**Car Plate Characters Recognition**

| Tan Kuang Chen  Faculty of Computer Science and Information Technology, Multimedia University  1171103939@student.mmu.edu.my | Chan Jun Yang  Faculty of Computer Science and Information Technology, Multimedia University  1171103573@student.mmu.edu.my |
| --- | --- |
| Chan Jun Ting  Faculty of Computer Science and Information Technology, Multimedia University  1171103572@student.mmu.edu.my | Tee Wai Bing  Faculty of Computer Science and Information Technology, Multimedia University  117110537@student.mmu.edu.my |

# **Abstract**

The number of cars around the world keeps increasing and traffic conditions in cities have attracted more and more attention. As we know, it is hard for traffic officers to keep track of driver car number plates especially in bad weather like rain. The goal of this project is to study computer vision for different country car plate recognition during rainy weather using different methods. Then, compare the methods using evaluation metric results. To define which methods perform a better result in recognizing number plates in rain.

100 images of different angle car number plates collected from one website to test the hypothesis of the methods that we apply are capable of displaying a good result. The methods that applied in this study are k-nearest neighbors (KNN), EasyOCR, Convolution Neural Network (CNN) and Tesseract LSTM OCR. To save time, we add rain effects in every image instead of searching images that were captured during rain one by one.

In conclusion, K-nearest neighbours have shown the highest accuracy rate among four models on the original image, with 80% and 38% on the rain image. Convolutional Neural Network (CNN) had the lowest accuracy rate of any of the four models on the original image, with 24%, and the rain image, with 10%.

*Keywords — Car Plate; KNN; EasyOCR; CNN; Tesseract LSTM OCR*

# **1. Introduction**

Car Plate recognition is an image-processing system that is used to identify the car number plate. With the continual expansion in the number of cars in many nations throughout the world, traffic conditions and challenges have drawn increasing attention. The issue of how to efficiently control traffic has captured the attention of key government departments in a number of nations. To address this issue, an Intelligent Transportation System (ITS) is developed.

The most essential problem that ITS can address is to reliably recognise a vehicle's information when a driver violates the law on the road, which is frequently speeding. According to Wikipedia contributors the number plate was registered as a numeric or alphanumeric ID that uniquely identifies a vehicle or vehicle owner [10] which is the source and direct processing object of a large amount of information in the ITS that makes number plate recognition systems research the core of it. The number plate recognition system is an analytical approach that extracts the number plate area from an image and then uses that area for character segmentation and recognition. As a result, the methods for detecting the number plate are the first and most important issues in the number plate recognition system. The objective of this project is to study computer vision for car plate recognition during rainy weather using different methods. We will compare the car plate characters recognized by different methods. The whole string characters recognized on the car plate are used to compare with the original characters of the car plate.

# **2. Background/Related Work**

For Car Plate character recognition, many approaches are being explored. This section describes some recent related efforts for car number plate recognition that are relevant to the proposed technique.

## **2.1. k-nearest neighbors**

Dudhe and Mohekar proposed a system where it can find and display the owner details of the number plate [2]. This is because the authors think that the system would help the Educational Institutions in recognizing and keeping the records of the students in the parking lot. Firstly, it used a thresholding method to segmentize the image. Then KNN classifiers have been trained to identify the characters appearing in the car number plates. The system then localized the number plate and used a character recognition algorithm to find the alphabets and number of the characters recognized by KNN classifier. Lastly, the system used the number plate detected to find the plate owner details then display it in the application developed.

A plate recognition process has been proposed to analyse the vehicle license plate captured on the road [12]. The authors have detected and processed the contours of the images captured and then recognize the characters appearing in the license plate found with KNN machine learning model. The KNN algorithm is trained with data consisting of 36 different characters to recognize the characters segmented in the license plate. The KNN algorithm successfully achieved 87.43% on identifying the characters in license plates by using the K = 1.

## **2.2. EasyOCR**

Awalgaonkar, Bartakke and Chaugule (2021) proposed research for an Automatic License Plate Recognition System using a Single Shot Detector (SSD). [1] The research states that the majority of present ALPR systems fail to demonstrate adequate performance in real-time image and video scenarios, which is the problem they want to solve. The data for this research article is 800 images of car number plates. The methods they use for licence plate detection are Single Shot Detectors (SSD). For recognition, they use Tesseract OCR and Easy OCR. The complete system was built using the NVIDIA Jetson Nano and Raspberry Pi 3B hardware platforms. These three sub-blocks' characteristics have been tuned to produce real-time ALPR performance with acceptable precision. The result shows the implemented system on the Jetson Nano enables the processing of movies for ALPR with greater than 95% accuracy.

