UNIVERSITI TEKNIKAL MALAYSIA MELAKA

FACULTY OF ELECTRONIC AND COMPUTER ENGINEERING

BENU 3863 INTEGRATED DESIGN PROJECT

ASSIGNMENT: TECHNICAL REPORT

TITLE: DEEP NEURAL NETWORK BASED LICENSE PLATES DETECTION-RECOGNITION (LPDR) SYSTEM

SUBMITTED BY:

NAME	MATRIC NO
IBRAHIM SOLIMAN	B021510269
WENG SHIH YING	B021510102
WONG YUIN SHEN	B021510040
CHEONG JIA MUN	B021510051
YEW POH LENG	B021510106

SUBMITTED FOR:

PROFESOR DR. ZULKALNAIN BIN MOHD YUSSOF

DATE OF SUBMISSION: 18 MAY 2018

A TECHNICAL REPORT FOR DEEP NEURAL NETWORK BASED LICENSE PLATES DETECTION-RECOGNITION (LPDR) SYSTEM

IBRAHIM SOLIMAN	**B021510269**
WENG SHIH YING	**B021510102**
WONG YUIN SHEN	**B021510040**
CHEONG JIA MUN	**B021510051**
YEW POH LENG	**B021510106**

Faculty of Electronic and Computer Engineering

Universiti Teknikal Malaysia Melaka

May 2018

A TECHNICAL REPORT FOR DEEP NEURAL NETWORK BASED LICENSE PLATES DETECTION-RECOGNITION (LPDR) SYSTEM

IBRAHIM SOLIMAN	**B021510269**
WENG SHIH YING	**B021510102**
WONG YUIN SHEN	**B021510040**
CHEONG JIA MUN	**B021510051**
YEW POH LENG	**B021510106**

This Report Is Submitted in Partial Fulfillment of Requirements for Integrated Design Project (BENU 3863)

Faculty of Electronic and Computer Engineering

Universiti Teknikal Malaysia Melaka

May 2018

ACKNOWLEDGEMENTS

We owe a huge debt of gratitude to our supervisor, Profesor Dr.Zulkalnain Bin Mohd Yussof who had lead and guide us throughout of our project. We referred a lot of things towards him about what we had been confused upon this project. Without his patience and counsel, our technical report would have been a frustrating and difficult to be done. In this project, every member has contributed a lot of effort and time in order to complete our project report. With the patience, hard work and determination of the group members, we could finish the project successfully. Cooperation among us is the key to achieve satisfaction of our works. Lastly, we also want to thank our parents and friends for their constant encouragement

ABSTRACT

License Plate Detection-Recognition (LPDR) is a type of technology, mainly software that enables computer systems to read automatically the registration number (license number) of vehicles from digital pictures. Reading automatically the license plate means transforming the pixels of the digital image into the American Standard Code for Information Interchange (ASCII) text of the number plate. LPDR was always an issue in the field of intelligent transportation systems and computer vision solutions. LPDR engine will be designed to read all vehicle license plate types in Malaysia in different traffic speed. Toll collection and congestion charging systems, traffic monitoring and security, speed and journey time measurement, bus lane and traffic light enforcement, parking or access control and many other systems can benefit from the fast, exact, automatic identification and recognition capabilities of LPDR engine. With the advent and enhancements of artificial intelligence and deep learning fields with cheaper, faster processing hardware, we need to take a fresh look at the problem. Advances in the design of Convolutional Neural Networks “CNNs” have resulted in significant increases in performance accuracies for many tasks. In our project, we will harness the power of CNNs in an end-to-end system capable of detecting and recognizing license plates with low error rates to design a smart vehicle payment system which can be used in Free Flow Tolls.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS AND SYMBOLS

YOLO	You Only Look Once
mAP	Mean Average Precision
IOU	Intersection Over Union
CRNN	Convolutional Recurrent Neural Network
RNN	Recurrent Neural Networks
CTC	Connectionist Temporal Classification
OCR	Optical Character Recognition
SSD	Single Shot Multi-Box Detector
SWT	Stroke Width Transform
MSER	Maximally Stable Extremal Regions
EAST	An Efficient And Accurate Scene TexT detector
FCN	Fully Convolutional Networks
RBOX	Rotated Box
QUAD	Quadrangle
CNN	Convolutional Network
BLSTM	Bidirectional Long Short-Term Temporal Memory
OPL	Open Programming Language
UI	User Interface
SDK	Software Development Kids
RDBMS	Relational Database Management System
SQL	Structured Query Language
PHP	Hypertext Preprocessor

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.0 Problem Statement

According to John Lim published at 17 Aug 2015, one of the reason causing traffic congestion is unprepared when vehicle payment [1]. From this statement, we can know that most of the office buildings and shopping mall had built underground parking and multilevel parking to overcome the number of cars which is increasing rapidly. However, drivers are still facing difficulties to find an available parking slot to park their vehicle. The process of looking for a parking lot is time consuming, confusing and wasting fuel as well. Currently, PCS vision have provided some Malaysia shopping malls Car Park Management Display. This display system shows clearly the number and location of parking lots available in a particular parking zone with sensor [2]. It also causes congestion as everyone is rushing to the same parking spot during peak hours. On the other hand, vvehicle payment on car parking, bridge or road a charge (toll) is also one of the factors causing traffic congestion because drivers need to stop by the gate to pay [3]. Malaysia vehicle payment at toll does have launch by Touch 'n Go Sdn Bhd (TNGSB) since 1997 to save time on paying payment at toll, but there are also many problems occurs. [4] Some even reach the toll gate or parking gate and only realize that they did not reload their card “Touch and go” or pay their parking tickets. This is wasting time and causing congestion. The side effect of these problems is serious and need a better solution to handle it. In this project, we propose a License Plate Detection-Recognition (LPDR) system (instead of buying from outside) that able to computer systems to read automatically the registration number (license number) of vehicles from digital pictures in real time. By having this type of system, it could help to reduce the cost instead of buying the latest technologies from Europe and other country. This system only needs camera and engine based on GPU server to detect, recognize vehicle. It even has higher accuracy and save cost

than using sensor or car loop sensor to detect car. It will give convenience to all the Malaysian to avoid congestion and accident for vehicle payment.

1.1 Objective

The objective of this project is as follows:

i. To develop an engine based on GPU server which can automatically detect and recognize vehicle license plates using Deep learning with a high accuracy that will lead other to use our engine for a variety of application such as, Smart Parking System which is able to identify parking space and assign a parking lot for entering cars, and "Just GO" Vehicle Payment System which can save time on vehicle payment with license plate detection and recognition and help in tracking and security solutions.

1.2 Scope

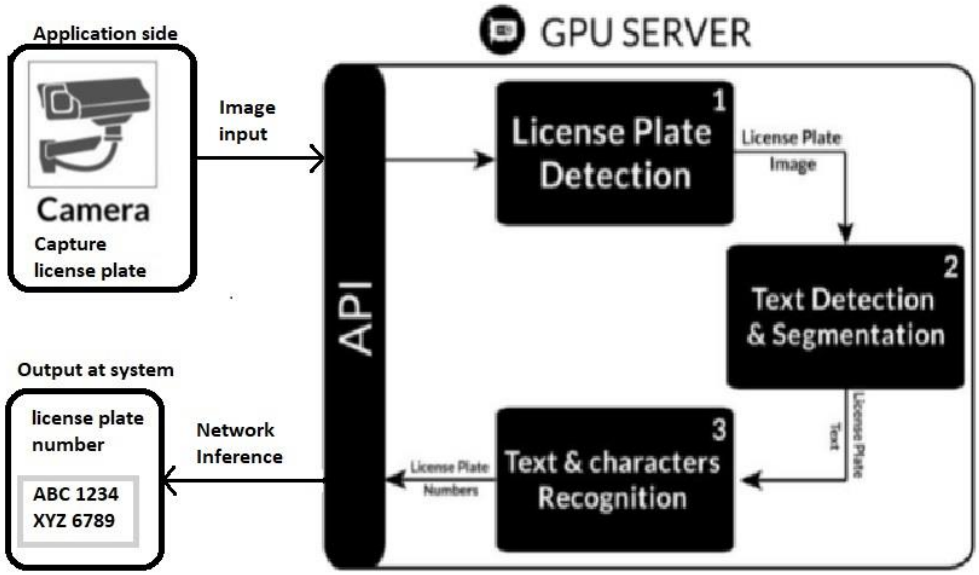


Figure 1.2.1: The Complete Structure of License Plate Detection-Recognition (LPDR) System.

The scope of the project consists of software parts and hardware. On the hardware, the equipment needed are PC with graphics processing unit (GPU) and 2 units of outdoor security camera CCTV. The main scope of the project is to design and implement LPDR engine in GPU server side and computer application with the use of deep learning advantages to reach to minimum error rate. Software parts divided to 3 parts but it is made by using Ubuntu software. First, License Plate

Detection with Text Segmentation under Tenorflow framework using python. The car is capture once the system detect car. Next, the text detection and segmentation on the license plate image from the first part. The third part is the text and character recognition under Torch framework using LUA programming language to send back to the system network the license plate number. The system will then have the car image, license plate and license plate number. The engine built will be able for general purpose application such as which can help save time on toll collection for vehicle which also reduce accident and traffic congestion at toll. Ionic with cordova or android studio is use for application development. MySQL server as database for our solutions. On the other hand, other application for example congestion charging systems, traffic monitoring and security, parking system also can be built with this system.

1.3 Project Structure

The body of the contents in this project is organized into eight chapters that are introduction, literature review, project management, financial consideration, technical design, environment and sustainability, health and safety and conclusion. The first chapter briefly describes the background and overview about the conducted work. Besides that, this chapter also covers problem statement, objective and the scope of the project. The second chapter covers important literature review related to the project. This chapter starts with the License Plate Detection. Then, this part also do some review on the text detection and segmentation. License Plate Detection with Text Segmentation under Tensorflow framework using python. Next is text and character recognition under Torch framework using LUA programming language. The third chapter is about the project management. For this chapter, the organization chart is given. This part will discuss about the task divided by our supervisor. Then, all of the engineers need to do their task within the given time. The fourth chapter covers the financial management. For this part, all the material and equipment used and their cost are listed out in tables. The total cost is calculated. The fifth chapter covers all the technical design that used to produce the final result. In this chapter, every part detailed design and operation are discussed part by part. The parts are including comparison of two different models in text detection and segmentation. The simulation result, trained results for License Plate Detection with yolo3 and text and character recognition using CRNN are discussed too. The sixth chapter covers

environment and sustainability. The sustainability for our project includes these two pillars which are economy and social development. For example, this system will save time and reduce parking congestion. This project is important in establishing the economy inside and outside of Malaysia. The seventh chapter covers health and safety. From this chapter, we explained how our system can reduce traffic congestion. We also concern about the security safety of our vehicle and the impacts on health due to the traffic congestion. Other than that, this chapter also remind us about the precaution for the implantation of the system. The eighth chapter concludes all the parts that we have done. This part will also conclude the overall achievement of this project. Future work also will be discussed in this project so that improvement can be made to sustain a better quality of project.

CHAPTER 2

LITERATURE REVIEW

2.0 You Only Look Once (YOLO)

Object detection is to recognize a particular object in an image or video. General purpose of object detection should be fast, accurate, and able to identify a wide variety of objects. You Only Look Once (YOLO) is not just a traditional classifier, it is a real time object detection. YOLO takes a completely different approach. YOLO runs single convolutional single network on full image to predict a (S, S, B×5+C) tensor. This network divides the image into a grid of S x S cells and predicts the bounding box (B) and the conditional class probabilities (C). Each bounding box consists of 5 predictions which are x, y, w, h, and confidence. The grid cell is responsible for detecting the object if the centre of that object falls into a grid cell. The confidence score of the bounding box and the conditional class probabilities combined into 1 final score that tells us the probability of that bounding box contain of certain type of object. The confidence scores show how confident the model contains a specific kind of object and how accurate the box is predicted. The score is zero if there is no object exists in the cell. To makes the final predictions, we only keep those scorings are more than 0.25. On a Pascal Titan X,YOLO v2 processes image in real time at 45 FPS while Fast YOLO processes at 145 FPS. Fast YOLO achieved double the mean average precision of others detector.

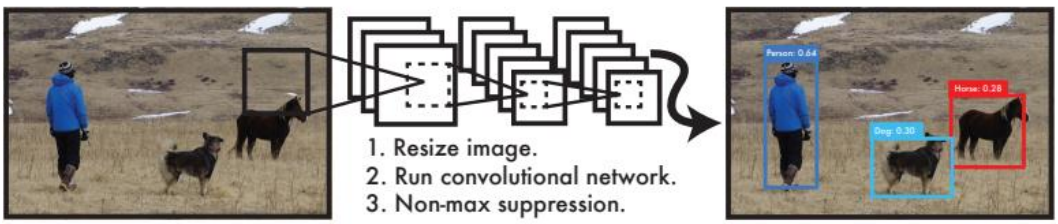


Figure 2.0.1: YOLO Detection System [5]

Comparing the speed and the accuracy of YOLO to others detector, Fast YOLO is the fastest detector on Pascal and its accuracy is twice times more accurate than any

other real-time detector. Besides, YOLO is similar with R-CNN as each grid cell proposes potential bounding boxes and bounding boxes are scoring using convolutional features. YOLO face the problem which unable to localize objects correctly whereas Fast R-CNN have fewer localization errors because it is almost three times more likely to predict background detections than YOLO. YOLO makes fewer background error as YOLO eliminate background detections from Fast R-CNN we get a significant boost in performance. Unlike classifier-based approaches, YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly. MultiBox replace the confidence prediction with a single class prediction to perform single object detection. However, MultiBoxcan only perform single object detection but cannot perform general object detection. This is because general object detection is a large detection and it require further image patch classification. Both YOLO and MultiBox use a convolutional network to predict bounding boxes in an image but YOLO is a complete detection system. [5]

We not only want the detection to be accurate, we also want it to be faster. In YOLO v2, we used a model called Darknet-19 which has 19 convolutional layers and 5 maxpooling layers. To process an image on Darknet-19, it only requires 5.58 billion operations yet achieves 72.9% top-1 accuracy and 91.2% top-5 accuracy on ImageNet. We use batch normalization to stabilize training, speed up convergence, and regularize the model. Also, we removed the last convolutional layer and adding on three 3×3 convolutional layers with 1024 filters each followed by a final 1×1 convolutional layer with the number of outputs we need to modify the network for detection.[6]

Moreover, YOLOv1 uses an end-to-end regression method without undergoes region proposal step and direct completes the position and category determination. This is the reason which makes YOLOv1 less accurate to locate the target which directly cause the accuracy of the YOLO's detection is not very high. YOLO uses fully connected layers to predict the coordinates or location of bounding box. However, YOLOv2 remove the fully connected layers from YOLO and use anchor boxes to predict bounding boxes. Although there is a small decrease in accuracy by using anchor boxes but the recall is increasing. Increase in recall means the model can still improve. Furthermore, YOLO trains the classifier network at resolution of 224×224 and YOLOv2 has increase the resolution to 448×448 for detection. High resolution classifier network can help us to get almost 4% improvement in mAP. Adding batch

normalization in convolutional layers in YOLO can gives an increase of more than 2% in mAP. [6]

YOLO is a fast and accurate object detector. Hence, it is ideal for computer vision applications. To maintain real time performance, we connect YOLO to a webcam and verify that. Fast YOLO is the fastest general-purpose object detector and YOLO pushes the state-of-the-art in real-time object detection. YOLO also generalizes well to new domains making it ideal for applications that rely on fast, robust object detection. Furthermore, YOLOv2 is way much faster than other detection systems so it can be run at different size of image to provide a smooth trade off between speed and accuracy.

2.1 The Text Detection And Segmentation On The License Plate

2.1.1 Textboxes++: A Single-Shot Oriented Scene Text Detector

TextBoxes plus plus (TextBoxes++)is an application for a single-shot oriented scene text detection and recognition (Convolutional Recurrent Neural Network, CRNN).TextBoxes++ is a unified framework for oriented scene text detection with a single network[7]. It is an extended work of TextBoxes. CRNN is an open-source text recognizer which is a combination of Convolutional Neural Networks (CNN)[8], Recurrent neural networks (RNN) models[9] and Connectionist Temporal Classification (CTC) layer[10] loss for image-based sequence recognition tasks, such as scene text recognition and Optical character recognition (OCR)[11]. The code of TextBoxes++ is based on single shot multibox detector (SSD) [12] and TextBoxes. The code of CRNN is modified from CRNN [13]. TextBoxes++ relies on an end-to-end trainable fully convolutional neural network to detect arbitrary-oriented text. The first basic idea is inspired by an object detection which aims to detect general objects in images. The object detection algorithm SSD proposed some special designs for adapting SSD network to efficiently detect oriented text in natural images. There are two mainstream CNN- based methods on this topic: R-CNN based methods [14] and YOLO-based "You only look once: Unified, real-time object detection," methods [15]. Next, follow by scene text reading system which composed of two main components: text detection and text recognition. The former component localizes text in images mostly in the form of word bounding boxes. The latter one transcripts cropped word

images into machine-interpretable character sequences. Most text detectors can be roughly classified into several categories following two classification strategies based on primitive detection targets and shape of target bounding boxes, respectively. Classification strategy based on primitive detection targets with methods categorized into three categories: 1. Character-based, 2. Word-basedand 3. Text-line-based. Classification strategy based on the shape of target bounding boxes with horizontal or nearly horizontal and multi-oriented.

