JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

PROJECT REPORT

AUTOMATED NUMBER PLATE RECOGNITION

Under the guidance of:

Dr. Ankit Vidyarthi

Submitted By:
Archit Agarwal (16103157 - B9)
Ishita Kalsi (16103064 – B10)
Samidha Verma (16103199 – B9)
Harshvardhan Pratap Singh (16103090 – B9)

ABSTRACT

ANPR is an image-processing innovation which is used to perceive vehicles by their tags. This expertise can be used in security and traffic installation. Tag Recognition System is an application of PC vision. PC vision is a technique for using a PC to take out abnormal state information from a digital image. The useless homogeny among various tags for example, its dimension and the outline of the License Plate. The ANPR system consists of following steps:-

i. Vehicle image capture.

ii. Preprocessing.

iii. Number plate extraction.

iv. Character segmentation.

v. Character recognition.

The ANPR system works in these strides, the initial step is the location of the vehicle and capturing a vehicle image of front or back perspective of the vehicle, the second step is the localization of Number Plate and then extraction of vehicle Number Plate is an image. The final stride use image segmentation strategy, for the segmentation a few techniques neural network, mathematical morphology, color analysis and histogram analysis. Segmentation is for individual character recognition. Optical Character Recognition (OCR) is one of the strategies to perceive the every character with the assistance of database stored for separate alphanumeric character.

OBJECTIVE

The objective of this project is **to detect number plate of a vehicle from real time images**. Basically, we plan to use ANPR to build an application which detects what is written on the number plate. This system works by capturing image of the car, detecting the car, finding the number plate of the car and extracting the license number from that region.

NOVEL USES OF ANPR:

1) Can be used at entrance for security control of a highly restricted area like military zones or area around top government offices e.g. Parliament, Supreme Court etc.
2) For automated toll booth tax collection.
3) To catch traffic signal breakers.

BACKGROUND STUDY AND FINDINGS

1) Soomro, Shoaib Rehman, Mohammad Arslan Javed, and Fahad Ahmed Memon. "Vehicle number recognition system for automatic toll tax collection." *2012 International Conference of Robotics and Artificial Intelligence*. IEEE, 2012.

The objective of this research paper was to design a number plate recognition system and to implement it for automatic toll tax collection.

Above research paper discussed the following steps :
1 . Detecting vehicle and capturing image of front view of vehicle.
2. Extract number plate and recognize the characters.
3. Template matching to convert characters of pixels to alphanumeric values
4. Using the above result charge toll tax accordingly.
Limitations:
Technology used in the above research paper was old and not efficient as it cannot detect the moving targets only works on still objects.
Template matching is used for character recognition which makes it difficult to differentiate between the characters like B and 8 or O & 0.

2)https://www.springer.com/cda/content/document/cda_downloaddocument/9789811016776-c2.pdf?SGWID=0-0-45-1586559-p180037601

In this paper, morphological operation-based approach is presented for number plate area extraction. Effective segmentation of characters is done after plate area extraction. Histogram-based character segmentation is a simple and efficient technique used for segmentation. Template matching approach is used for character extraction.
Algorithm proposed in this paper identifies characters present on single line number plate. This research paper has not implemented the

model on video surveillance and hence it can detect only steady object not moving targets.

- 3) Shevale, Ketan S. "Automated License Plate Recognition for Toll Booth Application." *Int. Journal of Engineering Research and Applications* 4.10: 72-76.

  In this paper, there are numerous steps followed to detect the license plate of a car:

  - Input image: Image of the car is captured when it stops.
  - Pre-processing of image: Converting image to gray-scale.
  - Number-plate localization: Template-based matching and convolution is used to detect number plates. This technique can be easily performed on grey images or edge images.
  - Character Segmentation and recognition: Template matching is used to recognize characters after segmentation of characters.

  It has some limitations:

  - It is only has database of 3 to 4 different types of fonts of Indian number plates.
  - It has not been implemented for video surveillance and so it cannot work on video targets without modification.

- 4) Balaji, G. Naveen, and D. Rajesh. "Smart Vehicle Number Plate Detection System for Different Countries Using an Improved Segmentation Method." *Imperial Journal of Interdisciplinary Research (IJIR) Vol* 3 (2017): 263-268.

