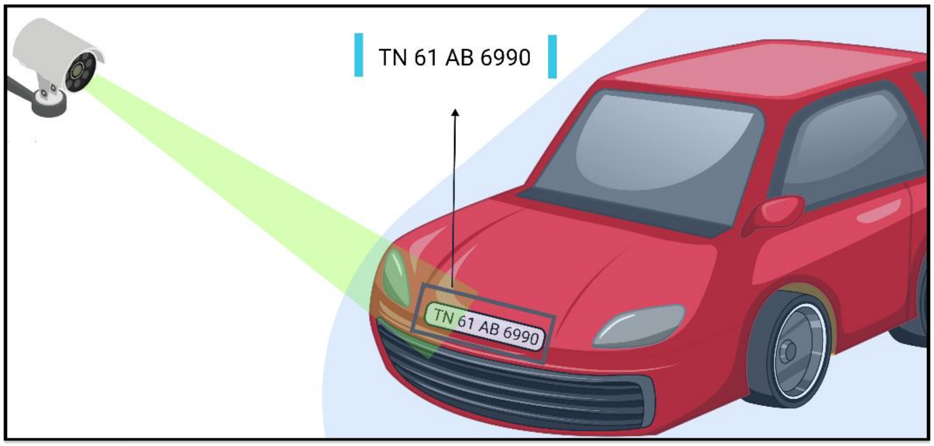
AUTOMATIC LICENSE PLATE RECOGNITION



GUIDED BY:
PROF.DR. VISWANATHAN P

Submitted By:
20BDS0350 R.SHAMINI

SCOPE

PAGE OF CONTENTS

CHAPTER 1

ABSTRACT

1.1. Abstract:

License Plate Detection plays a crucial role in road safety such as CCTV cameras with license plate detection system can recognise over speeding, car accidents etc.The project will basically detect the plates then make the recognition of the plate that is to extract the text from an image.

Technologies → OpenCV, Python, Keras, Skilearn
Dataset→ https://github.com/Shamini13/License_plate_recognition/blob/main/Dataset_lp.zip

Key words:ANPR, Computer Vision, Convolutional Neural network(CNN), Deep learning, Image Preprocessing, License Plate Text Detection ,Text Recognition

1.2. Introduction

The detection & recognition of license plates is a very good source of information for record detection and recognition.It plays a significant role in role safety as the CCTV cameras with license plate detection system can recognise vehicles engaged in over speeding, car accidents etc. and take action.

Automatic recognition of license plates is an important stage in the intelligent traffic network. And it takes a considerable amount of effort & time to extract license plate numbers from CCTV camera footage. Conventional method of manually going through footage takes a considerable amount of time, so our ANPR project will extract the license plate number from the video footage itself.

Our proposed method for recognizing license plates includes the three main steps. That is the region of extraction of interest, extraction of plate numbers and recognition of character. This is implemented via computer vision to extract region of interest and plate numbers and convoluted neural networks to train our model which recognize characters of number plates.

CHAPTER 2

OBJECTIVES

1. To show improvements in previous papers and projects.
2. To detect the license plate from the image
3. To recognize characters of license plate from images and videos.
4. To increase the accuracy of the model
5. To learn the working of various previous models

