License Plate Recognition

Samridhi Bandlish
McMaster University
April 10, 2022

Abstract

In this study, I have proposed an algorithm that recognizes and extracts the license plates of vehicles by using Machine Learning (ML) and image processing techniques. Vehicle images captured tend to have more focused background than a clear license plate. The algorithm is trained to highlight the license plate encompassed in a rectangular box and retrieve it using feature extraction on the input image. It follows the series of steps that are commonly used in basic ML model training: image pre-processing, feature extraction and classification. Additionally, I have explored various currently existing solutions to this problem, most of which high-level Deep Learning procedures, and compared their results against the proposed method.

---

1. Introduction

License plates are the most important feature which is used to identify details of the vehicle. Recognizing the vehicles' registration plate number is very important for law enforcement, in case of speed-control, thefts and other crimes, and for the parking management systems. It's application is also essential in traffic flow control.

A simpler approach of recognizing the license plate of any vehicle is by using Machine Learning algorithms to train a model that specializes in plate detection. This includes (but is not limited to) inputting a set of images to train the model, preprocessing the images, implementing feature extraction (in this case, the license plate of the vehicle), classifying the model and testing it against a new set of images.

2. Related Works

Several models exist in current period that deal with the importance of vehicle's license plate detection and recognition. These models are programmed by utilizing the various aspects of Machine Learning, ranging from advanced model training to the deep learning algorithms.

The most common method is Automatic License Plate Recognition (ALPR)[1]. But many of the models using this method lack robustness and accuracy. The ALPR method was developed in 1976 and has been in use for a long period as it further evolved with time. ALPR method can be used in a number of ways to recognize the license plate of a vehicle. Some of these ways are:

1. Extraction Using Edge Information
2. Extraction Using Global Image Information
3. Extraction Using Color Features
4. Extraction Using Feature Learning

Another existing model uses Neural Networks to achieve the goal[2]. Fikriye and Figen implemented Probabilistic Neural Networks (PNN) approach that works on gray-level image. Preprocessing techniques that are added to the image include Bottom-Hat filtering, Otsu's thresholding and image correction. Once the license plate is recognized, the algorithm segments characters using CSV and the extracted output is fed to the PNN. They have obtained a 91% success rate in recognizing the license plate and over 96% success rate in character recognition.

A very common method that is in practice today is the use of Convolutional Neural Networks (CNN), a deep learning algorithm, for number plate detection. An image is chosen to preprocess and extract features of, before the training and testing datasets are generated.

Xie, Zhang, Zhao, Yang, Liu, and Yuan propose a new method of solving this problem based on Backpropagation Neural Network (BNN), another deep learning algorithm [3]. Their model uses the same basic steps, with the addition of a vertical and horizontal integral projection to detect the location of the license plate. They have achieved an overall success rate of 97.7% on number plate recognition.

3. Proposed Method

In this paper, I propose a novel method of detecting and recognizing the license plate of vehicles from any country using the techniques and concepts of image processing and machine learning systems. Figure 1 shows the flowchart of the proposed license plate detection algorithm.

I have placed the images of vehicles and license plates in two different folder: 'license' for the images that have a clear image of license plate and 'no license' for the images that do not contain the plates.

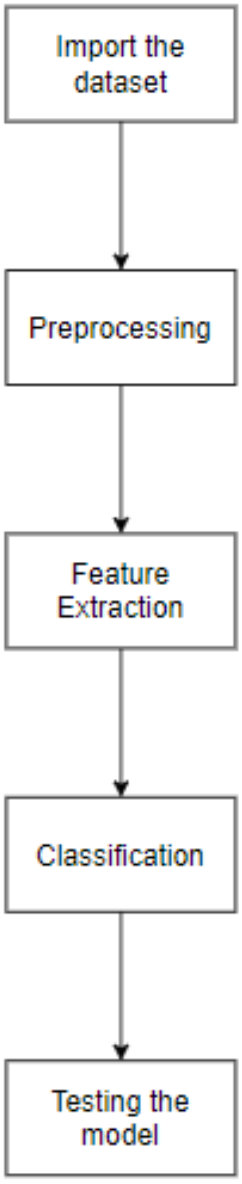

Figure 1. Process Diagram of the proposed method

3.1. Importing the Dataset

In order for the algorithm to begin implementing the editing steps, there needs to be a clear set of images that will be used for training the model. This could be done manually or by using python libraries such as *keras* or *os*.

In this project, I have used *os* library to join the directory path to both the 'license' and 'no license' folders containing the images. These folders also serve as the label for each image. This dataset is then separated into an image set and a corresponding label list (which must be converted into an array). I will be using an example image (Figure 2) to display the changes made to the images in the training set.

Figure 2. Original Image from the training set

3.2. Preprocessing

It is essential to preprocess an image(s) to prepare it before training the model. This can include resizing the image, orienting it or applying colour correction to the imported image.

