# A COMPARATIVE ANALYSIS OF LICENSE PLATE DETECTION AND RECOGNITION METHODS

*Chan Y.H., Boe C.H., Tan K.Y., Tan W.T.*

**ABSTRACT**

Automated License Plate Recognition (ALPR) systems are widely used for various applications such as traffic enforcement, parking fee payment, toll payment, vehicle theft prevention, and others. Due to different font styles, low-resolution photos, and uneven lighting, license plate identification is still a difficult process, despite its widespread use. In this study, we sought to create an ALPR system that is reliable and accurate and can quickly identify license plates.A total of 400 images of Malaysian license plates were used to evaluate five different model pipelines for license plate recognition: LPRNet, WPOD-net + MobileNets, WPOD-net + Pytesseract, VGG16 + Keras OCR (10% expansion), and OpenCV + EasyOCR. The evaluation metrics used were Average Character Level Accuracy, Average Levenshtein Distance, and Accuracy. The outcomes demonstrated that the LPRNet model performed best, with an average Levenshtein Distance of 0.7200 and an accuracy of 78.5%. The accuracy values of the WPOD-net + MobileNets and WPOD-net + Pytesseract models, respectively, were 16.25% and 15.25%, showing lower performance levels. The accuracy of the OpenCV + EasyOCR model was 28.00%, compared to 41.75% for the VGG16 + Keras OCR (10% expansion) model. In conclusion, of the models tested, the LPRNet model performed the best at identifying license plates. The findings of this study can assist practitioners and researchers working on license plate recognition to make informed decisions and select the best approach for their unique needs.

**Keywords - Automated License Plate Recognition (ALPR), Malaysian license plates, LPRNet, WPOD-net with MobileNets, WPOD-net with Pytesseract, VGG16 with Keras OCR, OpenCV, and EasyOCR**

## 1. INTRODUCTION

A system that automatically reads and identifies the characters on a vehicle's license plate is called "automotive plate recognition," often referred to as "license plate recognition". This method is often used in a variety of applications, such as license plate recognition may be used to track traffic flow and identify vehicles that have entered or departed a certain area. Law enforcement officials may use car plate recognition as a technique to find vehicles that have been involved in crimes or are wanted on outstanding warrants. Auto plate recognition may be used to detect vehicles entering or departing a secured location, such as a military base or a parking lot.

To accurately extract the text from a vehicle plate and recognize the characters is the problem statement for a car plate identification system. Due to changes in the appearance of license plates, such as varied typefaces, widths, and lighting conditions, this can be a complex process. It is an interesting technology because it has the ability to increase the effectiveness and precision of many jobs, including traffic enforcement and monitoring. Because it may be used to identify cars engaged in crashes or other emergency situations, it also has the potential to increase safety.

In Malaysia, LPR systems are widely used to automate parking fee payments, toll payments, and vehicle theft prevention. However, the recognition of Malaysian license plates is a challenging task due to several factors such as the variation in font styles, low-resolution images, and non-uniform illumination. The problem we aim to address is the development of a robust and accurate LPR system that can recognize Malaysian license plates accurately and efficiently. To achieve this goal, this project aimed to implement different visual image processing algorithms and deep learning algorithms and work on a comparative analysis of these different methods.

## 2. RELATED WORK

Based on the paper [1], Sérgio Silva describes a full ALPR system in his work "Automatic License Plate Recognition (ALPR) for Unconstrained Environments," which focuses on unconstrained capture settings where license plates may be deformed owing to oblique perspectives. The research offers a unique Convolutional Neural Network (CNN) capable of recognizing and correcting numerous deformed license plates in a single image, which is then put into an Optical Character Recognition (OCR) approach to produce the final result. In difficult cases, the suggested solution surpasses both commercial and academic alternatives. The approach suggested in this research addresses the difficulty of identifying license plates in unconstrained circumstances, where the license plate may be significantly deformed due to oblique views. The paper presents a unique Convolutional

Neural Network (CNN) capable of recognizing and correcting numerous deformed license plates in a single image, which is then put into an Optical Character Recognition (OCR) approach for the final result.

The suggested WPOD-net employs a convolutional neural network (CNN) particularly built for recognizing and correcting deformed license plates in images. The network takes a bounding box that has been pre-detected as having a car as input and uses a dense sliding window search to locate the license plate inside the region. After locating the license plate, the network reverts back to an affine transformation matrix which can be used to correct the license plate to a frontal perspective. This rectification phase is essential for the following OCR network because it guarantees that the license plate letters are aligned and easy to identify. The WPOD-net network is in charge of detecting and unwarping distorted license plates in the proposed system. The network is built with the TensorFlow framework, while the first vehicle detection and OCR-NET are built with the DarkNet framework. To connect the two frameworks, a Python wrapper was employed. The studies were run on an Intel Xeon CPU with 12GB of RAM and an NVIDIA Titan X GPU. This setup allowed the entire ALPR system to operate at an average of 5 frames per second, depending on the number of cars identified in the input image. According to the authors, boosting the vehicle detection threshold results in greater FPS but lower recall rates.

From the paper, most existing ALPR algorithms are focused on a single license plate location and usually investigate datasets with nearly frontal pictures. To overcome this, the suggested technique is intended to operate in unrestricted situations where the license plate may be significantly distorted. The approach detects and corrects deformed license plates using a CNN, and the final output is obtained using an OCR method. In addition, the research includes hand annotations for a difficult collection of license plate photos from various areas and acquisition settings. As a result, the data reveal that the proposed technique beats alternatives in normal conditions and outperforms both in difficult cases.

The paper [5] "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" introduces a new class of efficient models for mobile and embedded vision applications. The paper presented a novel architecture based on depthwise separable convolutions that enable the construction of lightweight deep neural networks. Because of its basic structure, the suggested model, MobileNet, is easy to develop and explore, and it features two hyperparameters to balance latency and accuracy. The research contains comprehensive experiments to analyze the resource-accuracy trade-off and shows that MobileNets surpass other recent models on ImageNet categorization.

Furthermore, the study illustrates the usefulness of MobileNets in a variety of applications and use scenarios, such as item identification, fine-grain classification, facial characteristics, and large-scale geo-localization. The study finishes by emphasizing the importance of MobileNets and aims to distribute TensorFlow models to facilitate adoption and investigation. The authors also propose a method for developing mobile FaceNet models that uses distillation to reduce the squared disparities between FaceNet and MobileNet outputs in the publication. The results suggest that tiny MobileNet models can outperform the bigger FaceNet model. As previously stated, the MobileNet model is based on depthwise separable convolutions, which factorize a normal convolution into a depthwise convolution and a 1x1 convolution called a pointwise convolution. By applying a single filter to each input channel using depthwise convolution and combining the outputs using pointwise convolution, this factorization reduces processing and model size.

Based on the paper, other than the first layer, which is a complete convolution, the MobileNet structure is constructed on depthwise separable convolutions. With the exception of the last fully connected layer, which feeds into a softmax layer for classification, all layers are followed by batch norm and ReLU nonlinearity. The study compares a standard convolution layer to a factorized layer using depthwise and pointwise convolutions. In the depthwise convolutions and the first layer, downsampling is accomplished via stridden convolution. The paper's conclusion is that the authors suggested a novel model architecture dubbed MobileNets, which uses depthwise separable convolutions. They explored the critical design choices that lead to an effective system and illustrated how to use width and resolution multipliers to construct smaller and quicker MobileNets. When compared to popular versions, MobileNets demonstrated improved compactness, speed, and accuracy. The authors also proved the efficiency of MobileNets in a variety of tasks and want to provide models in TensorFlow to facilitate adoption and investigation.

From paper [6], License Plate Recognition (LPR) systems are becoming increasingly essential in a variety of applications, including traffic control, parking management, toll collecting, and security systems. This literature study focuses on the LPR system suggested in a recent publication, which detects and recognizes car license plates using popular open-source tools, OpenCV and PyTesseract.

The suggested LPR system transforms the input image to grayscale before using the bilateral filter method from OpenCV to blur the image while retaining the sharpness of the edges. This smoothed image is subsequently processed using OpenCV's Canny technique, which detects all of the image's edges. The find contours function is then used to locate all of the closed forms in the image. The 10 biggest forms with four edges are chosen from among them, and the rectangle shapes among them are cropped from the actual picture using the NumPy array slicing technique. The recovered rectangular shape is then

processed by PyTesseract's optical character recognition (OCR) technology to extract the characters from the license plate. These characters can be used for a variety of tasks, including detecting the car or its owner, registering the vehicle, locating the vehicle in a parking lot, and so forth.

The paper then shows that the proposed LPR system has demonstrated promising and efficient outcomes. In the future, the technology might be incorporated into parking lots where cameras record photos of cars and store the retrieved license plate numbers in a database. Furthermore, the technology may be used in toll plazas, where a live stream of passing cars is streamed into the system, and license plate numbers are retrieved and saved in a database for further searches. Finally, the suggested LPR system based on OpenCV and PyTesseract is a reliable and effective approach for detecting and identifying car license plates that may be used for a variety of applications. The system's functionality will be improved by using it in real-world circumstances in the future.

Recently, license plate detection and recognition have also been accomplished using OpenCV, a well-liked open-source computer vision library. Several image processing and computer vision algorithms are available in OpenCV that can be utilized for detecting and identifying license plates. Various works have used OpenCV's adaptive thresholding and morphological procedures for character segmentation and license plate region detection. Recent works have also used the optical character recognition (OCR) library EasyOCR to recognize license plates. EasyOCR offers a deep learning-based OCR engine that can accurately identify characters in photos. The pre-trained OCR models from EasyOCR have been utilized in particular works to recognize license plates.

In [10], the authors state that EasyOCR is the most user-friendly technique for extracting text from photographs which can improve the text extraction's accuracy and dependability. In paper [8], from the experimental results, it can be inferred that the EasyOCR gives better accuracy than Tesseract OCR for detecting and recognizing license plate images by the accuracy of Tesseract OCR is 62 %, whereas EasyOCR operates with an accuracy of 90%. Besides, the images with harsh conditions and with regional language license plates are excluded then EasyOCR shows a 95% accuracy, and Tesseract OCR shows 88%; we come to know that EasyOCR performs well with numerical and Tesseract OCR performs well with letters supported by [9]. These have proven that EasyOCR is more suitable than Tesseract OCR in this project. This work will use a combination of OpenCV and EasyOCR to perform license plate detection and recognition.

