Rakhlevski Ilia

316932847

Final project – deep learning part

License Plate Recognition

Goal: Developing of neural network that recognizes car license plates.

Dataset: Cars images containing license plates.

Environment: Python 3.10, PyTorch, Spyder IDE 5.4.1, Jupiter Notebook, via-2.0.12, Kaggle.

Dataset creation

Restriction:

Within the project we use regular Israeli license plates: black foreground, yellow background.

Plates for military, police, diplomatic cars, also poor-quality plates are not considered.

Searching for data:

Most of the cars images were taken from the site Yad2 [1]. The rest were taken from the dataset “license_plate_israel” [2].

Labeling:

The labeling of the digits is performed by using the software VGG Image Annotator [3].

Images padding:

The images have different sizes, but the neural network requires certain input size. So, we need to convert all the images to the same size. For this purpose, it is used image padding.

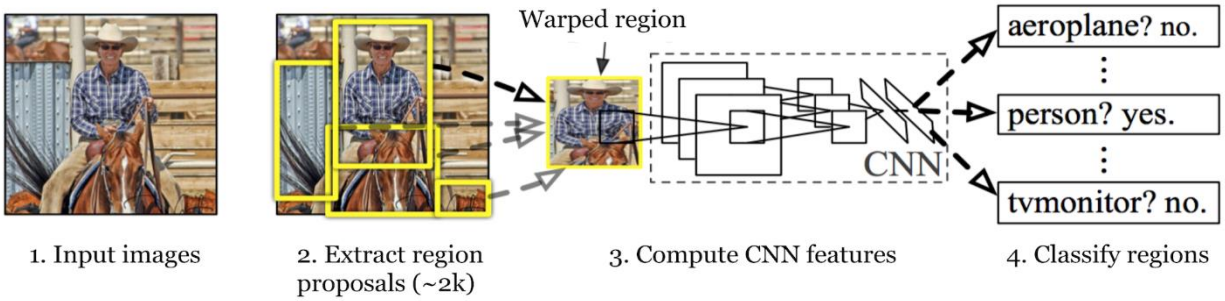
Model

Model that is used in the project is Faster R-CNN [4][11].

R-CNN

R-CNN [6] is short for “Region-based Convolutional Neural Networks”. The main idea is composed of two steps.

- First, using Selective Search [5] or some other region proposal algorithm, it identifies a manageable number of bounding-box object region candidates (“region of interest” or “RoI”).
- And then it extracts CNN features from each region independently for classification.



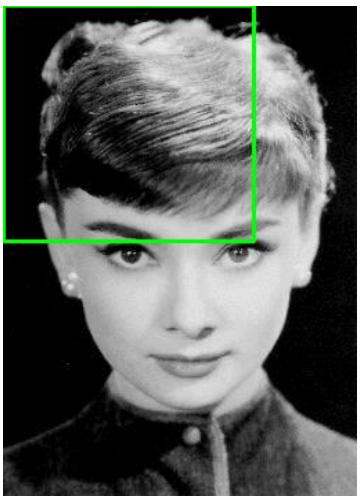
Each image needs to classify *2000* region proposals. So, it takes a lot of time to train the network. It requires *49* seconds to detect the objects in an image on GPU. To store the feature map of the region proposal, lots of disk space is also required.

Region proposal

Region proposals are just smaller parts of the original image, that we think could contain the objects we are searching for. There are different region proposal algorithms we can choose from. Examples:

Sliding window:

A sliding window is a rectangular region of fixed width and height that “slides” across an image. When using this method, you just go over the whole image with different sized rectangles and look at those smaller images in a brute-force-method. The problem is you will have a giant number of smaller images to look at.



Selective Search:

Selective Search [5] is a region proposal algorithm used in object detection. It is designed to be fast with a very high recall. It is based on computing hierarchical grouping of similar regions based on *color*, *texture*, *size* and *shape* compatibility.

Color of the cats is different but textures are same.

Chameleon and grass.
Colors are similar but textures are different.

Car and wheels. There no color or texture similarity. But there is size and shape similarity between wheel and fender (wing).