In this project, I have resized the entire dataset to maintain a uniform dimension throughout (200x300), to avoid running into any errors. This was followed by converting the images to grayscale. A smooth filter was applied to the grayscale images, and then the edge detection was applied to find the edges within the image.

To implement these changes, *OpenCV* library's built in functions were used. Canny Edge Detector was implemented for the edge detection within the image. Figure 6 shows the edges in the example image used for this report.

Figure 3. Image resized to dimensions 600x400

Figure 4. Image after Grayscale conversion

Figure 5. Image after Blur application

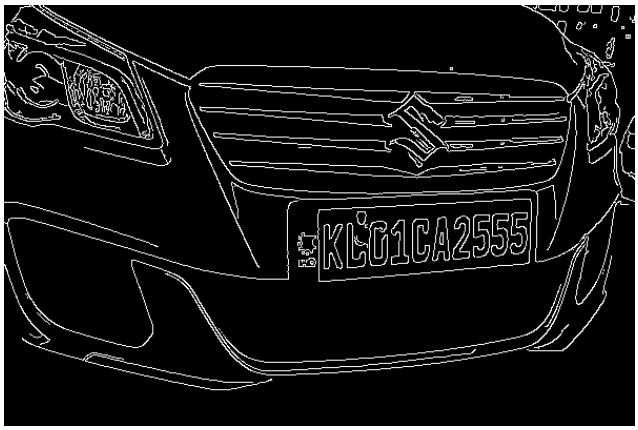

Figure 6. Detected edges in the image

3.3. Feature Extraction

Feature Extraction is crucial when training a model because it tells the algorithm what exactly it should be looking for in the set of images.

In this project, I have established the contours in the images and sorted them according to their sizes. Contours are an outline bounding the shape of an object in an image (in this case, the shape is a rectangle). As an element loops through the detected contours, an approximation of the largest rectangle from the contours is calculated. If the rectangle exists, a mask of the same shape is created and the contour lines are placed on it. The mask is then applied to the resized version of the image using the *bitwise-and* function of *OpenCV* library. The resulting image is the license plate (Figure 8).

If there is no rectangle detected in the image, the resized image is returned and the label is changed to 'no license'. This indicates that the image has no visible license plate.

Figure 7. Image after the largest rectangular contour is calculated

Figure 8. The Extracted License Plate

3.4. Classification

Classification is the most important aspect of Machine Learning model training. In simple words, image classification is categorizing images based on the labels or class they are awarded. There are many ways of building a classification model but I have used Support Vector Machine (SVM) on HOG for this study.

A HOG train was created and a loop runs through the length of the entire training set. The license plate images are converted to grayscale, and *fd* and *hog image* are found using the *hog* function imported from the *skimage* library. The fd is then appended to the list. From the SVM, the SVC function is imported and the kernel is set to 'linear'. The model is fit to the HOG train and the expected labels. This creates the ML model that is ready for testing.

3.5. Testing the Model

Testing the model allows us to visualize the accuracy of the algorithm. In this study, I have created a HOG test list, following the steps in the previous section, but using the images in the testing set instead. The prediction of the labels was retrieved using HOG train and the classification report (with the actual labels and the predicted labels) was printed.

4. Experimental Results

In this study, I have tested the model against various range of images for the testing dataset and each test has produced a different accuracy score.

For ten images in the test set, the accuracy of the model was found to be 60%. For more ten different images, the model accuracy dropped to 30%.

For a training set of 330 images and a testing set of 120 images, the model accuracy ranged from 60-90%.

Compared to the CNN model, the results of the proposed method are low. The CNN model generated an ideal and optimized model using the data from the directory and the *Sequential* library. Assuming that the model is trained and tested over hundreds or thousands of images, the accuracy was shown to be over 60%.

The difference in accuracy could be due to several reasons. First, the images I used for my algorithm might be labelled incorrectly or be too distorted. Second, the method I used for classification is not optimized and is bound to interfere with the results.

5. Conclusion

Overall, the proposed method resulted in fairly low accuracy for some ranges of input while it performed adequately for the others. This was due to the data images chosen to train and test the model. It features preprocessing of the images, extracting the license plate for some, while placing others in the 'no license' category, and classifying the images into the resulting categories.

6. References

[1] Lubna, N. Mufti, and S. Shah, "Automatic Number Plate Recognition:A Detailed Survey of Relevant Algorithms." *Sensors (Basel, Switzerland)*, vol. 21, April 2021.

[2] Fikriye Öztürk, and Figen Özen, "A New License Plate Recognition System Based on Probabilistic Neural Networks", *Procedia Technology*, vol. 1, 2012.

[3] F. Xie, M. Zhang, J. Zhao, J. Yang, Y. Liu, and X. Yuan, "A Robust License Plate Detection and Character Recognition Algorithm Based on a Combined Feature Extraction Model and BPNN", *Journal of Advanced Transportation*, 2018.