In paper [11], there were researchers who aimed to experiment with performing license plate recognition task specifically for Malaysian vehicles. In that paper, the algorithm for detecting the car number plate region is based on the AdaBoost algorithm and is validated using connected component analysis. KNN classifier is used to complete the task of character recognition. On static photos and video sequences, the proposed technique can obtain detection rates of 98% and 95%, respectively. The advantages of this method are low computational time, and feasible to be implemented in real-time. However, according to the authors, more work is necessary to be done as the algorithm is not robust enough towards poor images.

In paper [12], a segmentation-free model is proposed by incorporating CNN and LSTMs within its architecture. The proposed method achieves segmentation-free by utilizing the sequence labeling-based method
so that the method can recognize the whole license plate without character-level segmentation.

According to the research conducted in paper [13], an Automatic License Plate Detection and Recognition (ALPD-R) has been developed to challenge the application of deep learning and OCR towards traffic safety, security, and surveillance. An architecture namely, the fusion of multiple Faster Regions with Convolutional Neural Network (Faster-RCNN) was proposed. The proposed model included 3 Faster-RCNN modules which were the pre-trained model of AlexNet, VGG16, and VGG19. Each of the models were trained independently and the results from the output were fused.

The fusion algorithm proposed by the aforementioned researchers includes a fusing layer. The fusing layer employed the average value of the X and Y coordinates from the output of the 3 Faster-RCNN modules. A dataset which was retrieved from a publicly available repository was put into the experiment. The accuracy has been employed to serve as the metric for the performance of the proposed model. The proposed model managed to detect the precise coordinate of the license plates for 97 images out of 100 testing images. Thus, the proposed model had achieved an accuracy of 97%.

The paper [14] had proposed a license recognition plate model for automatic license plate recognition system (ALPRS) which played an important role in the area of Intelligent Transportation System (ITS). The proposed methodology involved the employment of a pre-trained CNN model known as VGG16. The input image will be put into the VGG16 model to extract low level features. The low level features will be aided to capture the region of interest(ROI) through integrating and pooling via transfer learning mechanism.

After the ROI of the license plate was detected, the feature of the license plate output was extracted and fed into the license plate detection network to extract the license plate as string. The proposed methodology for the ALPRS system achieved an accuracy of 97.13%.

The proposed model by researchers on paper [15] introduced a vehicle license plate detection pipeline using a region-based convolutional network. In this study, cutting-edge object identification techniques such as convolutional neural networks with region proposal

(RCNN), its successors (Fast-RCNN and Faster-RCNN), and the exemplar-SVM are employed to solve the problem.

There are 3 RCNN put into the experiment with the similar dataset. The dataset is divided into 4 different parts, namely visible license plate, partially visible license plate, angled license plate, and no license plate. The Selective search-VGG16 model outperformed all other models for achieving the accuracy of 96.03% in detecting the region of the license plate. The RPN-VGG16 (Region Proposal Network-VGG16) came into second place with the overall accuracy of 87.76%.

This research also shows the shortcomings of conventional image processing techniques, the probability of accurately predicting the license plate region on a visible license plate frame was only 71.76%. On the other hand, the partially visible license plate and angled license plate dataset performed way worse at 31.25% and 11.18% respectively compared to the Selective search-VGG16 model which performed significantly better at 94.37% and 87.57% respectively. This had brought down the overall accuracy of the conventional image processing techniques on detecting the region of the license plate to a staggering 64.67%.

From the study conducted by researchers in paper [16], a real-time automatic license plate recognition system using YOLOv4 was proposed. The proposed methodology had eliminated the necessity to capture the region of interest (ROI) of the license plate without affecting the performance of the prediction on the recognition of the license plate. The proposed methodology involved a real-time 1-stage Single Shot MultiBox Detector (SSD) which employed a VGG16 model as the base network because of the outstanding performance for image classification. Convolutional feature layers are added to gradually decrease the size which enables the model to predict objects at different scales by the aspect ratio. The YOLO model was trained to allow character recognition and it was trained with a 256*256 resolution to increase the speed of the training.

The KETI-ALPR dataset has been utilized for training the model. The KETI-ALPR dataset consists of over 2000 license plates that were collected in South Korea and the ground truth of the license plate for each license plate in the dataset. The YOLOv4 model had managed to achieve the accuracy of F1-score of 1 and IoU of 94.25% in extracting the characters of the license plate from the validation dataset.

## 3. APPROACH

There are four different approaches used to working on license plate detection and recognition.

### 3.1. LPRNet

LPRNet [7] is designed to overcome the problems faced by most of the deep neural network models that are implemented to perform license plate recognition tasks, which is the difficulty in running models on embedded devices. This is very challenging as the models proposed in the past works are heavyweight and unsuitable for such time-sensitive tasks. LPRNet is recognized as the first real-time license plate recognition model that does not incorporate Recurrent Neural Networks, which gives it advantages in computational complexity. In short, LPRNet is a real-time framework for high-quality license plate recognition that can be trained from scratch for various national license plates and supports template and character-independent variable-length license plates.

Moreover, LPRNet is a segmentation-free model. For context, generally automatic license plate recognition tasks are done by separating the task into two stages, which are plate segmentation and plate recognition stages. Plate segmentation involves extracting the region of interest that shows the license plate while plate recognition classifies the characters that appeared in the segmented image. However, this kind of pipeline has one shortcoming, which is that the character recognition relies heavily on the upstream stage and could result in a scenario where there are no characters being recognized if the segmented image outputted by the upstream stage does not contain characters.

#### *3.1.1 Model Architecture*

As mentioned above, LPRNet is aimed to be lightweight so those powerful networks whose backbone is GoogleNet, VGG or ResNet are not considered in the paper [7]. Instead, the authors have redesigned the backbone to be as lightweight as possible. The model architecture does follow the neural network training's best practices such as applying batch normalization to standardize the layer inputs in order to lower the number of training epochs needed to construct deep networks and stabilize the learning process. The backbone of the LPRNet is described in Table 1.

| Layer Type | Parameters |
|---|---|
| Input | 94x24 pixels RGB image |
| Convolution | #64 3x3 stride 1 |
| MaxPooling | #64 3x3 stride 1 |
| Small basic block | #128 3x3 stride 1 |
| MaxPooling | #64 3x3 stride (2, 1) |
| Small basic block | #256 3x3 stride 1 |
| Small basic block | #256 3x3 stride 1 |
| MaxPooling | #64 3x3 stride (2, 1) |
| Dropout | 0.5 ratio |
| Convolution | #256 4x1 stride 1 |
| Dropout | 0.5 ratio |
| Convolution | # class_number 1x13 stride 1 |

Table 1. The backbone network architecture of LPRNet

| Layer Type | Parameters/Dimensions |
|---|---|
| Input | $C_{in} \times H \times W$ feature map |
| Convolution | # $C_{out}/4$ 1x1 stride 1 |
| Convolution | # $C_{out}/4$ 3x1 strideh=1, padh=1 |
| Convolution | # $C_{out}/4$ 1x3 stridew=1, padw=1 |
| Convolution | # $C_{out}$ 1x1 stride 1 |
| Output | $C_{out} \times H \times W$ feature map |

Table 2. Small basic block of LPRNet

The network architecture of the LPRNet is explained as follows:

a) LPRNet consists of 3 main convolutional layers, 3 max-pooling layers, 3 small basic blocks which contain convolution layers and 2 dropout layers.
b) First of all, the backbone of LPRNet accepts an RGB image which is resized into 94x24 as input to reduce computation time
c) There are 3 main convolution layers used to calculate the spatially distributed rich features of the image. The first convolution layer consists of 64 kernel filters with a filter size of 3x3 with a stride of 1, the result of the output will be passed to a max pooling layer with a pool size of 3x3 with a stride of 1. The second convolution layer consists of 256 kernel filters with a filter size of 4x1 with a stride of 1. The last convolution layer is considered a wide convolution layer as it consists of 37 kernel filters with a filter size of 1x13 with a stride of 1, whereby 1 × 13 kernel is implemented to consider the local character context. Note that the number of kernel filters is 37, which is equal to the number of classes computed by summing the number of alphabets and numeric characters and a blank.
d) There are 3 small basic blocks located within the backbone architecture and its architecture is described in Table 2. The first small basic block is fed with parameters of 128 kernel filters with a filter size of 3x3 with a stride of 1, the result of the output will be passed to a max pooling layer with a pool size of 3x3 with a stride of (2,1). The second small basic block is fed with parameters of 256 kernel filters with a filter size of 3x3 with a stride of 1. The third small basic block is also fed with parameters of 256 kernel filters with a filter size of 3x3 with a stride of 1, the result of the output will be passed to a max pooling layer with a pool size of 3x3 with a stride of (2,1).
e) Loss function used here is Connectionist Temporal Classification (CTC) loss. LPRNet will take an RGB image as input and output a sequence of character probabilities. This sequence will tally with the input image pixel width. CTC loss is suitable for the case when the input and output sequences are not tallied and have variable lengths, which corresponds to the situation here where the ground truth character sequence and the decoder output of LPRNet lengths are of different lengths. The output from LPRNet and the ground truth characters are fed to CTC loss to guide the training. CTC loss is feasible as it will consider the position of characters by trying out all possible alignments of the ground truth characters in the image and sum up all the scores. If the value that sums up the alignment scores has a high value, then it indicates that the score of ground truth characters is high. The objective here is to train the LPRNet such that it outputs a high probability (ideally, a value of 1) for correct character recognition. Therefore, the model will maximize the product of probabilities of correct character recognitions during training.
f) To improve the performance, a global context embedding is calculated using a fully connected layer over the output of the backbone, tiled to the required size, and concatenated with the output of the backbone. This embedding will then be augmented with the pre-decoder intermediate feature map to enrich the spatial features.
g) When LPRNet has completed training, it will be used for recognizing the text in unseen images. During the inference stage, the model needs to calculate the most likely characters given the output of LPRNet. In this implementation, a path decoding algorithm called beam search is used to decode the output as it will maximize the total probability of the output sequence to ensure the result is as close as possible to the ground truth.

