

# **CS 5542 Big Data Analytics and Applications**

## **Project report**

### **Deep learning-based license plate recognition system**

**Team number:** 8

**Team Members:**

Karthik Yanagandula

Gayathri Garikapati

Amarnadha Reddy Ankireddypalli

Bindu Madhavi Valiveti

#### **Objective:**

As part of this project, we intend to create an application with a trained deep learning CNN model using the RTO API and this model is used for recognizing the characters on the vehicle's license plate from an image. By processing this image and utilizing the RTO API we can retrieve the owner's information. Finally, we are displaying all this information using Flask web application.

#### **Scope of the project:**

With the increasing number of vehicles and traffic violation has been the major cause of the road accidents. Even though the road safety rules and regulations are in place the violators are still increasing. Having a system to identify these violators will assist the law enforcement to impose the road safety rules and reduce the road accidents.

#### **Implementation:**

The main aim of our project is to retrieve the vehicle information with the help of image processing using license plate number. This work uses a real time embedded Vehicle Plate Number Recognition system to identify the license plate number.

Below are the requirements to create and train the model

1. Here we are using Google Colab and installing the required Python and Deep Learning libraries.
2. Data that contains the image or video using which we can detect the plate number and get the vehicle information.

3. Flask application installation
4. RTO API

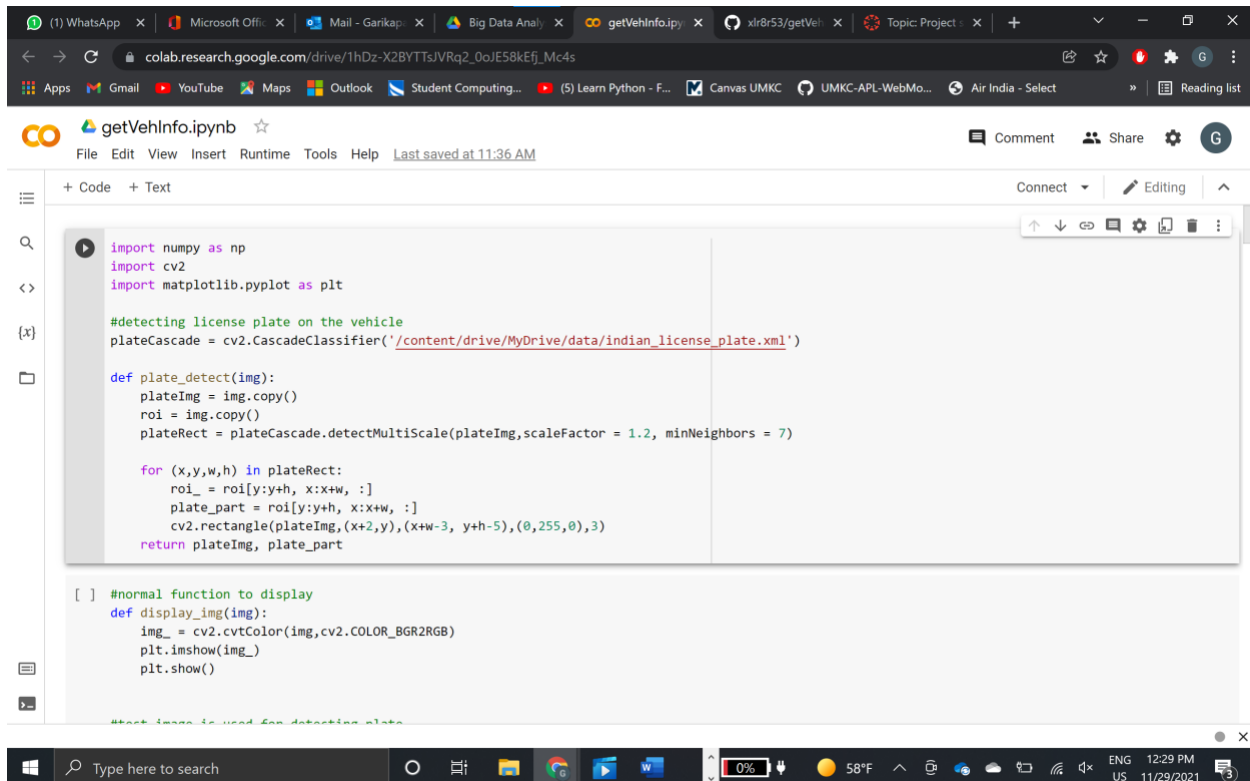
We have followed the below steps to create and train the CNN deep learning model for vehicle number plate detection.

Imported the required libraries such as Numpy, Matplotlib and Cv2. Here we are using Cascaded Classifier whose main purpose is to detect a particular region inside an image and here it is to identify the vehicle number plate region. Usually, these Cascading classifiers are trained with many positive and negative sample and this trained model is used to identify a particular region in an image.

Plate\_detect() function is used to detect the vehicle number plate and mark it in green colour rectangle and crop that part and return it to a function which in turn used by display\_img() function.

display\_img() function is using img as the parameter and converting the BGR colour code to RGB colour code and displaying the image on the screen using matplotlib.

