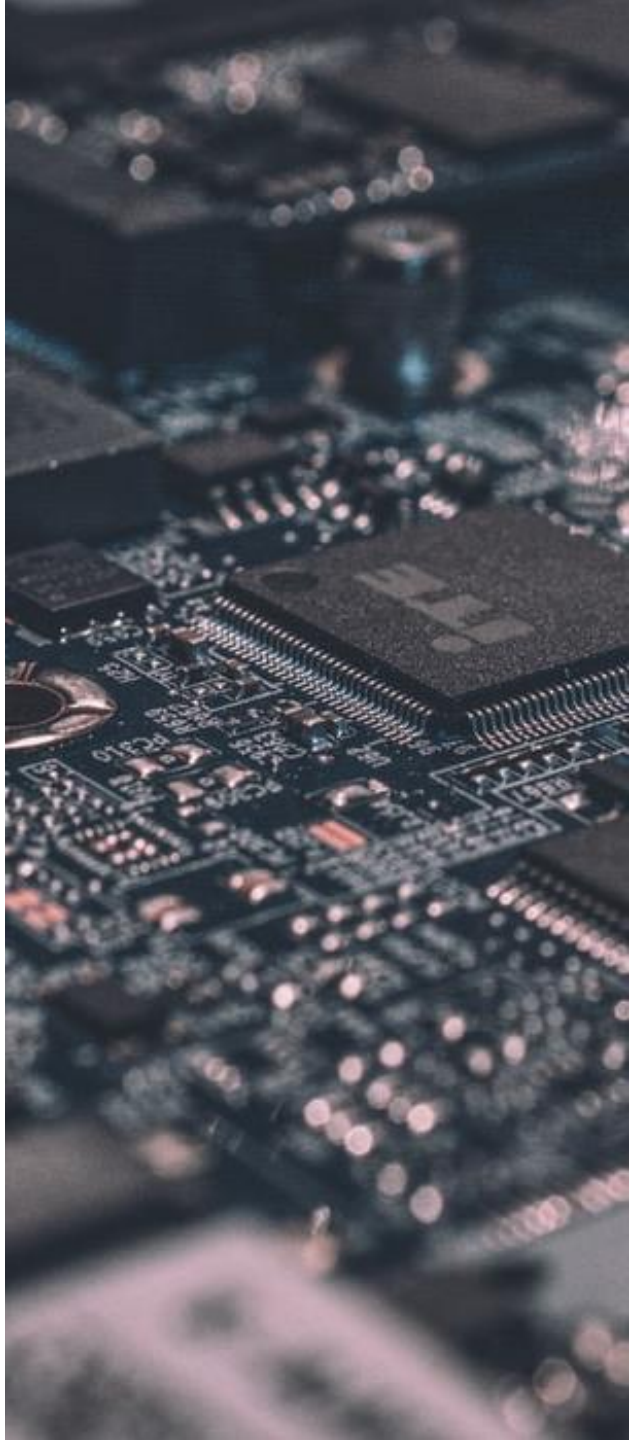
Applied Data Science Projektarbeit
«Vivino für Autos»

Applied Data Science
Studenten: Tim Baenziger, Philippe Fuhrer, Silvan Kirchhofer
Prüfende Dozentin: Maria Pelli
FS2021

Agenda

I. Einleitung
II. Methode & Vorgehensweise
III. Base Model
IV. Transfer Learning Model
V. Price Prediction
VI. Ethische Fragestellungen
VII. Beantwortung der Forschungsfrage
VIII. Schlussfolgerungen
IX. Quellenverzeichnis



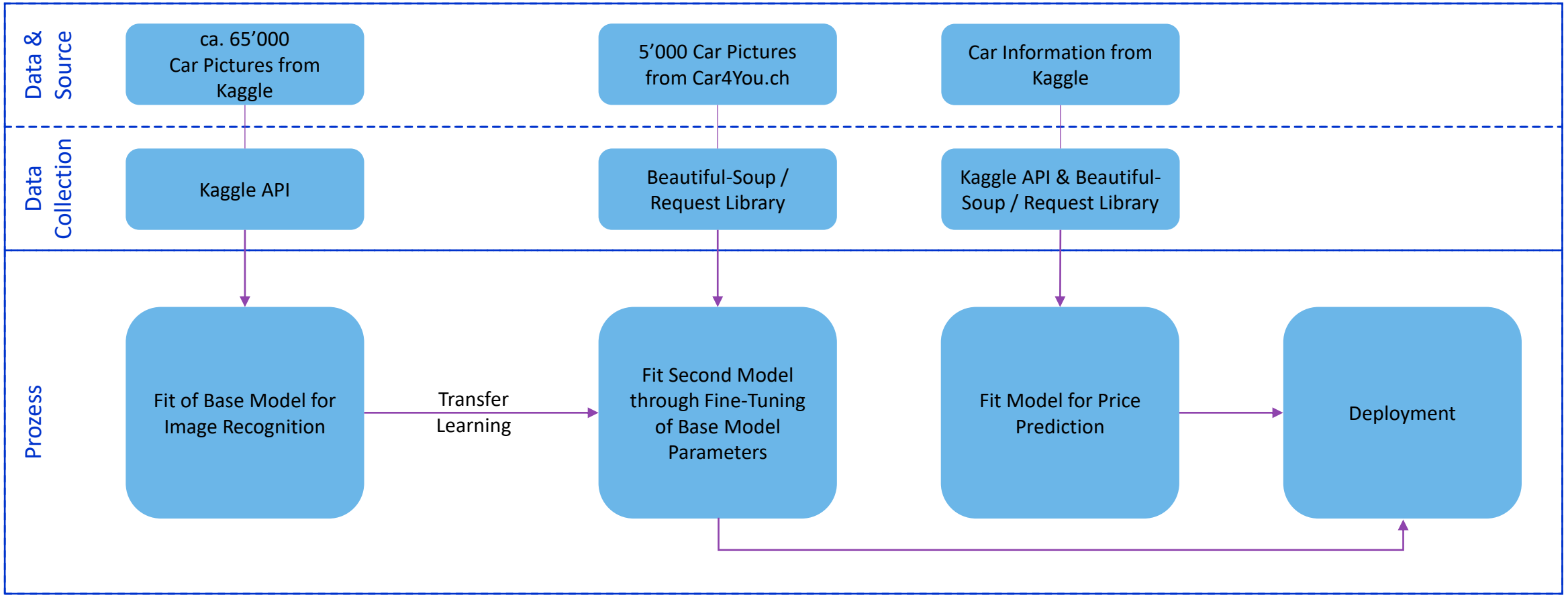
Einleitung

- Hintergrund
  - Fortschritte im Bereich AI erlaubt es Automarken zu erkennen
  - Die Preisbestimmung von Autos
    - Wichtig für die Automobilbranche & für Privatpersonen
    - Kann automatisiert werden
  - Entwicklung einer App im Stil von “Vivino”

- Problemstellung
  - Erkennung der Automarke basierend auf Bildern sowie Bestimmung des Preises aufgrund weiterer Attribute
- Zielsetzung
  - Optimierung der Bilderkennung von Automarken via CNN und Transfer Learning
  - Zusätzlich: Preisbestimmung aufgrund von weiteren Attributen
- Forschungsfrage
  - Kann basierend auf dem Kaggle Car-Dataset eine genaue Bild-Klassifikation von ausgewählten Auto-Marken sowie eine Bestimmung der Verkaufspreise für den Schweizer Markt gemacht werden?