*3.1.2 Training Detail*

The model training is implemented and trained using TensorFlow.

The model is trained using Adam optimizer with a batch size of 8 as the dataset is small. Then, the learning rate is set as 0.001 and the learning rate will decay every 500 steps with a base of 0.995. The number of training epochs is set as 1000

*3.1.3 Fine-tuning of Model*

This section discusses experimenting with fine-tuning the performance of the model.

| Layer Type | Parameters | |
|---|---|---|
| Input | 94x24 pixels RGB image | |
| AvgPooling | #32 3x3 stride 2 | — |
| Convolution | #32 5x5 stride 3 | #32 5x5 stride 5 |
| Concatenation | channel-wise | |
| Dropout | 0.5 ratio | |
| FC | #32 with TanH activation | |
| FC | #6 with scaled TanH activation | |

Table 3. LocNet architecture on a license plate image

a) In the paper [7], the author proposes an optional step to preprocess the input image, which is applying the Spatial Transformer Layer whose architecture is described in Table 3. During experimentation, the application of LocNet is introduced but it causes the degradation of the performance of the model. As explained by the author, it could be because LocNet is not suitable for the first few iterations of the training as the first few iterations of training usually yield unreasonable gradients and models that are too weak.
b) To improve the computational efficiency and runtime performance, 2x2 strides are used for all max pooling layers. While it did improve the inference speed, the accuracy has dropped by 1.17% on average.

### 3.2. WPOD-net with MobileNets / Pytesseract

WPOD-net, also known as Wider Probability Distribution based Object Detection Network, is a deep learning-based object detection network that predicts object limits in images using a wider probability distribution. This strategy strengthens the model's resistance to ambiguity in object placement. Using a multi-stage architecture, WPOD-net extracts features from the input picture and predicts them using many layers of convolution and non-linear algorithms.

The model's use of a broader probability distribution has possible applications in fields such as driverless cars, surveillance cameras, and industrial automation. WPOD-net is a substantial development in the field of object identification by embracing a larger probability distribution and has the potential to increase the precision and effectiveness of detection systems for objects in real-world applications. To extract the plate segmentation, a WPOD-net pre-trained model with its purposed pipeline was used. The plate segmentation output will be given into the pre-trained MobileNets and Pytesseract models, which will extract the license plate as text.

MobileNets are a type of compact convolutional neural network (CNN) that is intended for use on smartphones and embedded devices. They employ depthwise separable convolutions, which divide a typical convolution operation into two independent operations, resulting in a network that is more computationally efficient. Despite their small size, MobileNets may perform well in image classifiers, making them a common choice for tasks like object identification, feature extraction, and instance segmentation. Transfer learning may be used to fine-tune MobileNets for specific tasks, in which a pre-trained MobileNet model is utilized as a preliminary step and then further trained on a new dataset. This enables the model to acquire task-specific features while exploiting data collected during the large-scale pre-training.

On the other hand, Pytesseract is a popular Python optical character recognition (OCR) program for extracting text from photos. The application gives a simple and intuitive interface to Google's open-source Tesseract OCR engine, which would be acclaimed for its accuracy and adaptability in recognizing text from documents. Customization options in Pytesseract include modifying the recognition threshold, allowing image pre-processing, and specifying the languages of the text to be identified. Pytesseract offers potential applications in fields like document digitization, text-based information retrieval, and data input automation, among others, due to its ease of use and adaptability. Pytesseract has become a helpful tool for a broad range of OCR applications as a commonly used OCR solution for Python.

#### *3.2.1 Data Preprocessing*

The pipeline first reads the image datasets to preprocess the data. To make sure that the images are accurately mapped to the ground truth while doing the evaluation, the image datasets are sorted in accordance. The color space for the image will then be changed from BGR to RGB. The picture is then divided by 255 to scale the pixel values between 0 and 1. In order to employ the pre-trained WPOD-net model for plate detection, the images were also scaled to 224*224 in size. The preprocessed image is then ready for the plate segmentation procedure.

#### *3.2.2 Plate Segmentation*

The WPOD-net model then does license plate detection using the pre-processed image as input. Each of the region suggestions produced by WPOD-net is a potential region for a license plate. These region suggestions are created by the WPOD-net model's region proposal networks (RPN) and are predicated on the characteristics that CNNs have been trained to recognize. The RPNs in WPOD-net produce region proposals and forecast the class label, either a license plate or the background, and position for each proposal using sliding windows and anchor boxes. The next step is to extract the plate region from the region suggestions that WPOD-net has generated. The license plate zones are separated by screening the ideas based on specific standards, such as their aspect ratio and size. In order to decrease the


Figure 1. Sample output of the cropped license plate images by WPOD-net pre-trained model

search space for the recognition phase and get rid of false detections, this step is essential. The top-k proposals are chosen as the final plate areas after the filtered proposals are sorted according to their confidence levels. Following the extraction of the plate region, the original image is cropped to serve as the input for the recognition stage. This phase is crucial because it separates the area containing the license plate from the remainder of the image, narrowing the search field for the recognition step and improving the system's overall accuracy. The figure above shows the sample output of the cropped plate images.

*3.3.3 Plate Segment Postprocessing*

Preprocessing is a crucial stage in the plate recognition process since it helps to boost the image quality and legibility of the text on the license plate. The image of the detected license plate is first resized and transformed to an 8-bit format, which helps to increase the image's contrast and enhance the legibility of the license plate text. Following this, the image is changed to grayscale, which streamlines subsequent image processing steps and improves the image's suitability for processing based on intensity levels. When a binary threshold is applied to the blurred image, it transforms into a binary image with black and white pixels. This step is essential because it makes the letters on the license plate easier to read. The text on the license plate is reduced in size while keeping its shape by applying morphological erosion on the binary image using a rectangular kernel. This step is essential because it makes the words on the license plate more visible and recognizable, which makes it simpler for later procedures to recognize the text. An additional method, stretching the license plate segments into a normalized aspect ratio, was also tested. In the following portion of the experiment, the outcomes of the suggested image-processing techniques will be thoroughly discussed.

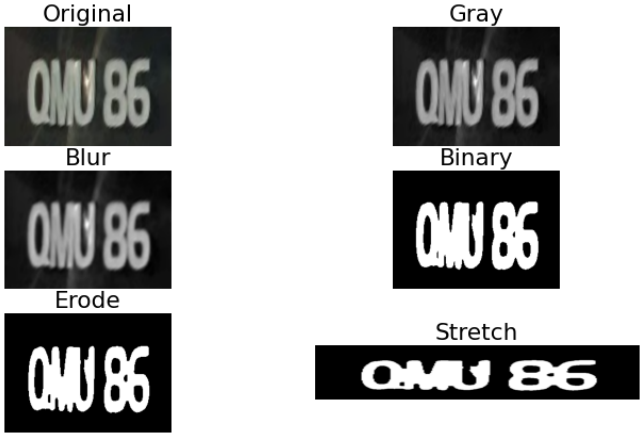


Figure 2. Visualization of the image processing techniques on a license plate image

*3.3.4 Plate Recognition*

MobileNets and Pytesseract are the two proposed plate recognition models for this section.

*3.3.4.1 Pipeline for MobileNets*

The contours in the post-processed plate image are first located in the MobileNets implementation for plate recognition. The identified contours are then compared to certain standards, including aspect ratio and height, to ascertain their likelihood of being characters. Following the identification of the contours that satisfy the stated criteria, the original plate image is cropped, and the contours are thresholded into a binary image. A pre-trained MobileNets model is then used to recognize this binary image. The model's output, which comes from the post-processing image, is a string of the predicted license plate text. The typical height and breadth of a character are set at 30x60 in order to increase recognition accuracy. To make sure they are handled in the proper sequence, the characters are also sorted by contours. The location of the character is established by locating the bounding box around each contour. Overall, the MobileNets plate recognition solution uses a combination of preprocessing methods and machine learning models to effectively detect and categorize the characters in a license plate image.

*3.3.4.2 Pipeline for Pytesseract*

The Pytesseract recognition pipeline starts with the post-processed plate image, which is then turned into a binary image format by scaling the grayscale image into 255 values aids in the recognition of characters. The binary image is then sent into the Pytesseract OCR engine, which can recognize text in images. Some unusual characters, such as I, O, and Z, are eliminated from the identification process in this pipeline since they cannot exist on a Malaysian license plate. The OCR engine's output is then post-processed to eliminate any undesired characters before returning the final result, which is the projected license plate text.

**3.3.VGG16 with Keras OCR**

VGG16 is a type of Convolution neural network (CNN) architecture that has won the runner-up of the 2014 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and only ranked behind GoogLeNet due to its high accuracy of prediction result. It is a model proposed by Andrew Zisserman and Karen Simonyan from the University of Oxford. The model was named VGG after the Visual Geometry Group department at the University of Oxford, where they worked. Transfer learning from the output of feature learning from VGG16 was applied to extract the plate segmentation. The output of the plate segmentation will be fed into the pre-trained Keras OCR model to extract the license plate as text. Keras OCR is an optical character recognition model that ignores case sensitivity, punctuations, and non-English text. It is also among the best pre-trained OCR models that provide high precision and recall rate within the mentioned condition.

*3.3.1 Data Preprocessing*

The image datasets along with the annotation files are read into the pipeline to perform data preprocessing. The image datasets and annotation files are sorted accordingly to ensure that the image and annotation details are correctly mapped. Furthermore, the images were resized to a dimension of 224*224 to be able to use the pre-trained VGG16 model for transfer learning. The annotations were transformed from Pascal VOC format into a Python flat list to be fed into the VGG16 model to predict the region of the plate segmentation. The image and annotation files are normalized before being fed into the model.

