Car Number Plate Detection using YOLO V3

**Work Flow** **–**

* Create your own data set, consisting of car images. (at least 150-200)
* Label the number plates in each image with help of “IMAGE LAB” software.
* We train a Yolo V3 model on our dataset, using GPU services provided by Google Colab.
* Use the Trained weights and the config. file for testing our model. During the testing phase, we use OpenCv’s framework for yolo to obtain number plates from given image.

**\*\*** before testing the model on car images from every angle, it was tested on car images from front, to determine:

* Is it feasible to solve the number detection problem with YoloV3.
* No of images required to obtain feasible results. (worked well with 220 images)
* No. of iterations required to get a fine working model. (decent outputs after 2000 iterations)

Below shown is an output from this prototype model.



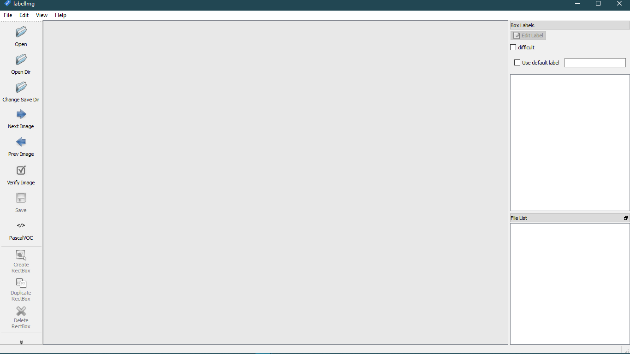
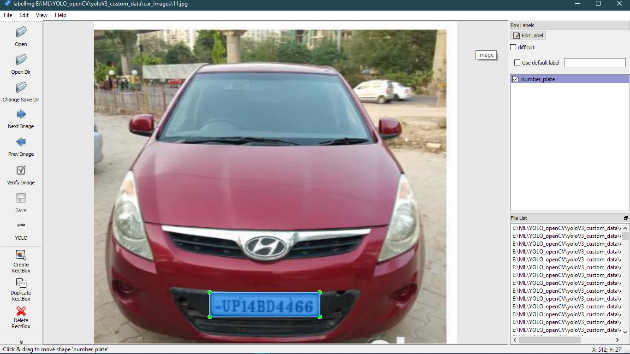
**Creating your own dataset**

Downloaded 1077 car images from different angles. It was ensured that there is no bias in the data towards a specific angle image of cars. Cars of different manufacturers, colours and sizes were taken for this data, to maintain versatility of the dataset. (it has images from front, back and sides of the cars, such that number plates are visible in each of the images)



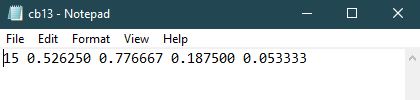
**Labelling obtained data**

Now we need to create annotations for each image, to specify the number plate position in each image. For this purpose, we use “label Img – v1.8.0” software.

We use the software to generate .txt files in YOLO format. The Folder containing the images is opened.

**Create Rectbox** is selected to mark down the object in image. Annotation format is set to **YOLO**. After making the box and labelling it, **save** it as .txt file in the same folder which has the images.



<object-class> <x\_center> <y\_center> <width> <height>

Yolo Format for image cb13.jpg

**Training Model on Google Colab with darknet framework.**

* We use google Drive to access our data in google Colab. Hence, we upload our data as a *zip* file in Drive.

Eg. I have stored zip file in following location: /mydrive/Yolo\_V3\_for\_OpenCV/car\_images\_number\_plate

* Now we open a new Notebook in google colab and Mount the drive in it.

from google.colab import drive

drive.mount('/content/gdrive')

!ln -s /content/gdrive/My\ Drive/ /mydrive

!ls /mydrive

* We clone the official darknet github repository in our notebook environment.

!git clone https://github.com/AlexeyAB/darknet

* Now we make necessary edits in darknet folder to give our GPU the necessary access and notify, that we will have to use it later in OpenCV.

# change makefile to have GPU and OPENCV enabled

%cd darknet

!sed -i 's/OPENCV=0/OPENCV=1/' Makefile

!sed -i 's/GPU=0/GPU=1/' Makefile

!sed -i 's/CUDNN=0/CUDNN=1/' Makefile

!make

* The following steps will configure the Darknet to train for our custom label.

Copying the config file and naming it is *yolov3\_training.cfg.*

!cp cfg/yolov3.cfg cfg/yolov3\_training.cfg

Do the following changes in *.cfg* file to configure it for one custom class.

!sed -i 's/batch=1/batch=64/' cfg/yolov3\_training.cfg

!sed -i 's/subdivisions=1/subdivisions=16/' cfg/yolov3\_training.cfg

!sed -i 's/max\_batches = 500200/max\_batches = 4000/' cfg/yolov3\_training.cfg

!sed -i '610 s@classes=80@classes=1@' cfg/yolov3\_training.cfg

!sed -i '696 s@classes=80@classes=1@' cfg/yolov3\_training.cfg

!sed -i '783 s@classes=80@classes=1@' cfg/yolov3\_training.cfg

!sed -i '603 s@filters=255@filters=18@' cfg/yolov3\_training.cfg

!sed -i '689 s@filters=255@filters=18@' cfg/yolov3\_training.cfg

!sed -i '776 s@filters=255@filters=18@' cfg/yolov3\_training.cfg

Now we make *obj.names* file and *obj.data* file inside *data* folder. Also make a directory *data/obj*, which will contain all the data.

!echo "number\_plate" > data/obj.names

!echo -e 'classes= 1\ntrain  = data/train.txt\nvalid  = data/train.txt\nnames = data/obj.names\nbackup = /mydrive/Yolo\_V3\_for\_OpenCV/car\_images\_number\_plate' > data/obj.data

!mkdir data/obj

Next step is to download the weights file with this command. This has to be done for the first time only. If current model has been trained before, we can use our trained weights file from our drive directly.