Here we are using car.jpg as the input image i.e., inputImg and using this with plate\_detect() function to retrieve on the number plate.



```
import numpy as np
import cv2
import matplotlib.pyplot as plt

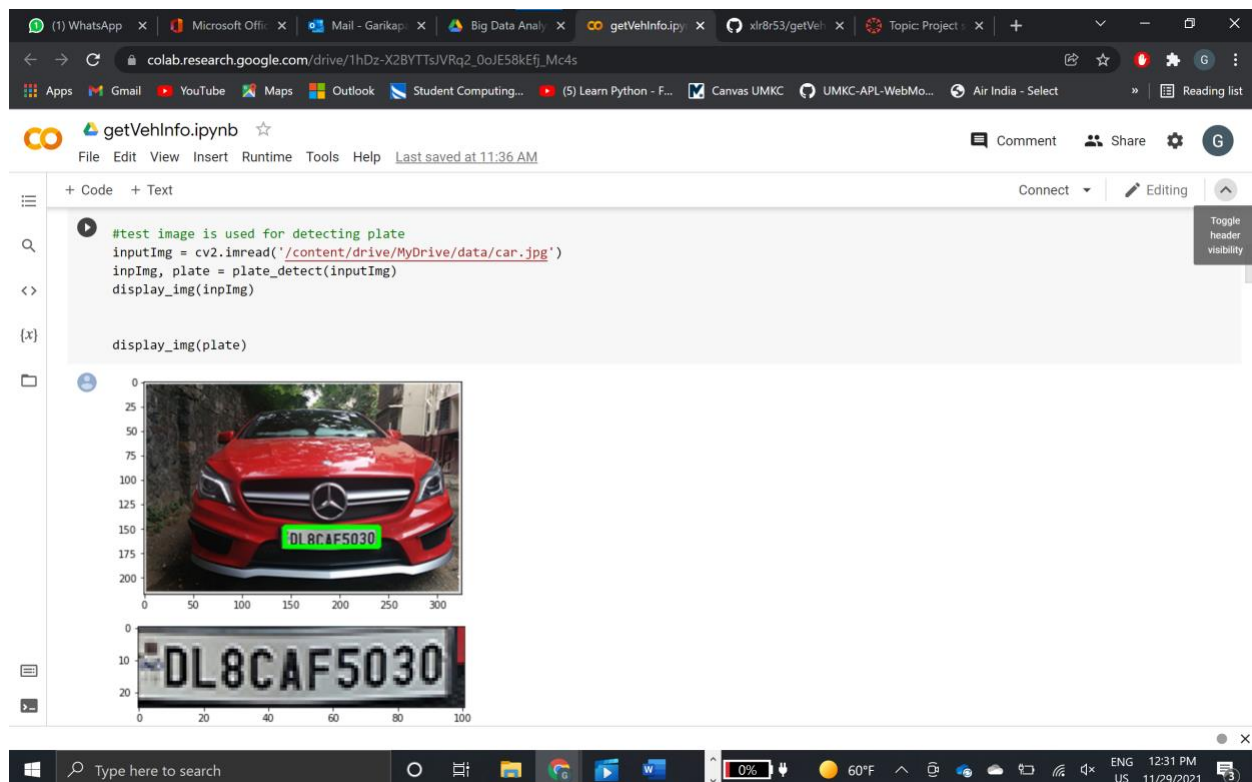
#detecting license plate on the vehicle
plateCascade = cv2.CascadeClassifier('/content/drive/MyDrive/data/indian_license_plate.xml')

def plate_detect(img):
    plateImg = img.copy()
    roi = img.copy()
    plateRect = plateCascade.detectMultiScale(plateImg, scaleFactor = 1.2, minNeighbors = 7)

    for (x,y,w,h) in plateRect:
        roi_ = roi[y:y+h, x:x+w, :]
        plate_part = roi[y:y+h, x:x+w, :]
        cv2.rectangle(plateImg, (x+2,y), (x+w-3, y+h-5), (0,255,0), 3)
    return plateImg, plate_part

[ ] #normal function to display
def display_img(img):
    img_ = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img_)
    plt.show()

#test image is used for detection plate
```



Here we are preprocessing the image using the `find_contours()` function with the parameters such as dimension of the image and the preprocessed image. These parameters are useful in easily extracting the numbers and characters from the license plate.

This function will find the contours (Curve joining all the continuous points) in the image i.e., the outline or shape of the numbers with their position and create a rectangle around them and using this function with CV2 library we can get the boundaries of an object in the image. After identifying and cropping the numbers they are extracted and stored in `img_res_copy` in the form of array.

WhatsApp x Microsoft Offi x Mail - Garikap x Big Data Analy x getVehInfo.py x xlr8r53/getVel x Topic: Project x +

colab.research.google.com/drive/1hDz-X2BYTtsjVRq2\_0oJE58kEfj\_Mc4s#scrollTo=8DhJBEDxt5sp

Apps Gmail YouTube Maps Outlook Student Computing... (5) Learn Python - F... Canvas UMKC UMKC-APL-WebMo... Air India - Select Reading list

getVehInfo.ipynb

File Edit View Insert Runtime Tools Help Last saved at 11:36 AM

+ Code + Text

```
def find_contours(dimensions, img) :  
  
    #finding all contours in the image using  
    #retrieval mode: RETR_TREE  
    #contour approximation method: CHAIN_APPROX_SIMPLE  
    cnts, _ = cv2.findContours(img.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)  
  
    #Approx dimensions of the contours  
    lower_width = dimensions[0]  
    upper_width = dimensions[1]  
    lower_height = dimensions[2]  
    upper_height = dimensions[3]  
  
    #Check largest 15 contours for license plate character respectively  
    cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:15]  
  
    ci = cv2.imread('contour.jpg')  
  
    x_cnr_list = []  
    target_contours = []  
    img_res = []  
    for cntr in cnts :  
        #detecting contour in binary image and returns the coordinates of rectangle enclosing it  
        intX, intY, intWidth, intHeight = cv2.boundingRect(cntr)  
  
        #checking the dimensions of the contour to filter out the characters by contour's size  
        if intWidth > lower_width and intWidth < upper_width and intHeight > lower_height and intHeight < upper_height :  
            x_cnr_list.append(intX)
```

Type here to search 0% 60°F ENG 12:48 PM US 11/29/2021

WhatsApp x Microsoft Offi x Mail - Garikap x Big Data Analy x getVehInfo.py x xlr8r53/getVel x Topic: Project x +

colab.research.google.com/drive/1hDz-X2BYTtsjVRq2\_0oJE58kEfj\_Mc4s#scrollTo=8DhJBEDxt5sp

