INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD
VII Semester B.Tech in Information Technology

C3 Report

Visual Recognition Project

NUMBER PLATE DETECTION AND IDENTIFICATION OF VEHICLES

By :-

Chinmay Tayade	(IIT2018138)
Inayat Baig	(IIT2018165)
Madhu	(IIT2018068)
Gurutej	(IIT2018193)

Problem Statement:

Automatic License Plate Recognition (ALPR) has been a frequent topic of research due to many practical applications. However, many of the current solutions are still not robust in real-world situations, commonly depending on many constraints.
The vehicles have their own unique Number plate. Therefore we can identify the vehicle and the owner details of any particular vehicle.
Therefore we propose a model based on YOLO(a Deep learning based object detection architecture) and OCR which detects the vehicles and their number plates and identifies the other details of the vehicle.

Data set :
The dataset contains images taken from inside a vehicle driving through regular traffic in an urban environment and also taken from various different parking lots.
Dataset collected from kaggle website.
This dataset contains vehicles of different countries, so we will use this dataset for detection and recognition purposes only.(Not for identification purposes)

Dataset link:-https://www.kaggle.com/andrewmvd/car-plate-detection

3. Language ,tools and model

- Python (3.8.3)notebook Using Conda environment system as well as Jupyter server on Google colab
- OpenCV is a helpful library for the study of computer vision.
- Matplotlib.pylab is a module which provides a matlab type namespace in python
- Glob helps to return files paths related to a specific pattern in code.
- Numpy this module helps to deal with arrays and matrices easily.
- JupyterLab

Research Papers:

1. https://www.irjet.net/archives/V8/i1/IRJET-V8I1363.pdf
2. https://ieeexplore.ieee.org/document/8489629
3. https://www.researchgate.net/publication/236888959_Automatic_Number_Plate_Recognition_System_ANPR_A_Survey
4. https://ieeexplore.ieee.org/document/8748287
5. https://ieeexplore.ieee.org/document/9236870

Scope :

Real time number plate recognition plays an important role in maintaining law enforcement and maintaining traffic rules. It has wide application areas such as toll plaza, parking area, highly security areas, boarder's areas etc. Number plate recognition is designed to identify the number plate and then recognize the vehicle number plate from a moving vehicle automatically.
The automatic vehicle recognition system plays a major role in detecting threats to defense. Also it can improve the security related to the women and children as they can easily detect the number plate before using cab or other services. The system robustness can be increased if bright and sharp cameras are used.

Methodology :

The project works in 3 models - detection, recognition and identification.
Our project will be divided into 3 steps :

Step1 : Number plate detection
Number plate extraction is the stage where the vehicle number plate is detected and extracted from the image/video or we can call it simply as Number Plate Localization..
Using the following extraction technique:
1. Convert original image to grayscale
2. Apply bilateral filter on grayscale image
3. Then using edges we pick the best approximation-contour with four corners.

In order to detect Number Plates, we will use Yolo (You Only Look One) Deep learning object detection architecture based on convolution neural networks.

Yolo is a single network trained end to end to perform a regression task predicting both object bounding box and object class.This architecture was introduced by Joseph Redmon, Ali Farhadi, Ross Girshick and Santosh Divvala.
YOLO is better than R-CNN because R-CNN takes a huge amount of time to train the model, however YOLO takes comparatively very less time to train the same model.

Step2 : Optical Character Recognition
Once the number plates have been successfully detected, they will be cropped from the image using their bounding box coordinates. Image processing operations will be applied to convert the image to grayscale, sharpen it and enhance it. At last the optical character information will be converted into encoded text. The characters are recognized using Template matching. The final output will be in the form of a string of characters.

Step3 : Vehicle and Owner Identification

The identification of the vehicle owner is part of the identification model. After the successful detection of the vehicle number plate we can design the system for identification of the owner details. We can check the various available databases(of RTO) or we can create our local database.