# Download weights darknet model 53

!wget https://pjreddie.com/media/files/darknet53.conv.74

* Extracting the images into our required folder.

!unzip /mydrive/Yolo\_V3\_for\_OpenCV/car\_images\_number\_plate/car\_images.zip -d data/obj

* This piece of code ensures that our data has only one object class, else our configurations may create an error.

# We're going to convert the class index on the .txt files. As we're working with only one class, it's supposed to be class 0.

# If the index is different from 0 then we're going to change it.

import glob

import os

import re

txt\_file\_paths = glob.glob(r"data/obj/\*.txt")

for i, file\_path in enumerate(txt\_file\_paths):

    # get image size

    with open(file\_path, "r") as f\_o:

        lines = f\_o.readlines()

        text\_converted = []

        for line in lines:

            print(line)

            numbers = re.findall("[0-9.]+", line)

            print(numbers)

            if numbers:

                # Define coordinates

                text = "{} {} {} {} {}".format(0, numbers[1], numbers[2], numbers[3], numbers[4])

                text\_converted.append(text)

                print(i, file\_path)

                print(text)

        # Write file

        with open(file\_path, 'w') as fp:

            for item in text\_converted:

                fp.writelines("%s\n" % item)

* Now we need to create *train.txt* file, which will contain list of all training images. If we want a test/validation set, we can also create a *test.txt/val.txt* file. All these files have to be in *data* folder.

import glob

images\_list = glob.glob("data/obj/\*.jpg")

print(images\_list)

#Create training.txt file

file = open("data/train.txt", "w")

file.write("\n".join(images\_list))

file.close()

* Now we will train our model.

The following code trains with initial weights.

# Start the training

!./darknet detector train data/obj.data cfg/yolov3\_training.cfg darknet53.conv.74 -dont\_show -map

If you have already trained the model earlier, use below given code to resume training from where you left off.

# Start the training

!./darknet detector train data/obj.data cfg/yolov3\_training.cfg /mydrive/Yolo\_V3\_for\_OpenCV/car\_images\_number\_plate/yolov3\_training\_last.weights -dont\_show -map

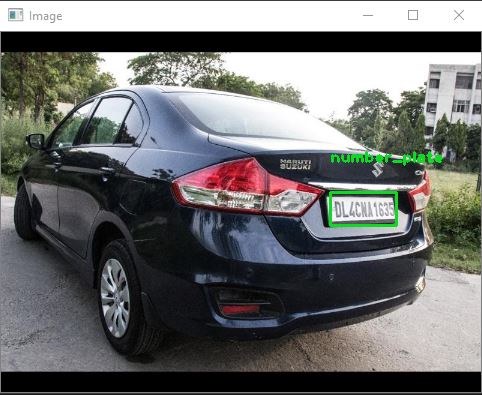
\*\* we used -map to get mAP after 1000 values and -don’t\_show to neglect some basic errors from occouring (eg. If there is some image with no annotations). Model was trained till 3000 iterations and provided with pretty accurate results.

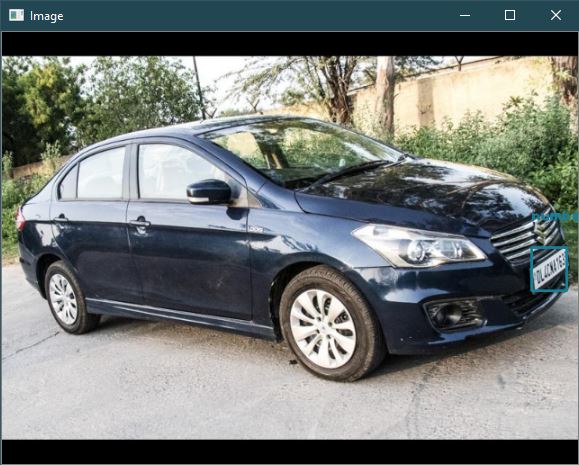
**Testing the trained model**

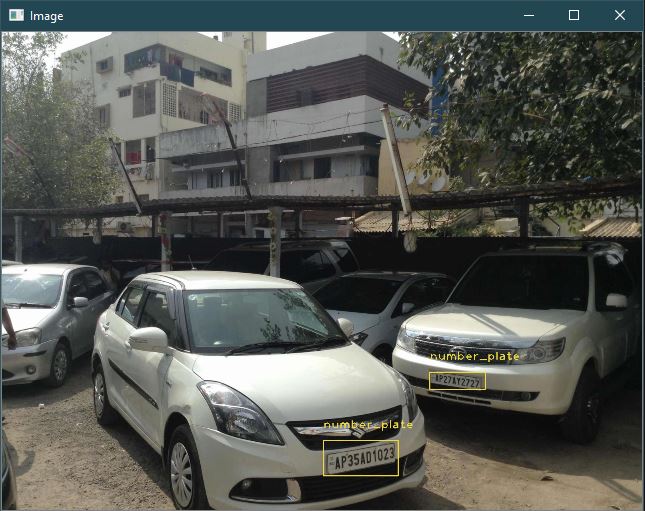
We get our config file from following path - /content/darknet/cfg/yolov3\_training.cfg and trained weights from the backup path we specified earlier. Keep both of them in a folder and run *yolo\_car\_number\_plate\_detection.py* script.

We will load the trained network and weights, define the names of classes and give path to test images. After that Yolo is applied on images in out test folder and Number Plates are detected.

Following are the results from my trained model:

****

**References –**

* <https://pysource.com/2020/04/02/train-yolo-to-detect-a-custom-object-online-with-free-gpu/>
* <https://github.com/AlexeyAB/darknet>