**Udacity Machine Learning Nanodegree**

**Capstone Project**

Andreas Börzel

March 25, 2020

**German License Plate Recognition**

**Definition**

**Project Overview**

Machine learning methods are used today in many different areas. One particularly exciting and important area is machine vision, which is about the ability to understand and interpret image content with a computer. The range of tasks is roughly divided into classification, object detection and localization, segmentation and recognition. Today, machine learning, especially deep learning, makes it possible to detect, identify (identity) and localize objects within an image with the help of a computer, and even to read gestures, feelings or behavioral patterns from images - abilities that until a few years ago were reserved for humans alone.

In this project I created a small Android app that can recognize the license plate of a car quickly and easily with the camera of a smartphone or tablet and translate the license into plain text. The app marks the recognized license plate within the camera image with a bounding box and displays the determined license in plain text as annotation above the bounding box, as outlined in the following demo:



**Problem Statement**

The goal is to create a license number detection and recognition for German car license plates, running on Android devices; the tasks involved are the following:

1. Collect data and create a dataset for license plate detection
2. Train a classifier that can detect and localize a car license plate in an image
3. Create a dataset for license number recognition
4. Train a CRNN Model that can recognize the car license number as plain text
5. Create an Android app that detects license plates with the camera and displays the license number in plain text

The application can be used as a basis for many practical, mobile applications that require the license number of a car as an identification feature.

For simplicity, the project is initially limited to the recognition of German license plates. A later extension to European or even worldwide license plates is possible!

**Metrics**

\*\*\* TODO \*\*\*

Accuracy is a common metric for binary classifiers; it takes into account both true positives and true

negatives with equal weight.