*3.3.2 Plate Segmentation*

**Model Description**

The architecture of the VGG16 neural network is explained as follow:

a) The VGG16 model comprises 13 convolutional layers, 5 max-pooling layers, and 3 fully connected layers. Consequently, there will be 16 layers that consist of tunable parameters which are the 13 convolutional layers and 3 fully connected layers.
b) The input of the VGG16 model required an RGB image with the height and width of 224. The mean of the RGB value is computed for all the images in the training set. Subsequently, the computed image serves as the input for the model.
c) The first two convolutional layers consist of 64 features kernel filters with the filter size of 3x3. As the input image passed through the layers, the dimension of the result of the output will be 224*224*64 and will be passed to the max pooling layer with a stride of 2.
d) The third and fourth convolutional layers consist of 128 features kernel filters with the filter size of 3x3. As the input image passed through the layers, the dimension of the result of the output will be 112*112*128 and will be passed to the max pooling layer with a stride of 2.
e) The fifth, sixth, and seventh convolutional layers consist of 256 feature kernel filters with the filter size of 3x3. As the input image passed through the layers, the dimension of the result of the output will be 56*56*256 and will be passed to the max pooling layer with a stride of 2.
f) The eighth to thirteenth layers are two pieces of convolutional layers consisting of 512 feature kernel filters with the filter size of 3x3. As the input image passes through those layers, it will be passed to the max pooling layer with a stride of 2.
g) The first 3 of the last 4 layers are the fully connected hidden layers of 4096 units to be utilized

for image classification. The last layer is a max pooling layer with a stride of 1 and softmax activation function with the output of 1000 units indicates it could be classified into 1000 categories of image which was fed with the ImageNet dataset.

Below depicts the architecture of the VGG16 model:

| **Block** | **Description** | **Output Dimension** | **Kernel Size** | **Stride** | **Activation** |
|---|---|---|---|---|---|
| Input | Scaled image | 224*224*3 | - | - | - |
| 1 | 2*Conv64<br>Max Pooling | 224*224*64<br>112*112*64 | 3*3<br>3*3 | 1<br>2 | ReLU<br>ReLU |
| 2 | 2*Conv128<br>Max Pooling | 112*112*128<br>56*56*128 | 3*3<br>3*3 | 1<br>2 | ReLU<br>ReLU |
| 3 | 3*Conv256<br>Max Pooling | 56*56*256<br>28*28*256 | 3*3<br>3*3 | 1<br>2 | ReLU<br>ReLU |
| 4 | 3*Conv512<br>Max Pooling | 28*28*512<br>14*14*512 | 3*3<br>3*3 | 1<br>2 | ReLU<br>ReLU |
| 5 | 3*Conv512<br>Max Pooling | 14*14*512<br>7*7*512 | 3*3<br>3*3 | 1<br>2 | ReLU<br>ReLU |
| 6 | 3*FC4096 | 4096 | - | - | ReLU |
| Output | FC | 1000 | - | - | Softmax |

Table 4. The summary of the layout of VGG16 architecture

**Model Construction**

A pre-trained VGG16 model was loaded without the last 4 layers in order to conduct transfer learning. The input shape of the VGG16 model was configured as (224, 224, 3) in order to use the existing pre-trained model from the ImageNet dataset. Aside from this, the learning ability of every layer in the model was freezed to speed up the training process. A sequential model was constructed and the aforementioned pre-trained VGG16 model was appended into the sequential model as the base model. Consequently, a flatten layer was appended to the sequential model in order to be served as the input for the fully connected hidden layers.

There are 3 layers of hidden layers along with 1 output layer after the base model for the proposed model. The first two hidden layers are dense layers with 128 units and ReLU was selected as the activation function. The last hidden layer is a dense layer with 64 units and ReLU has been applied as the activation function for the layer. The output later is a dense layer with 4 units and Sigmoid has been picked as the activation function.

To summarize the purpose of the model, the model will receive the resized input image as the input and return the predicted annotation for the plate segmentation which has been normalized as the output. The output segmentation will need to be rescaled before applying it to the original image.

**Model Fitting**

Before the model was being fitted, the model was compiled by using the mean squared error function as the loss function and accuracy has been employed as the metric to evaluate the model. Adam Optimizer has been chosen as the optimization technique for gradient descent for the deep learning model.

The model has been fitted with the training set and validated with the validation set for 40 epochs. The batch size for the model was set to 32 due to the limitation of the memory of the GPU in Google Colab. The accuracy and loss of each epoch of model was plotted for visualization of the model performance.

**Fine Tuning of Model**

The fine tuned model was constructed for comparison in order to deploy the best available model between the untuned and tuned model. The fine tuning process involved unfreezing the last block, the VGG16 model, in which the layers are depicted on the Block 6 of the table N. The unfreeze layers will get their weights updated in every epoch during the model fitting process.

Similar dataset was fitted into the model and a similar amount of epochs were run. The accuracy and loss of each epoch of model was plotted for visualization of the model performance in order to compare with the untuned model.

*3.3.3 Plate Segment Postprocessing*

Figure 3. Grountruth and Predicted output denoted as red box and green box respectively

After the segmentation of the plate, the output of the segmentation was rescaled back to the resolution of the original size for the image. Due to the terrible accuracy of the VGG16 model, the region of interest is expanded by 10% based on the size of the original image in order to ensure the ROI on most of the images included the car plate as part of the segmentation region.

Aside from this, a Gaussian blurring had been applied to the image for smoothing the characters on the plate in order to achieve better accuracy on the OCR plate recognition model.

*3.3.4 OCR Plate Recognition*

**Model Description**

The Keras OCR model which was a lightly packaged and polished version of the implementation of the Keras CRNN and based on the published CRAFT text detection model. The Keras OCR model also provided an application programming interface in Python which is easy to apply with high accuracy.

The CRNN recognition model was trained with the Synth90k dataset. Aside from the CRNN recognition model, the CRAFT text detection model was trained with the SynthText, IC15 dataset for the English language model.

There are a few limitations with the model but those limitations are not affecting the requirement of plate recognition. For instance, the model had ignored the case sensitivity as well as the punctuations. Other than that, the model had also ignored non-English alphabets and illegible text. This will not affect the prediction accuracy as the license plate in Malaysia is not case sensitive.

**Model Implementation**

The Keras OCR model is implemented. The processed image of the plate segmentation was fed into the model. The output from the model was further processed to handle special characteristics in which Malaysia car plates do not exist.

For instance, due to the height issue of the car plate as most of the photos were not captured in an exact straight manner, the number might be recognized retrieved earlier than the alphabets of the plate. For example, the plate "ABC1234" might be misrecognized as "1234ABC". Therefore, the process of swapping the alphabets and numbers was conducted to improve the accuracy of the prediction.

Other than that, there was also an issue with the letter "Q" being misrecognized as "O". Since there is no letter "O" in the standard Malaysia car plate. Consequently, the letter "O" has been replaced with the letter "Q" before the end of the pipeline.

Lastly, the final result after the postprocessing of the predicted output for the license plate text was returned.

**3.4. OpenCV and EasyOCR**

EasyOCR is built on top of the Tesseract OCR engine, which is an open-source library for OCR that has been widely used in the OCR community. The library provides a convenient way for users to perform OCR on images with just a few lines of code. The library can recognize text in multiple languages and on various types of images, including color and grayscale images. OpenCV, on the other hand, is a popular open-source computer vision library for image and video processing. It provides a wide range of functionalities for image processing, including image filtering, feature detection, and image segmentation. In the context of OCR, OpenCV can be used to perform image pre-processing and segmentation tasks, such as thresholding, contour detection, and morphological operations, to improve the accuracy of the OCR results.

*3.4.1 Data Preprocessing*

Pre-process the input image is done by converting the input image from the RGB color space to the grayscale color space. This is followed by an adaptive thresholding operation, which helps to improve the quality of the license plate region by removing the background noise.

*3.4.2 Plate Segmentation*

For segmenting the license plate region from the input image, this is done by performing a morphological opening operation on the thresholded image. This helps to remove small noise and fill in gaps in the license plate region. The resulting image is then processed to detect rectangular contours, which correspond to the candidate license plate regions. In cases where the license plate number is not recognized correctly, or if the license plate region could not be segmented, the code falls back to recognizing the license plate number directly from the grayscale image of the car.

*3.4.3 Plate Segment Post-processing*

Next, to perform post-processing on the candidate license plate regions to ensure that only the best candidate is selected as the final license plate region. This is done by determining the size of the candidate license plate regions and selecting the region with the largest size lead to minimize the impact of false detections and improve the accuracy of the license plate recognition process.

*3.4.4 Pipeline for EasyOCR*

For performing EasyOCR on the selected license plate region, this is done using the EasyOCR library, which provides a highly accurate OCR model. The OCR process involves detecting the text in the candidate license plate

region, recognizing the text using the built-in OCR model, and returning the recognized text as the output of the OCR process. The EasyOCR library is fed the segmented license plate region, and it returns the text that it recognizes in the region. The recognized text is then post-processed to extract the license plate number. This is done by using a regular expression (regex) to match the license plate number format.

## 4. EXPERIMENT

### 4.1 Dataset Detail

#### *4.1.1 Data Collection*

The collection of 400 Malaysian license plates was assembled by hand by several members. Each member was responsible for capturing 100 plates. To ensure diversity in the dataset, each member attempted to capture images of plates from different states in Malaysia, as well as a wide range of aspect ratios, i.e., the proportions of the plates from width to height. To account for different lighting situations, the collected photos were divided into three categories: bright light, out-of-focus, and low light.

This procedure was performed to obtain a diverse and representative sample of Malaysian license plates, which can be used to train various models proposed by the group members for the purpose of license plate recognition. Aside from this, including a variety of aspect ratios in the dataset can help improve the model's ability to generalize and recognize license plates with different proportions and dimensions.

#### *4.1.2 Data Annotation*

Annotations are critical in assuring the accuracy and reliability of machine learning models. Bounding Box Annotation and Image Classification Annotation are the two types of annotations used to help and verify model training and validation. These annotations help models to learn and recognize objects in photos by identifying and classifying data.

In the case of license plate recognition, the license plate region of interest (ROI) in an image is annotated using the “labelImg” software written in Python. This program provides a user-friendly interface for manual image annotation, allowing the development of bounding box annotations for each image in the dataset. Annotations are encoded in the Pascal VOC format, which is a popular standard for image recognition annotations.

The bounding box annotations provide information about the coordinate and size of the license plate in the image, which can then be used to train the license plate recognition models. These models can then be validated and tested against the annotations to assess their accuracy and reliability. This information can be used to fine-tune the models, resulting in improved performance and accuracy.

In addition to the bounding box annotations, a ground truth file in JSON format was created to evaluate the accuracy of the license plate recognition model's character recognition. This file contains the license plate number of each license plate image in the dataset against which the model can compare its predictions. The ground truth file serves as the gold standard for measuring the character recognition accuracy of the license plate recognition model.