  In this paper, first various vehicle images have been captured through camera. After that the coloured image is converted to gray scale image, brightness adjustment, contrast up to optimum values and removing noise using median filtering is done in order to get better quality image i.e. image pre-processing is done. Exact location of

license plate region is masked and extracted from image. Characters are segmented on the extracted region. After segmentation, character recognition is done by the help of template matching. If characters are matched then an output text is displayed.

Limitations: The model is used for images and not implemented for video surveillance.

5) https://www.irjet.net/archives/V3/i1/IRJET-V3I1191.pdf

International Research Journal of Engineering and Technology (IRJET) by Prof.PradnyaRandive, Shruti Ahivale, Sonam Bansod, Sonal Mohite, Sneha Patil.

In this research paper the approach followed is:

ALPR is the technique of extracting the information from an image or a sequence of images of Vehicle's Number Plate. The image will be captured from Android device. This captured device will be then sent to the System. System will perform pre-processing on this on this image using algorithm image Segmentation. In the next step feature extraction will take place and after that matching of template will be done using the algorithm Bozorth3. The result will be sent from the server to the android device to determine the correct authority of the person.

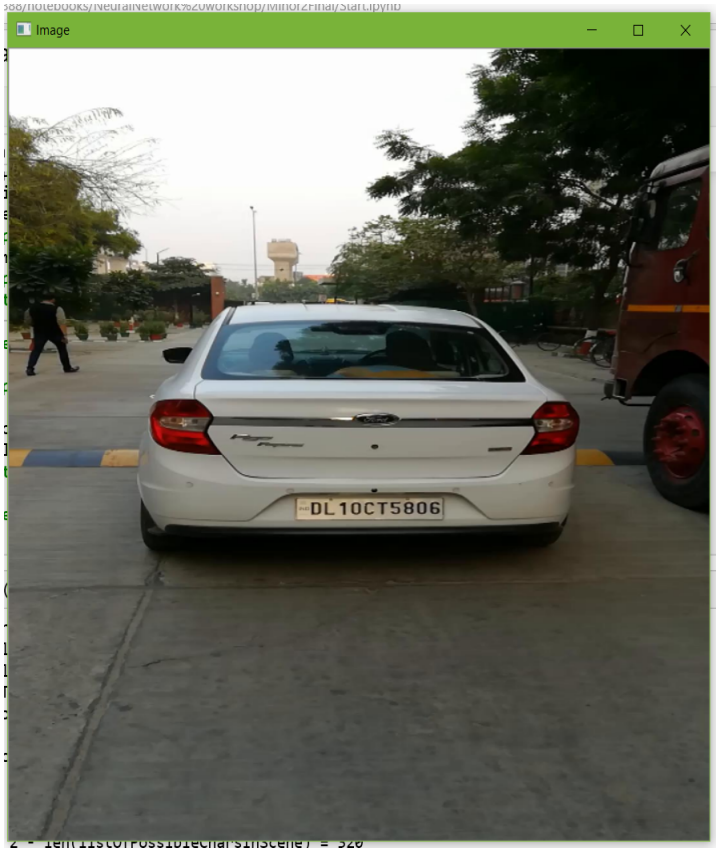
DESIGNING DETAILS AND IMPLEMENTATION:

The following are the design details and the methodology used in the project:

- In this project we start with the **start.py** file that asks to input the name of the video file for which we want the prediction.
- This then breaks the video into frames and stores them in a folder named '**data**'.
- Then we send each frame to the **main.py** which will return the predicted image and the cropped license plate from the frame.

1) Loading the Image :

- We have displayed a test run on a car image shown below



2) Preprocessing the input image :

- This step in done in the **Preprocess.py** file. As most of the opencv function require the input image to be in the grayscale and greyscale images are easy for computation so we convert the input image to greyscale.