CHAPTER 3

LITERATURE REVIEW

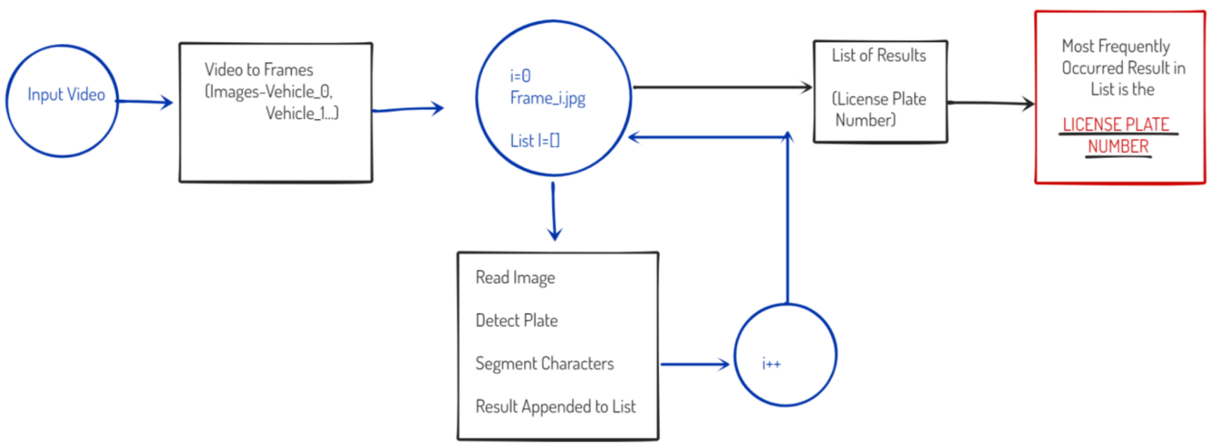
S.No	Title of the resource (journal paper/conference paper/ title of the web page)	Year	Journal name/ Conference title/ Website link	Methodology (Key algorithms / approach)	National status	Data set / Data used in the resource	Evaluation observations/ Comments
1	A Review Paper on License Recognition System	2020	A Review Paper on License Plate Recognition System iMedPub Journals	Describes a study of vehicle number plate identification in traffic surveillance	International	None	Concludes that use high-resolution cameras gives improved accuracy for recognition
2	License plate recognition:A Review with experiments for malaysia case study	2013	License Plate Recognition (LPR): A Review with Experiments for Malaysia Case Study	Image processing techniques	International	License Plate Image	Describes various image processing techniques for license plate recognition
3	Automatic Number Plate Recognition System	2017	Automatic Number Plate	Describes algorithm on how to get license text	National	Vehicle Image	Implemented using Matlab

			Recognition System	image from the vehicle image.			
4	License Plate Recognition From Still Images and Video Sequences: A Survey.	2008	License plate recognition from video	This recognizes license plate from images and video	International	Car images and video	This paper has implemented OCR to recognize the individual characters
5	Automatic License Plate Recognition System Using SSD And EasyOCR	2021	Automatic license plate recognition using SSD	Tesseract OCR Algorithm	International	Text Images	Explain how Tesseract OCR works
7	License plate detection using convolutional neural network	2017	License plate detection using CNN	CNN algorithm	International	License plate -Car	This method can detect various types of license plates with high accuracy and relatively short running time with the help of CNN algorithm.
8	Review paper on ANPR using machine learning algorithms	2019	ANPR using machine learning algorithms	Machine learning Algorithms	International	Number plate	Recognizes and differentiates between genuine and fake license plate
9	Accurate Detection and Recognition of Dirty Vehicle Plate Numbers for High Speed Applications	2017	Detection using SVM	Support vector machine	International	Cropped license plate	Plate detection using SVM
10	An efficient method to improve the accuracy of Vietnamese vehicle license plate recognition in unconstrained environment’	2021	Vietnamese lp recognition	OCR	International	Vietnamese License plate	Model that recognizes at unconstrained environment
11	Challenges in Automatic License Plate Recognition System	2021	Challenges faced in ANPR	Challenges faced in detection and segmentation	International	Indian License plate	This explains about the future work and development

CHAPTER 4

DESIGN

4.1. Proposed Architecture - Block diagram



CHAPTER 5

INVENTION DETAILS

5.1. Objective of the Invention

5.1.1 Character recognition is faster and accurate:

I have improved the rate of detecting the license plate and rate of recognizing the characters. Accuracy of recognition is comparatively greater than the referred papers.

5.1.2. Implementation of recognizing the car license plate from a video

This code can even detect the license plate of the car and can return its license plate number after processing.

5.1.3. Overcoming the drawbacks:

Irrespective of the position or color objective was to make it work for almost all the inputs.