The ground truth file is crucial for ensuring that the license plate recognition model reads the characters in the license plate pictures correctly. The model's predictions may be compared to the actual data, and any errors can be found and corrected. This procedure is continued until the model's accuracy in character recognition achieves the specified threshold.

In summary, the ground truth file in JSON format is an important tool for measuring the quality of the character recognition component of the license plate recognition model. This data can be used to improve the performance of the model to ensure that it reads license plates correctly under real-world conditions.

#### *4.1.3 Training and Validation Data Splitting*

Deep learning-based license plate recognition models include the LPRNet model and the VGG16 with Keras OCR model. To properly train these models, the dataset must be separated into two parts, namely the training set and the validation set. The purpose of this separation is to allow the models to learn from the training data before validating their predictions on the validation data.

For these models, the data set was divided into 360 images for the training set and 40 images for the validation set. The training set is used to train the model to recognize patterns and features in the data, while the validation set is used to evaluate the model's performance on unseen data. This ensures that the model does not overfit to the training data and that it can effectively generalize to unseen data.

### 4.2 Result of the Experiments

For evaluating the performance of plate recognition, this project uses Accuracy, Average Character Level Accuracy and Average Levenshtein Distance as evaluation metrics. The metrics are explained as follows:

a) Accuracy: For context, the output generated by the algorithm is only considered accurate when the outputted license plate’s characters are completely equivalent to the respective ground truth label provided. It is a standard evaluation metric that measures the percentage of license plates that are correctly recognized by the algorithm. It gives a

simple and straightforward understanding of the overall performance of the system.

b) Average Character Level Accuracy: Character level accuracy is computed by dividing the number of correctly recognized characters per position by the number of ground truth characters. This metric measures the average accuracy of recognizing each individual character on the license plate. It enables us to see in more fine detail that the character's output is not completely wrong.

c) Average Levenshtein Distance: This metric measures the average edit distance between the predicted license plate number and the ground truth. The Levenshtein Distance is a measure of how different two strings are, and it is often used to evaluate the performance of optical character recognition (OCR) systems. In the context of license plate recognition, it provides a more nuanced understanding of the performance of the system, as it takes into account the proximity of the predicted license plate to the actual license plate, even if it's not an exact match.

Furthermore, this project also evaluates different approaches on the whole dataset and different segments of the dataset to check how the various approaches perform in specific conditions such as images with bright light, images with dim light, and blurred images. This helps us to evaluate how robust the implemented approaches are towards varying conditions. It is very crucial for real-world applications as the images in real-world applications are not always of high quality and might include noises from the actual environment and the devices which capture the license plates.

| **All 400 images dataset** | **Accuracy** | **Average Character Level Accuracy** | **Average Levenshtein Distance** |
|---|---|---|---|
| **LPRNet** | 78.50% | 87.89% | 0.7200 |
| **WPOD-net + MobileNets** | 16.25% | 39.21% | 3.6675 |
| **WPOD-net + Pytesseract** | 15.25% | 38.9% | 3.6875 |
| **VGG16 + Keras OCR (10% expansion)** | 41.75% | 63.21% | 2.4275 |
| **OpenCV + EasyOCR** | 28.00% | 41.94% | 3.1950 |

Table 5. Result of each model pipeline for the whole dataset of 400 images

| **Images with bright light condition** | **Accuracy** | **Average Character Level Accuracy** | **Average Levenshtein Distance** |
|---|---|---|---|
| **LPRNet** | 78.24% | 87.54% | 0.7265 |
| **WPOD-net + MobileNets** | 17.65% | 42.97% | 3.4118 |
| **WPOD-net + Pytesseract** | 17.35% | 41.81% | 3.4176 |
| **VGG16 + Keras OCR (10% expansion)** | 42.35% | 65.25% | 2.3294 |
| **OpenCV + EasyOCR** | 26.47% | 38.86% | 3.4290 |

Table 6. Result of each model pipeline for the images with bright light condition

| Images with dim light condition | Accuracy | Average Character Level Accuracy | Average Levenshtein Distance |
|---|---|---|---|
| **LPRNet** | 80.00% | 89.84% | 0.6833 |
| **WPOD-net + MobileNets** | 8.33% | 17.95% | 5.1166 |
| **WPOD-net + Pytesseract** | 3.33% | 22.41% | 5.2167 |
| **VGG16 + Keras OCR (10% expansion)** | 38.33% | 51.67% | 2.9833 |
| **OpenCV + EasyOCR** | 20.00% | 57.13% | 2.2666 |

Table 7. Result of each model pipeline for the images with dim light condition

| Blurred images | Accuracy | Average Character Level Accuracy | Average Levenshtein Distance: |
|---|---|---|---|
| **LPRNet** | 72.73% | 86.58% | 0.7878 |
| **WPOD-net + MobileNets** | 0.0% | 11.26% | 5.6969 |
| **WPOD-net + Pytesseract** | 9.09% | 19.75% | 5.0909 |
| **VGG16 + Keras OCR (10% expansion)** | 3.03% | 33.69% | 4.1212 |
| **OpenCV + EasyOCR** | 36.67% | 59.35% | 1.8667 |

Table 8. Result of each model pipeline for the blurred images

The ROI (Region of Interest), which gauges how accurately the segmentation model locates the license plate inside the image, is crucial for the evaluation of plate segmentation. The performance of the segmentation model can be deemed satisfactory if the ROI it produces correctly encompasses the license plate. For evaluating the performance of plate segmentation, the produced license plate ROI is contrasted with a ground truth that is manually annotated on each of the 400 images in order to assess the effectiveness of various detection techniques. The average error metric calculates the average locational difference between the produced ROI and the ground truth. A segmentation model's overall performance across several images is assessed using average error, average accuracy, average recall, and average IoU. These metrics give a consolidated perspective of the model's performance rather than focusing on specific image findings by averaging the outcomes. The justification for each metric is as follows:

a) Average error: The average error is a measurement of the average locational difference between the produced ROI and the ground truth. The location of the license plate is accurately represented by the ground truth ROI, which is manually labeled on each image. On the other side, the segmentation model's output is the created ROI. The Euclidean distances between the centers of the created ROI and the ground truth are averaged to determine the average error. A smaller average error demonstrates improved license plate location accuracy for the model.

b) Average precision: The accuracy of the model's detection of the license plate is quantified by its average precision. It calculates the ratio of true positive detections, or accurately detected license plates, to all positive detections, including false positives, which are all detected license plates. The average precision ratings over several images is used to compute average precision. A high average accuracy score means the model detects false positives infrequently.

c) Average recall: The model's average recall serves as a gauge of how accurately it detected the license plate. It calculates the percentage of true positive detections against all the true positive instances or all valid license plates in the images. The average of memory ratings across numerous images is used to compute average recall. A high average recall score denotes a high rate of true positive detections in the model.

d) Average IoU: The ground truth and produced ROI overlap is measured by the average IoU. The amount that the computed ROI encompasses the license plate as shown by the ground truth ROI is measured. The average of the IoU scores across many images is used to compute the average IoU. An improved ability to find the license plate inside the image is indicated by a model with a higher average IoU score.

| All 400 images dataset | Average Error | Average Precision | Average Recall | Average IoU |
|---|---|---|---|---|
| **WPOD-net** | 41.51% | 74.29% | 50.41% | 49.39% |
| **VGG16** | 20.77% | 75.38% | 84.50% | 68.51% |

Table 9. Result of each detection model in ROI evaluation for the whole dataset of 400 images

Due to the fact that isolating the detector errors is important in order to evaluate the actual performance of the recognition. This is accomplished by building the dataset with the highest recall. The proportion of real positive cases that are accurately labeled as positive is referred to as recall. This approach can reduce the influence of detector mistakes and better evaluate the performance of the recognition section of the pipeline by choosing a dataset with the highest recall. This aids in understanding the recognition model's limits and identifying opportunities for improvement. The examination was restricted to three pipelines with a plate detection method. This implies that the metrics were only assessed for these three pipelines and not for any other models or pipelines included in the research. This focus on the three pipelines allowed for a more in-depth investigation of the plate detection performance, as well as the isolation of any faults produced during the detection step.

| Isolate detection error dataset | Accuracy | Average Character Level Accuracy | Average Levenshtein Distance: |
|---|---|---|---|
| **WPOD-net + MobileNets** | 38.33% | 70.28% | 1.7166 |
| **WPOD-net + Pytesseract** | 30.0% | 68.98% | 1.8166 |
| **VGG16 + Keras OCR (10% expansion)** | 48.30% | 67.02% | 2.1667 |

Table 10. Result of each model pipeline for the isolate detection error dataset

The license plate dataset obtained by the annotation will then be used to test each of the recognition models, but only for the pipelines that contain detectors. Here are the outcomes.

| Annotated dataset | Accuracy | Average Character Level Accuracy | Average Levenshtein Distance: |
|---|---|---|---|
| **MobileNets** | 0.0% | 0.05% | 6.825 |
| **Pytesseract** | 38.5% | 56.05% | 2.5075 |
| **Keras OCR** | 43.75% | 75.40% | 3.6025 |

Table 11. Result of each OCR model pipeline for the manually annotated dataset

Table 11 shows that MobileNets' accuracy is a dismal 0%. This is mostly due to the fact that the annotation was manually cropped into a rectangular shape, which makes it incompatible with the pre-trained MobileNets model. Instead of the present rectangular box, the input for the model is often a polygonal segmented license plate that was realigned. The annotated dataset's characters could not be detected by the MobileNets contour processing, which is the major cause of the model's poor output metrics. Both Pytesseract and Keras OCR are OCR models made to extract text from images. In contrast to Pytesseract, a well-known OCR application that uses the Tesseract OCR engine, KerasOCR is a deep learning-based OCR tool that recognizes text using a Convolutional Neural Network (CNN). Thus, the outcomes for both text recognition algorithms were anticipated because they are reasonably reliable pre-trained models.


Figure 4. Example of an annotation

### 4.3 Result Analysis

*4.3.1 Result Analysis for LPRNet*

While working on error analysis for the result of LPRNet, there is a fact found that the model has recognized correctly on the majority of the training set but none of the images in

the validation set is recognized correctly. This is obviously due to the overfitting problem that causes the model to not be able to generalize on the data other than the training set.