## **2.3. Convolution Neural Network**

Nguyen proposed research for Vietnamese Car Plate Recognition using CNN. The research states that number plate recognition is a form of an intelligent transportation system [4]. Despite extensive study on plate detection, character segmentation, and character recognition, there are still plenty of difficulties to be identified and overcome. CNN displays an effective classification method for achieving cutting-edge results on a variety of identification tasks. CNN-based approaches are being applied to handle difficulties such as plate detection, character segmentation, and character identification with different challenges of number plate recognition. It depends on the quality of each task to identify the quality of identification. The goal of the research is to recognise the entire number plate sequence without segmentation to achieve a high accuracy using CNN-based. 1000 US and 1000 Vietnamese number plate photos collected to be tested in the research. The results show that CNN precision, recall and F1 score dominated for both US and Vietnam car number plate dataset. For the Vietnam car number plate dataset the score for precision was 98.5%, for recall was 99.3% and for F1 was 98.9%. Meanwhile, on the US car number plate dataset the score for precision was 99.5%, for recall was 99.6% and for F1 was 99.5%. These CNN-based are very capable in number plate recognition.

## **2.4. Tesseract LSTM OCR**

Jaskirat Singh and Bharat Bhushan proposed a Automatic Number Plate Recognition (ANPR) system that acts more robustly in different challenging scenarios than the previously proposed ANPR systems [3]. This work aims to design a robust technique for license plate detection in the image by using deep neural networks and the perform the license plate recognition using LSTM Tesseract OCR Engine. According to our experimental results, they have successfully achieved robust results with an accuracy of 99% of detection license plate and an accuracy of 95% of license plate recognition.

# **3. Approach**

Figure 3.1 shows the framework of the car plate recognition in this project This framework is based on the combination of the car plate detection and character detection that is used to find out the location of the character in the images as well as different ways to recognize the character.



Figure 3.1(a) Framework of Car Plate Recognition

KNN, EasyOCR and LSTM Tesseract methods have used the same plate detection method while we are having hard time on using the same plate detection on the CNN approach. Hence, the license plate detection method in the CNN approach is different from the other approaches.

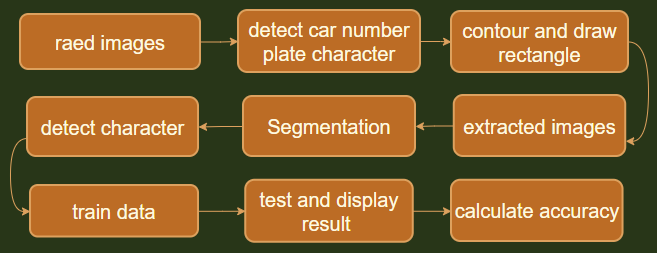


Figure 3.1(b) Framework of Car Plate Recognition for CNN

Figure 3.1(b) shows the flow for CNN methods. First read images and detect the number plate by contour and draw the rectangle line. Then extracted images to make it more visible. Continue with segmentation and detect the character one by one. Before doing testing, train the collected data. The prediction model will predict the output and calculate the accuracy.

**3.1. Image Preprocessing**

First of all, the images will convert to the HSV color space and obtain the value image as Grayscale images. This is because when compared to the RGB color space, the HSV color space can very intuitively express the brightness, tone ,and vividness of color, so it can be more convenient in contrasting between colors. Then, MorphologyEx and GaussianBlur were used for smoothing and denoising. For the thresholding, an adaptive threshold was selected as it can divide images into different areas and calculate the threshold for each area. Figure 3.2 shows the effect of images preprocessing.

| (a) | (b) |
| --- | --- |
| (c) | (d) |
| Figure 3.2 Image after Preprocessing  (a)Grayscale, (b) MorphologyEx,  (c) GaussianBlur, (d) Adaptive Threshold | |

**3.2. Car Plate Detection**

In the part of car plate detection, contours feature would be extracted to find the contours of the characters. Contours is a curve composed of a series of connected points that can represent the basic shape of the object in image. Generally, the operation of finding the contour is used for the binarization images and the binary images were already obtained by using Adaptive Threshold. The contours would be extracted continuously until there are no character sizes that can be extracted in the images. The contour that has the most value inside would be defined as the location of the car plate. Figure 3.3 shows the car plate that is detected by using contours feature.

|  | |
| --- | --- |
| Figure 3.3 Image after Detection | |

**3.3. Character Recognition**

Lastly, character recognition would be implemented. In this paper, we have implemented four different methods on doing character recognition on the license plate images which included KNN, EasyOCR, CNN ,and Tesseract LSTM OCR .