Apps Gmail YouTube Maps Outlook Student Computing... (5) Learn Python - F... Canvas UMKC UMKC-APL-WebMo... Air India - Select Reading list

getVehInfo.ipynb

File Edit View Insert Runtime Tools Help Last saved at 11:36 AM

+ Code + Text

```
if intWidth > lower_width and intWidth < upper_width and intHeight > lower_height and intHeight < upper_height :  
    x_cnr_list.append(intX)  
    char_copy = np.zeros((44,24))  
    #extracting each character using the enclosing rectangle's coordinates.  
    char = img[intY:intY+intHeight, intX:intX+intWidth]  
    char = cv2.resize(char, (20, 40))  
    cv2.rectangle(ci, (intX,intY), (intX+intWidth, intY+intHeight), (50,21,200), 2)  
    plt.imshow(char, cmap='gray')  
    char = cv2.subtract(255, char)  
    char_copy[2:42, 2:22] = char  
    char_copy[0:2, :] = 0  
    char_copy[:, 0:2] = 0  
    char_copy[42:44, :] = 0  
    img_res.append(char_copy) # List that stores the character's binary image (unsorted)  
  
#return characters on ascending order with respect to the x-coordinate  
  
plt.show()  
#arbitrary function that stores sorted list of character indeces  
indices = sorted(range(len(x_cnr_list)), key=lambda k: x_cnr_list[k])  
img_res_copy = []  
for idx in indices:  
    img_res_copy.append(img_res[idx])# stores character images according to their index  
img_res = np.array(img_res_copy)  
  
return img_res
```

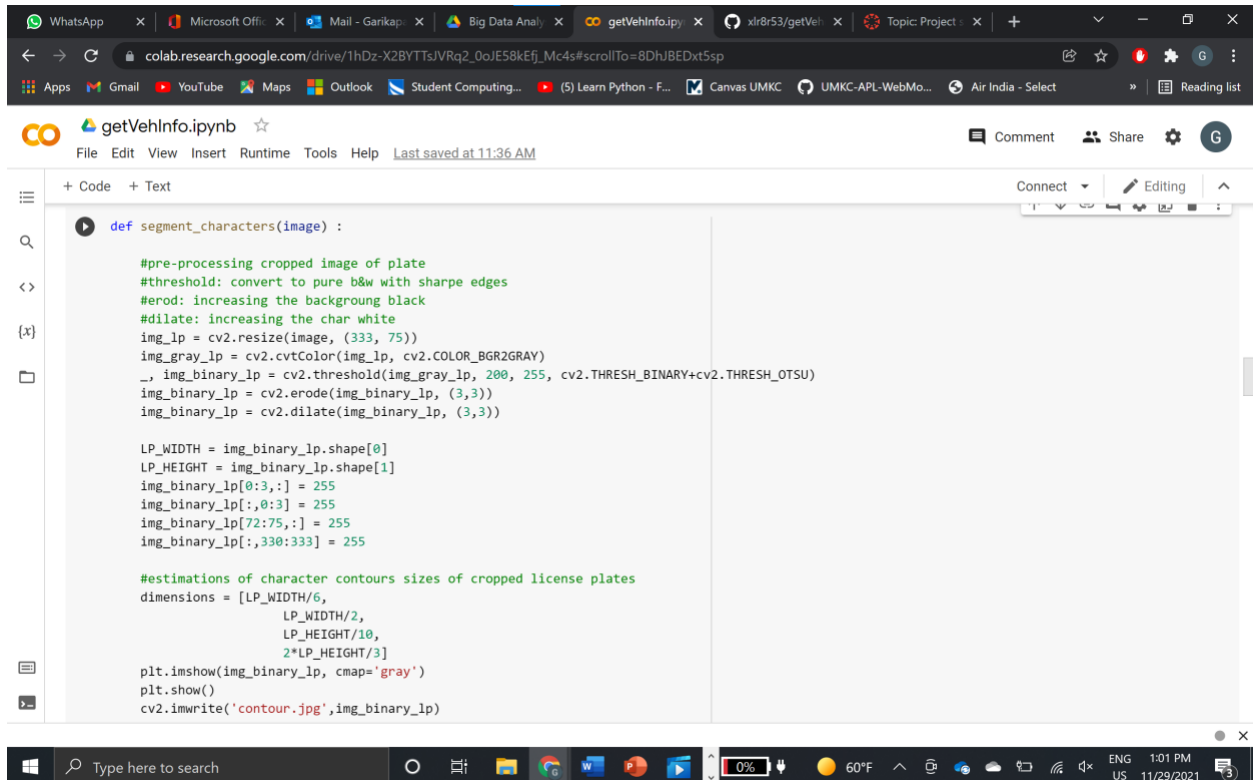
Type here to search 0% 60°F ENG 12:48 PM US 11/29/2021

## Character segmentation:

The technique of segmenting an image into multiple pieces is known as image segmentation and its main purpose is to turn the image into more meaningful.

As part of `segment_character()` function

- Pre processing the cropped image of the number plate
- Thresholding i.e., converting the image to pure black and white
- Erosion i.e., increasing the background to black
- Dilation which is increasing the character colour intensity to white.



The screenshot shows a Jupyter Notebook titled 'getVehInfo.ipynb' in a web browser. The notebook contains a Python function named `segment_characters(image)`. The function performs the following steps:

- Pre-processing the cropped image of the plate.
- Thresholding: converting the image to pure black and white using `cv2.cvtColor` and `cv2.threshold`.
- Erosion: increasing the background to black using `cv2.erode`.
- Dilation: increasing the character white using `cv2.dilate`.
- Estimations of character contours sizes of cropped license plates using `plt.imshow` and `cv2.imwrite`.

```
def segment_characters(image) :  
  
    #pre-processing cropped image of plate  
    #threshold: convert to pure b&w with sharpe edges  
    #erod: increasing the backgroung black  
    #dilate: increasing the char white  
    img_lp = cv2.resize(image, (333, 75))  
    img_gray_lp = cv2.cvtColor(img_lp, cv2.COLOR_BGR2GRAY)  
    _, img_binary_lp = cv2.threshold(img_gray_lp, 200, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)  
    img_binary_lp = cv2.erode(img_binary_lp, (3,3))  
    img_binary_lp = cv2.dilate(img_binary_lp, (3,3))  
  
    LP_WIDTH = img_binary_lp.shape[0]  
    LP_HEIGHT = img_binary_lp.shape[1]  
    img_binary_lp[0:3,:] = 255  
    img_binary_lp[:,0:3] = 255  
    img_binary_lp[72:75,:] = 255  
    img_binary_lp[:,330:333] = 255  
  
    #estimations of character contours sizes of cropped license plates  
    dimensions = [LP_WIDTH/6,  
                  LP_WIDTH/2,  
                  LP_HEIGHT/10,  
                  2*LP_HEIGHT/3]  
  
    plt.imshow(img_binary_lp, cmap='gray')  
    plt.show()  
    cv2.imwrite('contour.jpg',img_binary_lp)
```