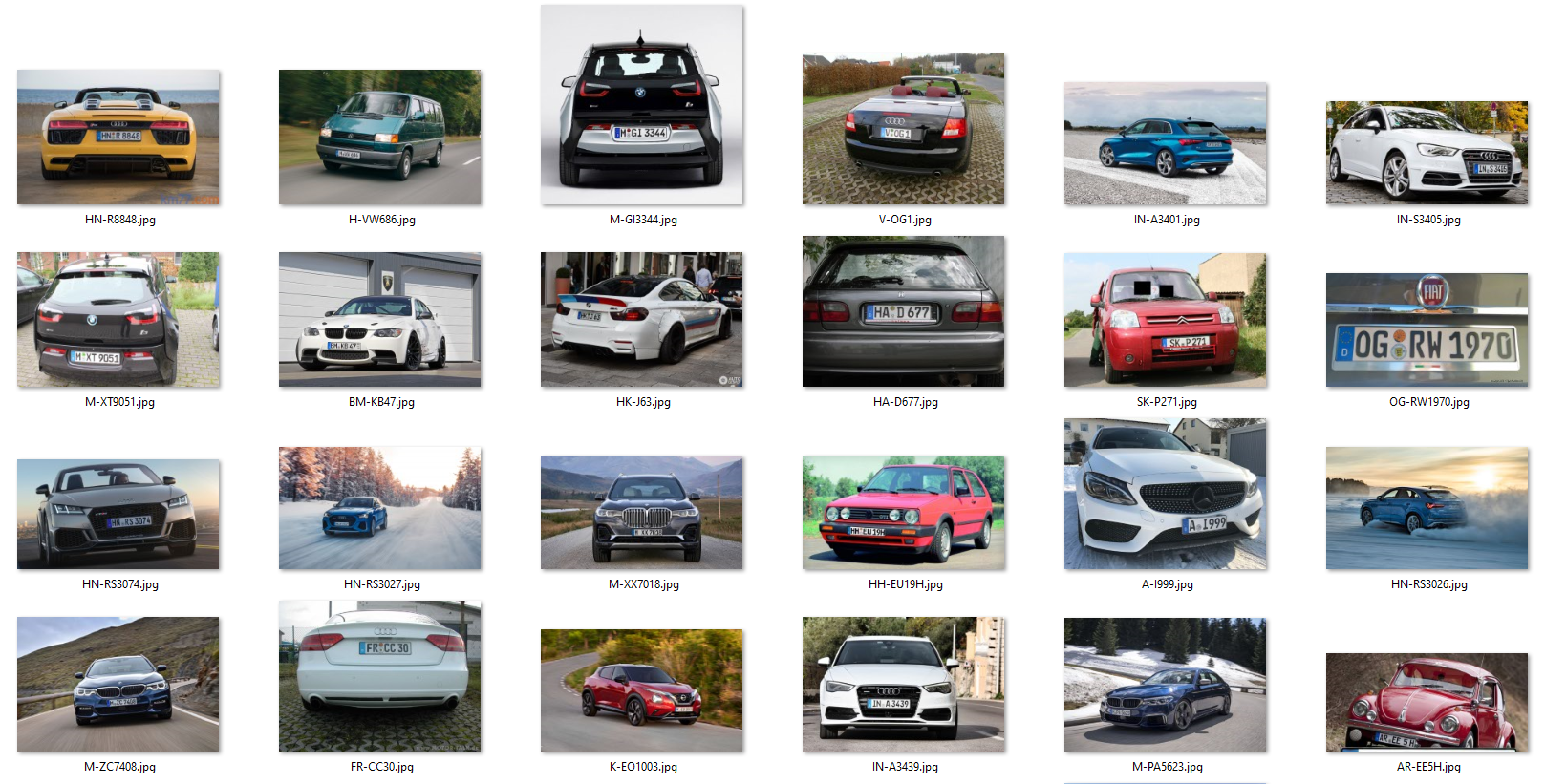
accuracy = dataset size

true positives + true negatives

**Analysis**

**Data Exploration**

**License Plate Detection**

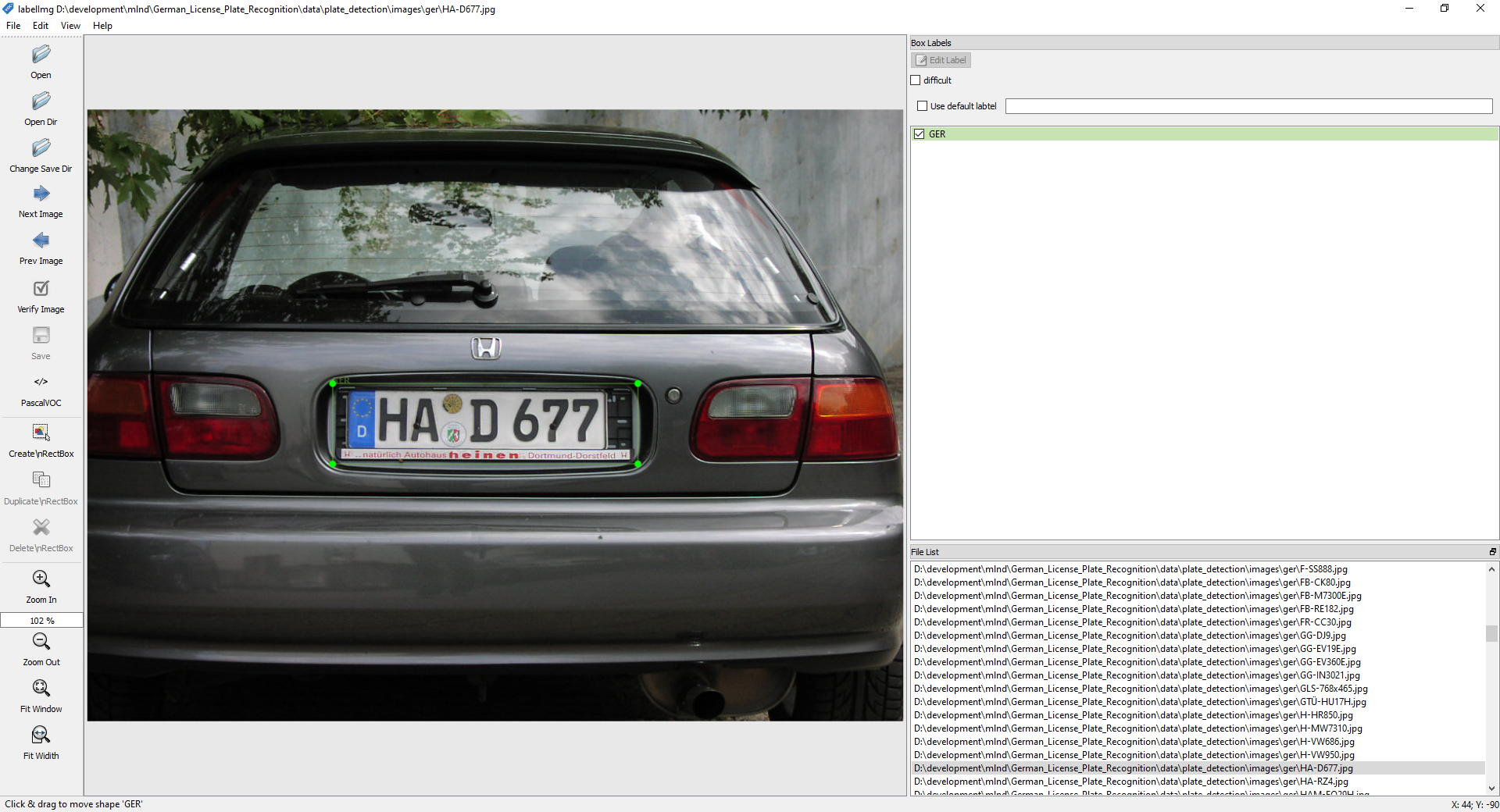
Unfortunately there are no public data collections with pictures of cars with recognizable German license plates. Therefore, a own data collection was created for license plate recognition, with pictures collected from the internet: 