Next, since LPRNet is a segmentation-free model and trained using an end-to-end pipeline, therefore there is no image to be shown in the intermediate steps. The model is aimed to optimize the CTC loss which is computed by the decoder output of the model and the ground truth characters by learning the non-linear relationship between them.

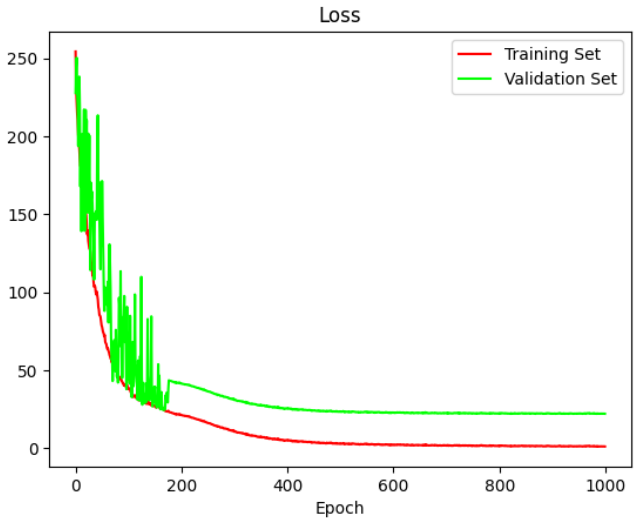


Figure 5. The trend of train loss and validation loss of LPRNet

The figure above shows the trend of train loss and validation loss of the LPRNet. It is clearly shown that the training loss is continuously decreasing, which means that the model is learning the relationship between input and output well during the learning process. However, the fluctuation of validation loss indicates that the model faces difficulties when generalizing on unseen data.

*4.3.2 Result Analysis for WPOD-net with MobileNets / Pytesseract*

The WPOD-net pipeline research revealed that both MobileNets and Pytesseract have low accuracy in identifying Malaysian license plates. This might be due to a variety of variables such as the model's architecture, lighting and background conditions, and the angle and aspect ratio of the images captured. The architecture of the model being utilized is critical in determining accuracy. MobileNets and Pytesseract were utilized for recognition in this scenario, however, if the design of these models was not well adapted to the task of license plate recognition, this might result in low accuracy.

First of all, MobileNets is a deep learning model built for quick and easy recognition tasks. However, MobileNets may struggle to recognize Malaysian license plates for a variety of reasons. The contour problem is one of the difficulties. Malaysian license plates frequently have complicated curves and forms that may not be effectively represented by the MobileNets model. This might make locating the license plate area of interest (ROI) and recognizing the characters more challenging. Another consideration is the characters' aspect ratio. The aspect ratio of the characters on a license plate, which is the relationship between the characters' height and width, can also have an effect on the MobileNets model's performance. If the aspect ratio of the characters on the license plate images differs greatly from the aspect ratio of the characters in the training data, the model may struggle to correctly recognize the characters.

Unfortunately, Malaysian license plates feature a different aspect ratio than European license plates, with characters that are somewhat thicker. Because of the aspect ratio discrepancy, license plate recognition algorithms such as MobileNets and Pytesseract may have problems effectively identifying the license plate ROI and recognizing the characters. This is due to the fact that the models are trained on a specific aspect ratio, and a considerable variation from that aspect ratio might damage the recognition's accuracy. Furthermore, the curve of the letters on Malaysian license plates may pose recognition problems for these algorithms. All of these issues combined to reduce the MobileNets pipeline's ability to detect character contours.

Additional processes, such as attempting to stretch poorly segmented license plates into a normalized aspect ratio before going into the contours, were explored to increase the accuracy of the whole MobileNets pipeline, but the results were unsatisfactory. According to the figures below, forcing the WPOD-net pipeline to generate the input image as a one-line plate or changing the aspect ratio of the segmented plate image does not increase the accuracy of detecting the contour, but rather completely eliminates the existing contours and significantly reduces its accuracy. This is due to the fact that the aspect ratio normalization procedure may not adequately capture the true form and aspect of the license plate, which is required for reliable contour detection. If the aspect ratio normalization method is faulty, existing contours may be eliminated, resulting in a considerable drop in accuracy. The aspect ratio normalization procedure must be carefully developed to capture the correct aspect ratio of the license plate while keeping the curves and forms of the letters. If the license plate image is altered in any way during the normalization process, crucial data required for precise contour detection may be lost. As a result, forced aspect ratio normalization may not always increase, but rather dramatically impair contour identification accuracy.


Figure 6. Results for QMU86


Figure 7. Results for QMT58

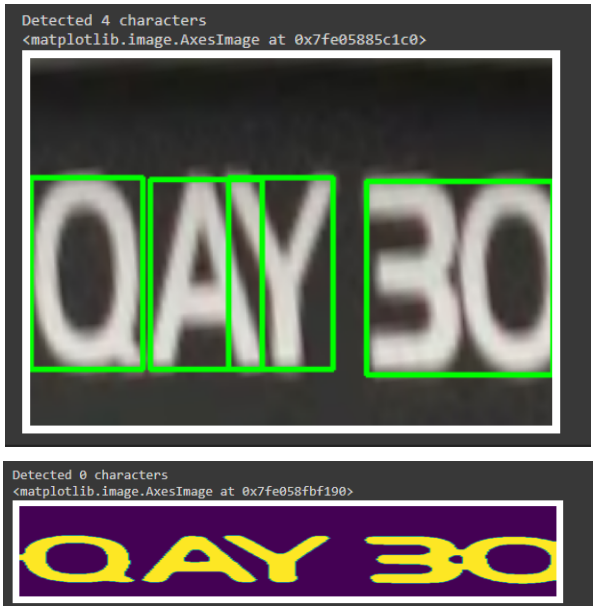


Figure 8. Results for QAY30

Because of the limits in modifying the pre-trained MobileNets pipeline, model accuracy may decrease. This is due to the fact that pre-trained models are created and optimized for a specific task, and the model's weights are fine-tuned to it. Any modifications to the model's design or parameters might have a detrimental influence on its performance since the model may not be able to adapt to the new changes efficiently. As a result, without sufficient understanding and competence in updating the MobileNets pipeline, the modifications made are likely to reduce its accuracy. As a result, the decision is taken not to include the additional process in the final pipeline for both the MobileNets and Pytesseract recognition models since it will drastically reduce accuracy and will not aid in accurately identifying or recognizing the plates.

On the other hand, Pytesseract is a character recognition library that is well-known for its optical character recognition (OCR) engine designed primarily for text recognition in documents. It was created to handle a broad variety of font styles and languages, making it a popular choice for text recognition tasks. However, because of the variety of font styles used in Malaysian license plates, Pytesseract may fail to attain high accuracy when detecting Malaysian license plates. As mentioned earlier, Malaysian license plates frequently utilize a different font style than European license plates, making it difficult for Pytesseract to detect the characters effectively. This is due to the fact that the font type used on Malaysian license plates may be thicker and have a different aspect ratio than the font styles used on European license plates, on which Pytesseract was trained. Furthermore, the variety of font styles used on Malaysian license plates might make Pytesseract difficult to generalize and properly detect the letters.

The angle and aspect ratio of the images captured can also have an impact on the recognition pipeline's accuracy. If the license plate images are captured at an angle, the model may struggle to identify the license plate ROI and recognize the characters. Similarly, if the aspect ratio of the license plate images differs greatly from the aspect ratio of the license plates used in the training data, the model may struggle to reliably recognize the license plate ROI and characters. The notion that angle and aspect ratio may have an effect on the accuracy of plate recognition pipelines is demonstrated by testing the pipelines on a dataset that reduces the effects of angle and aspect ratios by isolating previous errors from the detector. According to the results in Tables 10 and 5, the accuracy of the MobileNets and Pytesseract pipelines has significantly increased. For MobileNets, the accuracy has grown from 16.25% to 38.33%, and for Pytesseract, it has gone from 15.25% to 30%, which is a huge improvement. Likewise, it appears from the findings in the tables that the MobileNets and Pytesseract pipelines had trouble processing input images with poor angles and aspect ratios.

Furthermore, the lighting and background conditions in the images might have a major influence on the pipeline's accuracy. In both bright and dim lighting conditions, MobileNets and Pytesseract pipeline accuracy is quite poor, as shown in Tables 6 and 7. For example, MobileNets only achieves 17.65% for bright light condition images and 8.33% for dim light condition images. Only 17.35% of Pytesseract images are accurate in strong light, whereas only 3.33% are accurate in poor light. This is probably due to the fact that the models were developed using images captured in ideal lighting circumstances, and as a consequence, they might not be able to recognize license plates in varying lighting situations. The disparity between the appearance of the license plates in these situations and the ideal lighting settings used for training is probably what causes the low accuracy in bright and dim lighting situations. The variations in brightness, contrast, and color in the images may make it difficult for the model to identify the license plate, resulting in a lower accuracy rating.



Figure 9. Example of bright light condition image


Figure 10. Example of dim light condition image

Besides that, due to the fact that MobileNets and PyTesseract depend on the sharpness and clarity of the picture to identify and recognize the license plate characters, hence blurry images can significantly affect the WPOD-net pipelines' accuracy. It is challenging for the model to precisely detect the contour of the license plate and identify the letters when an image is hazy because the features of the license plate get distorted. Because of this, the accuracy of the Pytesseract pipeline was only 9.09%, whereas the accuracy of the MobileNets pipeline was 0% with an average character level accuracy of 11.26%. The algorithms' failure to successfully interpret hazy images and extract the information required to generate precise predictions is the cause of their low accuracy.



Figure 11. Example of blur image

Last but not least, the ROI assessment findings for the WPOD-net detection model are potentially biased because the ground truth was hand generated and may be inaccurate. Additionally, the license plate ROI's form might be altered in real-world images owing to angle problems, when the ground truth was rectangular in shape. Because the metrics may not accurately reflect the model's actual performance, the WPOD-net detection model's performance may really be better than what they suggest. The average error should be lower than 41.51% based on Table 9. Due to the angle at which the photo of the automobile was taken, Figure 11 below provides an example of a license plate that is not a rectangle.