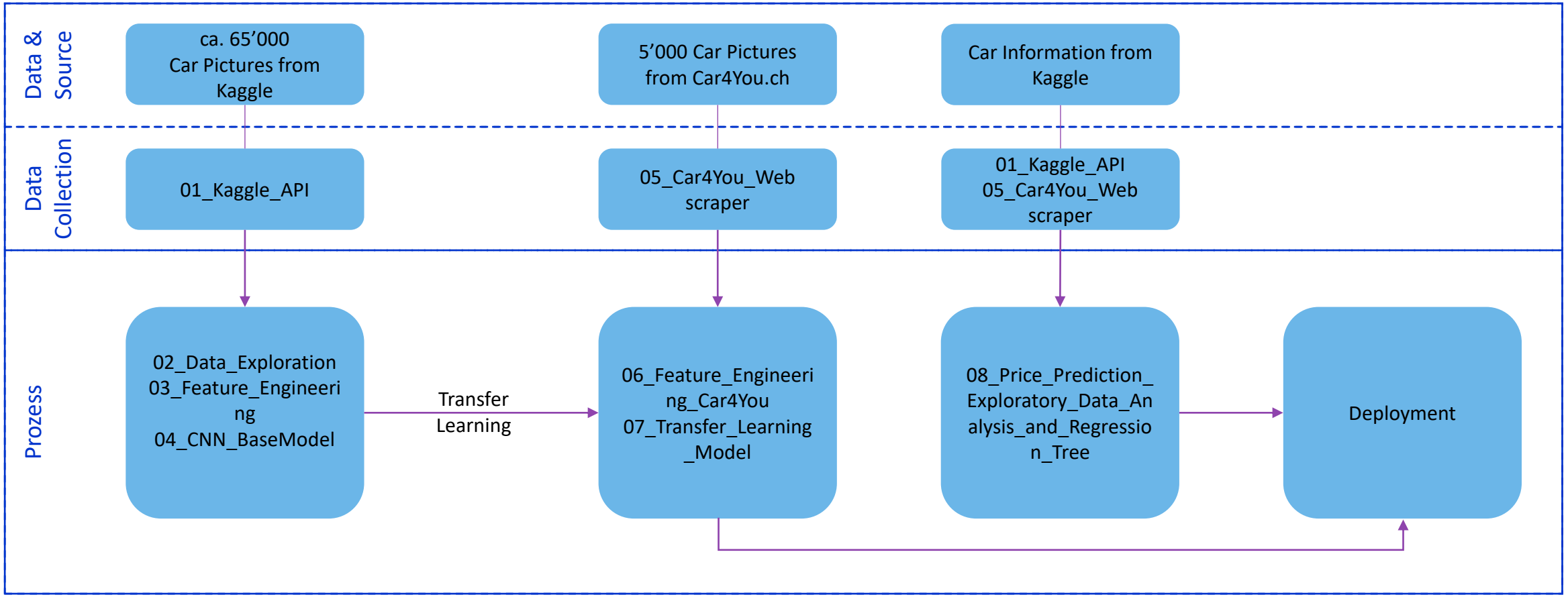
Methodik & Vorgehensweise

Überblick der Vorgehensweise



Methodik & Vorgehensweise

Namen der Jupyter Notebooks im Überblick



Methodik & Vorgehensweise

Kollaboration & Data Sources

Data Sources: Kaggle API + Webscraping

Beautiful Soup

kaggle +

=

Kollaboration & Versionierung: Google Colab Pro & GitHub

Colab Pro

GitHub

Base Model

Datenbezug über kaggle API

kaggle

API

Using Kaggle's beta API, you can interact with Competitions and Datasets to download data, make submissions, and more via the command line. Read the docs

- kaggle.json -> API Key
- 690 MB .zip
- Ca 1.5 GB Image .jpg

Download Dataset über Kaggle API

```
[ ] !pip install -q kaggle
    !pip install -q kaggle-cli
    !mkdir -p ~/.kaggle
    !cp "/content/gdrive/MyDrive/Kaggle/kaggle.json" ~/.kaggle/
    !cat ~/.kaggle/kaggle.json
    !chmod 600 ~/.kaggle/kaggle.json
    # For competition datasets
    #!kaggle competitions download -c dataset_name -p download_to_folder
    # For other datasets
    !kaggle datasets download -d prondeau/the-car-connection-picture-dataset -p /content/gdrive/MyDrive/Kaggle/Dataset
```

Unzip Downloaded data

```
[ ] !apt install unzip
```

```
[ ] !unzip  /content/gdrive/MyDrive/Kaggle/Dataset/the-car-connection-picture-dataset.zip -d /content/gdrive/MyDrive/Kaggle/Dataset
```

Base Model

Data Exploration

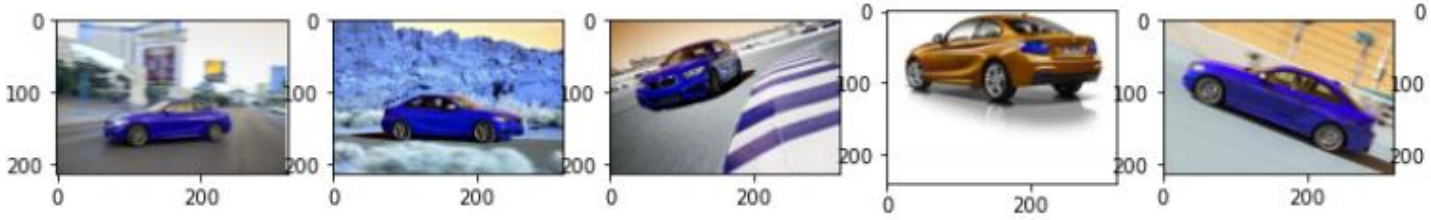
- 64’467 Auto-Bilder
- 20 Marken
- Analyse der Bilder
  - RGB
  - Unterschiedliche Grössen
  - Unterschiedliche Perspektiven
  - Achtung «unbrauchbare» Bilder

```
print(data.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 64467 entries, 0 to 64466
Data columns (total 6 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   src         64467 non-null  object
 1   brand       64467 non-null  object
 2   model       64467 non-null  object
 3   year        64467 non-null  object
 4   MSRP        64467 non-null  object
 5   Horespower  64467 non-null  object
dtypes: object(6)
memory usage: 3.0+ MB
None
```

```
data['brand'].value_counts().head(20)
```

```
Chevrolet        5079
Toyota           4598
Ford             4416
BMW              4121
Nissan           3881
Audi             3131
Mercedes-Benz    3097
Honda            2675
Kia              2160
Lexus            2125
Hyundai          2091
GMC              2067
Volkswagen       1752
Subaru           1605
Mazda            1475
Dodge            1345
Porsche          1344
Lincoln          1324
Cadillac         1311
Volvo            1231
Name: brand, dtype: int64
```

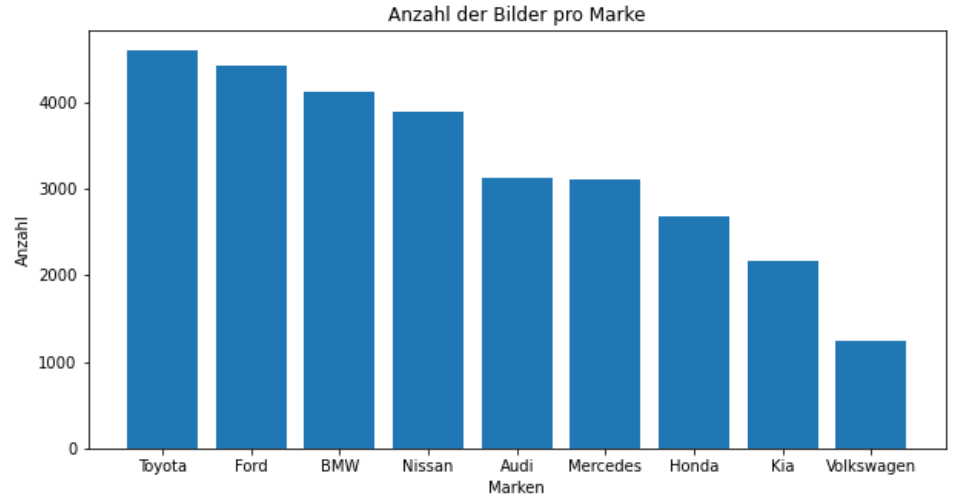


Base Model

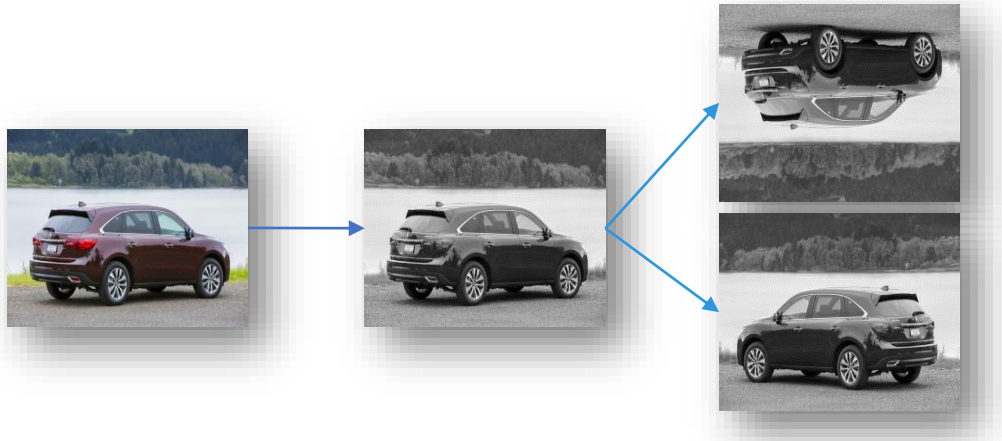
Processing der Daten & Feature Engineering