Here we have two approach(Known and Unknown Vehicles) :
First we can fetch all the vehicle details such as Vehicle Owner Name, vehicle color, chassis number, PUC, Insurance and other details from any state or Central Government Site. Here our model needs to surpass the captcha and other security measures taken by the government.

Secondly, we can manage our local database for our Colony residents to receive a notification everytime when daily utility services like postman, milkman or trash-collecting van arrives by storing the numbers of their vehicles in our database after detection. All residents would have the option to feed in certain vehicles as guest number plates. Our model would tag such plates with the "Guest" tag and Vehicles with unknown Licence Number will be tagged as "Unknown Visitor".

4. Implementation :

- From google colab imported drive

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

- Cloned Yolov5 model

```python
# Clone YOLOv5 code
!git clone https://github.com/rkuo2000/yolov5
%cd yolov5
```

```
Cloning into 'yolov5'...
remote: Enumerating objects: 3390, done.
remote: Total 3390 (delta 0), reused 0 (delta 0), pack-reused 3390
Receiving objects: 100% (3390/3390), 7.02 MiB | 25.69 MiB/s, done.
Resolving deltas: 100% (2276/2276), done.
/content/yolov5
```

- Accessed the Car images dataset

```
!unzip /content/drive/My\ Drive/dataset_cars.zip
```

```
rchive:  /content/drive/My Drive/dataset_cars.zip
   creating: dataset_cars/
   creating: dataset_cars/annotations/
  inflating: dataset_cars/annotations/Cars0.xml
  inflating: dataset_cars/annotations/Cars1.xml
  inflating: dataset_cars/annotations/Cars10.xml
  inflating: dataset_cars/annotations/Cars100.xml
  inflating: dataset_cars/annotations/Cars101.xml
  inflating: dataset_cars/annotations/Cars102.xml
  inflating: dataset_cars/annotations/Cars103.xml
  inflating: dataset_cars/annotations/Cars104.xml
  inflating: dataset_cars/annotations/Cars105.xml
  inflating: dataset_cars/annotations/Cars106.xml
  inflating: dataset_cars/annotations/Cars107.xml
  inflating: dataset_cars/annotations/Cars108.xml
  inflating: dataset_cars/annotations/Cars109.xml
  inflating: dataset_cars/annotations/Cars11.xml
  inflating: dataset_cars/annotations/Cars110.xml
  inflating: dataset_cars/annotations/Cars111.xml
  inflating: dataset_cars/annotations/Cars112.xml
  inflating: dataset_cars/annotations/Cars113.xml
  inflating: dataset_cars/annotations/Cars114.xml
  inflating: dataset_cars/annotations/Cars115.xml
  inflating: dataset_cars/annotations/Cars116.xml
  inflating: dataset_cars/annotations/Cars117.xml
```

- Train and test split of dataset

```
!echo "train: Dataset/train/images" > data/alpr.yaml
!echo "val:   Dataset/train/images" >> data/alpr.yaml

!echo "nc : 1" >> data/alpr.yaml
!echo "names: ['license']" >> data/alpr.yaml

!cat data/alpr.yaml
```

```
train: Dataset/train/images
val:   Dataset/train/images
nc : 1
names: ['license']
```

- Prepared the Dataset

```
FILE_ROOT = "dataset_cars/"
IMAGE_PATH = FILE_ROOT + "images"
ANNOTATIONS_PATH = FILE_ROOT + "annotations"

DATA_ROOT = "Dataset/"
DEST_IMAGES_PATH = "train/images"
DEST_LABELS_PATH = "train/labels"
```

```
!mkdir -p Dataset/train/labels
```

