Capstone Project
Machine Learning Engineer Nanodegree

Andreas Börzel
March 28th, 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 only.

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 simplification, 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

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.

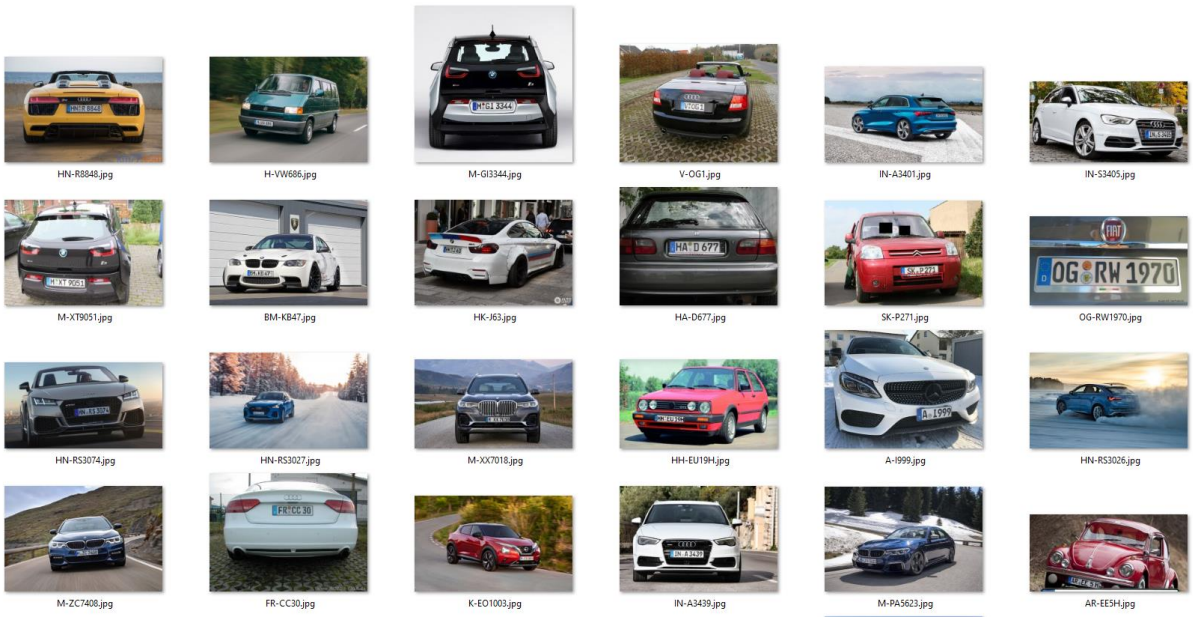
Here is a detailed description about the mAP score.

Analysis

Data Exploration

License Plate Detection

Unfortunately there are no public data collections with pictures of cars with recognizable German license plates. Therefore, an own data collection was created for License Plate Detection, with pictures collected from the internet, see examples below:

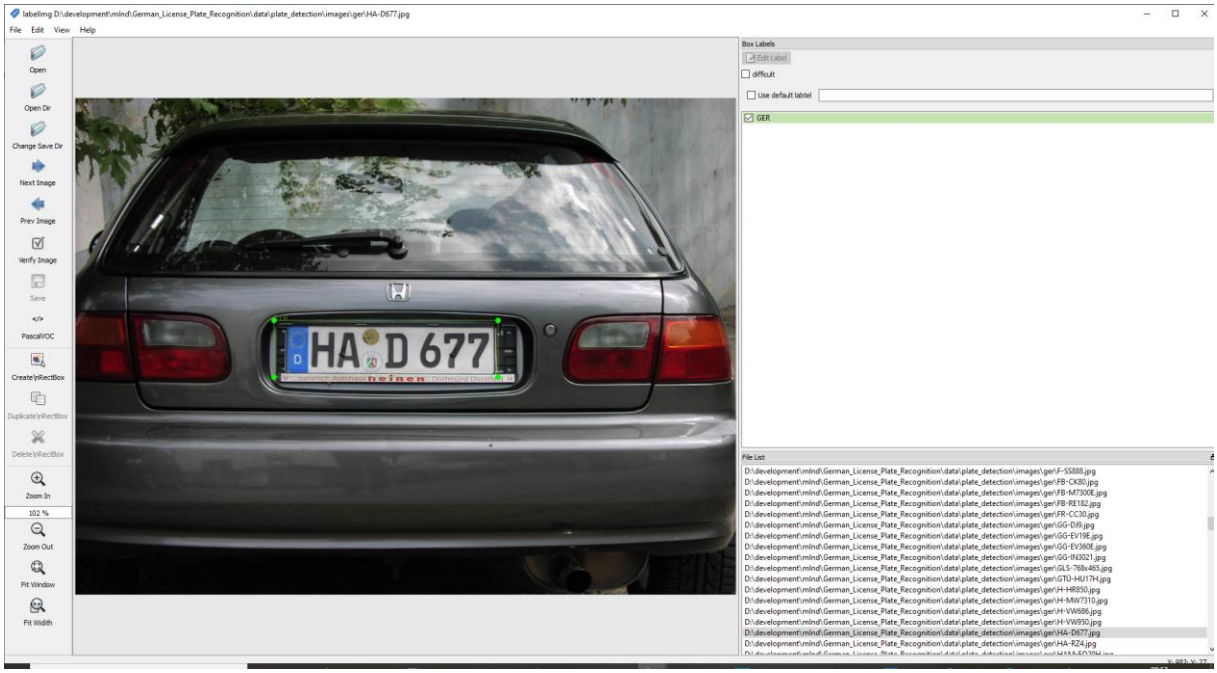


The following criteria were taken into account when selecting the images:

- License number clearly recognisable by human
- Exclusively German number plates
- Sufficient picture quality (minimum resolution)
- Greatest possible diversity (various distances, car brands, colors, environments, weather, license numbers, rotations, …)

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

The captured images were labeled with the tool LabelImg. The car license plates in each image were framed with a bounding box, to which the class “GER” for Germany was assigned. This makes the data collection extendable for license plates of other countries, e.g. IT (Italy), FR (France), ...



LabelImg stores the label information as annotations in the PASCAL VOC format. For each image a corresponding XML file is created, which contains the label information like class and bounding box, etc. Here is an example file (HA-D677.xml) for the picture shown above:

```
<annotation>
    <folder>ger</folder>
    <filename>HA-D677.jpg</filename>
    <path>.\data\plate_detection\images\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>655</xmax>
            <ymax>431</ymax>
        </bndbox>
    </object>
</annotation>
```

From the images and the annotations a data collection in TFRecord format was generated, which was used for training the License Plate Detection with the TensorFlow Object Detection API.

The notebook License Plate Detection Data Exploration And Preparation shows how the data set for License Plate Detection was created.

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 for complete 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.

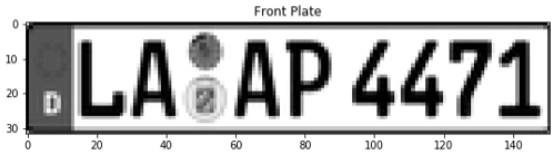
Example:

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 web service to generate an image from a car license number.

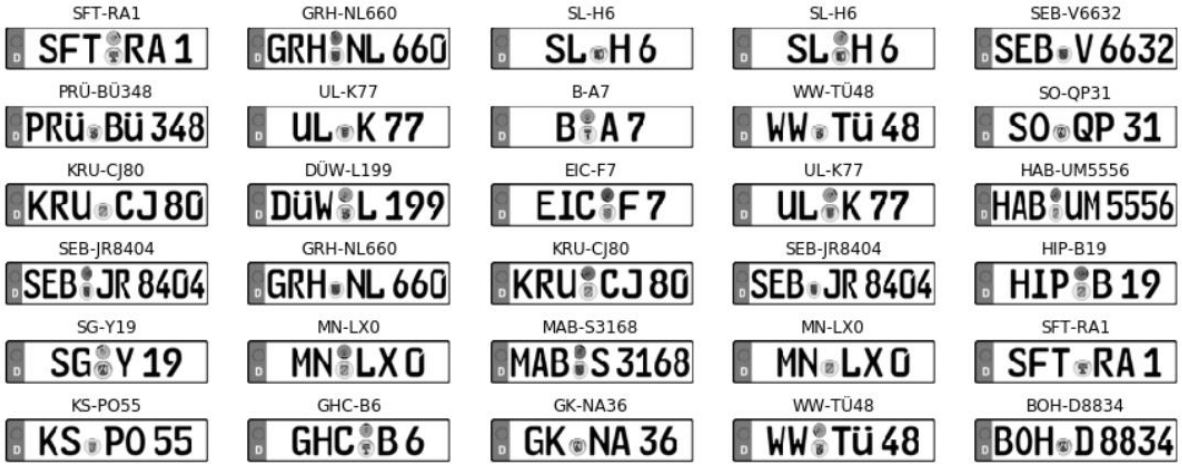
The generator observes the following criteria:

- 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 generated images for the front and rear license plate of a car:



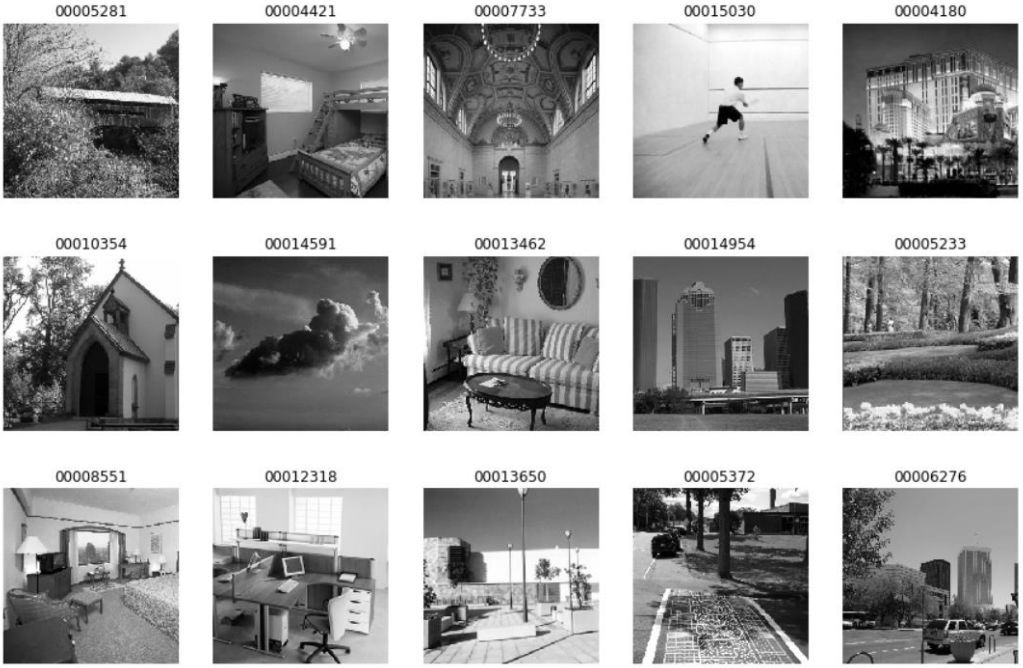
Examples of randomly generated images of car license plates:



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 data set and packed into an HDF5 data set. Here are some examples:



The notebook License Recognition Data Collection And Exploration shows how the data set for License Number Recognition was created.

Algorithms and Techniques

Number plate detection and number recognition are two different tasks with different requirements. For each of them a model was trained with the help of a method that meets the respective requirements.

The task of License Plate Detection is to recognize German license plates in a camera image (classification problem) and to localize them within the image (regression problem). For this the open source framework TensorFlow Object Detection API was used, that makes it easy to construct, train and deploy “state of the art” object detection models.

As a starting point I decided to use the ssdlite_mobilenet_v2_coco model, an Single Shot MultiBox Detector (SSD) which was pre-trained on the COCO data set. This offers a good balance between accuracy and speed, both factors that are important for reliable and user-friendly detection on a mobile device.

