License Plate Character Recognition using Neural Networks

Aditya Sharma 170050043 Suraj 170050044 Debabrata 170050073 Rohan Abhishek 170050078

Abstract

Vehicles are widely used in our daily lives. The most efficient method to identify a vehicle is via its License Plate(LP) number, which is unique for each vehicle.The detection and recognition of license plates becomes difficult under different lighting conditions and orientations. Thus we have divided our approach into three parts :- License plate detection, character segmentation and character recognition. Below, we have provided in detailed descriptions of each sub-parts. We have two datasets, one made by us for training purposes for character recognition, and other one for testing our overall LPR algorithm. The dataset used for testing contains images under image perspective distortions, complex environments but have the license plate in a limited range of size(essential for LP detection). We have used some image processing techniques to tackle the different scenarios as mentioned above on the input image before we do the detection of plates.

Applications and Motivation

With the rapidly increasing number of vehicles, traffic violations are also increasing in public traffic, such as parking, speeding, theft of cars, etc. This makes the automatic LP detection and recognition crucial in the system of LP extraction. Our aim is to dive into the various techniques available and compare their performances on different scenarios and try to build a robust ALPR system. Although , our model might fail under certain test scenarios we are able to make it work for most of the cases possible.

Technique

The system is divided into three sub-categories:-

1. LP detection - Detecting the rectangular licence plate region.
2. Character Segmentation - Segmenting the licence plate into different characters
3. Character Recognition - Identifying different characters present

License Plate Detection

This stage outputs the bounding box coordinates of the plate. For this, first we enhanced the image quality. Then we converted the image to binary using a threshold. The value of threshold is obtained using the function `threshold_otsu()`. This function uses `Otsu`'s method for the threshold which is determined by minimizing intra-class intensity variance, or equivalently, by maximizing inter-class variance.

We have used `cv2`, `skimage`libraries in `python3` for detection of license plates that are of rectangular boxes which have height, width in a particular range as compared to the whole image.

This step may result in selecting more than 1 license plates. All the rectangular regions selected are shown as red rectangular regions. After LP detection, the cropped regions are used for the second step to extract the characters present.

Character Segmentation

For all the rectangular regions detected above, we performed segmentation to extract all the characters present in that region. This also involves constraints on the size of characters as compared to that of the region.

This again involves converting image to binary and the threshold is calculated with respect to the plate (and not the previous threshold for the whole picture) for better results.

We used `scikit` function `regions.bbox` which detects the regions. The parameters were optimized after training on different images.

Character Recognition

After extracting characters from the previous step, we now need to recognize the characters.

For this, we resized the characters to `20 x 20` which is the input for the neural network. We trained a neural network consisting of 2 fully connected layers. The layers were trained using character dataset(mentioned in Datasets section) and the model is saved as `my-model.npy`. This model is used to recognize the characters extracted and a threshold value(= 0.5) for prediction probability is used to remove the non-characters.