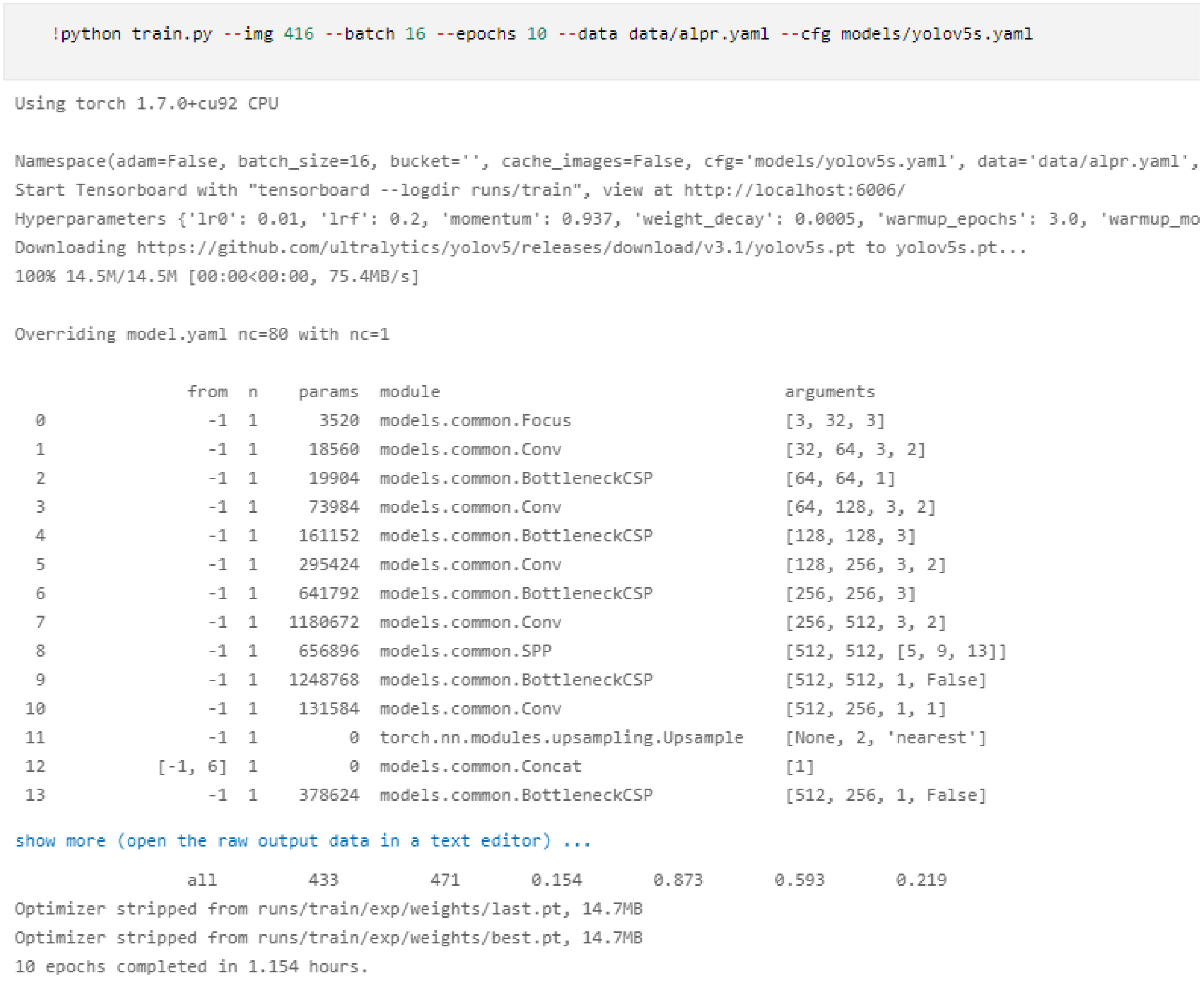
```
# copy images
!mkdir -p Dataset/train
!cp -rf dataset_cars/images Dataset/train
```

```
!mkdir -p Dataset/val
!cp -rf dataset_cars/images/Cars1*.png Dataset/val
```