# **4. Experiment**

**4.1. Dataset Collection**

In this project, different angle car number plate images were collected from Srinivas-Natarajan (2020) [6] where each of the images were labelled with the actual plate number of the images shown. The reason we collected the images labelled with license plate numbers is to compare the predicted license plate characters from the models with the actual license plate character. This is because by comparing two license plates, we can see the result easily and calculate the accuracy of the model precisely. In addition, we also referred from UjjwalSaxena [8] to add a rain effect on all the images to challenge the methods of detecting car number plates. After adding the rain effect in every image, the first 100 images were saved in a folder called ‘Img\_rain’ to be tested for our model.



Figure 4.1(a) Input Images



Figure 4.1(b) Input Images with Rain Effect(Heavy)



Figure 4.1(c) Input Images with Rain Effect(Torrential)



Figure 4.1(d) Input Images with Rain Effect(light)



Figure 4.2(a) Number Plate After Extracted



Figure 4.2(b): Number Plate After Extracted with Rain Effect

Figure 4.1(a) shows the images of the dataset used, which figure 4.1(b), 4.1(c) and 4.1(d) clearly show the rain effects applied on the original images. Input images with heavy rain effects have been selected as the test data because we think that the torrential and light rain effects are not suitable for testing the models. Figure 4.2 shows the Number Plate After Extracted for both original and rain effect images to let function to detect the characters more accurately.

**4.2. Evaluation Measurement**

To define the accuracy of the approaches used on recognizing the license plate characters, we used the SequenceMatcher function in Python to find the ratio of the predicted characters on license plates of the models and the actual license plate characters observed by the human eye. The overall accuracy of different approaches are measured by using (total accuracy) / (total images used).

## **4.3. K-nearest neighbors Result**

A K-nearest Neighbour approach on recognizing the characters of license plates has been developed by Srinivas-Natarajan where the KNN approach in this project is referred to [6]. We have added the evaluation method on the existing program to see the result and accuracy of the predicted values of the license plate. Two different text files named ‘classification.txt’ and ‘flatten\_images.txt’ have been used by the author to train the KNN model where we used the same training data on training the KNN model to identify the alphabets and numbers on the license plate [6]. We tuned the K values of the KNN model while predicting the license plate and the result of the tuned KNN model is shown below in the figures.

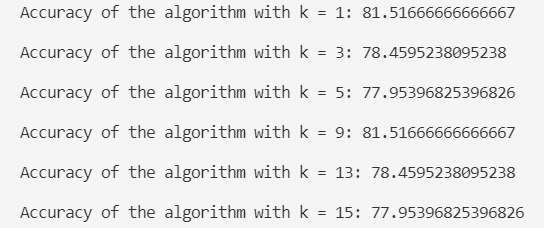


Figure 4.3: Original Images Accuracy (KNN)

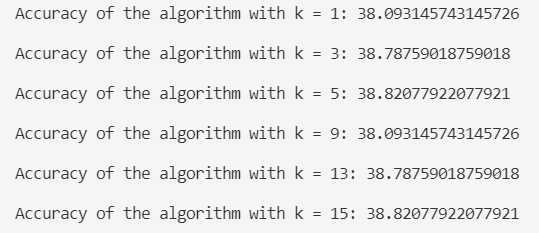


Figure 4.4: Rain Effect Images Accuracy (KNN)



Figure 4.5: Detected Number Plate in Original Images (KNN)



Figure 4.6: Detected Number Plate in Rain effect Images KNN)

Figure 4.3 and 4.4 show the result of the accuracy of KNN on recognizing the character on the license plate. We can clearly see that after rain effect is applied on the images, the accuracy of the on recognizing the characters on license plate has significantly dropped. The result is different most probably because the rain effect is affecting the detection of the plate and recognition of the characters of the model. Figure 4.6 clearly shows that the rain effect image affected performance of the model while recognizing characters as compared to figure 4.5.