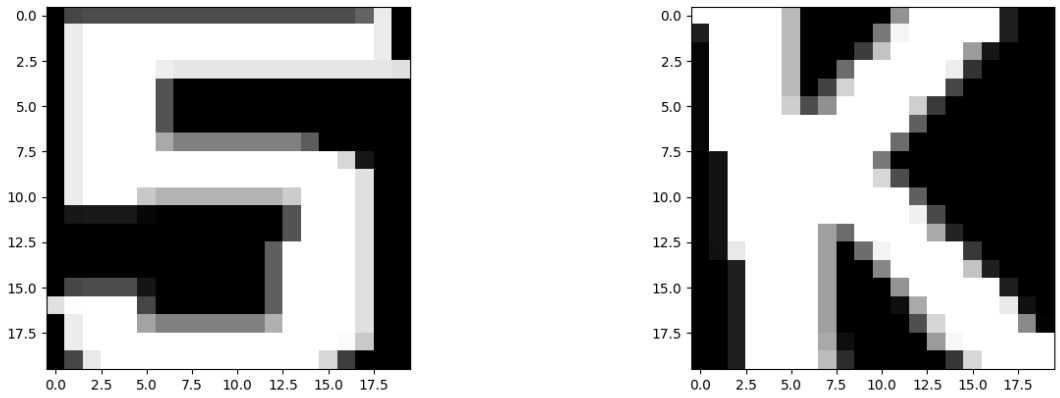

Figure 1: Images of `5` and `K` saved for training character recognition

Combining the results

Then the characters extracted from all the regions detected are appended and re-ordered to as present in the image. Most of the above wrongly-detected regions results in less than 3 character numbers which we can ignore and the correct license plate gives the longest output which is the output.

Datasets

For character recognition, we made our own dataset of 1498 characters which included the capital english alphabets(A-Z) and digits(0-9).
The License Plate dataset we used is taken from the website below to extract the licence plate number.

`https://platerecognizer.com/number-plate-datasets/`

We used the following research paper as reference.

`https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8270118`

Other existing approaches

Input:- Images of vehicles including their license plates. Eg: Vehicle images taken from security cameras installed at traffic signals. Output: Extracted licence plate number text on the detected license plate. We will evaluate our approach by testing it on a test dataset of images with vehicles and comparing the result with the actual text and calculating the accuracy. Approach1: Given the image captured by the security camera, Smearing Algorithm for the detection of license plate in the given image Pre-processing on the obtained license plate (Gray level image) Applying MedianBlur Smoothing Adaptive Gaussian Thresholding Morphological Transformation Creation of training model (Given License plate (Gray level Image)) CNN(Convolutional Neural Networks) Features Extract Then, obtained image features are given to LSTM(Long short-term memory) network. Decryption algorithm is applied to the output of LSTM network, to get the extracted text of license plate. The main approach remains the same for other implementations, but differ in method for detecting license plate. For ex., one implementation uses license plate color for detection as in some countries, license plates use some specific color.

Experimental Analysis

Using following images we provide a thorough experimental analysis of the project and the situations where it gives successful answer and the situations where it fails.

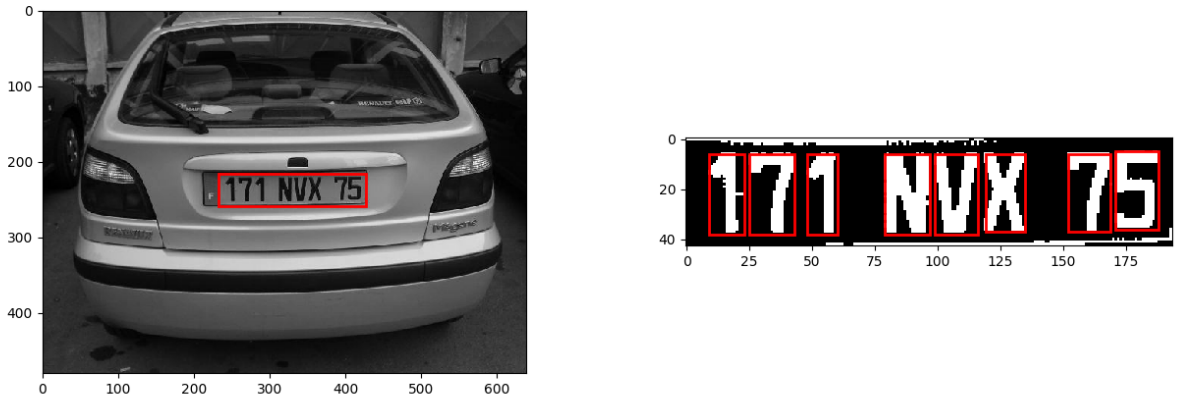

Figure 2: Successful Plate detection and segmentation

Simple class of aligned images: Figure 2 offers insight to the most general and simple class of images. Here the plate is horizontally aligned and has appropriate contrast, illumination and intensity for character segmentation. This class of images gives successful output when tested through out code.