- Converted COCO animations to YOLOv5 labels

```python
def convert_labels(label_size, input_variables):
    a1 = int(input_variables[0])
    b1 = int(input_variables[1])
    a2 = int(input_variables[2])
    b2 = int(input_variables[3])

    dw = np.float32(1. / int(label_size[0]))
    dh = np.float32(1. / int(label_size[1]))

    width = a2 - a1
    hight = b2 - b1
    x = a1 + (width / 2)
    y = b1 + (hight / 2)

    x = x * dw
    width = width * dw
    y = y * dh
    hight = hight * dh
    return [x, y, width, hight]

def saving_file(my_file, size):
    classes = ['license']
    temp = DATA_ROOT + DEST_LABELS_PATH + '/' + my_file + '.txt'
    path = open(temp, "a+")
    for box in img_box:
        label = convert_labels(size, box[1:])
        cls_num = 0
        path.write(f"{cls_num} {label[0]} {label[1]} {label[2]} {label[3]}\n")
    path.flush()
    path.close()

def get_xml(path, image_xml):
    image_path = path + '/' + image_xml + '.xml'

    dom = parse(image_path)
    temp = dom.documentElement
    img_name = temp.getElementsByTagName("filename")[0].childNodes[0].data
    img_size = temp.getElementsByTagName("size")[0]
    items = temp.getElementsByTagName("object")
    img_h = img_size.getElementsByTagName("height")[0].childNodes[0].data
    img_w = img_size.getElementsByTagName("width")[0].childNodes[0].data
    img_c = img_size.getElementsByTagName("depth")[0].childNodes[0].data
    image_variables = []
    for label in items:
        cls_name = label.getElementsByTagName("name")[0].childNodes[0].data
        a1 = int(label.getElementsByTagName("xmin")[0].childNodes[0].data)
        b1 = int(label.getElementsByTagName("ymin")[0].childNodes[0].data)
        a2 = int(label.getElementsByTagName("xmax")[0].childNodes[0].data)
        b2 = int(label.getElementsByTagName("ymax")[0].childNodes[0].data)
        image_variables.append([cls_name, a1, b1, a2, b2])
    saving_file(image_xml, [img_w, img_h], image_box)
```

- YOLOv5 TRAINING

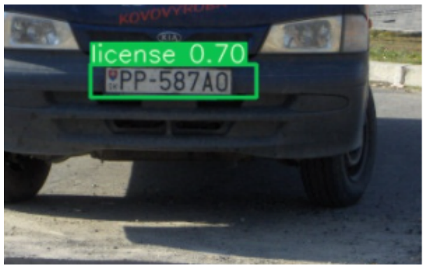


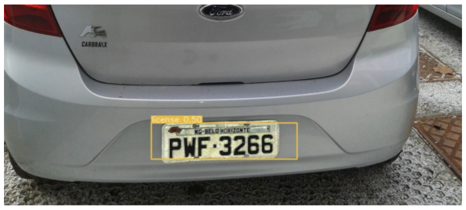
```
!python train.py --img 416 --batch 16 --epochs 10 --data data/alpr.yaml --cfg models/yolov5s.yaml
```

```
Using torch 1.7.0+cu92 CPU

Namespace(adam=False, batch_size=16, bucket='', cache_images=False, cfg='models/yolov5s.yaml', data='data/alpr.yaml',
Start Tensorboard with "tensorboard --logdir runs/train", view at http://localhost:6006/
Hyperparameters {'lr0': 0.01, 'lrf': 0.2, 'momentum': 0.937, 'weight_decay': 0.0005, 'warmup_epochs': 3.0, 'warmup_mo
Downloading https://github.com/ultralytics/yolov5/releases/download/v3.1/yolov5s.pt to yolov5s.pt...
100% 14.5M/14.5M [00:00<00:00, 75.4MB/s]

Overriding model.yaml nc=80 with nc=1

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Focus                     [3, 32, 3]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     19904  models.common.BottleneckCSP             [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  1    161152  models.common.BottleneckCSP             [128, 128, 3]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  1    641792  models.common.BottleneckCSP             [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1    656896  models.common.SPP                       [512, 512, [5, 9, 13]]
  9                -1  1   1248768  models.common.BottleneckCSP             [512, 512, 1, False]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    378624  models.common.BottleneckCSP             [512, 256, 1, False]

show more (open the raw output data in a text editor) ...

                 all         433         471       0.154       0.873       0.593       0.219
Optimizer stripped from runs/train/exp/weights/last.pt, 14.7MB
Optimizer stripped from runs/train/exp/weights/best.pt, 14.7MB
10 epochs completed in 1.154 hours.
```