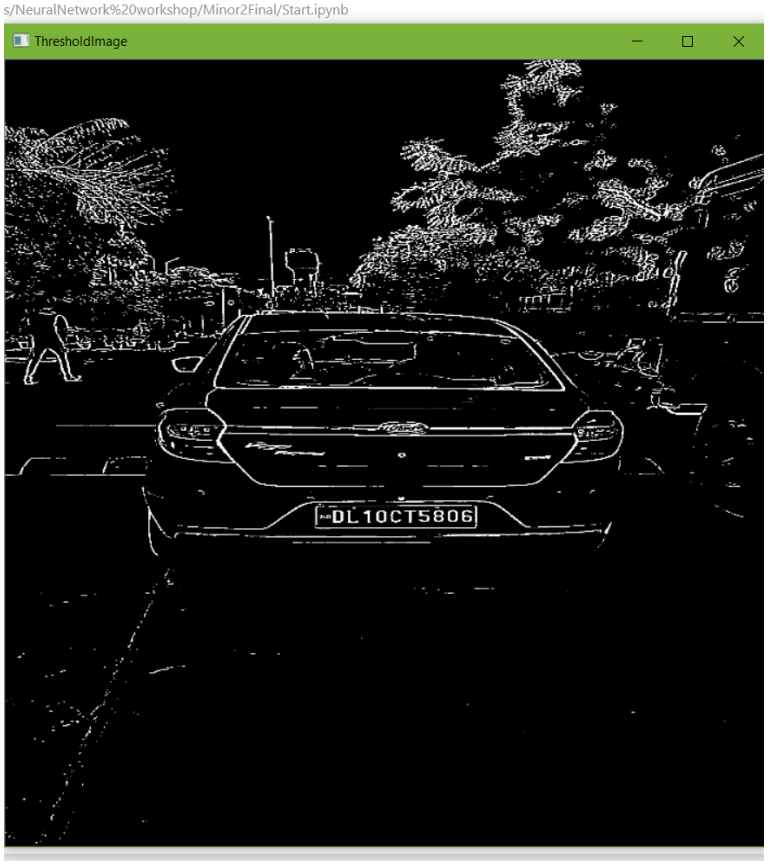
- Now the image may of any quality but to extract the number plate we need it to be clear. So to do so we enhance the features of the image by maximizing the contrast of the image that is by making each region more clear or differientiable from the others.

- To increase the contrast of the image we :
    1. TopHat Morphology transformation :- Difference of input image and Opening of the image.
    2. BlackHat Morphology transformation :- Difference of closing of the input image and input image.'

3. GrayscalePlusTopHat = Grayscale + TopHat

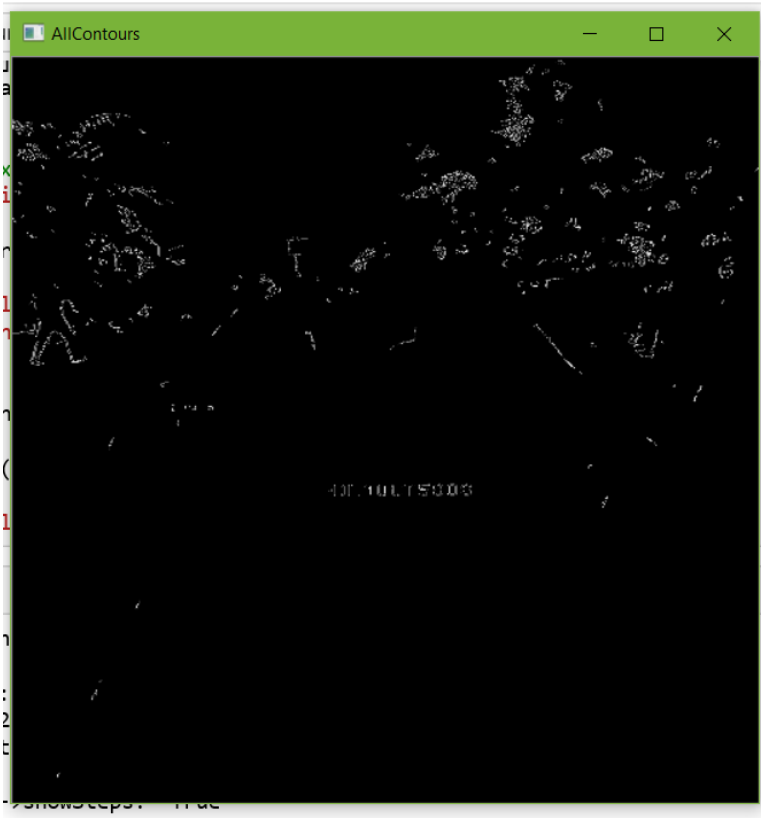
- Then we do blurring to bring a uniformity to the image. It is also a method to remove noise and a recommended step before thresholding.
- Finally we apply adaptive thresholding to the image using the BINARY_INV function, window size as 19 and combination weight as 9.