- Beschränkung auf 10 Auto-Marken
- Feature Engineering
  - Grey Scaling
  - Resizing (200, 300)
  - Speichern der Daten in einem Numpy-Array

1. VW
2. Toyota
3. Ford
4. BMW
5. Nissan
6. Audi
7. Mercedes
8. Honda
9. Kia
10. Volvo

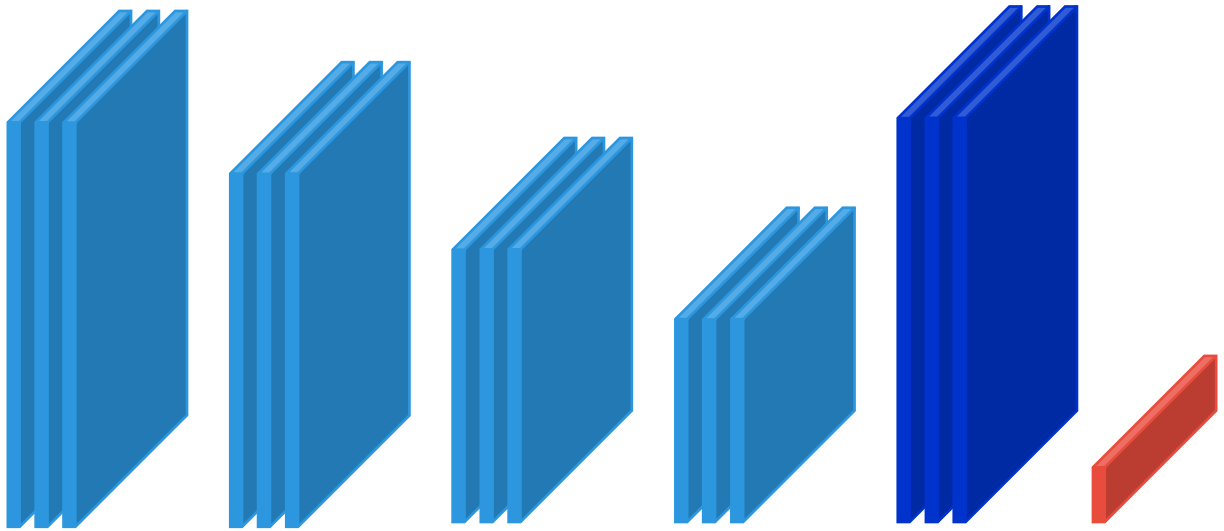



```
[ ]  X= []
     y= []
     for i in range(len(data)):
         src = data.loc[i,'src']
         src = cv2.imread(src, cv2.IMREAD_COLOR)
         dst = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
         X.append(cv2.resize(dst, dsize=(200, 300), interpolation=cv2.INTER_AREA))
         y.append(data.loc[i,'brand'])

     fig, axes = plt.subplots(1,10,figsize=(25,10))
     for i in range(0,10):
         axes[i].imshow(X[i])
```



Base Model

Architektur



Conv2D	Conv2D	Conv2D	Conv2D	Flatten	Dense
MaxPool	MaxPool	MaxPool	MaxPool	Dense	
Dropout	Dropout	Dropout	Dropout	Dropout	

```
model.summary()

Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 296, 196, 32)      832
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 148, 98, 32)       0
_________________________________________________________________
dropout_15 (Dropout)         (None, 148, 98, 32)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 148, 98, 32)       25632
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 74, 49, 32)        0
_________________________________________________________________
dropout_16 (Dropout)         (None, 74, 49, 32)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 74, 49, 64)        18496
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 37, 24, 64)        0
_________________________________________________________________
dropout_17 (Dropout)         (None, 37, 24, 64)        0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 37, 24, 64)        36928
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 18, 12, 64)        0
_________________________________________________________________
dropout_18 (Dropout)         (None, 18, 12, 64)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 13824)             0
_________________________________________________________________
dense_6 (Dense)              (None, 256)               3539200
_________________________________________________________________
dropout_19 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_7 (Dense)              (None, 10)                2570
=================================================================
Total params: 3,623,658
Trainable params: 3,623,658
Non-trainable params: 0
_________________________________________________________________
```

Die Architektur des Base Models basiert auf der bekannten VGG16 Architektur, wurde jedoch mangels Rechenleistung etwas verschlankt.

Base Model

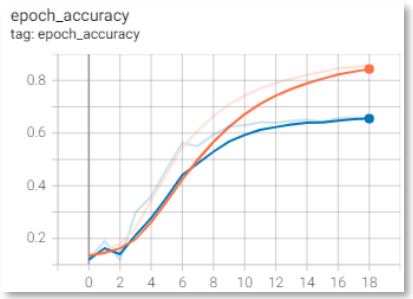
Finetuning

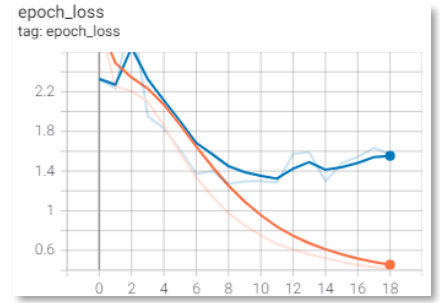
Finetuning Aktion	Resultat
Anzahl Layers erhöht	Accuracy sinkt
Optimizer von rmsProp zu adam	Accuracy steigt
Activation von relu auf elu	Overfit sinkt
Anpassen von learning rate	Keine Veränderung
Verschiedene Regulizer	Keine Veränderung
Dropout Layer	Overfit sinkt
Batch size verkleinert	Accuracy sinkt
Kernel Anzahl vergrössert und verkleiner	Accuracy sinkt Overfitt steigt

```
input_shape = X_train.shape[1:]
model = models.Sequential()

#CNN
model.add(layers.Conv2D(filters = 32, kernel_size = (5,5), activation ='elu', input_shape = X_train.shape[1:]))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same', activation ='elu'))
model.add(layers.MaxPool2D(pool_size=(2,2)))
model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same', activation ='elu'))
model.add(layers.MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(layers.Dropout(0.25))

model.add(layers.Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same', activation ='elu'))
model.add(layers.MaxPool2D(pool_size=(2,2), strides=(2,2)))
model.add(layers.Dropout(0.25))

#dense
model.add(layers.Flatten())
model.add(layers.Dense(256, activation = "relu"))
model.add(layers.Dropout(0.5))

model.add(layers.Dense(10, activation = "softmax")) # set number of outputs

optimizer = RMSprop(learning_rate=0.001, rho=0.9, epsilon=1e-08, decay=0.0)

model.compile(loss=keras.losses.categorical_crossentropy, optimizer=optimizer , metrics=['accuracy'])
```

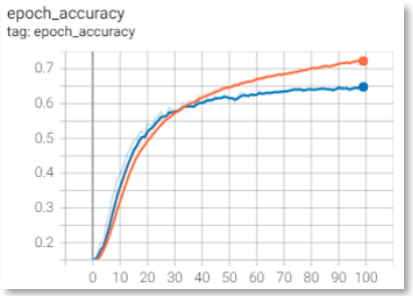
```
datagen = ImageDataGenerator(
        featurewise_center=False,  # set input mean to 0 over the dataset
        samplewise_center=False,  # set each sample mean to 0
        featurewise_std_normalization=False,  # divide inputs by std of the dataset
        samplewise_std_normalization=False,  # divide each input by its std
        zca_whitening=False,  # apply ZCA whitening
        #rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
        #zoom_range = 0.1, # Randomly zoom image
        #width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
        #height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
        horizontal_flip=True,  # randomly flip images
        vertical_flip=True)  # randomly flip images


datagen.fit(X_train)
```