## **4.4. EasyOCR Result**

The optical character recognition library EasyOCR reads brief texts (such as serial numbers, part numbers and dates). For recognition on the car number plate using EasyOCR, it was based on Renotte (2020) [5]. The result is achieved for the image in figure 4.7 and 4.8.

Original images:

|  |
| --- |

Figure 4.7: Detected Number Plate in Original Images (EasyOCR)

Rain effect images:

|  |
| --- |

Figure 4.8: Detected Number Plate in Rain effect Images (EasyOCR)

Based on figures 4.7 and 4.8, one of the original and rain-effect images using EasyOCR on recognition. The original image recognises with a good result that all the characters are correct. For the rain effect image, it shows that not all characters are correct, which means 4 out of 5 characters are correct. This result shows that rain has an effect on recognition of car number plates using EastOCR.

Original images:



Figure 4.9: Original Images Accuracy (EasyOCR Result)

Rain effect images:



Figure 4.10: Rain Effect Images Accuracy (EasyOCR Result)

Based on figures 4.9 and 4.10, the accuracy result on 100 original images and 100 rain effect images was calculated. For original images, accuracy is 66.19%, which means 66.19% of the recognition on car number plates is correct. Rain effect images show a huge difference from the original image, which only contains 32.98% accuracy. This shows that the images containing rain have an effect on the recognition of car number plates using EastOCR.

## 

## **4.5. Convolution Neural Network Result**

For CNN we referred to a work done by Vajpayee from Kaggle [9]. Not many changes were made from the original source code. Only the first model.add( ) output shape has been modified from 16 to 32, the dropout was adjusted from 0.4 to 0.5 and epoch quantity changed from 80 to 20. Due to lack of programming skill we are unable to read more than one image for CNN methods. Hence, only one image is able to do recognition at one time. In this section we will show the difference between number plate recognition before (original image) and after adding the rain effect.



Figure 4.11(a): Number Plate Detection



Figure 4.11(b): Number Plate Detection with Rain Effect



Figure 4.12(a): Extracted Incorrect Area



Figure 4.12(b): Fail to Detect Number Plate

First import all the necessary libraries. Loads the data required for detecting the license plates from CascadeClassifier( ) which the source code also provides and saves under ‘number\_plate.xml’. Then, construct a function called detect\_plate which is mainly to detect the number plate's coordinates and contours it by drawing rectangles around the edges.

Sometimes the detector fails to define correct coordinates, and detect more than one area like Figure 4.11(b). Occasionally, it also extracts the wrong images like Figure 4.12(a). If the detector fails to define any character error occurs like Figure 4.12(b) and causing the accuracy of this image is 0.

By using the display function 4.11 shows the rectangle range that has been contoured from the input images, while Figures 4.2 show images after the extraction for both original and rain effect.

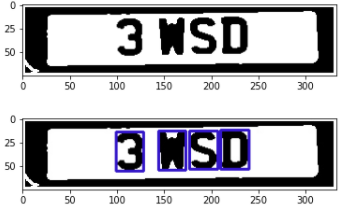


Figure 4.13(a): Segmented Number Plate



Figure 4.13(b): Segmented Number Plate with Rain Effect

Then, using the segment\_characters( ) function to segment the images into black and white along with the find\_contours( ) function do detect the characters. As we can see from Figure 4.13(b) which is unable to display the images with blue rectangles which mean the function fails to detect any characters due to the rain effect and different angles of images.

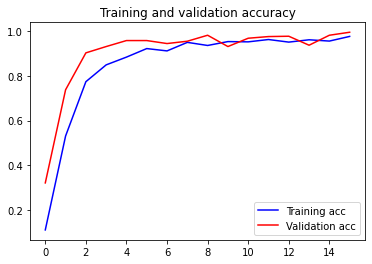


Figure 4.14(a): Training and Validation Accuracy

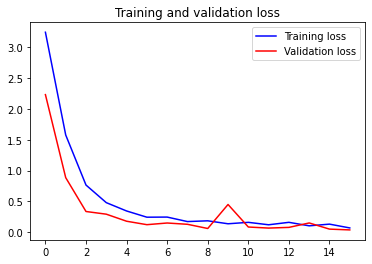


Figure 4.14(b): Training and Validation Loss

After the 20 times epoch training Figure 4.14 shows the accuracy and loss for train and validation dataset. The train dataset has 864 photos, while the validation dataset contains 216 images, with both datasets comprising 36 classes provided by Vajpayee [9]. Both datasets are stored in a folder called ‘Dataset’.