WhatsApp Microsoft Office Mail - Garikap Big Data Analy getVehInfo.ipynb xlr8r53/getVeh Topic: Project +

colab.research.google.com/drive/1hDz-X2BYTtsjVRq2\_0oJE58kEfj\_Mc4s#scrollTo=8DhJBEDxt5sp

Apps Gmail YouTube Maps Outlook Student Computing... (5) Learn Python - F... Canvas UMKC UMKC-APL-WebMo... Air India - Select Reading list


getVehInfo.ipynb

File Edit View Insert Runtime Tools Help Last saved at 11:36 AM

+ Code + Text

```
[ ]  
LP_WIDTH/2,  
LP_HEIGHT/10,  
2*LP_HEIGHT/3]  
plt.imshow(img_binary_lp, cmap='gray')  
plt.show()  
cv2.imwrite('contour.jpg',img_binary_lp)  
  
#getting contours  
char_list = find_contours(dimensions, img_binary_lp)  
  
return char_list
```

char = segment\_characters(plate)



0 25 50  
0 50 100 150 200 250 300

0 25 50  
0 50 100 150 200 250 300

Type here to search 0% 60°F 1:01 PM US 11/29/2021

WhatsApp Microsoft Office Mail - Garikap Big Data Analy getVehInfo.ipynb xlr8r53/getVeh Topic: Project +

colab.research.google.com/drive/1hDz-X2BYTtsjVRq2\_0oJE58kEfj\_Mc4s#scrollTo=8DhJBEDxt5sp

Apps Gmail YouTube Maps Outlook Student Computing... (5) Learn Python - F... Canvas UMKC UMKC-APL-WebMo... Air India - Select Reading list


getVehInfo.ipynb

File Edit View Insert Runtime Tools Help Last saved at 11:36 AM

+ Code + Text

```
[ ] return char_list
```

char = segment\_characters(plate)




0 25 50  
0 50 100 150 200 250 300

0 25 50  
0 50 100 150 200 250 300

+ Code + Text

```
[ ] for i in range(10):  
plt.subplot(1, 10, i+1)  
plt.imshow(char[i], cmap='gray')  
plt.axis('off')
```



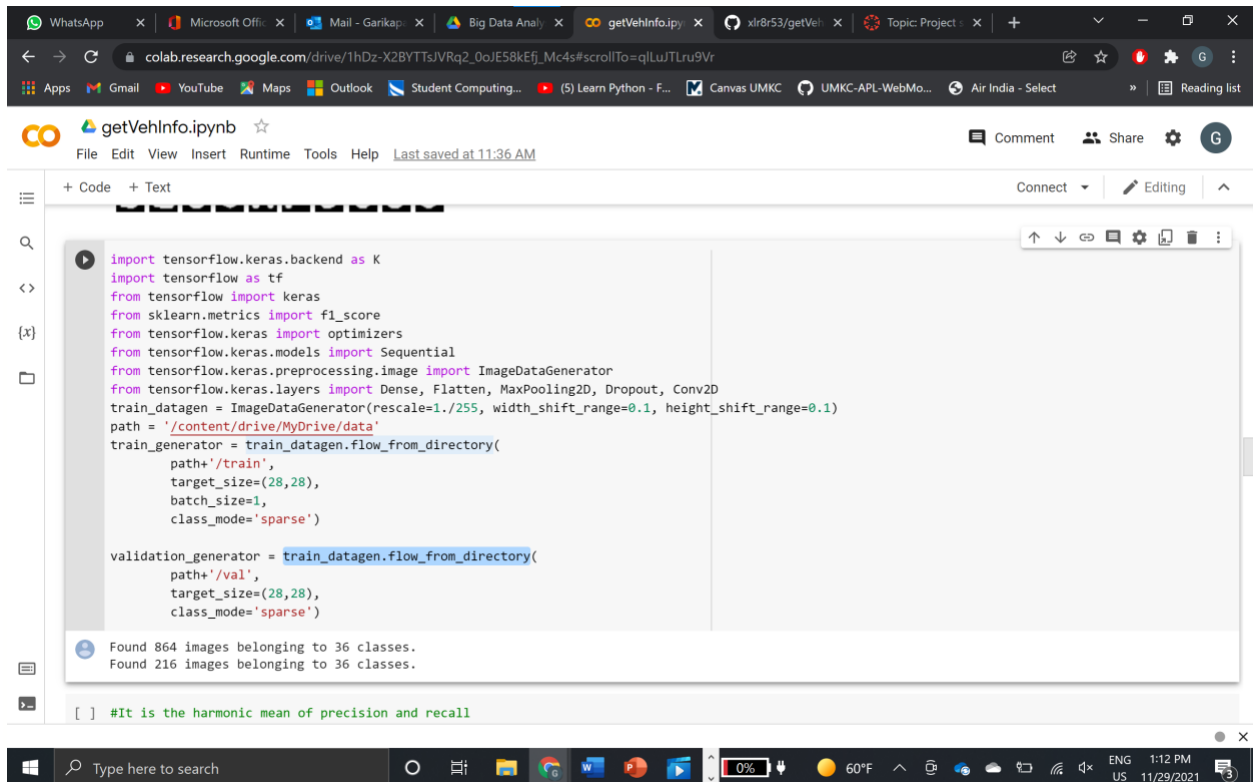
DL8CAF5030

Type here to search 0% 60°F 1:02 PM US 11/29/2021

## Image Augmentation:

Imported the required libraries such as Sklearn, Tensorflow and Keras model. Creating the augmented image data using the preprocessed dataset with the Keras ImageDataGenerator which allows us to make real time enhancements to the images while the model is still training so that we can apply the transformation to each training image.

