**[Automatic-License-Plate-Recognition](https://github.com/ikigai-aa/Automatic-License-Plate-Recognition)**

**Overview**

This is a four phased Object Detection project mainly focussing on detecting Vehicle's license plates and thereby reading the license number and saving them in a text file for use by the concerned authority. This Deep Learning Project uses YOLOv4(You Only Look Once) as its Neural Network Architecture which is made above a framework called Darknet. It is then made deployment ready with TensorFlow Lite making it compatible for using it in various edge devices such as android, iOS, raspberry Pi etc.

**Motivation**

* Automatic License Plate Recognition (ALPR) has been a frequent topic of research due to many practical applications, such as automatic toll collection, traffic law enforcement, private spaces access control and road traffic monitoring.
* ALPR systems typically have three stages: License Plate (LP) detection, character segmentation and character recognition. The earlier stages require higher accuracy or almost perfection, since failing to detect the LP would probably lead to a failure in the next stages either. Many approaches search first for the vehicle and then its LP in order to reduce processing time and eliminate false positives.

**Technical Aspect**

In order to have an efficient object detection (here, the vehicle number plates), I have used the most recently developed algorithm of YOLO series i.e. YOLOv4 backed up over a framework of Darknet.

**Yolov4:** Yolo abbreviates to You Only Look Once depicting its ability to detect objects and entities by using CNN (Convolutional Neural Network). Neural Network in YOLO uses weights trained by the user through annotated training data by using bounding boxes. Hence YOLO takes an image as input puts it through a Neural Network and gives the output in the image through bounding boxes. The input image is divided into SXS grid of cells. Each cell contributes to the object detection. Each cell predicts Bounding Boxes as well as Class probabilities. The prediction consists of 5 components (x,y,w,h,confidence). (x,y) represents the centre of the bounding box and (w,h) are the width and the height of the boxes. Confidence represents the Estimated Prediction Accuracy of the object. YOLO is extremely fast and accurate as compared to other algorithms and hence was our primary choice for this project.

**Dataset Source**

As, this project needed a lot of images to perform transfer learning with the weights of pre-trained model of YOLOv4 architecture which is trained on large scale dataset called COCO dataset(having more than 80 classes for object detection), so I used the Google Open Image Dataset for my custom retraining with that model. For extracting images for both training and testing, I used an toolkit called OIDToolKit4 which extracts the images from the Open Images Dataset by Google and create annotations for each files as YOLO architecture takes the image annotations present in a text file as an input. A complete directory for that toolkit is present above in this repository. Kindly follow the readme.md file of it to get into the specifications.

**Important Python Scripts**

ANPR\_YOLOV4\_Darknet.ipynb: This takes care of the transfer learning of my custom image dataset which I extracted using the OIDToolKIt4.For better performance and use free GPU benefits by Google, Open this notebook on Google Colab.and perform the neccessary steps mentioned on it.

convert\_annotations.py: This python script is present under the subdirectory of OIDToolKit and it helps to extract out the images annotations on a .txt file.

generate\_train.py and generate\_test.py: The configuration files needed before we can begin to train our custom detector are the train.txt and test.txt files which hold the relative paths to all our training images and valdidation images.These scripts easily generate these two files withe proper paths to all images.

extract\_frames.py: This python script extract frames out of the output video with bounding boxes into frames in order to perform Optical Character Recognition.

pytesseract\_ocr.py: This python script takes care of performing Optical Character Recognition with the help of Google Tesseract API on the output images.

**References**

* M M Shidore, and S P Narote. (2011) “Number Plate Recognition for Indian Vehicles” International Journal of Computer Science and Network Security 11 (2): 143-146
* Sang Kyoon Kim, D. W. Kim and Hang Joon Kim. (1996) “A recognition of vehicle license plate using a genetic algorithm based segmentation,” Proceedings of 3rd IEEE International Conference on Image Processing, Lausanne.
* https://docs.opencv.org/master/
* https://github.com/anuj-badhwar/Indian-Number-Plate-Recognition-System