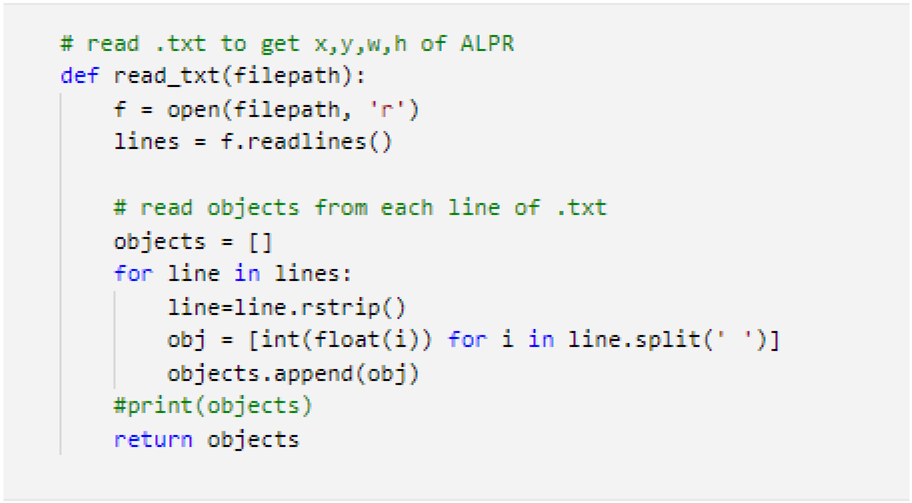
- YOLOv5 Detect and Display detected Images

Examples -



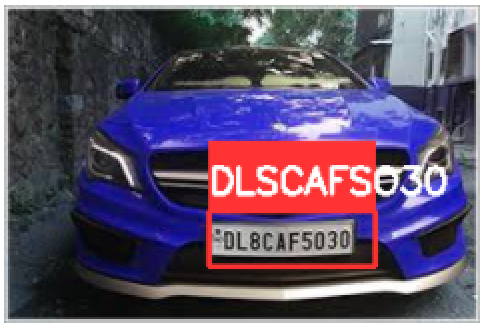


- OCR get detected alpr bounding box



```python
# read .txt to get x,y,w,h of ALPR
def read_txt(filepath):
    f = open(filepath, 'r')
    lines = f.readlines()

    # read objects from each line of .txt
    objects = []
    for line in lines:
        line=line.rstrip()
        obj = [int(float(i)) for i in line.split(' ')]
        objects.append(obj)
    #print(objects)
    return objects
```

```
DLSCAFSO30
Car Belongs to Delhi
DLSCAFSO30
```



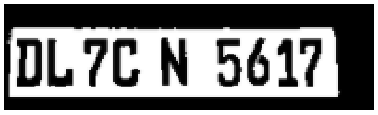
- Finally we fed the output of OCR to csv file of RTO Dataset