Training parameters:

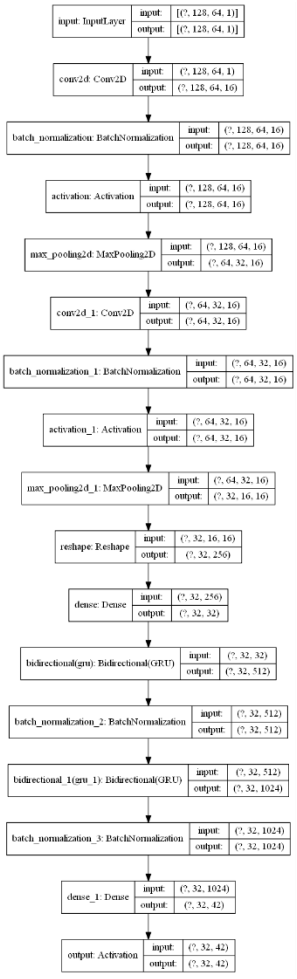
- Training length (number of epochs)
- Batch size (how many images to look at once during a single training step)
- Optimizer (what algorithm to use for learning)
- Learning rate (how fast to learn; this can be dynamic)
- Weight decay (prevents the model being dominated by a few “neurons”)
- Momentum (takes the previous learning step into account when calculating the next one)

See training pipeline configuration ssdlite_mobilenet_v2_coco.config.

Task of the License Number Recognition is to recognize the car license number as plain text from the detected license plate image section. I have decided to train a CRNN (Convolutional Recurrent Neural Network) inspired by the Keras OCR example. The network architecture is as follows:

Layer (type)	Output Shape	Param #
input (InputLayer)	[(None, 128, 64, 1)]	0
conv2d (Conv2D)	(None, 128, 64, 16)	160
batch_normalization (BatchNo	(None, 128, 64, 16)	64
activation (Activation)	(None, 128, 64, 16)	0
max_pooling2d (MaxPooling2D)	(None, 64, 32, 16)	0
conv2d_1 (Conv2D)	(None, 64, 32, 16)	2320
batch_normalization_1 (Batch	(None, 64, 32, 16)	64
activation_1 (Activation)	(None, 64, 32, 16)	0
max_pooling2d_1 (MaxPooling2	(None, 32, 16, 16)	0
reshape (Reshape)	(None, 32, 256)	0
dense (Dense)	(None, 32, 32)	8224
bidirectional (Bidirectional	(None, 32, 512)	1677312
batch_normalization_2 (Batch	(None, 32, 512)	2048
bidirectional_1 (Bidirection	(None, 32, 1024)	3151872
batch_normalization_3 (Batch	(None, 32, 1024)	4096
dense_1 (Dense)	(None, 32, 42)	43050
output (Activation)	(None, 32, 42)	0

Total params: 4,889,210
Trainable params: 4,886,074
Non-trainable params: 3,136



Training parameters:

- Training length (number of epochs)
- Batch size (how many images to look at once during a single training step)
- Optimizer (what algorithm to use for learning)
- Learning rate (how fast to learn; this can be dynamic)

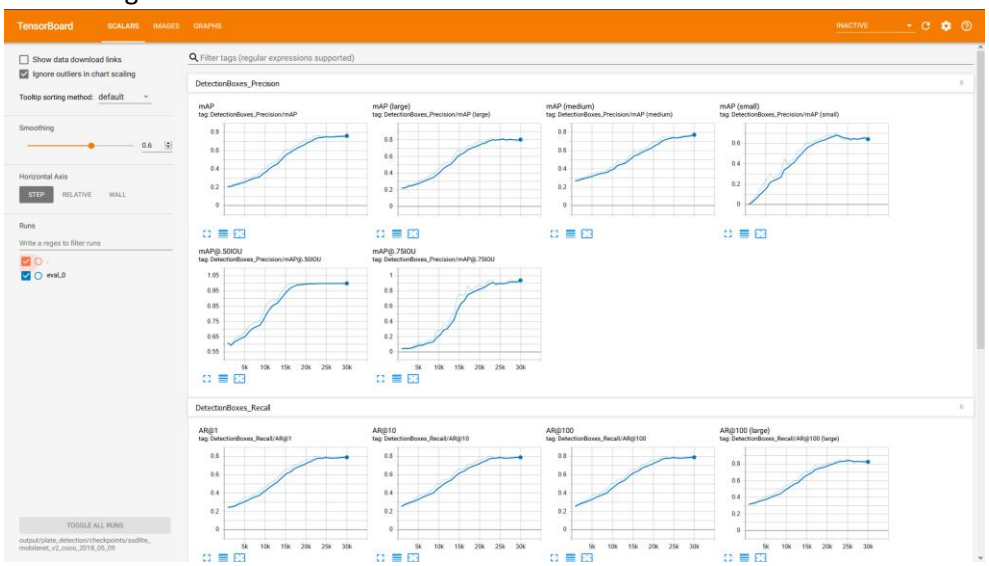
Adagrad (Adaptive Gradient Algorithm) is used as Optimizer, this dynamically adapts the learning rate during the training. Furthermore Early Stopping is used to stop the training if there is no improvement for some time, this reduces the risk of Overfitting and shortens unnecessarily long training time.