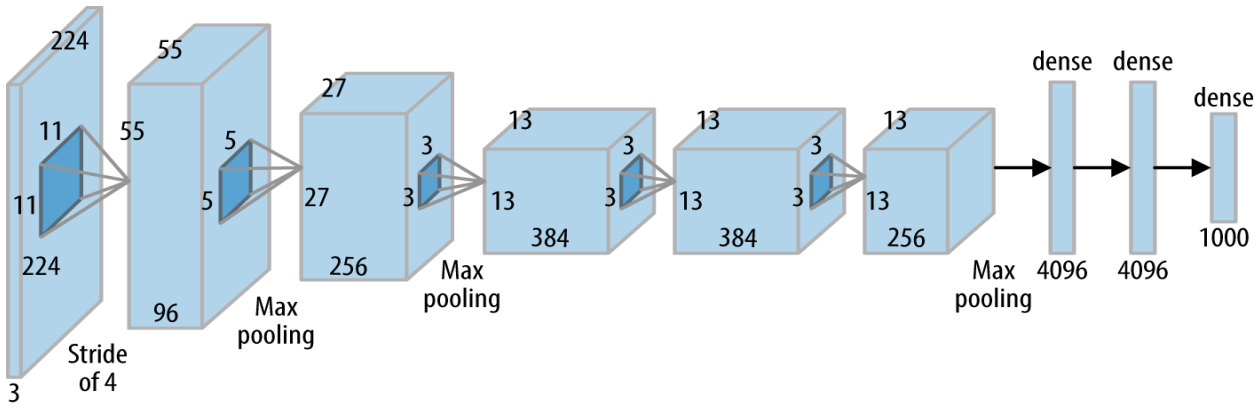
Selective Search algorithm takes performs the following steps:

1. Initialize proposal regions
2. Add all bounding boxes corresponding to segmented parts to the list of regional proposals
3. Group adjacent segments based on similarity
4. Go to step 2

At each iteration, larger segments are formed and added to the list of region proposals. Hence, we create region proposals from smaller segments to larger segments in a bottom-up approach.

CNN

In the next step, we take each region proposal and we will create a feature vector representing this image in a much smaller dimension using a Convolutional Neural Network (CNN). They use the AlexNet architecture [7] as a feature extractor.



SVM

At the previous step we created feature vectors from the image proposals. Now we will need to classify those feature vectors. We want to detect what class of object those feature vectors represent. For this purpose, we use a Support Vector Machine classifier. We have one SVM for each object class and we use them all.

Fast R-CNN

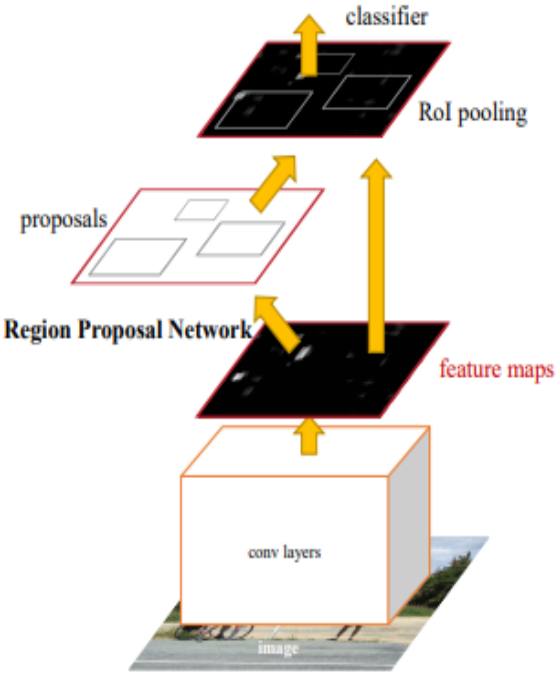
In R-CNN we passed each region proposal one by one in the CNN architecture and selective search generated around *2000* region proposal for an image. So, it is computationally expensive to train and even test the image using R-CNN.

To deal with this problem Fast R-CNN [8] was proposed, it takes the whole image and region proposals as input in its CNN architecture in one forward propagation.

Fast R-CNN drastically improves the training *(8.75 hrs vs 84 hrs)* and detection time from R-CNN. It also improves Mean Average Precision (mAP) [9] marginally as compare to R-CNN. Most of the time taken by Fast R-CNN during detection is a selective search region proposal generation algorithm.

Faster R-CNN

The bottleneck of the architecture Fast R-CNN is Selective Search. Since it needs to generate *2000* proposals per image. In Faster R-CNN [4], it was replaced by the region proposal network.