Figure 12. Example of polygonal segmentation from WPOD-net detection model

*4.3.3 Result Analysis for VGG16 with Keras OCR*

The VGG16 model was found to have low accuracy in segmenting the license plate coordinates in vehicle images. This is most likely due to overfitting, a typical problem in deep learning. Overfitting occurs when a model is trained too many times on a small data set and learns the noise and idiosyncrasies in the data rather than the underlying patterns. As a result, the model's generalization performance suffers and it is unable to make accurate predictions on new and previously unseen data.

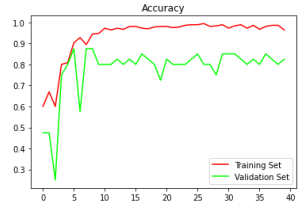


Figure 13. Accuracy before model tuning

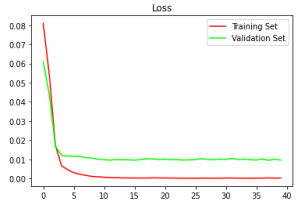


Figure 14. Loss before model tuning

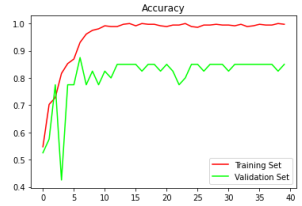


Figure 15. Accuracy after model tuning

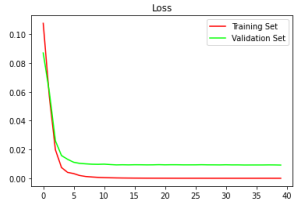


Figure 16. Loss after model tuning

The limited amount of training dataset in the case of the VGG16 license plate recognition model is most likely a contributing factor to its low accuracy. Because the model was trained on a small number of samples, it learned the specific features of the training images rather than the more general features of license plates. When applied to real-world images, which may contain differences in the appearance of license plates that the model did not see during training, this results in poor performance.

From the 4 figures above, it had shown that regardless of the tuning of the VGG16 model, the model had failed to generalize the coordinates of the segmentation of the license plate. Other than that, it also shows that the learning process of the tuned and untuned model is also looking identical due to the limited amount of dataset being fed to the model.

Figure 17. Plate Segmentation Before Fine Tuning

Figure 18. Plate Segmentation After Fine Tuning

In addition to the limitations of the VGG16 model mentioned above, it is important to evaluate the influence of

external factors on the prediction of license plate segmentation. Weather is one such factor that can have a significant impact on the appearance of license plates. For example, rain or foggy can make the plate appear blurry than its usual condition. On the other hand, intense sunshine towards the plate might cause reflection that make the plate harder to identify.

However, as seen in the previous two figures, the weather does not seem to have a significant influence on the prediction of the license plate segmentation on the VGG16 model. This implies that the model can cope with the variations in plate appearance caused by weather changes.

Aside from this, despite the effort made to improve the model performance by fine-tuning the last block of the VGG16, the accuracy of the model was still barely improved. This is also another suggestion that the model may be overfitting due to the limited amount of the dataset. A reasonable solution would be to collect more data and retrain the model in order for the model to generalize it better by actually learning the features of the license plate.

Other than that, there is also a significant noticeable limitation on the license plate segmentation pipeline of this model. For instance, another pre-trained model such as WPOD-net proposed by another group member has the ability to extract the license plate in a non-rectangular segmentation. This is generally helpful as the license plates from the dataset are not regularly shaped as a rectangle. Therefore, a segmentation model which allows polygon segmentation of the license plate would be a much more ideal solution than a conventional segmentation using a rectangular bounding box.

On the other hand, the pre-trained Keras-OCR model depicts a significant advantage over its competitors such as Pytesseract and Easy OCR proposed by other group members in terms of recognizing and extracting the string from the image of the segmented license plate.

This is because the Keras OCR model had ignored case sensitivity and non-English alphabets compared to the other models available in this project. Consequently, the license plate in Malaysia is not case sensitive as the uppercase alphabets were always standard used by all the license plates nationwide.

The limitations of the Keras OCR model came in as the letter “Q” in the license plate is very often being detected as “O”. A post-processing function has resolved the issue by replacing all the letter “O” from the output generated by the Keras OCR model into letter “Q”.

*4.3.4 Result Analysis for OpenCV and EasyOCR*

Several factors may impact how well computer vision algorithms recognize license plate numbers. In this project, OpenCV and EasyOCR's poor performance in identifying Malaysian license plates can be attributable to a mix of OCR process constraints and image segmentation problems.

The license plate is separated from the backdrop and other unimportant areas of the image during the image segmentation procedure, which is a vital step in license plate detection. When the license plate is partially concealed, or there is a lot of noise in the image, OpenCV's ability to accurately segment license plates is hampered. This may lead to less-than-ideal image quality, which will have a negative impact on how well the OCR process works.

On the other hand, the OCR models used by EasyOCR are deep learning-based and were trained on a sizable corpus of text images. However, these models might have trouble correctly identifying characters in photos of license plates, particularly if the plate is dimly illuminated, blurry, or has an irregular font size. In addition, the quality of the pre-processed image, which is mainly dependent on the accuracy of the image segmentation step, can affect how well the OCR algorithm performs.

The current approach needs help in accurately detecting license plates, as seen by the low accuracy of 28% in identifying Malaysian license plates, as well as the average character level accuracy of 41.94% and the average Levenshtein distance of 3.195. To boost the precision of license plate identification and recognition, additional advancements in picture segmentation and OCR processes are necessary.

In conclusion, poor image segmentation and OCR process constraints can be attributed to OpenCV and EasyOCR's poor accuracy in identifying Malaysian license plates. More study is needed to address these issues and raise the precision of license plate detection and recognition.

**4.4 Comparative Analysis of Different Methods**

*4.4.1 Quantitative Analysis*

The results of evaluating plate recognition for each model pipeline for the whole dataset of 400 images are shown in Table 5. The best performance was achieved by the LPRNet model, with an accuracy of 78.5% and an average Levenshtein Distance of 0.7200. The LPRNet model was able to accurately recognize the license plates in the majority of the images in the dataset. Its Average Character Level Accuracy was also the highest, which is 87.89%. The WPOD-net + MobileNets and WPOD-net + Pytesseract models had a lower performance than the LPRNet model, with accuracy levels of 16.25% and 15.25%, respectively. The average Levenshtein Distance was also higher for these models, at 3.6675 for WPOD-net + MobileNets and 3.6875 for WPOD-net + Pytesseract. This indicates that these models were not as effective in accurately recognizing the license plate characters as the LPRNet model. The VGG16 + Keras OCR (10% expansion) model had an accuracy of 41.75% and an average Levenshtein Distance of 2.4275. Although this performance is better than that of the WPOD-net + MobileNets and WPOD-net + Pytesseract

models, it is still lower than that of the LPRNet model. The OpenCV + EasyOCR model had an accuracy of 28.00% and an average Levenshtein Distance of 3.1950. This performance was lower than that of the LPRNet, VGG16 + Keras OCR (10% expansion), and WPOD-net + MobileNets models, indicating that this model was not as effective in accurately recognizing the license plates in the images. In conclusion, the LPRNet model was the most effective in recognizing license plates in the images in the dataset, with the highest accuracy and the lowest average Levenshtein Distance. The WPOD-net + MobileNets and WPOD-net + Pytesseract models had a lower performance, while the VGG16 + Keras OCR (10% expansion) and OpenCV + EasyOCR models had intermediate performances.

As referred from Table 6, the performance ranking of all the methods on images with bright light conditions follows the same as the one when evaluated in the whole dataset. When the methods are evaluated on images with the dim light condition and blurred images, the overall performance rankings are still similar, which LPRNet still has the most superior performance while WPOD-net + MobileNets and WPOD-net + Pytesseract models have lower performance. It indicates that the varying conditions of the images does not really affect the performance of the different approaches. However, as explained before, the extremely superior performance of LPRNe might be caused by its overfitting issue and resulting in unfair advantages.

Although LPRNet has the most superior performance, it is questionable whether it is feasible to be implemented in real-world applications as it cannot generalize on unseen data. It is not flexible enough as it is only able to recognize the seen combinations of characters but there are almost infinite combinations of characters displayed on the license plate. LPRNet is better than other approaches as it is lightweight and can be productized into real-time monitoring which is very crucial for parking payment systems. Moreover, LPRNet is segmentation-free therefore its performance is decoupled from the performance of the upstream plate segmentation stage. For instance, there are three approaches here that utilize plate segmentation for completing automatic license plate recognition tasks, which are WPOD-net + MobileNets, WPOD-net + Pytesseract, and VGG16 + Keras OCR. These plate-segmentation-included methods suffer from the scenario where the license plates are not segmented and not detected and resulting in no characters being recognized.

The WPOD-net’s inability to detect and segment a few of the license plates has a significant impact on the overall performance of the WPOD-net pipelines, which include MobileNets and Pytesseract, and are shown in Tables 5, 6, 7, and 8. This is so because the WPOD-net represents the first step in the recognition process, and any mistakes made here will have a negative impact on the effectiveness of the recognition models. The recognition models MobileNets and Pytesseract will be unable to make an accurate prediction of the WPOD-net is unable to detect and segment the license plates accurately. In addition, plate recognition tasks are not a good fit for MobileNets or Pytesseract due to their flaws. MobileNets' lightweight design makes it quick and effective but also restricts its ability to recognize intricate patterns and shapes, such as the letters on a license plate. On the other hand, Pytesseract is not optimized for recognizing license plates with their distinctive font styles and aspect ratios because it is made for document text recognition. As a result, the WPOD-net pipelines' findings demonstrate that these recognition models are inadequate for correctly identifying Malaysian license plates. Therefore, it is anticipated that, among all other models, WPOD-net with MobileNets/Pytesseract pipelines will ultimately receive the lowest evaluation.

Other than that, the VGG16 with Keras OCR is second superior to the LPRNet in terms of overall performance. However, it had a few shortcomings as the Keras OCR model does not support special characters which means that the model is unable to perform towards the license plate that is not from Malaysia. On the other hand, the segmentation of the license plate, and the failure of capturing the correct coordinates of the license plate from the image will also result in the failure of the model. This can be particularly seen in the dataset which contains blurred images. This is also one of the major disadvantages of this model pipeline as the prediction dependency must be based on the correct segmentation of the license plate from the original image.

