Development for Application of Malaysia License Plate Recognition (MLPR) using Convolutional Neural Network (CNN).

by

Muhamad Faiz bin Ibrahim

17004095

Dissertation submitted

in partial fulfilment of the requirement for the

BACHELOR OF INFORMATION SYSTEMS (Hons)

SEPTEMBER 2021

Universiti Teknologi PETRONAS

32610

Seri Iskandar

Perak Darul Ridzuan

Development for Application of Malaysia License Plate Recognition (MLPR) using Convolutional Neural Network (CNN).

by

Muhamad Faiz bin Ibrahim

17004095

A project dissertation submitted to the

Universiti Teknologi PETRONAS

in partial fulfilment of the requirement for the

BACHELOR OF INFORMATION SYSTEMS (Hons)

Approved by,

[illegible]

Dr Hitham Seddig A A
Senior Lecturer, SMIEEE
Core Member, CeRDas, IAS
CISD, FSIT, UTP
Perak, Malaysia

____

(Ts Dr Hitham Seddig Alhassan)

UNIVERSITI TEKNOLOGI PETRONAS

BANDAR SERI ISKANDAR, PERAK

SEPTEMBER 2021

CERTIFICATION OF ORIGINALITY

This certification certifies that I am totally responsible for the work presented in this project, that the original work belongs to me except as mentioned in the reference and acknowledgement, and that the original work included herein was not undertaken or completed by the unnamed source or individuals.

[illegible]

____

MUHAMAD FAIZ BIN IBRAHIM

ABSTRACT

Despite the increasing number of commercial and academic techniques for Automatic License Plate Recognition (ALPR), most existing methods are only region-specific (i.e., Europe, US, Brazil, Taiwan, or others) and lack value in the accuracy achieved. ALPR frequently explore datasets containing approximately frontal images (i.e., frontal images from a license plate). A comprehensive ALPR system that focuses on unrestrained capture scenarios where the number plate (NPs) may be significantly distorted to ensure it has a very robust force of accuracy. Key contribution was the creation of a deep learning Convolutional Neural Network (CNN) that could detect and repair numerous deformed license plates in a single picture, which was then integrated with the next method, the Optical Character Recognition (OCR). The purpose in utilising algorithm method to obtain good investigation results. Furthermore, we provide manual annotations for difficult number plate (NPs) image sets from various regions and acquisition settings as additional contributions. Although our experimental results show that the proposed method performs similarly to the latest commercial systems in traditional scenarios, they also show that it overcomes together academic and commercial approaches in challenging scenarios when no parameter alteration or fine tuning is done for a particular scenario. The deep learning algorithm in the technique of detecting plate numbers is the best indication.

Keywords: Number Plate · Deep learning · Convolutional Neural Network

ACKNOWLEDGEMENTS

During difficult times such as the current pandemic, the route to completion of my Final Year Project has been quite difficult. Even though it is quite difficult, I have managed to get through it with ease thanks to a fantastic amount of support. My family is the first and most important individual I had like to express my gratitude to. Because of their continuous love and support, I have been able to complete and complete my project on schedule and on budget. I am extremely grateful to them. To complete my project, they have served as a source of strength and encouragement during my entire trip. I am grateful to them for their assistance.

Second, I had like to offer my heartfelt gratitude and admiration to my excellent supervisor, who has been a tremendous help to me in finishing this project. Big thanks to Ts Dr Hitham Seddig Alhassan, for everything he has done for me. Despite his hectic schedule, he provided his with unwavering support, inspiration, and direction always. He has been an incredible mentor and guided me in depth through the process of deciding what I need to do in this project. Not to forget he also helped in how I planned to complete the final year project which was solely concerned with the quality of the work I produced. I am very grateful for his undivided instruction and driving, as he is the best supervisor in ensuring to achieve of goals and objectives.

Not to forget that the backbone in this success is, to my friends who have helped and assisted me in my time of need without expecting a return and even money. If I was not surrounded by supportive and motivated people, I would not have taken up the challenge of observing something I did not know at the time. As a result, I am as technologist, I hope the talents, skills and experience I have gained will help me to be a stronger and inventive person in the future in my field.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 Background Study of Convolutional Neural Network

Convolutional Neural Networks (Convent/CNN) are a standard method for deep learning modelling that can take an input image and assign relevance (learnable weights and biases) to various aspects or objects within it, allowing it to distinguish between them. [1]. Figure 1 demonstrates the framework for the convolutional neural network to recognize the images.

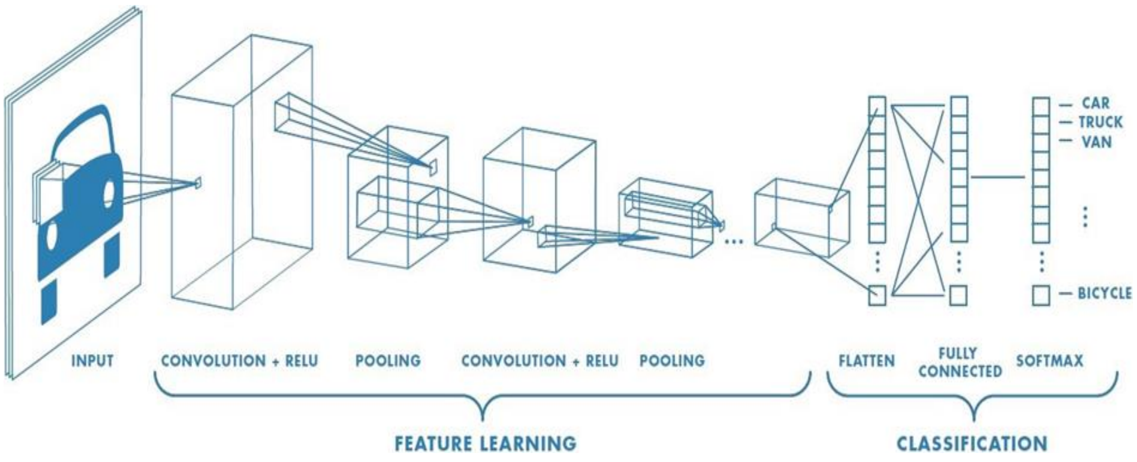


Figure 1.Framework for Convolutional Neural Networks (CNNs)

The term "Convolutional Neural Network" was established from a Convolution mathematical process, combining (employ together) two or more mathematical procedures like sigmoid, leaky ReLU, or Tanh [3]. The implementation has stated that there are various applications in recognizing images. The success of CNN can be attributed to the hidden layers, which are not entirely coupled to the previous layers [2] and perform several calculations between convolution and pooling (subsampling layer). To one side from other deep learning neural networks, backpropagation makes CNN easy to train. Each layer is extremely sparsely connected, and a linear filter is utilised to aid with image recognition. [2]

In 1990, LeCun et al. presented the first-ever architecture. It was created to predict handwritten digits, and however, because of a lack of training data and computing capacity, the architecture was not very successful. In 2012, Krizhevsky et al. introduced AlexNet, which successfully lowered the error rate and won the ILSVRC 2012. After becoming the most widely used Neural Network, CNN is now utilized in various applications, including object detection and tracking [3], posture estimation, text recognition, and numerous others.

1.2 Background Study of Optical Character Recognition

Optical Character Recognition (OCR) is a process that converts images of typed, handwritten, or printed text into machine-encoded text by electronic or mechanical means. This can be accomplished via a scan of the document, an image of the document, a scene image, or by superimposing subtitle text on the image itself. Figure 2 shows the framework for optical character recognition, which is used to recognise text on images.

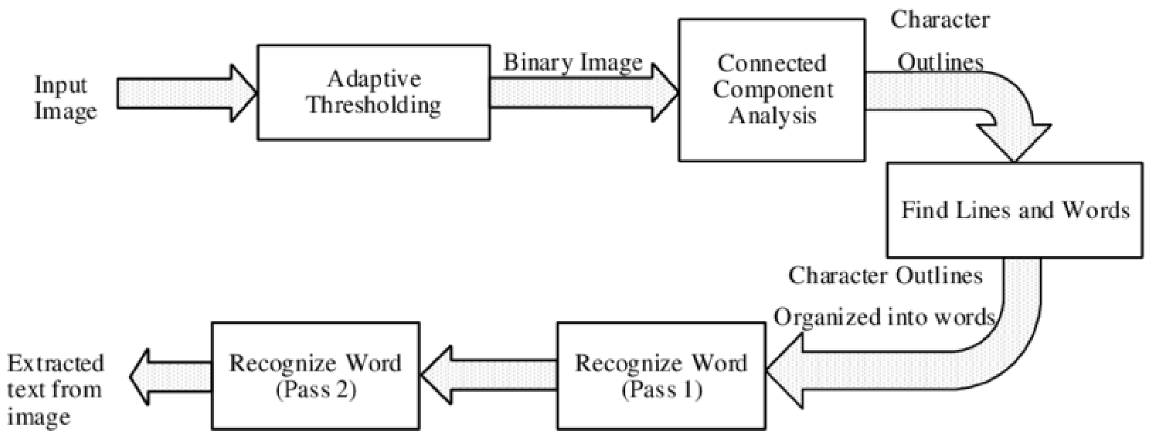


Figure 2.OCR Framework

The fields of Artificial Intelligence and computer vision have concentrated their efforts on optical character recognition (OCR), which is important for automatic number plate recognition. OCR has advanced drastically since its inception and is now considered one of the most important aspects of Artificial Intelligence and computer vision research field [4]. OCR makes use of the fact that each character has a distinct of characteristics from the others, such as corners, ends, and bifurcations. As a result of inheriting these qualities, the algorithm gets better and less challenging. The input character is changed to an edge image in an iterative process. Using this edge image processing procedure, the features are extracted, and the features vector is then updated with the number, state of the features, and direction. The entire number of features and their associated states.

1.3 Licence Plate Recognition

License plate recognition system (LPR) implemented in various algorithms has been widely used in real world applications. It is mainly such as, automated toll collection, automated performance optimization of Automatic Enforcement System (AES), indoor entry and exit detection and crime pursuit [5] - [6]. Specifically, in safety measure, LPR has been utilized in traffic management to identify vehicle owners who have violated various sorts of traffic rules, as well as to find vehicles that have been stolen by irresponsible parties. There is no denying that LPR is a good security initiative as it has been used since 1979 with various techniques. Figure 3 illustrates a collection of image recognition scenarios involving license plate recognition.

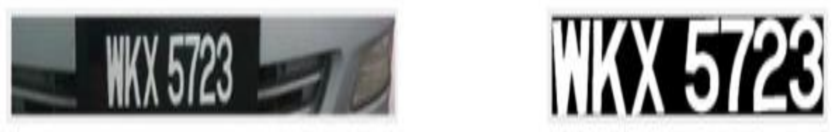


Figure 3.Example of the License Plate Recognition

Image processing will be the primary approach used in the MLPR recognition system. Due to the restricted capacity for multi-scaling, developing the MLPR system through image processing might be challenging, as the MLPR image may appear unclean, motion blurred, low resolution, poor lighting, and with low contrast. The process of recognising the license plate is divided into five parts. The license plate image of an input vehicle was first found and isolated using localization techniques. Following that, plate orientation is used to compensate for the license plates skew condition. Resizing is used to modify the dimensions to the needed size. Following that, image normalisation will be implemented to accommodate the brightness and contrast of the images. Character segmentation will be utilised to separate individual characters from the licence plate.

The recognition algorithm in the MLPR system is practically implemented to ensure its accuracy. There are several steps involved, including adaptive thresholding, component labelling, feature extraction, and recognition. The character identification procedure is the most difficult of the five major phases. This is because the type of algorithms used in the first four key elements significantly impact character recognition. The segmented characters can take on a variety of appearances. As a result, a reliable character identification system is required, and CNN can focus on the issues. This is because the type of algorithms used in the first four key elements significantly impact character recognition. The segmented characters can take on a variety of appearances. As a result, a reliable character identification system is required, and CNN can address the issue mentioned above. At the end of the process, OCR converts the characters into encoded text information, from which the text identified from the Region of Interest (ROI) is shown as an output result. Figure below demonstrate the way how two algorithms being implemented in recognizing the license plate.

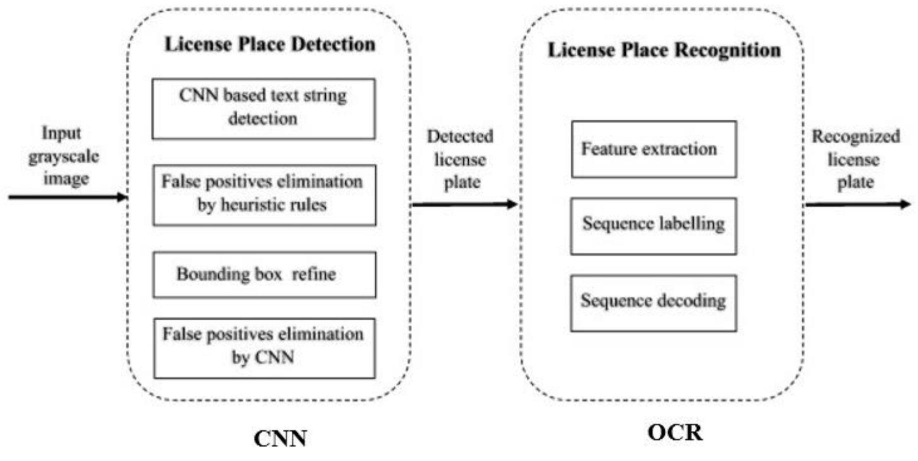


Figure 4. CNN + OCR algorithm

1.4 Advantages of License Plate Recognition

In general, an enhanced and efficient security system is a significant advantage of License Plate Recognition (LPR). With the existence of this system, it helps to identify license plates and analyze them quickly even at high incidences. That said, with the help of robust AI implementation, the results and accuracy are very excellent. Using LPR, authorities may analyze data for suspicious vehicles or vehicles implicated in crime extremely fast and without reluctance and can automatically recognize license plates with just one image capture. Thanks to keeping the data for a while, the LPR data can provide both an alibi and the accusing data. At a basic level, LPR provides security in a variety of settings such as open parking, entrances to buildings, staff vehicle parking permits, or recognizing vehicles that were previously barred from the premises or building. LPR provides an additional level of protection for both public and private environments.

Then, the automated services are part of the LPR function. The cameras are an efficient and cost-effective way to monitor parking solutions, vehicles, and Automated Enforcement Systems (AES). Their high-accuracy readings and 24-hour operation make them more efficient rather than dependencies on manual labour such as security guards. LPR also provide a non-confrontational alternative, which some drivers have found helpful while receiving fines. Once AES captures the image is implemented by CNN algorithms, it will automatically detect the license plate owner and automatically send the fine tickets. LPR users' teams often find that both traffic personnel and LPR cameras work well together, especially in traffic and parking enforcement, where staff can rely on LPR to provide the necessary information, minimizing the time they spend on the streets.

The next benefit provided by LPR is real-time and instant benefits; as of the real-time and instant imaging capabilities of LPR, it is advantageous to a wide range of sectors. In the past, number plate recording took time, and then sending out penalty notifications to people who broke traffic laws took even longer. Number plate can now be identified and checked against a database practically instantly, thanks to LPR adoption. The rapid reaction time of these cameras allows for a prompt response to illegal activity, guaranteeing that no undesirable behaviour goes ignored.

1.5 Problem Statement

Based on past related works in License Plate Recognition (LPR), the commonly utilized method is called the convolutional neural network. Despite the widely used CNN, the conventional systems lack accuracy in the current ALPR. On the other hand, additional algorithm to be pair with CNN, sometimes not suit and provide lower accuracy result. The surrounding bumper plates can now be identified and checked against a database practically instantly, with the LPR adoption. The quick response time of these smart recognition system enables for a quick response to criminal conduct, ensuring that no unwelcome behaviour goes unnoticed tightness. There is room for improvisation for the plate recognition system. The second problem is that most License Plate Recognition has limitations in detecting license plate origin in the current LPR. It only recognizes the image of the license plate. On the other hand, there is a significant limitation to detect license plates by using manual labour as it is very time-consuming.

1.6 Objectives

This project's major goal is to create a real-time License Plate Recognition algorithm. Furthermore, prior to attaining the primary purpose, a few critical objectives must be met, such as enhancing the present system's accuracy in identifying the vehicle plate and avoiding spoofing of the license plate recognition system. Furthermore, prior to attaining the primary purpose, a few critical objectives must be met, such as enhancing the present system's accuracy in identifying the vehicle plate and avoiding spoofing of the license plate recognition system.

i. To design and detect Malaysia License plate using better image recognition algorithm by using CNN and OCR.
ii. To recognize and process plate numbers and characters of vehicles to fulfil the security requirement.
iii. To develop interactive website application that accessible by the users and authorities of MLPR using Python Flask.

1.7 Scope of Study

The goal of this research is to develop smart plate recognition application. This research focuses on the development of a low-cost, non-invasive Malaysian Plate Recognition system based on CNN software and OCR software in preparation for future hardware and software implementations in Malaysia. Therefore, this study will utilize the camera and the CNN software. This study emphasizes the importance of image recognition for security usage to achieve a better security system. Then, the study's coverage is focusing on developing an efficient platform based on Artificial Intelligence.

1.8 Limitation

MLPR aims to develop intelligent plate identification using CNN and OCR algorithms, however, this study only focuses on developing unique license plate identification for the most widely used vehicles in Malaysia, namely cars. This is so to ensure the system used in the initial phase has a level of good accuracy. Once it starts showing success and attracts investors, it will be expanded towards more vehicles. Then, the MLPR prototype developed by this study is only for High-definition cameras with high fps. This is to avoid problems in terms of accuracy in the segmentation of each character and number recognized. In addition, the MLPR has limitations in collecting data on local license plates. This is so open source related, and data does not exceed a thousand image data.

CHAPTER 2
LITERATURE REVIEW

2.1 Overview of Image Recognition Technology

Automatic image data processing is increasingly being used in the society and industrial settings. It is used for quality control, measurement, component recognition [7], operating IoT devices and others. It is also used for monitoring system and human identification [8]. Image data processing cannot be equated same as image processing because it is totally different. Image processing in the same way as digital media, creative image design, and image manipulation. Images are utilized in industrial image data processing to get relevant information about parts or things. Based on this information, choices are made about whether the associated parts are in good working order in daily life the image recognition is crucial, thus, a deep neural network such as CNN being used heavily in the industry. We always focus on accuracy; it is very crucial the machine can detect the images correctly without any interferences. Figure 5 represents an example of an image recognition system in use by the CNN.

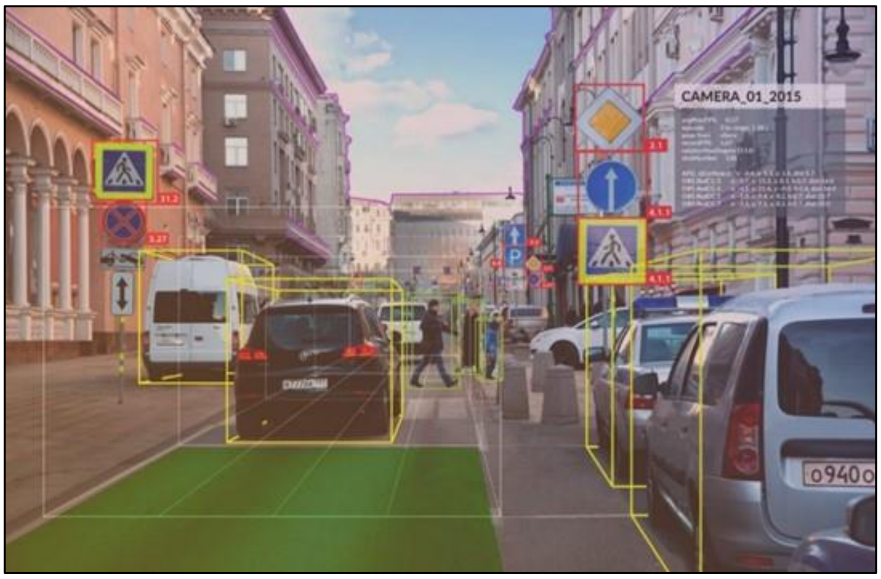


Figure 5.Image Recognition using CNN Software.

Using image recognition options in the security industry saves time and cost, ensures product quality, and ensures high accuracy. Implementation of AI can further increase the level of accuracy and efficiencies. Each technology adapted indeed has additional challenges, and teaching time should be drastically reduced. Teaching and learning towards the use of image recognition technology need to be improved. Indirectly it can increase the level of public awareness of this technology. Image recognition generally has excellent advantages therefore, it needs to be implemented well and thoroughly.

Following that, an image is a means of delivering information, and an image contains a great deal of useful information itself. The ability to comprehend an image and extract information from it in order to complete certain tasks is critical in automated image technology and machine learning applications. Image segmentation is a procedure involved in comprehending the image. In practice, it is frequently not interested in the entire image, but rather in select portions with similar characteristics. Image segmentation is one of the hotspots in image data processing and computer vision. It is also an essential basis for image recognition. It is based on specific criteria to divide an input image into several the exact nature of the category to extract the area that people are interested in. Moreover, it is the basis for image analysis and understanding of image feature extraction and recognition [9].

Finally, due of its efficiency and adaptability, Artificial Intelligence is increasingly routinely used to assist in the detection of breaches into computer systems. The neural network is the Artificial Intelligence component that has received the greatest attention. A neural network analyzes the data and delivers a probability evaluation that the data corresponds to the qualities learn rate that it has been trained to recognize. This has a significant impact on lowering the false-positive rate [10].

2.2 CNN Architecture

Since 2012, academics and deep learning developers have been developing high-efficiency CNN architectures using structural modification approaches such as Inception-ResNet-v2 [11]. Layers are added up to 16 and 19 layers in the neural network. The deep learning of this Inception module is designed using the GoogLeNet architecture [12,13]. This module is assigned to utilize circumferences of various sizes, including those to assure precision, and it provides filters of various sizes such as 1x1, 3x3, and 5x5. The output of each Inception module is entered through each filter together (Combined Filter). In the ResNet Architecture [14] this training process can skip more than one layer by planning Balance Blocks. ResNet is designed to have from 18, 34, 50 and 101, to 152 nerve layers of neural network. The trend in the development of the CNN architecture is to increase the number of layers, but to reduce the number of parameters, for example, Inception-ResNet [13] and DenseNet [15] are implemented.

2.2.1 Inception-ResNet-v2 architecture

Inception-ResNet-v2 [16] was designed with batch normalisation to improve the training pace of the neural network. Only 7% of the training procedures have the potential to improve the efficacy of the architecture. Factorization is used to lower the filter size of any related usability, which results in a reduction in filter size. In addition, due of the overfitting problem, the number of parameters has been raised as well. The increase in the amount of residual block between inception increased the completion process. A huge number of Inception modules are generated because of this module. The fundamental structure of Inception-ResNet-v2 is shown in Figure 6, and it divides the work function into blocks, each of which comprises the Stem, Inception-ResNet, and Reduction blocks.