`train_datagen.flow_from_directory()` function will give the path of the training and validation dataset.



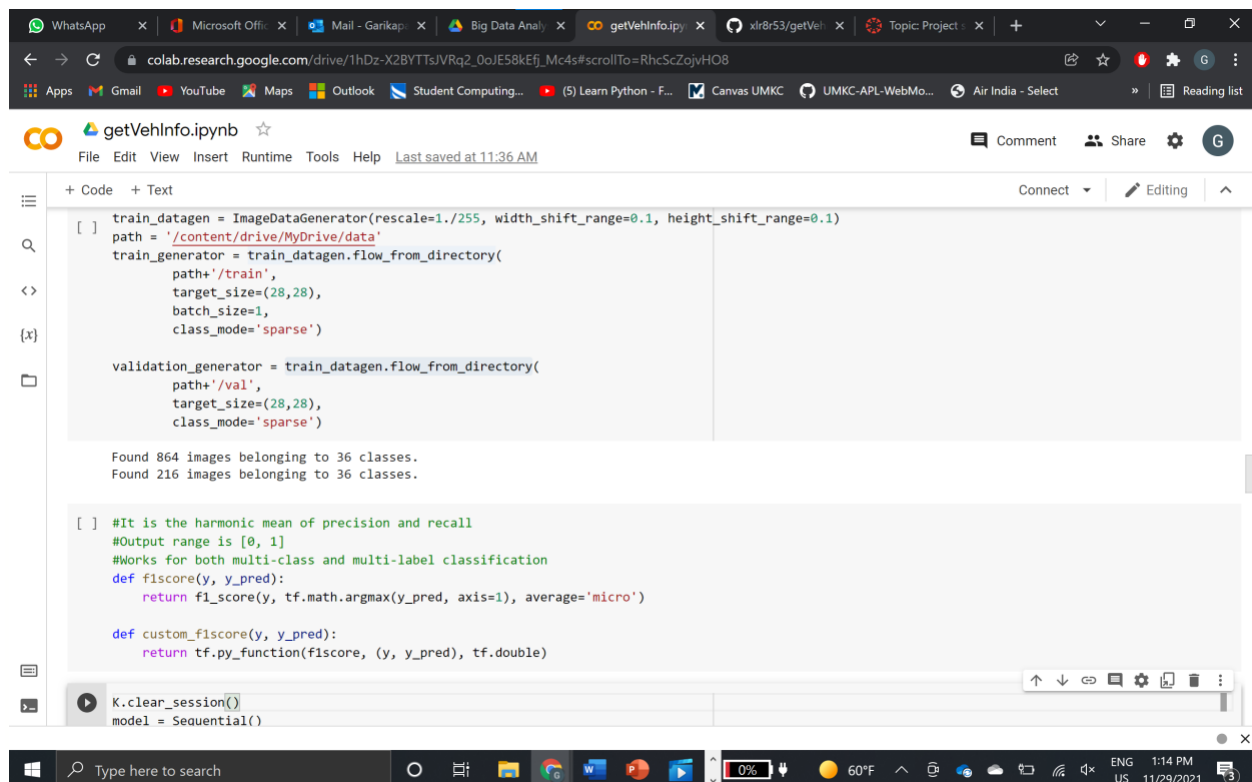
```
import tensorflow.keras.backend as K
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import f1_score
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Flatten, MaxPooling2D, Dropout, Conv2D
train_datagen = ImageDataGenerator(rescale=1./255, width_shift_range=0.1, height_shift_range=0.1)
path = '/content/drive/MyDrive/data'
train_generator = train_datagen.flow_from_directory(
    path+ '/train',
    target_size=(28,28),
    batch_size=1,
    class_mode='sparse')

validation_generator = train_datagen.flow_from_directory(
    path+ '/val',
    target_size=(28,28),
    class_mode='sparse')
```

Found 864 images belonging to 36 classes.  
Found 216 images belonging to 36 classes.

[ ] #It is the harmonic mean of precision and recall

Defined the `f1score` and `custom_f1score` to train the model with more precision and accuracy.



```
[ ] train_datagen = ImageDataGenerator(rescale=1./255, width_shift_range=0.1, height_shift_range=0.1)
path = '/content/drive/MyDrive/data'
train_generator = train_datagen.flow_from_directory(
    path+'train',
    target_size=(28,28),
    batch_size=1,
    class_mode='sparse')

validation_generator = train_datagen.flow_from_directory(
    path+'val',
    target_size=(28,28),
    class_mode='sparse')

Found 864 images belonging to 36 classes.
Found 216 images belonging to 36 classes.

[ ] #It is the harmonic mean of precision and recall
#Output range is [0, 1]
#Works for both multi-class and multi-label classification
def f1_score(y, y_pred):
    return f1_score(y, tf.math.argmax(y_pred, axis=1), average='micro')

def custom_f1_score(y, y_pred):
    return tf.py_function(f1_score, (y, y_pred), tf.double)

K.clear_session()
model = Sequential()
```

Creating and training the model:

Using Keras sequential model we have added four Conv2D layers with the activation function as 'relu'. Added the MaxPooling2D layer which is used for ordering the layers within a convolutional neural network and this layer helps in reducing the number of parameters and the computation of the network.

Added a Dropout() layer which helps in preventing the model from overfitting and next we have added the Flatten() layer which helps in converting the data into one dimensional array.

Added Dense() layers with the activation functions 'relu' and 'softmax'. After this we are compiling the model using "sparse\_categorical\_crossentropy" loss function and "Adam" optimizer.