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. For evaluation the “mAP score” was considered in the License Plate Detection and the “validation loss” in the License Number Recognition.

Monitoring the mAP score with Tensorboard:



The achieved mAP score for License Plate Detection is 0.76. Considering the few images available for the training and the partly bad image quality this is very good and sufficient for the use case.

The goal was that the Android app should provide the user with a fluent and equally reliable License Number Recognition. For this purpose, the Android App was subjected to intensive practical testing under real conditions.

Methodology

Preprocessing

The pre-processing steps for License Plate Detection are done by the TensorFlow Object Detection API out-of-the-box, so there was nothing else to do here.

This is different with License Number Recognition, where realistic images for the training are generated from the generated license number images during the pre-processing step using data augmentation.

Data augmentation steps:

- Varying background (background contains no features)
- Random rotation in X,Y,Z direction (different camera perspectives)
- Varying scale (different camera distances)
- Varying brightness (different times of day and lighting conditions)
- Slight interference effects (Blur and shake of the camera)

Here are some examples of how realistic images are randomly generated from the generated number plate images:

Further pre-processing steps are:

- Image resizing to the network input size
- Normalizing image data between 0.0 and 1.0

Project Design and Implementation

The implementation is roughly divided into 3 main tasks:

1. Training of a model for License Plate Detection
2. Training of a model for License Number Recognition
3. Implementation of an Android app that recognizes car license numbers using the trained models

Training of a model for License Plate Detection and Localization in images

Implementation steps for License Plate Detection:

- Download the pre-trained model from Tensorflow detection model-zoo
- Train a finetuned model for License Plate Detection and Localization using the TensorFlow Object Detection API
- Export the finetuned model as TFLite model that can be used by the Android app
- Test the trained License Plate Detection model

The implementation of these steps was done in notebook License Plate Detection Model Training And Evaluation

Training of a model for License Number Recognition from license plate images

Implementation steps for License Number Recognition:

- Download the License Plate Recognition data set (optional if not already done)
- Load and split the data into a train, validation and test set
- Create a LicensePlateImageAugmentor instance for data augmentation
- Create data generators for the train, validation and test set
- Create the License Number Recognition model to be trained
- Train and evaluate the License Number Recognition model (Keras model)
- Convert the trained Keras model into an TFLite model, which can be used by the Android app
- Test the trained License Number Recognition (TFLite) model

The following two Python classes are of particular importance:

- LicensePlateImageAugmentor: Implements Data Augmentation to create realistic images for training from the generated license plate images.
- LicensePlateDatasetGenerator: Creates batches with input data for model training from randomly selected license plate images from the dataset, which are varied using data augmentation.

The implementation was done in the notebook License Recognition Model Training And Evaluation

Android App Implementation

The Android app was developed based on the Tensorflow Lite Object Detection example app.
The application was extended from pure object detection to the detection of license plates and the recognition of the license number as plain text and underwent a major refactoring.

The most important classes are:

- ClassifierActivity.kt (Calls license plate detection and license number recognition for the current camera image and draws the determined license number onto the image.)
- PlateDetector.kt (License plate detection and localization)
- LicenseRecognizer.kt (License number recognition)
- Detection.kt (Stores score, location and license number of a recognized license plate)

As development environment was Android-Studio and the programming language Kotlin.

Refinement

No particular adjustments were necessary for license plate detection, as the use of TensorFlow Object Detection API by default produced sufficiently good results. Only the number of training epochs (num_steps) was increased to 30.000.

The initial problem with license number recognition was that although good training results and also good test results were achieved with the generated test data, the recognition failed miserably with real camera images. A lot of experiments and adjustments of the data augmentation were necessary to recognize real camera images. In particular, the use of a slight blur effect in data augmentation led to a breakthrough here.

Furthermore I experimented with different training parameters to achieve the best possible results. Here the parameters "Batchsize", "Optimizer" and "Epochs" were changed. The optimizers SDG, Adam, Adagrad, Adadelta and RMSprop were evaluated, but ultimately achieved similar results. The use of Adagrad achieved the best results with a relatively short training duration. By using the early stopping technique, the choice of epochs does not really matter, so that it was simply set to the relatively high value of 1000.

The final used training parameters are:

- Batchsize: 64
- Optimizer: Adagrad
- Epochs: 1000

Results

Model Evaluation and Validation

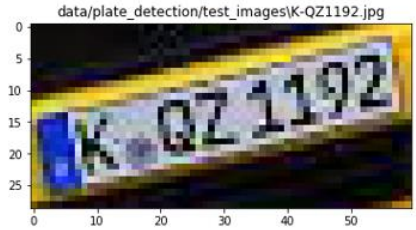
During development, each model was evaluated using a test set with data that was not used for training. The mAP score for license plate recognition and the validation loss for license number recognition were considered.

Furthermore, the recognition of license numbers was tested with additional test images using the Notebook License Recognition Workflow. Here is an example:

Test image as starting point

Result of the License Plate Detection is the extracted license plate image and the classification score

```
Box: [0.43043226 0.57734394 0.5922101  0.78368795]
Class: 0
Score: 0.9999966621398926
Detections: 1
```



Result for the License Number Recognition is the license number as plain text

Justification

The final Android app has been extensively tested with a Xiaomi M8 smartphone, here are some findings:

- The detection works quite reliably as long as the user keeps an optimal distance and a certain tolerance of the angle to the license plate.
- The recognition works quite reliably with real images. If images are filmed from the monitor, it depends very much on the image quality whether the recognition works or not.
- Heavily rotated license plates are not or incorrectly recognized.
- The recognition takes place within a reasonable time, but even faster recognition would be desirable. Sometimes the bounding box is displayed at the wrong position because the camera has been moved in the meantime and the camera position does not match the position at the time of detection.
- Sometimes license numbers are recognized incorrectly and still displayed.

Here are some test examples:

Reliable detection, at optimal position and distance

Recognition becomes error-prone with increasing rotation

If the distance is too great, no license plate will be detected.

The following demo video shows the final Android app in action.

Improvement

The results achieved are very satisfying, but there are still some possibilities to increase the reliability and usability of the application:

- Extension of the training data for license plate detection:
  Since license plate detection was trained with very few data, some license plates are not recognized or the user must have the optimal distance and alignment to the license plate. Also, many real conditions, such as different weather and light conditions, are still too less considered. More and better training data could make license plate recognition easier for the user.

- Improvement of data augmentation:
  Strongly rotated or distorted numbers are not taken into account enough in the current data augmentation, therefore the recognition does not work reliably in these cases. This could be improved by extending data augmentation.

- Use GPU:
  Due to issue #36016 of Tensorflow, currently the CPU and not the GPU of the mobile device is used. By using the GPU, a speed increase in detection and recognition by a factor of 3-4 could be achieved, which would result in a much smoother working for the user.

- Recognition in portrait and landscape format:
  Currently, recognition only works when the user holds the mobile device in landscape format. Additional support of the portrait format would increase usability.

- Determination and consideration of score value for license recognition:
  Currently, the detected license number is always displayed, even if it is incorrect. By additionally determining a score value for the license recognition, bad results could be faded out by means of a threshold, thus increasing the reliability of the application.