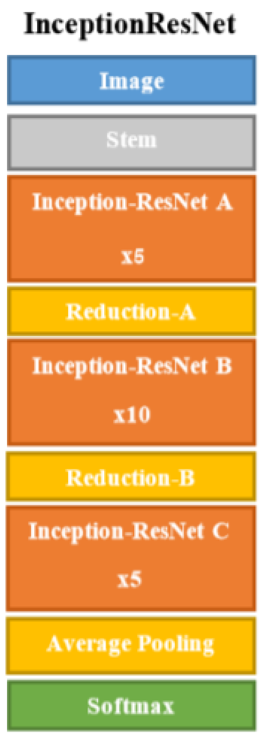


Figure 6. Inception-ResNet-v2 architecture

2.2.1.1 Stem Block

The Stem block is the first architectural layer, and it is made up of three blocks. It comes before the Inception module in the hierarchy. The convolution filter in the Stem block is 3x3, and the stride values are 2 (s2). As a result, the feature map would grow smaller, which would result in a direct drop in the parameter values, as seen in Figure 7.

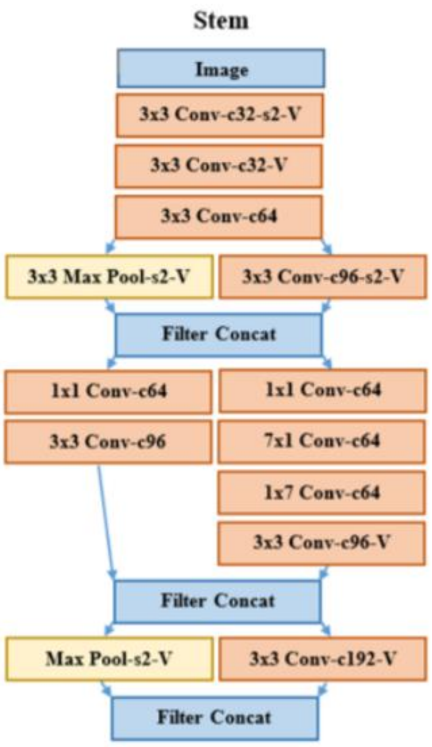


Figure 7. Detail of the Stem block

2.2.1.2 Inception-ResNet Block

The Inception module's benefit is the combination of ResNet Architecture with the Inception layer in the model. That is why the block is referred to as the Inception-ResNet block. The Inception-ResNet is composed of three blocks [16], denoted by the letters Inception-ResNet A, Inception-ResNet B, and Inception-ResNet C in Figure 8. In case of parameter reduction, the space between Inception-ResNet blocks is filled by Reduction blocks.

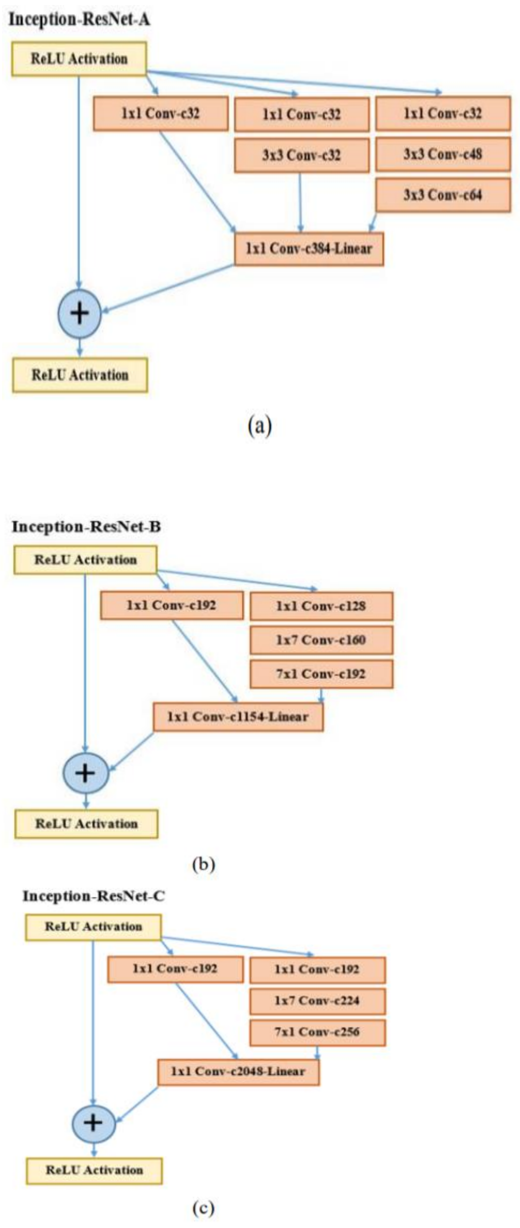


Figure 8Architecture details of the Inception-ResNet.

2.2.1.3 Reduction Block

The Reduction block located between the InceptionResNet blocks is used to lower the size of the feature map. Two Reduction blocks are included in the Inception-ResNet design. As seen in Figure 9, these are the reduction-A and reduction-B.

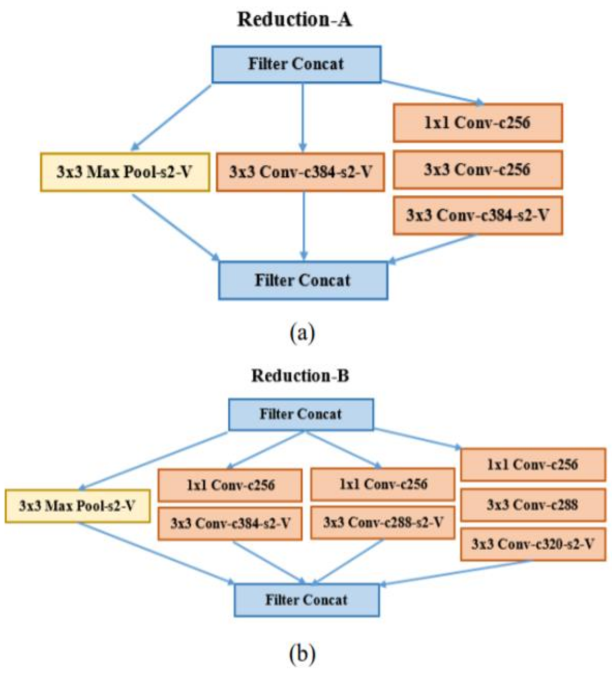


Figure 9.The reduction block: (a) & (b)

2.3 The Importance of Image Recognition in The Industry Using CNN.

In image recognition industry usability, the advancement of image recognition is extensively used in managing IoT devices, facial recognition and for quality assurance control. Existing image segmentation approaches include region-based segmentation, edge detection segmentation, clustering-based segmentation, and segmentation based on supervised learning in CNN to provide the best recognition output. This project analyses and summarizes these image recognition algorithms and compares the advantages and disadvantages of different algorithms. Finally, by combining these algorithms, the future development trend of image processing techniques may be improved. Figure 10 shows CNN's picture recognition technology in operation.

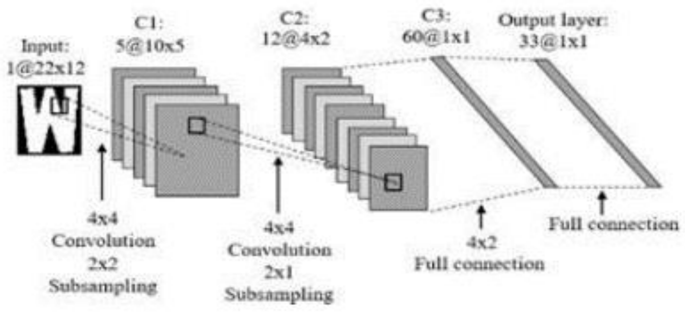


Figure 10.Components of CNN [3]

2.4 The Implementation of Image Recognition in The Industry Using CNN.

Through a deep nonlinear network model in deep learning CNN may approximate have the complicated function. But it saves time by avoiding the tedious task of manually extracting features and allows the data to be described more fully features [17]. Driven by the fact that colour differences play an important role in image recognition, MLPR proposes a new solution to use RGB proportions clearly in the initialization process for this neural network in CNN. The proposed method was used based on the traditional initiation method, which was designed to use the pre-training method to simulate the RGB distribution after training. We conducted various studies to assess the efficacy of RGB-based weight initialization and to provide a feasible way for implementing the initialization to speed up training procedures. This suggested strategy can improve convergence performance in circumstances when the colour effect difference in the dataset is considerable, and it also suggests a viable avenue for pre-training study. Figure 11 indicates the initialization of the RGB in the neural converting network (CNN).

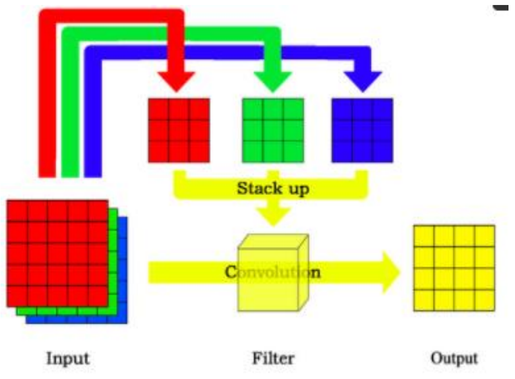


Figure 11.Implementation of RGB initialization in (CNN) [11].

2.5 A Review of Image Recognition using CNN Techniques and Accuracy

In general, the image recognition accuracy of CNN is measured in terms of the different sizes and numbers of filters that are implemented in the model. The process is to ascertain their effect classification [18]. Results may vary from different filter sizes and the number of filters that have been applied in the model. The number of filters and the size of the filters have a major impact on the accuracy of image recognition. By applying filter size of 3×3 pixels and 5×5 pixels, testing accuracy is greater when there are 128 filters than when there are 32 or 64 filters. However, training time is longer when there are 128 filters compare to 64 or 32 filters.

Furthermore, when the filter size is 3×3 pixels, the testing accuracy is higher than when the filter size is 5×5 pixels or 7×7 pixels. The CNN structure with four convolutional layers and two pooling layers produced the highest testing accuracy. When using a bigger filter size and number of filters 32 for data input layers, improved results can be gained by decreasing the filter size and increasing the number of filters in following layers, also the other way around. This structure has efficiencies increased classification results, but it has taken a long time to achieve them. More filter sizes 9×9 and 11×11 with CNN will be available in the future for usage in the workplace. In order to determine the influence on classification accuracy, increase the number of layers (convolution and pooling layers) that are applied to the pictures and observe their consequences [14]. Figure underneath depicts the accuracy and loss of the model when different filters and sizes are used.

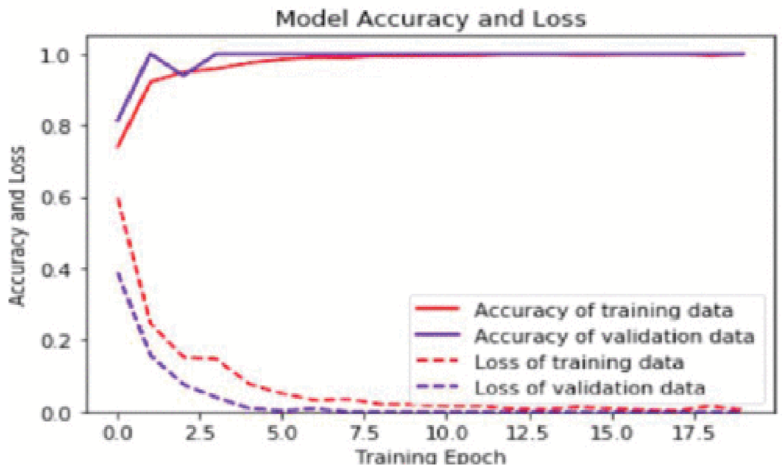


Figure 12.Model Evaluation: 32 Filters, 3x3 Pixels

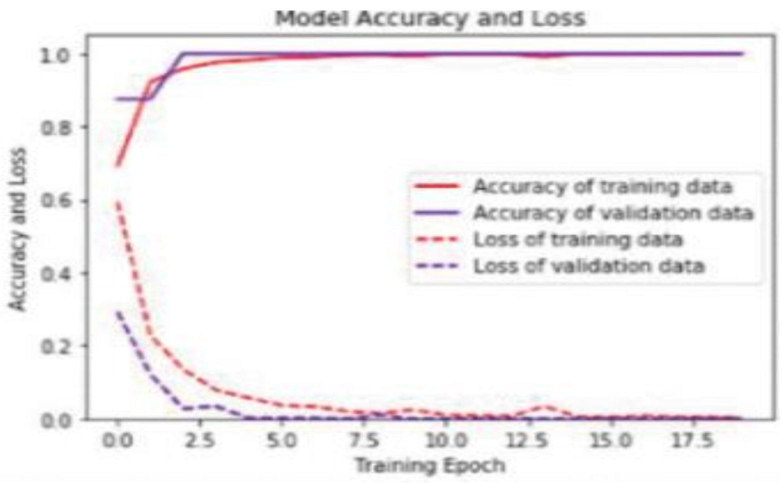


Figure 13. Model Evaluation: 64 Filters and 3x3 Pixel Filters

.

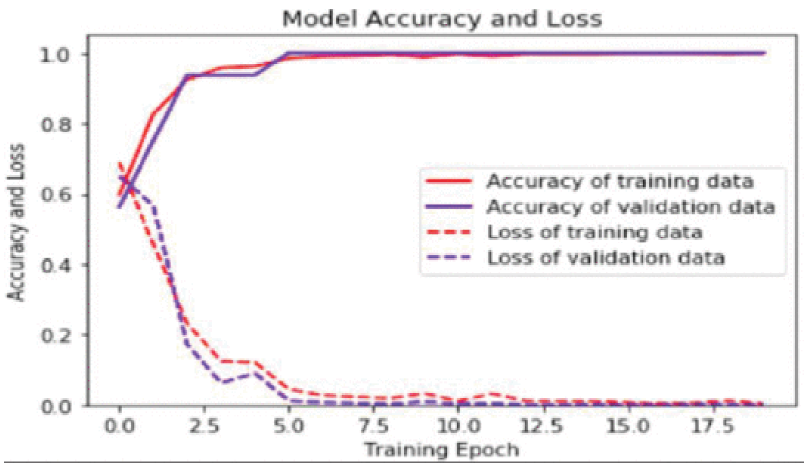
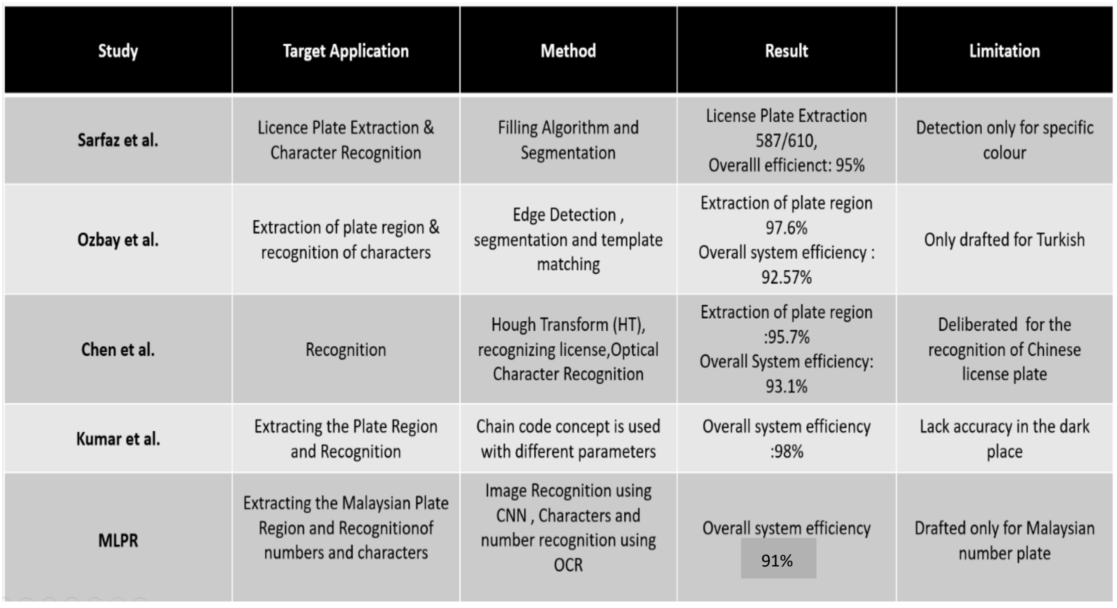


Figure 14. Model Evaluation: 128 filters with a 3x3 pixel size.

2.6 Comparative Study with The Existence of LPR

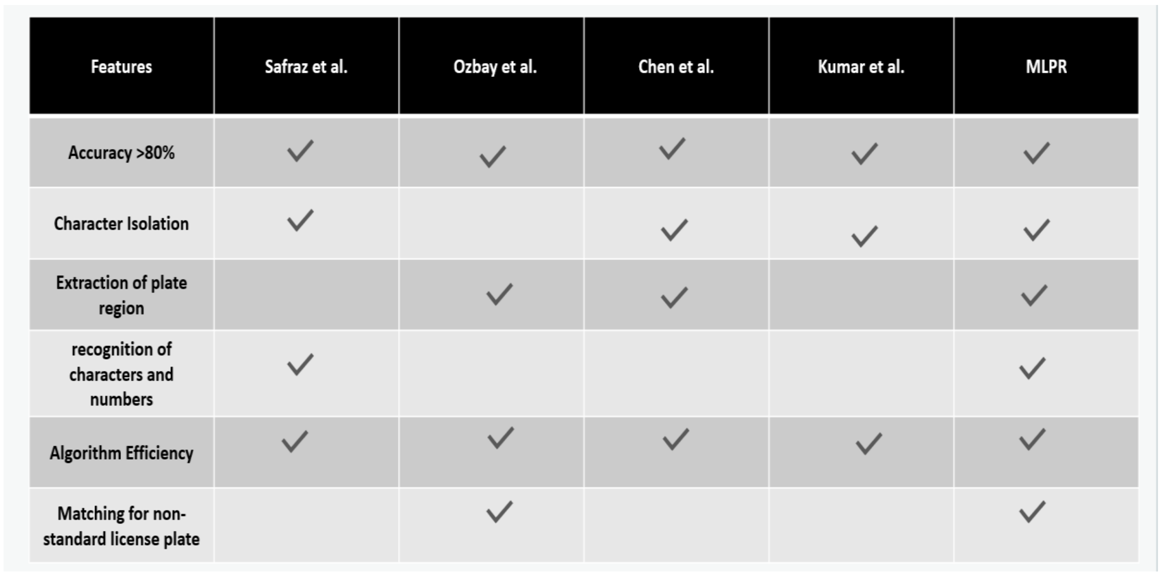
Numerous LPR systems have been developed before the MLPR. However, each system has its advantages and disadvantages. Based on studies [19], [20], [17], [21], each prototype uses the same approach that is image recognition by CNN. However, each prototype has a different approach to fulfilling the needs. This is because some only focus on the scope for their respective countries plate number only. However, it must be remembered, and efficiency needs to be emphasized. Each LPR focuses on the efficiency that best meets the requirements. Table 1 and Table 2 shows a comparison of existing LPR.

Table 1. LPR Comparative Studies



Study	Target Application	Method	Result	Limitation
Sarfaz et al.	Licence Plate Extraction & Character Recognition	Filling Algorithm and Segmentation	License Plate Extraction 587/610, Overalll efficient: 95%	Detection only for specific colour
Ozbay et al.	Extraction of plate region & recognition of characters	Edge Detection , segmentation and template matching	Extraction of plate region 97.6% Overall system efficiency : 92.57%	Only drafted for Turkish
Chen et al.	Recognition	Hough Transform (HT), recognizing license,Optical Character Recognition	Extraction of plate region :95.7% Overall System efficiency: 93.1%	Deliberated for the recognition of Chinese license plate
Kumar et al.	Extracting the Plate Region and Recognition	Chain code concept is used with different parameters	Overall system efficiency :98%	Lack accuracy in the dark place
MLPR	Extracting the Malaysian Plate Region and Recognitionof numbers and characters	Image Recognition using CNN , Characters and number recognition using OCR	Overall system efficiency 91%	Drafted only for Malaysian number plate

Table 2. LPR Efficiencies Comparison



Features	Safraz et al.	Ozbay et al.	Chen et al.	Kumar et al.	MLPR
Accuracy >80%	✓	✓	✓	✓	✓
Character Isolation	✓		✓	✓	✓
Extraction of plate region		✓	✓		✓
recognition of characters and numbers	✓				✓
Algorithm Efficiency	✓	✓	✓	✓	✓
Matching for non-standard license plate		✓			✓

CHAPTER 3
METHODOLOGY

3.1 Software Development Life Cycle

The CRISP-DM process consists of several steps, which are summarized in the picture below Figure 15:

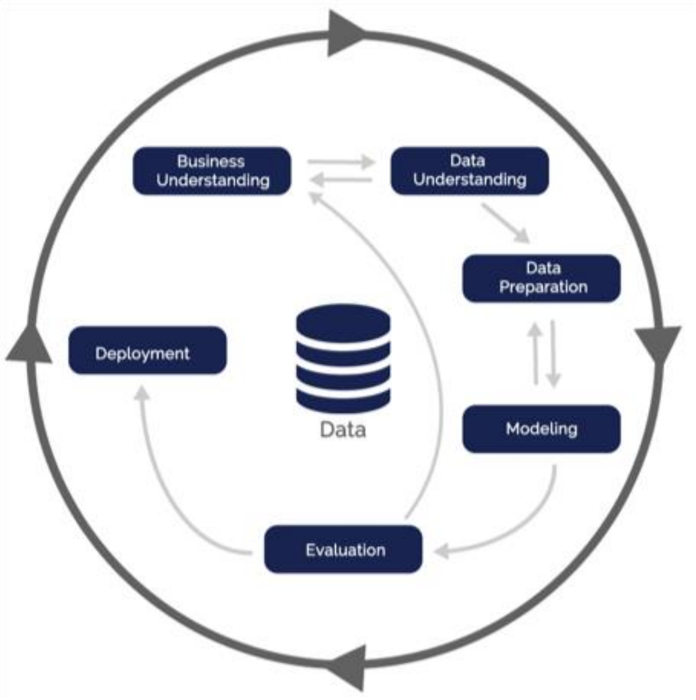


Figure 15. CRISP-DM Model

The CRISP-DM software development life cycle model was utilised for this project's software development life cycle. The incremental, consistent, and long-term CRISP-DM paradigm is built on a long-term aim. It is adaptable, thorough, and focused on a short-term perspective. The CRISP-DM methodology, as per shown in Figure 9, emphasizes a cyclic process for all the involved. The initial and critical step is business understanding. Subsequently, processes consist of data understanding, data preparation, modelling, evaluation, and deployment of the model. The business understanding process has multiple inputs, including explicit feeds from data understanding and evaluation [22]. Using the CRISP-DM methodology, users may plan a data mining project in a more organised and systematic manner. It is a method that is both trustworthy and well-proven. However, we are evangelists of its robust practicality, flexibility, and usefulness when using analytics to solve thorny business issues. It is the golden thread that runs through almost every client engagement.

3.2 MLPR System Architecture

Figure 16 provides the design process of the proposed Malaysian License Plate Recognition in depth. To begin with, the algorithm utilizes the Convolutional Neural Networks (CNN) for image recognition and detection. Optical Character Recognition (OCR) for character and number recognition [23]. The recognition of Malaysian plate recognition is divided into several main stages: Localization of license plate and character segmentation, train the deep learning model and followed by OCR matching. The listed phases are the main steps in the machine learning process. In the first stage is collect the data and label the data. Then, license plate is identified and localized in the scene and improve plate visual using pre-processing techniques. Next, characters are segmented from the detected license plate. The vehicle plate itself serves as the sole piece of usable information for identification, therefore in this situation, the license plate itself serves as the sole piece of valuable information for identification. Then CNN image recognition technique by implementing Inception-ResNet-v2 will be utilised to train the model with the set data of licence plate that have been collected. Follows by the OCR character and number recognition method. These two separate algorithms with different functions to perform due improve the capabilities of the MLPR system, several deep learning algorithms were developed and deployed. When it reaches its final step in modelling, OCR converts each character into encoded text information, which is then used to turn text in the Region of Interest into the desired output. Lastly, the application deployed into web application by creating REST API using flask. It ensures the system running just by uploading the image license plate the page will automatically appear the results.

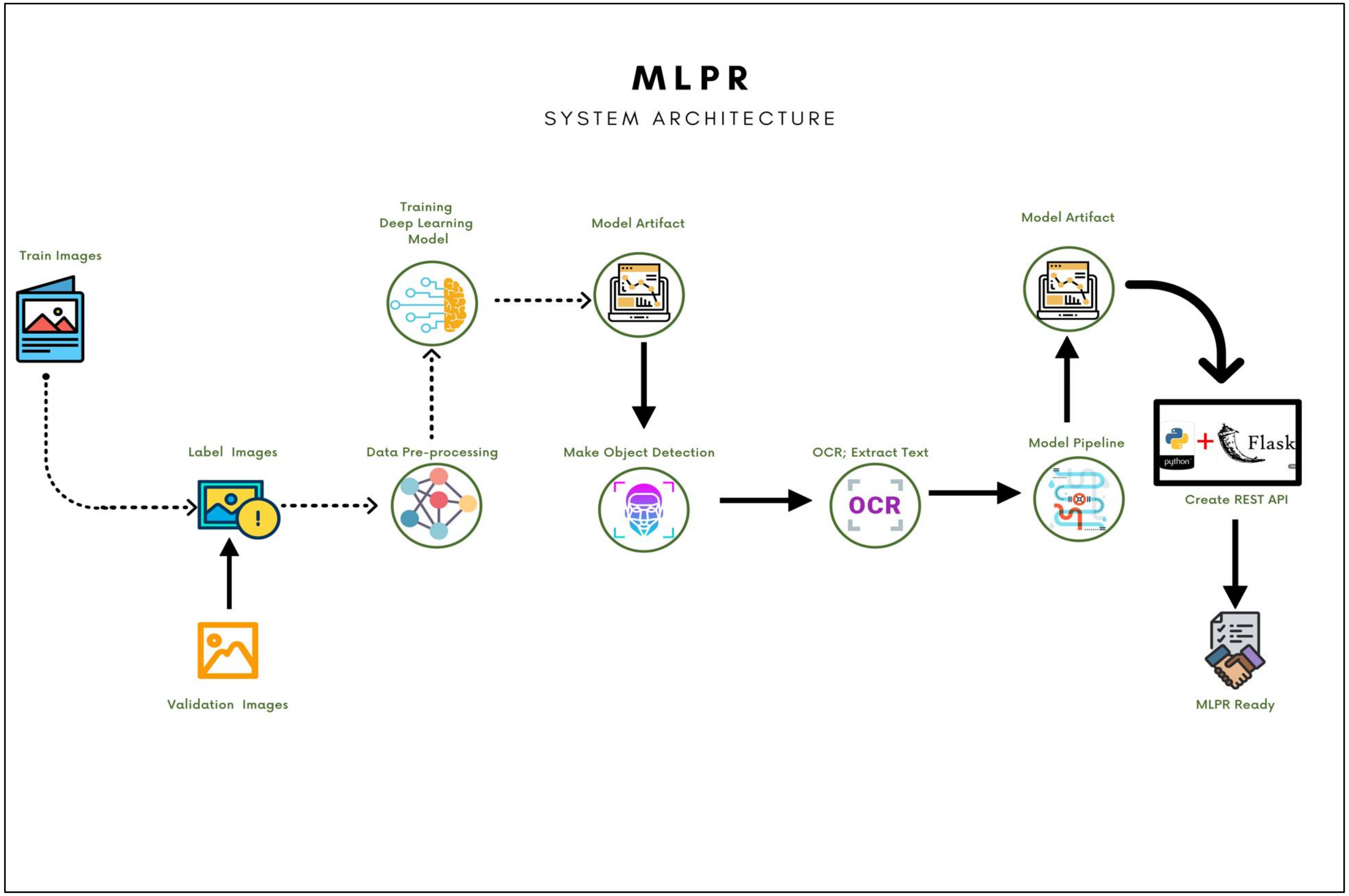


Figure 16.MLPR System Architecture

3.3 MLPR System Flow

Figure 17 provides the system flow of the proposed Malaysian License Plate Recognition. To begin, the algorithm utilizes the Convolutional Neural Networks (CNN) for image recognition and Optical Character Recognition (OCR) for character and number recognition [24].

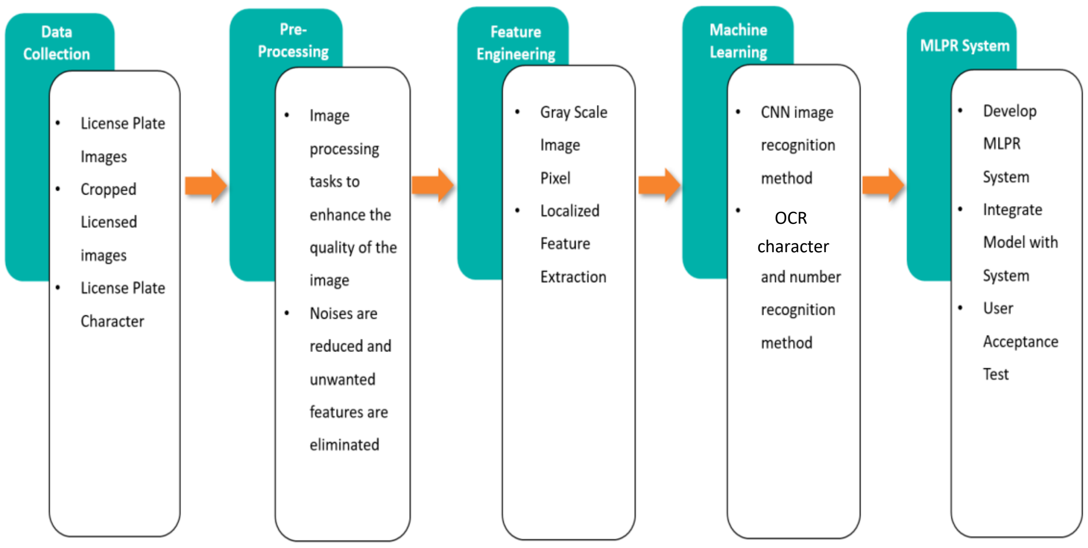

Data Collection
License Plate Images
Cropped Licensed images
License Plate Character
Pre-Processing
Image processing tasks to enhance the quality of the image
Noises are reduced and unwanted features are eliminated
Feature Engineering
Gray Scale Image Pixel
Localized Feature Extraction
Machine Learning
CNN image recognition method
OCR character and number recognition method
MLPR System
Develop MLPR System
Integrate Model with System
User Acceptance Test

Figure 17. MLPR System Flow

3.4 Pre-processing and Feature Engineering

One of the primary image pre-processing procedures is feature selection, which seeks to eliminate irrelevant and redundant properties for a given data collection depending on the uploaded data [25]. In any machine learning model, to enhance its accuracy, it is crucial to start with pre-process the data. In this project, the images of license plates will be pre-processed by several processes. At this stage, noise is decreased, and undesirable elements are removed to alleviate CNN's Inception-ResNet-v2 workload during the subsequent image recognition stage. The following pre-processing processes are required for MLPR: licence plate localization, character segregation, character division, resizing and character liners, and normalising. Thus, this active phase enables the system to accurately identify images.

3.4.1 License Plate Localization

Prior to capturing the vehicle plate number and alphabet, the subject's full license plate must be traced. License plate detection attempts to locate the alphabet and number of the vehicle's licence plate in the image or video fed to the system. After converting the input video from colour RGB to grayscale (black, white, and grey), it will be applied to detect the license plate well. Haar's distinguishing feature is the veiled rectangular area on the image. It will recognise grayscale films and classify them based on the sum of the specified pixel intensities. Figure 1 illustrates the Haar feature classifier. According on the pixel intensity, it can be classed as lines or edges. Lines can be classed as A, B, and D in Figs. As a result, C in the preceding figure can be classed as a line.

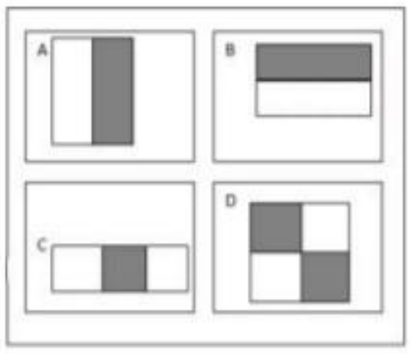


Figure 18.Feature Classifier

3.4.2 License Plate Characters Isolation

Characters in the license plate are isolated after determining the location of the license plate in the previous process. This process where the height and width will be calculated and cut using four specified angles, x min, x max, y min, and y max. The license plate image will be cut directly from the original RGB format input image and the license plate are converted to grayscale and then to binary for morphological operations. It is implemented to eliminate unwanted noise. In ensuring this process is successful morphological techniques are performed license indirectly. It is next followed by creating a rectangular structural element. Widening and erosion techniques are used to separate the foreground and background pixels of the license plate image. The widening process increases the size of the foreground pixels, while erosion reduces the size of the foreground pixels. The edge of the foreground pixel can be obtained by pushing both output processes. Furthermore, by altering the foreground object's diameter and colours, the foreground object will become clearer. Finally, the plate's and the surrounding area's characteristics may be recognized and filtered. Figure 19 depicts the procedure character isolation outcomes.

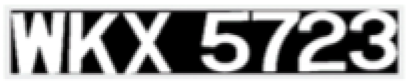


Figure 19.License plate isolation using morphological operation

3.4.3 License Plate Character Segmentation

Following the completion of the character isolation phase, the only elements left in the image of characters in the license plate and nothing else. Connected Component Analysis (CCA) process the license plate images being segmented. As well the characters in the image by identifying the pixels that are connected to one another. To make it simpler to recognise each character, each character is divided into an independent image. The projected result of the character segmentation method is displayed in Figure 20.

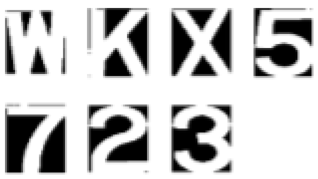


Figure 20. Segmented characters

3.4.4 License Plate Character Resize and Padding.

Post the license plate being segmented, the characters will be resized into 18×8-pixel size. It is to ensure to reduce the complexity of the images. Besides that, the image is further padding by 2 pixels on the surrounding of the image to become segmented characters will be resized to a size of 18×8 pixels in order to make the complex composition of the image more understandable. Next, the image overlays another 2 pixels on the edges around the license plate image. The result is a 22 × 12-pixel image [26]. Padding the images will be implemented to ensure all features are accessible during the recognition procedure. F. The image post padding and after it is layered process is shown in Figure 21.

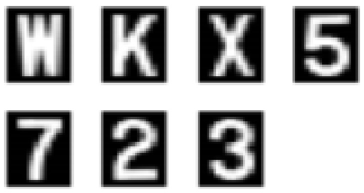


Figure 21. The character's image is resized and padded

3.4.5 Normalization

In the neural network recognition system in the CNN, images are trained using image numerical data. The accepted numerical data range is from -1 to 1. The implemented step is where the maximal normalization average is applied to the input image to resize it within the provided range Batch Normalization (BN) is a technique for increasing the efficiency and stability of neural networks as well as preventing fading gradients [27]. The preceding activation output was normalized by removing the group mean and then dividing by the group standard deviation by the overactive BN. As a result, each layer of the network will learn more independently than the others. The BN developed enables us to leverage larger learning rates while not being concerned with new firms. It also acts as a coordinator, reducing the requirement for dropouts and enhancing neural network stability [28].

3.5 Create the Base Model from The Pre-Trained Convnets

Transfer learning is the most popular approach in deep learning. In this model, pre-trained models utilised as the starting point on computer vision techniques. The Inception-ResNet-v2 base model is part of deep learning model which called convolutional neural network that has been trained on more than a million images from the ImageNet database, which is a large library of images [29]. The network is 164 layers deep and can classify images into 1000 object categories, such as the monitor, pen, mouse, pencil, and sea creatures. It is derived from the Inception structure and the Residual connection. Multiple convolutional filters of various sizes are mixed with residual connections in the Inception-Resnet block. The Figure 22 shows the schematic diagram on how Inception-ResNet-v2 interpreted.

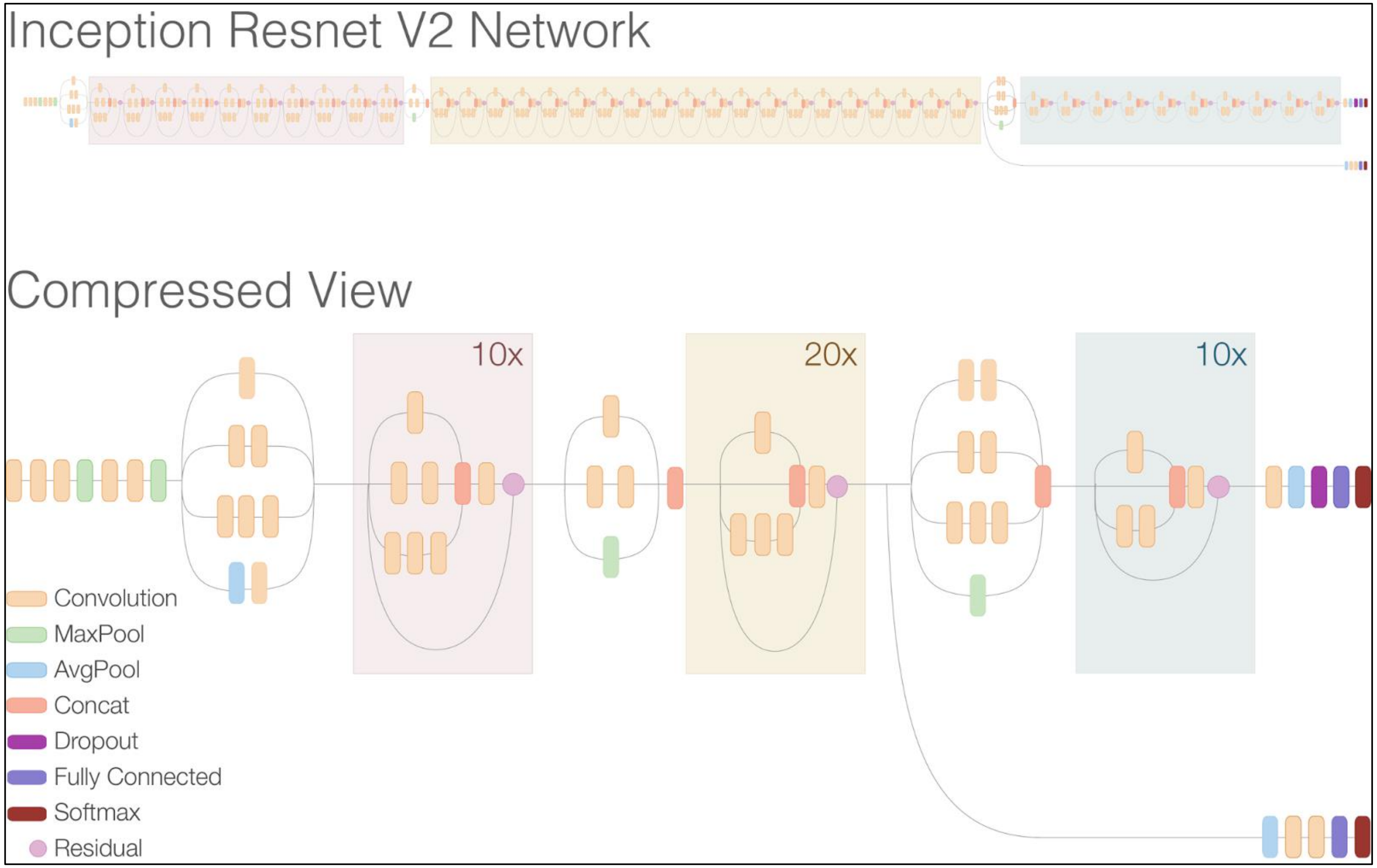
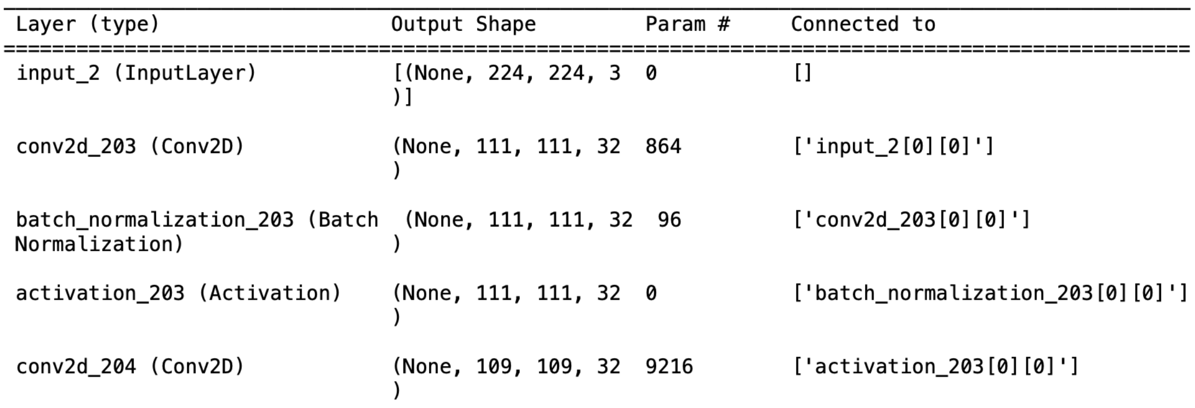


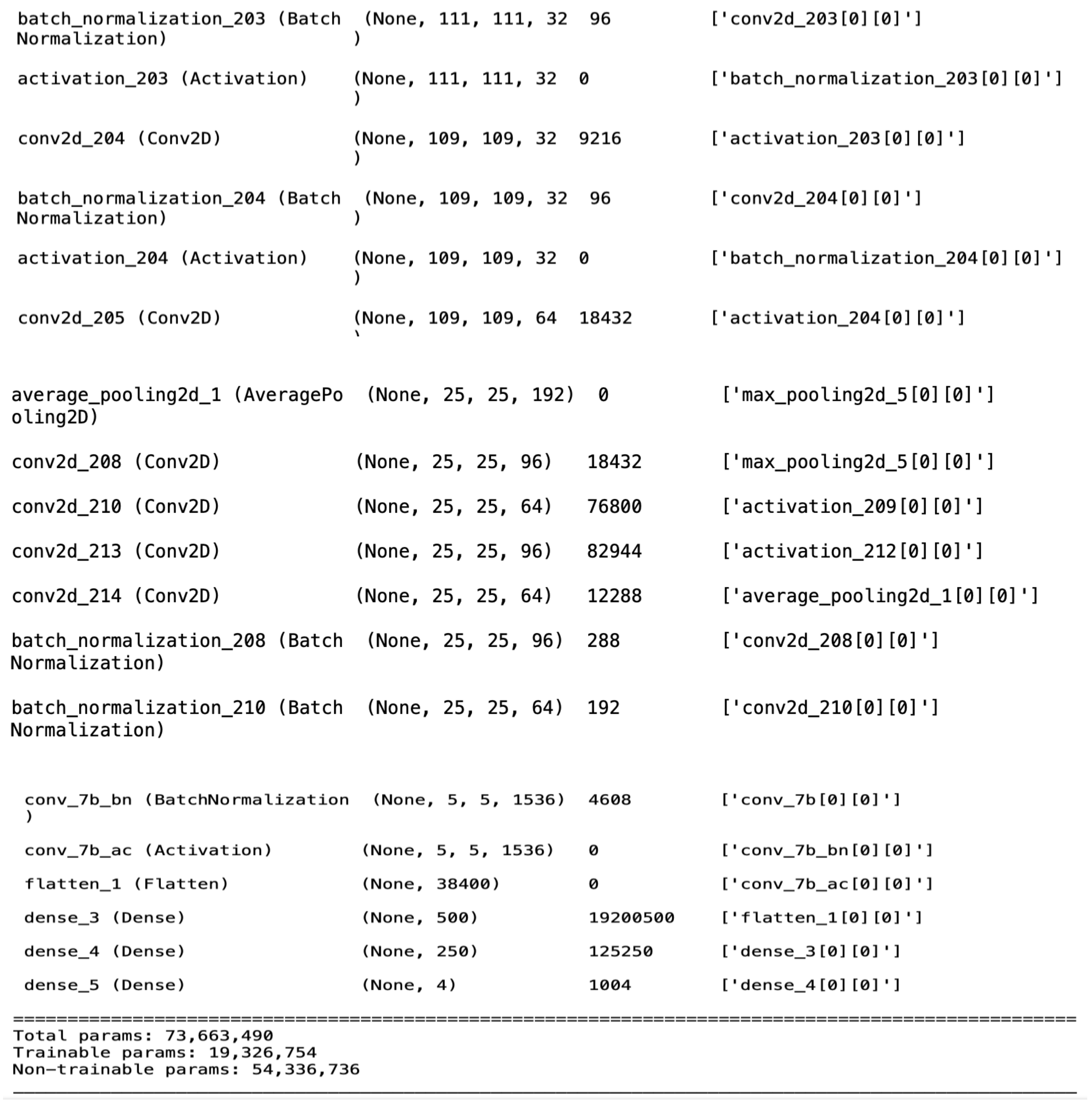
Figure 22.Schematic diagram of InceptionResNetV2 model

3.5.1 Compile the Model and Select Optimizer

Adam optimization “tf. keras. optimizers. Adam” is a stochastic gradient descent approach that is based on adaptive estimate of first-order and second-order moments in the first and second orders. Adam is the best among the adaptive optimizers in most of the cases especially in this image recognition. Thus, for this model Adam optimizer being utilised. Good with sparse data: the adaptive learning rate is perfect for this type of datasets. The approach is computationally efficient, has low memory demand, is invariant to diagonal rescaling of gradients, and is well suited for big data/parameter issues. [30]. The learning rate has been set to 1e-4 which equivalent to 0.0001 as a decimal number. Because of the high-level learning rate, it is possible for the model to train and test the data quickly, but it is at the risk of arriving at a less-than-optimal final set of weights in the process which lead less accuracy level. A slower learning rate may allow the model to learn a more optimum, or perhaps globally optimal, set of weights, but it will take much longer to train. There are big layers of convolutional that have been implemented. From the neural network layers itself, there are three additional layer which is “flatten”,” Dense_1” and “Dense_2”. The action's goal is Flattening is the process of converting data to a one-dimensional array for input to the next layer. The output of the convolutional layers is flattened to form a single long feature vector. Additionally, it is connected to the final recognition model, forming what is referred to as a fully connected layer. While the dense functionality is each neuron in the dense layer receives all outputs from the preceding layer, with each neuron providing one output to the next layer. Figure 23 shows a part of model summary of Inception-ResNet-v2.



```
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_2 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv2d_203 (Conv2D)            (None, 111, 111, 32  864         ['input_2[0][0]']
                                )

 batch_normalization_203 (Batch  (None, 111, 111, 32  96         ['conv2d_203[0][0]']
 Normalization)                 )

 activation_203 (Activation)    (None, 111, 111, 32  0           ['batch_normalization_203[0][0]']
                                )

 conv2d_204 (Conv2D)            (None, 109, 109, 32  9216        ['activation_203[0][0]']
                                )
```

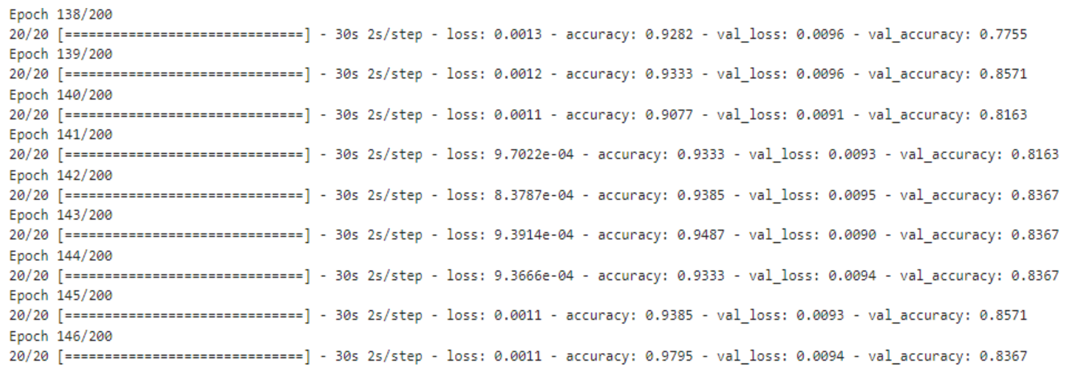


```
batch_normalization_203 (Batch  (None, 111, 111, 32  96        ['conv2d_203[0][0]']
Normalization)                  )

activation_203 (Activation)    (None, 111, 111, 32  0         ['batch_normalization_203[0][0]']
                               )

conv2d_204 (Conv2D)            (None, 109, 109, 32  9216      ['activation_203[0][0]']
                               )

batch_normalization_204 (Batch  (None, 109, 109, 32  96        ['conv2d_204[0][0]']
Normalization)                  )

activation_204 (Activation)    (None, 109, 109, 32  0         ['batch_normalization_204[0][0]']
                               )

conv2d_205 (Conv2D)            (None, 109, 109, 64  18432     ['activation_204[0][0]']


average_pooling2d_1 (AveragePo  (None, 25, 25, 192)  0         ['max_pooling2d_5[0][0]']
oling2D)

conv2d_208 (Conv2D)            (None, 25, 25, 96)   18432     ['max_pooling2d_5[0][0]']

conv2d_210 (Conv2D)            (None, 25, 25, 64)   76800     ['activation_209[0][0]']

conv2d_213 (Conv2D)            (None, 25, 25, 96)   82944     ['activation_212[0][0]']

conv2d_214 (Conv2D)            (None, 25, 25, 64)   12288     ['average_pooling2d_1[0][0]']

batch_normalization_208 (Batch  (None, 25, 25, 96)  288       ['conv2d_208[0][0]']
Normalization)

batch_normalization_210 (Batch  (None, 25, 25, 64)  192       ['conv2d_210[0][0]']
Normalization)


conv_7b_bn (BatchNormalization  (None, 5, 5, 1536)  4608      ['conv_7b[0][0]']
)

conv_7b_ac (Activation)        (None, 5, 5, 1536)   0         ['conv_7b_bn[0][0]']

flatten_1 (Flatten)            (None, 38400)        0         ['conv_7b_ac[0][0]']

dense_3 (Dense)                (None, 500)          19200500  ['flatten_1[0][0]']

dense_4 (Dense)                (None, 250)          125250    ['dense_3[0][0]']

dense_5 (Dense)                (None, 4)            1004      ['dense_4[0][0]']

==================================================================================================
Total params: 73,663,490
Trainable params: 19,326,754
Non-trainable params: 54,336,736
__________________________________________________________________________________________________
```

Figure 23. Model Summary

3.5.2 Training Inception-ResNet-v2

The term "epoch" refers to a complete iteration over samples. The number of epochs specifies how frequently the algorithm will execute. The number of epochs has a direct (or indirect) influence on the outcome of the training phase; with few epochs, users can only obtain a local minimum accuracy and high losses. However, with more epochs, users can reach a global minimum or at least a better local minimum. In MLPR training of the model was separated into 200 epochs. After 200 epochs results was collected into model history. Figure 24 below illustrates on a part of 200 epochs training model:

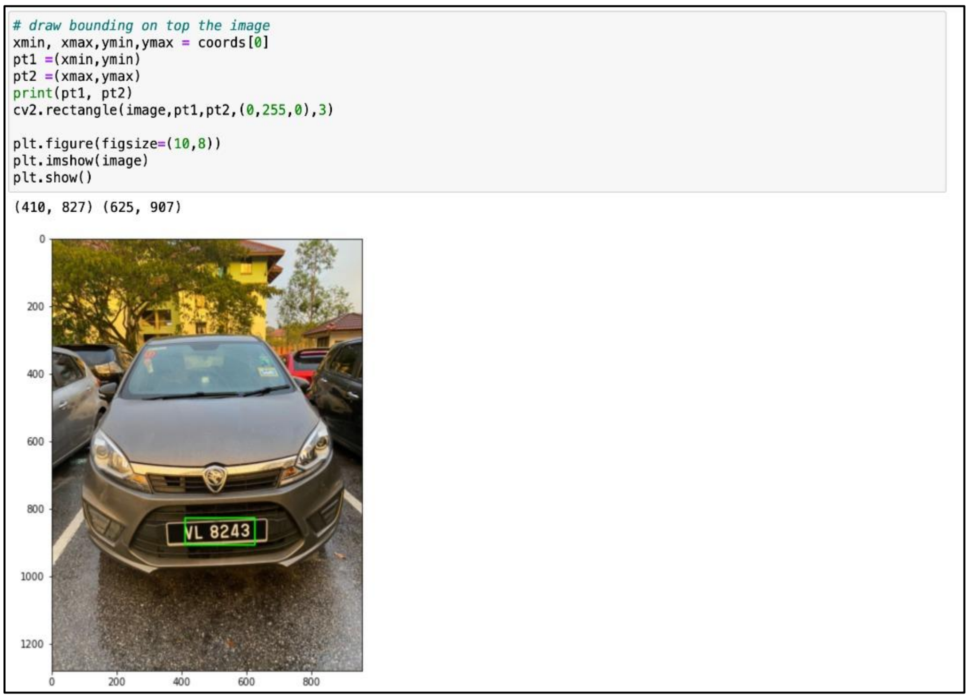


```
Epoch 138/200
20/20 [==============================] - 30s 2s/step - loss: 0.0013 - accuracy: 0.9282 - val_loss: 0.0096 - val_accuracy: 0.7755
Epoch 139/200
20/20 [==============================] - 30s 2s/step - loss: 0.0012 - accuracy: 0.9333 - val_loss: 0.0096 - val_accuracy: 0.8571
Epoch 140/200
20/20 [==============================] - 30s 2s/step - loss: 0.0011 - accuracy: 0.9077 - val_loss: 0.0091 - val_accuracy: 0.8163
Epoch 141/200
20/20 [==============================] - 30s 2s/step - loss: 9.7022e-04 - accuracy: 0.9333 - val_loss: 0.0093 - val_accuracy: 0.8163
Epoch 142/200
20/20 [==============================] - 30s 2s/step - loss: 8.3787e-04 - accuracy: 0.9385 - val_loss: 0.0095 - val_accuracy: 0.8367
Epoch 143/200
20/20 [==============================] - 30s 2s/step - loss: 9.3914e-04 - accuracy: 0.9487 - val_loss: 0.0090 - val_accuracy: 0.8367
Epoch 144/200
20/20 [==============================] - 30s 2s/step - loss: 9.3666e-04 - accuracy: 0.9333 - val_loss: 0.0094 - val_accuracy: 0.8367
Epoch 145/200
20/20 [==============================] - 30s 2s/step - loss: 0.0011 - accuracy: 0.9385 - val_loss: 0.0093 - val_accuracy: 0.8571
Epoch 146/200
20/20 [==============================] - 30s 2s/step - loss: 0.0011 - accuracy: 0.9795 - val_loss: 0.0094 - val_accuracy: 0.8367
```

Figure 24.Part of 200 epochs training model

3.5.3 Draw Bounding Box

A bounding box is a fictional rectangle that serves as a reference point for object detection and creates a collision box when a license plate is identified. When labelling an image, data is annotated to construct these rectangles across images, indicating the X and Y coordinates of the object of interest inside each image. This makes the role of machine learning algorithms easier in terms of finding objects, determining collision paths, and preserving valuable computer resources. Figure 25 shows how the bounding box is constructed using the following:



```python
# draw bounding on top the image
xmin, xmax,ymin,ymax = coords[0]
pt1 =(xmin,ymin)
pt2 =(xmax,ymax)
print(pt1, pt2)
cv2.rectangle(image,pt1,pt2,(0,255,0),3)

plt.figure(figsize=(10,8))
plt.imshow(image)
plt.show()
```

```
(410, 827) (625, 907)
```

Figure 25. Forming Bounding Box

3.6 Character Recognition using OCR

After pre-processing, the process is character recognition, which is the last stage in the modelling. The optical character recognition approach is a sort of recognition in which an image is used as input and a string of characters is output. Optical character recognition (OCR) is a procedure that distinguishes between distinct characters extracted from an image. Template matching is one of the procedures used in optical character recognition. In this step, the cropped image is compared to the template data that has been recorded in the database. OCR is a technology that automatically detects and recognises characters without the need for any indirect input. When the characters on the license plate have uniform typefaces, the OCR for license plate is less difficult when compared to other techniques of recognition.

3.6.1 Using Feature Extraction to Recognize Characters

In this method, the fact that each character has a unique collection of properties such as corners, ends, and bifurcations is taken advantage of in term of recognition. The algorithm becomes faster and less difficult because of inheriting these features. A technique of iteration is used to turn the input character into an edge picture, and then to extract the features from it. The number, direction, and state of the features are then recorded in the feature vector, along with the number of features in the feature vector. It is detailed in depth in the following Figure 26:

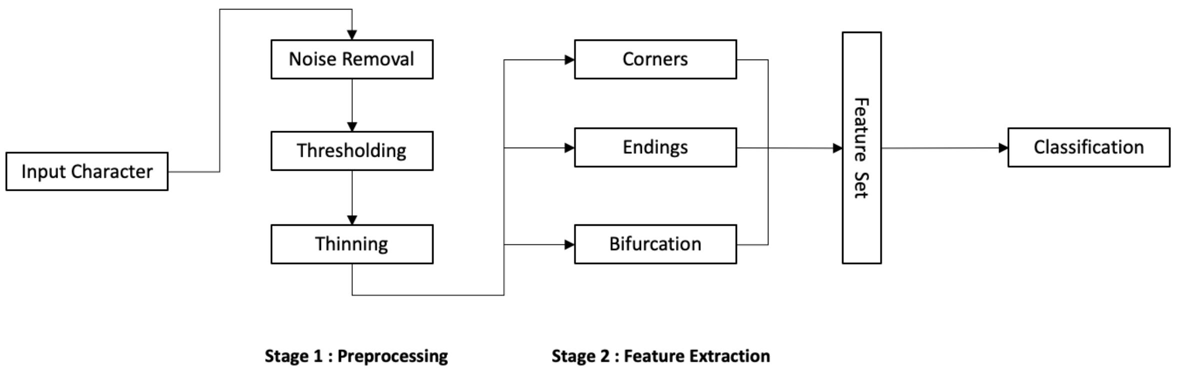


Figure 26.Feature Extraction: OCR

Another character characteristic was obtained using the i-novel technique proposed and applied by Sushruth is one of the best ways to implement [26]. The process which based on the segment titled feature set. Then, by tracing the image segments in numerous directions and recording the projection values, each character becomes distinct in terms of its one-dimensional vector. As a result, character features were derived utilizing a unique width analysis technique. When viewed from the edge of a segmented character picture, each character has its own set of lines or slopes. This approach makes efficient use of this characteristic to identify characters. After dividing the character, it is extended such that it contacts the border on all sides of the license plate. The distance between the first black pixel in each letter and the left border is then calculated. The distance between the first dark pixel and the right border is calculated in the same way. Figure 27 shows an illustration of this approach. The resulting feature set has dimensions (2xh) represents the height of the resized character. The height of each character picture should be consistent.

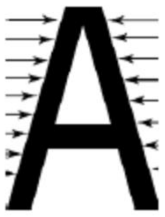


Figure 27. Character's physical distance from the other characters

The border feature set may now be used to define the character width for each row. The width of the character is defined as the distance between the leftmost and rightmost dark pixels taken. The pixels taken from the border feature collection. This produces a feature set for (lh). This stage is critical since it normalises the feature set of character widths. Normalization is essential since segment will be decreased while maintaining the aspect ratio of image. Thus, while the height of each character is constant, the width varies according on the character's kind. For example, the characters W and H have the greatest width, while the characters I and 1 have the smallest width. As a result of normalizing the feature set, character width features range from 0 to 1. The feature set may be implemented as shown in Figure 28.

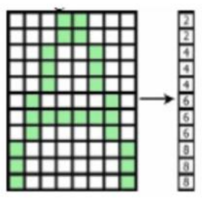


Figure 28. Feature vector extraction of character

During the character recognition phase, the template matching techniques being utilised. The template could contain any languages such as English, Arabic, Hindi, and others. This process crucial because it enables for ranging from large with a greater level of accuracy, even with low quality images and bad angles. This method makes advantage of a clear two-dimensional link with other character templates [31].

3.7 MLPR Deployment

To create a user-friendly MLPR system, the system will be deployed into a web application. The system is developed in Python Flask software. The deployment to ensure end users can utilize the application virtually and have good flexibility. Front end developed for the Graphical User Interface (GUI) of the system representation. The platform offers a user-friendly interface and contributes to the system's long-term viability by increasing its sustainability. It is classified as a microframework because it does not necessitate the usage of any specific tools or libraries. Users may rapidly identify their license plate by just simply upload image in the application. Without a doubt, the production can be created in a very brief length of time. The interactive layout of web applications is depicted in the Figure 29 below.

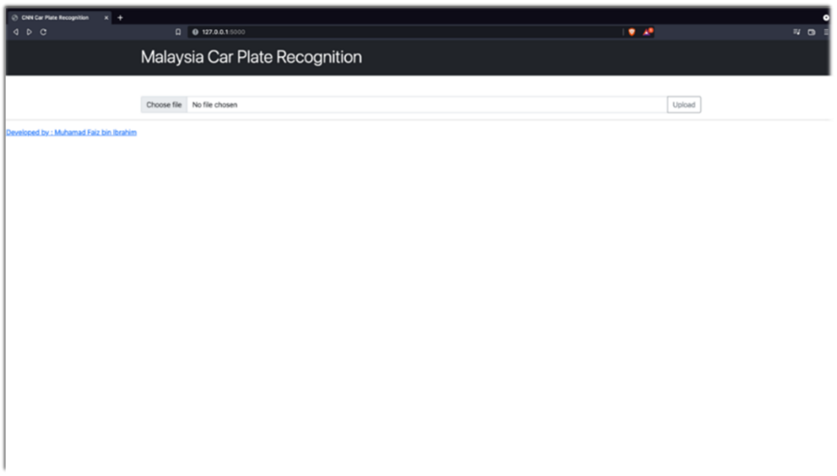

CNN Car Plate Recognition
127.0.0.1:5000
Malaysia Car Plate Recognition
Choose file
No file chosen
Upload
Developed by : Muhamad Faiz bin Ibrahim

Figure 29.MLPR System Website Layout

3.7.1 Create REST API for MLPR

The deep learning model developed by CNN has been integrated into the web application. The Representative State Transfer Application Programming Interface (REST API) allows for the development of website applications (API). The usage of Flask is essential in the development of this application. It is a method of communication between a user as a client and a server in which the user requests something from the server, and the server responds with the number of the detected and recognized plate. Figure 30 depicts the operation of the REST API:

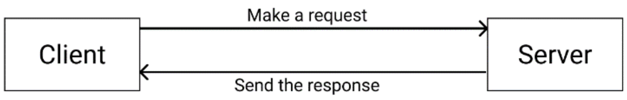
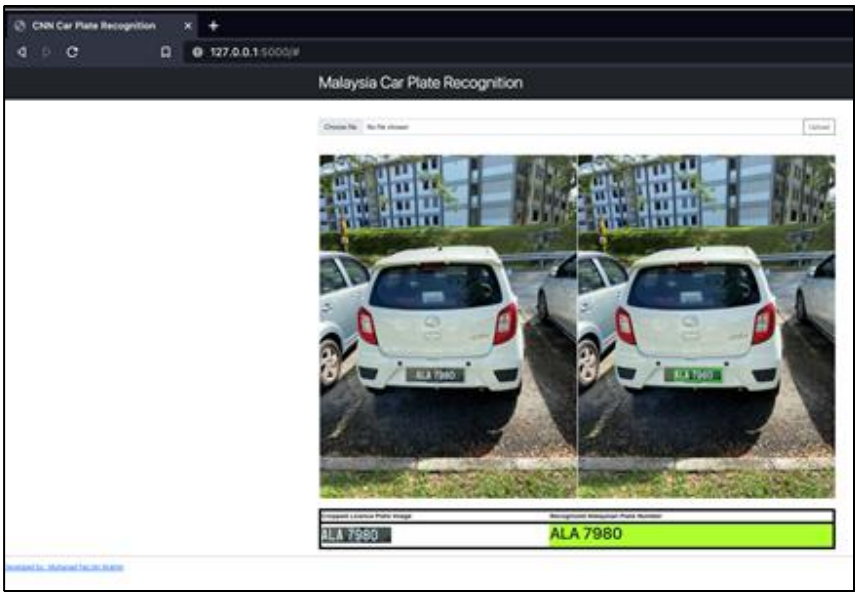


Figure 30.REST API Framework

Flask is used to create REST API, the methods that we will be discussing are as follows: To begin with, import the library and load the Inception-ResNet-v2 machine learning model from the model library. After that, create routines for preparing images of license plates and for recognizing license plate images. Following that, start the flask object. Setting up a route and a function that returns anything to the user's browser should follow that. Finally, make use of the API and test it on local host. It is executed and the API is tested in accordance with figure 31.



ALA 7980

Figure 31.Run and tested API

3.8 Software Utilization

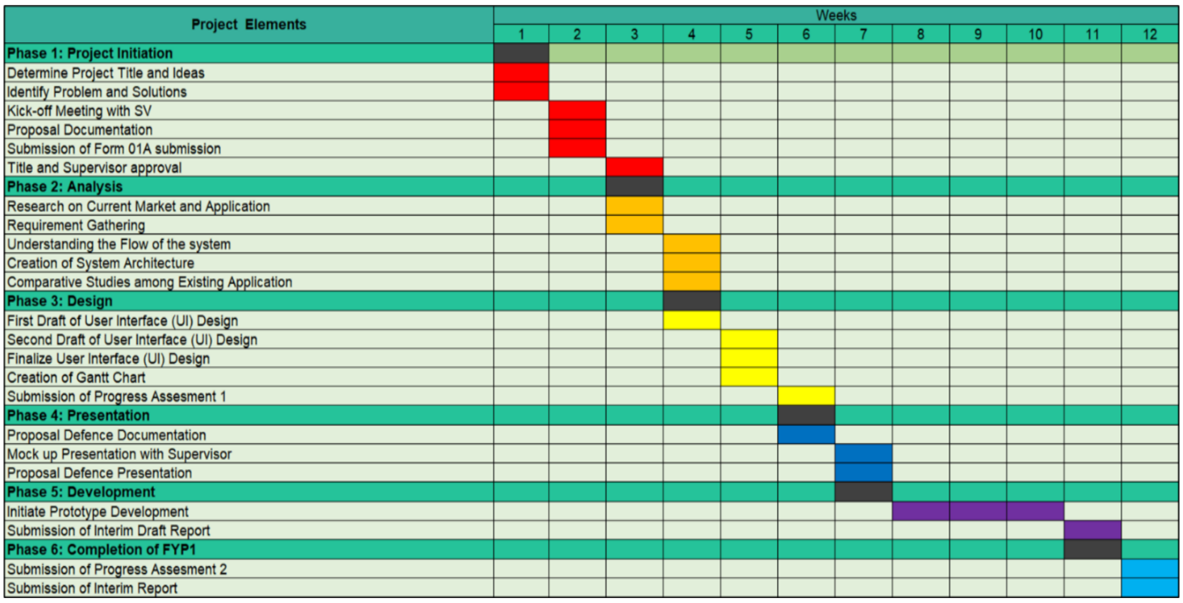
The software used in this MLPR algorithm in the Python programming language, the latest version 3.10 is the most important base. The NumPy, Pandas, Matplotlib, TensorFlow and CV2 libraries were all installed separately and explicitly adapted to the project because the libraries were dedicated to ensuring effectiveness plate recognition procedures. The IDEs for editing line code are Visual Studio Code for web development and Jupyter Notebook for modelling. For potential benchmarking purposes, a data set will be taken around Universiti Teknologi PETRONAS. Figure 31. shows the set of personal data that collected around the Universiti Teknologi PETRONAS.

Figure 32. Dataset gathered around UTP

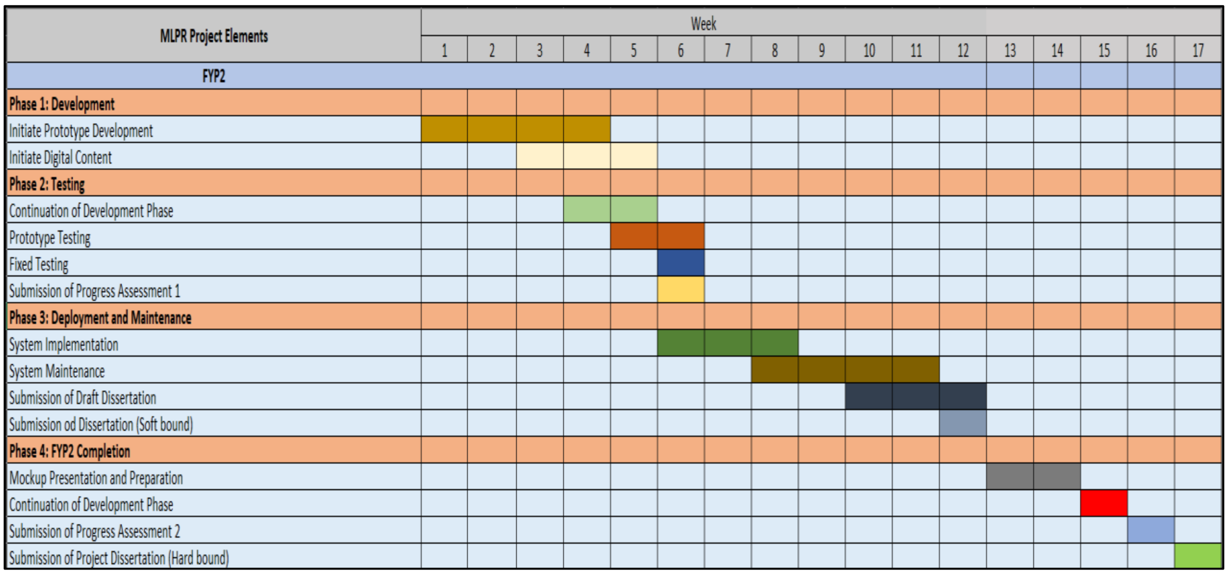
3.9 Gantt Chart

Table 3 presents a Gantt chart summarizing the project flow for Final Year Project 1 and Final Year Project 2.

Table 3.Gantt Chart for MLPR Project



Project Elements	Weeks											
	1	2	3	4	5	6	7	8	9	10	11	12
Phase 1: Project Initiation	⬛											
Determine Project Title and Ideas	🟥											
Identify Problem and Solutions	🟥											
Kick-off Meeting with SV		🟥										
Proposal Documentation		🟥										
Submission of Form 01A submission		🟥										
Title and Supervisor approval			🟥									
Phase 2: Analysis			⬛									
Research on Current Market and Application			🟧									
Requirement Gathering			🟧									
Understanding the Flow of the system				🟧								
Creation of System Architecture				🟧								
Comparative Studies among Existing Application				🟧								
Phase 3: Design				⬛								
First Draft of User Interface (UI) Design				🟨								
Second Draft of User Interface (UI) Design					🟨							
Finalize User Interface (UI) Design					🟨							
Creation of Gantt Chart					🟨							
Submission of Progress Assesment 1						🟨						
Phase 4: Presentation						⬛						
Proposal Defence Documentation						🟦						
Mock up Presentation with Supervisor							🟦					
Proposal Defence Presentation							🟦					
Phase 5: Development							⬛					
Initiate Prototype Development								🟪	🟪	🟪		
Submission of Interim Draft Report											🟪	
Phase 6: Completion of FYP1											⬛	
Submission of Progress Assesment 2												🟦
Submission of Interim Report												🟦



MLPR Project Elements	Week																
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
FYP2																	
Phase 1: Development																	
Initiate Prototype Development	🟫	🟫	🟫	🟫													
Initiate Digital Content			🟨	🟨	🟨												
Phase 2: Testing																	
Continuation of Development Phase				🟩	🟩												
Prototype Testing					🟧	🟧											
Fixed Testing						🟦											
Submission of Progress Assessment 1						🟨											
Phase 3: Deployment and Maintenance																	
System Implementation						🟩	🟩	🟩									
System Maintenance								🟫	🟫	🟫	🟫						
Submission of Draft Dissertation										⬛	⬛	⬛					
Submission od Dissertation (Soft bound)												🟦					
Phase 4: FYP2 Completion																	
Mockup Presentation and Preparation													⬛	⬛			
Continuation of Development Phase															🟥		
Submission of Progress Assessment 2																🟦	
Submission of Project Dissertation (Hard bound)																	🟩

3.8 Milestone

Figure 32 displays the critical project milestones that will be used to develop this Malaysian License Plate Recognition.

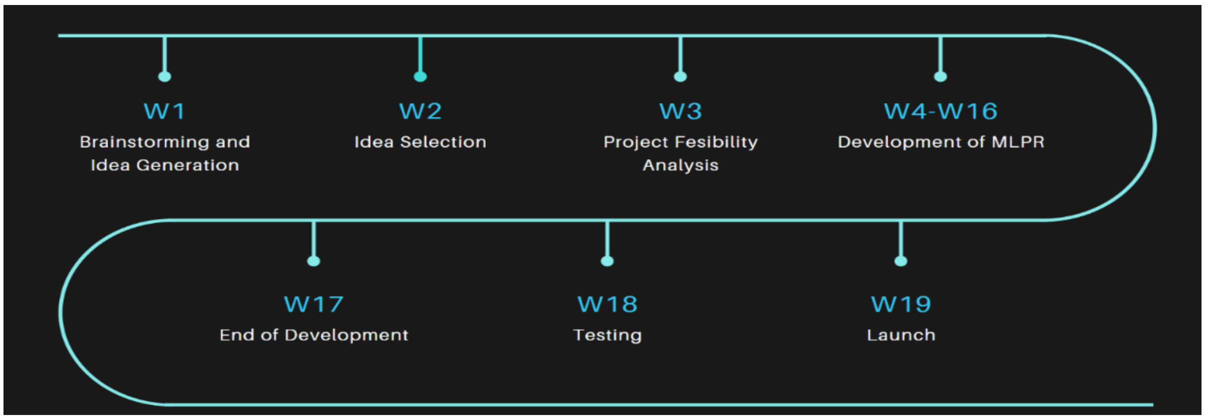


Figure 33. Key Project Milestone

CHAPTER 4
RESULT AND DISCUSSION

4.1 Dataset

This licence plate identification algorithm was evaluated using a private dataset of cars in the academic and hostel buildings of Universiti Teknologi PETRONAS. For training, data gathered from open-source car plate library which contains 265 images. The dataset displayed in Figure 34 was acquired using an iPhone 11 camera from a range of vehicle subjects, including a variety of various vehicle types. The images obtained have a width and height of 1280 x 720 pixels. A total of sixteen images of autos from various vantage points. This is done to ascertain the model's correctness.

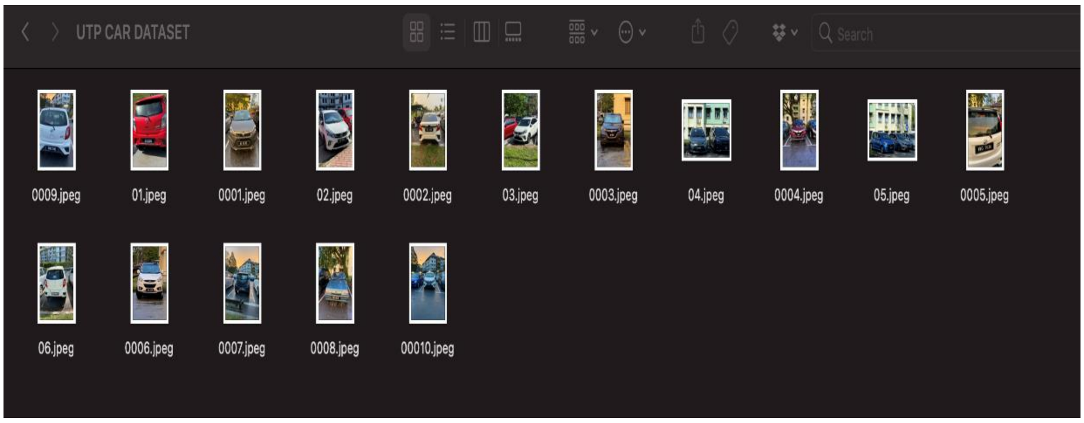


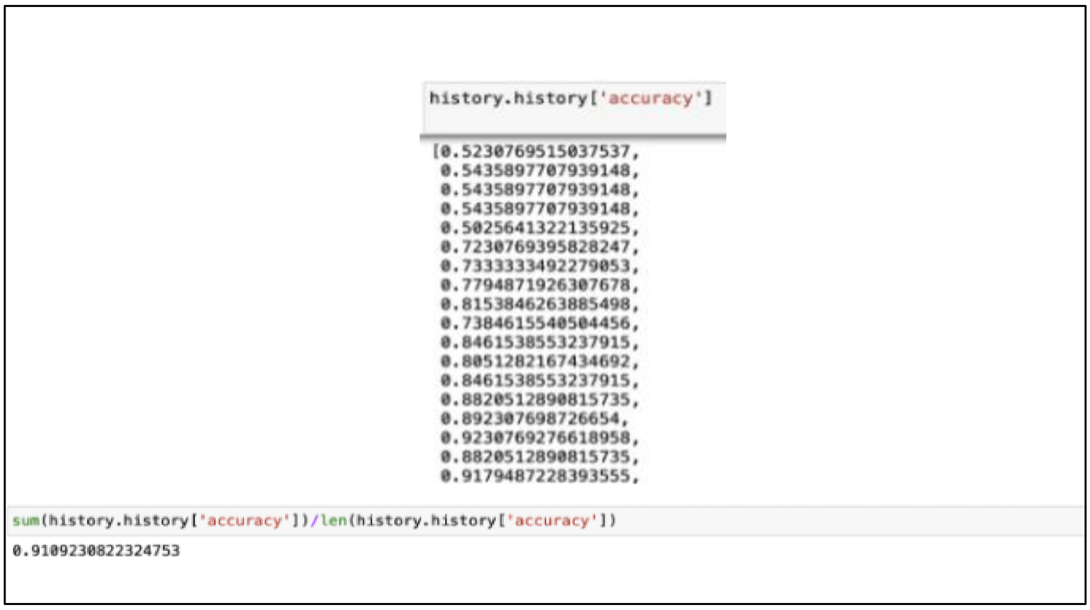
Figure 34. Image Dataset

4.2 Results of Convolutional Neural Network and OCR Algorithm Application

The findings of the system validation module are presented in this section. Results of the CNN application, it shows how the plate number is detected and recognized. Region of Interest (ROI) selection of license plate is also displayed. A high accuracy rating guarantees that number plate recognition requirements are met.

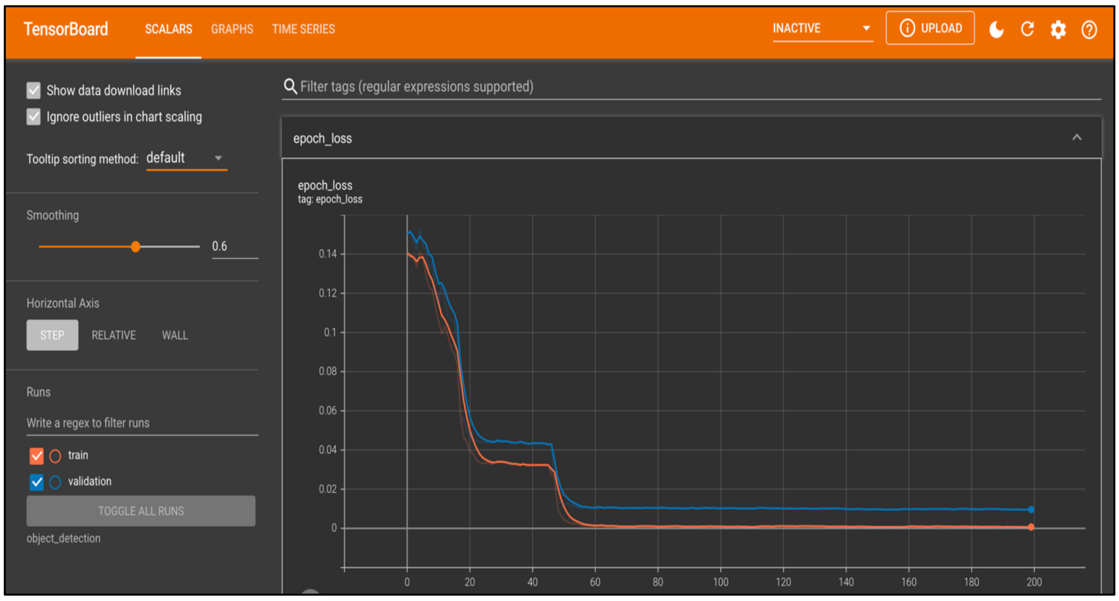
4.2.1 Accuracy of Model

Loss is defined as the error over the training set, which is commonly expressed as mean squared error (for regression). To guarantee that the model recognises the vehicle plate number accurately, it is trained and validated to ensure that it suffers the fewest number of losses possible. The epoch loss visualized by implementing TensorFlow extension which is Tensorboard. The platform itself very flexible and user interactive. The smallest amount of epoch losses indicates that the model is ready for deployment. The effectiveness of the methods was tested on computer called MacBook pro with spec Apple M1 chip 8-core GPU, Memory 8GB,256GB SSD and MacOS Operating system. Figure 36 illustrates that the epoch loss is at its lowest point after 200 epochs in the Tensorboard. Based on the 200-epoch 200 accuracy was recorded. To get the actual result of the average of 200 accuracy was taken. Figure 35 illustrate the result of average accuracy.



```
history.history['accuracy']

[0.5230769515037537,
 0.5435897707939148,
 0.5435897707939148,
 0.5435897707939148,
 0.5025641322135925,
 0.7230769395828247,
 0.7333333492279053,
 0.7794871926307678,
 0.8153846263885498,
 0.7384615540504456,
 0.8461538553237915,
 0.8051282167434692,
 0.8461538553237915,
 0.8820512890815735,
 0.892307698726654,
 0.9230769276618958,
 0.8820512890815735,
 0.9179487228393555,

sum(history.history['accuracy'])/len(history.history['accuracy'])
0.9109230822324753
```

Figure 35.Average Accuracy results



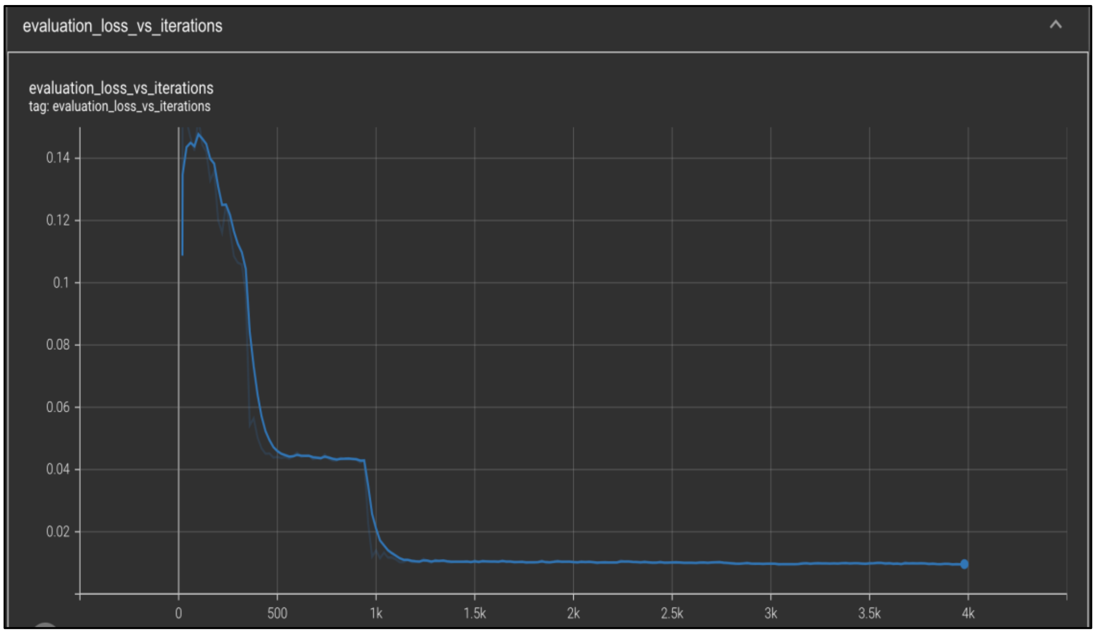


Figure 36.Epoch Loss Visualization

4.2.2 License plate Image Masking and Region of Interest (ROI)

The following step is a software simulation in which the Region of Interest (ROI) in the license plate topic data set is effectively recognised. Following vehicle detection, the subject number plate may be readily recognised by generating a ROI based on an array that records the position of points on the number plate's surface. The ROI selection and image masking application outcomes indicated an average accuracy of 100 percent since all license plate subjects passed the procedure even if the images was not in a decent angle, the algorithm successfully recognised ROI and applied mask needed. Figure 37 depicts the data set selection of Regions of Interest (ROI), and the bounding box is drawn around the number plate allowing it to be more clearly seen and detected.

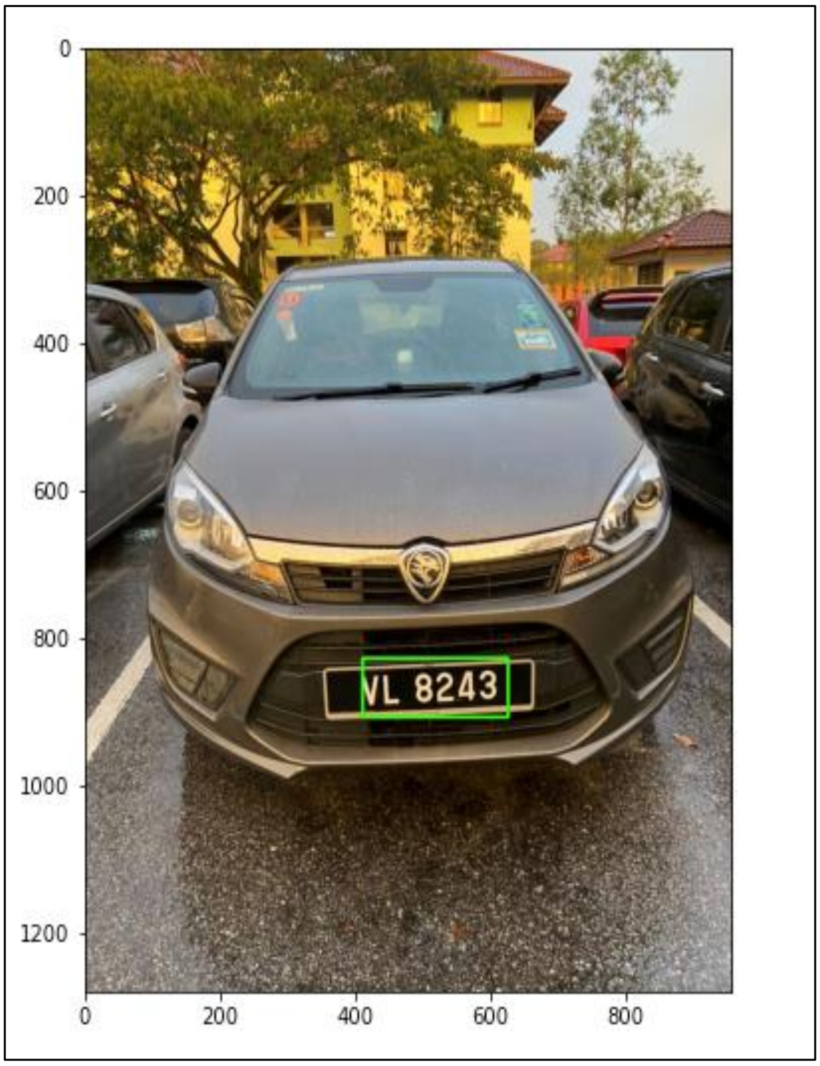


Figure 37. Region of Interest

4.2.3 License Plate Recognition

The simulation resulted in the successful implementation of the license plates recognition module in the private dataset of UTP car dataset subjects. The machine successfully detected and displayed the output license plate in this model. Figure 38 shows the image recognition by deep neural network successfully implemented. Figure 39 – Figure 41 displays the successful recognition in the dataset ranging from 16 images of private UTP car dataset, respectively. Generally, the OCR recognition process, recognized the characters from the image then convert it into text that can be stored in any way preferrable. This process is crucial part of the whole project. The text then, will be transmitted into database for further usage.

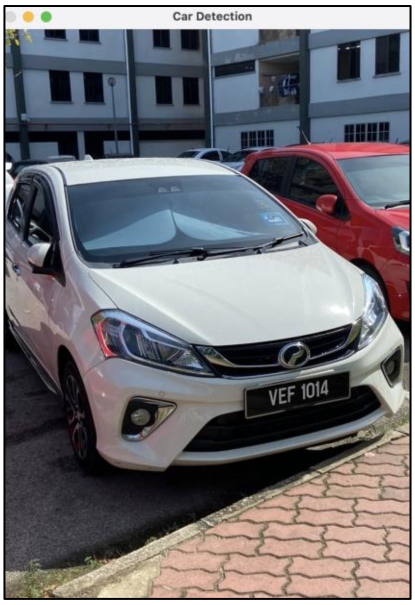
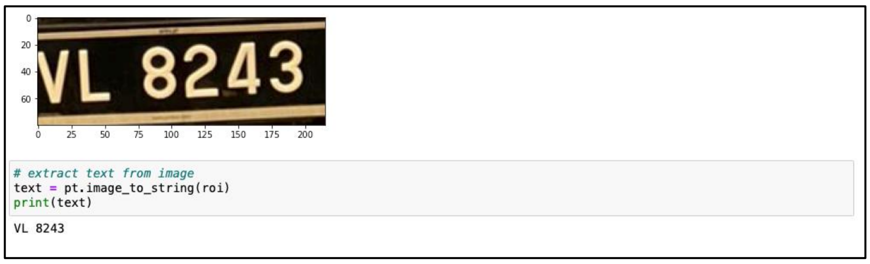


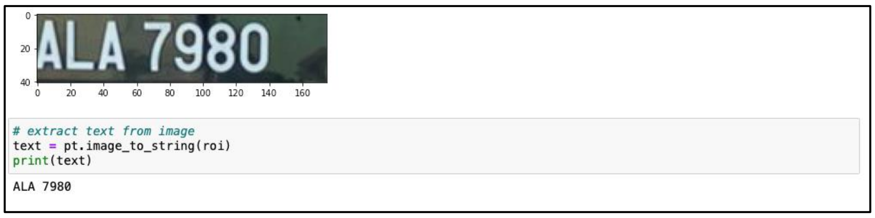
Figure 38. Image Detection by CNN



```
# extract text from image
text = pt.image_to_string(roi)
print(text)
```

```
VL 8243
```

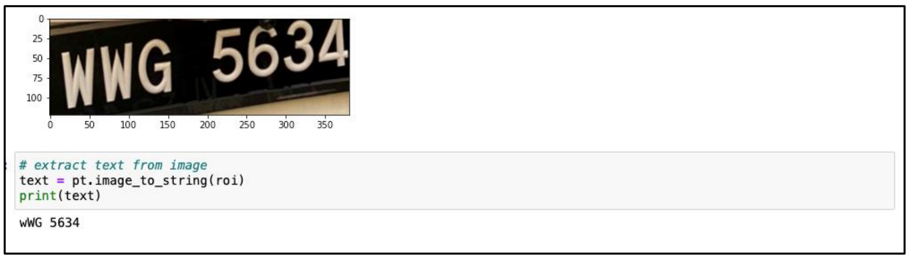
Figure 39.Plate Recognition by OCR



```
# extract text from image
text = pt.image_to_string(roi)
print(text)
```

```
ALA 7980
```

Figure 40.Plate Recognition by OCR



```
# extract text from image
text = pt.image_to_string(roi)
print(text)
```

```
wWG 5634
```

Figure 41.Plate Recognition by OCR

4.3 MLPR Web Application

This Figure 42. shows a piece of the output from the plate identification method. Certainly, it is uncomplicated, simple to work with and user-friendly. Users may submit images by tap the “upload” of their preferred licence plate numbers that they wish to be recognised using this online application. The images of license plate will be identified automatically by the programme itself in less than 2 seconds. The information will be maintained in a database for the purposes of toll management, security, AES, and other security services and among other things.

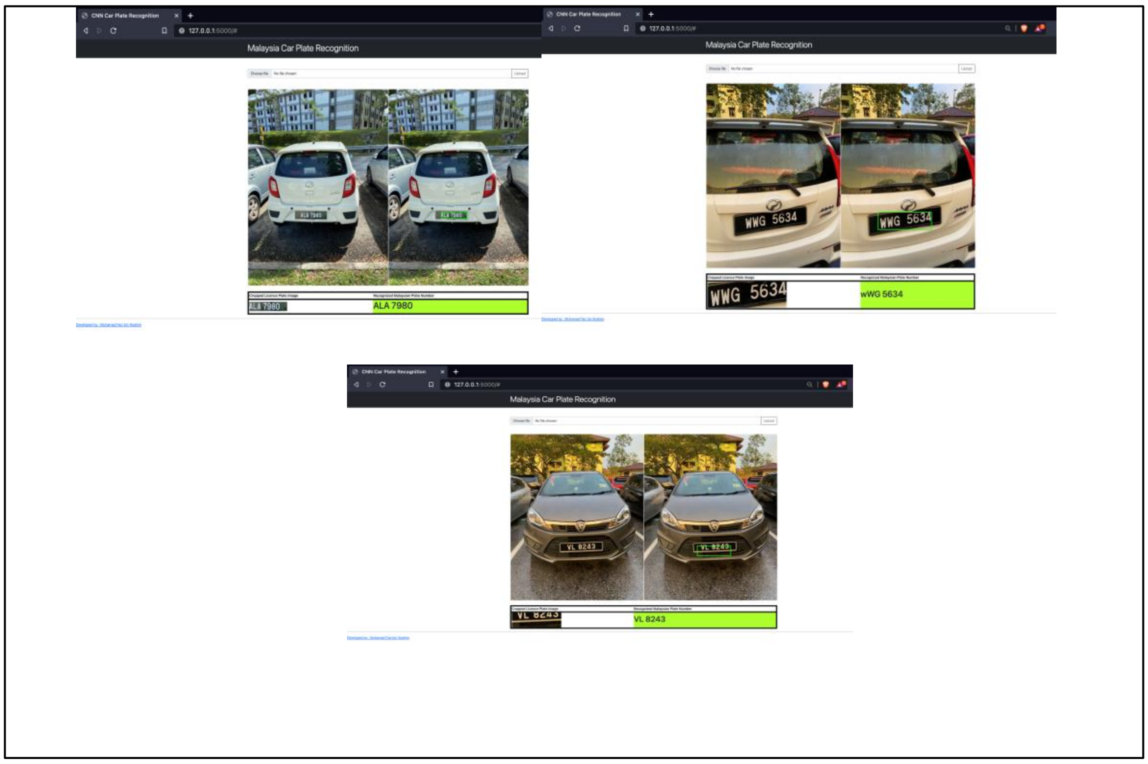

CNN Car Plate Recognition
127.0.0.1:5000/#
Malaysia Car Plate Recognition
ALA 7980
ALA 7980
CNN Car Plate Recognition
127.0.0.1:5000/#
Malaysia Car Plate Recognition
WWG 5634
wWG 5634
CNN Car Plate Recognition
127.0.0.1:5000/#
Malaysia Car Plate Recognition
VL 8243
VL 8243

Figure 42.MLPR Website User Interface

CHAPTER 5
CONCLUSION

5.1 Conclusion

In conclusion, this project developed Malaysia License Plate Recognition (MLPR) system using an enhanced deep neural network which is Convolutional Neural Network (CNN). The process for determining the optimal model for the four-layered CNN architecture utilised as the recognition method is the work's essential contribution. License plate recognition accuracy is crucial in developing an efficient License Plate Recognition system. This deep learning, neural network algorithm will be tested on a set of images collected in a private dataset. It aims to measure its effectiveness in detecting Malaysian plates for future real-time applications. The deep learning algorithm of plate recognition requires an image recognition system that works to determine the location of the plate number, followed by the division of characters and numbers and finally, the original recognition of the plate number. Next, to improve the accuracy of the MLPR, research and analysis based on comparative results on how to obtain better license plate recognition.

5.2 Future work on MLPR

Soon, the development of MLPR will be implemented, and the proposed algorithm will be integrated into real-time detection utilizing the camera. Multiple license plate detection libraries and live videos will be used to develop a real-time Malaysia License Plate Recognition algorithm. Next, the process is to complete the design of MLPR on increasing the accuracy and capabilities. Then, to meet the requirement of CRISP-DM by having an efficient deployment for a better business process. As stated in the description above, several aspects such as business understanding, data understanding, data preparation, modelling, evaluation, and deployment of MLPR must be focused on depth. It is to ensure that the development of MLPR can run smoothly without any problems. MLPR, in turn, can provide benefits to the community. The goal of MLPR's future development is to improve the user interface (UI) and user experience (UX), with a particular emphasis on the front-end and back-end development. The goal in the deep learning model is to detect and recognize license plates, it is to maximize its usability and ability to be the best application that can be customized and can be used for securities regulators and the public with extremely efficient and time-saving capabilities. Researchers and developers need to continue to develop a more in-depth and emphasize CNN architecture with a low number of parameters and a short learning period. Although training time may be reduced as a result of in -depth research, it should be remembered that the quality must be comparable to or greater than the previous architecture as the current accuracy is 91%. It is certain that it will be checked using hand characters from different languages such as Thai, Arabic, Pakistani and Hindi.

REFERENCES

[1] S. Kido, Y. Hirano, and N. Hashimoto, “Detection and classification of lung abnormalities by use of convolutional neural network (CNN) and regions with CNN features (R-CNN),” 2018 International Workshop on Advanced Image Technology, IWAIT 2018, pp. 1–4, May 2018, doi: 10.1109/IWAIT.2018.8369798.

[2] D. Arora, M. Garg, and M. Gupta, "Diving deep in Deep Convolutional Neural Network," Proceedings - IEEE 2020 2nd International Conference on Advances in Computing, Communication Control and Networking, ICACCCN 2020, pp. 749–751, Dec. 2020, doi: 10.1109/ICACCCN51052.2020.9362907.

[3] P. Marzuki, A. R. Syafeeza, Y. C. Wong, N. A. Hamid, A. Nur Alisa, and M. M. Ibrahim, "A design of license plate recognition system using convolutional neural network," International Journal of Electrical and Computer Engineering, vol. 9, no. 3, pp. 2196–2204, Jun. 2019, doi: 10.11591/IJECE.V9I3.PP2196-2204.

[4] B. v. Kakani, Di. Gandhi, and S. Jani, “Improved OCR based automatic vehicle number plate recognition using features trained neural network,” 8th International Conference on Computing, Communications and Networking Technologies, ICCCNT 2017, Dec. 2017, doi: 10.1109/ICCCNT.2017.8203916.

[5] W. W. Keong and V. Iranmanesh, "Malaysian automatic number plate recognition system using Pearson correlation," ISCAIE 2016 - 2016 IEEE Symposium on Computer Applications and Industrial Electronics, pp. 40–45, Sep. 2016, doi: 10.1109/ISCAIE.2016.7575034.

[6] M. I. Khalil and S. M. S. Imran, "Automated Toll Collection and Charging System using Radio Frequency Identification in Bangladesh".

[7] C. K. Jalba, A. Muminovic, R. Jung, and S. Epple, "Student’s course' introduction to industrial image recognition,"’ Proceedings of the 2016 International Conference and Exposition on Electrical and Power Engineering, EPE 2016, pp. 069–074, Dec. 2016, doi: 10.1109/ICEPE.2016.7781305.

[8] H. Yamasaki and Y. Hiranaka, “Multi-dimensional intelligent sensing systems using sensor array,” Transducers ’91, pp. 316–321, 1991, doi: 10.1109/SENSOR.1991.148873.

[9] Y. Song and H. Yan, “Image Segmentation Techniques Overview,” AMS 2017 - Asia Modelling Symposium 2017 and 11th International Conference on Mathematical Modelling and Computer Simulation, pp. 103–107, Aug. 2018, doi: 10.1109/AMS.2017.24.

[10] Y. Sani, A. Mohamedou, K. Ali, A. Farjamfar, M. Azman, and S. Shamsuddin, “An overview of neural networks uses in anomaly intrusion

detection systems,” SCOReD2009 - Proceedings of 2009 IEEE Student Conference on Research and Development, pp. 89–92, 2009, doi: 10.1109/SCORED.2009.5443289.

[11] C. Chen, J. Huang, C. Pan, and X. Yuan, “Military image scene recognition based on CNN and semantic information,” Proceedings - 2018 3rd International Conference on Mechanical, Control and Computer Engineering, ICMCCE 2018, pp. 573–577, Nov. 2018, doi: 10.1109/ICMCCE.2018.00126.

[12] W. S. Ahmed and A. A. A. Karim, “The Impact of Filter Size and Number of Filters on Classification Accuracy in CNN,” Proceedings of the 2020 International Conference on Computer Science and Software Engineering, CSASE 2020, pp. 88–93, Apr. 2020, doi: 10.1109/CSASE48920.2020.9142089.

[13] M. M. Shidore and S. P. Narote, “Number plate Recognition for Indian Vehicles,”2011. [Online]. Available: https://www.researchgate.net/publication/265991428

[14] He, K., Zhang, X., Ren, S. and Sun, J. 2016. Deep Residual Learning for Image Recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Dec. 2016), 770–778.

[15] Huang, G., Liu, Z., Maaten, L. van der and Weinberger, K.Q. 2017. Densely Connected Convolutional Networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Nov. 2017), 2261–2269.

[16] Szegedy, C., Ioffe, S., Vanhoucke, V. and Alemi, A. 2017. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. The Thirty-First AAAI Conference on Artificial Intelligence (2017), 4278–4284

[17] Hendry and R. C. Chen, “Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning,” Image and Vision Computing, vol. 87, pp. 47–56, Jul. 2019, doi: 10.1016/J.IMAVIS.2019.04.007.

[18] M. Rajalakshmi, P. Saranya, and P. Shanmugavadivu, “Pattern Recognition-Recognition of Handwritten Document Using Convolutional Neural Networks,” IEEE International Conference on Intelligent Techniques in Control, Optimization and Signal Processing, INCOS 2019, Apr. 2019, doi: 10.1109/INCOS45849.2019.8951342.

[19] M. J. Ahmed, M. Sarfaz, A. Zidouri, and K. G. AI-Khatib, “License plate recognition system,” Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems, vol. 2, pp. 898–901, 2003, doi: 10.1109/ICECS.2003.1301932.

[20] S. Ozbay and E. Erçelebi, “Automatic Vehicle Identification by Plate Recognition,” undefined, 2007.

[21] A. Aggarwal, A. Rani, and M. Kumar, “A robust method to authenticate car license plates using segmentation and ROI based approach,” Smart and Sustainable Built Environment, vol. 9, no. 4, pp. 737–747, Dec. 2019, doi: 10.1108/SASBE-07-2019-0083

[22] R. Hobbs, “Integrating ethically align design into agile and CRISP-DM,” Conference Proceedings - IEEE SOUTHEASTCON, vol. 2021-March, Mar. 2021, doi: 10.1109/SOUTHEASTCON45413.2021.9401899.

[23] M. Satsangi, M. Yadav, and P. S. Sudhish, “License Plate Recognition: A Comparative Study on Thresholding, OCR and Machine Learning Approaches,” International Conference on Bioinformatics and Systems Biology, BSB 2018, pp. 58–63, Oct. 2018, doi: 10.1109/BSB.2018.8770662.

[24] A. ben Salah, J. P. Moreux, N. Ragot, and T. Paquet, “OCR performance prediction using cross-OCR alignment,” Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, vol. 2015-November, pp. 556–560, Nov. 2015, doi: 10.1109/ICDAR.2015.7333823.

[25] M. Oyamada, “Extracting Feature Engineering Knowledge from Data Science Notebooks,” Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019, pp. 6172–6173, Dec. 2019, doi: 10.1109/BIGDATA47090.2019.9006522.

[26] A. D. Nguyen, S. Choi, W. Kim, S. Ahn, J. Kim, and S. Lee, “Distribution Padding in Convolutional Neural Networks,” *Proceedings - International Conference on Image Processing, ICIP*, vol. 2019-September, pp. 4275–4279, Sep. 2019, doi: 10.1109/ICIP.2019.8803537.

[27] Q. Al-Tashi, S. J. Abdul Kadir, H. M. Rais, S. Mirjalili, and H. Alhussian, “Binary Optimization Using Hybrid Grey Wolf Optimization for Feature Selection,” IEEE Access, vol. 7, pp. 39496–39508, 2019, doi: 10.1109/ACCESS.2019.2906757.

[28] M. G. Ragab, S. J. Abdul Kadir, Norshakirah Aziz, H. Alhussian, “A Novel One-Dimensional CNN with Exponential Adaptive Gradients for Air Pollution Index Prediction,” Sustainability 2020, Vol. 12, Page 10090, vol. 12, no. 23, p. 10090, Dec. 2020, doi: 10.3390/SU122310090.

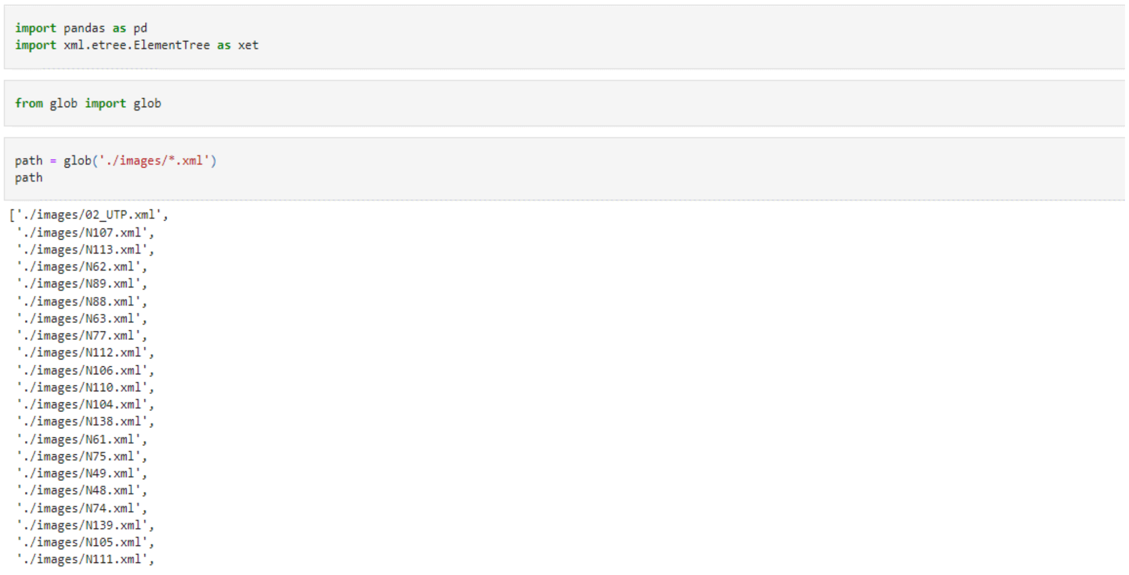
[29] A. Demir and F. Yilmaz, “Inception-ResNet-v2 with Leakyrelu and Averagepooling for More Reliable and Accurate Classification of Chest X-ray Images,” TIPTEKNO 2020 - Tip Teknolojileri Kongresi - 2020 Medical Technologies Congress, TIPTEKNO 2020, Nov. 2020, doi: 10.1109/TIPTEKNO50054.2020.9299232.

[30] D. P. Kingma and J. L. Ba, “Adam: A Method for Stochastic Optimization,” 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, Dec. 2014, Accessed: Nov. 28, 2021. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[31] S. Shastry, G. Gunasheela, T. Dutt, D. S. Vinay, and S. R. Rupanagudi, "'i' - A novel algorithm for optical character recognition (OCR)," Proceedings - 2013 IEEE International Multi Conference on Automation, Computing, Control, Communication and Compressed Sensing, iMac4s 2013, pp. 389–393, 2013, doi: 10.1109/IMAC4S.2013.6526442.

APPENDIX

1. XML file to CSV

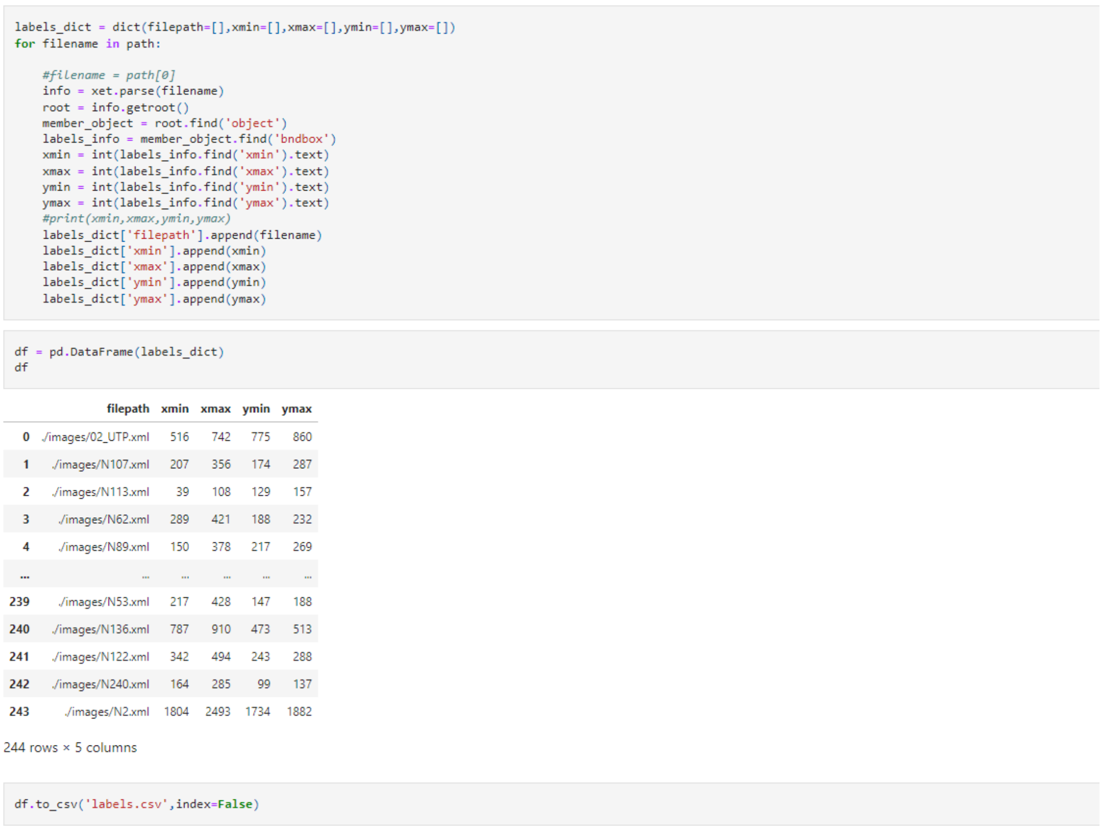


```
import pandas as pd
import xml.etree.ElementTree as xet
```

```
from glob import glob
```

```
path = glob('./images/*.xml')
path
```

```
['./images/02_UTP.xml',
 './images/N107.xml',
 './images/N113.xml',
 './images/N62.xml',
 './images/N89.xml',
 './images/N88.xml',
 './images/N63.xml',
 './images/N77.xml',
 './images/N112.xml',
 './images/N106.xml',
 './images/N110.xml',
 './images/N104.xml',
 './images/N138.xml',
 './images/N61.xml',
 './images/N75.xml',
 './images/N49.xml',
 './images/N48.xml',
 './images/N74.xml',
 './images/N139.xml',
 './images/N105.xml',
 './images/N111.xml',
```



```
labels_dict = dict(filepath=[],xmin=[],xmax=[],ymin=[],ymax=[])
for filename in path:

    #filename = path[0]
    info = xet.parse(filename)
    root = info.getroot()
    member_object = root.find('object')
    labels_info = member_object.find('bndbox')
    xmin = int(labels_info.find('xmin').text)
    xmax = int(labels_info.find('xmax').text)
    ymin = int(labels_info.find('ymin').text)
    ymax = int(labels_info.find('ymax').text)
    #print(xmin,xmax,ymin,ymax)
    labels_dict['filepath'].append(filename)
    labels_dict['xmin'].append(xmin)
    labels_dict['xmax'].append(xmax)
    labels_dict['ymin'].append(ymin)
    labels_dict['ymax'].append(ymax)
```

```
df = pd.DataFrame(labels_dict)
df
```

```
                 filepath  xmin  xmax  ymin  ymax
  0  ./images/02_UTP.xml    516   742   775   860
  1    ./images/N107.xml    207   356   174   287
  2    ./images/N113.xml     39   108   129   157
  3     ./images/N62.xml    289   421   188   232
  4     ./images/N89.xml    150   378   217   269
...                  ...    ...   ...   ...   ...
239     ./images/N53.xml    217   428   147   188
240    ./images/N136.xml    787   910   473   513
241    ./images/N122.xml    342   494   243   288
242    ./images/N240.xml    164   285    99   137
243      ./images/N2.xml   1804  2493  1734  1882

244 rows × 5 columns
```

```
df.to_csv('labels.csv',index=False)
```

2. Object Detection

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import cv2
```

```
df = pd.read_csv('labels.csv')
df.head()
```

	filepath	xmin	xmax	ymin	ymax
0	./images/02_UTP.xml	516	742	775	860
1	./images/N107.xml	207	356	174	287
2	./images/N113.xml	39	108	129	157
3	./images/N62.xml	289	421	188	232
4	./images/N89.xml	150	378	217	269

```
import xml.etree.ElementTree as xet
```

```
filename = df['filepath'][0]
filename
```

```
'./images/02_UTP.xml'
```

```
def getFilename(filename):
    filename_image = xet.parse(filename).getroot().find('filename').text
    filepath_image = os.path.join('./images',filename_image)
    return filepath_image
```

```
getFilename(filename)
```

```
'./images/02.jpeg'
```

```
image_path = list(df['filepath'].apply(getFilename))
image_path
```

```
['./images/02.jpeg',
 './images/N107.jpeg',
 './images/N113.jpeg',
 './images/N62.jpeg',
 './images/N89.jpeg',
```

Verify image and Output

```
file_path = image_path[0]
file_path
```

```
'./images/02.jpeg'
```

```
img = cv2.imread(file_path)
cv2.namedWindow('Car Detection',cv2.WINDOW_NORMAL)
cv2.imshow('Car Detection',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
# 1093  1396    645     727
cv2.rectangle(img,(1093,645),(1396,727),(0,255,0),3)
cv2.namedWindow('Car Detection',cv2.WINDOW_NORMAL)
cv2.imshow('example',img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Data Preprocessing

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

```
labels = df.iloc[:,1:].values
```

```
data = []
output = []
for ind in range(len(image_path)):
    image = image_path[ind]
    img_arr = cv2.imread(image)
    h,w,d = img_arr.shape
    # prepprocesing
    load_image = load_img(image,target_size=(224,224))
    load_image_arr = img_to_array(load_image)
    norm_load_image_arr = load_image_arr/255.0 # normalization
    # normalization to labels
    xmin,xmax,ymin,ymax = labels[ind]
    nxmin,nxmax = xmin/w,xmax/w
    nymin,nymax = ymin/h,ymax/h
    label_norm = (nxmin,nxmax,nymin,nymax) # normalized output
    # -------------- append
    data.append(norm_load_image_arr)
    output.append(label_norm)
```

```
X = np.array(data,dtype=np.float32)
y = np.array(output,dtype=np.float32)
```

```
X.shape,y.shape
```

```
((244, 224, 224, 3), (244, 4))
```

```
x_train,x_test,y_train,y_test = train_test_split(X,y,train_size=0.8,random_state=0)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

```
((195, 224, 224, 3), (49, 224, 224, 3), (195, 4), (49, 4))
```

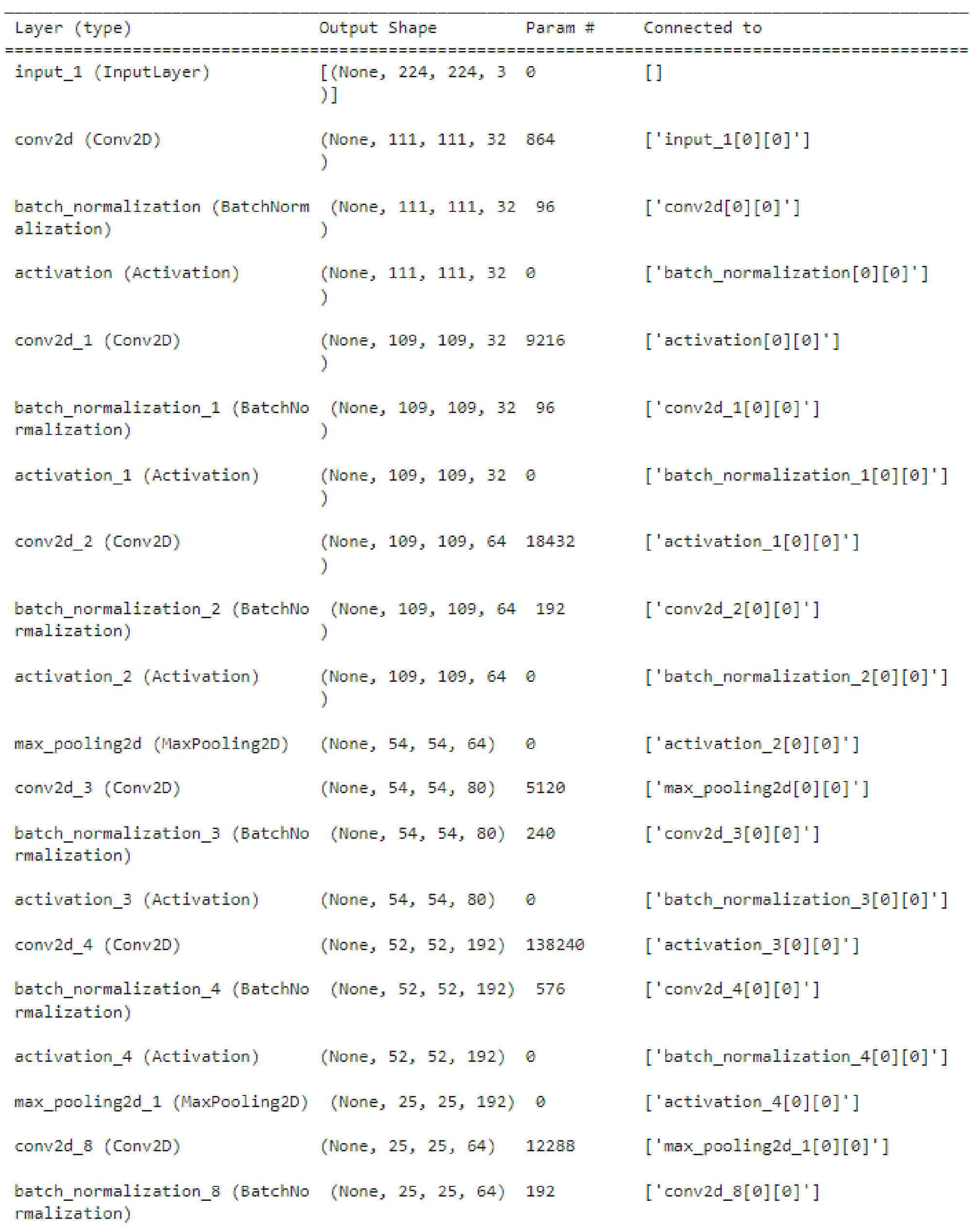
Deep Learning Model

```
from tensorflow.keras.applications import MobileNetV2, InceptionV3, InceptionResNetV2
from tensorflow.keras.layers import Dense, Dropout, Flatten, Input
from tensorflow.keras.models import Model
import tensorflow as tf
```

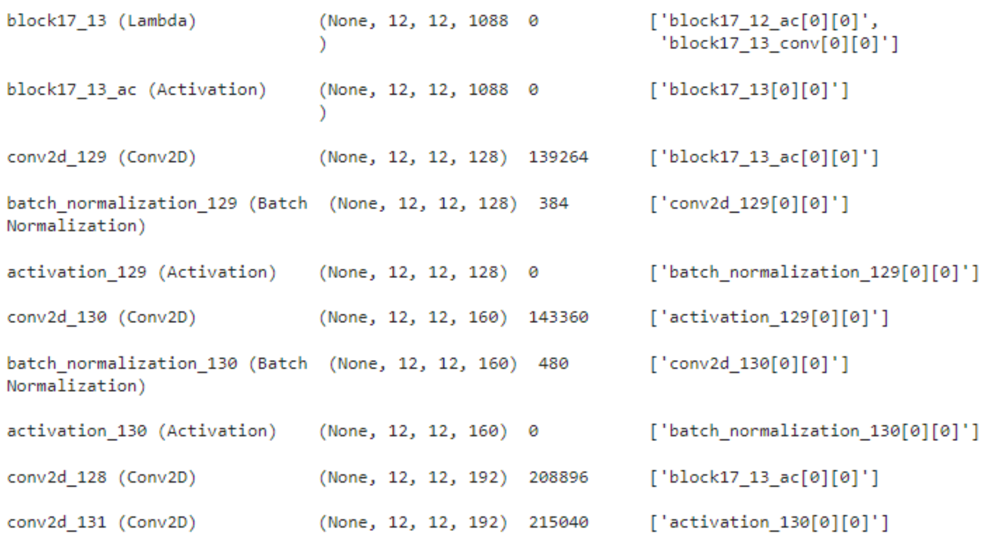
```
inception_resnet = InceptionResNetV2(weights="imagenet",include_top=False,
                                     input_tensor=Input(shape=(224,224,3)))
inception_resnet.trainable=False
# ---------------------
headmodel = inception_resnet.output
headmodel = Flatten()(headmodel)
headmodel = Dense(500,activation="relu")(headmodel)
headmodel = Dense(250,activation="relu")(headmodel)
headmodel = Dense(4,activation='sigmoid')(headmodel)
# ---------- model
model = Model(inputs=inception_resnet.input,outputs=headmodel)
```

```
2021-11-26 16:13:48.365663: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Dee
A
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
```

```
# compile model
model.compile(loss='mse',optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4),metrics = ['accuracy'])
model.summary()
```



```
 Layer (type)                  Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)          [(None, 224, 224, 3  0           []
                               )]

 conv2d (Conv2D)               (None, 111, 111, 32  864         ['input_1[0][0]']
                               )

 batch_normalization (BatchNorm  (None, 111, 111, 32  96        ['conv2d[0][0]']
 alization)                    )

 activation (Activation)       (None, 111, 111, 32  0           ['batch_normalization[0][0]']
                               )

 conv2d_1 (Conv2D)             (None, 109, 109, 32  9216        ['activation[0][0]']
                               )

 batch_normalization_1 (BatchNo  (None, 109, 109, 32  96        ['conv2d_1[0][0]']
 rmalization)                  )

 activation_1 (Activation)     (None, 109, 109, 32  0           ['batch_normalization_1[0][0]']
                               )

 conv2d_2 (Conv2D)             (None, 109, 109, 64  18432       ['activation_1[0][0]']
                               )

 batch_normalization_2 (BatchNo  (None, 109, 109, 64  192       ['conv2d_2[0][0]']
 rmalization)                  )

 activation_2 (Activation)     (None, 109, 109, 64  0           ['batch_normalization_2[0][0]']
                               )

 max_pooling2d (MaxPooling2D)  (None, 54, 54, 64)   0           ['activation_2[0][0]']

 conv2d_3 (Conv2D)             (None, 54, 54, 80)   5120        ['max_pooling2d[0][0]']

 batch_normalization_3 (BatchNo  (None, 54, 54, 80)  240        ['conv2d_3[0][0]']
 rmalization)

 activation_3 (Activation)     (None, 54, 54, 80)   0           ['batch_normalization_3[0][0]']

 conv2d_4 (Conv2D)             (None, 52, 52, 192)  138240      ['activation_3[0][0]']

 batch_normalization_4 (BatchNo  (None, 52, 52, 192)  576       ['conv2d_4[0][0]']
 rmalization)

 activation_4 (Activation)     (None, 52, 52, 192)  0           ['batch_normalization_4[0][0]']

 max_pooling2d_1 (MaxPooling2D)  (None, 25, 25, 192)  0         ['activation_4[0][0]']

 conv2d_8 (Conv2D)             (None, 25, 25, 64)   12288       ['max_pooling2d_1[0][0]']

 batch_normalization_8 (BatchNo  (None, 25, 25, 64)  192        ['conv2d_8[0][0]']
 rmalization)
```



```
block17_13 (Lambda)            (None, 12, 12, 1088  0           ['block17_12_ac[0][0]',
                               )                                 'block17_13_conv[0][0]']

block17_13_ac (Activation)     (None, 12, 12, 1088  0           ['block17_13[0][0]']
                               )

conv2d_129 (Conv2D)            (None, 12, 12, 128)  139264      ['block17_13_ac[0][0]']

batch_normalization_129 (Batch  (None, 12, 12, 128)  384        ['conv2d_129[0][0]']
Normalization)

activation_129 (Activation)    (None, 12, 12, 128)  0           ['batch_normalization_129[0][0]']

conv2d_130 (Conv2D)            (None, 12, 12, 160)  143360      ['activation_129[0][0]']

batch_normalization_130 (Batch  (None, 12, 12, 160)  480        ['conv2d_130[0][0]']
Normalization)

activation_130 (Activation)    (None, 12, 12, 160)  0           ['batch_normalization_130[0][0]']

conv2d_128 (Conv2D)            (None, 12, 12, 192)  208896      ['block17_13_ac[0][0]']

conv2d_131 (Conv2D)            (None, 12, 12, 192)  215040      ['activation_130[0][0]']
```

```
 block8_9_ac (Activation)       (None, 5, 5, 2080)  0           ['block8_9[0][0]']

 conv2d_200 (Conv2D)            (None, 5, 5, 192)   399360      ['block8_9_ac[0][0]']

 batch_normalization_200 (Batch  (None, 5, 5, 192)  576         ['conv2d_200[0][0]']
 Normalization)

 activation_200 (Activation)    (None, 5, 5, 192)   0           ['batch_normalization_200[0][0]']

 conv2d_201 (Conv2D)            (None, 5, 5, 224)   129024      ['activation_200[0][0]']

 batch_normalization_201 (Batch  (None, 5, 5, 224)  672         ['conv2d_201[0][0]']
 Normalization)

 activation_201 (Activation)    (None, 5, 5, 224)   0           ['batch_normalization_201[0][0]']

 conv2d_199 (Conv2D)            (None, 5, 5, 192)   399360      ['block8_9_ac[0][0]']

 conv2d_202 (Conv2D)            (None, 5, 5, 256)   172032      ['activation_201[0][0]']

 batch_normalization_199 (Batch  (None, 5, 5, 192)  576         ['conv2d_199[0][0]']
 Normalization)

 batch_normalization_202 (Batch  (None, 5, 5, 256)  768         ['conv2d_202[0][0]']
 Normalization)

 activation_199 (Activation)    (None, 5, 5, 192)   0           ['batch_normalization_199[0][0]']

 activation_202 (Activation)    (None, 5, 5, 256)   0           ['batch_normalization_202[0][0]']

 block8_10_mixed (Concatenate)  (None, 5, 5, 448)   0           ['activation_199[0][0]',
                                                                 'activation_202[0][0]']

 block8_10_conv (Conv2D)        (None, 5, 5, 2080)  933920      ['block8_10_mixed[0][0]']

 block8_10 (Lambda)             (None, 5, 5, 2080)  0           ['block8_9_ac[0][0]',
                                                                 'block8_10_conv[0][0]']

 conv_7b (Conv2D)               (None, 5, 5, 1536)  3194880     ['block8_10[0][0]']

 conv_7b_bn (BatchNormalization  (None, 5, 5, 1536)  4608       ['conv_7b[0][0]']
 )

 conv_7b_ac (Activation)        (None, 5, 5, 1536)  0           ['conv_7b_bn[0][0]']

 flatten (Flatten)              (None, 38400)       0           ['conv_7b_ac[0][0]']

 dense (Dense)                  (None, 500)         19200500    ['flatten[0][0]']

 dense_1 (Dense)                (None, 250)         125250      ['dense[0][0]']

 dense_2 (Dense)                (None, 4)           1004        ['dense_1[0][0]']

==================================================================================================
Total params: 73,663,490
Trainable params: 19,326,754
Non-trainable params: 54,336,736
__________________________________________________________________________________________________
```

Model Trainig

```
from tensorflow.keras.callbacks import TensorBoard
```

```
tfb = TensorBoard('object_detection')
```

```
history = model.fit(x=x_train,y=y_train,batch_size=10,epochs=200,
                    validation_data=(x_test,y_test),callbacks=[tfb])
```

```
Epoch 1/200
20/20 [==============================] - 47s 2s/step - loss: 0.0792 - accuracy: 0.5231 - val_loss: 0.0679 - val_accuracy: 0.6122
Epoch 2/200
20/20 [==============================] - 31s 2s/step - loss: 0.0649 - accuracy: 0.5436 - val_loss: 0.0804 - val_accuracy: 0.6122
Epoch 3/200
20/20 [==============================] - 30s 1s/step - loss: 0.0628 - accuracy: 0.5436 - val_loss: 0.0538 - val_accuracy: 0.6122
Epoch 4/200
20/20 [==============================] - 31s 2s/step - loss: 0.0472 - accuracy: 0.5436 - val_loss: 0.0467 - val_accuracy: 0.6122
Epoch 5/200
20/20 [==============================] - 29s 1s/step - loss: 0.0297 - accuracy: 0.5026 - val_loss: 0.0187 - val_accuracy: 0.5510
Epoch 6/200
20/20 [==============================] - 30s 2s/step - loss: 0.0161 - accuracy: 0.7231 - val_loss: 0.0143 - val_accuracy: 0.6327
Epoch 7/200
20/20 [==============================] - 29s 1s/step - loss: 0.0095 - accuracy: 0.7333 - val_loss: 0.0138 - val_accuracy: 0.7551
Epoch 8/200
20/20 [==============================] - 45s 2s/step - loss: 0.0067 - accuracy: 0.7795 - val_loss: 0.0121 - val_accuracy: 0.6939
Epoch 9/200
20/20 [==============================] - 37s 2s/step - loss: 0.0061 - accuracy: 0.8154 - val_loss: 0.0113 - val_accuracy: 0.8163
Epoch 10/200
20/20 [==============================] - 35s 2s/step - loss: 0.0053 - accuracy: 0.7385 - val_loss: 0.0113 - val_accuracy: 0.8980
Epoch 11/200
20/20 [==============================] - 30s 2s/step - loss: 0.0051 - accuracy: 0.8462 - val_loss: 0.0129 - val_accuracy: 0.7959
Epoch 12/200
20/20 [==============================] - 27s 1s/step - loss: 0.0049 - accuracy: 0.8051 - val_loss: 0.0113 - val_accuracy: 0.8367
Epoch 13/200
20/20 [==============================] - 27s 1s/step - loss: 0.0041 - accuracy: 0.8462 - val_loss: 0.0110 - val_accuracy: 0.8367
Epoch 14/200
20/20 [==============================] - 27s 1s/step - loss: 0.0029 - accuracy: 0.8821 - val_loss: 0.0104 - val_accuracy: 0.7755
Epoch 15/200
20/20 [==============================] - 27s 1s/step - loss: 0.0026 - accuracy: 0.8923 - val_loss: 0.0115 - val_accuracy: 0.7143
Epoch 16/200
20/20 [==============================] - 27s 1s/step - loss: 0.0025 - accuracy: 0.9231 - val_loss: 0.0104 - val_accuracy: 0.8163
Epoch 17/200
20/20 [==============================] - 27s 1s/step - loss: 0.0029 - accuracy: 0.8821 - val_loss: 0.0103 - val_accuracy: 0.8163
Epoch 18/200
20/20 [==============================] - 27s 1s/step - loss: 0.0029 - accuracy: 0.9179 - val_loss: 0.0104 - val_accuracy: 0.6939
Epoch 19/200
20/20 [==============================] - 27s 1s/step - loss: 0.0029 - accuracy: 0.8923 - val_loss: 0.0106 - val_accuracy: 0.8776
Epoch 20/200
20/20 [==============================] - 27s 1s/step - loss: 0.0029 - accuracy: 0.9077 - val_loss: 0.0128 - val_accuracy: 0.8367
Epoch 21/200
20/20 [==============================] - 27s 1s/step - loss: 0.0036 - accuracy: 0.8718 - val_loss: 0.0109 - val_accuracy: 0.8163
Epoch 22/200
20/20 [==============================] - 30s 2s/step - loss: 0.0033 - accuracy: 0.8667 - val_loss: 0.0127 - val_accuracy: 0.6939
Epoch 23/200
20/20 [==============================] - 32s 2s/step - loss: 0.0039 - accuracy: 0.8667 - val_loss: 0.0118 - val_accuracy: 0.8367
 Epoch 138/200
 20/20 [==============================] - 30s 2s/step - loss: 0.0013 - accuracy: 0.9282 - val_loss: 0.0096 - val_accuracy: 0.7755
 Epoch 139/200
 20/20 [==============================] - 30s 2s/step - loss: 0.0012 - accuracy: 0.9333 - val_loss: 0.0096 - val_accuracy: 0.8571
 Epoch 140/200
 20/20 [==============================] - 30s 2s/step - loss: 0.0011 - accuracy: 0.9077 - val_loss: 0.0091 - val_accuracy: 0.8163
 Epoch 141/200
 20/20 [==============================] - 30s 2s/step - loss: 9.7022e-04 - accuracy: 0.9333 - val_loss: 0.0093 - val_accuracy: 0.8163
 Epoch 142/200
 20/20 [==============================] - 30s 2s/step - loss: 8.3787e-04 - accuracy: 0.9385 - val_loss: 0.0095 - val_accuracy: 0.8367
 Epoch 143/200
 20/20 [==============================] - 30s 2s/step - loss: 9.3914e-04 - accuracy: 0.9487 - val_loss: 0.0090 - val_accuracy: 0.8367
 Epoch 144/200
 20/20 [==============================] - 30s 2s/step - loss: 9.3666e-04 - accuracy: 0.9333 - val_loss: 0.0094 - val_accuracy: 0.8367
 Epoch 145/200
 20/20 [==============================] - 30s 2s/step - loss: 0.0011 - accuracy: 0.9385 - val_loss: 0.0093 - val_accuracy: 0.8571
 Epoch 146/200
 20/20 [==============================] - 30s 2s/step - loss: 0.0011 - accuracy: 0.9795 - val_loss: 0.0094 - val_accuracy: 0.8367
20/20 [==============================] - 32s 2s/step - loss: 7.1198e-04 - accuracy: 0.9641 - val_loss: 0.0087 - val_accuracy: 0.8367
Epoch 194/200
20/20 [==============================] - 33s 2s/step - loss: 7.3352e-04 - accuracy: 0.9590 - val_loss: 0.0092 - val_accuracy: 0.7959
Epoch 195/200
20/20 [==============================] - 32s 2s/step - loss: 7.8265e-04 - accuracy: 0.9692 - val_loss: 0.0089 - val_accuracy: 0.7959
Epoch 196/200
20/20 [==============================] - 32s 2s/step - loss: 9.1928e-04 - accuracy: 0.9590 - val_loss: 0.0088 - val_accuracy: 0.8571
Epoch 197/200
20/20 [==============================] - 34s 2s/step - loss: 8.5928e-04 - accuracy: 0.9692 - val_loss: 0.0088 - val_accuracy: 0.8367
Epoch 198/200
20/20 [==============================] - 29s 1s/step - loss: 8.1656e-04 - accuracy: 0.9538 - val_loss: 0.0091 - val_accuracy: 0.8163
Epoch 199/200
20/20 [==============================] - 28s 1s/step - loss: 8.1391e-04 - accuracy: 0.9641 - val_loss: 0.0091 - val_accuracy: 0.8367
Epoch 200/200
20/20 [==============================] - 27s 1s/step - loss: 8.4495e-04 - accuracy: 0.9692 - val_loss: 0.0093 - val_accuracy: 0.7959
```

```
history.history['accuracy']
```

```
[0.5230769515037537,
 0.5435897707939148,
 0.5435897707939148,
 0.5435897707939148,
 0.5025641322135925,
 0.7230769395828247,
 0.7333333492279053,
 0.7794871926307678,
 0.8153846263885498,
```

```
sum(history.history['accuracy'])/len(history.history['accuracy'])
```

```
0.9109230822324753
```

```
model.save('./models/object_detection.h5')
```

3. Make Prediction

```
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

```
# Load model
model = tf.keras.models.load_model('./models/object_detection.h5')
print('model loaded sucessfully')
```

```
2021-11-23 01:36:34.646460: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized with oneAPI Deep
A
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
model loaded sucessfully
```

```
path = './images/0001.jpeg'
image = load_img(path) # PIL object
image = np.array(image,dtype=np.uint8) # 8 bit array (0,255)
image1 = load_img(path,target_size=(224,224))
image_arr_224 = img_to_array(image1)/255.0  # convert into array and get the normalized output
```

```
# size of the orginal image
h,w,d = image.shape
print('Height of the image =',h)
print('Width of the image =',w)
```

```
Height of the image = 1280
Width of the image = 958
```

```
plt.figure(figsize=(10,8))
plt.imshow(image)
plt.show()
```

```
image_arr_224.shape
```

```
(224, 224, 3)
```

```
test_arr = image_arr_224.reshape(1,224,224,3)
test_arr.shape
```

```
(1, 224, 224, 3)
```

```
# make predictions
coords = model.predict(test_arr)
coords
```

```
array([[0.4280041 , 0.65298146, 0.6463429 , 0.70935315]], dtype=float32)
```

```
# denormalize the values
denorm = np.array([w,w,h,h])
coords = coords * denorm
coords
```

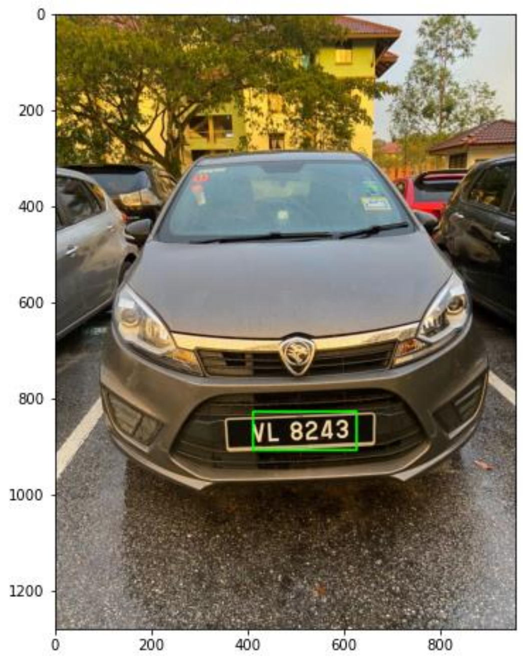
```
array([[410.0279144 , 625.55623877, 827.31887817, 907.97203064]])
```

```
coords = coords.astype(np.int32)
coords
```

```
array([[410, 625, 827, 907]], dtype=int32)
```

```
# draw bounding on top the image
xmin, xmax,ymin,ymax = coords[0]
pt1 =(xmin,ymin)
pt2 =(xmax,ymax)
print(pt1, pt2)
cv2.rectangle(image,pt1,pt2,(0,255,0),3)

plt.figure(figsize=(10,8))
plt.imshow(image)
plt.show()
```

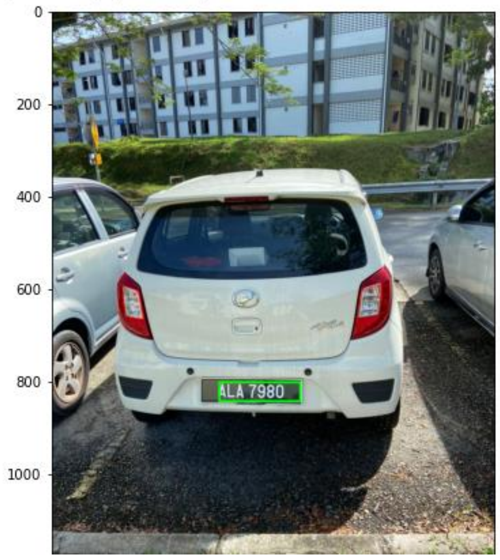
```
(410, 827) (625, 907)
```



```python
# create pipeline
path = 'images/05.jpeg'
def object_detection(path):
    # read image
    image = load_img(path) # PIL object
    image = np.array(image,dtype=np.uint8) # 8 bit array (0,255)
    image1 = load_img(path,target_size=(224,224))
    # data preprocessing
    image_arr_224 = img_to_array(image1)/255.0  # convert into array and get the normalized output
    h,w,d = image.shape
    test_arr = image_arr_224.reshape(1,224,224,3)
    # make predictions
    coords = model.predict(test_arr)
    # denormalize the values
    denorm = np.array([w,w,h,h])
    coords = coords * denorm
    coords = coords.astype(np.int32)
    # draw bounding on top the image
    xmin, xmax,ymin,ymax = coords[0]
    pt1 =(xmin,ymin)
    pt2 =(xmax,ymax)
    print(pt1, pt2)
    cv2.rectangle(image,pt1,pt2,(0,255,0),3)
    return image, coords
```

```python
path = './images/06.jpeg'
image, cods = object_detection(path)

plt.figure(figsize=(10,8))
plt.imshow(image)
plt.show()
```

```
(361, 801) (536, 842)
```

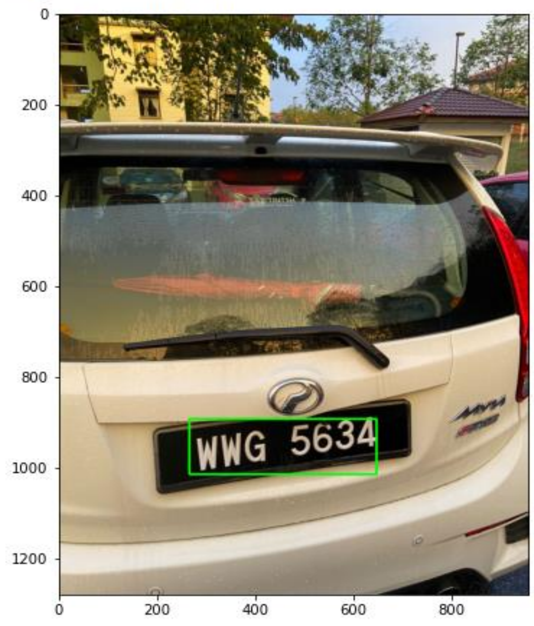


Optical Character Recognition - OCR

```
import pytesseract as pt
```

```
path = 'images/0005.jpeg'
image, cods = object_detection(path)

plt.figure(figsize=(10,8))
plt.imshow(image)
plt.show()
```

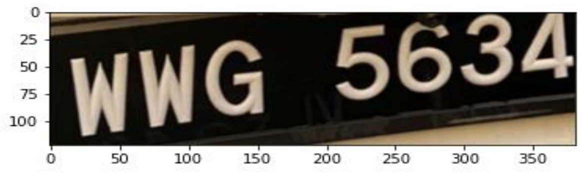
```
(265, 892) (646, 1014)
```



```
img = np.array(load_img(path))
xmin ,xmax,ymin,ymax = cods[0]
roi = img[ymin:ymax,xmin:xmax]
```

```
plt.imshow(roi)
plt.show()
```

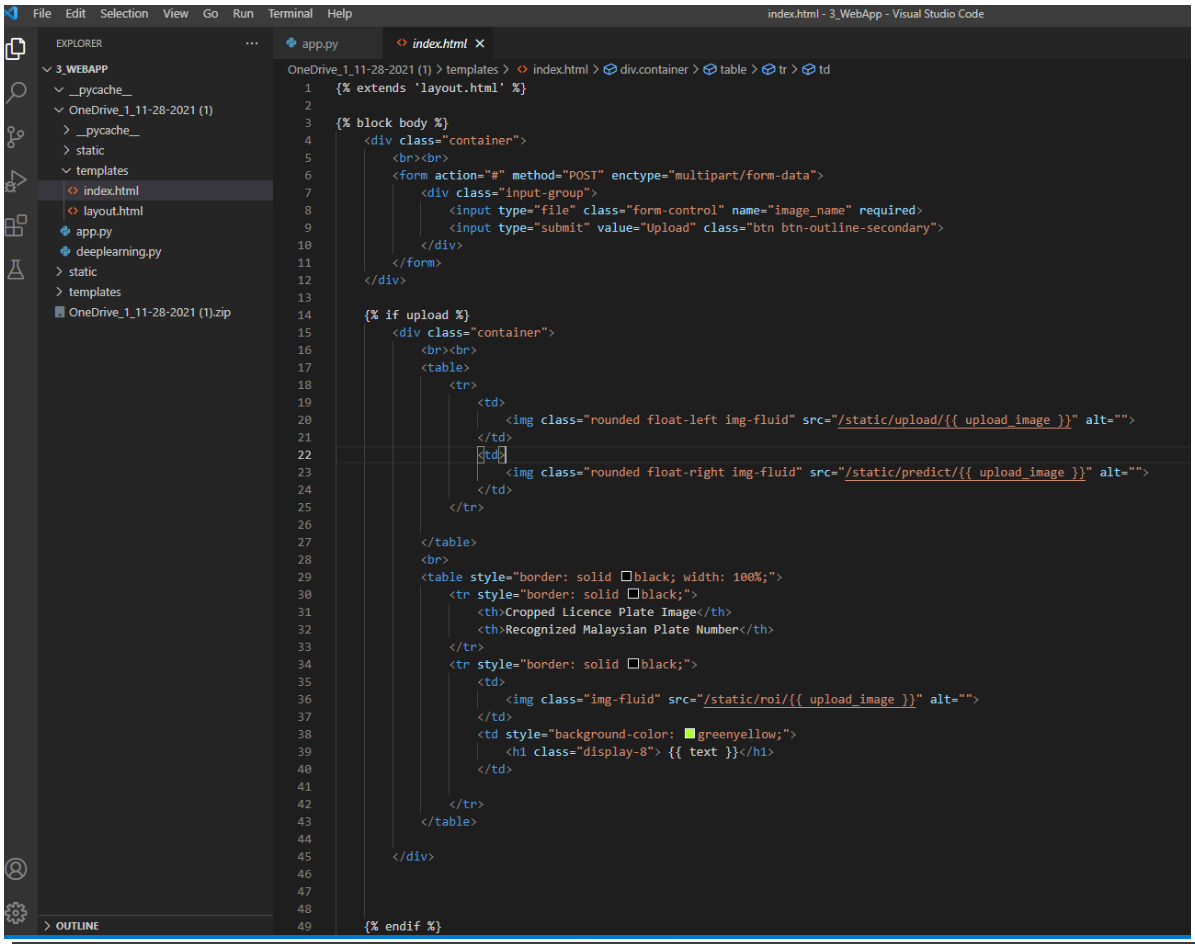
```
plt.imshow(roi)
plt.show()
```



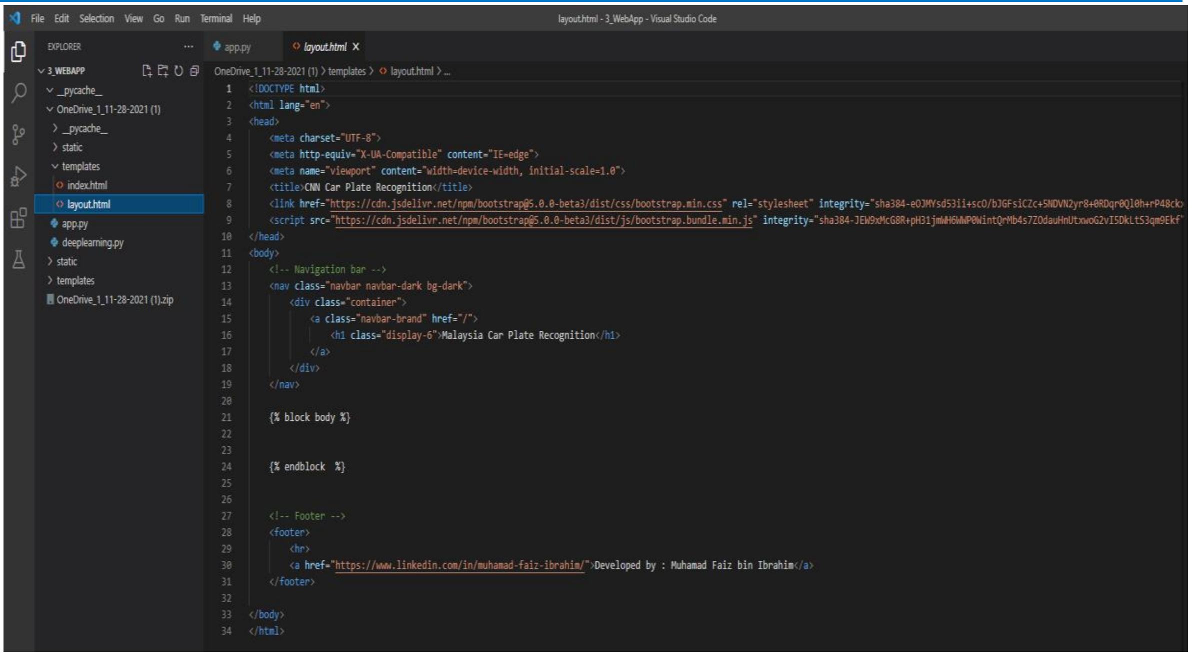
```
# extract text from image
text = pt.image_to_string(roi)
print(text)
```

```
WWG 5634
```

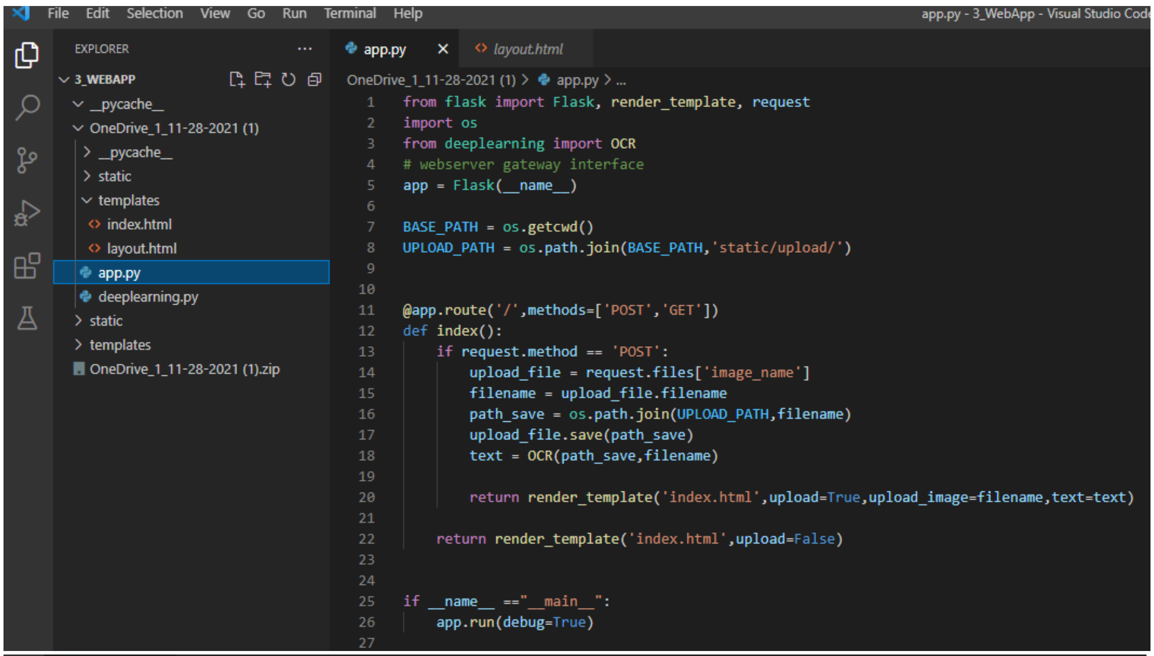
4. Web Deployment using Flask



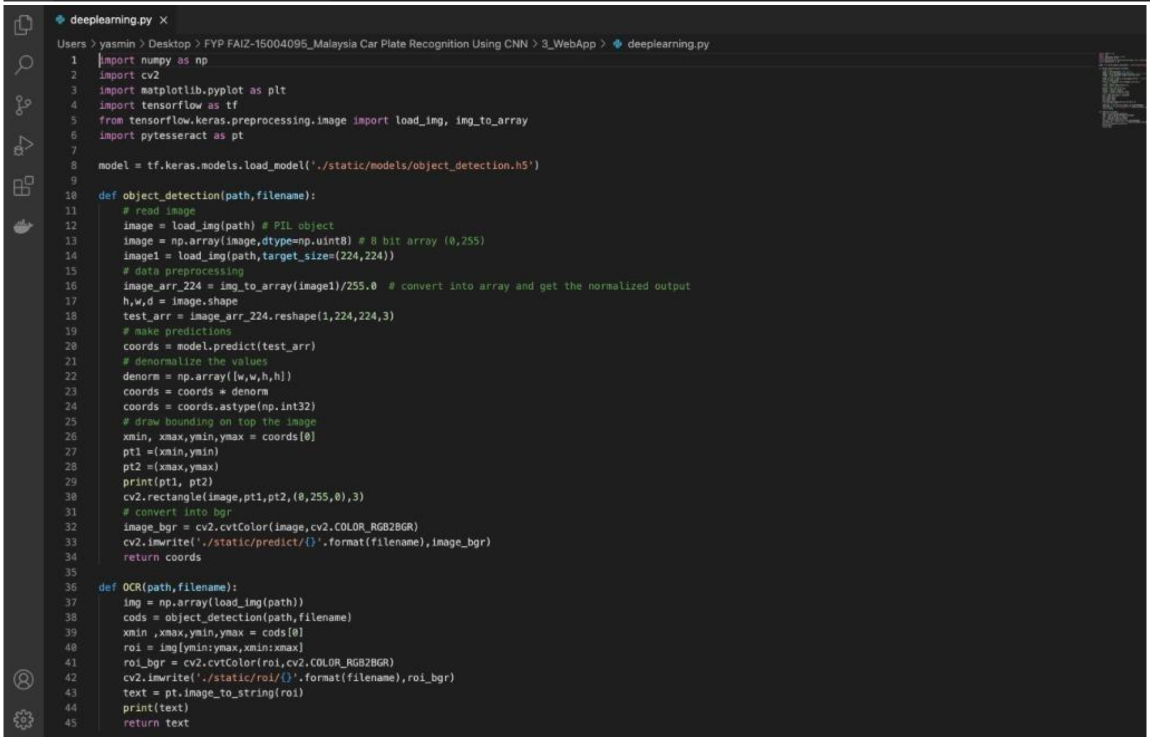
```
{% extends 'layout.html' %}

{% block body %}
    <div class="container">
        <br><br>
        <form action="#" method="POST" enctype="multipart/form-data">
            <div class="input-group">
                <input type="file" class="form-control" name="image_name" required>
                <input type="submit" value="Upload" class="btn btn-outline-secondary">
            </div>
        </form>
    </div>

    {% if upload %}
        <div class="container">
            <br><br>
            <table>
                <tr>
                    <td>
                        <img class="rounded float-left img-fluid" src="/static/upload/{{ upload_image }}" alt="">
                    </td>
                    <td>
                        <img class="rounded float-right img-fluid" src="/static/predict/{{ upload_image }}" alt="">
                    </td>
                </tr>

            </table>
            <br>
            <table style="border: solid black; width: 100%;">
                <tr style="border: solid black;">
                    <th>Cropped Licence Plate Image</th>
                    <th>Recognized Malaysian Plate Number</th>
                </tr>
                <tr style="border: solid black;">
                    <td>
                        <img class="img-fluid" src="/static/roi/{{ upload_image }}" alt="">
                    </td>
                    <td style="background-color: greenyellow;">
                        <h1 class="display-8"> {{ text }}</h1>
                    </td>

                </tr>
            </table>

        </div>



    {% endif %}
```



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CNN Car Plate Recognition</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP48ck
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUtxwoG2vI5DkLtS3qm9Ekf"
</head>
<body>
    <!-- Navigation bar -->
    <nav class="navbar navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="/">
                <h1 class="display-6">Malaysia Car Plate Recognition</h1>
            </a>
        </div>
    </nav>

    {% block body %}


    {% endblock  %}


    <!-- Footer -->
    <footer>
        <hr>
        <a href="https://www.linkedin.com/in/muhamad-faiz-ibrahim/">Developed by : Muhamad Faiz bin Ibrahim</a>
    </footer>

</body>
</html>
```


File Edit Selection View Go Run Terminal Help
app.py - 3_WebApp - Visual Studio Code
EXPLORER
3_WEBAPP
__pycache__
OneDrive_1_11-28-2021 (1)
__pycache__
static
templates
index.html
layout.html
app.py
deeplearning.py
static
templates
OneDrive_1_11-28-2021 (1).zip
app.py
layout.html
OneDrive_1_11-28-2021 (1) > app.py > ...

```python
from flask import Flask, render_template, request
import os
from deeplearning import OCR
# webserver gateway interface
app = Flask(__name__)

BASE_PATH = os.getcwd()
UPLOAD_PATH = os.path.join(BASE_PATH,'static/upload/')


@app.route('/',methods=['POST','GET'])
def index():
    if request.method == 'POST':
        upload_file = request.files['image_name']
        filename = upload_file.filename
        path_save = os.path.join(UPLOAD_PATH,filename)
        upload_file.save(path_save)
        text = OCR(path_save,filename)

        return render_template('index.html',upload=True,upload_image=filename,text=text)

    return render_template('index.html',upload=False)


if __name__ =="__main__":
    app.run(debug=True)
```


deeplearning.py
Users > yasmin > Desktop > FYP FAIZ-15004095_Malaysia Car Plate Recognition Using CNN > 3_WebApp > deeplearning.py

```python
import numpy as np
import cv2
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import pytesseract as pt

model = tf.keras.models.load_model('./static/models/object_detection.h5')

def object_detection(path,filename):
    # read image
    image = load_img(path) # PIL object
    image = np.array(image,dtype=np.uint8) # 8 bit array (0,255)
    image1 = load_img(path,target_size=(224,224))
    # data preprocessing
    image_arr_224 = img_to_array(image1)/255.0  # convert into array and get the normalized output
    h,w,d = image.shape
    test_arr = image_arr_224.reshape(1,224,224,3)
    # make predictions
    coords = model.predict(test_arr)
    # denormalize the values
    denorm = np.array([w,w,h,h])
    coords = coords * denorm
    coords = coords.astype(np.int32)
    # draw bounding on top the image
    xmin, xmax,ymin,ymax = coords[0]
    pt1 =(xmin,ymin)
    pt2 =(xmax,ymax)
    print(pt1, pt2)
    cv2.rectangle(image,pt1,pt2,(0,255,0),3)
    # convert into bgr
    image_bgr = cv2.cvtColor(image,cv2.COLOR_RGB2BGR)
    cv2.imwrite('./static/predict/{}'.format(filename),image_bgr)
    return coords

def OCR(path,filename):
    img = np.array(load_img(path))
    cods = object_detection(path,filename)
    xmin ,xmax,ymin,ymax = cods[0]
    roi = img[ymin:ymax,xmin:xmax]
    roi_bgr = cv2.cvtColor(roi,cv2.COLOR_RGB2BGR)
    cv2.imwrite('./static/roi/{}'.format(filename),roi_bgr)
    text = pt.image_to_string(roi)
    print(text)
    return text
```