The screenshot shows a Jupyter Notebook in Google Colab. The code defines a sequential model with five layers: two Conv2D layers (16, 32, 64, 64), one MaxPooling2D layer, and one Dropout layer. The model is compiled with sparse\_categorical\_crossentropy loss and Adam optimizer. A custom callback, stop\_training\_callback, is defined to stop training when the validation custom f1 score exceeds 0.99. The model is then trained for 80 epochs.

```
K.clear_session()
model = Sequential()
model.add(Conv2D(16, (22,22), input_shape=(28, 28, 3), activation='relu', padding='same'))
model.add(Conv2D(32, (16,16), input_shape=(28, 28, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (8,8), input_shape=(28, 28, 3), activation='relu', padding='same'))
model.add(Conv2D(64, (4,4), input_shape=(28, 28, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(4, 4)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(36, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizers.Adam(lr=0.0001), metrics=[custom_f1score])

class stop_training_callback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('val_custom_f1score') > 0.99):
            self.model.stop_training = True

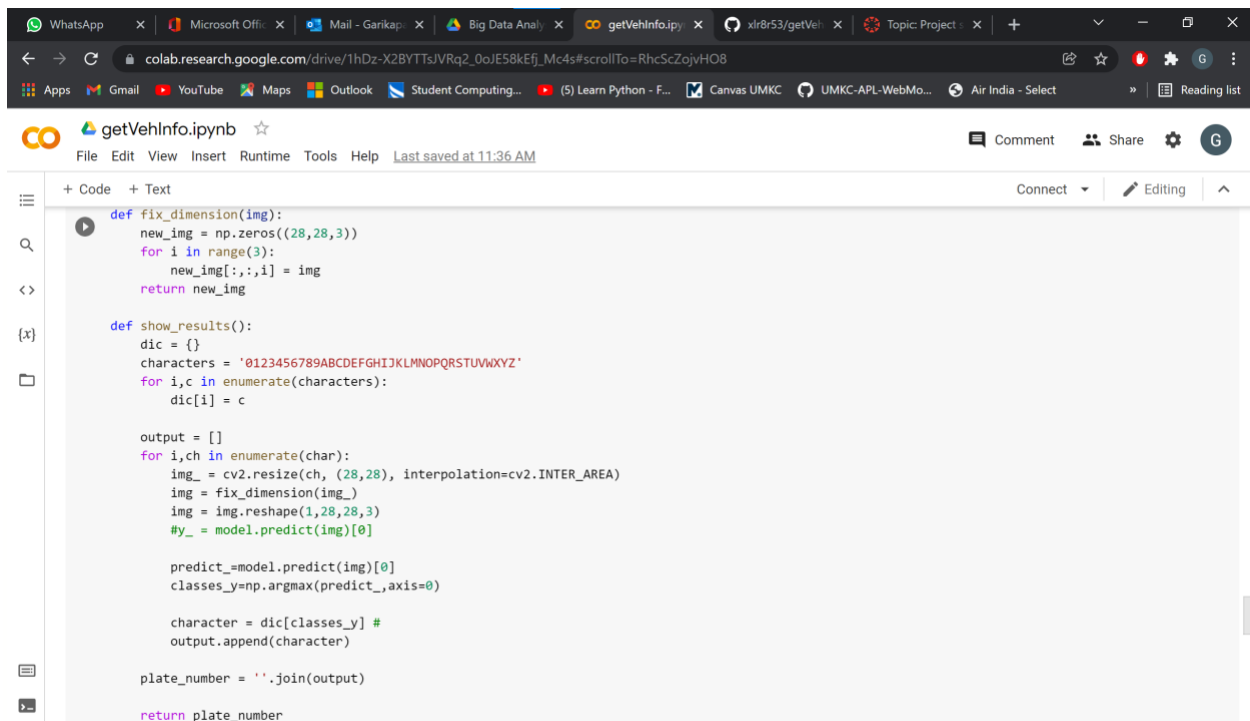
batch_size = 1
callbacks = [stop_training_callback()]
model.fit_generator(
    train_generator,
    steps_per_epoch = train_generator.samples // batch_size,
    validation_data = validation_generator,
    epochs = 80, verbose=1, callbacks=callbacks)
```

The screenshot shows the output of the model training process. It displays a series of lines indicating the progress of training over 14 epochs. Each line shows the epoch number, training time, loss, custom f1 score, validation loss, and validation custom f1 score. The custom f1 score increases from 0.4866 in epoch 1 to 0.9583 in epoch 14, which is above the 0.99 threshold set in the callback, indicating that the training should have stopped.

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: The `lr` argument is deprecated, use
super(Adam, self)._init__(name, **kwargs)
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:27: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future vers:
Epoch 1/80
864/864 [=====] - 347s 400ms/step - loss: 3.1603 - custom_f1score: 0.1262 - val_loss: 1.7699 - val_custom_f1score: 0.4866
Epoch 2/80
864/864 [=====] - 70s 82ms/step - loss: 1.2805 - custom_f1score: 0.6169 - val_loss: 0.5641 - val_custom_f1score: 0.8289
Epoch 3/80
864/864 [=====] - 71s 82ms/step - loss: 0.6401 - custom_f1score: 0.7917 - val_loss: 0.3294 - val_custom_f1score: 0.8869
Epoch 4/80
864/864 [=====] - 70s 81ms/step - loss: 0.3901 - custom_f1score: 0.8773 - val_loss: 0.2318 - val_custom_f1score: 0.9182
Epoch 5/80
864/864 [=====] - 70s 81ms/step - loss: 0.2850 - custom_f1score: 0.9062 - val_loss: 0.1342 - val_custom_f1score: 0.9583
Epoch 6/80
864/864 [=====] - 69s 80ms/step - loss: 0.2145 - custom_f1score: 0.9259 - val_loss: 0.1375 - val_custom_f1score: 0.9554
Epoch 7/80
864/864 [=====] - 69s 80ms/step - loss: 0.2330 - custom_f1score: 0.9236 - val_loss: 0.4170 - val_custom_f1score: 0.8914
Epoch 8/80
864/864 [=====] - 69s 79ms/step - loss: 0.1494 - custom_f1score: 0.9491 - val_loss: 0.0980 - val_custom_f1score: 0.9524
Epoch 9/80
864/864 [=====] - 69s 80ms/step - loss: 0.1424 - custom_f1score: 0.9491 - val_loss: 0.1397 - val_custom_f1score: 0.9628
Epoch 10/80
864/864 [=====] - 68s 79ms/step - loss: 0.1552 - custom_f1score: 0.9502 - val_loss: 0.0690 - val_custom_f1score: 0.9717
Epoch 11/80
864/864 [=====] - 69s 80ms/step - loss: 0.1112 - custom_f1score: 0.9606 - val_loss: 0.0422 - val_custom_f1score: 0.9777
Epoch 12/80
864/864 [=====] - 68s 79ms/step - loss: 0.1246 - custom_f1score: 0.9549 - val_loss: 0.0927 - val_custom_f1score: 0.9732
Epoch 13/80
864/864 [=====] - 68s 79ms/step - loss: 0.1327 - custom_f1score: 0.9514 - val_loss: 0.1365 - val_custom_f1score: 0.9583
Epoch 14/80
```

stop\_training\_callback() function is used to stop the training of the model when the accuracy reaches 99% which is done by checking f1score.

Using below part of the code we are displaying the number plate characters by using the functions `fix_dimension()` and `show_result()` by matching the numbers and characters.