For the other hand, One of the five model pipelines that were tested for identifying license plates in the 400 picture dataset was the OpenCV + EasyOCR model. This model was less accurate than the LPRNet, VGG16 + Keras OCR (10% expansion), and WPOD-net + MobileNets models, with an average Levenshtein Distance of 3.1950. This suggests that, in comparison to the other models, the OpenCV and EasyOCR model did not do as well in correctly identifying the license plates in the photos. In this project, the segmentation of the license plate using OpenCV does not function well so it affects the result. The model will fail if it is unable to accurately extract the license plate's coordinates from the picture. This would improve the model's accuracy and allow it to better match the unique properties of the license plates in the photos. Second, by employing a wider variety of pictures, the training data might be improved. Improved recognition outcomes would arise from the model being able to manage changes in license plate appearances and lighting circumstances better.

Moreover, ROI assessment results for two detection models on a dataset of 400 images are presented in Table 9. The evaluation was conducted using averages for error, precision, recall, and IoU. The average error and average precision for the WPOD-net model were 41.51% and 74.29%, respectively. It had a median recall of 50.41% and a median IoU of 49.39%. The VGG16 model, on the other hand, had an average precision of 75.38% and an average error of 20.77%. It had an average IoU of 68.51%

and a recall of 84.50%. However, because the ROI assessment's ground truth has limitations, including the possibility of human error and the ability to only segment in rectangular form, it indicates that the evaluation is biased toward a rectangular approach, which lowers WPOD-net's performance. Due to the fact that the VGG16 was trained to segment license plates in a rectangular form using the annotation data from the Pascal VOC files. Therefore, it is also expected that the VGG16 model will result in a lower error in terms of segmentation of the license plate due to the original annotation file when compared to the WPOD-net which was a pre-trained model.

The output of each model pipeline for the isolate detection error dataset is shown in Table 10 above. WPOD-net with MobileNets, WPOD-net with Pytesseract, and VGG16 with Keras OCR (10% expansion) are the three pipelines that were assessed. First of all, with an accuracy of 38.33%, an average character level accuracy of 70.28%, and an average Levenshtein distance of 1.7166, the WPOD-net model. On the other hand, the VGG16 model is relatively stable in terms of performance compared to the WPOD-net model. This indicates that the VGG16 model has a better ability to generalize and extract the vehicle license plate under more extreme conditions such as night time or blurry images. Aside from this, the Keras OCR model is also known to be generally better than the EasyOCR model under more complex environments. This had likely contributed to the better performance of the VGG16 model with Keras OCS model compared to the WPOD-net.

*4.4.2 Qualitative Analysis*

Figure 19 displays the whole image that was utilized in assessing the recognized license plate characters under various situations, using one image from each image condition as the reference.


Figure 19. Example input images for qualitative analysis

with the MobileNets pipeline performs well. Next, with a 30.0% accuracy rate, a 68.98% average character level accuracy, and an average Levenshtein distance of 1.8166, the WPOD-net with Pytesseract pipeline is highly accurate. Finally, the accuracy, average character level accuracy, and average Levenshtein distance for the pipeline using VGG16 with Keras OCR (10% expansion) are 48.30%, 67.02%, and 2.1667 respectively. The results are as expected because the dataset was carefully chosen to remove the majority of detection errors and concentrate on the effectiveness of the recognition models. The accuracy of the WPOD-net pipelines, MobileNets and Pytesseract, dramatically improves in comparison to the VGG16 + Keras OCR (10% expansion), as the performance of the pipelines is significantly impeded by the pipelines for the detection

| **Approach** | **Recognized License Plate Characters on Different Conditions** | | |
|---|---|---|---|
| | Bright Light | Dim Light | Blurred |
| **Ground truth** | QMU86 | QPA2412 | BKR5558 |
| **LPRNet** | QMU86 | QPA2412 | BKR5558 |
| **WPOD-net + MobileNets** |  W6 |  PA2412 |  None |

| | | | |
|---|---|---|---|
| **WPOD-net + Pytesseract** |  7 |  QPK2412 |  BAR55568 |
| **VGG16 + Keras OCR (10% expansion)** |  QMU86 |  QPA2412 |  BIR5558 |
| **OpenCV + EasyOCR** | Tooa | 2412 | 58 |

Table 12. Recognized license plate characters on different conditions

Based on Table 12 above, it lists several license plate characters that have been recognized under different conditions. The conditions are "Bright Light ", "Dim Light " and "Blurred". First, LPRNet has correctly recognized all the license plates regardless of the condition of the images. Next, for WPOD-net using MobileNets, the contours had a significant influence on the prediction outcome for the MobileNets model as the recognizer pipeline. For example, take a look at the image taken in a bright light condition to see why the precision is low and the desired outcome is simply "W6." Only 2 characters would be anticipated by MobileNets since the contour algorithm for MobileNets could only distinguish 2 characters from the license plate. Then, for WPOD-net with Pytesseract, the image of the plates connected in the table above is the identical image that was cropped by WPOD-net and was then passed straight into Pytesseract for identification. As a result, even though the model still has issues with images taken in bright light, it was still able to accurately identify a few characters under other circumstances. Aside from that, the VGG16 model could extract the region of the vehicle license plate after performing expansion. However, the limitation came into the Keras OCR model for detecting the blurry image, an extra image preprocessing method should be conducted in order to solve the issue for the blurry image to improve the ability for the Keras OCR model to extract the text from the vehicle license plate. Lastly, for OpenCV and EasyOCR, plate segmentation will detect the rectangular contours and then select the best candidate license plate region. After selecting the license plate region, EasyOCR library detects and recognizes text and also post-process recognized text using a regular expression to extract the license plate number. In the first condition (Bright Light), the original plate number "QMU86" is recognized as "Tooa" by the system. In the second condition (Dim Light), the original plate number "QPA2412" is correctly recognized as "2412". However, in the third condition (Blurred), the original plate number "BKR5558" is recognized as "58".

## 5. CONCLUSION

In conclusion, the purpose of this study was to create a reliable and precise license plate recognition (LPR) system that could recognise Malaysian license plates. To accomplish this, multiple deep learning and visual image processing algorithms were applied, and a comparative analysis of these diverse approaches was carried out. The LPRNet model outperformed the other model pipelines, with an accuracy of 78.5% and an average Levenshtein Distance of 0.7200 for a dataset of 400 pictures. This was much better than other models with lesser performance levels, such as WPOD-net + MobileNets, WPOD-net + Pytesseract, VGG16 + Keras OCR (10% expansion), and OpenCV + EasyOCR. Further research may aim to enhance the LPR system's performance by adding new features and using cutting-edge deep learning techniques.

## 6. REFERENCES


[1] S. M. Silva, & C. R. Jung (2018). License Plate Detection and Recognition in Unconstrained Scenarios. In 2018 European Conference on Computer Vision (ECCV) (pp. 580-596).

[2] Kiruri, S. (2021). License Plate Detection And Recognition Using OpenCv And Pytesseract. Section.io.

[3] Tushar Goel, Dr. K.C. Tripathi, & Dr. M.L. Sharma. (2020). SINGLE LINE LICENSE PLATE DETECTION USING OPENCV AND TESSERACT. International Research Journal of Engineering and Technology, 07(05), 4.

[4] Quangnhat185. (n.d.). Plate detect and recognize [GitHub repository]. Retrieved from https://github.com/quangnhat185/Plate_detect_and_recognize

[5] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications (Version 1). arXiv. https://doi.org/10.48550/ARXIV.1704.04861

[6] Chadha, A., Kashyap, S., Gupta, M., & Kumar, V. (2020). License Plate Recognition System using OpenCV & PyTesseract. CSI Journal of, 31.

[7] Sergey Zherzdev, & Alexey Gruzdev (2018). LPRNet: License Plate Recognition via Deep Neural Networks. CoRR, abs/1806.10447.

[8] Awalgaonkar, N., Bartakke, P., & Chaugule, R. (2021, September 1). Automatic License Plate Recognition System Using SSD. IEEE Xplore. https://doi.org/10.1109/IRIA53009.2021.9588707

[9] Vedhaviyassh, D. R., Sudhan, R., Saranya, G., Safa, M., & Arun, D. (2022, December 1). Comparative Analysis of EasyOCR and TesseractOCR for Automatic License Plate Recognition using

Deep Learning Algorithm. IEEE Xplore. https://doi.org/10.1109/ICECA55336.2022.10009215

[10] Kartikey, K. S. (2022). VEHICLE NUMBER RECOGNITION AND COUNTING THE NUMBER OF VEHICLES. Journal of Pharmaceutical Negative Results, 3881–3888. https://doi.org/10.47750/pnr.2022.13.S06.517

[11] Soon, C.K., Lin, K.C., Jeng, C.Y., & Suandi, S.A. (2012). Malaysian Car Number Plate Detection and Recognition System.

[12] Li, Hui and Shen, Chunhua (2016). Reading Car License Plates Using Deep Convolutional Neural Networks and LSTMs.

[13] Omar, N., Mohsin Abdulazeez, A., Sengur, A., & Saeed Al-Ali, S. G. (2020). Fused faster RCNNs for efficient detection of the license plates. In Indonesian Journal of Electrical Engineering and Computer Science (Vol. 19, Issue 2, p. 874). Institute of Advanced Engineering and Science. https://doi.org/10.11591/ijeecs.v19.i2.pp874-982

[14] X. Ascar Davix & E.Praynlin & D.Judson (2021). VGG-16 CNN APPROACH FOR VEHICLE LICENSE PLATE LOCALIZATION. In August 2021, Volume 20, Issue 10, Advances and Applications in Mathematical Sciences (pp. 2479-2486)

[15] Rafique, M. A., Pedrycz, W., & Jeon, M. (2017). Vehicle license plate detection using region-based convolutional neural networks. In Soft Computing (Vol. 22, Issue 19, pp. 6429–6440). Springer Science and Business Media LLC. https://doi.org/10.1007/s00500-017-2696-2

[16] J. -Y. Sung, S. -B. Yu and S. -h. P. Korea, "Real-time Automatic License Plate Recognition System using YOLOv4," 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), Seoul, Korea (South), 2020, pp. 1-3, doi: 10.1109/ICCE-Asia49877.2020.9277050.