Plate detection bottleneck: Figure 3 and Figure 3 offer insight to the scenarios when due to imperfect lighting or contrast conditions misleads the program to detect more than license plates leading to false positives. This could also be because of some tags/accessories pasted/installed on the vehicle(as in Figure 4 where a different sticker is also detected). **Possible get-around:** A possible solution to this is to treat each detected license plate as an entity and then reject that instance if no characters are detected through neural network.

Misaligned view and different orientations: Figure 4 and Figure 5 offer an insight to the conditions where the image capturing device is not perfectly aligned with the vehicle and hence produces spacial distortions in the image. The leads to reduced width of characters and the license plate itself. This creates

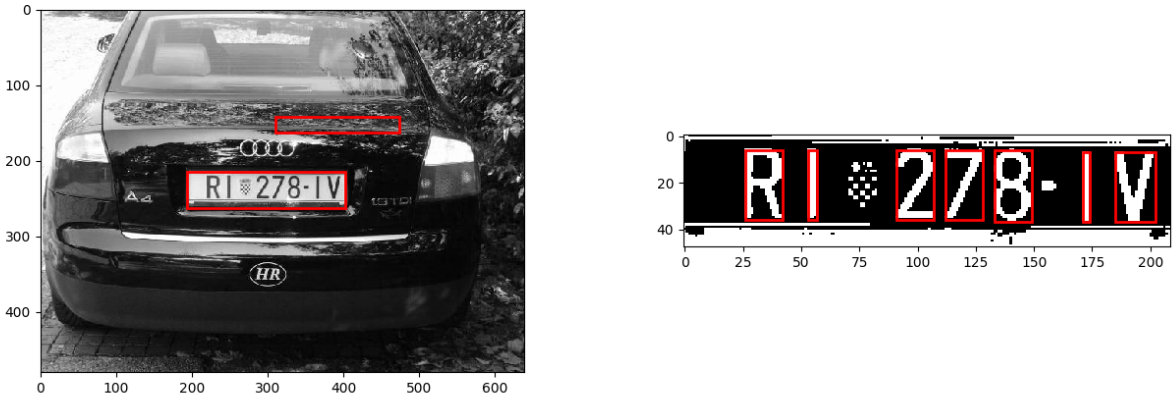

Figure 3: False positive to due illumination and contrast

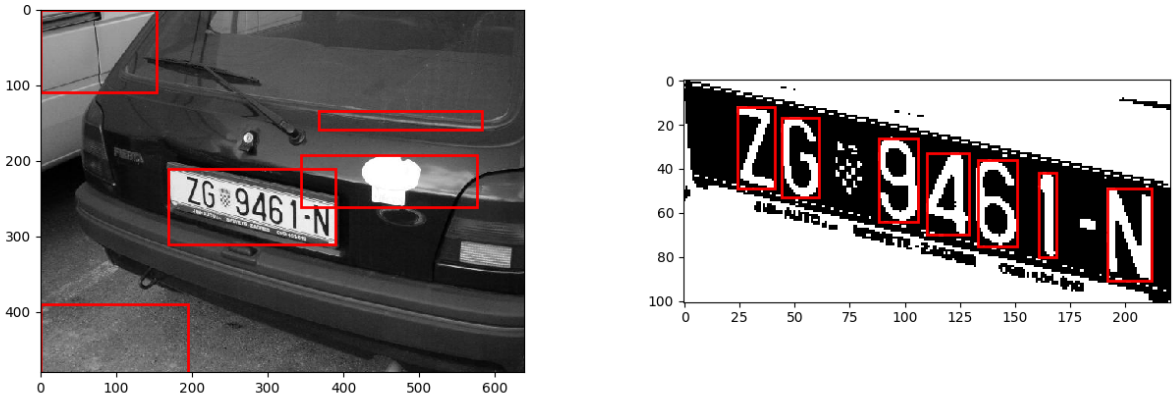

Figure 4: False positive to due external tag/stickers

problems even for Neural Network because now it has to accommodate the tilted characters. **Possible get-around:** A possible solution to this at the level of Machine learning is to accommodate tilted characters in the dataset. As far as detection and segmentation is concerned, there is scope of improvement if the initial parameters are changed accordingly. This is demonstrated in Figure 5 where initially, the program failed to detect plate, but after a few changes to the initial parameters, it successfully segmented characters.

