legt, dass das Modell sich nicht mehr verbessert.

Weitere Hyperparameter beim Training neuronaler Netze sind die Aufteilung in Trainings- und Validierungsdaten sowie die Anzahl der Epochen beim Training.



**Single Choice:
Validierungsdaten**



Multiple Choice: Accuracy



**Multiple Choice: Entwicklung
des Fehlers**



Drag and Drop: Early Stopping



**Multiple Choice:
Validierungsdaten**



Drag the Words: Early Stopping

Dr. Antje Schweitzer

Universität Stuttgart
Institut für Maschinelle Sprachverarbeitung



Universität Stuttgart

Institut für Maschinelle Sprachverarbeitung
Institut für Software Engineering



IHK Industrie- und Handelskammer
Reutlingen

Reutlingen | Tübingen | Zollernalb



IHK Region Stuttgart



IHK Industrie- und Handelskammer
Karlsruhe



Lizenzbestimmungen

“Hyperparameter beim Training von Neuronalen Netzen” von Antje Schweitzer, KI B³ / Uni Stuttgart

Das Werk - mit Ausnahme der folgenden Elemente:

- Logos der Verbundpartner und des Förderprogramms
- im Quellenverzeichnis aufgeführte Medien

ist lizenziert unter:

 [CC BY 4.0 \(https://creativecommons.org/licenses/by/4.0/deed.de\)](https://creativecommons.org/licenses/by/4.0/deed.de)

(Namensnennung 4.0 International)

Quellenverzeichnis

Titelfoto: [Gabriel Rodrigues](https://unsplash.com/@gabrielrodriguesjpg/likes) (https://unsplash.com/@gabrielrodriguesjpg/likes), ohne Titel, auf [Unsplash](https://unsplash.com/photos/4Cc0rs9pWl8) (https://unsplash.com/photos/4Cc0rs9pWl8), ist lizenziert unter [Unsplash-Lizenz](https://unsplash.com/license) (https://unsplash.com/license).
Bildausschnitt verändert.