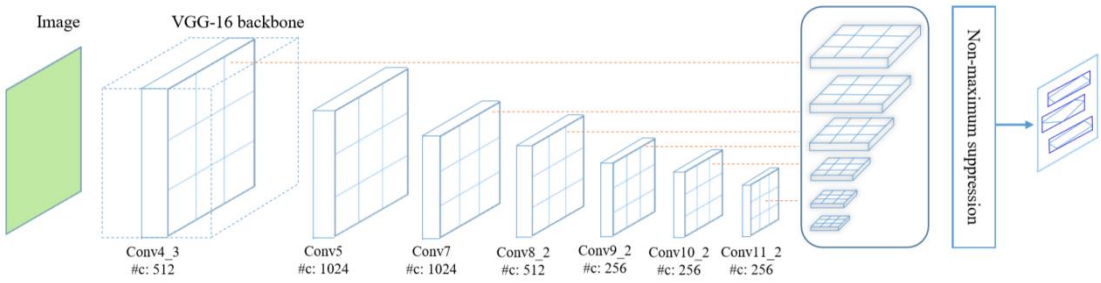


Figure 2.1.1.1: The Network Architecture Of Textboxes++

The architecture of TextBoxes++ is depicted in Figure 1, a fully convolutional network including 13 layers from VGG-16 followed by 10 extra convolutional layers, and 6 Text-box layers connected to 6 intermediate convolutional layers.[7] It keeping the layers from conv1_1 through conv5_3, and converting the last two fully-connected layers of VGG-16 into convolutional layers (conv6 and conv7) by parameters down-sampling. Each location of a text-box layer predicts an n-dimensional vector for each default box consisting of the text presence scores (2 dimensions), horizontal bounding rectangles offsets (4 dimensions), and rotated rectangle bounding box offsets (5 dimensions) or quadrilateral bounding box

offsets (8 dimensions). A non-maximum suppression is applied during test phase tomerge the results of all 6 text-box layers. Note that "#c" stands for the number of channels.[16] Another eight convolutional layers divided into four stages (conv8 to conv11) with different resolutions by max-pooling are appended after conv7. Multiple output layers, which we call text-box layers, are inserted after the last and some intermediate convolutional layers. They are also convolutional layers to predict outputs for aggregation and then undergo an efficient.

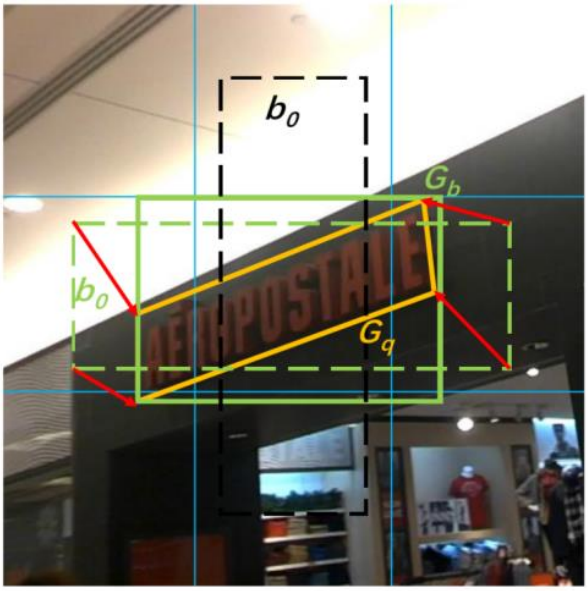


Figure 2.1.1.2: The Minimum Bounding Horizontal Rectangle

As shown in Figure 2.1.1.2, there are a number of default boxes with different aspect ratios at each location. Illustration of the regression (red arrows) from a matched default box (green dashed) to a ground truth target quadrilateral (yellow) on a 3 × 3 grid. The black dashed default box is not matched to the ground truth. The regression from the matched default box to the minimum horizontal rectangle (green solid) containing the ground truth quadrilateral is not shown for a better visualization.

2.1.2 EAST:An Efficient And Accurate Scene Text Detector

Currently, many active research topics approaches for scene text detection and recognition in computer vision for a long period of time and numerous inspiring ideas and effective approaches have already investigated and achieved promising performances across various bench mark shave. However, many previous work have challenging scenarios facing, the overall performance is determined by the interplay of multiple stages and components in the pipelines even when equipped with deep neural network models. EAST:An Efficient and Accurate Scene Text Detector propose a simple yet powerful pipeline that yields fast and accurate text detection in natural scenes. This model, the pipeline directly predicts words or text lines of arbitrary orientations and quadrilateral shapes in full images, eliminating unnecessary intermediate steps with aggregation and word partitioning, with a single neural network. The pipeline allows concentrating efforts on designing loss functions and neural network architecture. The pre-trained model are use standard data sets including ICDAR 2015[16][17], COCO-Text[18] and MSRA-TD500[19] demonstrate that the proposed algorithm significantly outperforms state-of-the-art methods in terms of both accuracy and efficiency. The proposed algorithm achieves an F-score of 0.7820 at 13.2fps at 720p resolution using only training images from ICDAR 2015 and 2013.

There are several factors taken account in EAST when designing neural network on text detection which are mostly relevant to the proposed algorithm. Conventional approaches rely on manually designed features. Stroke Width Transform (SWT) [20] and Maximally Stable Extremal Regions (MSER) [21, 22] based methods generally seek character candidates via edge detection or external region extraction. Zhang et al. [23] made use of the local symmetry property of text and designed various features for text region detection. FASText [24] is a fast text detection system that adapted and modified the well-known FAST key point detector for stroke extraction. However, these methods fall behind of those based on deep neural networks, in terms of both accuracy and adaptability, especially when dealing with challenging scenarios, such as low resolution and geometric distortion.

Recently, the area of scene text detection has entered a new era that deep neural network based algorithms [26, 30, 27] have gradually become the mainstream. Huang et al. [26] first found candidates using MSER and then employed a deep

convolutional network as a strong classifier to prune false positives. The method of Jadebergetal. [30] scanned the image in a sliding-window fashion and produced a dense heatmap for each scale with a convolutional neural network model. Later, Jaderberg *et al*.[29]employed both a CNN and an ACF to hunt word candidates and further refined them using regression. Tian et al. [25] developed vertical anchors and constructed a CNN-RNN joint model to detect horizontal text lines. Different from these methods, Zhangetal.[27] proposed toutilize FCN for heat map generation and to use component projection for orientation estimation. These methods obtained excellent performance on standard benchmarks. However, as illustrated in Fig. 3(a-d), they mostly consist of multiple stages and components, such as false positive removal by post filtering, candidate aggregation, line formation and word partition. The multtude of stages and components may require exhaustive tuning, leading to sub-optimal performance, and add to processing time of the whole pipeline. EAST devise a deep FCN-based pipeline that directly targets the final goal of text detection: word or text line level detection. As depicted in Fig. 3(e), the model abandons unnecessary intermediate components and steps, and allows for end-to-end training and optimization.

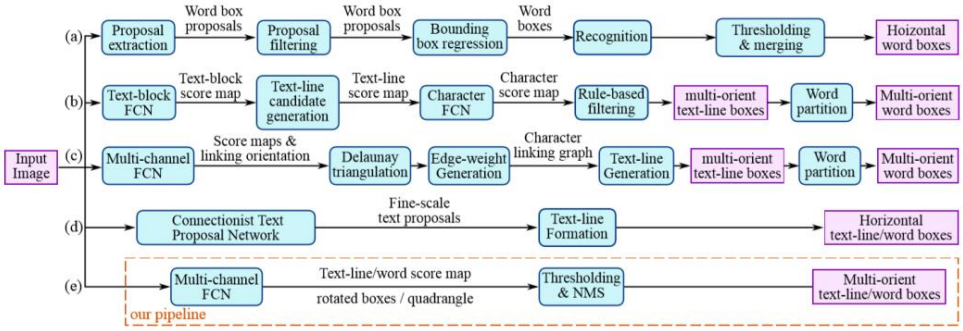


Figure 2.1.2.1: Pipelines Of Several Recent Works On Scene Text Detection

As illustrated in Fig. 2.1.2.1 comparison of pipelines of several recent works on scene text detection: (a) Horizontal word detection and recognition pipeline proposed by Jaderberg et al.; (b) Multi-orient text detection pipeline proposed by Zhang et al.; (c) Multi-orient text detection pipeline proposed by Yao et al.; (d) Horizontal text detection using CTPN, proposed by Tian et al.; (e) Our pipeline, which eliminates most intermediate steps, consists of only two stages and is much simpler than previous solutions.

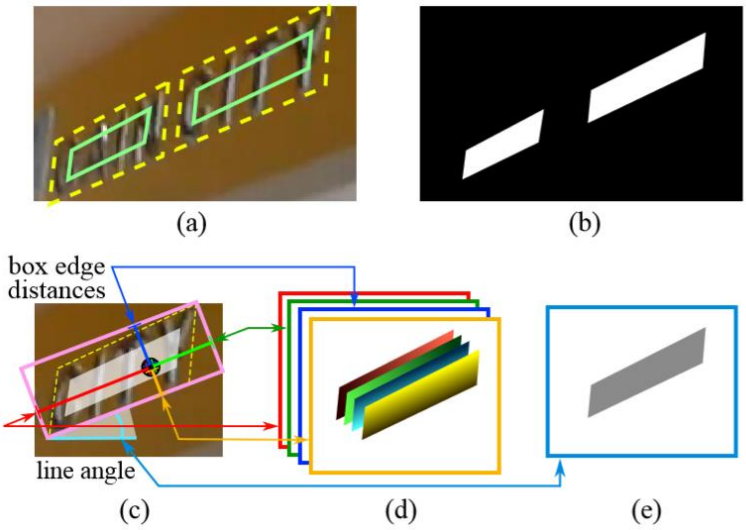


Figure 2.1.2.2: Label Generation Process

Figure 2.1.2.2 shows the label generation process. (a) Text quadrangle (yellow dashed) and the shrunk quadrangle (green solid); (b) Text score map generation for quadrangle; (c) RBOX geometry map generation is either one of RBOX or QUAD. Without loss of generality, the case only consider geometry is a quadrangle. ForRBOX, the geometry is represented by 4 channels of axis-aligned bounding box (AABB) R and 1 channel rotation angle θ. The formulation of R is the same as that in [9], where the 4 channels represents 4 distances from the pixel location to the top, right, bottom, left boundaries of the rectangle respectively. For those datasets whose text regions are annotated in QUAD style (e.g.,ICDAR2015),rotated rectangle IS first generated that covers the region with minimal area used 8 numbers to denote the coordinate shift from four corner vertices $\{p_i|i\in\{1,2,3,4\}\}$ of the quadrangle to the pixel location. As each distance offset contains two numbers $(\Delta x_i,\Delta y_i)$, the geometry output contains 8 channels. (d) 4 channels of RBOX distances of each pixel to rectangle boundaries. (e) Rotation angle of the bounding box and line angle.

2.2 Convolutional Recurrent Neural Network (CRNN)

Baoguang Shi, Xiang Bai and Cong Yao (2015) conducted a research to investigate the problem of text recognition, which is the most important part for image based sequence recognition. CRNN is the combination of the Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) and Connectionist Temporal

Classification (CTC). CRNN is also an end to end trainable neural network for image recognition and text recognition. This network is used to recognize the text and character from an image directly. The network architecture of CRNN is starting from convolutional layer to recurrent layer and the last step is transcription layer. [31]

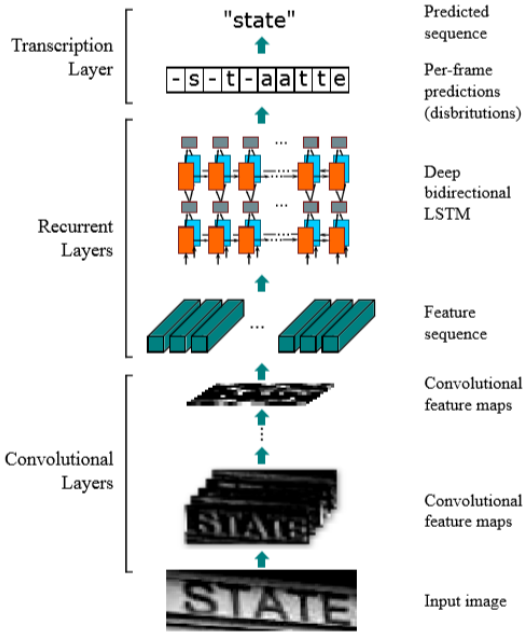


Figure 2.2.1: The Network Architecture [31]

2.2.1 Convolutional Layer

Convolutional layer is also called as Convolutional Neural Network (CNN). Convolutional layer consists of three components which are convolutional layer, activation function and max pooling layer. Convolutional layer is the multiplication or convo operation (3x3) with the input image and filter to produce a feature map. Filter is called as Kernel, which is a training dataset that used to convo with input image. Activation function is using RELU to make the model to learn non-linear function. RELU change the negative value to zero. After the convolutional layer with activation function, max-pooling layer (2x2) is used to reduce the size feature map. The steps will keep continuous until the fully connected layer. Convolution layer until fully connected layer is called as hidden layer. The whole process is used to extract the feature sequence from an input image.

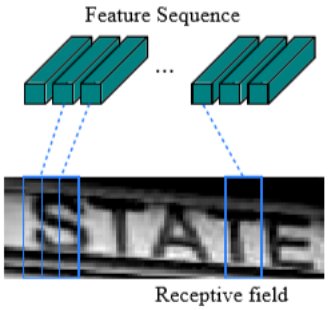


Figure 2.2.1.1: Feature Sequence [31]

2.2.2 Recurrent layer

Recurrent Neural Network (RNN) is mainly designed for handling sequences. The recurrent layers predict the label distribution for each frame in feature sequence.RNN will suffer the problem of limit the range of context and adds a burden to the training process. Therefore, Bidirectional Long Short Term Memory (BLSTM) is proposed to solve the problem. LSTM contains a memory cell and three multiplicative gates, which can store the contexts for a long period of time and capture long range dependencies between sequential features. BLSTM processes the one of the feature sequences forward and one of the feature sequences backward.

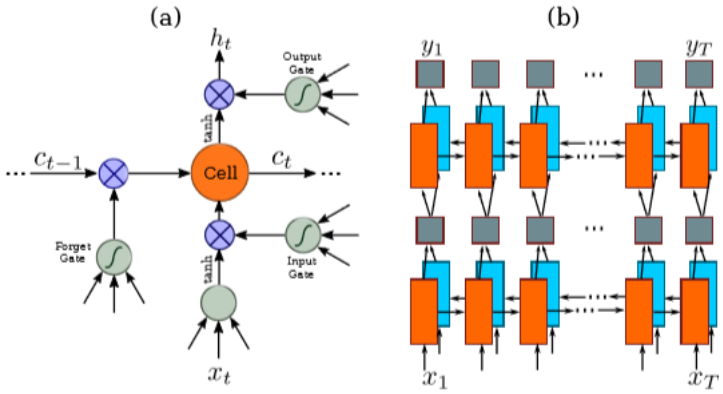


Figure 2.2.2.1 (a) show the structure of a basic LSTM. [31]

Figure 2.2.2.2 (b) show the structure of BLSTM [31]

2.2.3 Transcription Layer

Transcription is the process of converting the per-frame predictions made by RNN into a label sequence. Transcription layer is also called Connectionist Temporal Classification (CTC). CTC is a loss function useful for performed supervised learning on sequence data without alignment. Besides, label input no need to specific the alignment. CTC also reduce the complexity of computation. Transcription layers have

two modes such as lexicon free and lexicon based transcription. The predictions are made without any lexicon is lexicon free transcription. The predictions are made by choosing the label sequence that has the highest probability is lexicon based transcription.

5.3 Cascaded Network and Desktop Application

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

- Tkinter − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- wxPython − This is an open-source Python interface for wxWindows http://wxpython.org.
- PyQt − PythonQt is a C++ port for python which gives Python scripts seamless access to c++ class libraries on the local machine.

There are many other interfaces available, which you can find them on the net.

Tkinter Programming: Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Tkinter Widgets: Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

There have already been two books written about wxPython, making it worth a mention even if it isn't quite ready for Python 3. WxPython is based on wxWidgets, a cross-platform GUI library written in C++. In addition to the standard dialogs, it includes a 2D path drawing API, dockable windows, support for many file formats and both text-editing and word-processing widgets. There's a great set of demos provided with wxPython, along with several sets of tutorials to help get you started. Given that wxWidgets has a 22-year development pedigree, this is one of the most popular frameworks.

Qt is a multi-licensed cross-platform framework written in C++. If your application is completely open source, you can use Qt for free under the community license; otherwise you'll need a commercial license. Qt has been around for a long time and was owned by Nokia for a while; it's a very comprehensive library of tools and APIs, widely used in many industries, and covers many platforms including mobile. If a gadget such as a SatNav has a GUI, there's a good chance it'll be Qt based.

2.4 Development of Mobile Applications

Mobile app development is the act by which a mobile app is developed for mobile devices. Mobile applications are very important in our daily life and the growing availability of diverse interactive mobile applications assists us to fulfil our needs for communication, entertainment and get information. [32] Mobile applications are a set of designed computer program that runs on a mobile device and perform certain tasks for the user. Mobile application is user friendly, downloadable and able to run in most of the mobile phone. Mobile apps are first tested within the development environment by using emulators which enables the host system to run software or for the guest system. Mobile applications can be pre-installed on phones or delivered as web applications using server-side processing (e.g. JavaScript) to provide an interactive experience within a Web browser. [33]

Table 2.4.1: History of Mobile App Development

Psion EPOC	Palm OS	Symbian
		
EPOC was the first recognizable apps released in 90s which allowed users programs such as a word processor, database. Later models allowed users to add additional apps through software packs. EPOC was programmed in OPL (Open Programming Language) and allowed users to build their own apps.	Palm released a new generation of machines in 1996 which utilized the Palm OS. This had a touch screen GUI and came with a raft of basic apps as well as tons of third party apps programmed in C/C++. A WAP browser is included from Palm OS 3.0 onwards.	Symbian was originally developed by Symbian Ltd, a joint venture of Psion, Ericsson, Motorola and Nokia. In 2009, around 250 million devices were running Symbian. The last Symbian smart phone was the Nokia 808 Pure View ran the Nokia Belle update which used C++.

Back in the 1990's, it was created by device manufacturers like Nokia and Motorola and those apps were only able to carry out basic tasks. Later on, wireless service providers started to make apps to differentiate the devices sold on their network to others. At the same time, third party companies started to build apps for the mobile platforms like the Windows mobile OS and the Symbian OS which included games for the devices and other utility apps. Matthew Panzarino (co-editor of TechCrunch) has proposed three phases of app development that took mobile technology from telephony to gaming and utilities, to apps that wanted to be "home screen" and dominate experience, to apps armed with content that "are more about maximizing their usefulness without dominating attention [33]. The content produced is leading us towards handling apps in a different way. The most persuasive concept is the idea of the "card" which is a design treatment that can be seen on Twitter and Facebook across different devices and this idea allows the content to be aggregated and presented to the user in a consistent way.

Besides, app developers will design a single experience that will stretch across any internet connected terminal based on the idea of multi-screen. The concept of cards was suggested and how content presented in this way could work well and meaningful for users on almost any type of screen. In 2000, the mobile application developers were talking about internet based mobile application which ensure people can connect to Internet interacting with world, getting information, communicating using Facebook or Twitter and identifying location by mobile applications [33]. Those mobile applications are being designing to simplify our daily life.

In the development process, user interface (UI) design is also an essential element in the building of mobile apps. UI is the space where interactions between humans and machines occur. The main purpose of user interface design is to produce an easy, efficient, and user-friendly user interface to operate a machine which produces the desired result. Mobile UI considers contexts, input, screen, and mobility as outlines for design. The components of both hardware and software are required for the interface. User input allows for the users to manipulate a system, and output of the device allows the system to indicate the effects of the users' manipulation. Mobile UI contexts data from user activity, such as location and scheduling that can be obtained within a mobile app [32].

Today, hybrid mobile applications have been developed using various technologies. One of the most common approaches often being used is Software Development Kids (SDK) as it was the major development technique which capitalizes on native programming languages such as Java, C*, .NET and their individual SDKs. All these methods rely on individual platforms such as Windows, iOS, and android. A hybrid mobile application uses the cross platform frameworks such as Qt, Xamarin or Ionic for mobile development. They are built using multi-platform web technologies for example HTML5, CSS and Javascript. Hybrid multi-platform apps are fast and relatively easy to develop and low-cost maintenance and smooth updates just by using single code base for all platforms. [34]

CHAPTER 3

PROJECT MANAGEMENT

3.0 Supervisor

Name: PROFESOR DR. ZULKALNAIN BIN MOHD YUSSOF
Task: Team advisor. Advise engineers in software.

3.1 Leader

Name: IBRAHIM SOLIMAN

Task: Plans and manages the project to ensure the success of the project. Arrange the time for minutes meeting. Manage and separate all the members work that they need to focus on. Study on all the frameworks that members need to do. Help the members solve their problems when they cannot solve it. Combine all the works together.

3.2 Engineer 1

Name: WENG SHIH YING

Task: Designs and engineers the project in object detection. Do the object detection by using YOLO version 3. Collect own datasets of license plate cars around 2k images with different angles. Label the images by using labelImg-master. Coding in python to arrange the images in sequences.Train the datasets and test the new datasets.

3.3 Engineer 2

Name: CHEONG JIA MUN

Task: Designs and engineers the project in text detection. Do the text detection by using The EAST. Train the dataset to extract the text images in the images of the whole car view. Coding the bounding box can rotate. Test the dataset that have done after the Engineer 1.

3.4 Engineer 3

Name: YEW POH LENG

Task: Design and engineers the project in text and characters recognition. Do the text and characters recognition by using Pytorch. Search the type of fonts of the license plate. Coding to detect the blurriness and rotation of the text images.Generates the text images to train the weight. Test the new weight.

3.5 Engineer 4

Name: WONG YUIN SHEN

Task: Design and engineers the project in apps builder. Build an app that can check the information of the car driver based on license plate by using Java. Help engineer 1 to collect the 2k images.

3.6 The Organization Chart

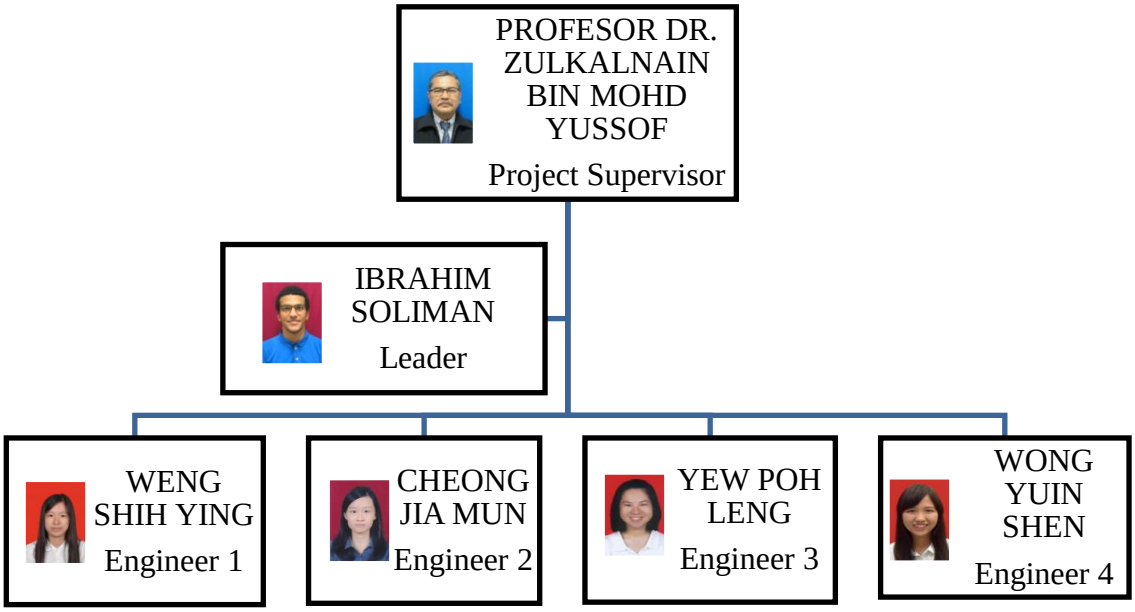


Figure 3.6.1: Organization Chart

CHAPTER 4

FINANCIAL CONSIDERATION

In order to build prototype for this project, the desired equipment or software are brought to develop the prototype. The financial for the project Deep Neural Network based License Plates Detection Recognition (LPDR) system will be explained clearly in this part.

4.0 Prototype Costing

Hardware: 2 outdoor Camera, PC with graphics processing unit (GPU)

Software:License Plate Detection with Text Segmentation under Tensorflow framework using python. Text and character recognition under Torch framework using LUA programming language.Ionic with cordova or android studio for application development.MySQLserver as database for our solutions.

Table 4.0.1: The estimation cost for the prototype

No.	Equipment/Software	Quantity	Price (RM)	Amount (RM)
1.	Security camera CCTV	2 unit	800/unit	1,600.00
2.	Camera stand	2 unit	90.00/unit	180.00
3.	LAN Cable (2 cable)	400 meters	1/meter	400.00

4.	PC with graphics processing unit (GPU)	1 unit	2,000.00 /unit	2,000.00
5.	Ubuntu - Tensoflow and torch framework	FOC	FOC	FOC
6.	MySQL Enterprise Edition web hosting	1	400.00	400.00
7.	MySQL Enterprise Edition domain name reservation	1	80.00	80.00
8.	Ionic for mobile apps	FOC	FOC	FOC
	TOTAL			RM 4,180.00

The total cost in constructing Deep Neural Network based License Plates Detection Recognition (LPDR) system is RM 4,180.00. The estimation is in the range of the budget and not all the equipment must being brought. Some of the equipment can be obtained from in our university. All software are freely download able version which no need to pay. With the cost that estimated by our group, the project is proceed to the next step.

CHAPTER 5

TECHNICAL DESIGN

5.0 Object Detection Using Yolov3

5.0.1 Preparation for Datasets

In this project, we are required to collect our own datasets as we need huge amounts of datasets for the training purpose. After we finished collect the datasets, we labelled each image with our custom objects which are “ROI”, “plate” and “text” by using LabelImg-master. LabelImg is a graphical annotation tool that written in Python and used Qt for its graphical interface.All the annotations will be save in XML format. Before we start label, we converted all the images formats from jpg to JPEG and changed the all the file name of the images in directory. The scripts we used are shown in the figure below

```
import os,sys
folder = '/home/shihying/Downloads/test'
for filename in os.listdir(folder):
        infilename = os.path.join(folder,filename)
        if not os.path.isfile(infilename): continue
        oldbase = os.path.splitext(filename)
        newname = infilename.replace('.jpg', '.JPEG')
        output = os.rename(infilename, newname)
```

Figure 5.0.1.1: Python script to convert image format from jpg to JPEG

```
import os
path = '/home/shihying/yolotraining/data 1'
filename=[]
files = os.listdir(path)
#for file1 in files:
#        filen, file_extension = os.path.splitext(file1)
#        filename.append(filen)
#filename.sort(key=int)
i = 1
print(files)
for file in files:
    os.rename(os.path.join(path,file), os.path.join(path,str(i)+'.JPEG'))
    i = i+1
```

Figure 5.0.1.2: Python script to change all the file name of the image

5.0.2 Preparation for Training

In this part, YOLOv3 is selected to detect the custom object. Before we start training, we installed the Ubuntu on the computer. After we installed Ubuntu, we installed Darkflow on GitHub. The dependencies needed to install Darkflow are Python3, TensorFlow 1.0, Numpy, OpenCV 3. However, we are facing problem while installing TensorFlow 1.0 and open cv 3. This is because TensorFlow 1.0 require CUDA version 8.0 but the CUDA version that we have are 7.5. So, we updated our CUDA version to 8.0.

To get started, we first built the Cython extensions in place. After that, we let pip installed Darkflow globally in dev mode and installed with pip globally.

```
python3 setup.py build_ext—inplace
pip install -e
pip install
```

We ran the real time detection demo on a Webcam. We connected a webcam to the computer that OpenCV can connect. We compiled the Darknet with CUDA and OpenCV to run this this demo. The command we used are

```
./darknet detector demo cfg/coco.datacfg/yolov3.cfgyolov3.weights
```

Before we start training on our own datasets, we trained the pre-trained weight for YOLOv3.

```
git clone https://github.com/pjreddie/darknet
cd darknet
make
```

The configuration file for YOLO is in the cfg/ subdirectory.We downloaded the pre-trained weight by ran this command.

```
wget https://pjreddie.com/media/files/yolov3.weights
```

Then we ran the detector by this command.

```
./darknet detect cfg/yolov3.cfg yolov3.weights data/25.jpg
```

5.0.3 Training

We trained YOLO on Pascal VOC format. We gathered all the training datasets and separated it into testing and training folder. Inside each folder, we also split the images and the annotations into two folders. We wrote all the images for training to the text file.

```
import os

folder = "/home/ubuntu/voctest/VOCdevkit/VOC2018/JPEGImages"

def write_file(data):
    """
    this function write data to file
    :param data:
    :return:
    """
    file_name = r'/home/ubuntu/voctest/VOCdevkit/VOC2018/ImageSets/Main/train.txt'
    with open(file_name, 'a+') as x_file:
        x_file.write('{}\n'.format(data))


for file in os.listdir(folder):
    name, ext = os.path.splitext(file)
    print name
    write_file(name)
```

Figure 5.0.3.1: Python script to write all the images to .txt file

i) Get the Pascal VOC Data

First, we need to all the Pascal VOC data from 2007 to 2012 to train YOLO. We downloaded the data from YOLO website. We make a directory to store all the data and we ran the command below from the directory.All the VOC training data located at subdirectory named VOCdevkit/ after we ran the command.

```
wgethttps://pjreddie.com/media/files/voctrainval_11-may-2012.tar
wgethttps://pjreddie.com/media/files/voctrainval_06-nov-2007.tar
wgethttps://pjreddie.com/media/files/voctest_06-nov-2007.tar
tar xf voctrainval_11-may-2012.tar
tar xf voctrainval_06-nov-2007.tar
tar xf voctest_06-nov-2007.tar
```

ii) Generate labels for VOC

Next, we generated labels for VOC Pascal data. Darknet wants a text file for each image with a line for each ground truth object in the image. We ran the voc_label.py script in the Darknet directory to generate the labels. The text file listed all the images as Darknet train on a text file with all the images. We changed the sets and classes as shown below:

```
sets=[('2018','train'),('2019','test')]

classes = ["ROI", "plate", "text"]
```

iii) Change the Cfg in yolov3-voc-weng.cfg

Next, we copied the yolov3-voc.cfg inside Cfg folder and renamed as yolov3-voc-weng.cfg. Then we changed the data in the yolov3-voc-weng.cfg configuration file. We changed the classes to 3 as we detect 3 custom objects which are “ROI”, “plate” and “text”. Filter changed to 24 using the formula 3*(number of classes+5) = 3*(3+5) = 24. [Figure 1.5] We changed the filter to 24 for each convolutional layer before every YOLO layer. Batch size is the number of samples that going to be propagated through the network. Subdivision is to split the batch into mini batch. For testing, we set both the batch size and subdivision to 1. Whereas for the training, we changed the batch size to 64 and subdivision to 8 which means the network will split into 8 mini batch and sent to the GPU. We split the batch into mini batch because mini

batch require less memory as we train using less images.Each mini batch will have 8 images (64/8 = 8). The process will then repeat for 8 times until the batch is completed. Once this batch is completed, new iterations will start with the new 64 images and repeat the process until the whole training process is completed. Network trains faster with mini batch because we updated weights after each propagation. We also changed the image size height and width from 416 to 832.

Anchor box is the predetermined set of boxes with particular height and width. We generated our own anchor box using K-means algorithm. [Figure 5.0.3.4] We cloned the AlexeyAB/darknet forked from pjreddie/darknet. Then we ran the command below in the darknet directory to get the anchor box.

```
darknet.exe detector calc_anchors data/obj.data -num_of_clusters 9 -width 416 -height 416
```

iv) Modify cfg for Pascal VOC

```
[convolutional]
size=1
stride=1
pad=1
filters=24
activation=linear

[yolo]
mask = 6,7,8
anchors = 20,7,  42,9,  62,12,  76,61,  139,55,  193,76,  233,90,  200,171,  310,139
classes=3
num=9
jitter=.3
ignore_thresh = .5
truth_thresh = 1
random=1
```

After that, we changed the cfg/voc.data configuration file inside the Darknet directory. We changed the number of classes and the directory as shown below.

```
classes= 3
train  = /home/ubuntu/voctest/2018_train.txt
valid  = /home/ubuntu/voctest/2019_test.txt
names = /home/ubuntu/darknet/data/voc.names
backup = backup
```

v) Download pretrained convolutional weights

We used convolutional weights that are pre-trained on Imagenet for training our datasets. Instead of using Darknet-19, YOLO v3 used a new network to perform featureextraction. The network used a model called Darknet-53 which has 53 convolutional networks. This is because the function of Darknet-53 is much better than Darknet-19. We then downloaded the weights for the convolutional layers by running the command.

```
wget https://pjreddie.com/media/files/darknet53.convo.74
```

vi) Train the model

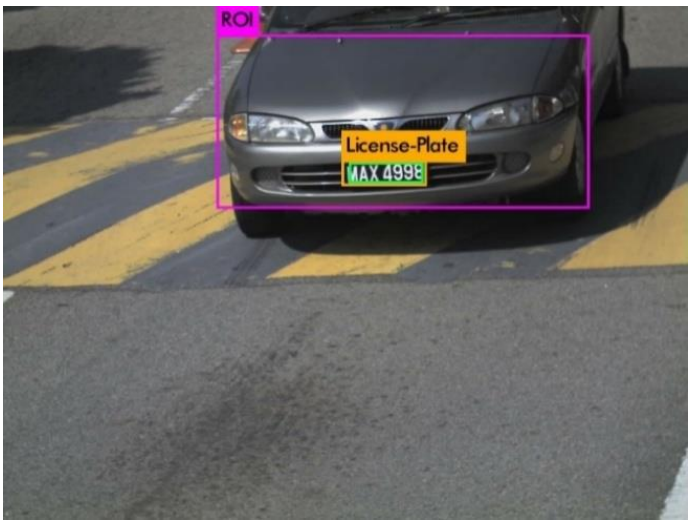
Lastly, we trained the model by run the detector using command below.

```
./darknet detector train cfg/voc.datacfg/yolov3-voc-weng.cfg darknet53.conv.74
```

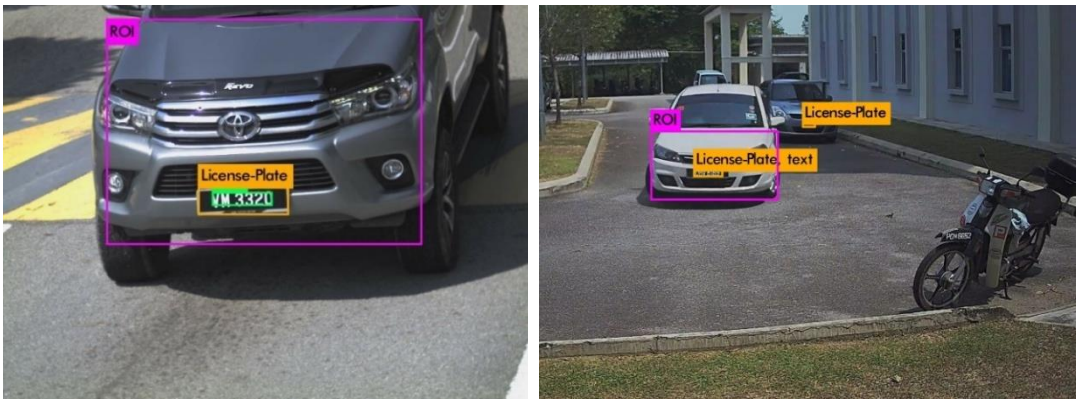
5.0.4 Results

The results that we obtained when we ran on the YOLOv3 detector are shown below. We also able to detect multiple objects in an image.

(A)



(B) (C)



(D) (E)

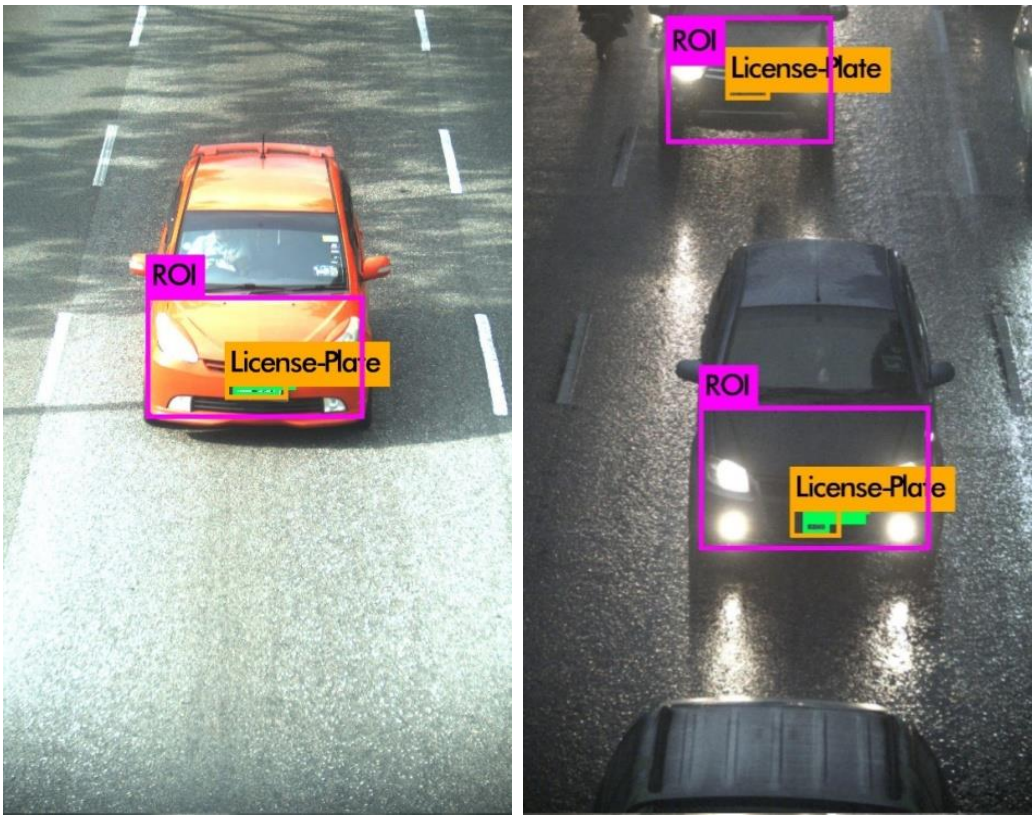
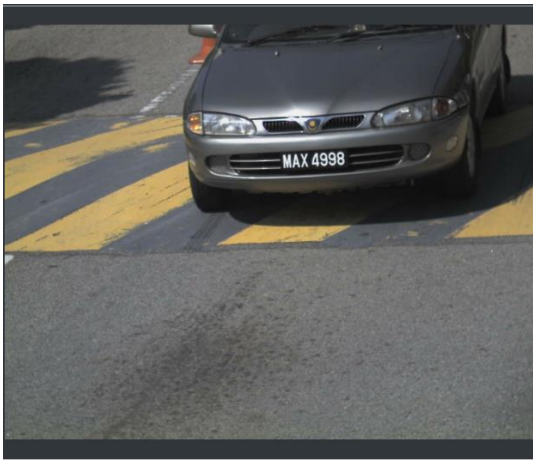


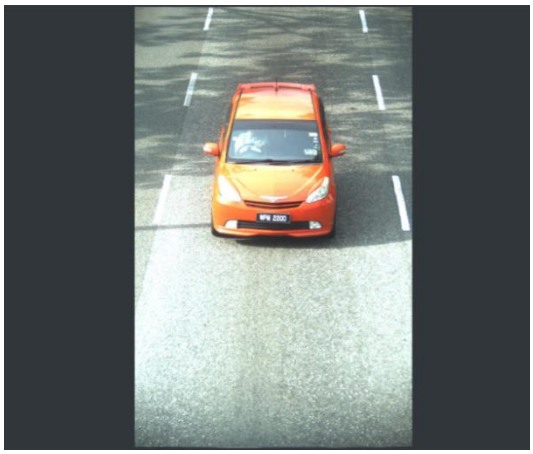
Figure 5.0.4.1 (A);(B);(C);(D);(E): Results run on YOLO detector

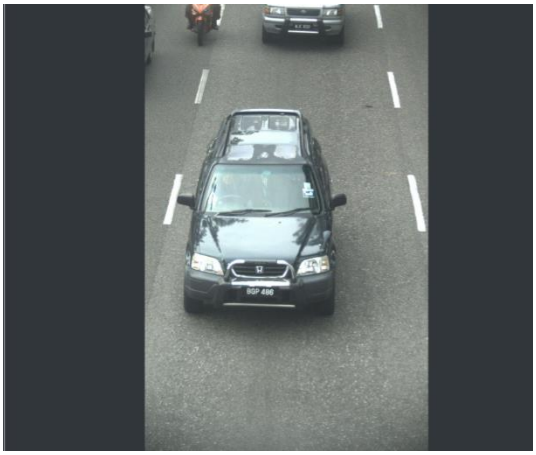
After the whole training is done, our system is able to detect our custom objects which are “ROI”, “License-Plate”, and “text”. When we put an input image or stream video in our Deep Neural Network based License Plates Detection-Recognition System, the ROI, license plate and the vehicle plate number are shown in the output.

Table 5.0.4.2: Results of Deep Neural Network based LPDR System

Input Output







5.1 Text Detection

5.1.1 Textboxes++: A Single-Shot Oriented Scene Text Detector

A demo testing server is setup for testing the result of TextBoxes++ when given a license plate image. Some environment is required to be download or install in Ubuntu. Torch 7 for CRNN, cuda 8.0 and cudnn v5.1 and opencv3.0. The model trained on ICDAR 2015 Incidental Text are downloaded from the open source given and use to setup demo testing server for comparison on result and accuracy. Before run the demo, download CRNN model and place the crnn model in "./crnn/model/" and the ICDAR 2015 model and place it in "./models/". Place the image in the image file and un the code “python examples/text/demo.py”. Finally, a text file will be then written to the output path. The written text file indecates the bounding box coordinates of the text detected from the input image. The detection results and recognition results are in "./demo_images"

Figure 5.1.1.1: The license plate before and after text detection by the TextBoxes++

The figure 5.1.1.1 shows the output of the TextBoxes++ for text detection on license plates.

Figure 5.1.1.2: The license plate before and after text detection by the TextBoxes++

According to the testing result, the accuracy of the textboxes plusplus is not accurate as mentioned in the paper. There are many text are missing detected as shown in figure 5.1.1.2.

There are also a problem in occur in textboxes plusplus which is the accuracy of the bounding box form when detect the text. Many bounding box are not form according to the text rotation angle from the origin. This situation will cut off some text detail during the image crop, a faulty or missing data image send to CRNN which will reduce the accuracy of the project.

Figure 5.1.1.3: The license plate before and after text detection by the TextBoxes++

5.1.2 EAST:An Efficient and Accurate Scene Text Detector

EAST is a tensorflow re-implementation, a fast Locality-Aware NMS in C++ provided by the paper's author. There are some differences from original paper [14] the train demo use ResNet-50 rather than PVANET. On the same time, dice loss (optimize IoU of segmentation) is used rather than balanced cross entropy. The third part that different with the original paper by Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang is that they use linear learning rate decay rather than staged learning rate decay.[14]According to the ICDAR 2013 (training set) + ICDAR 2015 (training set), EAST use more than 50,000k label image as training data set. It needed many times to achieve this accuracy, therefore we decided to compare different text detection and segmentation project to choose the most suitable model for our object. Download any version of tensorflow version more

than 1.0 to run the pre-trained model. The models trained on ICDAR 2013 (training set) + ICDAR 2015 (training set) and Resnet V1 50 provided by tensorflow slim: slim resnet v1 50 was download and place inside the EAST file. The image needed to be detected are located in "./tmp/images/". The test run is start using the code:"pythoneval.py--test_data_path=/tmp/images/--gpu_list=0-checkpoint_path=/tmp/east_icdar2015_resnet_v1_50_rbox/ \--output_dir=/tmp/".

The detection results and recognition results are in "./tmp/"

Figure 5.1.2.1: The License Plate Image Before The Detection By EAST

Figure 5.1.2.2: The Text Detection On The License Plate By EAST

EAST only detect text and draw bounding box according to the text with rotated bound box in the image but this situation will cause problem for CRNN for text and character recognition. Hence, the detected bounding box are needed to cropped out, connected in loop and rotate the rotated image to actual top of the image. In order to provide a nice image for CRNN, OpenCV and Perspective Warping (Step 5 of 6) by Adrian Rosebrock is use to warping the image. The image is first determined the coordinate in order to crop out the image and rotate it back to the actual top angle. A String is added to merge the total bounding boxes detected in a single image and create a new image.

Figure 5.1.2.3: Input Of Image Before Cropped And Rotate

Figure 5.1.2.4: The Image Is Cropped And Rotated

Figure 5.1.2.5: Output Of Image After Cropped And Rotate Image With Warping

EAST is a tensorflow re-implementation. Only RBOX part is implemented. This is one of the major difference between EAST and textboxes plusplus. According to the testing result, the accuracy of the textboxes plusplus is not precise as it mentioned in the paper. There are also a problem in textboxes plusplus which is the accuracy of the bounding box when detect the text. There are a high chance the bounding box rotation angle are not according to the angle of the text from the origin.

Figure 5.1.2.6: Single Character Missing In Detection Textboxes++

Figure 5.1.2.7: Single Character Missing In Detection In East

As shown in figure above, there are a same problem facing in textboxes plusplus and EAST which is single character missing in detection. According to EAST author, too small text area is ignored during EAST training. Therefore, the accuracy of detecting single character are low for both model. In order to improve the accuracy in the future, the parameters should tune and retrained the model and EAST model can not work well. As the comparison of the accuracy and the results detected. EAST will be more suitable and to use as it is more precise.

The advantage of the East is that it can detect and draw bounding box according to the text with rotated bound box. Conclusion, EAST:An Efficient and Accurate Scene Text Detector is chose as our project text detection and segmentation for license plate.

5.2 Text And Character Recognition Using Torch And Pytorch

5.2.1 Generates 1M text datasets

In this project, we have installed the Ubuntu on the laptop. After installing the Ubuntu, we installed the python 3 on Ubuntu and the step shown in below:

sudo add-apt-repository ppa:jonathonf/python-3.6
sudo apt-get update
sudo apt-get install python3.6

sudo update-alternatives --install /usr/bin/python3 python3 /usr/bin/python3.6 2
sudo update-alternatives --config python3

Figure 5.2.1.1: Install python 3

We are required to generate 1M of text datasets for training purpose. We required to install Font Creator to export the types of font based on license plate fonts. Besides, the character of the text in license plate do not have “O” and “I”, since both of them is similar to the number “0” and “1”. The types of font and number of datasets that we used are based on the license plate in Malaysia and shown in the table below:

Table 5.2.1: Types Of Font And Number Of Datasets That Generates

Types Of Font	Dataset That Generate
Arial Bold	350k
Arial Narrow Bold	350k
Kaiti	100k
FE	100k
Wright	100k

We generated the text datasets with rotation angle and the blurriness of text that can be seen by human being. We have set the range of rotation and blurriness in the same range and we have looped to get the datasets without fixed their rotation with blurriness. Besides, the datasets are saved in png format. We use python script to generate the text datasets that show in the below:

```
x in range (0,5):
for m in range (0,5):
    for y in range (0,2):
            plate1 =''.join(random.choice(letters) for _ in range(3))
            #plate1 = plate1+'  '
            plate2 =''.join(random.choice(numbers) for _ in range(4))
            #plate2 = plate2+'  '
            plate3 =''.join(random.choice(letters) for _ in range(1))
            plate = plate1 + plate2 + plate3
            fontsize = 1  # starting font size

            # portion of image width you want text width to be
            img_fraction = 0.99
            font = ImageFont.truetype(font_name,fontsize)

            while (font.getsize(plate)[0] < img_fraction*imgW) and (font.getsize(plate)[1] < 0.85*imgH):
              # iterate until the text size is just larger than the criteri
                 fontsize += 1
                 font = ImageFont.truetype(font_name,fontsize)

            # optionally de-increment to be sure it is less than criteria
            fontsize -= 1
            font = ImageFont.truetype(font_name, fontsize)


            print (plate)

            img=Image.new("RGBA", (imgW,imgH),(0,0,0))   #create empty image
            draw = ImageDraw.Draw(img)
            w,h = draw.textsize(plate,font=font)
            draw.text(((imgW-w)/2,((imgH-h)/2)/2),plate,(255,255,255),font=font)
            draw = ImageDraw.Draw(img)
            #draw = ImageDraw.Draw(img)
            plate=plate.replace(" ","")

            img = img.filter(ImageFilter.GaussianBlur(x))
            rotate = img.rotate(m)

            img1=Image.new("RGBA", (imgW,imgH),(0,0,0))   #create empty image
            img1.paste(rotate,(0,0),rotate)


            img.save('/home/yew/Generator/fe/plate'+str(y)+str(x)+'_'+plate+'.png')
            img1.save('/home/yew/Generator/fe/platedd'+str(y)+str(x)+'_'+plate+'.png')

            #rotate.save('/home/yew/Generator/hi/plater'+str(y)+str(x)+'_'+plate+'.png')
```

Figure 5.2.1.2: Python Script For Rotation And Blurriness Of The Text Datasets

5.2.2 Fine Tune The Original Datasets

After we have done to generate 1M text datasets, we required to separate each of them into two folders which is Training that contain 700k and Validation that contain 300k. We required change the png format to JPEG format. Then we required to create a LMDB dataset before train the data. We have installed the LMDB and the python script shown in below:

```
import os
import lmdb # install lmdb by "pip install lmdb"
import cv2
import numpy as np
```

Figure 5.2.2.1: Import the LMDB

5.2.3 Training

5.2.3.1 Train LMDB Dataset By Using Torch

In this project, we required to install Torch before we train the LMDB datasets. We used the framework Torch to train the LMDB dataset by using Lua language.

```
thmain_train.lua .. /models/foo_model/
```

Figure 5.2.3.1.1: Train the LMDB weight without save the weight

But this script has not got the accurate percentage for first row prediction, and saves it to the second prediction. Therefore, we have added the new script and shown it below:

```
thmain_train.lua .. /models/foo_model/ ../models/foo_model/crnn_demo_model.t7
```

Figure 5.2.3.1.2: Train the LMDB weight with save the weight

5.2.3.2 Test The Datasets

After we have done the training, we have started to test the new weight and we found that it works at Ubuntu 14 whereas it does not work at Ubuntu 16. Although we still got the result of text recognition, but our laptop and computer is installed with Ubuntu 16. The reason of the Torch cannot run in Ubuntu 16 is because one of the lib inside the Torch which is th++. The th++ is the C++ tensor lib, so we change to use the framework pytorch.

5.2.3.3 Transfer The LMDB To Another Computer With SSH

We run the training at another computer and the size of the file is too larger to transfer the LMDB faster by using SSH compare to pendrive. Secure Shell (SSH) is useful to transfer the file from computer to computer directly by using IP address.

```
sshubuntu@10.65.11.230
```

```
# ssh destination of folder@IP address from other computer
```

cd CRNN/crnn/model/lmdb/
cd directory of folder from the computer need to transfer the file/directory that save the file/directory that save the file/folder name that save the file/
scp. / Training1Million.mdb yew@10.65.13.112:/home/yew/
copy./ the file that want to transfer destination from own computer@IP address from own computer:/directory that save the file into own computer

Figure 5.2.3.3.1.: Transfer file by using SSH

After we are done to transfer the LMDB weight, we need to create two folders to save as Train and Test. Inside the both folders, we have to change the name of mdb file to the original name which is data.mdb and lock.mdb. Since we have tried it before that the mdb file name did not use the original name, we cannot run the training.

5.2.4 Train the LMDB weight by using Pytorch

In this project, we required to install the pytorch on Github to train the weight directly in the python language. Before that we have trained the weight by using Torch and we have to change the lua language to the python language if we want to read it. We have to install the BaiduNetdisk and put the file crnn.pth into the directory. Besides, we have to install the torch in the pytorch framework and shown in the below:

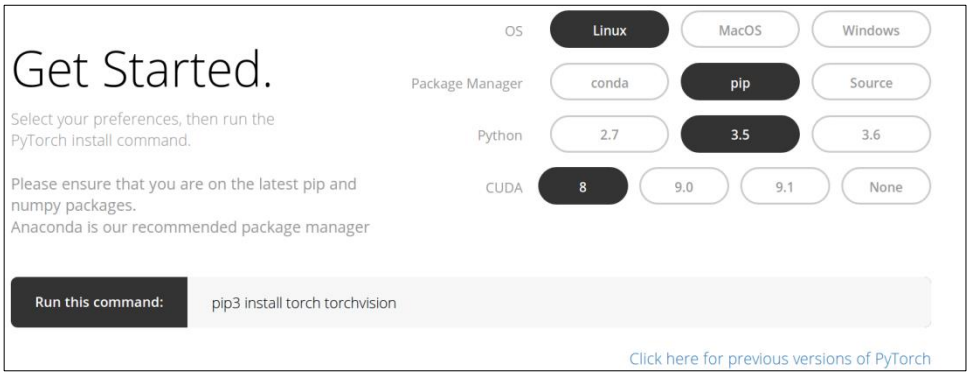


Figure 5.2.4.1: Install torch

We have to install the LMDB and we can use the LMDB weight that we done in Torch. After that, we have to install the warp-ctc and shown in below:

```
git clone https://github.com/SeanNaren/warp-ctc.git

cd warp-ctc

mkdir build; cd build

cmake ..

make
```

```
cd pytorch_binding

python setup.py install
```

Figure 5.2.4.2: Install Warp-Ctc In Pytorch

Before we start to train the LMDB weight using the python script, we have to run the demo.py to check it can train already. We have to make some change on the python script of the crnn_main.py to train our own project.

```
parser = argparse.ArgumentParser()
parser.add_argument('--trainroot', required=True, help='path to dataset')
parser.add_argument('--valroot', required=True, help='path to dataset')
parser.add_argument('--workers', type=int, help='number of data loading workers', default=2)
parser.add_argument('--batchSize', type=int, default=64, help='input batch size')
parser.add_argument('--imgH', type=int, default=32, help='the height of the input image to network')
parser.add_argument('--imgW', type=int, default=100, help='the width of the input image to network')
parser.add_argument('--nh', type=int, default=256, help='size of the lstm hidden state')
parser.add_argument('--niter', type=int, default=25, help='number of epochs to train for')
parser.add_argument('--lr', type=float, default=0.01, help='learning rate for Critic, default=0.00005')
parser.add_argument('--beta1', type=float, default=0.5, help='beta1 for adam. default=0.5')
parser.add_argument('--cuda', action='store_true', help='enables cuda')
parser.add_argument('--ngpu', type=int, default=1, help='number of GPUs to use')
parser.add_argument('--crnn', default='/home/ubuntu/crnn.pytorch-master/data/crnn.pth', help="path to crnn (to continue training)")
parser.add_argument('--alphabet', type=str, default='0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ')
parser.add_argument('--experiment', default=None, help='Where to store samples and models')
parser.add_argument('--displayInterval', type=int, default=100, help='Interval to be displayed')
parser.add_argument('--n_test_disp', type=int, default=50, help='Number of samples to display when test')
parser.add_argument('--valInterval', type=int, default=100, help='Interval to be displayed')
parser.add_argument('--saveInterval', type=int, default=500, help='Interval to be displayed')
parser.add_argument('--adam', action='store_true', help='Whether to use adam (default is rmsprop)')
parser.add_argument('--adadelta', action='store_true', help='Whether to use adadelta (default is rmsprop)')
parser.add_argument('--keep_ratio', action='store_true', help='whether to keep ratio for image resize')
parser.add_argument('--random_sample', action='store_true', default=True, help='whether to sample the dataset with random sampler')
opt = parser.parse_args()
print(opt)
```

Figure 5.2.4.3: Add On The Destination Of CRNN Path

```
        preds = crnn(image)
        preds_size = Variable(torch.IntTensor([preds.size(0)] * batch_size))
        cost = criterion(preds, text, preds_size, length) / batch_size
        loss_avg.add(cost)

        _, preds = preds.max(2)
        #preds = preds.squeeze(2)
        preds = preds.transpose(1, 0).contiguous().view(-1)
        sim_preds = converter.decode(preds.data, preds_size.data, raw=False)
        for pred, target in zip(sim_preds, cpu_texts):
            if pred == target.lower():
                n_correct += 1

    raw_preds = converter.decode(preds.data, preds_size.data, raw=True)[:opt.n_test_disp]
    for raw_pred, pred, gt in zip(raw_preds, sim_preds, cpu_texts):
        print('%-20s => %-20s, gt: %-20s' % (raw_pred, pred, gt))

    accuracy = n_correct / float(max_iter * opt.batchSize)
    print('Test loss: %f, accuray: %f' % (loss_avg.val(), accuracy))
```

Figure 5.2.4.4: Change The Code “Preds = Preds.Squeeze(2)” Into Command

We have obtained the percentage accuracy of the text recognition, increase when we have used the correct optimizer which is adadelta. The percentage of accuracy will increase nearly 100%. We have tried to use other optimizer which is adam. The percentage accuracy of adam for first time is higher than the percentage accuracy of adadelta, but after that it is keeps on decrease until 0%.

crnn_main.py –trainroot /home/ubuntu/Train –valroot /home/ubuntu/Test –cuda –keep ratio –adadelta
code –trainroot /destination fot Train folder –valroot /destination for Test folder –enable gpu –keep the height and weight in same ratio –optimizer

Figure 5.2.4.5: The python script of training the LMDB weight

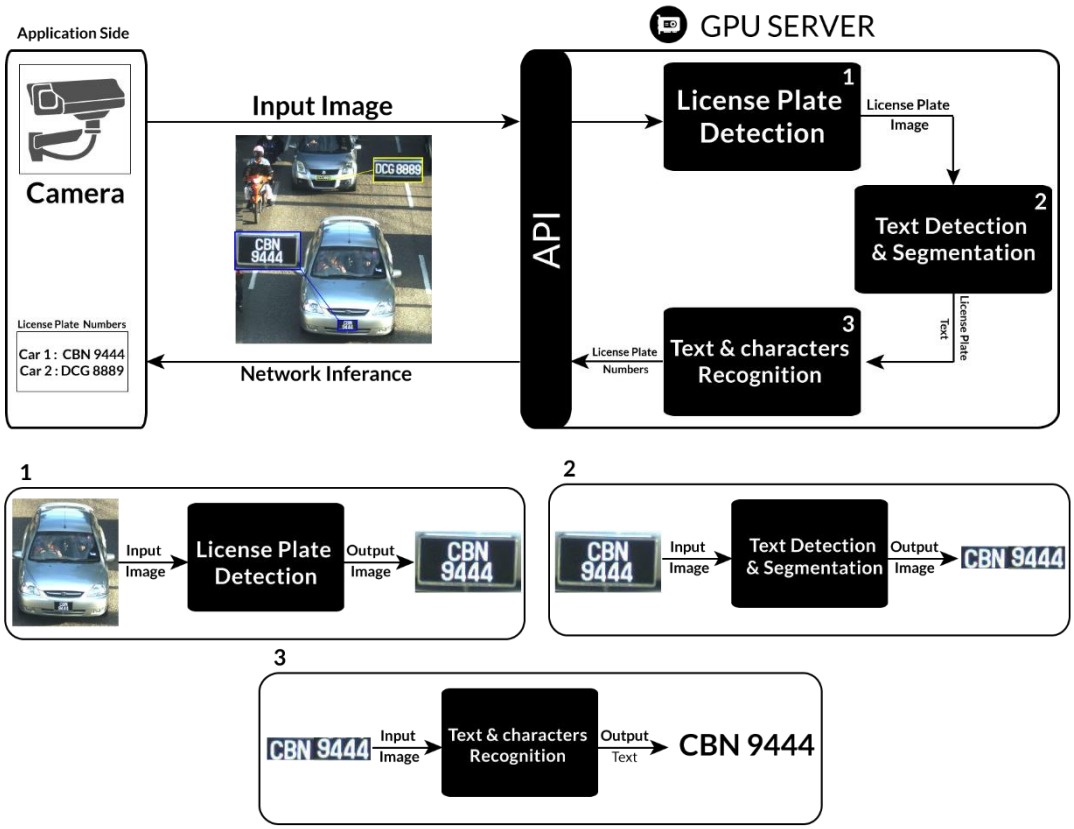
5.2.5 Result

After the whole training is done, our system is able to recognize the text and character on the license plate from the image directly. When we put the input image in our Deep Neural Network based License Plates Detection-Recognition System, the text of the license plate is shown in the output.

Table 5.2.5.1: The Result Of Text Recognition

INPUT	OUTPUT
	

5.3 Cascaded Network and Desktop Application



By designing a python program interface using PyQt5 library and cascading the three mentioned stages with their networks in above figure and interfacing a stream by IP camera with a top up functionality by connecting to our server we can obtain a full system that can run purely in different platform and operating system.

5.4 Building App

5.4.1 Design Interface Of Application Using Ionic 3

In this project, we are using Ionic 3 to build our system app. Ionic is a complete open-source SDK for hybrid mobile app development. The original version of Ionic was created by Max Lynch, Ben Sperry, and Adam Bradley of Drifty Co. and released in 2013 [4]. Ionic is built on top of AngularJS and Apache Cordova. A list of tools and services are provided in Ionic for developing hybrid mobile app using Web technologies like CSS, HTML5, and Sass. Apps can be built with these Web technologies and then distributed through native app stores to be installed on devices by Cordova [5].

Firstly, Node.js was installed. After setting up Node, command prompt was opened and run as following to install the Ionic and Cordova.

```
$ npm install -g ionic cordova
```

After installation of ionic and all the dependencies, command prompt was run as code below to start build our app.

```
$ ionic start justgo tabs
```

To run the application, cd into the folder and the *ionic serve* command was run to enable the app is viewable in a browser.

```
$ cd justgo

$ ionic serve -l
```

In Ionic, components are the basic building for an app which also known as User Interface (UI) elements. Components provided are made up of HTML, CSS, and JavaScript. Each of the Ionic components adapt to the platforms on which the app is running. Components allow a developer able to construct an interface for an app quickly. There are number of component in Ionic, however we only used some of it in our app.

Table 5.4.1.1: Component Used

COMPONENT	DESCRIPTION	CODING
Input	To collect and handle user input.	<**ion-item**> <**ion-label** fixed>Username</**ion-label**> <**ion-input** type="text" value=""></**ion-input**> </**ion-item**>
Button	To Interact and navigate through an app	<**button** ion-button block>LOGIN </**button**>
Icon	To insert icon available inside Ionicons docs.	<**ion-icon** name="CASH"></**ion-icon**>
Navigation	To navigate from one page to another simply push or pop a new page onto the stack:	@Component({ template:'<**ion-nav** [root]="rootPage"></ion-nav>' }) class MyApp { *// First page to push onto the stack* **rootPage = LoginPage;** }
Tabs	To create a multi-tabbed interface in Tab Bar that can be tabbed through.	**<ion-tabs>** <ion-tab tabIcon="heart" [root]="tab1">** </ion-tab>** <ion-tab tabIcon="star" [root]="tab2">** </ion-tab>** **</ion-tabs>**
Grid	Is composed of three units: *grid, rows and columns.*	<**ion-grid**> <**ion-row**> <**ion-col** col-12>This column will take 12 columns</**ion-col**> </**ion-row**> </**ion-grid**>

Img	To able the app to dedicate the resource to view the image.	<**ion-img** width="80" height="80" src="..."> </**ion-img**>
Card	To organize and display important contents. The header, lists, buttons and other card components can be easily combined with image cards.	<**ion-card**> <ion-card-content> <ion-card-title> ... </ion-card-title> ... </ion-card-content> </**ion-card**>

(A) (B)

Figure 5.4.1.2: (A) HTML code in Login Page; (B) Interface of Login Page

The components used are ion input, ion button and ion-row and ion img. The "JUST GO" logo was created by using Adobe Illustrator. In login page which required the user enters the username or email and password. The 'LOGIN' button click was added in order to navigate to Home Page once the username and password entered are correct.

(A) (B)

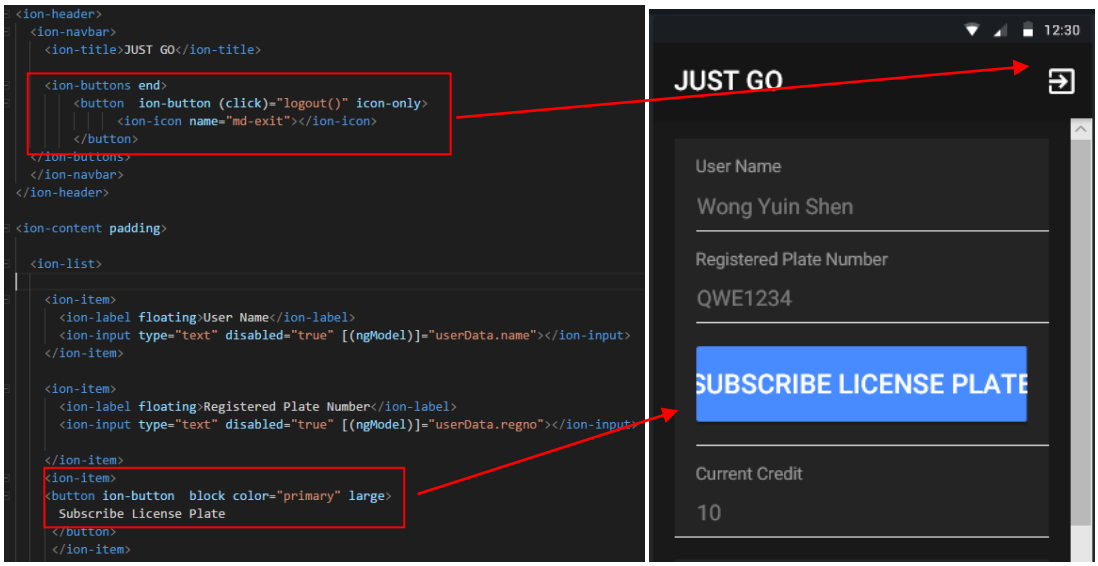


Figure 5.4.1.3: (A) HTML code in Home Page; (B) Interface of Home Page

The components used are ion input, ion button and ion icon. Figure shows the final interface of home page which will auto displayed user data (User Name, Registered Plate Number, and Current Credit) after user login. A button of "Subscribe License Plate" was added to let the users subscribe different license plate under an account.

(A) (B)

Figure 5.4.1.4: (A) HTML code in Tabs Page; (B) Tab Page Icon

The components used are ion tab and ion icon. Three tabs were created and named as "Home", "Top Up", and "About Us". Suitable icons which are "car", "cash" and "contacts" are searched in Ionicons docs and inserted to each tab to make the Tab Page more attractive.

(A) (B)

Figure 5.4.1.7: (A) HTML code in AboutUs Page; (B) AboutUs

The components used are ion grid and img. The grid system consists of a 12 columns, and each ion-col can be sized by setting the col-<width> attribute. Ionic also provides a set of utility attributes that can be used on any element in order to adjust or modify the text. Text alignment of centre was used to make sure the images inserted are located in the centre.

5.4.2 Using Php And Mysql To Create Application Database

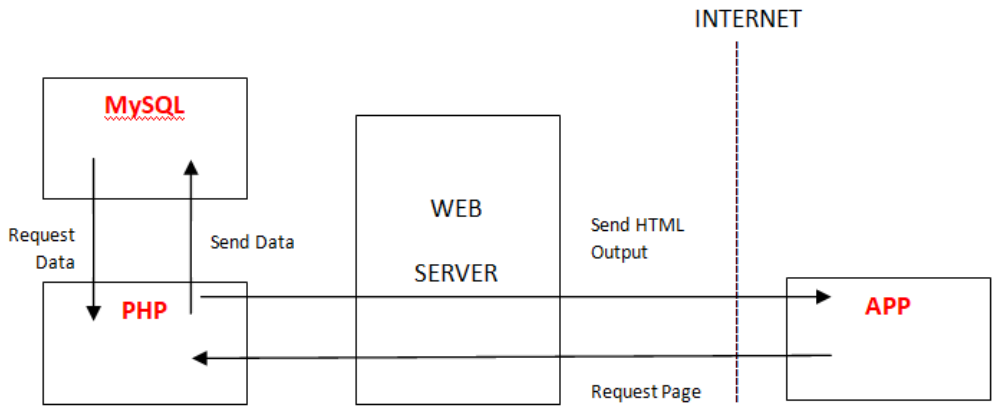


Figure 5.4.2.1: Block Diagram of PHP

PHP (Hypertext Preprocessor) is the most popular free, open source and server side programming language that handles HTTP requests for web development. It is fully featured and provides useful functions. PHP basically allows developers to Create, Read, Update, Destroy (CRUD) with Databases. MySQL is a Relational

Database Management System (RDBMS) that uses Structured Query Language (SQL). [4]

HTTP requests are:

GET - which allows read data from the data stored and it is done with a database (MySQL) query.

POST - which post user data from HTML form.PHP allows to access this global variable $_POST['example'] and store it in the data.

In PHP script, must connect to a SQL database before the database is able to be used. HTML pages usually will call PHP scripts through a user input form with the data submit by user to a database and in the condition of it must be on the same server and domain. [4]

First, a new provider is created by using the following command:

```
$ ionic generate providers
```

Providers are used to create a self-contained service that provides certain functionality to an application. The role of a provider might include things like fetching data from a server, performing operations on data, sharing data, providing a set of complicated mathematical operations, and so on.

Next, import the provider into any component that uses it and inject the provider in the constructor of any component that is using it.

```
import { Component } from '@angular/core';
import {  NavController, NavParams } from 'ionic-angular';
import { TabsPage } from '../tabs/tabs';
import { AuthService } from '../../providers/auth-service/auth-service';

/**
 * Generated class for the LoginPage page.
 *
 * See https://ionicframework.com/docs/components/#navigation for more info on
 * Ionic pages and navigation.
 */


@Component({
  selector: 'page-login',
  templateUrl: 'login.html',
})
export class LoginPage {
  responseData : any;
  userData = {"username": "", "password": ""};
  constructor(public navCtrl: NavController, public navParams: NavParams, public authService:AuthService) {
  }
```

Figure 5.4.2.2: HTML Code to import provider

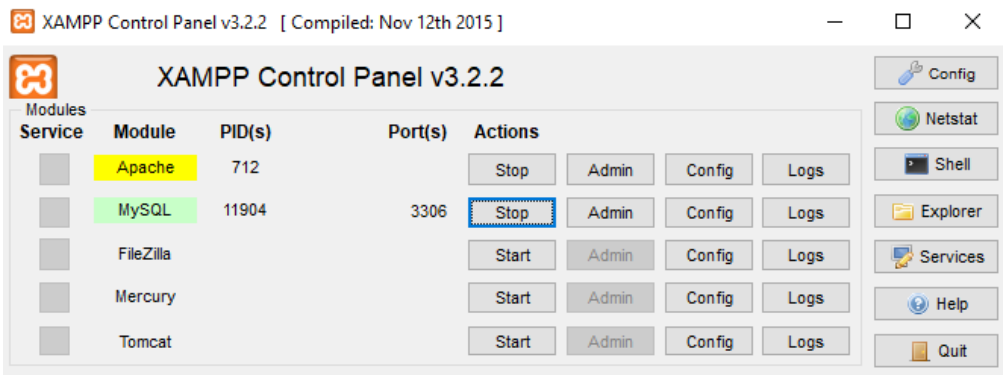


Figure 5.4.2.3: Interface of XAMPP Control Panel

After that, download XAMPP. The XAMPP platform is easy to install and configure and it is implemented on a variety of operating systems such as Linux, Windows, Mac OS X. The technologies have been used which are MySQL, PHP, HTML, CSS. Module Apache and MYSQL are chosen. After XAMPP being setup and PHP Slim Restful file is completely download, a new project named “just go” was created as below. Then, create users table in the new database by the above content in SQL query.

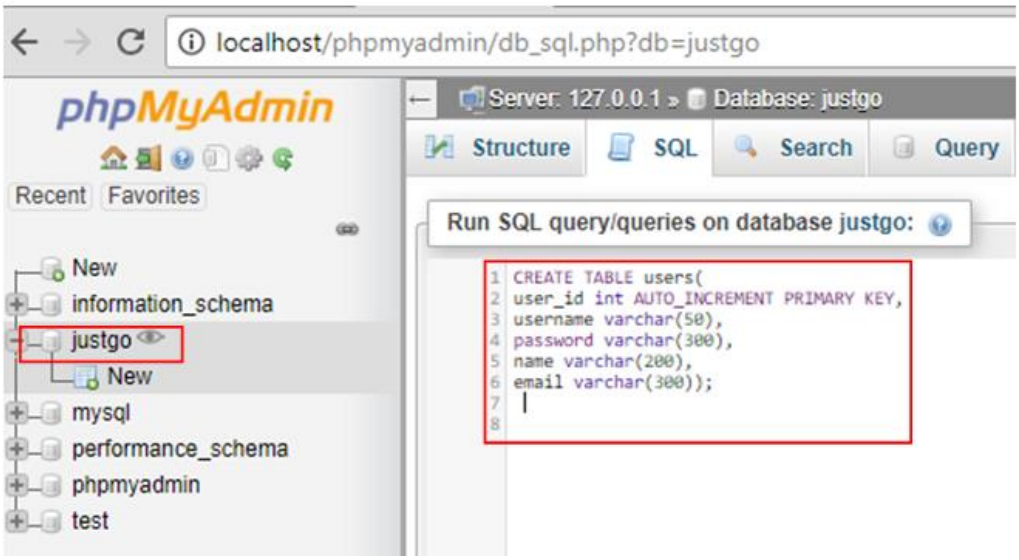


Figure 5.4.2.4: Code For Create Users Table

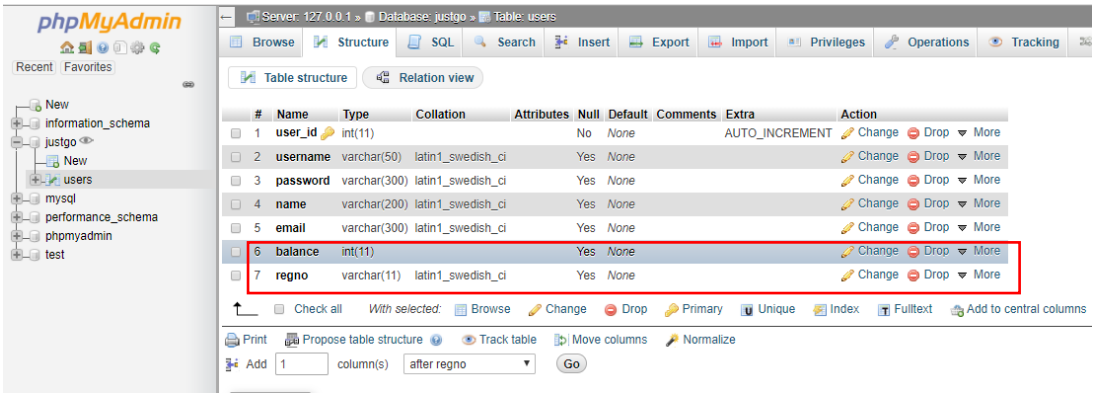


Figure 5.4.2.5: Interface of PHP

Figure shows the ‘users’ table is created and two columns balance (current credit) and regno (registerred plate number) are added in the table.

Next, install Postman. It makes the API document easier and faster through design, testing and full production as it can send requests individually or send all the requests in the collection. The responses and results are viewable.

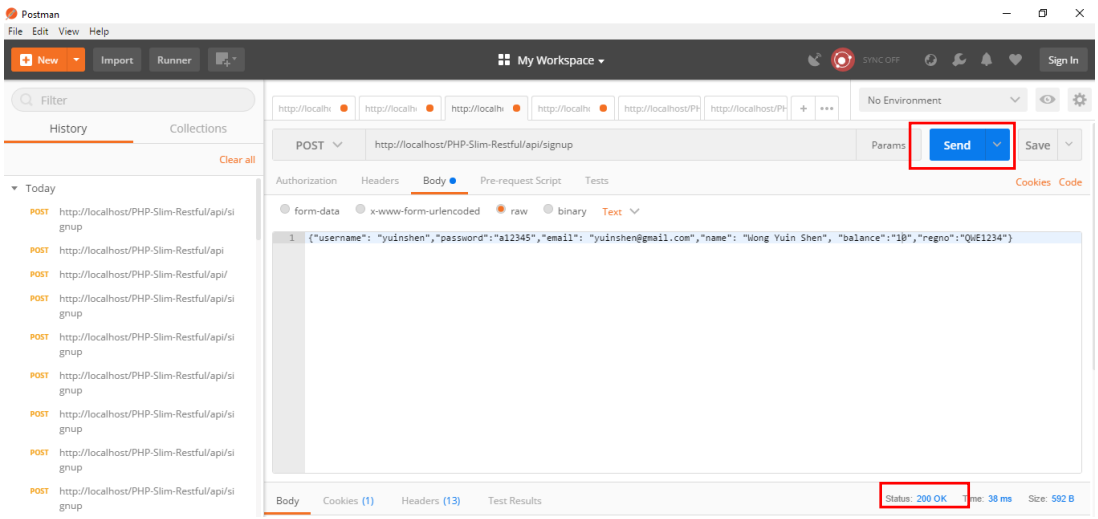


Figure 5.4.2.6: User Data Send To Database

Figure shows the users data which included username, password, email, name, balance and registered plate number has been sent to the database.

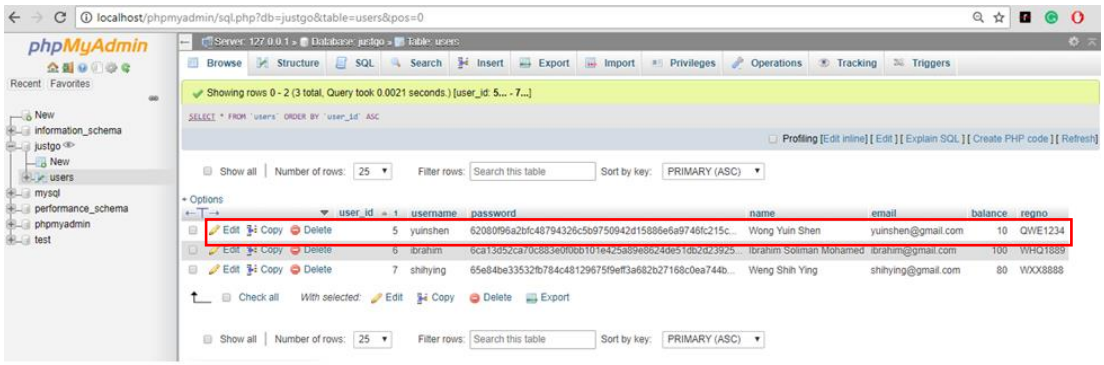


Figure 5.4.2.7: User Data Stored in Database

5.4.3 Display User Data From Database

The code is write in login.ts and index.php as following to receive the values and stores them in the database.

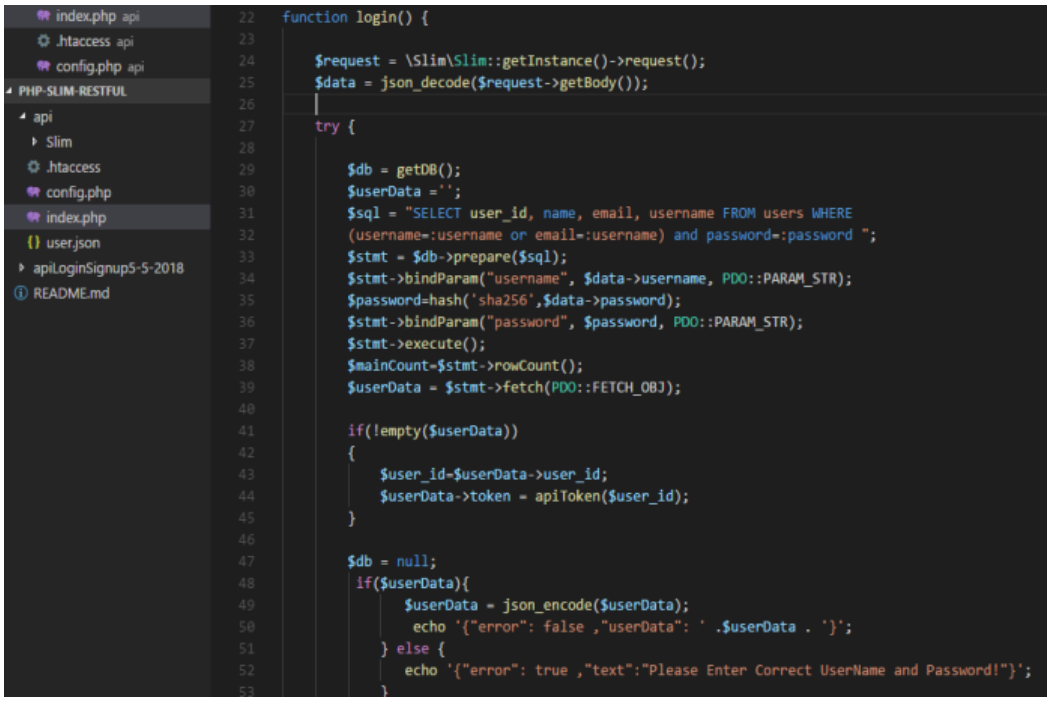


Figure 5.4.3.1: Coding In Index.Php

```
export class LoginPage {
  responseData : any;
  userData = {"username": "", "password": ""};
  constructor(public navCtrl: NavController, public navParams: NavParams, public authService:AuthService) {
  }

  ionViewDidLoad() {
    console.log('ionViewDidLoad LoginPage');
  }
  doLogin(){

    console.log(this.userData);
    this.authService.postData(this.userData,'login').then((result) => {
      this.responseData = result;
      console.log(this.responseData);
      if (!this.responseData.error){
        localStorage.setItem('userData', JSON.stringify(this.responseData.userData))
        this.navCtrl.setRoot(TabsPage);
      }else {
        alert(this.responseData.text)
      }
    }, (err) => {
      //Connection failed message
    });
```

Figure 5.4.3.2: Coding In Login Typescript

Figure 5.4.3.3: Alert Message

Figure above shows if the user enters with the wrong username or password, an alert message will be pop out should be displayed guiding them to do it correctly.

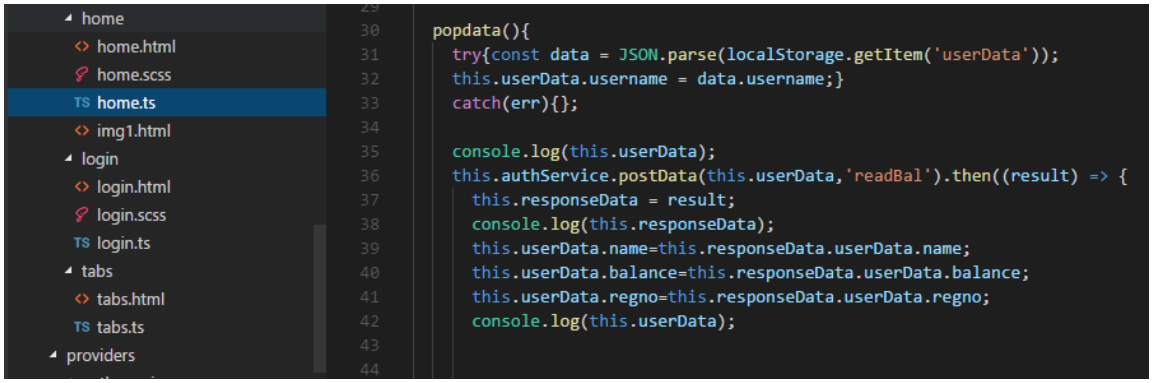


Figure 5.4.3.4: Coding In Home Typescript

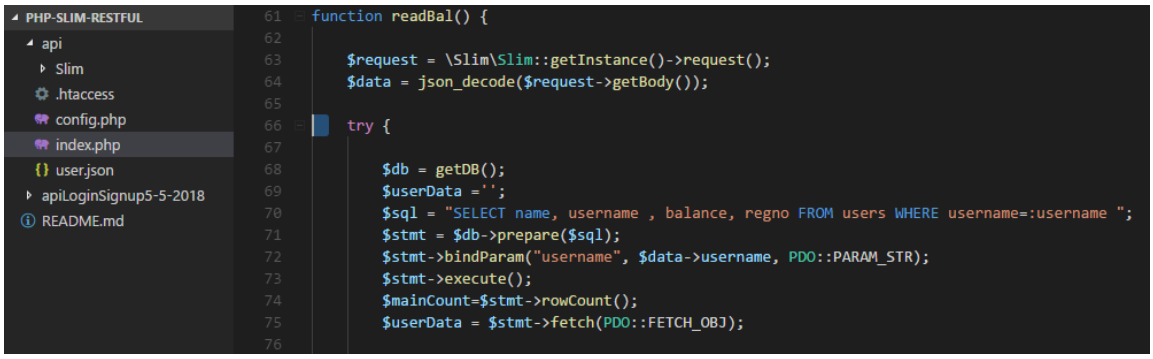


Figure 5.4.3.5: Function Of 'Readbal'

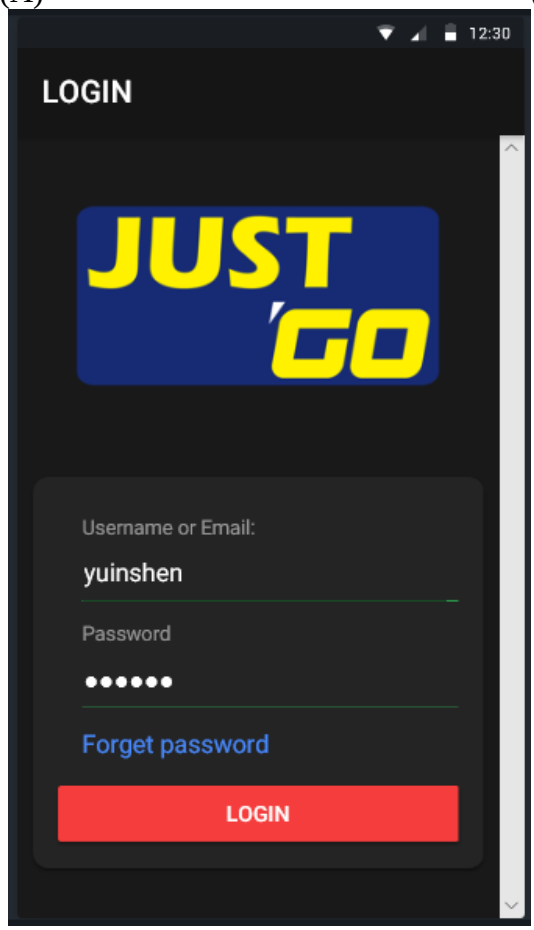
After user login with the correct username and password, the function "readBal" is called and the data required can be obtained from database which included name, balance, and registered plate number and balance will be displayed in homepage of the application.

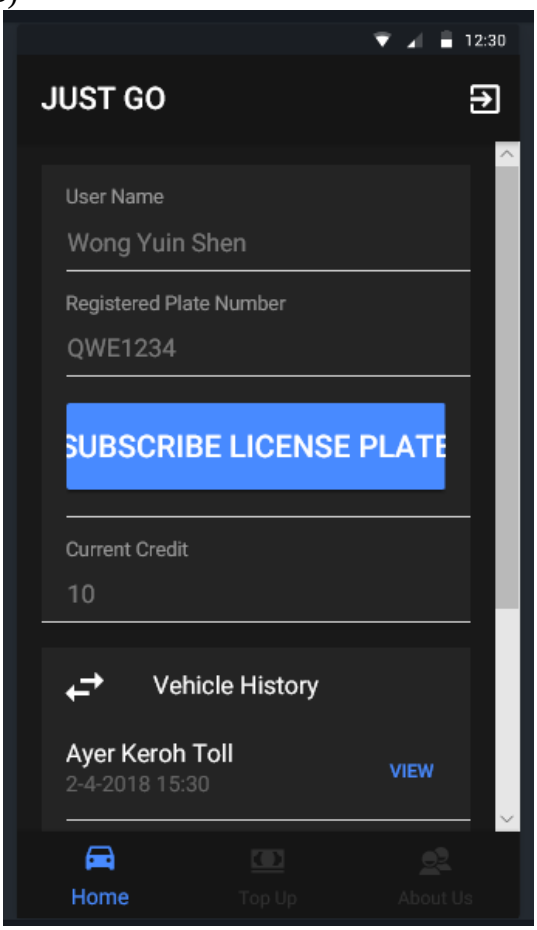
5.4.4 Final Interface of the "Just Go" Application

After designing the user-interface and creating the database of application, the final build of app was ready to be deployed by running following command:

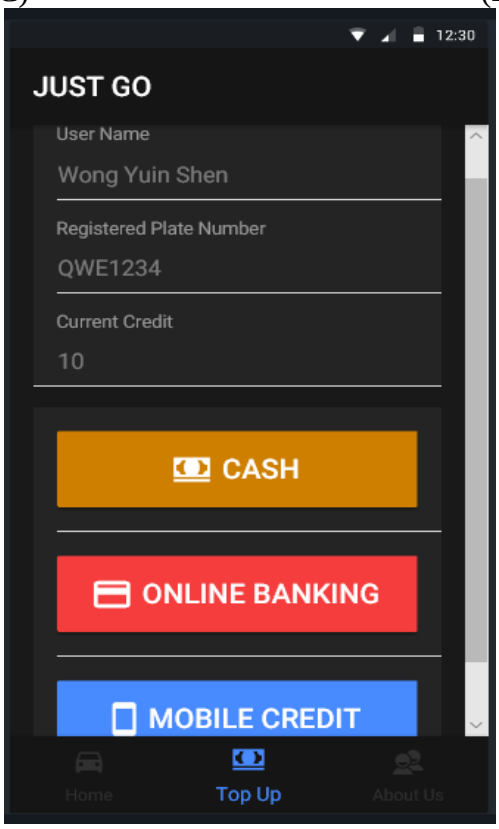
```
$ ionic cordova run android
```

(A) (B)

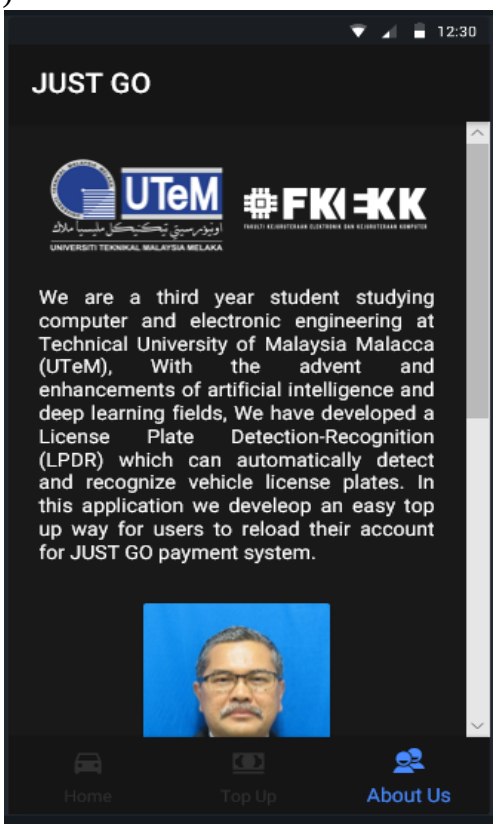




(C)



(D)



(E)

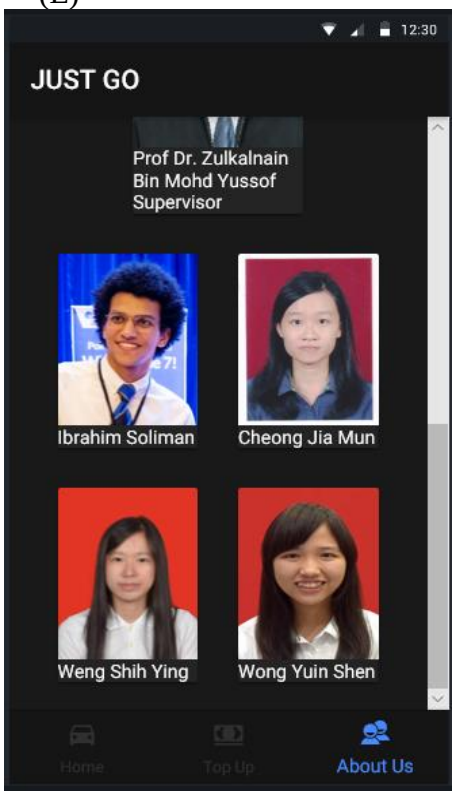


Figure 5.4.4.1: (A) LoginPage; (B) HomePage (C) TopUpPage; (D,E) AboutUsPage

The final build of the application on a mobile platform can be viewed as above which auto update the data from time to time and display to the user in order to let them check the current situation of the service being used.

CHAPTER 6

ENVIRONMENTAL AND SUSTAINABILITY

Environmental and sustainability are essential in designing or producing a new project. According to The Brundtland Commission, sustainability means meeting our own needs without compromising the ability of future generations to meet their own needs. It means how natural systems function, remain diverse and produce everything the ecologyneeds to remain in balance. In today's globalization era, it means an acknowledgement that human civilization takes resources to sustain theirneeds, protect natural environment, human and ecological health while driving innovation and not compromising our way of life. The consumption of the natural resources increase rapidly as the needs of the human beings increases. Sustainability is not just about environmental, it also requires aspect from social and economy. There are three pillars of sustainability which are environment, economy and society. Environmental sustainability means the ecological integrity is maintained and all the environmental systems on earth are remain in balance. Social sustainability signifies the quality of the nature-society. Economic sustainability is the ability of an economy to support a defined level of economic production indefinitely. To develop our project, we had considered it sustainability towards environment, social and economy.

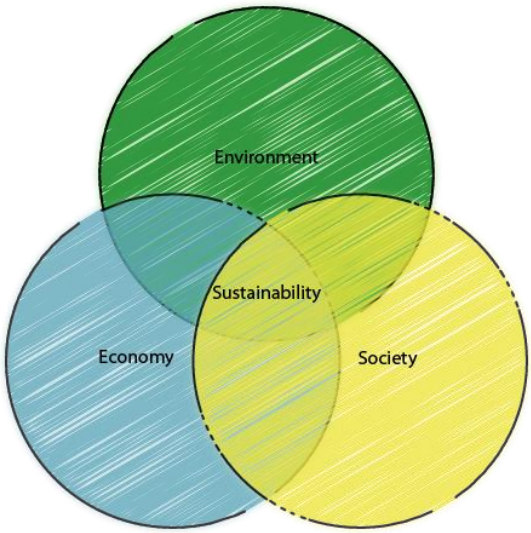


Figure 6.1: Three pillars of sustainability

6.1 Social

Social sustainability is the ability of a social systemto function at a defined level of social well-being indefinitely. To design or produce any new project, social sustainability should be considered so that the needs of every human beingscan be meet and able to be preserved over a long period of time.Our project, License Plate Detection-Recognition (LPDR) can be implemented and used with different application and in different environment like parking and vehicle payment systems.

LPDR system provides convenient to the drivers to find a parking slot in a car park in shopping mall, hospital, hotel, airport and so on. Smart Parking System which able to identify parking space and assign a parking lot for entering cars. The licence plate of the vehicle had been captured once they enter the parking place so if there are any crime happens in that parking place, our system can provide evidence in the investigation of crime. According to New Strait Times, parking places are commonly most at risk from theft of or from vehicles. Hence by using our system, the safety level of the car park can be improved.The recruitment of security guards can be replacedby using our system. There are many issues happens when the security guards fall asleep in the mid of the work especially at night. This could lead to the unsecure life of public.

In Malaysia, most of the people used Touch n Go to pay the tolls which takes approximately 6 seconds whereas SmartTAG payment takes around 3 seconds. Since

our system does not need the barrier gate so the drivers no need to slow down their vehicles or stop by the barrier gate to pay the toll fees. This system can help to create a multi lane free flow. Motorcyclists can also pass through the toll plazas with a seamless experience. The traffic congestion problem at the toll plazas can be solved. This can reduce stress of the drivers and drivers can enjoy their journey as they no longer need to stuck at the traffic jam while passing through the tolls.

6.2 Economy

Looking through the economic development aspect, our project which mainly software that enables computer systems to read automatically the license plate of vehicles from picture captured has the potential to be developed in the market. The authority who will first access our system for sure is the government. For example, PLUS Malaysia Berhad. This is because by using our system, the number of toll plazas can be reduced. The cost of building for toll plazas can be cut completely. Also, the maintenance fees can also be reduced as our system does not need maintenance fees for the barrier gate system, security camera CCTV, RFID reader and the car loop detector. Our system only needs two cameras which are front camera and rear camera to capture the license plate of the vehicle from different angle. This can save a huge amount of money. If there is any malfunction of the camera, it can be changed easily.Moreover, Malaysia can reduce its expenses by using the system that is made locally andthe economy can be sustained as the money only regulates inside the country.

Following is the local authorities for example shopping malls, airports and hospitals. Our system not only can be use at toll payment system, it can also be use for car parking system and it only needs camera and barrier gate.

Next, our project reduces the use of man power. Through this project which built a self-develop system, we believed that it can be improved and commercialized to be used in Malaysia and other country since this system used the knowledge and technologies that being applied in other country as well. By using our system, we no longer needtoll plazas. There are total 94 toll plazas are there on the PLUS highways.Hence, the cost use to build toll plazas can be cut completely. Moreover, a

lump sum of money can be saved up when we do not need to hire people to collect the toll fees at the toll plazas.

Although the cost for our project is low but it can bring a lot of benefits to the public. Hence, this project is economic sustainable since the cost of the material is much lesser than any other system.

6.3 Environmental

Environmental sustainability is the most important pillar among the three pillars. License Plate Detection-Recognition (LPDR) systemis environment friendly since it does not give any destroy to the natural resources or causes any pollution to the environment. The consumption of natural resources and energy is low.

Nowadays, most of the office buildings and shopping mall had built underground parking and multilevel parking to overcome the number of cars which is increasing rapidly. However, drivers are still difficult to find an available parking slot to park their car. The process of looking for a parking lot is time consuming, confusing and wasting fuel as well. Our system can identify parking space and assign a parking lot for entering cars. This can help to save time and fuel.

In our project, we just need to place the cameras on the tolls. Therefore, our system canhelp to lower energy consumptions for the environments since we do not need toll plazas and the barrier gate to pass through the toll.It is also one of the ways to reduce the energy costs instead of paid much electricity bill.

CHAPTER 7

HEALTH AND SAFETY

Our system is very useful for everyone. This is because Deep Neural Network based License Plates Detection Recognition (LPDR) system is not only can be use in toll vehicle payment, it can also be use in parking system in hospitals, shopping mall and hotels. With our system, it can help to save time on vehicle payment with license plate detection and recognition and at the same time reduce congestion in toll and car parking. On the same time, the system also can develop application for tracking and security solutions. For example, transportation can detect and trace easily and smart car tracker.

The LPDR system that designed by our group is easier to install and setup. The system just needs two security cameras for entrance and exit. There only need to do bits wiring to connect the electricity with camera at the entrance for the parking. The best about our system is it can use for 24 hours and required low power consumption. In the more detailed explanation, when the camera in the system is detected the car plate, it will activate all other parts to do the works together. If no car plate detected, the camera will in the standby mode. For the meantime it will save the energy and has lower power consumption.

With our Deep Neural Network based License Plates Detection Recognition (LPDR) system, the management of the parking area will be more systematic. First, the officer will register all the car plate of the users that live in this apartment. Once they have registered their car plate, they do not need to register anymore. The license plate of the vehicle had been captured once they enter the parking place. Then, the system will identify parking space and assign a parking lot for entering cars. If any crime happens, they can check the visitor's car park data in a shorter time. Also, our system can provide evidence in the investigation of crime. The vehicle's safety will be guaranteed when using this car plate recognition system. At the same time, the

problem of vehicle being steal may be reduced. Hence by using our system, the safety level of the car park can be improved as the crime rate can be reduced. So, this car plate detection and recognition is easier for the office to manage the parking area. With our invention, the officers just need to check on the visitor's car park by live stream from the camera and reduce the manpower and waiting time.

Vehicle tracking and security solution will be provided with license plate detection and recognition system. Nowadays, many housing areas have increase their residential safety level by using gated and guarded system. However, this cause a time-wasting problem for residential living in that area as they need to queue to scan their tag for entrance. Even though there is the gated and guarded system in housing area, the crime such as house break-ins and snatch thefts are still happening. One of the reasons of crime has not been solved because of the poor management of security system. Hence, the implementation of our system in the housing area can provide a systematic data record and reduce crimes rate. Residents can enter their housing area easier and safely without stopping to scan their tag. The security guard can play their roles to pay more attention on visitor entering residential area. Industry transportation system also can be track and trace using the end to end engine based system.

Accidents often occur at toll plaza. This is because of some driver do not tolerate with each other for example they cut queue to fasten their toll payment and causing two vehicles crashed in a single lane. According to Star Online, a reported accident of "Two killed as car crashes into pillar at unused BatuTiga toll plaza", the car were went out of control and rammed into the pillar although it is an unused toll plaza. This situation has showed that toll plaza is one of the major factors leading to vehicle accidents. With our system, a multi free flow lane of vehicle payment can be created. The drivers no need to pass through barrier gate to make payment and they also do not need to slow down.

Most of us do not know much about the **impacts of traffic congestion**, yet it's one of the factors that play a massive part in our daily lives, sending stress levels through the roof as we try to get around. Stress can cause a variety of symptoms and **affect our overall well-being** for example anxiety, depression, headaches and insomnia. Anything that increases stress while driving should be avoided. If our system can be implemented in future on many aspects, we will no longer struggle in traffic jams. Everything in life can be better.

CHAPTER 8

CONCLUSION

In conclusion, the objective of this project has been achieved by detecting and recognizing the license plate automatically with Deep Learning. We are able to detect the license plate from single car or multiple cars by using Yolo v3. Besides, we are also able to rotate the label box depend on the license plate and change the double row license plate to single row license plate by using EAST. After that, we able to detect and recognize the license plate image to text by using Pytorch. Then, we combine all together on a prototype called Deep Neural Network based License Detection-Recognition (LPDR) System by using Qtand create an app that called "Just Go Application" by using Ionic 3. The performance and accuracy of the license plate detection and recognition are evaluated by the prototype which has the better performance. All of these are the best helper to reduce the traffic jam at parking places and toll.

REFERENCE

[1] John Lim, "6 Ways You Could Cause A Traffic Jam Without Even Trying", 2015. [Online]. Available : http://says.com/my/fun/traffic-jams[Accessed: 10-Mar-2018]

[2] "Car Parking and Traffic Congestion", 2014. [Online]. Available:http://www.parking-net.com/parking-news/skyline-parking-ag/trafficcongestion/. [Accessed: 10-Mar-2018]

[3] Jenilyn, "Malaysian Tolls Are Contributing To The Highway Traffic Jams", 2015. [Online]. Available: https://www.carlist.my/news/malaysian-tolls-are-contributing-highway-traffic-jams/16882/. [Accessed: 11-Apr-2018]

[4] "Touch 'n Go Sdn Bhd (TNGSB) Company Background". [Online].Available:http://www.touchngo.com.my/CMS/Corporate/About-Us/Company-Background/. [Accessed: 15-Apr-2018]

[5] J.Redmon, S.Divvala, R.Girshick , A.Farhadi. You Only Look Once: Unified, Real-Time Object Detection. arXiv, 2015

[6] J. Redmon, A. Farhadi. *YOLO9000: Better, Faster, Stronger*. arXiv, 2016.

[7] M. Liao, B. Shi and X. Bai,"TextBoxes++: A Single-Shot Oriented Scene Text Detector", IEEE, arXiv:1801.02765v2, 2017.

[8] T. Wang, D. J. Wu, A. Coates, and A. Y. Ng, “End-to-end text recognition with convolutional neural networks” 2012.

[9] Zachary C. Lipton, John Berkowitz, Charles Elkan, “A Critical Review of Recurrent Neural Networks for Sequence Learning”, 2015.

[10] Graves, S. Fern´andez, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in Proc. ICML, pp. 369–376, 2006.

[11] Shalin A. Chopra, Amit A. Ghadge, Onkar A. Padwal, Karan S. Punjabi, Prof. Gandhali S. Gurjar, “Optical Character Recognition”, International Journal of Advanced Research in Computer and Communication Engineering Vol. 3, Issue 1, January 2014.

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed, “SSD: single shot multibox detector,” in Proc. ECCV, 2016.

[13] K. Simonyan and A. Zisserman, “Very deep convolutional networks for l arge-scale image recognition,” CoRR, vol. abs/1409.1556, 2014.

[14] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in Proc. NIPS, 2015.

[15] J. Redmon, S. K.Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in Proc. CVPR, 2016.

[16] Shahab, F. Shafait, and A. Dengel, “Robust reading competition challenge 2: Reading text in scene images”, In Proc. of ICDAR, 2011.

[17] D. Karatzas, L. Gomez-Bigorda, A. Nicolaou, S. Ghosh, A. Bagdanov, M. Iwamura, J. Matas, L. Neumann, V. R. Chandrasekhar, S. Lu, F. Shafait, S. Uchida, and E. Valveny. ICDAR 2015 competition on robust reading. In Proc. of ICDAR, 2015.

[18] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie, "Coco-text: Dataset and benchmark for text detection and recognition in natural images" In arXiv, 2016.

[19] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu, " Detecting texts of arbitrary orientations in natural images". In Proc. of CVPR, 2012.

[20] B.Epshtein,E.Ofek,andY.Wexler, "Detecting text in natural scenes with stroke width transform". In Proc. of CVPR,2010

[21] L. Neumann and J. Matas. "A method for text localization and recognition in real-world images". In Proc. of ACCV, 2010.

[22] L. Neumann and J. Matas. "Real-time scene text localization and recognition" In Proc. of CVPR, 2012.

[23] Z. Zhang, W. Shen, C. Yao, and X. Bai. Symmetry-based text line detection in natural scenes. In Proc.of CVPR ,2015.

[24] M. Busta, L. Neumann, and J. Matas. "Fastext: Efficient unconstrained scene text detector". In Proc. of ICCV, 2015.

[25] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao. Detecting text in natural image with connectionist text proposal network. In European Conference on Computer Vision, pages 56–72. Springer, 2016.

[26] W.Huang, Y.Qiao and X.Tang, “Robust scene text detection with convolution neural network induced mser trees” InProc. of ECCV, 2014.

[27] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “ Reading Text in the Wild with Convolutional Neural Networks”, *International Journal of Computer Vision*, 116(1):1– 20, jan 2016.

[28] Z. Zhang, C. Zhang, W. Shen, C. Yao, W. Liu, and X. Bai. Multi-oriented text detection with fully convolutional networks. In Proc. of CVPR, 2015.

[29] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation”, In Proc. of CVPR, 2015.

[30] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, “Reading Text in the Wild with Convolutional Neural Networks”, *International Journal of Computer Vision*, 116(1), 2016.

[31] Baoguang Shi, Xiang Bai and Cong Yao, “An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition”, 2015.

[32] T.K. Cheong, Y.S. Chong and Y.H Tay, “Segmentation-free Vehicle License Plate Recognition suing ConvNet-RNN”, 2017.

[33] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang Megvii Technology Inc., Beijing, China “EAST: An

Efficient and Accurate Scene Text Detector" arXiv:1704.03155v2 [cs.CV] 10 Jul 2017

[34] K. Wac, S. Ickin and J. H. Hong, "Studying the Experience of Mobile Applications Used in Different Contexts of Daily Life", pp. 1, 2011.

[35] Md. Rashedul Islam, "Mobile Application and Its Global Impact", *International Journal of Engineering & Technology* IJET-IJENS Vol:10 No:06.

[36] Mahesh Panhale, "Native vs Hybrid Mobile Application" *Beginning Hybrid Mobile Application Development*, 2016.

[37] W. Jason Gilmore, "Beginning PHP and MySQL",From Novice to Professional Fourth Edition, 2010

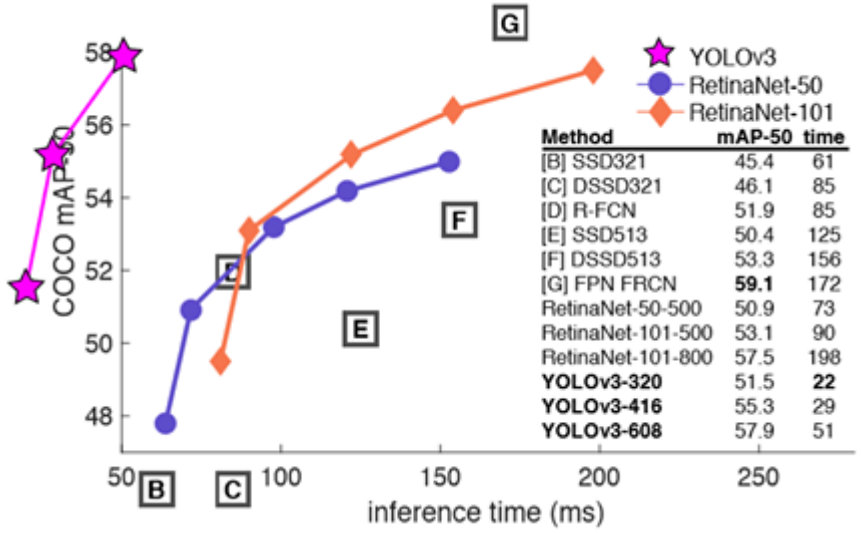
APPENDICES

Appendix A

Comparison Of Backbone

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Comparison Of YOLOV3 To Other Detectors



Appendix B

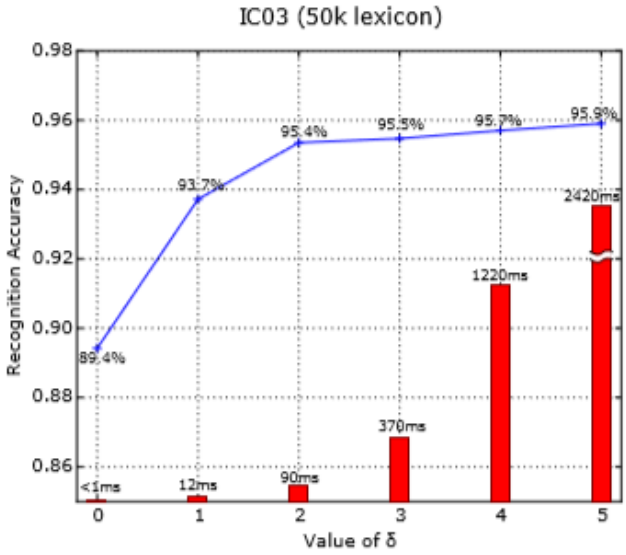
Comparison Among Various Methods

	E2E Train	Conv Ftrs	CharGT-Free	Unconstrained	Model Size
Wang *et al.* [34]	✗	✗	✗	✔	-
Mishra *et al.* [28]	✗	✗	✗	✗	-
Wang *et al.* [35]	✗	✔	✗	✔	-
Goel *et al.* [13]	✗	✗	✔	✗	-
Bissacco *et al.* [8]	✗	✗	✗	✔	-
Alsharif and Pineau [6]	✗	✔	✗	✔	-
Almazán *et al.* [5]	✗	✗	✔	✗	-
Yao *et al.* [36]	✗	✗	✗	✔	-
Rodrguez-Serrano *et al.* [30]	✗	✗	✔	✗	-
Jaderberg *et al.* [23]	✗	✔	✗	✔	-
Su and Lu [33]	✗	✗	✔	✔	-
Gordo [14]	✗	✗	✗	✗	-
Jaderberg *et al.* [22]	✔	✔	✔	✗	490M
Jaderberg *et al.* [21]	✔	✔	✔	✔	304M
CRNN	✔	✔	✔	✔	**8.3M**

Recognition Accuracies (%) On Four Datasets

	IIIT5k			**SVT**		**IC03**				**IC13**
	50	**1k**	**None**	**50**	**None**	**50**	**Full**	**50k**	**None**	**None**
ABBYY [34]	24.3	-	-	35.0	-	56.0	55.0	-	-	-
Wang *et al.* [34]	-	-	-	57.0	-	76.0	62.0	-	-	-
Mishra *et al.* [28]	64.1	57.5	-	73.2	-	81.8	67.8	-	-	-
Wang *et al.* [35]	-	-	-	70.0	-	90.0	84.0	-	-	-
Goel *et al.* [13]	-	-	-	77.3	-	89.7	-	-	-	-
Bissacco *et al.* [8]	-	-	-	90.4	78.0	-	-	-	-	87.6
Alsharif and Pineau [6]	-	-	-	74.3	-	93.1	88.6	85.1	-	-
Almazán *et al.* [5]	91.2	82.1	-	89.2	-	-	-	-	-	-
Yao *et al.* [36]	80.2	69.3	-	75.9	-	88.5	80.3	-	-	-
Rodrguez-Serrano *et al.* [30]	76.1	57.4	-	70.0	-	-	-	-	-	-
Jaderberg *et al.* [23]	-	-	-	86.1	-	96.2	91.5	-	-	-
Su and Lu [33]	-	-	-	83.0	-	92.0	82.0	-	-	-
Gordo [14]	93.3	86.6	-	91.8	-	-	-	-	-	-
Jaderberg *et al.* [22]	97.1	92.7	-	95.4	80.7*	**98.7**	**98.6**	93.3	**93.1***	**90.8***
Jaderberg *et al.* [21]	95.5	89.6	-	93.2	71.7	97.8	97.0	93.4	89.6	81.8
CRNN	**97.6**	**94.4**	**78.2**	**96.4**	**80.8**	**98.7**	97.6	**95.5**	89.4	86.7

Tested On IC03 Dataset With The 50k Lexicon



Comparison Of Pitch Recognition Accuracies Among CRNN And Two Commercial OMR Systems

	Clean	Synthesized	Real-World
Capella Scan [3]	51.9%/1.75	20.0%/2.31	43.5%/3.05
PhotoScore [4]	55.0%/2.34	28.0%/1.85	20.4%/3.00
CRNN	**74.6%/0.37**	**81.5%/0.30**	**84.0%/0.30**

APPENDIX C

4. Experiments

To compare the proposed algorithm with existing methods, we conducted qualitative and quantitative experiments on three public benchmarks: ICDAR2015, COCO-Text and MSRA-TD500.

[1]Consider the case that only a single text line appears the image. In such case, all geometries will be highly overlapped if the network is sufficiently powerful

Algorithm 1 Locality-Aware NMS

```
function NMSLocality(geometries)
    S ← ∅, p ← ∅
    for g ∈ geometries in row first order do
        if p ≠ ∅ ∧ ShouldMerge(g, p) then
            p ← WeightedMerge(g, p)
        else
            if p ≠ ∅ then
                S ← S ∪ {p}
            end if
            p ← g
        end if
    end for
    if p ≠ ∅ then
        S ← S ∪ {p}
    end if
    return StandardNMS(S)
end function
```

4.2. Base Networks

As except for COCO-Text, all text detection datasets are relatively small compared to the datasets for general object detection[21, 22], therefore if a single network is adopted for all the benchmarks, it may suffer from either over-fitting or under-fitting. We experimented with three different base networks, with different output geometries, on all the datasets to evaluate the proposed framework. These networks are summarized in Tab. 2.

Network	Description
PVANET [17]	small and fast model
PVANET2x [17]	PVANET with 2x number of channels
VGG16 [32]	commonly used model

Table 2. Base Models

Algorithm	Recall	Precision	F-score
Ours + PVANET2x RBOX MS*	**0.7833**	0.8327	**0.8072**
Ours + PVANET2x RBOX	0.7347	**0.8357**	**0.7820**
Ours + PVANET2x QUAD	0.7419	0.8018	0.7707
Ours + VGG16 RBOX	0.7275	0.8046	0.7641
Ours + PVANET RBOX	0.7135	0.8063	0.7571
Ours + PVANET QUAD	0.6856	0.8119	0.7434
Ours + VGG16 QUAD	0.6895	0.7987	0.7401
Yao *et al.* [41]	0.5869	0.7226	0.6477
Tian *et al.* [34]	0.5156	0.7422	0.6085
Zhang *et al.* [48]	0.4309	0.7081	0.5358
StradVision2 [15]	0.3674	0.7746	0.4984
StradVision1 [15]	0.4627	0.5339	0.4957
NJU [15]	0.3625	0.7044	0.4787
AJOU [20]	0.4694	0.4726	0.4710
Deep2Text-MO [45, 44]	0.3211	0.4959	0.3898
CNN MSER [15]	0.3442	0.3471	0.3457

Table 3. Results on ICDAR 2015 Challenge 4 Incidental Scene Text Localization task. MS means multi-scale testing.

Algorithm	Recall	Precision	F-score
Ours + VGG16	**0.324**	**0.5039**	**0.3945**
Ours + PVANET2x	0.340	0.406	0.3701
Ours + PVANET	0.302	0.3981	0.3424
Yao *et al.* [41]	0.271	0.4323	0.3331
Baselines from [36]			
A	0.233	0.8378	0.3648
B	0.107	0.8973	0.1914
C	0.047	0.1856	0.0747

Table 4. Results on COCO-Text.

Algorithm	Recall	Precision	F-score
Ours + PVANET2x	0.6743	**0.8728**	**0.7608**
Ours + PVANET	0.6713	0.8356	0.7445
Ours + VGG16	0.6160	0.8167	0.7023
Yao *et al.* [41]	**0.7531**	0.7651	0.7591
Zhang *et al.* [48]	0.67	0.83	0.74
Yin *et al.* [44]	0.63	0.81	0.71
Kang *et al.* [14]	0.62	0.71	0.66
Yin *et al.* [45]	0.61	0.71	0.66
TD-Mixture [40]	0.63	0.63	0.60
TD-ICDAR [40]	0.52	0.53	0.50
Epshtein *et al.* [5]	0.25	0.25	0.25

Table 5. Results on MSRA-TD500.

Approach	Res.	Device	T_1/T_2 (ms)	FPS
Ours + PVANET	720p	Titan X	58.1 / 1.5	16.8
Ours + PVANET2x	720p	Titan X	73.8 / 1.7	13.2
Ours + VGG16	720p	Titan X	150.9 / 2.4	6.52
Yao *et al.* [41]	480p	K40m	420 / 200	1.61
Tian *et al.* [34]	ss-600*	GPU	130 / 10	7.14
Zhang *et al.* [48]*	MS*	Titan X	2100 / N/A	0.476

Table 6. Overall time consumption compared on different methods. T_1 is the network prediction time, and T_2 accounts for the time used on post-processing. For Tian *et al.* [34], *ss-600* means short side is 600, and 130ms includes two networks. Note that they reach their best result on ICDAR 2015 using a short edge of 2000, which is much larger than ours. For Zhang *et al.* [48], MS means they used 200, 500, 1000 three scales, and the result is obtained on MSRA-TD500. The theoretical flops per pixel for our three models are 18KOps, 44.4KOps and 331.6KOps respectively, for PVANET, PVANET2x and VGG16.

B. Implementation details

TABLE I: Implementation details. "lr" is short for learning rate. The size of input image is denoted as "size"."nr" refers to the negative ratio in hard negative mining. "#iter" stands for the number of training iterations.

Dataset	All datasets			IC15	COCO-Text	SVT	IC13
Settings	lr	size	nr	#iter	#iter	#iter	#iter
Pre-train	10^{-4}	384	3	60k	60k	60k	60k
Stage 1	10^{-4}	384	3	8k	20k	2k	2k
Stage 2	10^{-5}	768	6	4k	30k	8k	8k

C. Quadrilateral VS Rotated Rectangle

The rotated rectangle is an approximate simplification of the quadrilateral, which is more flexible in representing arbitrary-oriented text bounding box. Although both representations

[1]https://github.com/bgshih/crnn

TABLE III: Text localization results on ICDAR 2015 Incidental Text dataset.

Methods	recall	precision	f-measure
CNN MSER [48]	0.34	0.35	0.35
AJOU [53]	0.47	0.47	0.47
NJU [48]	0.36	0.70	0.48
StradVision1 [48]	0.46	0.53	0.50
StradVision2 [48]	0.37	0.77	0.50
Zhang et al. [30]	0.43	0.71	0.54
Tian et al. [33]	0.52	0.74	0.61
Yao et al. [38]	0.59	0.72	0.65
Liu et al. [42]	0.682	0.732	0.706
Shi et al. [39]	0.768	0.731	0.750
EAST PVANET2x. RBOX [40]	0.735	0.836	0.782
EAST PVANET2x RBOX MS [40]	0.783	0.833	0.807
TextBoxes++	0.767	0.872	**0.817**
TextBoxes++_MS	**0.785**	**0.878**	**0.829**

TABLE IV: Text localization results on COCO-Text dataset.

Methods	recall	precision	f-measure
Baseline A [49]	0.233	0.8378	0.3648
Baseline B [49]	0.107	0.8973	0.1914
Baseline C [49]	0.047	0.1856	0.0747
Yao et al. [38]	0.271	0.4323	0.3331
Zhou et al. [40]	0.324	0.5039	0.3945
TextBoxes++	0.5600	0.5582	0.5591
TextBoxes++_MS	**0.5670**	**0.6087**	**0.5872**

D. Text localization

TABLE V: Text localization on ICDAR 2013 dataset. P, R, and F refer to precision, recall and f-measure, respectively.

Evaluation protocol	IC13 Eval			DetEval		
Methods	R	P	F	R	P	F
fasttext [54]	0.69	0.84	0.77	–	–	–
MMser [55]	0.70	0.86	0.77	–	–	–
Lu et al. [56]	0.70	0.89	0.78	–	–	–
TextFlow [57]	0.76	0.85	0.80	–	–	–
He et al. [58]	0.76	0.85	0.80	–	–	–
He et al. [59]	0.73	**0.93**	0.82	–	–	–
FCRNall+filts [27]	–	–	–	0.76	0.92	0.83
FCN [30]	0.78	0.88	0.83	–	–	–
Tian et al [60]	0.84	0.84	0.84	–	–	–
Qin et al. [61]	0.79	0.89	0.83	–	–	–
Shi et al. [39]	–	–	–	0.83	0.88	0.85
Tian et al. [33]	–	–	–	0.83	**0.93**	0.88
Tang et al. [62]	0.87	0.92	**0.90**	–	–	–
SSD [10]	0.60	0.80	0.68	0.60	0.80	0.69
TextBoxes [13]	0.74	0.86	0.80	0.74	0.88	0.81
TextBoxes MS [13]	0.83	0.88	0.85	0.83	0.89	0.86
TextBoxes++	0.74	0.86	0.80	0.74	0.88	0.81
TextBoxes++_MS	**0.84**	**0.91**	0.88	**0.86**	0.92	**0.89**

TABLE VI: Runtime and performance comparison on ICDAR 2015 Incidental Text dataset. "F" is short for F-measure. See corresponding text for detailed discussions.

Method	Res	FPS	F
Zhang et al. [30]	MS*	0.476	0.54
Tian et al. [33]	ss-600*	7.14	0.61
Yao et al. [38]	480p	1.61	0.65
EAST PVANET [40]	720p	**16.8**	0.757
EAST PVANET2x [40]	720p	13.2	0.782
EAST VGG16 [40]	720p	6.52	0.764
Shi et al. el. [39]	768×768	8.9	0.750
TextBoxes++	1024×1024	11.6	**0.817**
TextBoxes++_MS	MS*	2.3	**0.829**

TABLE VII: F-measures for word spotting and end-to-end results on ICDAR 2015 Incidental Text dataset. See the corresponding dataset description in Section IV-A for strong, weak, and generic lexicon settings. Note that the methods marked by "*" are published on the ICDAR 2017 Robust Reading Competition website: http://rrc.cvc.uab.es.

Methods	IC15 word spotting			IC15 end-to-end		
	strong	weak	generic	strong	weak	generic
Megvii-Image++ *	0.4995	0.4271	0.3457	0.4674	0.4	0.3286
Yunos_Robot1.0*	0.4947	0.4947	0.4947	0.4729	0.4729	0.4729
SRC-B-TextProcessingLab*	0.5408	0.5186	0.3712	0.526	0.5019	0.3579
TextProposals + DictNet*	0.56	0.5226	0.4973	0.533	0.4961	0.4718
Baidu IDL*	0.6578	0.6273	0.5165	0.64	0.6138	0.5071
HUST_MCLAB*	0.7057	–	–	0.6786	–	–
TextBoxes++	**0.7645**	**0.6904**	**0.5437**	**0.7334**	**0.6587**	**0.5190**

TABLE VIII: F-measures for word spotting and end-to-end results on ICDAR 2013 dataset. The lexicon settings are the same as for ICDAR 2015 Incidental Text dataset.

Methods	SVT spotting	SVT-50 spotting	IC13 spotting			IC13 end-to-end		
			strong	weak	generic	strong	weak	generic
Alsharif [63]	–	0.48	–	–	–	–	–	–
Jaderberg [8]	0.56	0.68	–	–	0.76	–	–	–
FCRNall+filts [27]	0.53	0.76	–	–	0.85	–	–	–
TextBoxes	**0.64**	**0.84**	0.94	0.92	**0.87**	0.91	0.89	0.84
TextBoxes++	**0.64**	**0.84**	**0.96**	**0.95**	**0.87**	**0.93**	**0.92**	**0.85**