CHAPTER 6

<u>WORKING SCENARIO</u>

6.1. CLASSIFICATION AND FEATURE EXTRACTION:

This basically differentiates Image containing license plate from other images. Using **cascade classifier** algorithm, we train using xml file and help it detect the license plate using **detectMultiscale** function.

```
#detecting license plate on the vehicle
plateCascade = cv2.CascadeClassifier('indian_license_plate.xml')
```

```
]: #test image is used for detecting plate
inputImg = cv2.imread('car.jpg')
inpImg, plate = plate_detect(inputImg)
display_img(inpImg)
```

6.2. PRE-PROCESSING:

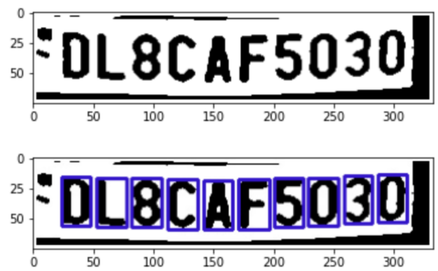
The cropped image is converted into grayscale using **cvtColor** function from cv2 library which takes the weighted average of all the pixels from the cropped image and returns an image with the averaged pixel.

```
grey=cv2.cvtColor(plate,cv2.COLOR_BGR2GRAY)
display_img(grey)
```

6.3. SEGMENTATION AND FEATURE EXTRACTION:

We use **thresholding segmentation** and convert the grayscale image to binary image. **Otsu threshold** determines the threshold value automatically by making assumptions using bimodal histogram and **threshold binary** sets the pixels which have less than threshold value to 0 and pixels greater than threshold value to 255.

```
char = segment_characters(plate)
```



6.4. MORPHOLOGICAL OPERATIONS:

6.4.1. EROSION: It removes pixels on object boundaries. Erosion shrinks the image.

6.4.2. DILATION: It adds pixels to the boundaries of objects in an image, dilation enlarges the image.

We find the **contours** using dimensions of the cropped image. Using the indexes of the contours, bounding box is applied and we get box around individual characters.

Subplot returns all the separate characters as single image.

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

6.5. CLASSIFICATION AND CHARACTER RECOGNITION:

We create and train our CNN model using our dataset to predict the numerical characters from the images which we have segmented from the above section. Sequential model has been used to add layers. This helps us in accurate prediction of the input image.

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

```
DL8CAF5030
```

6.6 VIDEO :

We have given the input video which returns frames(images). These images are run through a loop where it undergoes the above steps and returns a result which is appended to a list.

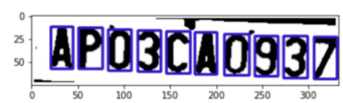
```
import math
count = 0
videoFile = "License.mp4"
cap = cv2.VideoCapture(videoFile)
frameRate = cap.get(5)
x=1
while(cap.isOpened()):
    frameId = cap.get(1)
    ret, frame = cap.read()
    if (ret != True):
        break
    if (frameId % math.floor(frameRate) == 0):
        filename ="Vehicle_%d.jpg" % count;
        count+=1
        cv2.imwrite(filename, frame)
cap.release()
print ("Done!")
```

```
Done!
```

```
img = plt.imread('Vehicle_7.jpg')
plt.imshow(img)
```

```
<matplotlib.image.AxesImage at 0x17e5e41ea00
```

```
l=[]
for i in range(0,count):
    try:
        frame="Vehicle_"+str(i)+".jpg"
        frame = plt.imread(frame)
        inpImg, plate = plate_detect(frame)
        char = segment_characters(plate)
        l.append(show_results())
    except:
        print("",end='')
```



This the most frequently occurring item in the list of results is outputted as the license plate number.

```
print(l)
```

```
['AP03CA0937', 'P0J3CA0D337', 'P03CA0L337', 'CACG1', 'AP03CA0337', 'AP03CA037', 'AP03CA0937']
```

```
print("License Plate is",max(set(l), key=l.count))
```

```
License Plate is AP03CA0937
```

CHAPTER-7

IMPLEMENTATION

7.1.Algorithm and code:

7.1.1.Algorithm:

- Implementation of Classifier that trains the model to differentiate between license plate from the other images

- Detects the plate and return car and plate image

- Preprocessing is performed for the cropped image

- Image is segmented into individual characters using thresholding segmentation and by finding contours

- Plate number is returned by the defined function known as show_results

- Training is done using CNN model

7.1.2 Code:

Link:

Executed code with output

https://github.com/Shamini13/License_plate_recognition/blob/main/License%20plate%20recogition.ipynb

7.2.SNAPSHOT:

LICENSE PLATE RECOGNITION

NAME: R. Shamini

REGN.NO: 20BDS0350

In [2]:
```
#import libraries
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

In [2]:
```
#detecting license plate on the vehicle
plateCascade = cv2.CascadeClassifier('indian_license_plate.xml')
```

In [3]:
```
#detect the plate and return car + plate image
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
```

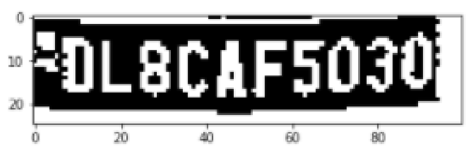
In [4]:
```
#normal function to display
def display_img(img):
    img_ = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
    plt.imshow(img_)
    plt.show()
```

In [6]:
```
#test image is used for detecting plate
inputImg = cv2.imread('car.jpg')
inpImg, plate = plate_detect(inputImg)
display_img(inpImg)
```

In [7]:
```
display_img(plate)
```

In [8]:
```
grey=cv2.cvtColor(plate,cv2.COLOR_BGR2GRAY)
display_img(grey)
```

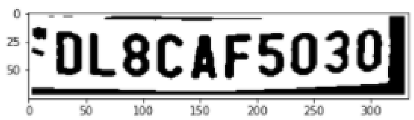
In [28]:
```
ret, bw_img = cv2.threshold(grey, 127, 255, cv2.THRESH_BINARY)
bw_img=255-bw_img
display_img(bw_img)
```

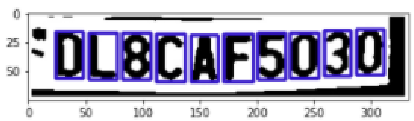


In [9]:
```
def find_contours(dimensions, img) :

    #finding all contours in the image using
    #retrieval mode: RETR_TREE
    #contour approximation method: CHAIN_APPROX_SIMPLE
    cntrs, _ = 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
    cntrs = sorted(cntrs, key=cv2.contourArea, reverse=True)[:15]

    ci = cv2.imread('contour.jpg')

    x_cntr_list = []
    target_contours = []
    img_res = []
    for cntr in cntrs :
        #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_cntr_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), (intWidth+intX, intY+intHeight), (50,21,200), 2)
            plt.imshow(ci, 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
            char_copy[:, 22:24] = 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_cntr_list)), key=lambda k: x_cntr_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
```

In [11]:
```
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)

    #getting contours
    char_list = find_contours(dimensions, img_binary_lp)

    return char_list
```

In [12]:
```
char = segment_characters(plate)
```





In [13]:
```python
for i in range(10):
    plt.subplot(1, 10, i+1)
    plt.imshow(char[i], cmap='gray')
    plt.axis('off')
```

In [14]:
```python
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 = 'data/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.
```

In [15]:
```python
#It is the harmonic mean of precision and recall
#Output range is [0, 1]
#Works for both multi-class and multi-label classification
def f1score(y, y_pred):
    return f1_score(y, tf.math.argmax(y_pred, axis=1), average='micro')

def custom_f1score(y, y_pred):
    return tf.py_function(f1score, (y, y_pred), tf.double)
```

In [16]:
```python
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(learning_rate=0.0001), metrics=[custom_f1score])
```

In [17]:
```python
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
```

In [20]:
```python

batch_size = 1

history= model.fit(
      train_generator,
      steps_per_epoch = train_generator.samples // batch_size,
      validation_data = validation_generator,
      epochs = 50)
```

```
Epoch 1/50
864/864 [==============================] - 59s 68ms/step - loss: 1.4542 - custom_f1score: 0.5868 - val_loss: 0.7490 - val_c
ustom_f1score: 0.7842
Epoch 2/50
864/864 [==============================] - 62s 72ms/step - loss: 0.6855 - custom_f1score: 0.7778 - val_loss: 0.4581 - val_c
ustom_f1score: 0.8304
Epoch 3/50
864/864 [==============================] - 66s 76ms/step - loss: 0.4386 - custom_f1score: 0.8611 - val_loss: 0.2608 - val_c
ustom_f1score: 0.9241
Epoch 4/50
864/864 [==============================] - 75s 87ms/step - loss: 0.3393 - custom_f1score: 0.8935 - val_loss: 0.2724 - val_c
ustom_f1score: 0.9196
Epoch 5/50
864/864 [==============================] - 65s 75ms/step - loss: 0.2843 - custom_f1score: 0.9109 - val_loss: 0.2237 - val_c
ustom_f1score: 0.9509
Epoch 6/50
864/864 [==============================] - 64s 74ms/step - loss: 0.1995 - custom_f1score: 0.9340 - val_loss: 0.1948 - val_c
ustom_f1score: 0.9211
Epoch 7/50
```

In [21]:
```
#model.save('ANPR_model_FINAL3.pkl')
```

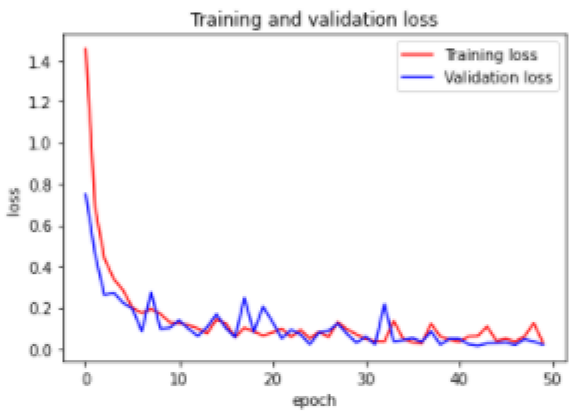
```
INFO:tensorflow:Assets written to: ANPR_model_FINAL3.pkl\assets
```

In [22]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        23248
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        131104
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        131136
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 64)        65600
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 7, 7, 64)          0
_________________________________________________________________
dropout (Dropout)            (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 128)               401536
_________________________________________________________________
dense_1 (Dense)              (None, 36)                4644
=================================================================
Total params: 757,268
Trainable params: 757,268
Non-trainable params: 0
_________________________________________________________________
```

In [25]:
```
#Graphing our training and validation
accuracy = history.history['custom_f1score']
val_accuracy = history.history['val_custom_f1score']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'r', label='Training acc')
plt.plot(epochs, val_accuracy, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'r', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend()
plt.show()
```


Training and validation accuracy
accuracy
epoch
Training acc
Validation acc


Training and validation loss
loss
epoch
Training loss
Validation loss

In [16]:
```
from keras.models import load_model
# Restore the weights
model.load_weights('ANPR_model_FINAL3.pkl')
```

Out[16]: <tensorflow.python.training.tracking.util.CheckpointLoadStatus at 0x1dde0eb5880>

In [26]:
```
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)
       # predict_x=model.predict(img)
        #classes_x=np.argmax(predict_x,axis=1)
        y_ = model.predict_classes(img)[0]
        character = dic[y_] #
        output.append(character)

    plate_number = ''.join(output)

    return plate_number
    character = dic[predict_x]
    output.append(character)

    plate_number = ''.join(output)

    return plate_number

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

```
C:\Users\joshi\anaconda3\lib\site-packages\tensorflow\python\keras\engine\sequential.py:455: UserWarning: `model.predict_classe
s()` is deprecated and will be removed after 2021-01-01. Please use instead:* `np.argmax(model.predict(x), axis=-1)`,   if your
model does multi-class classification   (e.g. if it uses a `softmax` last-layer activation).* `(model.predict(x) > 0.5).astype
("int32")`,   if your model does binary classification   (e.g. if it uses a `sigmoid` last-layer activation).
  warnings.warn('`model.predict_classes()` is deprecated and '
```

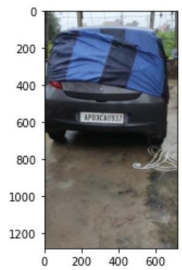
DL8CAF5030

n [17]:
```
import math
count = 0
videoFile = "License.mp4"
cap = cv2.VideoCapture(videoFile)
frameRate = cap.get(5)
x=1
while(cap.isOpened()):
    frameId = cap.get(1)
    ret, frame = cap.read()
    if (ret != True):
        break
    if (frameId % math.floor(frameRate) == 0):
        filename ="Vehicle_%d.jpg" % count;
        count+=1
        cv2.imwrite(filename, frame)
cap.release()
print ("Done!")
```

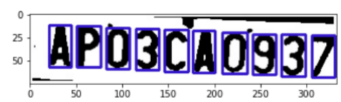
Done!

```
]: img = plt.imread('Vehicle_7.jpg')
   plt.imshow(img)
```

]: <matplotlib.image.AxesImage at 0x17e5e41ea00>



```
l=[]
for i in range(0,count):
    try:
        frame="Vehicle_"+str(i)+".jpg"
        frame = plt.imread(frame)
        inpImg, plate = plate_detect(frame)
        char = segment_characters(plate)
        l.append(show_results())
    except:
        print("",end='')
```



CHAPTER 8:

RESULT & ANALYSIS

We have given the input video which returns frames(images). These images are run through a loop where it undergoes:

1. Detection of Region of License Plate
2. Extraction of License Plate Region
3. Character Segmentation
4. Detecting the Characters with CNN
5. Appending the Results to List

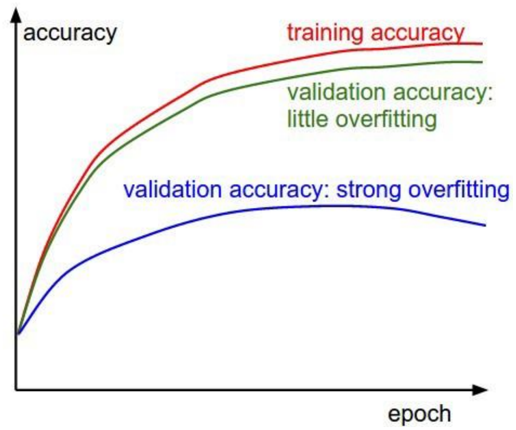
This the most frequently occurring item in the list of results is outputted as the license plate number.

The Convolutional Neural network is trained with an accuracy(F1 Score) of 99.31%. The F1 score can be interpreted as a harmonic mean of precision and recall, where an F1 score reaches its best value at 1 and worst score at 0.

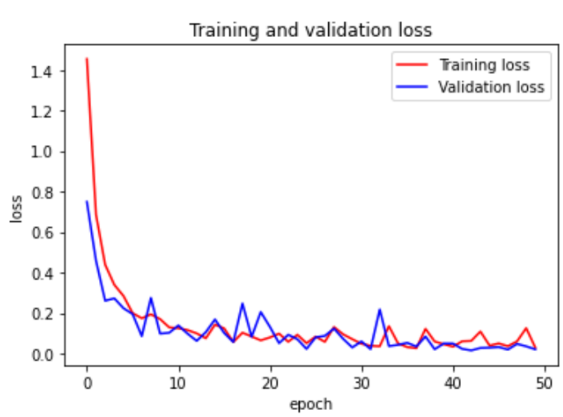
This CNN model predicts the license plate characters. It has the following architecture:

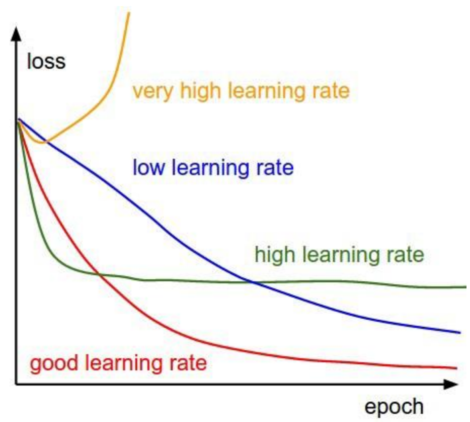
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 16)        23248
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 32)        131104
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 64)        131136
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 64)        65600
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 7, 7, 64)          0
_________________________________________________________________
dropout (Dropout)            (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 128)               401536
_________________________________________________________________
dense_1 (Dense)              (None, 36)                4644
=================================================================
Total params: 757,268
Trainable params: 757,268
Non-trainable params: 0
_________________________________________________________________
```

Graph between Epoch v/s Training & Validation Accuracy(F1 Score)



Graph between Epoch v/s Training & Validation Accuracy(F1 Score)





As we can see from the graphs, the CNN model has a good/high learning rate and has little overfitting.

CHAPTER 9

CONCLUSION

There has been a lot of research done on detection and recognition of license plates. Automatic recognition of license plates is an important stage in the intelligent traffic network. Considering the significance of ANPR, especially in road safety, we have implemented our own version of it. Our ANPR project will extract the license plate number from the video footage itself. Recognition combined with neural networks identified the number plates of different views from videos . We have achieved our aim of detecting and recognizing the license plate from a video and it is possible to implement the same for CCTV recording too!

CHAPTER 10

REFERENCES

10.1.WEBSITES:

https://towardsdatascience.com/useful-plots-to-diagnose-your-neural-network-521907fa2f45

https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

https://www.pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/

https://projectgurukul.org/license-number-plate-recognition-extraction/

https://automaticaddison.com/how-to-detect-and-draw-contours-in-images-using-opencv/

https://towardsdatascience.com/training-a-convolutional-neural-network-from-scratch-2235c2a25754

10.2.RESEARCH PAPERS:

C. E. Anagnostopoulos, I. E. Anagnostopoulos, I. D. Psoroulas, V. Loumos and E. Kayafas, "License Plate Recognition From Still Images and Video Sequences: A Survey," in IEEE Transactions on Intelligent Transportation Systems, vol. 9, no. 3, pp. 377-391, Sept. 2008, doi: 10.1109/TITS.2008.922938.

iss. Shraddha S. Ghadage Mr. Sagar R. Khedkar A Review Paper on Automatic Number Plate Recognition System using Machine Learning Algorithms International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 IJERTV8IS120398 (This work is licensed under a Creative Commons Attribution 4.0 International License.) Published by : www.ijert.org Vol. 8 Issue 12, December-2019

Li, H., Yang, R. and Chen, X. (2017) 'License plate detection using convolutional neural network', 2017 3rd IEEE International Conference on Computer and Communications (ICCC), Computer and Communications (ICCC), 2017 3rd IEEE International Conference on, pp. 1736–1740.

. Panahi and I. Gholampour, "Accurate Detection and Recognition of Dirty Vehicle Plate Numbers for HighSpeed Applications," in IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 4, pp. 767-779, April 2017, doi: 10.1109/TITS.2016.2586520.

AWALGAONKAR, N.; BARTAKKE, P.; CHAUGULE, R. Automatic License Plate Recognition System Using SSD. 2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA), Intelligent Robotics and Industrial Automation (IRIA),2021 International Symposium of Asian Control Association on, [s. l.], p. 394–399, 2021.

Quoc, K. N., Van, D. P. and Bich, V. P. T. (2021) 'An efficient method to improve the accuracy of Vietnamese vehicle license plate recognition in unconstrained environment', 2021 International Conference on Multimedia Analysis and Pattern Recognition (MAPR), Multimedia Analysis and Pattern Recognition (MAPR), 2021 International Conference on, pp. 1–6.

Mukhija, P., Dahiya, P. K. and Priyanka (2021) 'Challenges in Automatic License Plate Recognition System: An Indian Scenario', 2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT), Computational Intelligence and Communication Technologies (CCICT), 2021 Fourth International Conference on, CCICT, pp. 255–259. doi: 10.1109/CCICT53244.2021.00055.