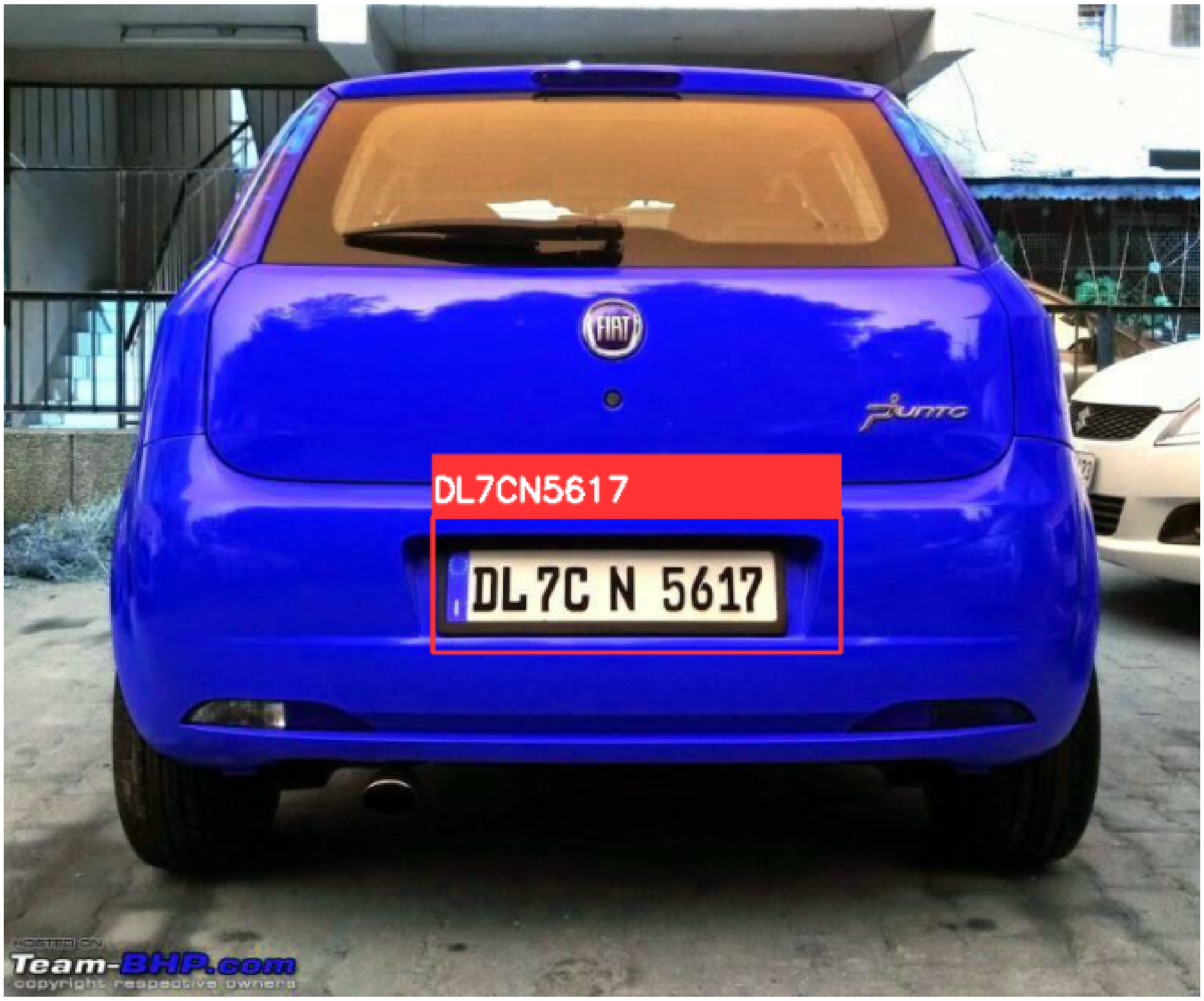
Which will print the address of location RTO from where car was registered

- Results

```
# Let's make a function call
extract_num('/content/car4.jpg')
```

```
DL7CN5617
location DY.DIR.ZONAL OFFICE,EAST DELHI,MAYUR VIHAR
state DL:Delhi
DL7CN5617
```





.

Link to RTO dataset of district wise registration of each state of india: https://drive.google.com/file/d/1W8ezZb11rej-PLF29yigEsc5Y78h4pFp/view?usp=sharing