The following criteria were taken into account when selecting the images:

* License number clearly recognisable
* Exclusively German number plates
* Sufficient picture quality (minimum resolution)
* Greatest possible diversity (scale, car brands, colors, backgrounds, weather, license numbers, rotation)

Due to legal reasons it is difficult to get corresponding pictures on the internet, so I could only find 211 pictures which meet the above mentioned criteria.

LabelImg…



Annotationen im PASCAL VOC Format (Label + Location)

Beispiel: HA-D677.xml

<annotation>

<folder>**ger**</folder>

<filename>**HA-D677.jpg**</filename>

<path>**D:\development\mlnd\German\_License\_Plate\_Recognition\data\plate\_detection\data\ger\HA-D677.jpg**</path>

<source>

<database>**Unknown**</database>

</source>

<size>

<width>**1024**</width>

<height>**768**</height>

<depth>**3**</depth>

</size>

<segmented>**0**</segmented>

<object>

<name>**GER**</name>

<pose>**Unspecified**</pose>

<truncated>**0**</truncated>

<difficult>**0**</difficult>

<bndbox>

<xmin>**308**</xmin>

<ymin>**344**</ymin>

<xmax>**691**</xmax>

<ymax>**445**</ymax>

</bndbox>

</object>

</annotation>

name: GER  
bndbox: x1, y1, x2, y2

From the images and the annotations a data collection in TF-Record format was generated, which was used for training the license plate detection with the TensorFlow Object Detection API.

**License Number Recognition**

To recognize the license plates in plain text, a CRNN model was especially trained, which takes into account the characteristic features of German license plates. Since there were not enough corresponding images available for the training of the model, I decided to create a data collection of generated images and to use data augmentation to create realistic images for the training.

*Structure of German number plates*

Germany is divided into 16 federal states, which in turn are divided into different districts. For each of the districts there are one or more different distinguishing marks consisting of 1 to 3 letters, such as B for Berlin or D for Düsseldorf. There are currently 660 different distinguishing marks:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Autokennzeichen** | **State** | **Stadt/Ort** | **Landkreis/Gemeinde** | **Bundesland** |
| **0** | A | BY | Augsburg | Stadt & Landkreis Augsburg | Bayern |
| **1** | AA | BW | Aalen | Ostalbkreis | Baden-Württemberg |
| **2** | AB | BY | Aschaffenburg | Landkreis & Stadt Aschaffenburg | Bayern |
| **3** | ABG | TH | Altenburg | Landkreis Altenburger Land | Thüringen |
| **4** | ABI | ST | Anhalt-Bitterfeld | Landkreis Anhalt-Bitterfeld | Sachsen-Anhalt |
| **5** | AC | NW | Aachen | Städteregion Aachen | Nordrhein-Westfalen |
| **6** | AE | SN | Auerbach im Vogtland | Vogtlandkreis | Sachsen |
| **7** | AH | NW | Ahaus | Kreis Borken | Nordrhein-Westfalen |
| **8** | AIB | BY | Bad Aibling | Landkreise München & Rosenheim | Bayern |
| **9** | AIC | BY | Aichach | Landkreis Aichach-Friedberg | Bayern |
| **10** | AK | RP | Altenkirchen | Landkreis Altenkirchen (Westerwald) | Rheinland-Pfalz |
| **11** | ALF | NI | Alfeld (Leine) | Landkreis Hildesheim | Niedersachsen |
| **12** | ALZ | BY | Alzenau | Landkreis Aschaffenburg | Bayern |
| **13** | AM | BY | Amberg | Stadt Amberg | Bayern |
| **14** | AN | BY | Ansbach | Landkreis & Stadt Ansbach | Bayern |
| **15** | ANA | SN | Annaberg-Buchholz | Erzgebirgskreis | Sachsen |
| **16** | ANK | MV | Anklam | Landkreis Vorpommern-Greifswald ohne Greifswald | Mecklenburg-Vorpommern |
| **17** | AÖ | BY | Altötting | Landkreis Altötting | Bayern |
| **18** | AP | TH | Apolda | Landkreis Weimarer Land | Thüringen |
| **19** | APD | TH | Apolda | Landkreis Weimarer Land | Thüringen |
| **\* \***  **\*** |  |  |  |  |  |
| **660** | ZZ | ST | Zeitz | Burgenlandkreis | Sachsen-Anhalt |

See [deutsche-autokennzeichen.de](https://laenderkennzeichen.net/deutsche-autokennzeichen/) for full list.

A car license plate consists of the distinctive mark and an identification number of 1 to 2 letters and up to 4 digits such as AB 1234. Between the distinguishing mark and the identification number is the inspection sticker of the last main inspection (HU) as well as the stamp sticker of the licensing authority with its seal and the state. See [Autokennzeichen.de](https://autokennzeichen.de/kennzeichen-aufbau/).

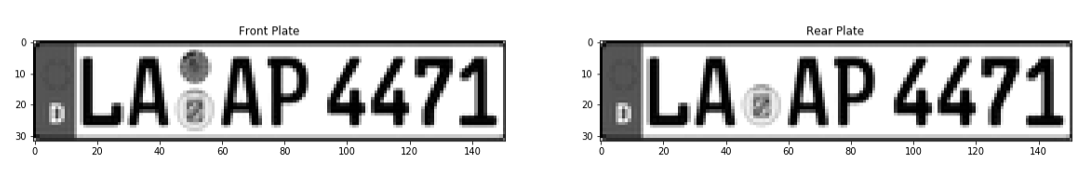
**Example:**



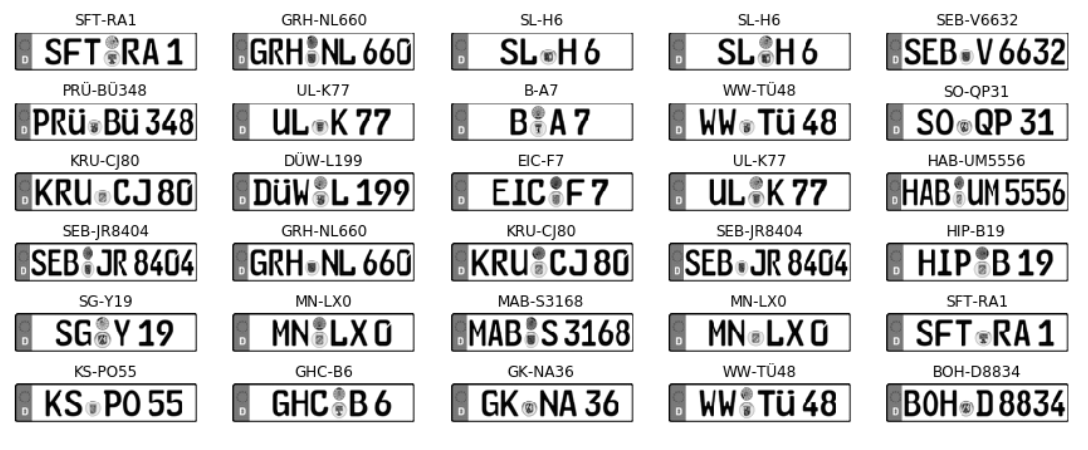
I created a license plate image generator, which generates images of car license plates from randomly generated car licenses, taking into account the regulations for German license plates. The generator uses the [heisnbrg.net](http://nummernschild.heisnbrg.net/fe/) web service to generate an image from a car license number.

The generator observes the following criteria:

* Valid distinguishing marks
* Use only valid characters
* Compliance with format and length restrictions
* Inspection sticker (year/month)
* Stamp sticker of the licensing authority
* Differences between the front and the rear license plate

Example of a generated front and rear license plate:

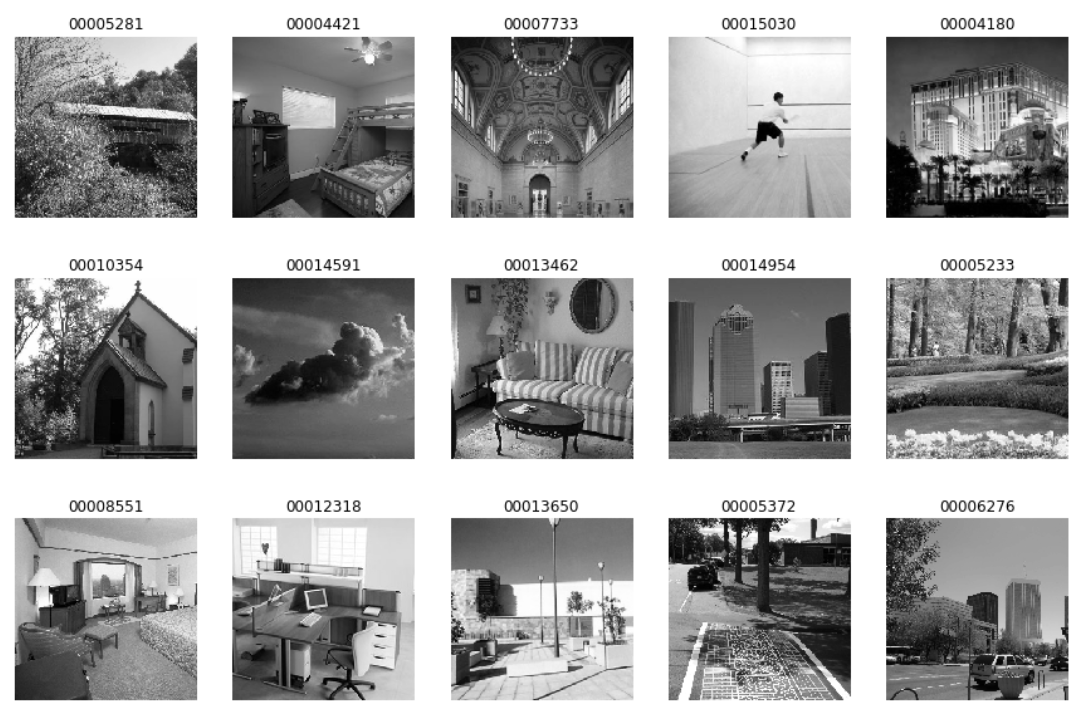
Examples of randomly generated images of car license plates:



For the training of license recognition, 71.533 randomly generated license plates, one image each for the front and rear license plate were generated, i.e. the data collection consists of 143.066 images in total. For easier and faster processing of the images during the training, an HDF5 data set was created from the images together with the labels. This provides easy handling and reduces the I/O load during the training, so that the training process is significantly accelerated.

**Background Images for Data Augmentation**

For the background data collection, 100,000 images were randomly selected from the [SUN2012](https://groups.csail.mit.edu/vision/SUN/) data set and packed into an HDF5 data set.

Here are some examples:

**Algorithms and Techniques**

The goal is to detect license plates within an image and to mark them with a bounding box, as well as to convert and display the license in plain text.

The first task is to detect license plates as such, this is a typical classification problem. The second task is to determine the position and dimensions of the license plate within the image in order to mark it with a bounding box, this is a regression problem. Both tasks can be solved well with the TensorFlow Object Detection API.

Last task is to convert the license into plain text, this is an OCR (Optical Character Recognition) problem, which can be solved with a CRNN (Convolutional Recurrent Neural Network).

Overall it is about supervised learning, i.e. during training the error between the labeled training data (ground truth data) and the model predictions are minimized.

**Benchmark Model**

Since this is a problem of its own and the models are based on data collections that have been created specifically for this purpose, there is unfortunately no possibility of direct comparison with other solutions at first. This means that to evaluate the performance, a part of the data collection is split off as test data, which is only used to evaluate performance with the common performance metrics, but not for training. Likewise, the completed mobile application is subjected to an extensive practical test with real camera images and the result is evaluated accordingly.

**Evaluation Metrics**

In object detection, evaluation is non trivial, because there are two distinct tasks to measure:

* Determining whether an object exists in the image (classification)
* Determining the location of the object (bounding box, a regression task)

The evaluation method used is the “mean average precision” or “mAP score”, this is a commonly accepted evaluation method for object detectors, which has also been used in object detection competitions, such as for the PASCAL VOC, ImageNet, and COCO challenges.

[Here](https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3) is a detailed description about the mAP score.

**Project Design**

The project is roughly divided into 3 tasks:

**Training of a model for detection and localization of license plates in images**

For the detection and localization of license plates I plan to use the [TensorFlow Object Detection API](https://github.com/TensorFlow/models/tree/master/research/object_detection). As the model will be executed from an Android app on a mobile device, the model will be trained in the first approach based on the pre-trained model `ssdlite\_mobilenet\_v2\_coco`, as it has a good balance between `speed` and `accuracy`, see [TensorFlow detection model zoo](https://github.com/TensorFlow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md).

For the data collection about 300 pictures of German license plates and about 300 pictures of other license plates are needed to train the model to distinguish German license plates from other license plates. For this purpose, corresponding images are collected from the internet and labelled with the tool [LabelImg](https://github.com/tzutalin/labelImg). LabelImg stores the label annotations in the [PASCAL VOC](http://host.robots.ox.ac.uk/pascal/VOC/voc2008/htmldoc/) format. From the images and the label annotations a data collection in [TF-Record](https://www.tensorflow.org/tutorials/load_data/tfrecord#tfrecords_format_details) format is generated, which is used by the TensorFlow Object Detection API for the model training.

**Training of a model to transform the license plate image into plain text**

The License Plate Detection above provides an image section which only contains the corresponding license plate. To transform the image section into plain text, another [RCNN](https://zpascal.net/cvpr2016/Lee_Recursive_Recurrent_Nets_CVPR_2016_paper.pdf) (Recurrent Convolutional Neural Network) model is trained. The model is created with [TensorFlow](https://www.tensorflow.org/tutorials/load_data/tfrecord) / [Keras](https://keras.io/) and the [Keras OCR example](https://keras.io/examples/image_ocr/) serves as a template. To achieve good performance on mobile devices, the trained Keras model is converted into a [TensorFlow Lite](https://www.tensorflow.org/lite) model.

For the data collection, synthetic images of car license plates are generated using the web service [heisnbrg.net](http://nummernschild.heisnbrg.net). However, since realistic images are required for training, real images are simulated from the synthetic images using data augmentation. Factors such as scaling, rotation, brightness, interference, etc. must be taken into account. It is planned to use about 60000 license plates for the training. Since front and rear license plates are slightly different, both images will always be created for each license plate, i.e. the data collection will contain about 120000 images. When generating the license plates, you must ensure that there are approximately the same number of license plates for all 30 cities/counties in Germany.

The trained model gets the image section with the recognized license plate as input and delivers the license plate in plain text as output.

**Creating the Android App**

The last step is to create an Android app that enables the user scenario described in the “Problem Statement” section above, using the two trained models. Template for the creation of the app is the Tensorflow Lite [Object Detection example app](https://github.com/tensorflow/examples/tree/master/lite/examples/object_detection/android). As development environment Android-Studio and the programming language Kotlin is used.

*App workflow:*

Camera Image => License Plate Detection => Boundary Box => License Recognition => License Text Annotation

*Tools and libraries:*Python, Numpy, Jupyter-Notebook, Pandas, Scikit-Learn, Scikit-Image, Pillow, Matplotlib, TensorFlow/Keras, TensorFlow Object Detection API, OpenVC, Android-Studio, Kotlin...