

06.06.24

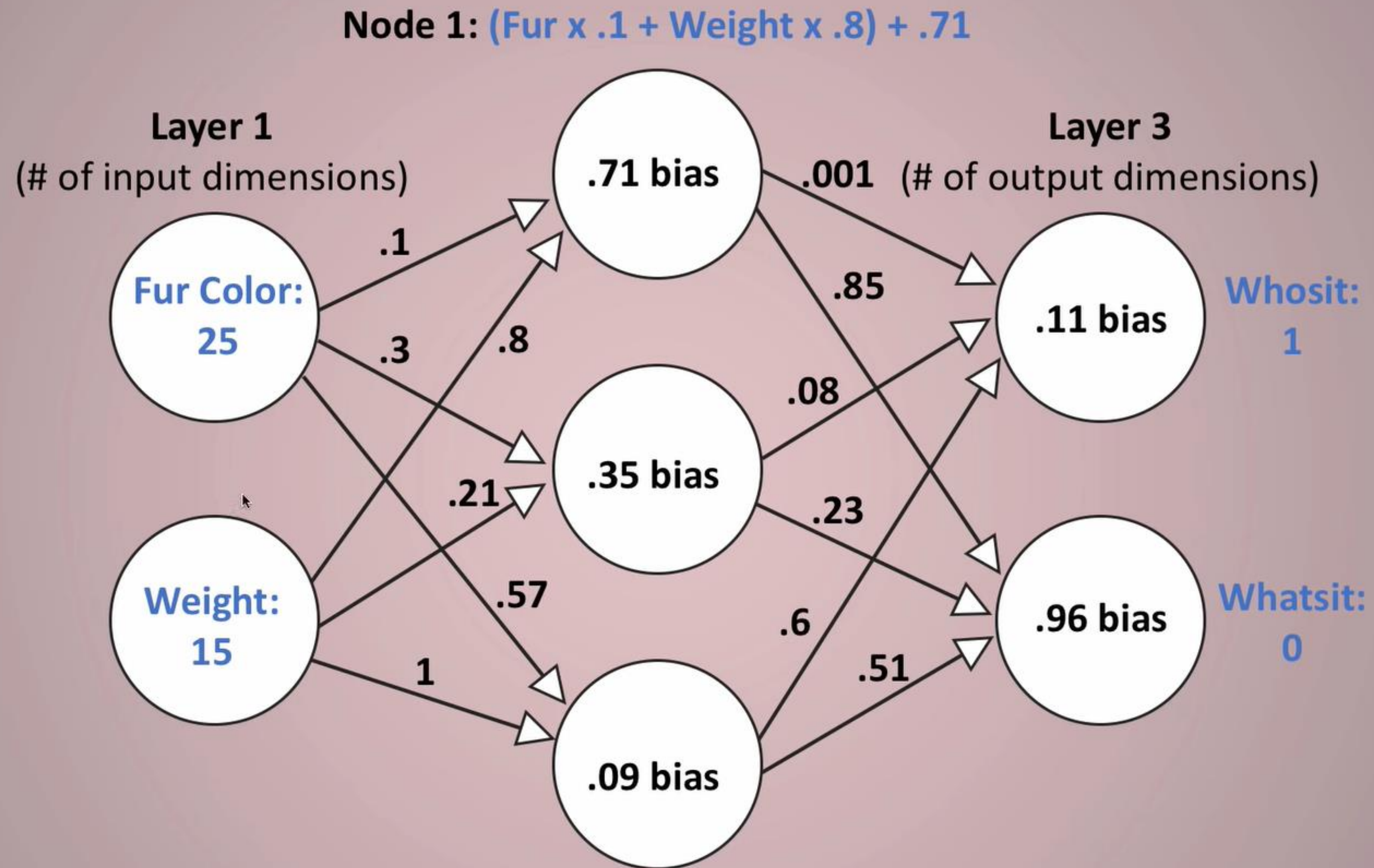
Einführung in Data Science und maschinelles Lernen

NEURONALE NETZE

- **Quiz**
- **Neuronale Netze (NN)**
- **Hyperparameter in NN**
- **Frameworks zur
Implementierung von NN**
- **Implementierung eines NN
mit TensorFlow**

QUIZ





Learning Rate:

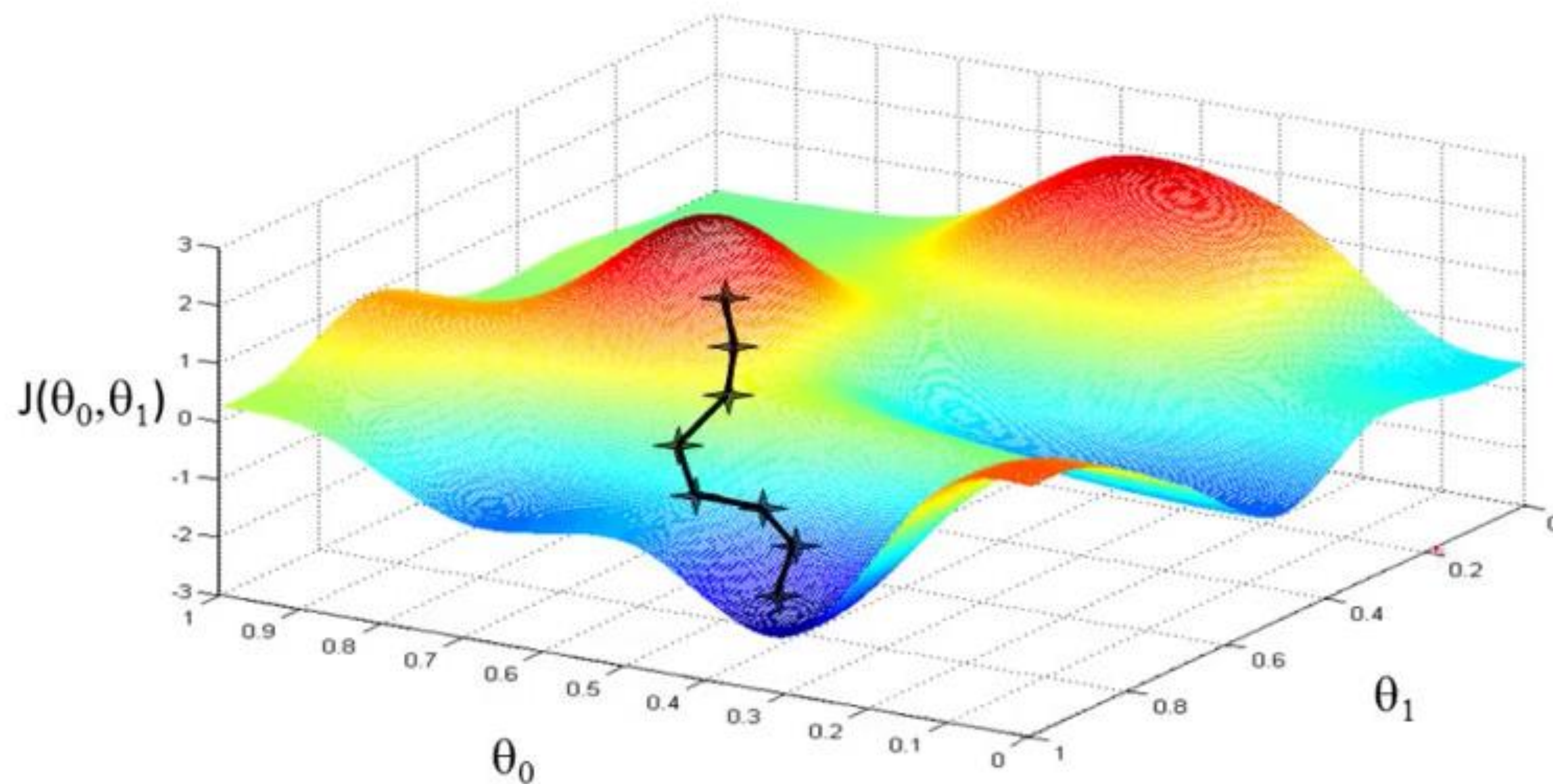
How much should this step outcome affect our weights and biases?

Momentum:

How much should past outcomes affect our weights and biases?

$$\text{change} = (\text{learningRate} * \text{delta} * \text{value}) + (\text{momentum} * \text{pastChange})$$

PARAMETER VON OPTIMIZERN



- **Schrittgröße für die Annäherung an das Kosten-Minimum („Learning Rate“)**
- **Trägheit bei Richtungsänderungen („Momentum“)**

PARAMETER DES OPTIMIZERS „ADAM“

- **Initialer Lernparameter (Schrittweite) für die Optimierung :**
alpha (learning rate)
- **Anteil des aktuellen Gradienten in der Berechnung des nächsten Optimierungsschritts:**
beta1 and beta2 (decay rates)

HYPERPARAMETER IN NEURONALEN NETZEN

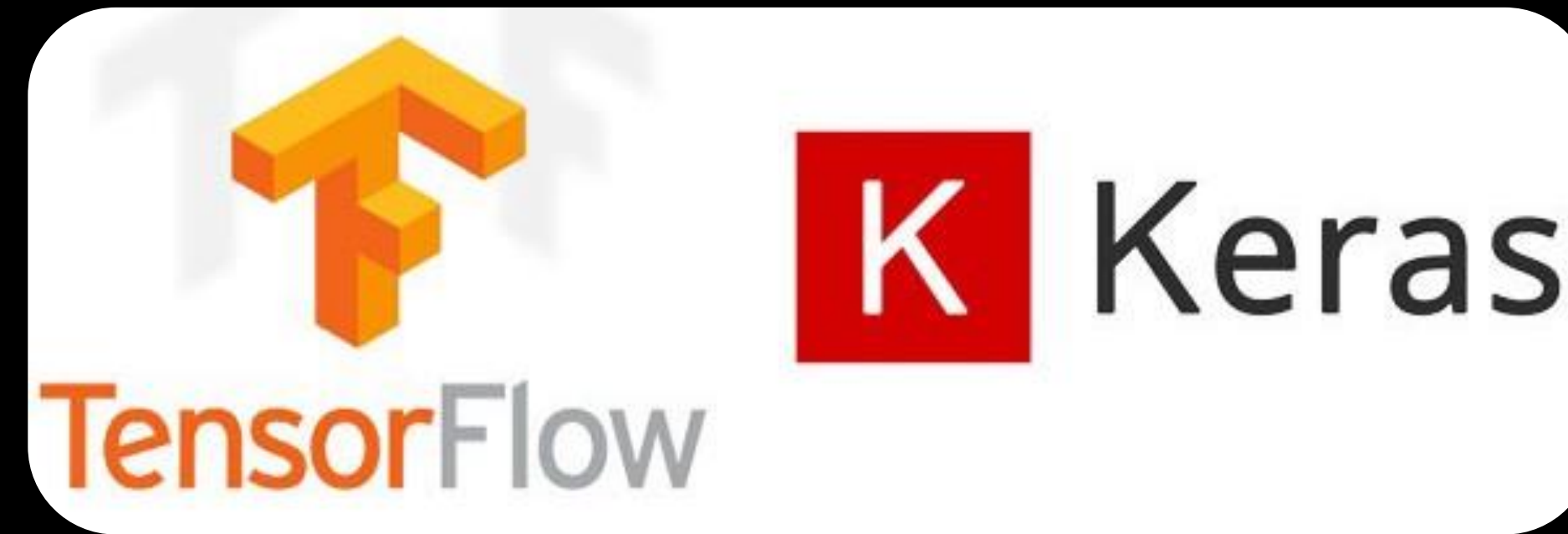
- **Wahl der Architektur:**
 - **Anzahl der Hidden Layer des Netzes**
 - **Typen der Hidden Layer**
 - **Anzahl der Neuronen je Hidden Layer**
 - **Wahl der Aktivierungsfunktionen**
- **Wahl der Kostenfunktion („Loss Function“)**
- **Wahl der Optimierungsfunktion („Optimizer“)**
- **Wahl der Learning Rate des Optimizers**



PYTORCH



Transformers



- **TensorFlow 0.1 (Nov 2015):** Veröffentlichung als Open Source Software durch Google; entwickelt durch das Google Brain Team für die interne Forschung und Produktion)
- **TensorFlow 1.4 (Nov 2017):** Entwicklung der Keras API als High-Level-API für TensorFlow und andere ML-Bibliotheken, zur Erhöhung der Benutzerfreundlichkeit für häufig verwendete Modelle.
- **TensorFlow 2.0 (Sep 2019):** Keras wird als High-Level-API in TensorFlow integriert.
- **TensorFlow 2.3 (Oktober 2020):** Erhebliche Leistungssteigerungen, verteiltes Training, quantisiertes Training und verbesserte Mobile-Deployments.
- **Keras 3.0 (Dez 2023):** Großes Release, das die Unterstützung für mehrere Backends, einschließlich TensorFlow, JAX und PyTorch, erweitert und Keras zu einem vielseitigen Framework für verschiedene Deep Learning-Bedürfnisse macht.



- **PyTorch 0.1 (Oktober 2016):** Veröffentlichung als Open Source durch Facebook AI
- **PyTorch 1.0 (Dezember 2018):** Stabile 1.0 Version mit Produktionsunterstützung.
- **PyTorch 1.3 (Oktober 2019):** Verbesserungen für quantisiertes Training und Mobile-Deployments.
- **PyTorch 1.8 (Dezember 2020):** **Bedeutende Leistungssteigerungen, verbesserte Unterstützung** für Reinforcement Learning.
- **(September 2022):** Übergabe von PyTorch an die Linux Foundation
- **PyTorch 2.0 (March 2023):** Verbesserte Performance insbesondere für Transformer Modelle.



Transformers

- **Transformers 1.0 (Dezember 2018):** Hugging Face veröffentlicht die erste Version der Transformer-Bibliothek als Open Source mit Implementierungen beliebter Transformer-Modelle wie BERT, GPT und anderen.
- **Transformers 2.0 (März 2020):** Unterstützung für Fine-Tuning, produktive Bereitstellung und Quantisierung. Höhere Leistung und intuitivere APIs.
- **Transformers 3.0 (November 2020):** Unterstützung für mehr Aufgabenbereiche wie Computer Vision, Audio und Reinforcement Learning. Leistungsoptimierungen.
- **Transformers 4.0 (Mai 2022):** Verbesserte Performanz und Integration weiterer neuer Techniken zur Verbesserung der Modelloptimierung.

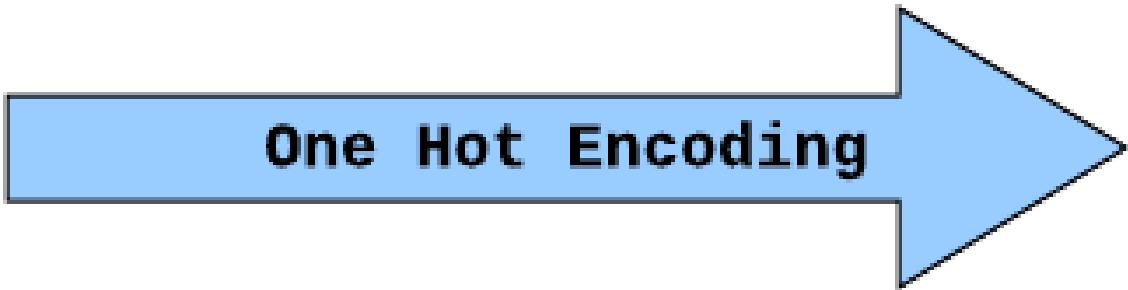
DATENAUFBEREITUNG

Bei jeder Modellierung müssen die Daten folgende Eigenschaften haben:

- 1. Es darf keine fehlenden Werte geben.**
- 2. Alle Werte müssen Zahlen sein.**
- 3. Kategoriale Variablen sind dummy-kodiert.**

ONE-HOT ENCODING

id	color
1	red
2	blue
3	green
4	blue



id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

DATENAUFBEREITUNG FÜR DAS TRAINING DES NEURONALEN NETZES

FileEditSelectionViewGoRunTerminalHelp

←→Untitled (Workspace)

🔍📄📄📄📄-🗑️✕

tokenizer.ipynbneural_net_data_preparation.ipynb •neural_net_estimation.ipynb

🔍+ Code + Markdown | ▶ Run All ↺ Restart 🗑️ Clear All Outputs | 📄 Variables 📄 Outline ...

Python 3.8.0

18

Import Libraries and Data

▶

```
# Import necessary libraries
import pandas as pd
import numpy as np
import os

# Import Data
data = pd.read_csv("https://raw.githubusercontent.com/opencampus-sh/einfuehrung-in-data-science-und-ml/main/house_pricing_data/house_pricing_train.csv")
data.head() # Print first few rows to verify
```

[1] ✓ 5.5s

Python

...

	id	date	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	...	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_livi
0	6840701095	20150403T000000	3	1.00	1740	4400	1.5	0	0	3	...	1740	0	1924	0	98122	47.6059	-122.300	
1	1025049114	20140717T000000	3	2.25	1270	1566	2.0	0	0	3	...	1060	210	2014	0	98105	47.6647	-122.284	
2	4025300360	20150326T000000	3	2.00	1130	16875	1.0	0	0	4	...	1130	0	1947	0	98155	47.7489	-122.300	
3	5536500200	20140918T000000	5	3.50	3760	4857	2.0	0	3	3	...	2820	940	2004	0	98072	47.7398	-122.167	
4	1245003660	20150321T000000	3	2.00	1470	6000	1.0	0	0	3	...	1090	380	1950	1996	98033	47.6829	-122.202	

5 rows × 21 columns

...

Data Preparation

```
# Define categorical features
categorical_features = ['bathrooms', 'zipcode', 'condition']

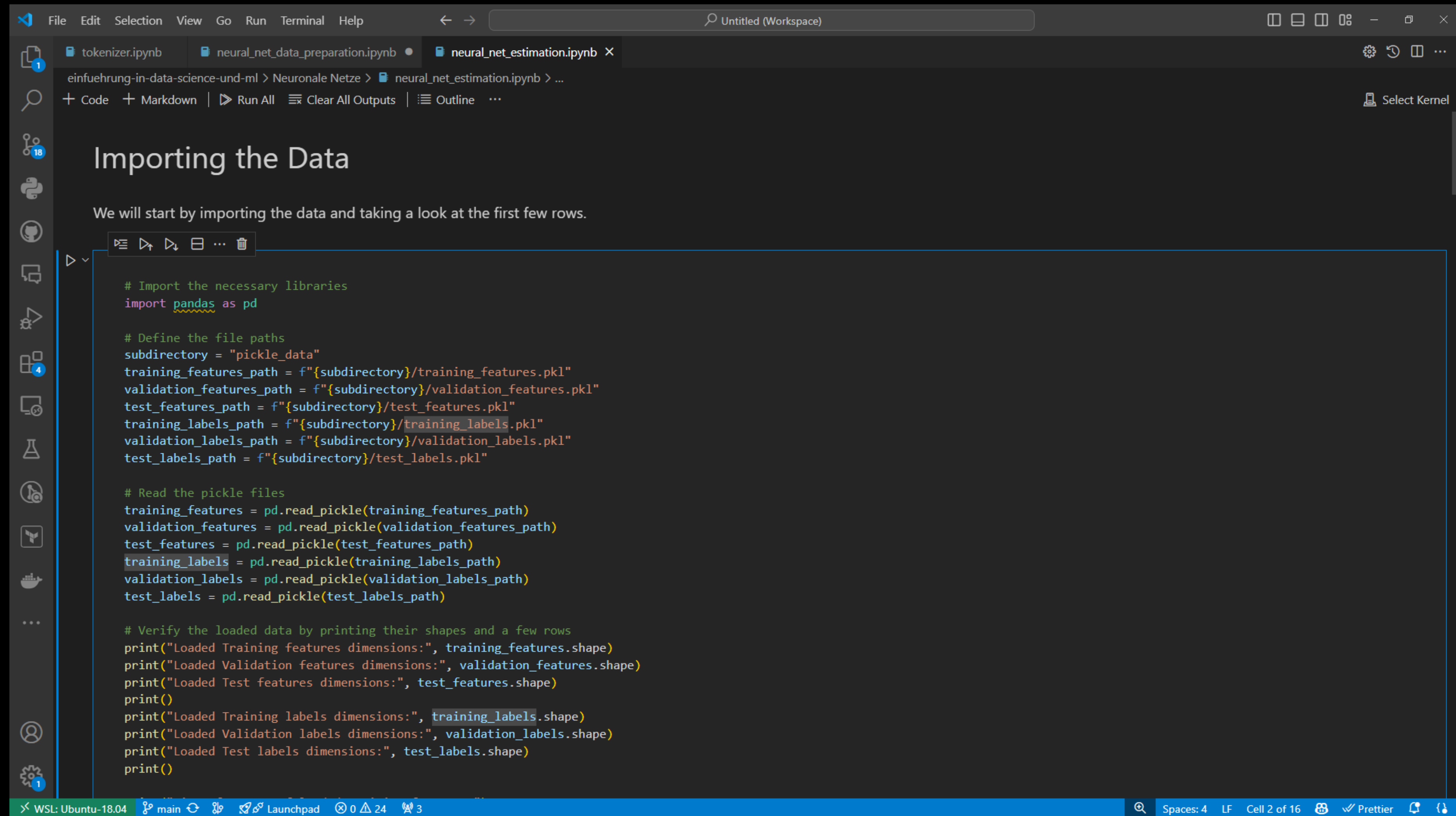
# Inspect data types and unique values for categorical columns
print(data[categorical_features].dtypes)
print("Unique Values:\n", data[categorical_features].apply(lambda x: x.unique()))
```

WSL: Ubuntu-18.04main ↺🔍Launchpad

0 24 🗑️ 3

🔍Spaces: 4LFCell 6 of 8🔍🔍

DEFINITION UND OPTIMIERUNG EINES NEURONALEN NETZES



```
# Import the necessary libraries
import pandas as pd

# Define the file paths
subdirectory = "pickle_data"
training_features_path = f"{subdirectory}/training_features.pkl"
validation_features_path = f"{subdirectory}/validation_features.pkl"
test_features_path = f"{subdirectory}/test_features.pkl"
training_labels_path = f"{subdirectory}/training_labels.pkl"
validation_labels_path = f"{subdirectory}/validation_labels.pkl"
test_labels_path = f"{subdirectory}/test_labels.pkl"

# Read the pickle files
training_features = pd.read_pickle(training_features_path)
validation_features = pd.read_pickle(validation_features_path)
test_features = pd.read_pickle(test_features_path)
training_labels = pd.read_pickle(training_labels_path)
validation_labels = pd.read_pickle(validation_labels_path)
test_labels = pd.read_pickle(test_labels_path)

# Verify the loaded data by printing their shapes and a few rows
print("Loaded Training features dimensions:", training_features.shape)
print("Loaded Validation features dimensions:", validation_features.shape)
print("Loaded Test features dimensions:", test_features.shape)
print()
print("Loaded Training labels dimensions:", training_labels.shape)
print("Loaded Validation labels dimensions:", validation_labels.shape)
print("Loaded Test labels dimensions:", test_labels.shape)
print()
```

BREAKOUT

- **Öffnet die Beispiel-Notebook in Colab, ladet sie herunter und dann bei Euch in den Codespace wieder hoch.**
- **Lasst die Notebooks einmal unverändert bei Euch durchlaufen.**
- **Ergänzt Eure Datenaufbereitung um die in [diesem Beispiel-Notebook](#) durchgeführten Schritte:**
 - 1) **One-Hot-Kodierung kategoriemer Variablen**
 - 2) **Entfernen von Fällen mit Missing Values**
 - 3) **Export der Trainings-, Validierungs- und Testdaten als Pickle-Dateien**
- **Schätzt ein erstes neuronales Netz auf Basis [dieses Beispiel-Notebooks](#).**

BATCH UND EPOCHE

Batch

- **Eine Menge von Fällen, die genutzt wird, um die Gewichte des Modells zu optimieren.**
- **Ein Optimierungsschritt wird auf Basis eines Batches durchgeführt.**

Epoche

- **Das Modell wurde einmal anhand aller Fälle bzw. aller zu einem Trainingsdatensatz existierenden Batches trainiert.**

Je nach Modell kann es notwendig sein, es über mehrere hundert oder tausend Epochen zu optimieren.

NORMALISIERUNG

Definition:

- **Abziehen des Mittelwerts und Teilen durch die Standardabweichung.**

Erleichtert es dem neuronalen Netz, die benötigten Parameter zu erlernen, indem alle Werte in einem ähnlichen Wertebereich liegen.

BATCH-NORMALISIERUNG

- **Durchführung der Normalisierung auf Batch-Ebene**

Zusätzliche Optimierung:

- **exakt gleiche Mittelwerte und Standardabweichungen sind nicht notwendigerweise optimal im Sinne der Modellierung**
- **Einfügen der Normalisierungsparameter als trainierbare Parameter**

LERNMATERIAL

- [Dieses Video](#) (7 Minuten) schauen, um die Eigenschaften von Dropout-Layern genauer zu verstehen.
- [Dieses Video](#) (5 Minuten) schauen, um die Vorteile der Normalisierung besser zu verstehen.

AUFGABEN

- **Untersucht alle Eure Modellvariablen auf die Existenz von fehlenden und unplausiblen Werten**
- **Trainiert ein erstes neuronales Netz für Euren Datensatz (Löscht dazu zunächst alle Zeilen mit fehlenden Werten)**