The epoch will stop running when the accuracy is more than 0.99. Based on the 4.11 the accuracy for training is excellent, with most of the epoch accuracy being close to 1 and even stop training after 16 times. Same as loss, most of the time the loss is close to zero, only a spike to 0.5 between 8 and 10 epochs resulted in validation loss.



Figure 4.15(a): Car Number Plate Recognition Result After Prediction

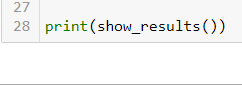


Figure 4.15(b): Car Number Plate Recognition Result After Prediction with Rain Effect

After finishing the training work, the program will predict the characters based on what it has been trained. As a result, Figure 4.15 shows one of the test image results for both types of images. For Figure 4.15(a) four out four characters are predicted correctly for the original images, while Figure 4.15(b) didn’t show any output since the program was unable to detect any character for rain effect.

As mentioned due to lack of programming skill we have to manually change the input images name to test one by one for 100 images for both original and rain effect dataset. To define the accuracy we will use the SequenceMatcher( ) function using the dataset called ‘CNN Resutl.csv’ which contains the results for both original and rain effect car number plate character recognition.

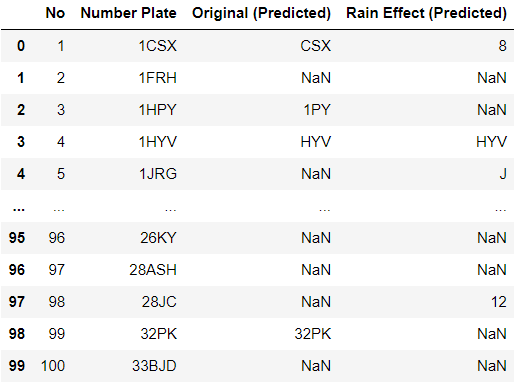


Figure 4.16(a): CNN Result.csv

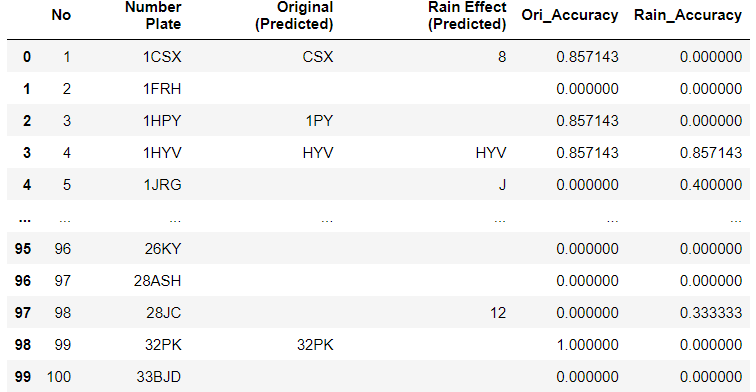


Figure 4.16(b): CNN Result.csv After SequenceMatcher

Figures 4.16 show the dataset before and after SequenceMatcher( ). Before displaying the test accuracy, for your information the nan value in the dataset are the images that are unable to be detected. To do SequenceMatcher all the nan values are replaced with “ ” which is the readable nan value.



Figure 4.17: CNN Original and Rain Effect Images Test Accuracy

As we can see from Figure 4.17 show the mean accuracy for 100 test images for both original and rain effect. The problem of number plate character detection, rain effect and maybe the data for training causing the accuracy records is as below par with 24% accuracy for original images test and 10% accuracy for rain effect images.

## **4.6. Tesseract LSTM OCR**

Tesseract LSTM OCR can read eleven different languages. The primary character classifier function in Tesseract OCR is based on an implementation of a Long Short-Term Memory neural network or LSTM network. LSTM neural networks outperform all other alternative neural network architecture models for this type of pattern recognition and also outperform the more "classical" character recognition algorithms used by the top selling commercial OCR products. If Tesseract's LSTM neural network recognizer fails on a particular character sequence, it can "fall-back" to its generic static shape classifier to make the determination. So in essence, Tesseract LSTM is actually two OCR classifiers. The LSTM Tesseract that was used on this paper based on Stefan Weil (2021) [7].Figure 4.183 shows the result of car plate recognition in original images and rain effect images by using LSTM Tesseract. Figure 4.19 shows the license plate recognition accuracy score obtained by using LSTM Tesseract.

| (a) |
| --- |
| (b) |
| Figure 4.18 Result of Tesseract LSTM OCR  (a) Result of Original Image, (b) Result of Rain Effect Image |

| (a) |
| --- |
| (b) |
| Figure 4.19 Accuracy Score of Tesseract LSTM OCR  (a) Accuracy on Original Image, (b) Accuracy on Rain Effect Image |

Based on the figures above, we noticed that the rain actually will affect the effectiveness of car plate recognition by using LSTM Tesseract and cause the accuracy score from 52.57% direct to 33.48%.