```
def fix_dimension(img):
    new_img = np.zeros((28,28,3))
    for i in range(3):
        new_img[:, :, i] = img
    return new_img

def show_results():
    dic = {}
    characters = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    for i, c in enumerate(characters):
        dic[i] = c

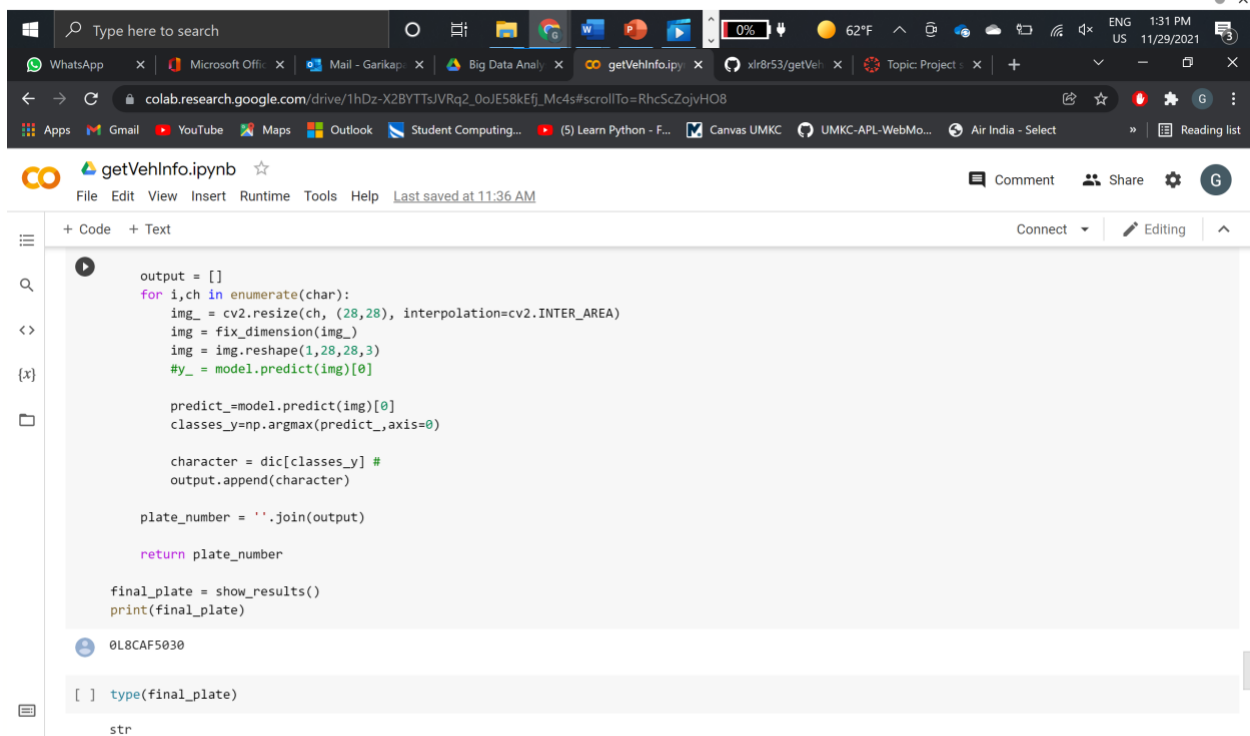
    output = []
    for i, ch in enumerate(char):
        img_ = cv2.resize(ch, (28,28), interpolation=cv2.INTER_AREA)
        img = fix_dimension(img_)
        img = img.reshape(1,28,28,3)
        #y_ = model.predict(img)[0]

        predict_ = model.predict(img)[0]
        classes_y = np.argmax(predict_, axis=0)

        character = dic[classes_y] #
        output.append(character)

    plate_number = ''.join(output)

    return plate_number
```



```
output = []
for i, ch in enumerate(char):
    img_ = cv2.resize(ch, (28,28), interpolation=cv2.INTER_AREA)
    img = fix_dimension(img_)
    img = img.reshape(1,28,28,3)
    #y_ = model.predict(img)[0]

    predict_ = model.predict(img)[0]
    classes_y = np.argmax(predict_, axis=0)

    character = dic[classes_y] #
    output.append(character)

plate_number = ''.join(output)

return plate_number

final_plate = show_results()
print(final_plate)
```

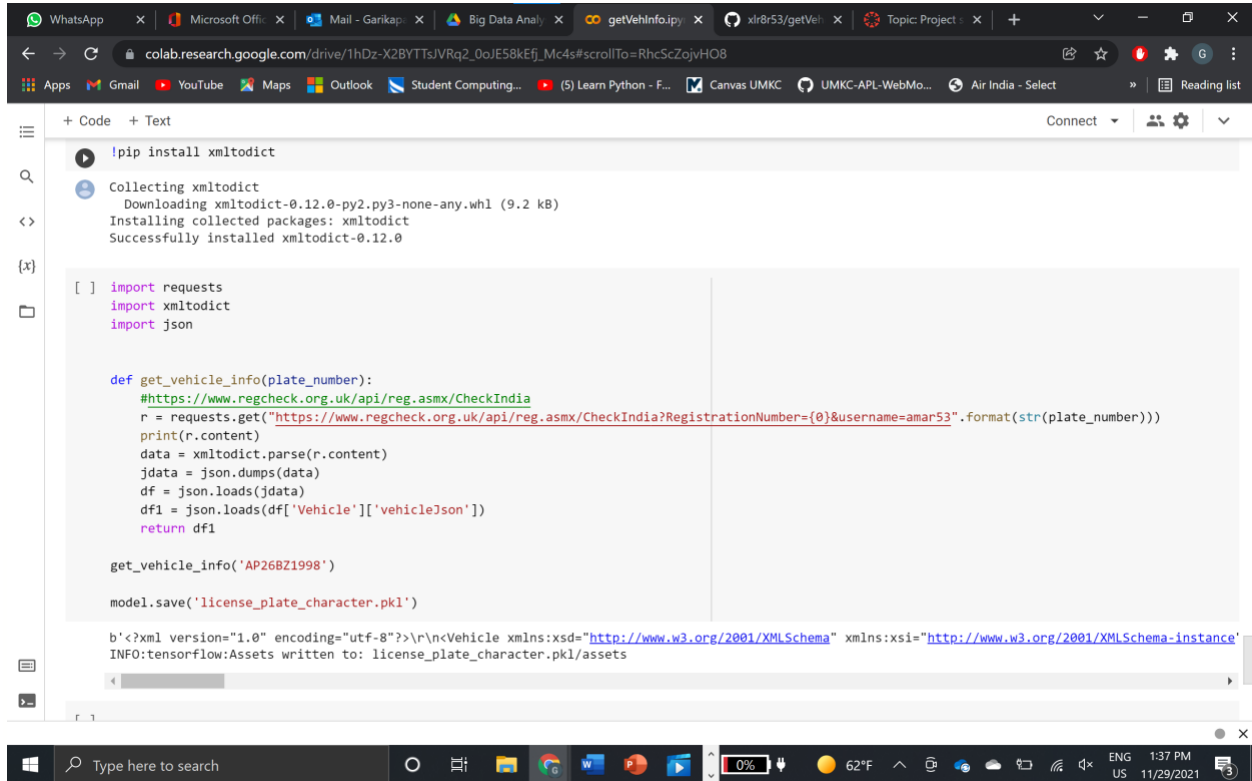
0L8CAF5030

```
[ ] type(final_plate)

str
```

## Model testing using API (Getting vehicle information):

Here we have imported the required libraries such as requests, xmltodict and json. Get\_vehicle\_info() function is used and plate number is passed as a parameter and a request will be sent to API, but here the output we get is XML data. Using xmltodict module we are converting the output to dictionary and in turn this will be converted to JSON.



```
!pip install xmltodict

Collecting xmltodict
  Downloading xmltodict-0.12.0-py2.py3-none-any.whl (9.2 kB)
Installing collected packages: xmltodict
Successfully installed xmltodict-0.12.0

[ ] import requests
import xmltodict
import json

def get_vehicle_info(plate_number):
    #https://www.regcheck.org.uk/api/reg.asmx/CheckIndia
    r = requests.get("https://www.regcheck.org.uk/api/reg.asmx/CheckIndia?RegistrationNumber={}&username=amar53".format(str(plate_number)))
    print(r.content)
    data = xmltodict.parse(r.content)
    jdata = json.dumps(data)
    df = json.loads(jdata)
    df1 = json.loads(df['Vehicle']['vehiclejson'])
    return df1

get_vehicle_info('AP26BZ1998')

model.save('license_plate_character.pkl')

b'<?xml version="1.0" encoding="utf-8"?>\n<n:Vehicle xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
INFO:tensorflow:Assets written to: license_plate_character.pkl/assets'
```

The trained model is saved in the form of pkl file.

## Flask implementation:

**Data:** We have used character dataset which contains the character styles of vehicle number plates. Though we have a smaller number of images for each character, we have used data augmentation by applying different preprocessing techniques. The data contains 30 images of each character i.e., 0-9 and A-Z. We then divided the data into 80/20 for training/validation.

**Use cases:** This type of model can also be used

- To identify the stolen vehicles by comparing the cars passing on the roadside with the list of stolen vehicles in the real time and an alert will be generated when a match is found.

- Detecting number plate system can be useful in calculating the parking fares by identifying the entry and exit timings.
- This system also helps in providing the entry access to the authorized personal.

### **End users:**

The end users of this project might be:

- Highway patrol officers or Traffic policemen
- Security Management of a closed space gateway
- General public and many more

### **Challenges:**

We tried to implement a flask web application to serve the model to end user, we were able to upload the image and view it on the application, but when it comes to predicting the characters using model, we need to use tensorflow package to load the model for prediction, which is not working on Macbook M1 chip due to configuration issues. We tried many ways to solve the issue but couldn't find a solution for the above problem.

Please go through the following link for more information on TF issue for Macbook M1.

<https://github.com/tensorflow/tensorflow/issues/45645>

### **Future Scope:**

The cascade classifier is not able to detect the entire plate region for certain images, which is not good for the application, to avoid this, it's better to give an edit option to change the detected region to user will yield more accuracy in finding the results for vehicle.

This can also be implemented to detect multiple vehicle information and save the results into a csv file to process further on the yielded data.

**Github link:** <https://github.com/xlr8r53/getVehicleInfo>

### **References:**

- <https://medium.com/programming-fever/license-plate-recognition-using-opencv-python-7611f85cdd6c>
- [https://www.researchgate.net/publication/332324949\\_Robust\\_License\\_Plate\\_Recognition\\_using\\_Neural\\_Networks\\_Trained\\_on\\_Synthetic\\_Images](https://www.researchgate.net/publication/332324949_Robust_License_Plate_Recognition_using_Neural_Networks_Trained_on_Synthetic_Images)
- <https://arxiv.org/ftp/arxiv/papers/1912/1912.02441.pdf>