Challenges in Implementation

The following were some challenges while implementing and executing the proposed project:

1. Detecting Number plate using Neural Network: We initially proposed detecting number plate using Neural Network itself. But that had following challenges:
    - Training Neural network for a particular orientation is easy and has a simpler structure in Layers.
    - Detecting plate for all possible orientations and arrangements require a lot of training data of images having license plates(of all possible arrangements) and images with no license plates at all.
    - We could not gather a satisfactory dataset satisfying above mentioned criteria.

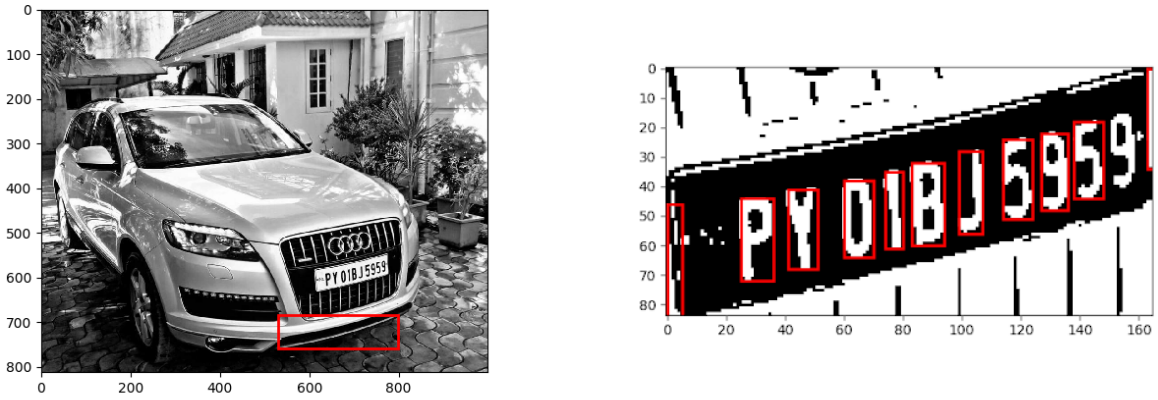

Figure 5: Changing boundary conditions on segmentation to detect license plate

2. Using appropriate segmentation parameters such as threshold, padding and filters was a difficult task so as to capture maximum possible set of plates.

3. Training an appropriate dataset for character recognition: We manually assigned tags to a 1500 strong set of characters in a number plate. This was necessary to maintain a uniform input layer in Neural Network.

Conclusion

After a careful and intensive thought involved in learning the approach and execution of code, we believe that there is a lot of scope of improvement in the current research work. This approach can be modified to adapt to individual capturing devices. For example we can hard code the orientation parameters for a fixed CCTV camera installed on roads. This is because orientation with respect to the device is always constant. Similarly, it is possible to change parameters dynamically to accommodate time of the day and adapt to contrast changes in the environment. We believe that this is quite an active field.

As far as our code is concerned, we have tried to accommodate all the constraints and parameters that we could think of. The existing libraries meant for character detection in python breaks down in quite a few situations. `Pytesseract`, most common alternative, fails to detect characters like 1 and i. It fails to detect isolated characters and hence we need to repeat the occurrence in an array. Even after doing so, it mixes up characters. Therefore our approach using Neural Network is much better than existing libraries.

References

1. Assignment 3

2. `scikit`, `cv2`, `skimage` libraries documentation in `python3`

3. `https://platerecognizer.com/number-plate-datasets/`

4. `https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8270118`

5. `https://github.com/apoorva-dave/LicensePlateDetector`