## **4.6. Comparison of Evaluation Result**

Table 4.1 Evaluation Result

| **Algorithm** | **Accuracy Original Image** | **Accuracy Rain Image** |
| --- | --- | --- |
| K-nearest neighbors | 80% | 38% |
| EasyOCR | 66.19% | 32.98% |
| CNN | 24% | 10% |
| Tesseract LSTM OCR | 52.57% | 33.48% |

Table 4.1 shows the accuracy score obtained from different algorithms for car plate recognition on the original images and rain effect images. The reason CNN algorithm receives the lowest accuracy is because the number plate detection used is different from other three algorithms whereby most of the time the detectMultiScale( ) is unable to detect the true location of the number plate. Sometimes, the function will show error if unable to define any characters. Plus, the find\_contours( ) also are not good in catching the number plate character causing the prediction model to predict the wrong characters.

# **5. Conclusion**

With the development of image processing and computer vision to be increasingly mature, it is more and more widespread for the car plate recognition system to be used in practice. In this paper, after the cars image has been processed by digital image processing, the contour feature of the image is extracted to find the car plate location, and the recognition of car plate is recognized with four different methods, which are k-nearest neighbors (KNN), EasyOCR, Convolution Neural Network (CNN) and Tesseract LSTM OCR. Those methods have a low accuracy score on recognizing the car plate image after applying the rain effect. Besides that, inaccurate on detecting the location of car plates and segmentation of character also may be the reasons for obtaining a low accuracy score. Thus, the algorithm needs to be further improved.

# **6. References**

[1]Awalgaonkar, N., Bartakke, P., & Chaugule, R. (2021). Automatic License Plate Recognition System Using SSD. 2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA), 394–399.<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9588707>

[2]Dudhe, P., & Mohekar, R. (2020). EXTRACTION OF OWNER DETAILS WITH INTELLIGENT NUMBER PLATE RECOGNITION. International Research Journal of Engineering and Technology (IRJET), 7(9), 3713–3718.<https://www.irjet.net/archives/V7/i9/IRJET-V7I9652.pdf>

[3]J. Singh & B. Bhushan. (2019). Real Time Indian License Plate Detection using Deep Neural Networks and Optical Character Recognition using LSTM Tesseract. Retrieved from<https://ieeexplore.ieee.org/abstract/document/8974469>

[4]Nguyen, T. N., & Nguyen, D. D. (2018, November). A new convolutional architecture for Vietnamese car plate recognition. In 2018 10th International Conference on Knowledge and Systems Engineering (KSE) (pp. 7-12). IEEE. Retrieved from<https://sci-hub.mksa.top/10.1109/KSE.2018.8573375>

[5]Renotte, N. (2020, December 13). GitHub - nicknochnack/ANPRwithPython. GitHub.<https://github.com/nicknochnack/ANPRwithPython>

[6]Srinivas-Natarajan. (2020). Srinivas-Natarajan/license-plate-recognition. GitHub. Retrieved from<https://github.com/Srinivas-Natarajan/License-Plate-Recognition>.

[7]Stefan Weil (2021, November 15). GitHub - tesseract-ocr/tesseract. GitHub.<https://github.com/tesseract-ocr/tesseract>

[8]UjjwalSaxena. (2018, May 22). Automold--Road-Augmentation-Library. GitHub.<https://github.com/UjjwalSaxena/Automold--Road-Augmentation-Library/blob/master/Usage%20Guide.ipynb>

[9]Vajpayee, S. (2021, March 20). License plate recognition using CNN. Kaggle.<https://www.kaggle.com/sarthakvajpayee/license-plate-recognition-using-cnn/notebook>

[10]Wikipedia contributors. (2021, November 12). Vehicle registration plate. Wikipedia.<https://en.wikipedia.org/wiki/Vehicle_registration_plate>