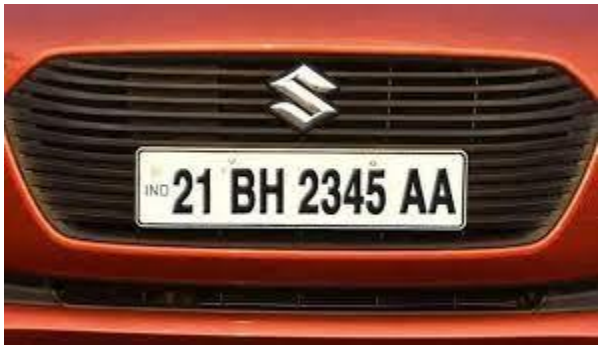
**To run the project run car_license_plate_detection_and_recognition.ipynb file.**

## Car license plate detection and recognition

Problem statement : Detection and recognition of car license plate form images or videos.




Approach:
I have done the project in two modules:
i) detect the car license plate using YOLOv5 model and crop out the ROI
ii) recognise the license plate number from cropped ROI using OCR or Custom CNN model.

**YOLOv5 model for car license plate detection:**
YOLOv5 pytorch model is used to detect the car license plate and crop out the detected ROI.
Requirements: path
Dataset : I have used two datasets for this model to train.
1st dataset contains —-------- images.
Path:

2nd dataset contains —------ images.
Path

Data is converted to YOLOv5 COCO format using Roboflow.

Training :
To train the model run :

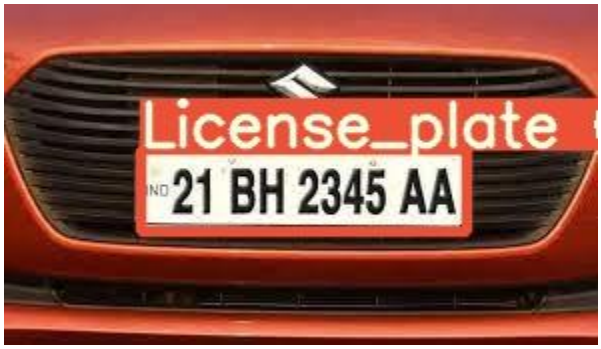
*$ python train.py --img IMG_SIZE --batch BATCH_SIZE --epochs NO_OF_EPOCHES --data data.yaml_FILE_PATH --weights ‘’ --cfg yolov5s.yaml_CONF_FILE_PATH*

All training results are saved to runs/train/ with incrementing run directories, i.e. runs/train/exp2, runs/train/exp3.

Testing/Inference Script for model:

*$ python detect.py - -weights PATH_TO_WEIGHTS  - -source PATH_TO_TEST_IMAGES*

All training results are saved to runs/train/ with incrementing run directories, i.e. runs/detect/exp2, runs/detect/exp3.





Results :
1. I have got 94% precision and 74% recall on the old_dataset of 300 images for 500 epochs.This trained model is unable to detect unfocused or small license plates.
2. I have got 85% precision and 75% recall on new_dataset of 1500 train images for 200 epochs.

## Car license plate number recognition :

For car license number recognition we take the cropped ROI number plate region from model 1 explained above. Some preprocessing steps are applied on the model like resize, grausclaing, thresholding etc. After this we input this to our custom CNN model or pytesseract library.

Requirements:
OpenCV == 4.1.0
Python
Keras
Scikit-Learn

Preprocessing Steps :
resizes it to a dimension such that all characters seem distinct and clear

convert the colored image to a grey scaled image i.e instead of 3 channels (BGR), the image only has a single 8-bit channel with values ranging from 0–255 where 0 corresponds to black and 255 corresponds to white. We do this to prepare the image for the next process.

now the threshold function converts the grey scaled image to a binary image i.e each pixel will now have a value of 0 or 1 where 0 corresponds to black and 1 corresponds to white. It is done by applying a threshold that has a value between 0 and 255, here the value is 200 which means in the grayscaled image for pixels having a value above 200, in the new binary image that pixel will be given a value of 1. And for pixels having value below 200, in the new binary image that pixel will be given a value of 0.

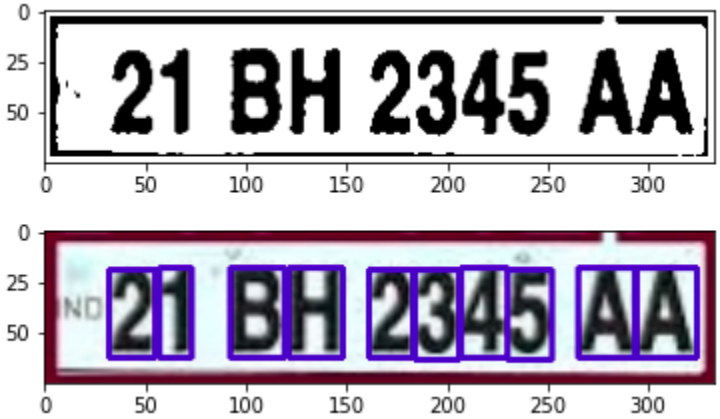
The image is now in binary form and ready for the next process Eroding. Eroding is a simple process used for removing unwanted pixels from the object's boundary meaning pixels that should have a value of 0 but are having a value of 1. It works by considering each pixel in the image one by one and then considering the pixel's neighbor (the number of neighbors depends on the kernel size), the pixel is given a value 1 only if all its neighboring pixels are 1, otherwise it is given a value of 0.

The image is now clean and free of boundary noise, we will now dilate the image to fill up the absent pixels meaning pixels that should have a value of 1 but are having value 0. The function works similar to eroding but with a little catch, it works by considering each pixel in the image one by one and then considering the pixel's neighbor (the number of neighbors depends on the kernel size), the pixel is given a value 1 if at least one of its neighboring pixels is 1.

The next step now is to make the boundaries of the image white. This is to remove any out of the frame pixel in case it is present.

Next, we define a list of dimensions that contains 4 values with which we'll be comparing the character's dimensions for filtering out the required characters.

Through the above processes, we have reduced our image to a processed binary image and we are ready to pass this image for character extraction.




For character extraction the opencv counters  module is used.

The function cv2.findContours returns all the contours it finds in the image. Contours can be explained simply as a curve joining all the continuous points (along the boundary), having the same color or intensity.

We have used a custom CNN model for the classification of single extracted characters between 0 to 9 and A to Z.
Training is performed on GoogleColab. We have trained the model for 100 epochs and got 97% accuracy on validation images.