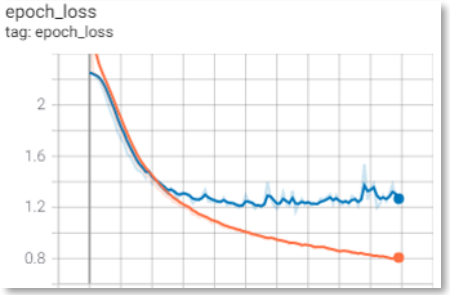
Epochs	Accuracy	Loss	Augmentation
19/100	0.593	1.5	No





Epochs	Accuracy	Loss	Augmentation
100	0.65	1.2	Yes

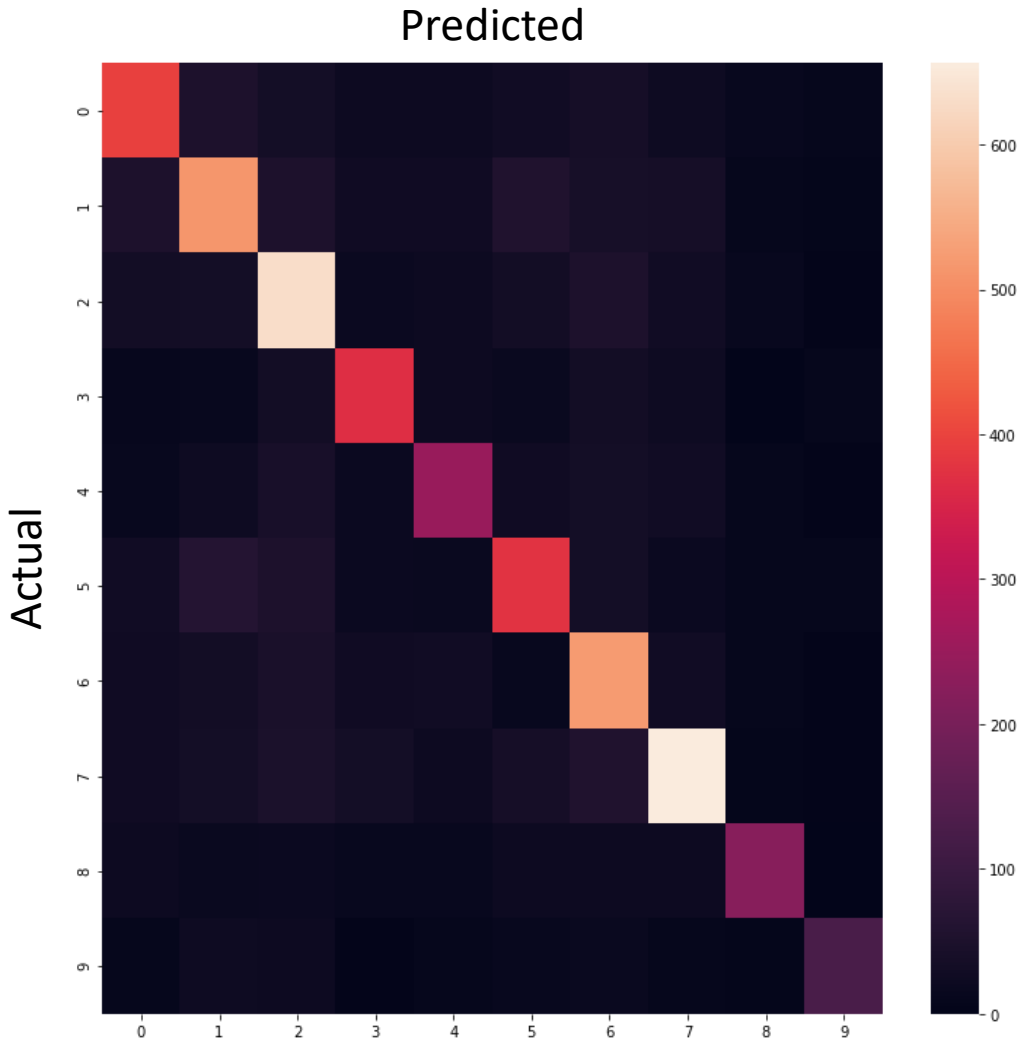




Base Model

Ergebnisse

- Hohe Rechenleistung erforderlich
- Hohe Anforderung an RAM
- Anfangs hoher Overfit
- Data Augmentation
  - Horizontal & Vertical Flip
  - Weniger Overfit
  - Bessere Accuracy
- Accuracy nie über 65%

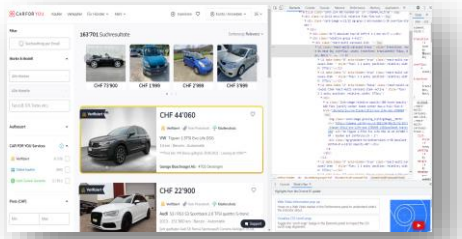


Transfer Learning Model

Webscraping für Second Model und Processing der Daten

Die Daten für das Finetuning des Base Models wurden mit einem Scraper von Car4You extrahiert und anschliessend ein Data-Cleansing durchgeführt.

CAR FOR YOU



Beautiful Soup

Requests



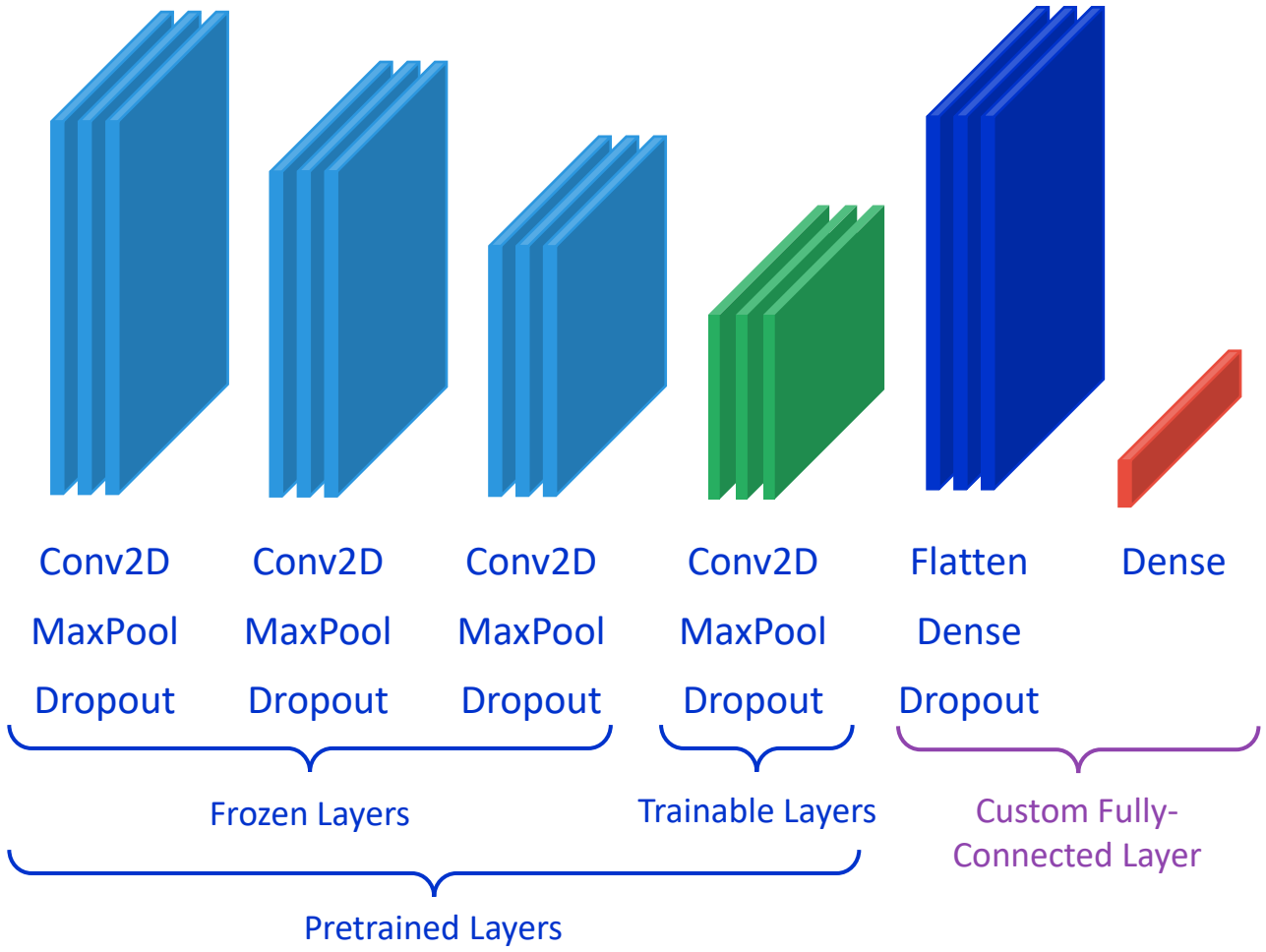


```
import requests
from bs4 import BeautifulSoup
import urllib.request
import csv

f = open("_car_info_scraped_from_car4you_V2.csv", "w", newline="")
thewriter = csv.writer(f)
thewriter.writerow(["brand", "model", "km", "fuel_type", "price"])

carbrands = ["vw", "toyota", "ford", "bmw", "nissan", "audi", "mercedes-benz", "honda", "kia", "volvo"]
chars_to_remove=["/","*"]

for brand in carbrands:
    page = 0
    while page <= 100:
        try:
            url = "https://www.carforyou.ch/de/auto/"+brand+"?page="+str(page)
            response = requests.get(url)
            soup = BeautifulSoup(response.content, "lxml")
            images = soup.find_all("img")

            ## price
            prices = soup.find_all("p", class_ = "text-grey-dark leading-sm font-bold w-12/12")
            #price = prices[3].text
            ## info
            infos = soup.find_all("p", class_ = "text-grey-4 md:text-md leading-xs pb-14")
            #info = infos[3].text
            ## model
            models = soup.find_all("h1", class_ = "text-md leading-xs text-grey-dark font-regular mb-10")
            #model = models[3].text

            count = 0
            for image in images:
                text_for_name = str(brand+","+str(models[count].text.replace("/",","))+","+str(infos[count].text.replace(" · ",","))+","+str(prices[count].text[4:])+","+str(count))
                for character in chars_to_remove:
                    text_for_name = text_for_name.replace(character, "")
                image_src = image["src"]
                thewriter.writerow([text_for_name])
                urllib.request.urlretrieve(image_src, text_for_name+".jpg")
                count = count+1
        except:
            break
        page = page+1
```

Nachfolgend wurden die Daten analog zum Base Model behandelt -> Beschränkung auf 10 Automarken und Feature Engineering

Transfer Learning Model

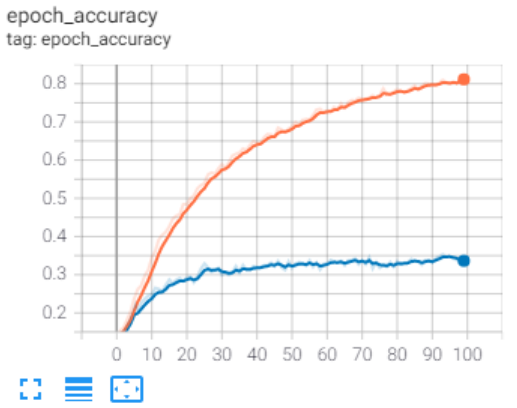
Architektur

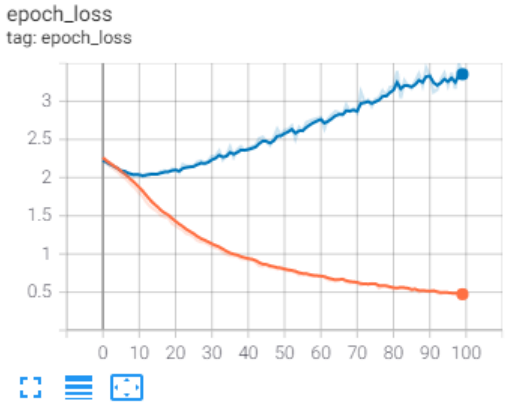



```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
module_wrapper_9 (ModuleWrap (None, 10)                3623658
_________________________________________________________________
flatten_9 (Flatten)          (None, 10)                0
_________________________________________________________________
dense_18 (Dense)             (None, 256)               2816
_________________________________________________________________
dropout_9 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_19 (Dense)             (None, 10)                2570
=================================================================
Total params: 3,629,044
Trainable params: 3,584,084
Non-trainable params: 44,960
_________________________________________________________________
```

Transfer Learning Model

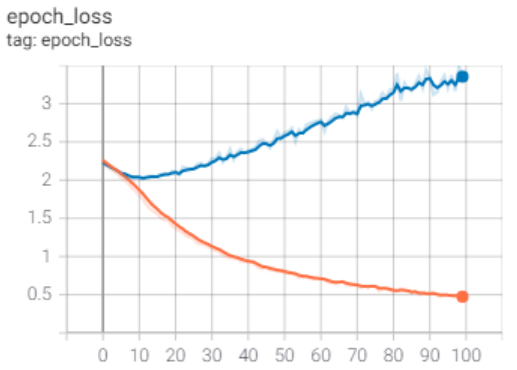
Finetuning

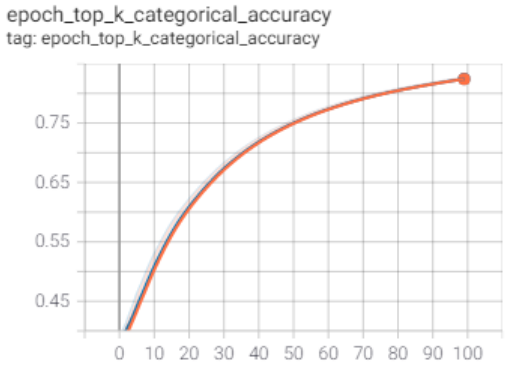




epoch_loss

epoch_top_k_categorical_accuracy





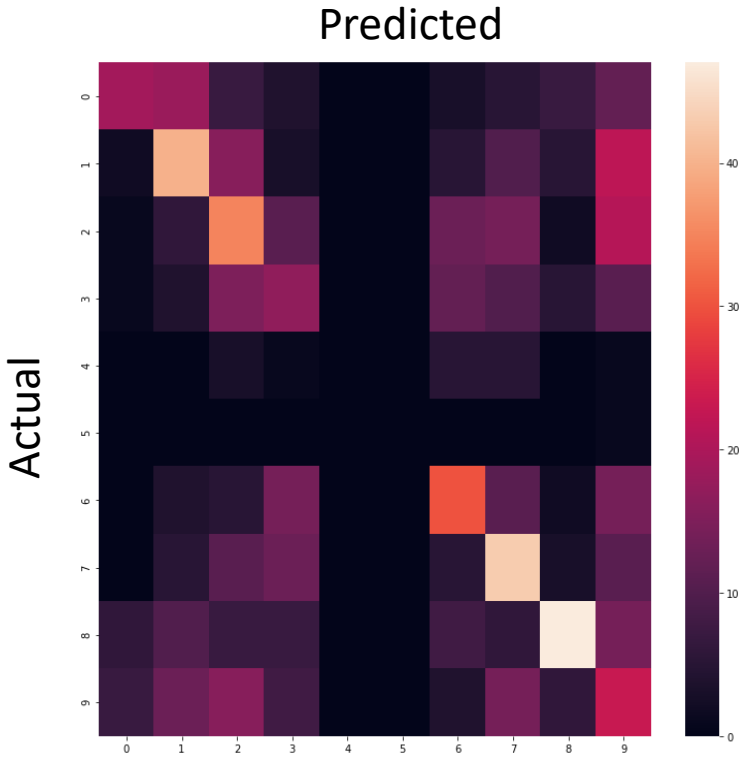
Finetuning Aktion	Resultat
Anzahl der trainierbaren Layers vergrössert/verkleinert	Accuracy wurde kleiner (9 trinable layers wurden als optimal identifiziert)
Veränderung der Activation Function von relu auf elu	Etwas bessere Accuracy aber vor allem schnellere Optimierung
Anzahl Dense Layers erhöht	Accuracy wurde kleiner
Data Augmentation	Keinen spürbaren Einfluss auf accuracy – jedoch auf Overfit
Anpassen von learning rate	Accuracy bleibt gleich
Verschiedene Regulizer	Keine Veränderung

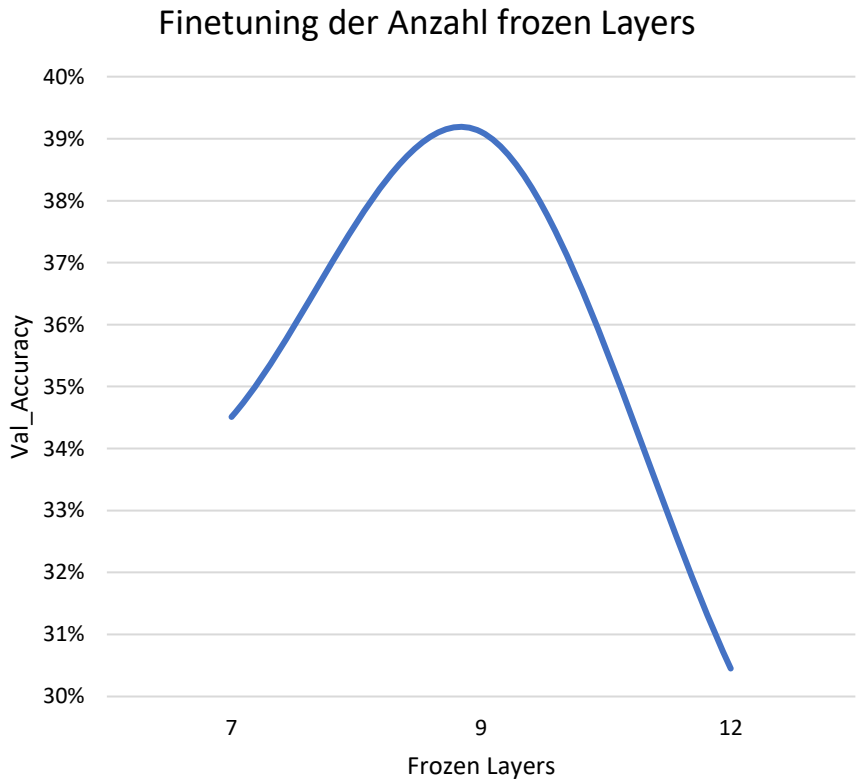
Trotzdem, dass zahlreiche Stunden investiert wurden, um das extreme Overfit-Problem zu lösen, konnte keine definitive Lösung gefunden werden.

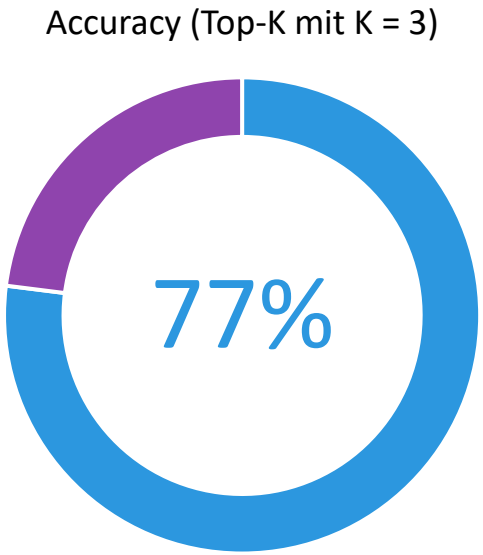
Durch die Eliminierungs-Methode kamen wir zum Schluss, dass das Data-Set zu klein für die Anwendung ist.

Transfer Learning Model

Ergebnisse





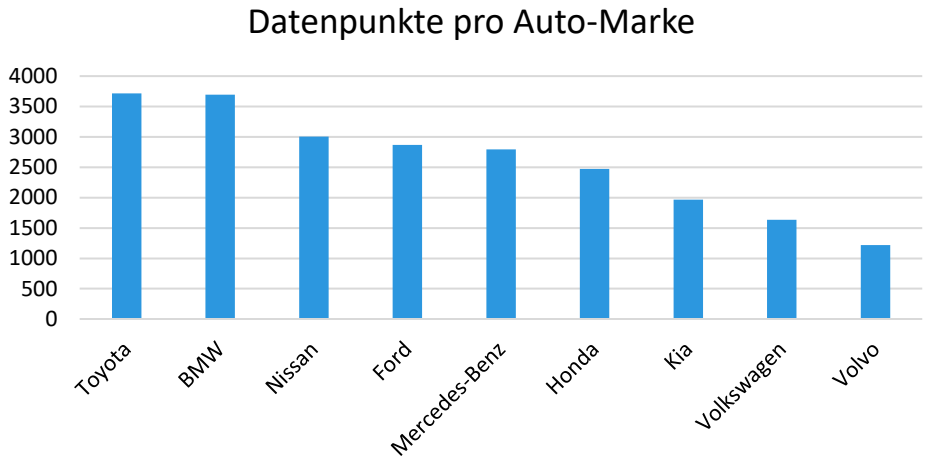



```
Epoch 100/100
12/12 [==============================] - 1s 116ms/step - loss: 0.4095 - accuracy: 0.8676 - val_loss: 2.6842 - val_accuracy: 0.3911
```

-> Val_Accuracy von 40% für eine Auto-Erkennungs-App nicht ausreichend.

Price Prediction

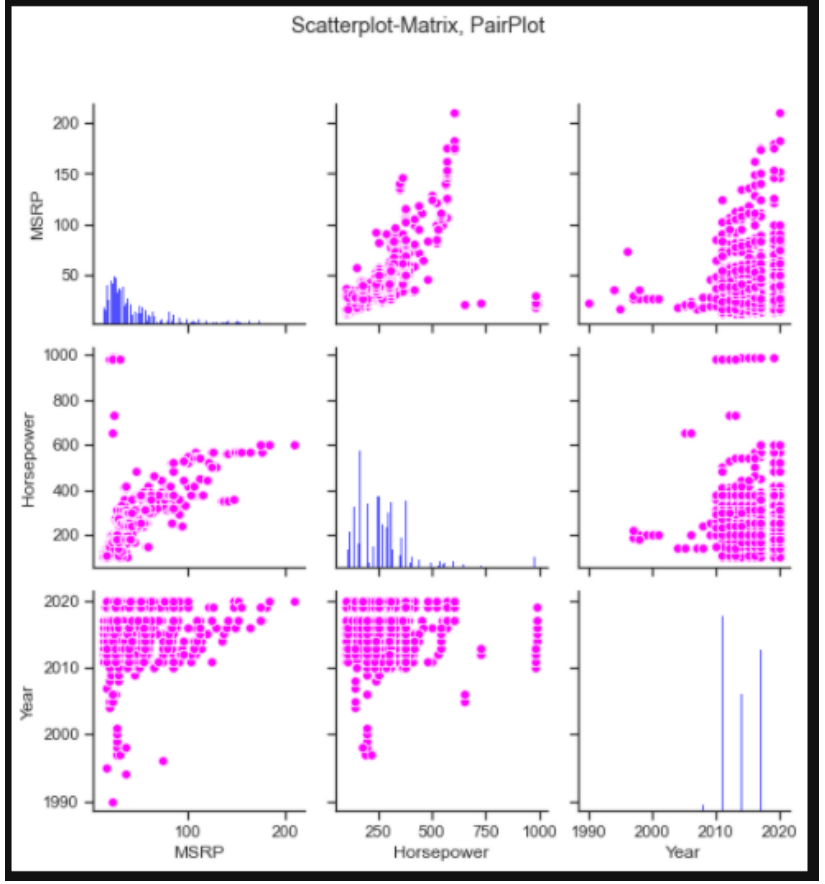
Exploratory Data Analysis




```
import seaborn as sns
sns.set_theme(style="whitegrid")
ax = sns.boxplot(data=df3[["MSRP"]], orient="h", palette="Set2").set_title("Distribution of Manufac
```


```
import seaborn as sns
sns.set(style="ticks")
g = sns.PairGrid(df3[['MSRP', 'Horsepower', 'Year']], height=2.5, aspect=1)
g.fig.suptitle("Scatterplot-Matrix, PairPlot", y=1.08) # y= some height>1
g.map_upper(sns.scatterplot,color='magenta')
g.map_lower(sns.scatterplot, color='magenta')
g.map_diag(sns.histplot, color='blue')
```

```
<seaborn.axisgrid.PairGrid at 0x19ebc572490>
```



Price Prediction

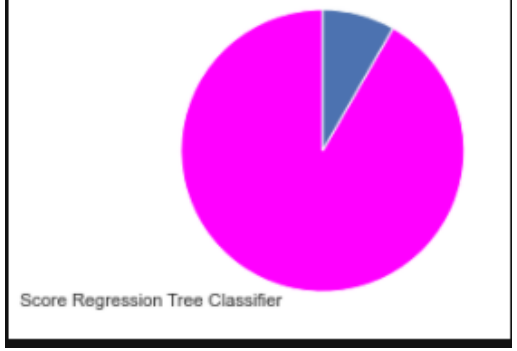
Model Architektur und Ergebnisse

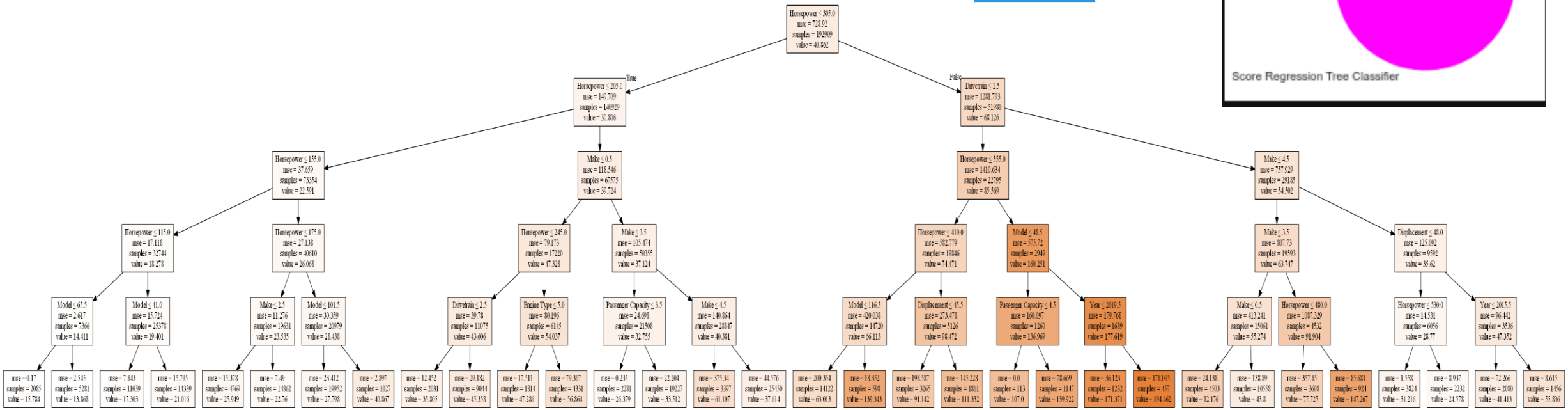
Model	Regression Tree Classifier von Sklearn
Max Depth	5 “keep it simple”

Regression Tree Classifier Score:

91%

```
import matplotlib.pyplot as plt
import numpy as np

y = np.array([score, 1-score])
mylabels = ["Score Regression Tree Classifi
mycolors = ["magenta", "b"]

plt.pie(y, labels = mylabels, startangle =
plt.show()
```





Ethische Fragestellungen

1. Datenschutz?
    - Keine personenbezogenen Daten (keine Personen oder Nummernschilder erkennbar)
    - Daher in Puncto Datenschutz unproblematisch
    - Scraping ist Datenschutz-Grauzone (Urheberrecht)
2. Weitere Gefahrenpotenziale?
    - Verstärkung Klassendenken (Preistransparenz Autos)
    - Preis wird falsch kalkuliert (Konfliktpotenziale)
    - Fremde Autos werden fotografiert (Konfliktpotenziale)

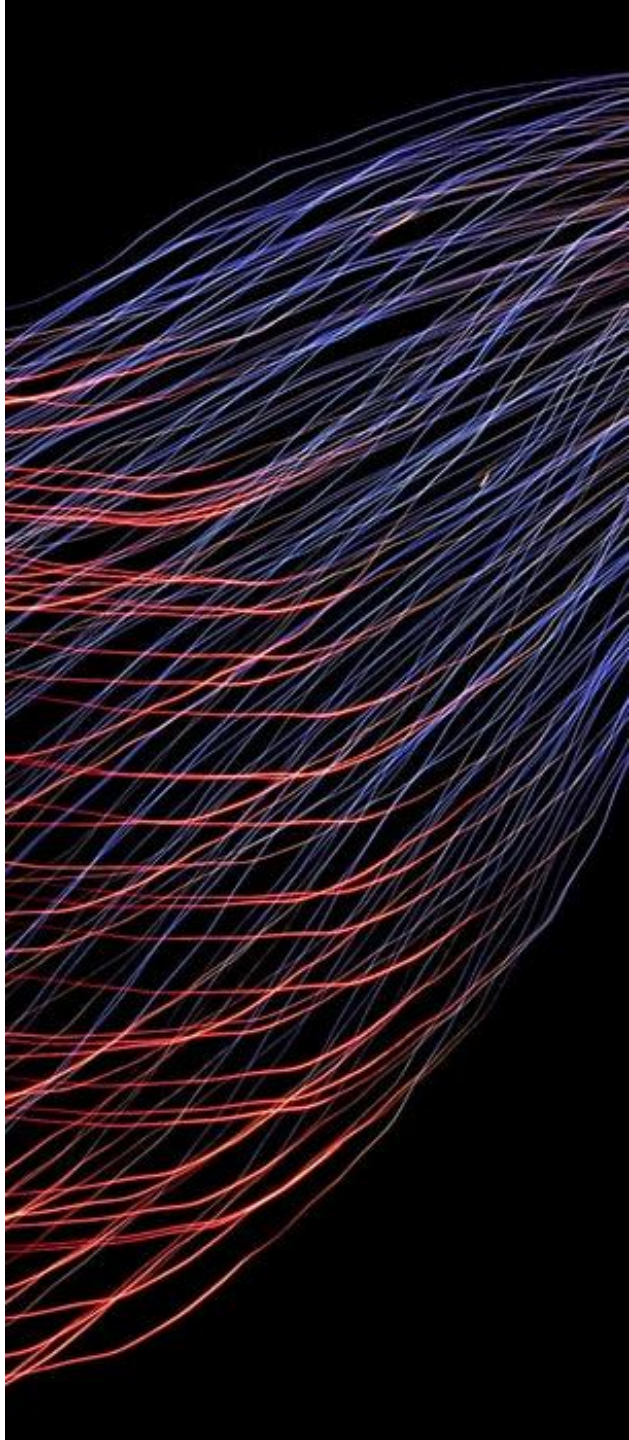
Beantwortung Forschungsfrage

Recap Forschungsfrage:

- Kann basierend auf dem Kaggle Car-Dataset eine genaue Bild-Klassifikation von ausgewählten Auto-Marken sowie eine Bestimmung der Verkaufspreise für den Schweizer Markt gemacht werden?

Antwort:

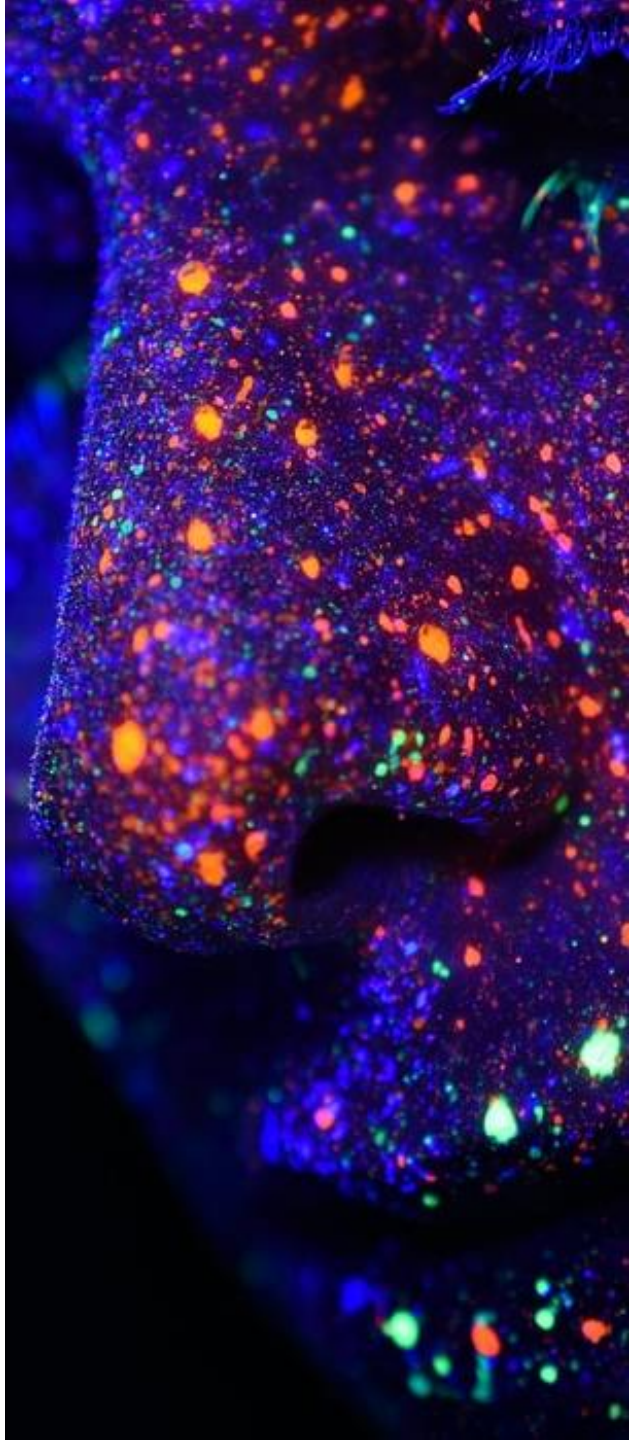
- Basierend auf dem Kaggle Car-Dataset konnte keine ausreichend akkurate Bild-Klassifikation für eine vergleichbare Applikation (wie Vivino) erstellt werden. Die Primären Gründe sind:
  - Mangelnde Datenqualität des Datasets
  - Unzureichende Rechenleistung trotz kostenpflichtiger Colab-Pro Version
  - Homogene Optik von Autos
- Die Bestimmung der Verkaufspreise konnte hingegen ohne grossen Aufwand umgesetzt werden



Schlussfolgerungen

Grundlegende Problemstellungen und Learnings

- Initiales Ziel zu ambitiös und ohne stärkere Rechenleistung nicht erreichbar (Stichwort: Ikarus)
- Viele Learnings aber auch ungelöste Problemstellungen
- Base Model: Datengrundlage heterogen & noisy
  - Das Kaggle Car-Dataset enthält viele «unbrauchbare» Bilder (Bilder von Rückspiegeln oder dem Interieur etc.)
  - Data-Set bereinigen -> hoher Aufwand
- Transfer Learning Model: Datengrundlage zu klein
- Data Augmentation noch experimentell



Quellenverzeichnis

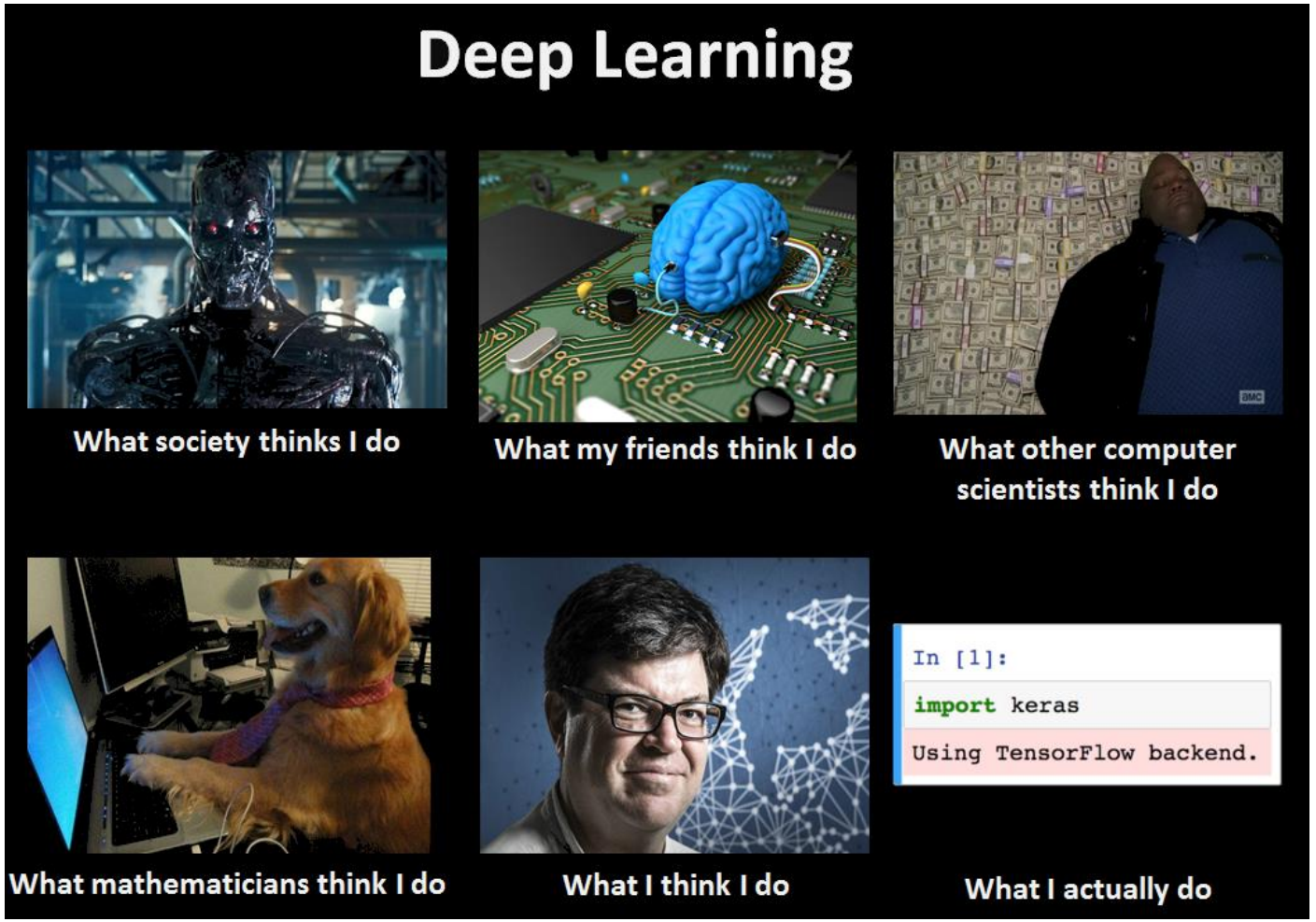
Allgemeine Quellen

- Kaggle 60’000 Images of Cars
- Tensorflow ImageData Generator
- Tensoforflow Conv2D
- Car4You Webpage
- Stackoverfolw (mind. 50 versch. Links)
- Medium
- Keras Documentation

GitHub Repository

- https://github.com/PhilippeFuhrer/Car_Image_Recognition_CNN_With_Kaggle_Car_Dataset

Q&A und Danke!


Deep Learning
What society thinks I do
What my friends think I do
What other computer scientists think I do
What mathematicians think I do
What I think I do
In [1]:
import keras
Using TensorFlow backend.
What I actually do