The threshold image



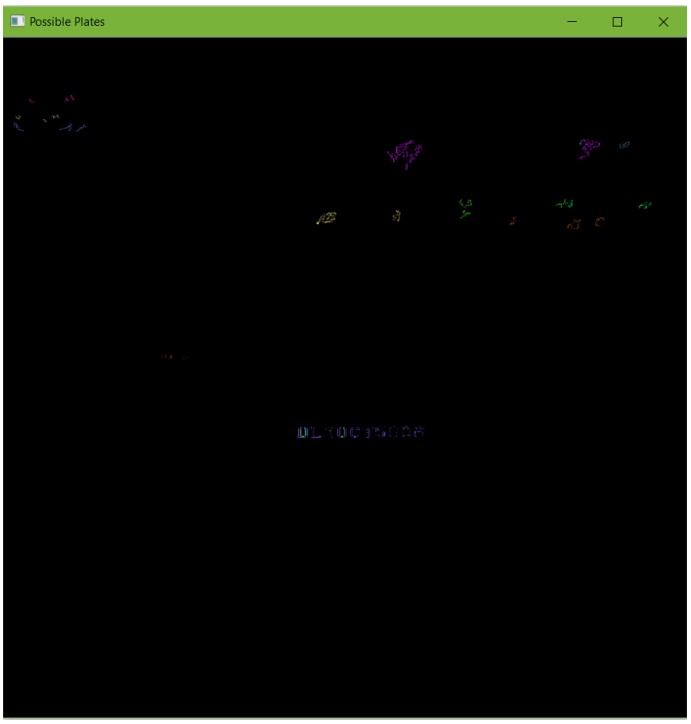
3) Plate detection:

- Now we have a grayscale image and a thresholded image. We send them to the **DetectPlates.py** file. Next we apply the findcontour function of opencv to detection all the boundaries in the thresholded image. This step is to extract the characters from the image so that they can be recognized.



Total number of contours found in image are 1367

- Then we filter the full list of the countours to check remove those countours those cannot be characters. As the government has specified some specific range of dimensions, padding and texture for the name plate and its characters so we check the height, width, area and aspect ratio of each countour and include only those those which fit in a particular range
    - Now, number of contours that may be characters = 34

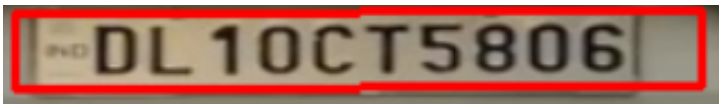


- After this we have a large list of countours. Next we re-group the big list of countors in sub-groups(we try to make a complete plate from the individual characters). We group these countours using thier distance from each other, their individual size and the combine dimensions of the rectangle formed. Next we remove overlapping structures from the plates

- Now we make python class object of each individual countour of the list that store the boundaries, area, aspect ratio, width, heigth and angle from the 0 degree. Then we return a list of these possible plates

4) Character Segmentation:

- This phase is done in the **DetectChar.py** file.
- We start with each possible plates and then we crop the plate from input image and resize it to 1.6 times height, 1.6 times width. Then we again apply the preprocessing operations on the plate image.



- We get the grayscale and thresholded image of the plate.

THRESHOLDED

- We again apply binary thresholding on the image to remove any noise in the image,resize it and get a plane image with white character on black background.

- We apply countouring on the plate, remove non characters from the list.

- Regroup the countours in sub-lists

- Remove overlapping countours.