First of all, we pass the image into the backbone network. This backbone network generates a convolution feature map.

These feature maps are then passed into the region proposal network. The region proposal network takes a feature map and generates the anchors (the center of the sliding window with a unique size and scale).

These anchors are then passed into the classification layer (which classifies that there is an object or not) and the regression layer (which localize the bounding box associated with an object).

Using Pre-Trained PyTorch Model

A pre-trained model refers to a model or a saved network created by someone else and trained on a large dataset to solve a similar problem.

The pre-trained models are available from sub-modules of ***models*** module of ***torchvision*** library. The sub-module named ***detection*** provides us with various methods that can be called to load pre-trained object detection models. We load the Faster R-CNN model for our purpose. It uses ***ResNet-50-FPN*** (Feature Pyramid Network) network for detecting important features in images. The network is loaded by calling method ***fasterrcnn_resnet50_fpn()***.

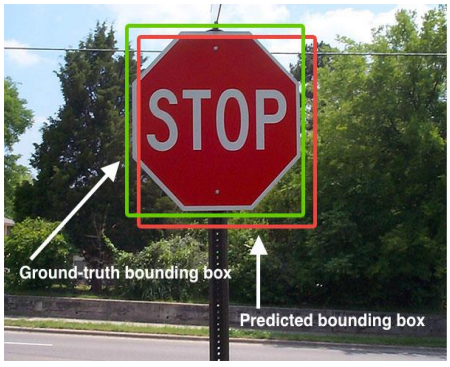
Measuring Object Detector Performance

When evaluating object detector performance, we use a combination of two evaluation metrics: Intersection over Union (IoU)[10] and mean Average Precision (mAP)[9].

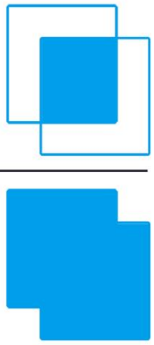
Intersection over Union

Algorithm that provides predicted bounding boxes (and optionally class labels) as output can be evaluated using IoU. To apply IoU to evaluate an arbitrary object detector, we need the following:

1. The ground-truth bounding boxes (i.e., the hand-labeled bounding boxes from our testing set that specify where in an image our object is).
2. The predicted bounding boxes from our model.
3. To compute recall and precision, you'll also need the ground-truth and predicted class labels.



$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



Mean Average Precision

AP (Average precision) is a popular metric in measuring the accuracy of object detectors like Faster R-CNN, SSD, etc. Average precision computes the average precision value for recall value over 0 to 1.

Tests Accuracies

The main goal of the license plate recognition is location and recognition of its digits. We need to find boxes containing digits and their order. We do not need to find exact size and location of each box containing a digit.

After finding all the boxes and sorting them according to the order we convert digits to the string and compare this string with the true license plate number converted to string.

Example:

Image of a car containing license plate - input:

Prediction output:


```
Predicted figures:
[8, 1, 3, 4, 5, 0, 9, 1]

Predicted sorted figures:
(4, 1, 8, 5, 3, 9, 0, 1)

Predicted car number:  41853901

Real car number:       41853901

Similarity:  1.0
```

By comparing of two strings, we check matching of two characters sequences. For this purpose we use the library ***difflib***, the class *`SequenceMatcher`*.