5) Vehicle recognition :

- This part is done in the **VehicleIdentification.ipynb** file.
- For the purpose of recognition of the vehicle we use a Trained Convolutional Neural Network classifier.

  ➔ The classifier is built using the python keras module.

  ➔ The classifier is trained with 15003 images constituting images 3 classes car, bus and truck.

  ➔ The classifier is tested with 4503 images constituting images 3 classes car, bus and truck.

  ➔ It takes input 32x32 input image in the first convolution layer.

  ➔ The **relu** activation function in the convolution layer.

  - The **softmax** function is used in the final prediction layer.
  - The classifier uses the sgd optimizer with the starting learning rate as 0.0008.
  - We trained the classifier for 8 epochs.

EPOCH	ACCURACY	LOSS
1.	0.5778	1.3687
2.	0.6858	0.7454
3.	0.7234	0.6857
4.	0.7578	0.5761
5.	0.7795	0.5326
6.	0.7720	0.5471
7.	0.8014	0.4882
8.	0.7962	0.5344

The Detailed summary of the basic structure of the Models is:

Layer (type)	Output State	Param#
conv2d	(None ,28,28,32)	1664
max_pooling2d	(None,14,14,32)	0
conv2d	(None,10,10,128)	204928
flatten	(None,12800)	0
dense	(None ,256)	3277056
dropout	(None ,256)	0
dense	(None ,3)	771

- Total params: 34,84,419
- Trainable params: 34,84,419
- Non-trainable params: 0

6) Character recognition :

- This part is done in the **train_detect.py** file.

- For the purpose of recognition of the cropped character we use a Trained Convolutional Neural Network classifier.

  ➔ The classifier is built using the python keras module.

  ➔ The classifier is trained with 47605 images constituting images 36 classes ranging from 0-9 and A-Z.

  ➔ The classifier is tested with 1292 images constituting images 36 classes ranging from 0-9 and A-Z

  ➔ It takes input 64x64 input image in the first convolution layer.

  ➔ The **relu** activation function in the convolution layer.

  - The softmax function is used in the final prediction layer.
  - The classifier uses the Root Mean square optimizer with the starting learning rate as 0.001 which gradually decreases by 0.005 after each epoch.
  - We trained the classifier for 5 epochs.

EPOCH	**ACCURACY**	**LOSS**
1.	0.9001	0.3825
2.	0.924	0.285
3.	0.929	0.265
4.	0.932	0.251
5.	0.934	0.246

The Detailed summary of the basic structure of the Models is:

Layer (type)	**Output Shape**	**Param#**
conv2d	(None, 62, 62, 32)	896
max_pooling2d	(None, 31, 31, 32)	0
conv2d	(None, 29, 29, 32)	9248
max_pooling2d	(None, 14, 14, 32)	0
flatten	(None, 6272)	0
dense	(None, 128)	802944
dropout	(None, 128)	0
Dense	(None, 36)	4644

- Total params: 817,732
- Trainable params: 817,732
- Non-trainable params: 0

#Detailed Working of CNN Classifier

- Our keras classifier at the time of training requires 2 sets :

1) Training set

2) Test set

- The classifier is trained on all the images of the training set in 1 epoch and after that the classifier is applied on the test set to check its performance on the test set.
- In one epoch we see the changes in accuracy and loss of the training set and at the end of the epoch we get the accuracy and loss of the classifier on the test set.
- Let's understand the meaning of the loss obtained:
    - ✓ A loss function (or objective function, or optimization score function) is one of the two parameters required to compile a model.
    - ✓ You can either pass the name of an existing loss function, or pass a TensorFlow/Theano symbolic function that returns a scalar for each data-point and takes the following two arguments:
    - ✓ **y_true**: True labels. TensorFlow/Theano tensor.
    - ✓ **y_pred**: Predictions. TensorFlow/Theano tensor of the same shape as y_true.

- For this project we uses the **categorical_crossentropy** loss function.
- The method in which the loss function works depends on the activation function of the final layer. In our case we have use the **softmax** function.

- The final layer of the NN will return the probability of the image being of a particular class, this function will scale all the probabilities to sum up to 1. Its formula is:

  f (z(j)) = (e^z(j))/(sigma(k)(e^k))

- To properly understand how these function operate on the training set and test set we need to know the meaning of all the parameters of the **model.compile()** function.
- These functions are included in the engine class of the keras

  Model.compile(optimizer,loss,metrics)

  Only these are the important parameters.

✓ **Optimizers :-** It is the function that is used to improve the performance of the model.

✓ **Loss :-** It is used to calculate how well the model performs on the sets.

✓ **Metrics :-** List of all the metric on which the model the is evaluated during training and ttesting.

- This function initilizes all the parameters and builds connections between all the layers.
- Then we call the .fit function of the model.Sequential class to start the process of training.
- **.fit(x,y,batch_size,epochs,steps_per_epoch,validation_splilt,validation_steps)**
  - ✓ x = Numpy array of training data
  - ✓ y = Numpy array of target data
  - ✓ epochs = Number of epoch the model has to trainied.
  - ✓ steps_per_epoch = Number of training examples
  - ✓ validation_steps = Number of testing examples

- ✓ batch_size = Size of the batch in which the gradient descent is run.
- ✓ validation_split = split the validation data in to parts to evaluate.

- This function starts the training:

✓ Validates the user data (are these that same as specified in the compile) and are they in accordance with the layers parameters.

✓ Validate the test data --> Checks the format of the test data.

✓ It then splits the dataset specified.

✓ sets the initial state of the weights

✓ After this the control is transfered to **.fit_loop** function.

✓ This function then makes keras object for all the traing examples, initialize there weights and metrics values.z

✓ Then then function makes an object of the **backend.function** class.

✓ Then we transfer each training object to the backend.function. This function is the real training funcrion.

- We have used **tensorflow backend** so we will use the function of tensorflow class.

✓ This will make placeholders for the input array.

✓ Evaluate the predicted output with the real output

✓ This will give the error.

✓ The error is calculated using the loss function in the **compile()**, in our case we have used the **categorical_crossentropy**.

✓ For the optimization of the model we need the **optimizer** function. We have use the Root Mean Square function.

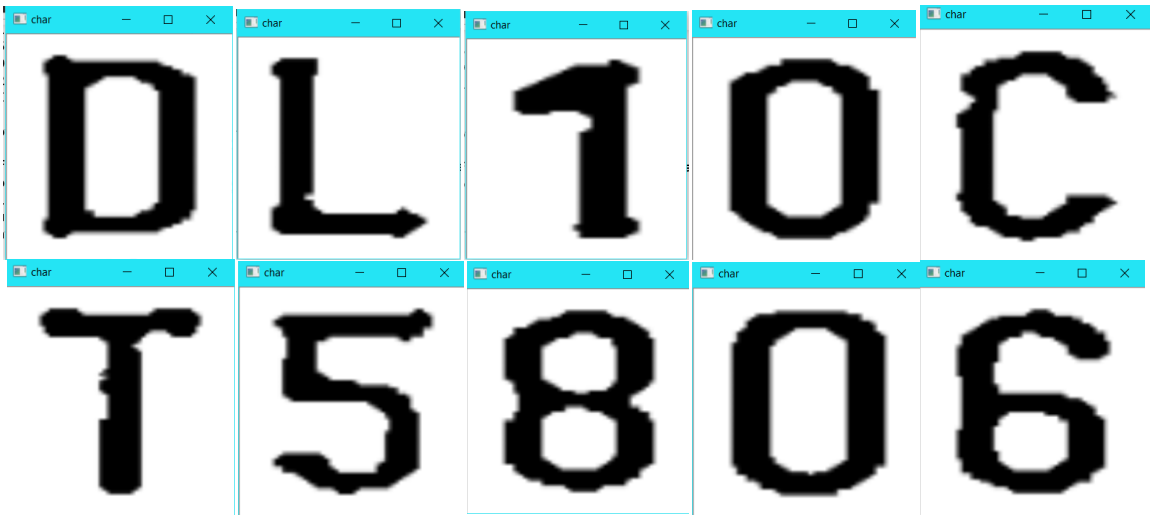
✓ The error for the whole batch is computing by applying the optimizer function on each example in a batch and the gradient is calculated.