Training

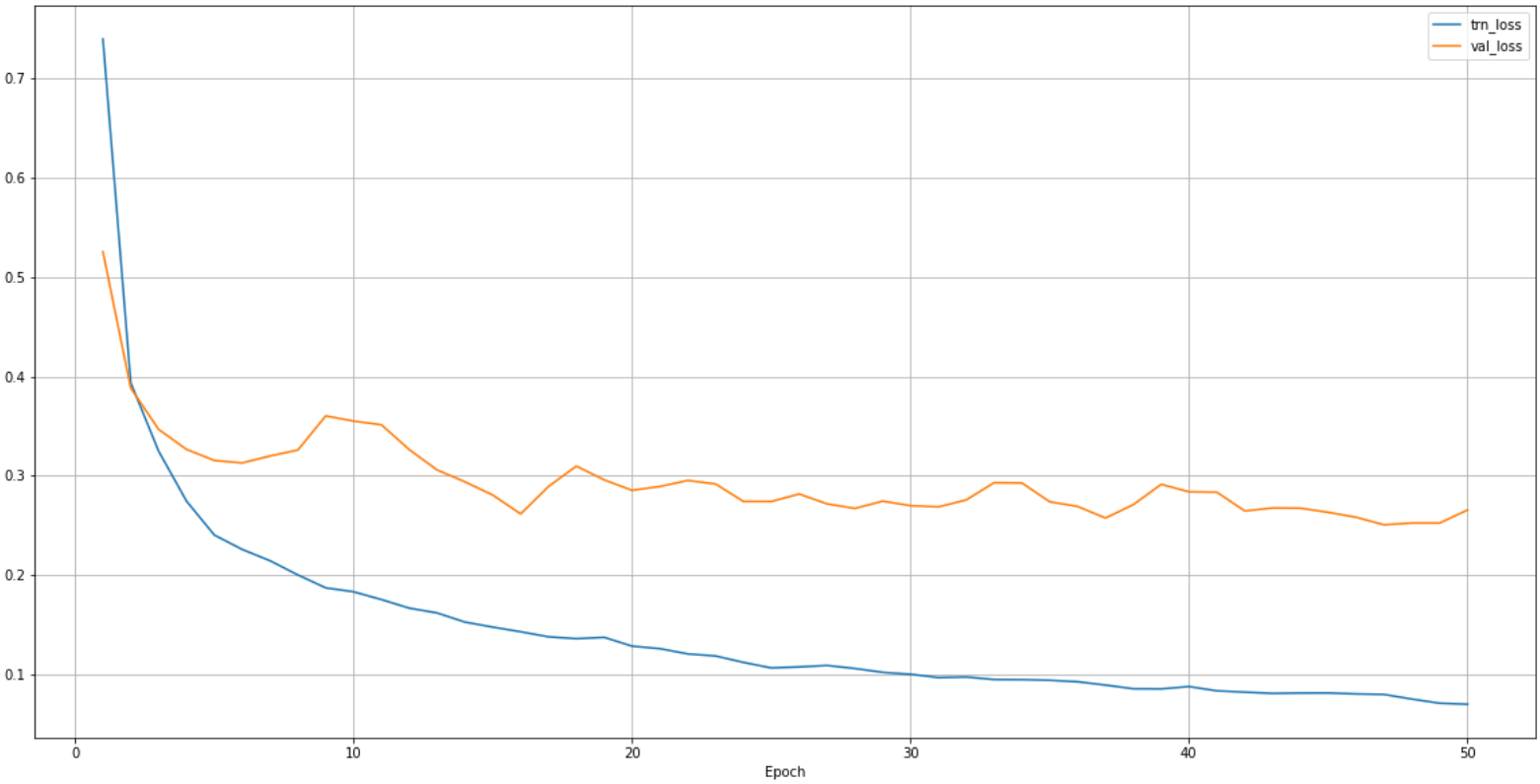
For the model training are used 500 images. 450 – for training, 50 – for validation.

The training was performed in three stages.

1st stage:

Epochs: 1-50

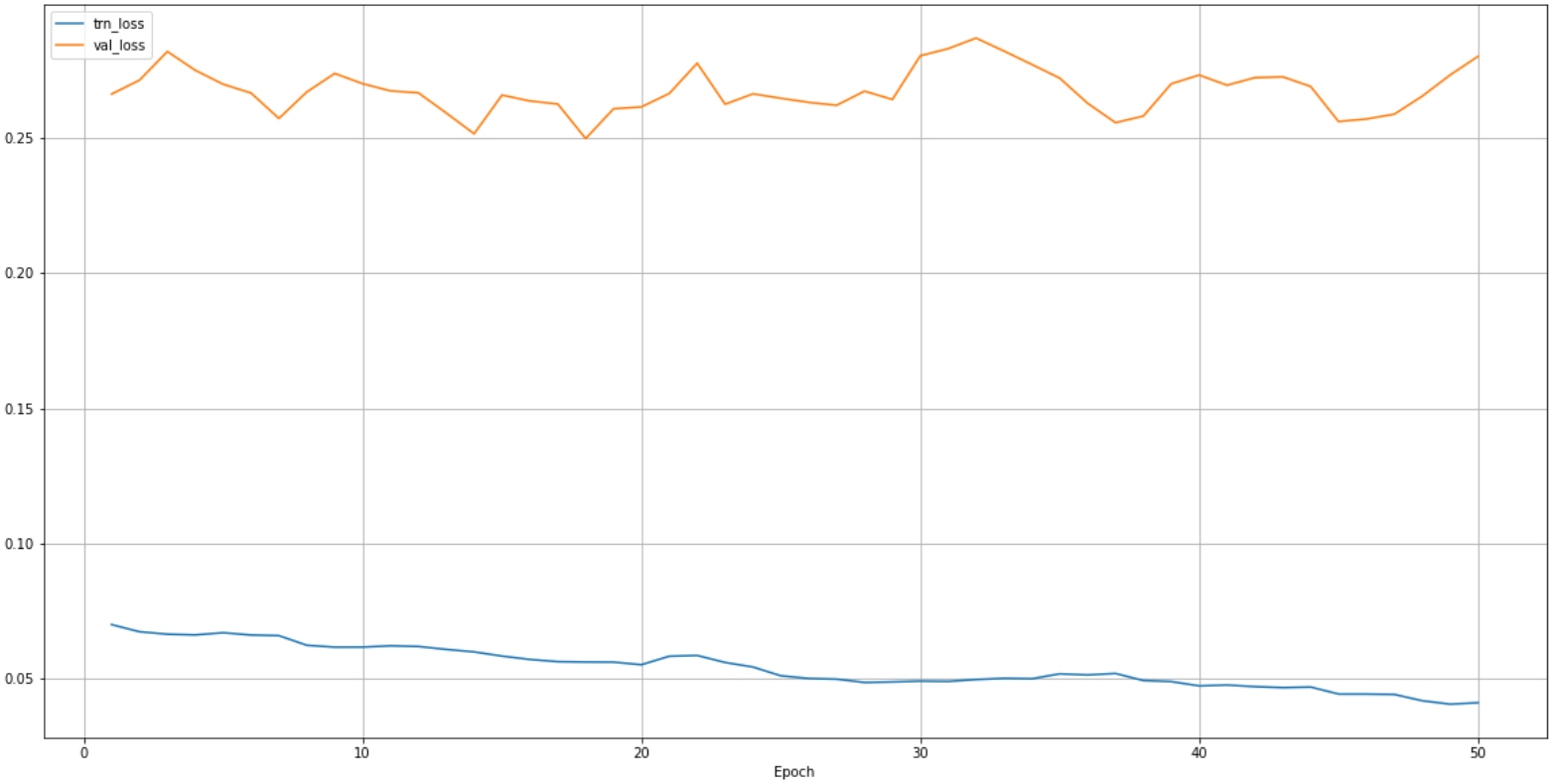
Learning rate: 0.005



2nd stage:

Epochs: 51-100:

Learning rate: 0.005

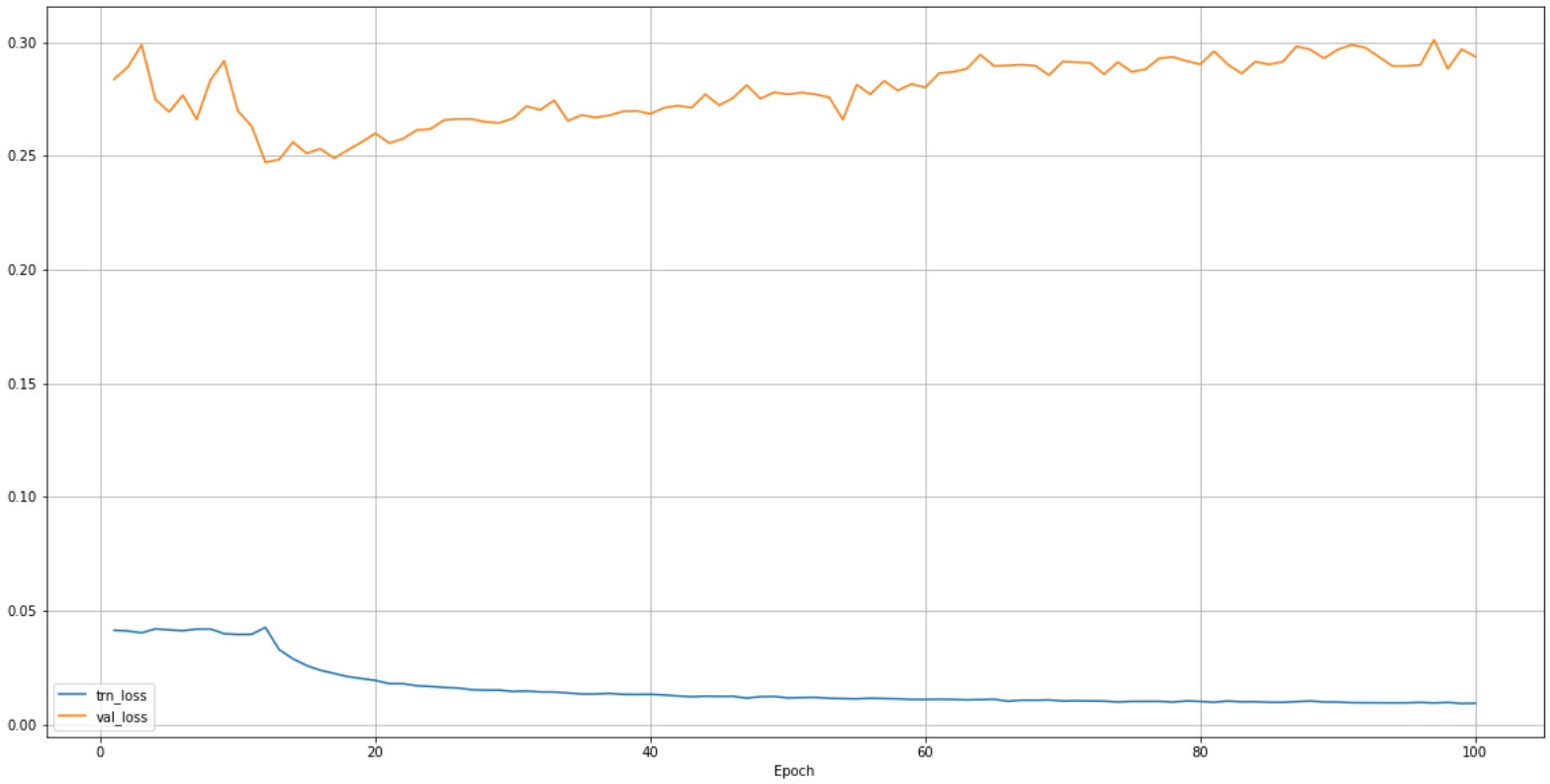


3rd stage:

Epochs: 101-110, learning rate: 0.005

Epochs: 111-170, learning rate: 0.001

Epochs: 171-200, learning rate: 0.0005



Testing

For the model training are used 50 images containing license plates. Name of each file is <license plate number>.jpg. We find accuracy (similarity) for each image and calculate their average.

Result: accuracies average - 0.966

Project Files

Code:

License-plate-faster-rcnn-py.ipynb – Model creation/training. The code is run on the Kaggle platform.

License_Plate_Faster_RCNN_prediction.ipynb/mhtml – Testing/prediction.

Plot_Results.py - Plot the results from the data frame is contained in the file.

CSV_convertation.py - Convert the data received from the VGG Image Annotator to more appropriate format is used for training of the model.

Utils.py - Some useful functions for image processing.

Images_padding_resizing.py - Resizing images by padding.

Data:

df.csv – Annotation data.

epochs_aver_df.csv – Training/testing data for each epoch.

Models:

model.pt – Trained model.

fasterrcnn_resnet50_fpn_coco-258fb6c6.pth – pretrained Faster R-CNN model.

Images:

images.zip – images for training.

images_test.zip – images for testing.

Further Work

Increasing size of the dataset. To add extra poor-quality images: bad light, too far from the car, damaged license plates and etc. Also, can be added plates of different types: military, diplomatic, police cars plates.

References:

[1] Yad2: https://www.yad2.co.il/vehicles/cars

[2] Dataset “license_plate_israel”: https://www.kaggle.com/datasets/gaelcohen/license-plate-israel

[3] VGG Image Annotator: https://www.robots.ox.ac.uk/~vgg/software/via/

[4] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks.

[5] Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition.

[6] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation.

[7] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks.

[8] Girshick, R. (2015). Fast r-cnn.

[9] mAP (mean Average Precision) for Object Detection: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

[10] Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., & Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression.

[11] V Kishore A., Yeshwanth R. (2020). Modern Computer Vision with PyTorch, Chapter 8 - Advanced Object Detection.