✓ This gradient is then used to calculate the correction parameter.

- ✓ The correction parameter is computed for each neuron of each layer.
- ✓ Then the weight for each neuron is updated using the correction parameter.
- ✓ This process is carried out for the whole network, by back propagating the error computed and updating the weights.
- ✓ This process is carried out for each batch.
- ✓ After this the training process is over.
- ✓ 13) The **backend.validate()** function is called to operate on the test set.

- Now we send each list of coutours where each can be a plate in itself to the character recognition function.We are in the **DetectChar.recognizeChar** function.

- Here we re-threshold each plate with a combination of BINARY_INV + OTSU to get a clean black characters on white backgroud.
- We then sort the list by the x coordinate of the center of each of the members countours.
- We then cropped each character from the plate image, add a white border of 8px in all the directions to it, resize it to (64x64) and send it to the CNN classifier to get the predicted character from 36 classes.



FINAL RESULTS

- **On testing with video file we had the following results:**

```
[*]: call()

Enter the name of the video: car6.mp4
TotalFramesInVideo:  126
TotalFrameGenerated:  10
TimeTaken =  4.490557670593262
Do you want to see the Intermediate images: no
False

license plate read from  frame1.jpg  : DL10CT5806

----------------------------------------
Do you want to see the Intermediate images: no
False

license plate read from  frame10.jpg  : DL1OD

----------------------------------------
Do you want to see the Intermediate images: no
False

license plate read from  frame2.jpg  : BL10CT580S

----------------------------------------
Do you want to see the Intermediate images: no
False

license plate read from  frame3.jpg  : DL10CT5806

----------------------------------------
Do you want to see the Intermediate images: no
False

license plate read from  frame4.jpg  : DL10CTS806

----------------------------------------
Do you want to see the Intermediate images: no
False

license plate read from  frame5.jpg  : DL10CT5806

----------------------------------------
Do you want to see the Intermediate images: no
False

license plate read from  frame6.jpg  : DL1OCTS806

----------------------------------------
Do you want to see the Intermediate images: no
False

license plate read from  frame7.jpg  : DL1QCTS8O6

----------------------------------------
Do you want to see the Intermediate images: no
False

license plate read from  frame8.jpg  : DL1OCT5QO6

----------------------------------------
Do you want to see the Intermediate images: no
False

license plate read from  frame9.jpg  : L1OGC5

----------------------------------------
The name plate is : DL10CT5806  frequency is:  3
Vehicle is  car
execution time is : 123.27
```

REFERENCES

- Soomro, Shoaib Rehman, Mohammad Arslan Javed, and Fahad Ahmed Memon. "Vehicle number recognition system for automatic toll tax collection." *2012 International Conference of Robotics and Artificial Intelligence*. IEEE, 2012.

- Shevale, Ketan S. "Automated License Plate Recognition for Toll Booth Application." *Int. Journal of Engineering Research and Applications* 4.10: 72-76.
- Balaji, G. Naveen, and D. Rajesh. "Smart Vehicle Number Plate Detection System for Different Countries Using an Improved Segmentation Method." *Imperial Journal of Interdisciplinary Research (IJIR) Vol* 3 (2017): 263-268.
- https://www.irjet.net/archives/V3/i1/IRJET-V3I1191.pdf
- https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html
- https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html
- https://medium.com/m/global-identity?redirectUrl=https%3A%2F%2Ftowardsdatascience.com%2Fa-gentle-introduction-to-ocr-ee1469a201aa
- https://www.learnopencv.com/deep-learning-based-text-recognition-ocr-using-tesseract-and-opencv/
- https://machinelearningmastery.com/image-augmentation-deep-learning-keras/
- https://medium.com/@arindambaidya168/https-medium-com-arindambaidya168-using-keras-imagedatagenerator-b94a87cdefad
- https://ieeexplore.ieee.org/document/5169511
- https://en.wikipedia.org/wiki/Automatic_number-plate_recognition
- https://blog.devcenter.co/developing-a-license-plate-recognition-system-with-machine-learning-in-python-787833569ccd
- https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa