

LICENSE PLATE DETECTION USING DEEP LEARNING
OBJECT DETECTION MODELS

LEONG KAR WAN

MASTER OF ENGINEERING SCIENCE

FACULTY OF ENGINEERING AND GREEN
TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN
NOVEMBER 2023

**LICENSE PLATE DETECTION USING DEEP LEARNING OBJECT
DETECTION MODELS**

By

LEONG KAR WAN

A dissertation submitted to the Faculty of Engineering and Green Technology,
Universiti Tunku Abdul Rahman,
in partial fulfillment of the requirements for the degree of
Master of Engineering Science
November 2023

ABSTRACT

LICENSE PLATE DETECTION USING DEEP LEARNING OBJECT DETECTION MODELS

Leong Kar Wan

Object detection – an extension of image classification task in computer vision can locate any object from any given image input. In the past, this is usually done by traditional hand-crafted feature algorithms i.e., SIFT, SURF, HOG, BRIEF, and ORB. These algorithms have been successful in their field however they do possess some downsides due to their nature. For example, they can be slow in detection speed, not as accurate, and is difficult to develop. Since 2012, deep learning has become an emerging technology that can solve object detection with relatively better performance. However, not many works has been done when it comes to developing a real life application e.g., license plate detection. License plate detection is a challenging task in computer vision because the input image captured can be in different sizes, color, distance, orientation, and lighting condition. This project aims to study and improve license plate detection using deep learning models. As of current year, the model YOLOv4 has achieved 43.5% AP on MS COCO. Meanwhile, EfficientDet-D7 has achieved 55.1 AP on COCO test-dev. This project will use the available off-the-shelves object detection model to train on CCPD license plate dataset. The impact of this project is that it provides informative insights and uncover the

potential of the development of real-life applications using recent deep learning object detection models.

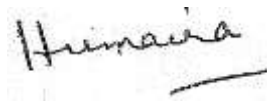
ACKNOWLEDGEMENT

I would like to thank my supervisor Prof Ts Dr Humaira Nisar and my co-supervisor Dr. Yap Voon Voi for guiding me through the UTAR master's degree program for many years, especially during the pandemic. I would also like to thank UTAR for giving me this opportunity to learn knowledge and have valuable experience in this master's degree program.

APPROVAL SHEET

This dissertation entitled “**LICENSE PLATE DETECTION USING DEEP LEARNING OBJECT DETECTION MODEL**” was prepared by LEONG KAR WAN and submitted as partial fulfillment of the requirements for the degree of Master of Engineering Science in Faculty of Engineering & Green Technology at Universiti Tunku Abdul Rahman.

Approved by:



(Prof. Ts. Dr. Humaira Nisar)
Professor/Supervisor
Department of Electronic Engineering
Faculty of Engineering Science
Universiti Tunku Abdul Rahman

Date: ...26 November 2023.....

FACULTY OF ENGINEERING & GREEN TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN

Date: 26 November 2023

SUBMISSION OF DISSERTATION

It is hereby certified that ***Leong Kar Wan*** (ID No: ***19AGM05725***) has completed this dissertation entitled “*License Plate Detection Using Deep Learning Object Detection Model*” under the supervision of Prof. Ts. Dr. Humair Nisar (Supervisor) from the Department of Electronic Engineering, Faculty of Engineering & Green Technology.

I understand that University will upload softcopy of my dissertation in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

Yours truly,



(*Leong Kar Wan*)

DECLARATION

I hereby declare that the dissertation is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTAR or other institutions.

Name LEONG KAR WAN

Date 26 November 2023

TABLE OF CONTENTS

	Page
ABSTRACT	ii
ACKNOWLEDGEMENTS	iv
APPROVAL SHEET	v
SUBMISSION SHEET	vi
DECLARATION	vii
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF ABBREVIATIONS/NOTATION/GLOSSARY OF TERMS	xii
CHAPTER	
1.0 INTRODUCTION	1
1.1 Background	1
1.2 Problem Statement	5
1.3 Research Objectives	5
2.0 LITERATURE REVIEW	7
2.1 Automatic License Plate Recognition (ALPR)	7
2.2 Object Detection	11
2.3 Non-Neural Network Object Detection Algorithm	13
2.3.1 Scale-Invariant Feature Transform (SIFT)	13
2.3.2 Speeded Up Robust Features (SURF)	13
2.3.3 Binary Robust Independent Elementary Features (BRIEF)	14
2.4 Neural Network-based Object Detection Models	15
2.4.1 EfficientDet	15
2.4.1.1 Issues Regarding EfficientDet	17
2.4.2 YOLOv4	17
2.4.3 CenterNet	20
2.4.4 Faster R-CNN	21
2.4.5 SSD	23
3.0 METHODOLOGY	25
3.1 Overview	25
3.2 Dataset	25
3.3 Hardware	27
3.4 Software	28
3.5 Data Preparation	28
3.5.1 YOLO format	28
3.5.2 TensorFlow format	29
3.6 Data Augmentation	30
3.6.1 YOLOv4	30
3.6.2 EfficientDet	32

3.7	Preprocessing	33
3.7.1	Enlargement	34
3.7.2	Sharpening	34
3.7.3	Gamma Correction	35
3.7.4	Contrast Limited Adaptive Histogram Equalization (CLAHE)	35
3.7.5	Non-Local Means Denoising	35
3.8	Models	36
3.9	Updated Model Changes	36
3.10	Evaluation Method	37
3.10.1	Steps to Calculate mAP@0.70	40
3.10.2	Performance Metrics	40
3.11	Detailed Diagram / Chart of Proposed Model	41
4.0	RESULT AND DISCUSSION	49
4.1	Overview	49
4.2	Model Training Loss	49
4.2.1	EfficientDet-D6	50
4.2.2	EfficientDet-D0	50
4.2.3	CenterNet-ResNet50-512x512	50
4.2.4	SSD-ResNet50-640x640	50
4.2.5	FasterRCNN-ResNet50-640x640	51
4.3	Results of Models Training	51
4.4	Preprocessing	55
4.5	Modified YOLOv4-CSP (Proposed Model)	58
5.0	CONCLUSION	63
	REFERENCES/BIBLIOGRAPHY	65

LIST OF TABLES

Table		Page
2.1	License Plate Detection (Non-DL)	8
2.2	License Plate Detection (DL)	9
2.3	Faster R-CNN Family Speed and Limitations	22
2.4	SSD Performance Comparison	23
3.1	YOLOv4-CSP Architecture	46
3.2	YOLOv4-CSP-Modified Architecture	50
4.1	Initial Models' Accuracy (mAP@0.70) and Speed (FPS) for Each Test Set	58
4.2	Pre-processing Result	62
4.3	0.69 IoU Problem	65
4.4	YOLOv4-CSP-Modified Architecture	66
4.5	Accuracy (mAP@0.70) of Modified YOLOv4-CSP vs Original YOLOv4-CSP	70

LIST OF FIGURES

Figures		Page
1.1	Malaysia Automotive Sales 2022	2
2.1	Example of Image Recognition	12
2.2	Example of Object Detection	12
2.3	EfficientDet Overall Architecture	15
2.4	Models FLOPS vs. COCO accuracy	16
2.5	Scaled-YOLOv4 vs. Others	18
2.6	Newest Version Scaled-YOLOv4 Compared to EfficientDet and Others	20
2.7	SSD: Single Shot MultiBox Detector	24
2.8	Output from SSD	25
3.1	Examples from CCPD Dataset	28
3.2	Example Images from Six Categories	30
3.3	Mosaic / Blur / Flip / Crop / Exposure / Aspect Ratio / Hue / Saturation	35
3.4	Example of Image Preprocessing	36
3.5	Model Detection with 31% Confidence after Preprocessing	37
3.6	Example of GT Box and Prediction Box	42
3.7	Intersection over Union	43
4.1	EfficientDet-D6 Training Loss	56
4.2	EfficientDet-D0 Training Loss	57
4.3	CenterNetResNet50-512x512 Training Loss	57
4.4	SSDResNet50-640x640 Training Loss	57
4.5	FasterR-CNNResNet50 Training Loss	58
4.6	Mean Average Precision for All Models for CCPD Dataset	59
4.7	Speed for All Models for CCPD Dataset	60
4.8	Ground Truth Box (Left) vs Model Prediction (Right)	65
4.9	YOLOv4-CSP-Modified Training Loss Graph	69

LIST OF ABBREVIATIONS

ALPR	Automatic License Plate Recognition
ANPR	Automatic Number Plate Recognition
BoF	Bag of Freebies
BoS	Bag of Specials
BRIEF	Binary Robust Independent Elementary Features
CCPD	Chinese City Parking Dataset
CLAHE	Contrast Limited Adaptive Histogram Equalization
CNN	Convolutional Neural Network
COCO	Common Objects in Context
DL	Deep Learning
FAST	Features from Accelerated Segment Test
FLOPS	Floating-Points Operations per Second
FN	False Negative
FP	False Positive
FPS	Frame Per Second
GMM	Gaussian Mixture Model
GT	Ground Truth
HSI	Hue, Saturation, Intensity
HSV	Hue, Saturation, Value
IoU	Intersection over Union
mAP	Mean Average Precision
MSE	Mean Squared Error

NMS	Non-Maximum Suppression
OCR	Optical Character Recognition
ORB	Oriented FAST Rotated BRIEF
R-CNN	Regional Convolutional Neural Network
RPN	Region Proposal Network
SCW	Concentric Sliding Window
SIFT	Scale Invariant Feature Transform
SOTA	State-Of-The-Art
SS	Selective Search
SSD	Single Shot Detector
SURF	Speeded Up Robust Features
SWSCD	Sliding Window Single Class Detection
TN	True Negative
TP	True Positive
TPUs	Tensor Processing Units
YOLO	You Only Look Once

CHAPTER 1

INTRODUCTION

1.1 Background

Computer vision is an important field of research for many real-life applications. Specifically, in object detection, traditional image processing algorithms such as Scale-Invariant Feature Transform (SIFT) (Ng, 2003), Speeded Up Robust Features (SURF) (Bay et al., 2008), Features from Accelerated Segment Test (FAST) (Viswanathan DG, 2011), Binary Robust Independent Elementary Features (BRIEF) (Calonder et al, 2010), and Oriented FAST and Rotated BRIEF (ORB) (Rublee et al, 2011) are used to locate and identify an object from a given image frame. Over the past decade, a new method known as deep learning (DL) has overtaken the traditional methods in the field. However, there are many unknown factors when deploying deep learning algorithms in real-life applications, e.g., how well does the DL model perform? This project will dive into a popular use case scenario i.e., license plate recognition task using DL models which in the past was performed using traditional image processing methods.

License plate recognition is an important task in many real-life applications e.g., parking management system, traffic control system.

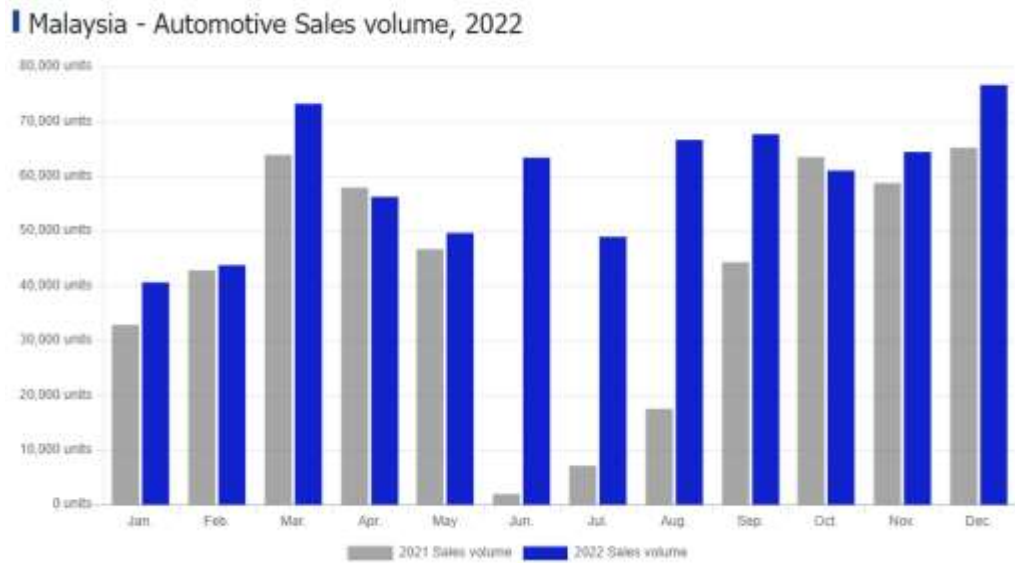


Figure 1.1 Malaysia Automotive Sales 2022 (*Malaysia - Flash Report, Automotive Sales Volume, 2022 - MarkLines Automotive Industry Portal, 2022*)

In Malaysia, the number of cars on the road has been increasing each year. Figure 1.1 shows the total number of car sales in 2022 compared with 2021. With the increase in automotive volume especially in the cities, license plate recognition systems can be useful in busy roads or shopping malls to avoid traffic congestion, car park management, etc. In China, the highway is always stuck with loads of traffic going back to their hometown every Chinese New Year. In Malaysia, due to the toll stop, traffic congestion will happen if the number of automobiles keeps increasing in the coming years. One innovative way to avoid the same happening as in China is to develop a license plate recognition system with a high-speed camera. The car doesn't need to stop at the toll station hence the traffic congestion is avoided. This can also eliminate many sub-systems from the toll station such as the automated car blocker, digital card scan, etc., saving a high amount of maintenance fees each year. Another example would be to track stolen cars on the road. In other countries, it is hard to search for any stolen car due to

its geometrical disadvantage. An unregistered stolen car can be used to conduct crime in many ways. With an on-the-road car tracking system, license plate recognition can solve such a problem.

In the past decade, Automatic License Plate Recognition (ALPR) had been a popular research topic in computer vision. The algorithm is generally divided into three tasks i.e., license plate detection, character segmentation, and character recognition. Image processing techniques e.g., edge detection, color matching, histogram analysis, and others were used to extract the location of the license plate and characters from the output of video-capturing devices. There are challenging issues regarding license plate detection that needs to be resolved. Therefore, much research has been conducted to improve the efficiency, speed, and accuracy of license plate detection tasks. A recent paper (Habeeb *et al.*, 2021) can detect and recognize Iraqi and Malaysian license plates with 90.23% and 90.60% accuracy using SVM and YOLOv2-ResNet50. Another paper (Jørgensen, 2017) using YOLOv2-darknet19 can achieve an accuracy of 99.8% out of 410 samples of license plates. This raises the question, is it worth continuing with research into vehicle license plate detection? But the answer is clear, as the dataset used is normally small, so the detection accuracy is good. However, if the larger datasets are used then work is more challenging.

Despite the availability of several well-known objection detection models, there are still problems that researchers face. One problem that researchers face is the non-uniformity of the license number plate models in different parts of the world. The license number plate comes in different sizes

and information on the number on the plate also varies. The other problem is the low resolution of the license number plate that shows up on the video on surveillance systems (Sharma, Karan, and Karan Sing, 2015). Researchers also face another challenge which is speed. When it comes to video, object detection models need to carry out analysis in an environment that is changing rapidly. This means that object detection models must be able to classify objects of interest and be very fast during prediction to be able to identify objects that are in motion. For example, in a football match, the object detection model must be able to track the movement of the ball and players on the football field. The next paragraph will further explore a license plate detection and recognition technique.

The vehicle license plate detection and recognition techniques are commonly known as the Automatic number-plate recognition (ANPR) system. In the past template matching techniques (Ashtari et al., 2011) were used to identify vehicle number plates. This approach identifies the width, height, and contour area of the number plate. Some ANPR systems use basic image processing techniques to identify fixed license plates pattern in controlled conditions such as lighting and distance. There are advanced ANPR systems that make use of object detection models to localize license plates in images. Another technique in some ANPR systems is to create a unique neural network that can output an enhanced image so that it can be interpreted by the Optical Character Recognition (OCR) system.

Although ANPR has been a success in detecting the license plate, it faces some challenges. There are several problems with ANPR systems. ANPR faces

the non-uniformity of license plate number problem as mentioned previously, that is, the length of the license plate number varies for different cities and countries. Another problem that the ANPR system is facing is that the ununiform light conditions throughout the day and night may affect the detection accuracy. One report (*ANPR Cameras | ANPR accuracy test, 2019*), shows that the camera can capture the image at a decent view during the daytime. The same report cited that a slower shutter speed produces smudged images. The report also cited that the success rate dropped considerably towards the end of the twilight.

1.2 Problem Statement

Object detection using traditional image processing methods such as SIFT, SURF, HOG, BRIEF, and ORB has various disadvantages when it comes to developing a real-life application and has low accuracy compared to the recent deep learning approach. For example, the number of detectable objects is limited. The orientation and angle of the object also affect the accuracy. The types, shapes, and sizes, and the presence of occlusion will also cause the object to be non-detectable. More importantly, the speed of detection is also slow.

1.3 Research Objectives

Hence, we formulate the following research objectives.

- i. Explore various deep-learning object detection models.
- ii. To improve the speed and accuracy of license plate detection tasks using deep learning models through image preprocessing, training, and modifying existing model architecture.

The rest of the dissertation is organized as chapter 2, the literature review; chapter 3, the methodology; chapter 4, results, and discussion; chapter 5, conclusion, and future works.

CHAPTER 2

LITERATURE REVIEW

2.1 Automatic License Plate Recognition (ALPR)

In the last 20 years of research, Automatic License Plate Recognition (ALPR) algorithms are divided into two to three tasks i.e., license plate detection, character segmentation, and character recognition. Image processing techniques e.g., edge detection, color matching, histogram analysis, and others were used to extract the locations of objects in an image.

In the license plate detection task, Anagnostopoulos et al used concentric sliding windows (SCWs) to perform segmentation and extract Region of Interest (RoI) (Anagnostopoulos et al., 2006). Zheng, Zhao, and Wang enhanced the luminance and contrast of the image before performing edge detection. Chang et al. used a combination of edge property and color property in the form of Hue, Saturation, and Intensity (HSI) to form a fuzzy map. Hsu, Chen, and Chung used Sobel vertical operator to extract edges followed by applying the Gaussian mixture model (GMM) and EM algorithm. Faradji, Rezaie and Ziaratban used vertical edges, histograms, dilations, erosion, and median filter. Ashtari, Nordin, and Seyed Mostafa Mousavi Kahaki used template matching and color features. Zang et al. used a modified traditional visual attention model with fusing color, intensity, and orientation feature maps. Table 2.1 gives an overview of image processing-based license plate detection algorithms.

Table 2.1 License Plate Detection Algorithm (Non-DL)

Year	Method	Image Input Siz/Resolution	Dataset Size	Accuracy	Speed	Test System
Anagnos-topoulos <i>et al.</i> , 2006	Sliding Concentric Windows (SCWs)	Avg 90x28 Avg 102x32 61x19-153x48 61x19-153x48 138x29-155x37 Total	427 258 310 154 185 1334	98.1% 92.2% 97.7% 95.5% 97.3% 96.5%	111ms	Pentium IV at 3.0 GHz with 512-MB RAM
Zheng, Zhao and Wang, 2005	Vertical edge	384x288	163 218 784	100.0% 100.0% 99.7%	47.9ms	Pentium-4 2.4 GHz, 256 MB RAM PC
Chang <i>et al.</i> , 2004	Color processing + edge detection + fuzzy map	640x480 768x512	639 449	98.8% 96.7%	0.4s	a Pentium IV-1.6 GHz PC
Hsu, Chen and Chung, 2013	Edge clustering	-	AC-681 LE-757 RP-611	93% 93% 94%	0.21s 0.26s 0.32s	Windows PC with a Pentium Dual Core 2.4-GHz processor and a 2-GB RAM with C++ compiler
Faradji, Rezaie and Ziaratban, 2007	Vertical Edge, Histogram, Compact Factor, Dilation, Regions in Common, Filling Holes, Erosion, Median Filter	384x288	400	83.50%	32.4ms	Pentium-4 2.4 GHz, 256 MB RAM
Ashtari, Nordin and Seyed Mostafa Mousavi Kahaki, 2011	Template Matching, Colour Feature,	-	250	96.80%		dual-core 1.7-GHz CPU and 4-GB RAM.
Zang <i>et al.</i> , 2015	Visual Attention Map	-	835	99.2%	140ms	Intel Core 2 Duo 2.2GHz desktop computer 4GB Ram

Today, many large datasets are available publicly. By the combination of large datasets, the computing power of newer hardware, and innovative and successful deep learning mechanisms, researchers can create an object detection

model with high accuracy and speed. Specifically, in the license plate detection task, researchers have been studying the performance of CNN models in detecting license plates. Selmi, ben Halima and Alimi used the image processing method before feeding the image into the CNN to classify the contours into LP and non-LP. Habeeb et al. used YOLOv2 with ResNet50 backbone to detect license plate bounding box. Jørgensen used darknet19 and YOLOv2 for detection. The last layer was removed and replaced with a linear classifier. Next, three additional convolutional layers were added. Hendry and Chen utilized Sliding Window Single Class Detection (SWSCD) to detect both plates and characters because the original YOLO has difficulty detecting bounding boxes of a small object due to its anchor-based detection. This sliding window approach is possible due to the speed of YOLO-tiny. They modified YOLO-tiny even further by decreasing its number of layers to increase its speed. Table 2.2 gives an overview of license plate detection method using deep learning approach.

Table 2.2 License Plate Detection Algorithm (DL)

Year	Method / Model	Pre-processing used	Image Input Size/Resolution	Dataset Size (Test)	Accuracy	Speed	Test System
Selmi, ben Halima and Alimi, 2017	2-conv-layers + 2-fc-layers CNN.	Morphological operations, adaptive threshold, contours, geometric	896x592	Caltech-126 + AOLP-2049,	P93.80% R91.30% P92.60% R96.80% P93.50% R93.30% P92.90% R96.20%	NA	Intel PC Core i7 CPU 2 GHz, 8 GB of RAM, and ubuntu LTS 16 as the operating system. python,

		filtering					opencv 3.1, and a TensorFlow framework.
Ha-beeb <i>et al.</i> , 2021	SVM, YOLOv2-Res-Net50, 7 anchor boxes	Skew correction	2720x1232	Iraqi-404 Malay-sian-681	90.23% 90.60% (Five-fold)	-	computer with a built-in GPU and 16 GB of RAM
Jørgensen, 2017	YOLOv2-darknet19-modified		2464x1632 → 416x416	410	99.8%	0.0173s (17ms)	GeForce GTX 1070 8GB GPU
Henry and Chen, 2019	Modified-tiny-YOLOv1, Sliding Window Single Class Detection		NA	AC-681 LE-757 RP-611 Total-2049	98.22%	825.81ms	Nvidia GTX970 GPU accelerator 4GB memory, i7 Central Processing Unit (CPU), and 16GB DDR2 memory

As mentioned in Chapter 1, this research is concerned with the object detection algorithm developed in recent years, the State-Of-The-Art (SOTA) deep learning convolution neural network models. Therefore, this chapter will explore various algorithms related to this research such as Google AI Team's EfficientDet (Tan et al., 2019), YOLO family (Bochkovskiy et al., 2020), and others. To date, Efficientdet models and YOLO models are among the top ten models from *paperswithcode.com* in Computer Vision – Object Detection task,

COCO test-dev dataset. In terms of accuracy, EfficientDet scored 57.3 AP while YOLOv4 scored 56.0 AP. In terms of speed, YOLOv4 scored 16 FPS while EfficientDet scored 6.5 FPS. However, other models, that is, Faster R-CNN, CenterNet, and SSD are also evaluated in this research to obtain a better result for evaluation and comparison purposes.

2.2 Object Detection

Since one of the aims of this study is to use CNN-based objection detection models to carry out license plate identification, it would be logical to explore object detection techniques. So, this section will explore object detection.

The term object recognition refers to a computer vision technique used to identify objects in each digital image or video. With object detection, once an object of interest is detected it draws bounding boxes around the detected object. This enables the algorithm to locate the object of interest in any given image or video.

There is some confusion concerning object detection and image recognition, so this dissertation would like to clarify this before proceeding further. With image recognition, the whole image is given a label. For example, an image of a car is labeled as “car” and an image of two cars is also labeled as “car”, as shown in Figure 2.1.



Figure 2.1 Example of Image Recognition

With object detection, a box is drawn around each car and the object ‘car’ is labeled accordingly. Different models will have different approaches how to determine the exact location of the object. As can be observed object detection provides more information, for example, it can predict the location of the car or cars and label them individually, as shown in Figure 2.2.



Figure 2.2 Example of Object Detection

Object detection methods can be divided into two approaches, that is, neural network-based or non-neural. Examples of non-neural approaches include but are not limited to scale-invariant feature transform (SIFT) (Ng, 2003), speeded-up robust features (SURF)(Bay *et al.*, 2008), and binary robust independent elementary features (BRIEF) (Calonder *et al.*, 2010). The following section will briefly look at these non-neural or image processing based approaches.

2.3 Non-Neural Network Object Detection Algorithm

2.3.1 Scale-Invariant Feature Transform (SIFT) (Ng, 2003)

The SIFT algorithm consists of four steps. These four steps are:

Scale-space peak selection: Potential location for finding features.

Step 1: Key point Localization - In this step, the key features are located.

Step 2: Orientation Assignment - The orientation is assigned to key points.

Step 3: Key point descriptor – Key points are constructed as a high-dimensional vector.

Step 4: Key point Matching

2.3.2 Speeded-Up Robust Features (SURF) (Bay *et al.*, 2008)

SURF is a feature detector and descriptor. The SIFT descriptor partly inspired SURF. SURF is a fast and robust algorithm used for local, similarity invariant representation and comparing of digital images. Bay *et al.* cited that SURF performance is close to or better than previously proposed schemes, for example, SIFT. This is possible because SURF uses integral images for image convolutions. SURF consists of the following steps:

Step 1: Feature Extraction

Feature extraction refers to the process of extracting relevant information features from an input image. The features extracted must contain important and unique attributes of the image.

Step 2: Feature Description

The SURF descriptor is generated by fixing a position based on information gathered around the point of interest. Next, a square region is constructed and aligned to the selected position and extract the SURF descriptor

from it. The square region is then divided up into smaller sub-regions. For each sub-region, a few features are computed at a regular-spaced sample point. The feature descriptor is based on the Haar wavelet response around the point of interest.

Step 3: Feature Matching

The purpose of feature matching is to analyze and match the features of two or more digital images. In the SURF algorithm, the matching degree is fixed by computing the Euclidean distance between two feature points.

2.3.3 Binary Robust Independent Elementary Features (BRIF) (Calonder *et al.*, 2010)

BRIF uses binary strings as an efficient feature point descriptor. The algorithm finds the binary strings directly without finding descriptors. Next, it makes use of the smoothed image patch and identifies a set of location pairs. After those compares the pixel intensity comparisons on the location pairs identified earlier.

2.4 Neural Network-based Object Detection Models

Several recent (2019~2020) State-Of-The-Art (SOTA) object detection models were reviewed, including EfficientDet (Tan, Pang, and Le, 2019), Yolov4 (Bochkovskiy, Wang and Liao, 2020), CenterNet (Duan *et al.*, 2019), and a few older models e.g., Faster R-CNN (Ren *et al.*, 2015) and Single Shot Detector (SSD) (Liu *et al.*, 2016) with ResNet backbone. As mentioned in chapter 1, this research aims to compare and evaluate the performance, strengths, and weaknesses of recent year's SOTA object detection models, therefore this

chapter will explore the design and architecture of each SOTA model and study their new innovative ideas and strategies, to further improve the accuracy and speed of the models.

2.4.1 EfficientDet

EfficientDet is developed by Google, a family of a model that is scalable and aims to be efficient. The classification by neural network (EfficientNet) is extended with a bi-directional feature network (BiFPN) that can be scaled freely for different resolutions. It has a high level of accuracy compared to other object detectors using significantly less computation (Tan and Yu, 2020). Figure 2.3 gives an overview of EfficientDet Overall Architecture.

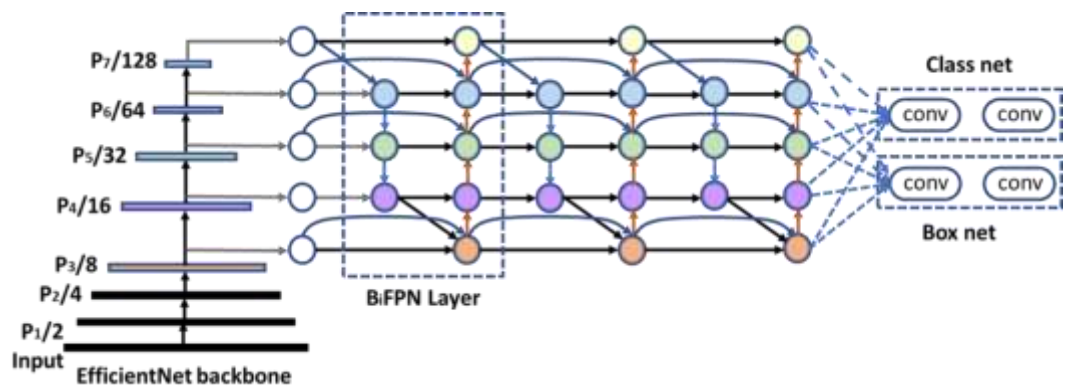


Figure 2.3 EfficientDet Overall Architecture (Tan, Pang, and Le, 2019)

The novelty of EfficientDet is that it is derived from constructing the backbone and neck. It used the EfficientNet as the backbone of the EfficientDet algorithm. EfficientNet is scalable and provided scales pre-trained on ImageNet. In addition, When EfficientNet was announced, it was one of the highest-performing ImageNet classifiers. For example, EfficientNet-B7 achieved 84.4% accuracy.

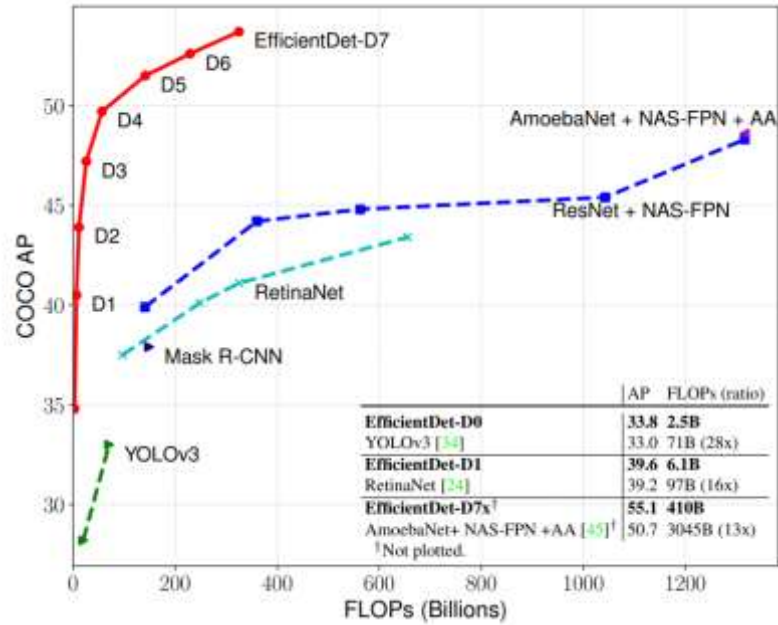


Figure 2.4 Model FLOPs vs. COCO accuracy (Tan, Pang, and Le, 2019)

(Tan, Pang and Le, 2019) also used a weighted bi-directional FPN (BiFPN) and the objective is to blend features at different levels of the backbone. Some new strategies were implemented in BiFPN. For example, the new weighted feature fusion is using Fast Normalized Fusion $O = \sum_i \frac{w_i}{\epsilon + \sum_j w_j} \cdot I_i$, replaced Softmax Normalized Fusion $O = \sum_i \frac{e^{w_i}}{\sum_j e^{w_j}} \cdot I_i$.

2.4.1.1 Issues Regarding EfficientDet (Jacob, 2020)

This section will discuss issues regarding EfficientDet, however, the following section will explore the areas that can prevent image detection systems from being implemented. The issues are as follows:

1. Data Collection – It is cited that EfficientDet can reduce the amount of data that is needed to generalize a new domain.
2. Model Design and Hyper Parameterization - A model must be constructed from the data collected. In addition, several of the hyperparameters need to be tuned to adapt to the model.

3. Training Time – The model based on the data collected needs to be trained and the time need to train the model is measured as floating-point operation per second (FLOPS).
4. Memory Footprint – The amount of memory needed to store the model has to be determined after the model has been trained.
5. Inference Time – The predations performance must be evaluated to determine whether the model is quick enough to be used in a production setting.

Overall, EfficientDet improved the design concept and architecture of the detection model, increasing the accuracy and speed by a large margin.

2.4.2 YOLOv4 (Bochkovskiy, Wang, and Liao, 2020)

The YOLO object detector has gained popularity for its incredible performance on speed where it can achieve real-time detection in the real world without sacrificing its accuracy. Similar to the other one-shot detector, e.g., SSD, it was trained in a single phase.

Redmon et al. wrote the original YOLO, it was then continued by Bochkovskiy et al. due to historical reasons. The model consists of three main components: the head, neck, and backbone. The backbone uses convolutional layers to detect image features before processing them. To detect the image features the model is first trained on a classification dataset. Once the classification network is trained, the neck and the head are responsible to predict the location offsets of the objects from the image. The final outputs are the

coordinates of the bounding boxes and their objectness probability. Refer to figure 2.5.

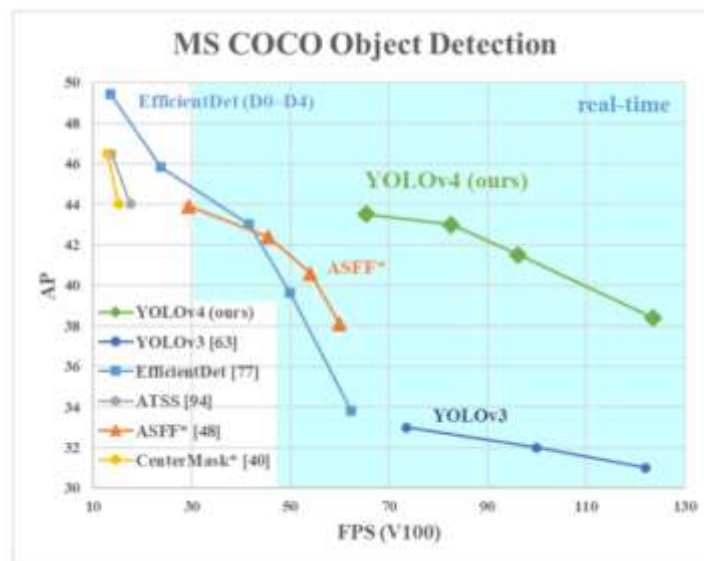


Figure 2.5 Scaled-YOLOv4 vs. Others (Bochkovski, Wang, and Liao, 2020)

As can be observed from Figure 2.3.2.1, YOLOv4 is faster and more accurate than YOLOv3, EfficientDet, ASFF, ATSS, and CenterMask.

It is cited that YOLOv4 was a collection of established computer vision algorithms. This collection of algorithms has been combined and validated through the research process.

The contribution of YOLOv4 as compared to its predecessor is that it introduced Bag-of-Freebies (BoF) and Bag-of-Specials (BoS). BoF is a collection of techniques that increases accuracy but only increases inference time by a little, e.g., improving bounding box loss using IoU, GIoU, and CioU instead of only Mean Squared Error (MSE). Other BoF includes various new data augmentation, e.g., include mixup, cutout, cut mix, mosaic, and focal loss for

data imbalance issues. BoS are techniques that are more focused on the post-processing part, e.g., SPP module, attention module, feature integration, Mish activation, and soft Non-Maximum Suppression (NMS).

Another contribution by YOLOv4 is to design the model so that it can be trained on commercial GPU, e.g., RTX2080TI. However, YOLOv4 is maintained and developed constantly, a better version of YOLOv4 i.e., Scaled-YOLOv4 has been released and tops the original YOLOv4. Figure 2.6 below shows the performance between them.

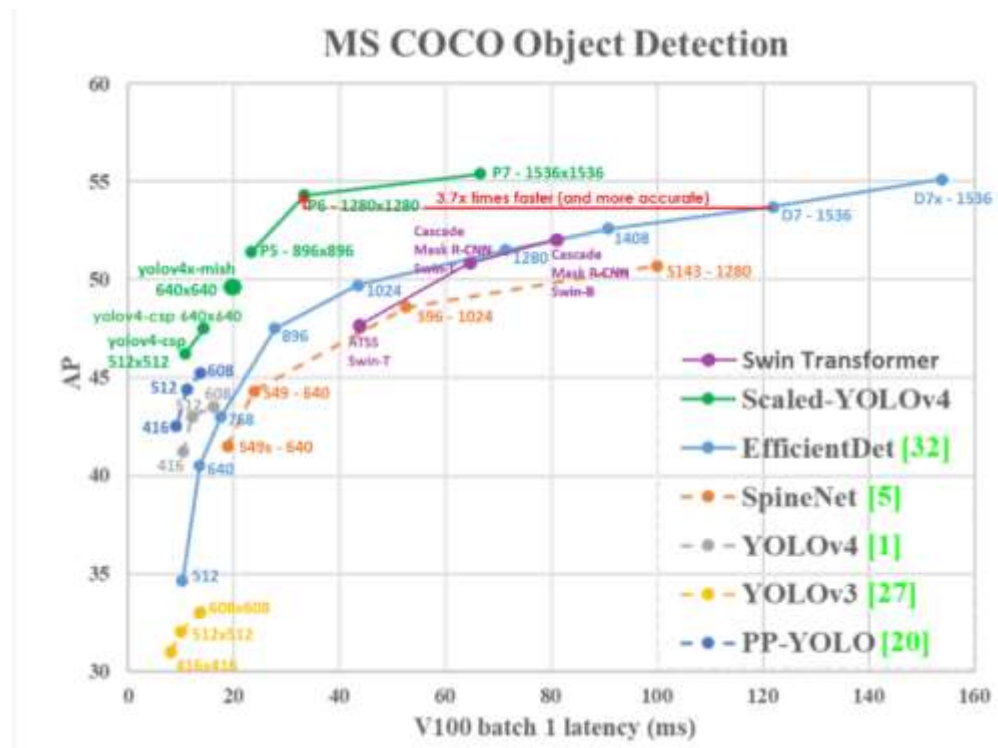


Figure 2.6 Newest Version Scaled-YOLOv4 Compared to EfficientDet and Others (Bochkovskiy et al., 2021)

2.4.3 CenterNet (Duan et al., 2019)

CenterNet is a highly accurate one-stage detector that works slightly differently from YOLO and SSD. It has an additional output, a model heatmap

that has a strong value towards the center of an object that can denote as the center. The steps to generate that center point is as follows:

- a. A Maxpool is applied to the model heatmap to generate a better midpoint representation.
- b. Take the model heatmap and the output from a. to perform a boolean operation.
- c. Multiply b. with the model heatmap.
- d. Reduce center point output with a confidence threshold.

The novelty of CenterNet is to remove the inefficient Non-Maximum Suppression (NMS) with the above center point representation. The combination of two corner points (top-left, bottom-right) and a center point formed key point triplets that can correctly determine the location of an object more efficiently compared to other anchor-based bounding boxes.

2.4.4 Faster R-CNN (Ren *et al.*, 2015)

This section will briefly look at R-CNN before exploring Fast R-CNN, and Faster R-CNN. This is necessary as Fast R-CNN precedes Faster R-CNN.

R-CNN stands for “Region-based Convolutional Neural Networks”. This is first proposed by (Girshick *et al.*, 2014). R-CNN consists of two steps. The first step is to identify object region bounding boxes. Secondly, it extracts CNN features from each region to be classified.

Girshick improved the speed of R-CNN by unifying three independent models into one common framework. In addition, the proposed model also increased the shared computation results. This model is called Fast R-CNN.

Ren et al. introduced the Faster R-CNN model. The Faster R-CNN model is a result of constructing a single, unified model which consists of the region proposed network (RPN) and the Fast R-CNN model with shared convolution feature layers. Table 2.3 shows difference between three R-CNN models in terms of speed and limitations.

Table 2.3 Faster R-CNN Family Speed and Limitations

Algorithm	Prediction (time/image)	Limitations
R-CNN (Girshick et al., 2014)	40 – 50 sec	Requires lots of computation time since each region is passed to the CNN separately. In addition, three different models are required to make predictions.
Fast R-CNN (Girshick, 2015)	2 sec	The computation time is high. This is the result of a slow selective search.
Faster R-CNN (Ren et al., 2015)	0.2 sec	The performance is dependent on the performance of the previous system. The reason is that the object proposal requires time and because three different systems are operating and working sequentially.

Faster R-CNN replaced selective search (SS) in the previous Fast R-CNN algorithm with two additional convolution layers, i.e., 256-d feature vector layer, and another output layer at each convolutional position, which contain regressed bounds + objectness score of k region proposals, depending on aspect ratios and scales, thus network can be trained end to end. The advantage of merging region proposal into the network is that it replaced CPU computing time with GPU and thus can reduce computing time (10ms per image) hence increasing performance (5fps). The Faster R-CNN algorithm has reported 73.2% mAP on PASCAL VOC 2007 and 70.4% mAP in 2012.

2.4.5 SSD (Liu *et al.*, 2016)

Single Shot Detector (SSD) paved the road to the design of one-shot detector alongside with YOLO model in the year 2016. Object detection is a difficult task, the goal is to find the four coordinates of the wanted object from the given image input, which sometimes consists of multiple objects of different sizes blocking each other. The innovative idea one can remove the region proposal network (RPN) as implemented in the two-shot detector Faster R-CNN is to divide an image into some grid cells. Then, many rectangular boxes of different ratios are preset on each grid cell, known as the default boxes. As cited in the paper, this method has a huge disadvantage when it comes to detecting small objects as compared to Faster R-CNN. The introduction of ‘default boxes’ with different ratios aims to solve this problem.

The following Table 2.4 shows the performance of SDD compared to some notable models:

Table 2.4 SSD Performance Comparison (Hui J, 2018)

	Image Resolution	Output Bounding-Boxes	mAP (VOC2007 test set)	Frame Per Second (Titan X GPU)
SSD512	512 x 512	24564	79.8	19
SSD300	300 x 300	8732	77.2	46
YOLO	448 x 448	98	63.4	45
Faster R-CNN	Around 1000 x 600	Around 6000	73.2	7

The SSD model consists of two parts:

1. a backbone, and
2. SSD head.

In the recent deep learning object detection models, this type of detector employs two major parts in their architecture, known as the backbone, and the head. Different from image recognition tasks, an object detector needs to output a coordinate for a detected object. However, one way to train this model is to extend an image recognition model with additional convolutional layers to get the correct output. SSD is among the successful models that had achieved a very good result with such implementation.

While the backbone is responsible to extract the features from the input image, the head of the SSD needs to use those feature maps to determine the location of the objects. A few designs concept is implemented in the SSD head, e.g., grid cell representation, multi-scale features, and different aspect ratio of the anchor box. With multi-level viewing of the image, SSD can determine the bounding box of an object from the image even if the object appears to be smaller or bigger compared to when the model is solely trained on the image recognition dataset. Figure 2.7 shows overall network architecture of SSD.

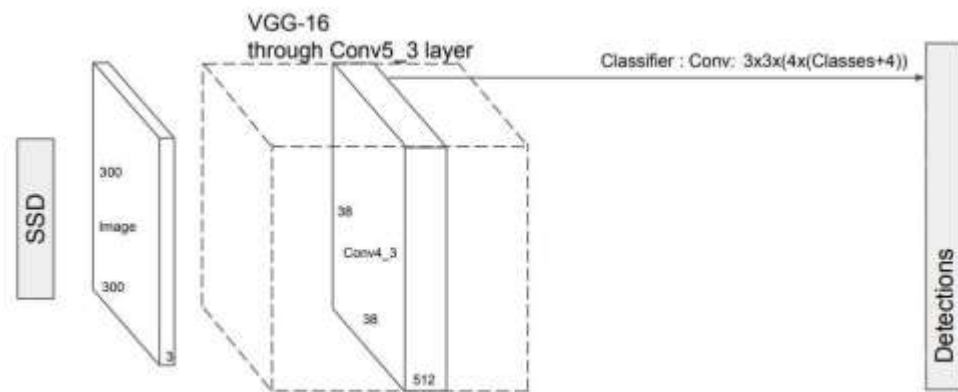


Figure 2.7 SSD: Single Shot MultiBox Detector (Hui J, 2018)

To improve accuracy, SSD uses different data augmentation techniques i.e., to deploy a larger scale of sampling, hard negative mining which only chooses negative examples of high confidence loss and extends the loss function to fit the number of default boxes. This combination of multiple essential techniques allowed SSD to extract semantic meaning from the input image and preserve the spatial structure of a low-level resolution image and achieved a better result than the other models i.e., Faster R-CNN and YOLO. Figure 2.8 demonstrates output from SSD.



Figure 2.8 Output from SSD (Karol Majek, 2018)

CHAPTER 3

METHODOLOGY

3.1 Overview

This section will elaborate the research methodology, the dataset used in this project, hardware, and software used. Next, how the data is being prepared to be able to use for training is discussed. Data augmentation used in the different frameworks will be presented. To increase models' accuracy, images are preprocessed before model detection. Image processing methods will be discussed next. Then, each model of choice and their corresponding framework will be presented. The selected model to improve its performance and its modification will be discussed. Lastly, the evaluation metric and steps will be briefly explained.

3.2 Dataset -- Chinese City Parking Dataset (CCPD) (Xu *et al.*, 2018)

CCPD is a dataset that consists of multiple challenging test images for license plates in China. The characteristic of Chinese license plate is that they have a blue background, white foreground that consists of letters, alphabets, and a Chinese character that represents the province. This dataset consists of train, valid, and test images. To test the models mentioned above, this project will be using test images from the Chinese City Parking Dataset (CCPD). This dataset holds 250k unique car license plate images. The image data include license plate location annotations in individual text files.

The following are examples taken from CCPD: refer to figure 3.1.



Figure 3.1 Examples from CCPD Dataset

The dataset used in this project is the Chinese City Parking Dataset (CCPD) (Xu *et al.*, 2018). This dataset consists of 341,978 total images where 100,000 are used for training, 99,996 are for validation and 141,982 are used for testing. Test images are grouped into `ccpd_blur` (20,611 images) – blurry images, `ccpd_db` (10,132 images) – dark bright images, `ccpd_fn` (20,967 images) – far near images, `ccpd_rotate` (10,053 images) – rotated images, `ccpd_tilt` (30,216 images) – tilted images, and `ccpd_challenge` (50,003 images) – combination/mixed of the above. Figure 3.2 shows all six categories of images.



Figure 3.2 Example Images from Six Categories

3.3 Hardware

A consumer-type desktop computer is used to run this project. The CPU used is AMD Ryzen 5 2600 3.40GHz 6 Cores 12 Threads. The RAM used is Kingston 32Gb 3200Mhz DDR4. The GPU used is Asus ROG Strix Nvidia Geforce Rtx2080ti 11gb ram. The storage disk used is Kingston 500Gb Solid State Drive. The power supply unit used is Fractal Design Ion+ Platinum 860 watt.

3.4 Software

The following softwares are used in the project:

- Ubuntu 20.04.5 LTS (Focal Fossa)
- Python 3.9
- LabelImg
- Darknet
- Yolomark
- TensorFlow 2.5.0
- Pytorch
- CUDA Toolkit 11.2
- CuDNN 8.1.0
- Anaconda
- OpenCV

3.5 Data Preparation

Raw training images and labels need to be converted into the corresponding framework following their file format.

3.5.1 YOLO Format

YOLO framework needs to specify several items for the training.

a.) obj.data – to specify several important parameters.

classes = 1	Specify the number of classes in the set.
train = train.txt	Specify the path to training samples.
valid = valid.txt	Specify the path to validating samples.
names = obj.names	Specify object classes label.
backup = backup	Specify the path to save the trained model.

b.) obj.names – to specify object label name.

c.) obj.cfg – a set of configs for the model, e.g., width, height, batches, etc.

After the parameters are set, training images and labels need to be separated into two standalone folders consisting of only images or labels. A list of file paths is generated using a bash script and saved as train.txt, valid.txt, and test.txt. Then, the label format for YOLO is different from TensorFlow. YOLO annotation format is as follows: <object_class> <x> <y> <width> <height> for each object in a different row. Example, 0 34 43 66 33. An automated Python script is used to convert CCPD object annotations into this format.

3.5.2 TensorFlow Format

For the TensorFlow framework, images and labeling need to be converted into .tfrecord file format. The advantages of tfrecords are That it can store data efficiently, fast I/O, and single-source data files (*Creating TFRecords*, 2021). These implementations allowed Google to take advantage of Tensor Processing Units (TPUs) on the cloud.

Data downloaded from CCPD encode their annotations as the filename, e.g.,

“025-95_113-154&383_386&473-386&473_177&454_154&383_363&402-
0_0_22_27_27_33_16-37-15.jpg”.

In this example, each attribute is separated by a ‘-’ sign, and each field represents different information. The first field encodes the area ratio of the license plate to the entire image, i.e., ‘025’. The second field encodes horizontal tilt degree and

vertical tilt degree, i.e., '95_113'. The third field encodes bounding-box coordinates of left-top and bottom-right, i.e., '154&383_386&473'. The fourth field encodes the exact location (x, y) of the four vertices from the image, starting from the bottom right, i.e., '386&473_177&454_154&383_363&402'. For the fifth field, there are seven characters in the Chinese license plate, i.e., one province, one alphabet, and five alphabets + digits. They were encoded in their corresponding index number i.e., '0_0_22_27_27_33_16'. The sixth field is the brightness, i.e., '37'. The seventh field is the blurriness, i.e., '15'.

These annotations are read using a Python script, then convert to tfrecord using TensorFlow API, such as:

- `tf.python_io.TFRecordWriter()` – for writing data.
- `tf.train.Example(features=tf.train.Features(feature={}))` – for converting variables into TensorFlow features.

3.6 Data Augmentation

Data augmentations are standard implementation for deep neural network training due to several reasons, e.g., to improve accuracy, to avoid overfitting, etc. In this project, model training follows the data augmentation strategy provided by the framework.

3.6.1 YOLOv4

YOLOv4 uses multiple data augmentation techniques in their model training, these techniques are categorized as Bag of Freebies (BoF) which means

they do not add detection time during the inference. YOLOv4 data augmentation techniques are as follows:

- Flip – flip training images left or right.
- Rotation – rotate the image 90, or 180 degrees clockwise or anticlockwise.
- Cutmix – cut a random part of an image and replace it in another image.
- Mosaic – combine multiple images into one.
- Mixup – stack images together with transparency.
- Blur – slightly blur the images.
- HSV – randomly slightly adjust the image’s Hue, Saturation, and Value.

It is important to mention that some of the augmentations, e.g., random flipping, rotation, are not appropriate and may reduce model accuracy. The figure 3.3 below shows example of random augmentation technique for YOLOv4.





Figure 3.3 Mosaic / Blur / Flip / Crop / Exposure / Aspect Ratio / Hue / Saturation

3.6.2 EfficientDet

EfficientDet uses Scale Jittering which resizes an image and crops it into a fixed size.

- Small jittering – uses a small ratio of [0.8, 1.2].
- Large jittering – uses a larger ratio of [0.1, 2.0].

The small jittering is good for shorter training time, i.e., 30 epochs. However, the accuracy decreases when using large jittering. For longer training time, i.e., 300 epochs, large jittering performs better (Tan et al., 2019).

3.7 Preprocessing

Preprocessing techniques are useful in solving many real-life problems, e.g., improving the visibility of a low-resolution image. This project explores the possibility of simple and low computation image preprocessing to improve object detection models' performance. Most of the preprocessing techniques used in this project are done through Python OpenCV and NumPy library. Figure 3.4 shows one example of image preprocessing to improve visibility.



Figure 3.4 Example of Image Preprocessing

The left image shows an original car with license plate taken in blurry condition. The right image shows the result after performing bilateralFilter with $d=9$, $\text{sigmaColor}=75$, $\text{sigmaSpace}=75$. The model was not able to detect the car license plate on the left however able to do so with 31% confidence, refer to figure 3.5.



Figure 3.5 Model Detection with 31% Confidence After Preprocessing

3.7.1 Enlargement

Enlargement is used to increase the resolution of an image. An image after being enlarged will have a greater number of pixels than the original image, though will not increase the visibility of the image because the algorithm simply copies the pixel color of the neighbor value. Hence, in the next stage image, preprocessing techniques such as image sharpening will be performed to hopefully make the image clearer to the human eye, as well as the computer vision algorithm. Adjustment to the size of the image depends on the scaling factor, 2.0 will enlarge the image by factor of 2, 0.75 will make the image smaller by 75%. In this project, the images are enlarged by 2.0.

3.7.2 Sharpening

Sharpening is an algorithm to make an image less blurry. This algorithm uses an $x*y$ kernel, also known as filters or convolution matrix, and loops through the dimensions of an image. Different values or sizes of the kernel will give different strengths of sharpening results.

- $$\text{kernel} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

3.7.3 Gamma Correction

Gamma correction is used to manipulate the brightness of an image.

$$O = \left(\frac{I}{255}\right)^{\frac{1}{\gamma}} * 255$$

Equation 3.1 Gamma Correction Equation

The gamma value γ is range from $0+$. If the value is between 0 and 1, the output image will be darkened. If the value is above 1, the image will be brightened. OpenCV provides an easy method to perform gamma correction using a lookup table (LUT).

3.7.4 Contrast Limited Adaptive Histogram Equalization (CLAHE)

Histogram equalization takes the pixel count of a low-contrast image and equalizes them evenly. However, it equalizes the whole image at a global scale. On the other hand, CLAHE applies equalization on smaller patches or tiles to obtain a higher-quality result. This is very useful in the test dataset due to dark, blurry images and small objects. The image needs to be first converted to grayscale or LAB color format.

3.7.5 Non-Local Means Denoising

Noise in the context of images is sometimes unavoidable, especially in image processing. Blurring techniques can remove some of the noise quite well. However, there is another approach which is a non-local means of denoising. This algorithm removes noise by finding a similar patch of pixel from another region in the same image.

3.8 Models

The models chosen to be implemented in this work are the state-of-the-art model in recent years which are very efficient and have high accuracy.

- EfficientDet (AutoML)
- YOLOv4 (Darknet)
- CenterNet (TensorFlow)
- SSD (TensorFlow)
- Faster R-CNN (TensorFlow)
- YOLOv5 (Pytorch)

Among the chosen models, four frameworks have been used. They are Darknet framework in C++ for YOLOv4, Automl in Tensorflow for EfficientDet, Tensorflow Object Detection API for Faster R-CNN, SSD, and CenterNet, and Pytorch for YOLOv5.

3.9 Updated Model Changes

The model YOLOv4 will be selected to improve its performance due to several reasons, i.e., fast detection speed, robustness to modification in DarkNet framework, etc.

YOLOv4 detects objects in three different scales, i.e., small, medium, and large. As stated in the paper, this helps to detect objects of different sizes. For this project, two identical layers have been added to each scale before detection layers, making it a total of six additional convolution layers added to the whole architecture. By doing this, the hypothesis is that it helps the network to learn more features regard to the new CCPD dataset that the model may have never seen before.

The initial weights of the six newly added layers were initialized randomly. This results in longer total training time, and this will be further discussed in chapter 4.

3.10 Evaluation Method

Evaluation of the accuracy of the models typically follows standard MS COCO object detection metrics ($mAP@0.5:0.05:0.95$) which stands for the mean Average Precision for IoU value ranges from 0.5 to 0.95 with 0.05 increment, averaged. For instance, this will calculate the ($mAP@0.5 + mAP@0.55 + mAP@0.60 + mAP@0.65 + mAP@0.70 + mAP@0.75 + mAP@0.80 + mAP@0.85 + mAP@0.90 + mAP@0.95$) / 10. However, since this project only consists of one object class category, i.e., license plate, therefore Average Precision (AP) is set to be equivalent to mean Average Precision (mAP). Other than that, only $mAP@0.70$ is used as the evaluation metric to allow the comparison of result one-to-one with the result from CCPD (Xu *et al.*, 2018), which is also using $mAP@0.70$.

In object detection, Intersection over Union (IoU) stands for the overlapped percentage of the predicted bounding-box generated by the model vs. then ground truth bounding-box labeled by human. Refer to figure 3.6 Example of GT Box and Prediction Box.



Figure 3.6 Example of GT Box and Prediction Box

In this example, the green color bounding-box is first labeled on the car by human. When the image is passed through an object detection model, it generates the coordinates output of the car object on the image hence a model predicted bounding box is produced. To calculate the IoU, the overlapped area is determined. See figure 3.7.

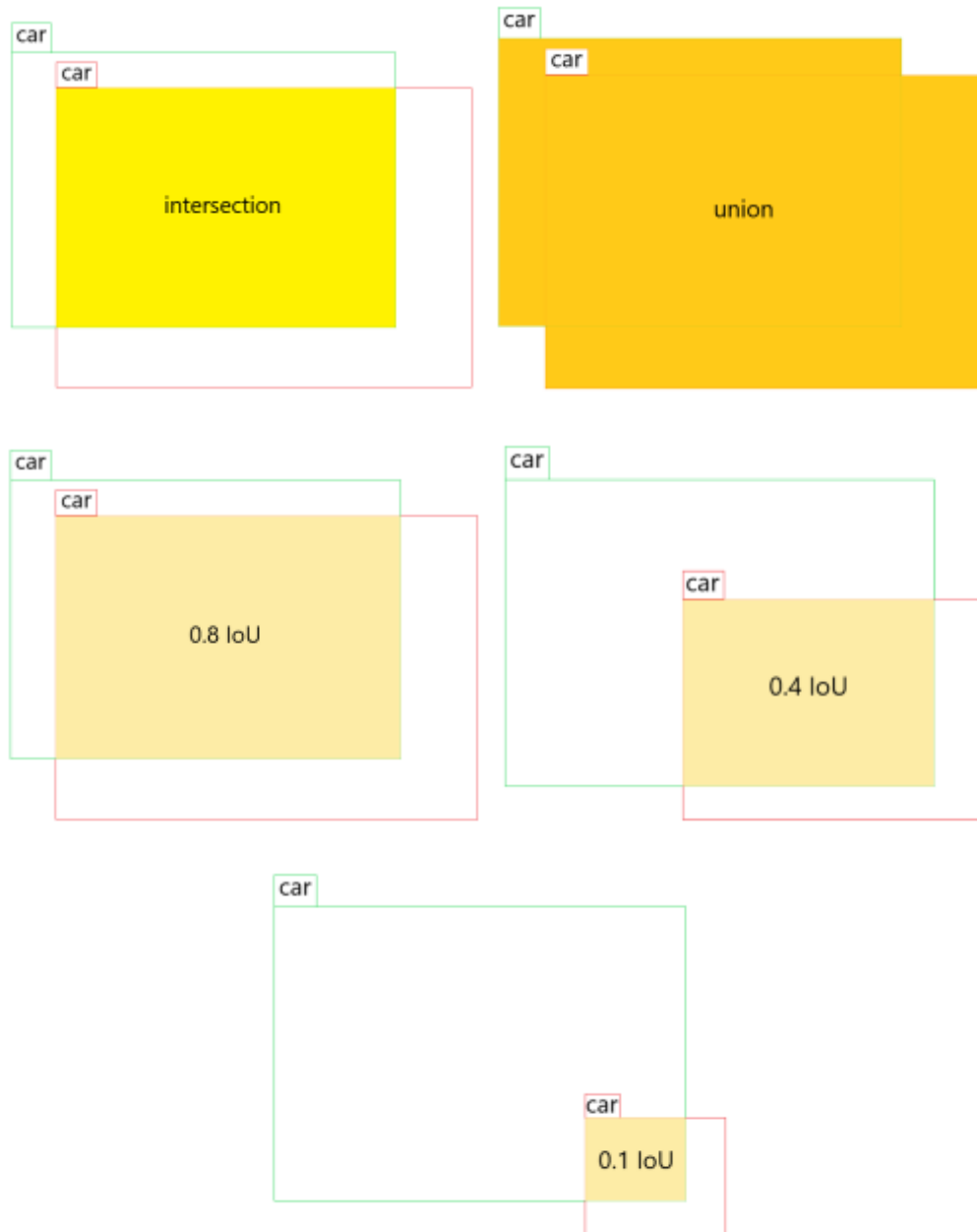


Figure 3.7 Intersection over Union

In this project, in the case of license plate bounding box detection (one object class scenario), TP, FP, TN, and FN are defined as follows:

- TP – when the prediction bounding box has $\geq 70\%$ IoU with the ground truth box.
- FP – when the prediction box has $< 70\%$ IoU with the ground truth box.
- FN – when ground truth exists in the image but there is no bounding box predicted by the model.

- TN case is not applicable for this project.

3.10.1 Steps to Calculate mAP@0.70

Below presents simplified steps to calculate mAP@0.70. First, set a confidence score threshold, i.e., only account for prediction above this value. In this project, a 0.4 threshold is empirically selected.

1. Is confidence \geq threshold?
2. Is predicted box IoU \geq 0.70? If yes, TP. If no, FP.
3. Calculate Precision and Recall.
4. Plot graph Precision vs Recall.
5. Calculate the Area Under Curve.

3.10.2 Performance Metrics

Following performance metrics are used in this work.

mAP:

$$\frac{AP_{car} + AP_{cat} + AP_{dog}}{3}$$

Precision:

$$\frac{TP}{TP + FP}$$

Recall:

$$\frac{TP}{TP + FN}$$

F1 Score:

$$2 * \frac{precision * recall}{precision + recall}$$

IoU:

$$\frac{\text{Intersection Area}}{\text{Union Area}}$$

3.11 Detailed Diagram / Chart of Proposed Model

The following table 3.1 and table 3.2 show the overall architecture of model YOLOv4-CSP and YOLOv4-CSP-Modified. Note that the 6 newly added layers in the modified model, namely 143conv, 144conv, 160conv, 161conv, and 177conv, 178conv. In the end, the modified model is 180 layers long, compared to the author's model which is 174 layers.

Table 3.1 YOLOv4-CSP Architecture

layer	filters	Size/strd(dil)	input	output	bflops
0conv	32	3x3/1	640x640x3	640x640x32	0.708BF
1conv	64	3x3/2	640x640x32	320x320x64	3.775BF
2conv	32	1x1/1	320x320x64	320x320x32	0.419BF
3conv	64	3x3/1	320x320x32	320x320x64	3.775BF
4Shortcut	1	wt=0	wn=0	320x320x64	0.007BF
5conv	128	3x3/2	320x320x64	160x160x128	3.775BF
6conv	64	1x1/1	160x160x128	160x160x64	0.419BF
7route	5			160x160x128	
8conv	64	1x1/1	160x160x128	160x160x64	0.419BF
9conv	64	1x1/1	160x160x64	160x160x64	0.210BF
10conv	64	3x3/1	160x160x64	160x160x64	1.887BF
11shortcut	8	wt=0	wn=0	160x160x64	0.002BF
12conv	64	1x1/1	160x160x64	160x160x64	0.210BF
13conv	64	3x3/1	160x160x64	160x160x64	1.887BF
14shortcut	11	c	wn=0	160x160x64	0.002BF
15conv	64	1x1/1	160x160x64	160x160x64	0.210BF
16route	15 6			160x160x128	
17conv	128	1x1/1	160x160x128	160x160x128	0.839BF
18conv	256	3x3/2	160x160x128	80x80x256	3.775BF
19conv	128	1x1/1	80x80x256	80x80x128	0.419BF
20route	18			80x80x256	
21conv	128	1x1/1	80x80x256	80x80x128	0.419BF
22conv	128	1x1/1	80x80x128	80x80x128	0.210BF
23conv	128	3x3/1	80x80x128	80x80x128	1.887BF
24shortcut	21	wt=0	wn=0	80x80x128	0.001BF
25conv	128	1x1/1	80x80x128	80x80x128	0.210BF
26conv	128	3x3/1	80x80x128	80x80x128	1.887BF
27shortcut	24	wt=0	wn=0	80x80x128	0.001BF
28conv	128	1x1/1	80x80x128	80x80x128	0.210BF
29conv	128	3x3/1	80x80x128	80x80x128	1.887BF

30shortcut	27	wt=0	wn=0	80x80x128	0.001BF
31conv	128	1x1/1	80x80x128	80x80x128	0.210BF
32conv	128	3x3/1	80x80x128	80x80x128	1.887BF
33shortcut	30	wt=0	wn=0	80x80x128	0.001BF
34conv	128	1x1/1	80x80x128	80x80x128	0.210BF
35conv	128	3x3/1	80x80x128	80x80x128	1.887BF
36shortcut	33	wt=0	wn=0	80x80x128	0.001BF
37conv	128	1x1/1	80x80x128	80x80x128	0.210BF
38conv	128	3x3/1	80x80x128	80x80x128	1.887BF
39shortcut	36	wt=0	wn=0	80x80x128	0.001BF
40conv	128	1x1/1	80x80x128	80x80x128	0.210BF
41conv	128	3x3/1	80x80x128	80x80x128	1.887BF
42shortcut	39	wt=0	wn=0	80x80x128	0.001BF
43conv	128	1x1/1	80x80x128	80x80x128	0.210BF
44conv	128	3x3/1	80x80x128	80x80x128	1.887BF
45shortcut	42	wt=0	wn=0	80x80x128	0.001BF
46conv	128	1x1/1	80x80x128	80x80x128	0.210BF
47route	46 19			80x80x256	
48conv	256	1x1/1	80x80x256	80x80x256	0.839BF
49conv	512	3x3/2	80x80x256	40x40x512	3.775BF
50conv	256	1x1/1	40x40x512	40x40x256	0.419BF
51route	49			40x40x512	
52conv	256	1x1/1	40x40x512	40x40x256	0.419BF
53conv	256	1x1/1	40x40x256	40x40x256	0.210BF
54conv	256	3x3/1	40x40x256	40x40x256	1.887BF
55shortcut	52	wt=0	wn=0	40x40x256	0.000BF
56conv	256	1x1/1	40x40x256	40x40x256	0.210BF
57conv	256	3x3/1	40x40x256	40x40x256	1.887BF
58shortcut	55	wt=0	wn=0	40x40x256	0.000BF
59conv	256	1x1/1	40x40x256	40x40x256	0.210BF
60conv	256	3x3/1	40x40x256	40x40x256	1.887BF
61shortcut	58	wt=0	wn=0	40x40x256	0.000BF
62conv	256	1x1/1	40x40x256	40x40x256	0.210BF
63conv	256	3x3/1	40x40x256	40x40x256	1.887BF
64shortcut	61	wt=0	wn=0	40x40x256	0.000BF
65conv	256	1x1/1	40x40x256	40x40x256	0.210BF
66conv	256	3x3/1	40x40x256	40x40x256	1.887BF
67shortcut	64	wt=0	wn=0	40x40x256	0.000BF
68conv	256	1x1/1	40x40x256	40x40x256	0.210BF
69conv	256	3x3/1	40x40x256	40x40x256	1.887BF
70shortcut	67	wt=0	wn=0	40x40x256	0.000BF
71conv	256	1x1/1	40x40x256	40x40x256	0.210BF
72conv	256	3x3/1	40x40x256	40x40x256	1.887BF
73shortcut	70	wt=0	wn=0	40x40x256	0.000BF
74conv	256	1x1/1	40x40x256	40x40x256	0.210BF
75conv	256	3x3/1	40x40x256	40x40x256	1.887BF
76shortcut	73	wt=0	wn=0	40x40x256	0.000BF
77conv	256	1x1/1	40x40x256	40x40x256	0.210BF
78route	77 50			40x40x512	
79conv	512	1x1/1	40x40x512	40x40x512	0.839BF
80conv	1024	3x3/2	40x40x512	20x20x1024	3.775BF
81conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
82route	80			20x20x1024	

83conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
84conv	512	1x1/1	20x20x512	20x20x512	0.210BF
85conv	512	3x3/1	20x20x512	20x20x512	1.887BF
86shortcut	83	wt=0	wn=0	20x20x512	0.000BF
87conv	512	1x1/1	20x20x512	20x20x512	0.210BF
88conv	512	3x3/1	20x20x512	20x20x512	1.887BF
89shortcut	86	wt=0	wn=0	20x20x512	0.000BF
90conv	512	1x1/1	20x20x512	20x20x512	0.210BF
91conv	512	3x3/1	20x20x512	20x20x512	1.887BF
92shortcut	89	wt=0	wn=0	20x20x512	0.000BF
93conv	512	1x1/1	20x20x512	20x20x512	0.210BF
94conv	512	3x3/1	20x20x512	20x20x512	1.887BF
95shortcut	92	wt=0	wn=0	20x20x512	0.000BF
96conv	512	1x1/1	20x20x512	20x20x512	0.210BF
97route	96 81			20x20x1024	
98conv	1024	1x1/1	20x20x1024	20x20x1024	0.839BF
99conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
100route	98			20x20x1024	
101conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
102conv	512	3x3/1	20x20x512	20x20x512	1.887BF
103conv	512	1x1/1	20x20x512	20x20x512	0.210BF
104max		5x5/1	20x20x512	20x20x512	0.005BF
105route	103			20x20x512	
106max		9x9/1	20x20x512	20x20x512	0.017BF
107route	103			20x20x512	
108max		13x13/1	20x20x512	20x20x512	0.035BF
109route	108 106	104 103		20x20x2048	
110conv	512	1x1/1	20x20x2048	20x20x512	0.839BF
111conv	512	3x3/1	20x20x512	20x20x512	1.887BF
112route	111 99			20x20x1024	
113conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
114conv	256	1x1/1	20x20x512	20x20x256	0.105BF
115upsam- ple	2x		20x20x256	40x40x256	
116route	79			40x40x512	
117conv	256	1x1/1	40x40x512	40x40x256	0.419BF
118route	117 115			40x40x512	
119conv	256	1x1/1	40x40x512	40x40x256	0.419BF
120conv	256	1x1/1	40x40x256	40x40x256	0.210BF
121route	119			40x40x256	
122conv	256	1x1/1	40x40x256	40x40x256	0.210BF
123conv	256	3x3/1	40x40x256	40x40x256	1.887BF
124conv	256	1x1/1	40x40x256	40x40x256	0.210BF
125conv	256	3x3/1	40x40x256	40x40x256	1.887BF
126route	125 120			40x40x512	
127conv	256	1x1/1	40x40x512	40x40x256	0.419BF
128conv	128	1x1/1	40x40x256	40x40x128	0.105BF
129upsam- ple	2x		40x40x128	80x80x128	
130route	48			80x80x256	
131conv	128	1x1/1	80x80x256	80x80x128	0.419BF
132route	131 129			80x80x256	
133conv	128	1x1/1	80x80x256	80x80x128	0.419BF

134conv	128	1x1/1	80x80x128	80x80x128	0.210BF
135route	133			80x80x128	
136conv	128	1x1/1	80x80x128	80x80x128	0.210BF
137conv	128	3x3/1	80x80x128	80x80x128	1.887BF
138conv	128	1x1/1	80x80x128	80x80x128	0.210BF
139conv	128	3x3/1	80x80x128	80x80x128	1.887BF
140route	139 134			80x80x256	
141conv	128	1x1/1	80x80x256	80x80x128	0.419BF
142conv	256	3x3/1	80x80x128	80x80x256	3.775BF
143conv	18	1x1/1	80x80x256	80x80x18	0.059BF
144yolo					
145route	141				
146conv	256	3x3/2	80x80x128	40x40x256	0.944BF
147route	146 127			40x40x512	
148conv	256	1x1/1	40x40x512	40x40x256	0.419BF
149conv	256	1x1/1	40x40x256	40x40x256	0.210BF
150route	148			40x40x256	
151conv	256	1x1/1	40x40x256	40x40x256	0.210BF
152conv	256	3x3/1	40x40x256	40x40x256	1.887BF
153conv	256	1x1/1	40x40x256	40x40x256	0.210BF
154conv	256	3x3/1	40x40x256	40x40x256	1.887BF
155route	154 149			40x40x512	
156conv	256	1x1/1	40x40x512	40x40x256	0.419BF
157conv	512	3x3/1	40x40x256	40x40x512	3.775BF
158conv	18	1x1/1	40x40x512	40x40x18	0.029BF
159yolo					
160route	156				
161conv	512	3x3/2	40x40x256	20x20x512	0.944BF
162route	161 113			20x20x1024	
163conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
164conv	512	1x1/1	20x20x512	20x20x512	0.210BF
165route	163			20x20x512	
166conv	512	1x1/1	20x20x512	20x20x512	0.210BF
167conv	512	3x3/1	20x20x512	20x20x512	1.887BF
168conv	512	1x1/1	20x20x512	20x20x512	0.210BF
169conv	512	3x3/1	20x20x512	20x20x512	1.887BF
170route	169 164			20x20x1024	
171conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
172conv	1024	3x3/1	20x20x512	20x20x1024	3.775BF
173conv	18	1x1/1	20x20x1024	20x20x18	0.015BF
174yolo					

Table 3.2 YOLOv4-CSP-Modified Architecture

layer	filters	Size/strd(dil)	input	output	bflops
0conv	32	3x3/1	640x640x3	640x640x32	0.708BF
1conv	64	3x3/2	640x640x32	320x320x64	3.775BF
2conv	32	1x1/1	320x320x64	320x320x32	0.419BF
3conv	64	3x3/1	320x320x32	320x320x64	3.775BF

4Shortcut	1	wt=0	wn=0	320x320x64	0.007BF
5conv	128	3x3/2	320x320x64	160x160x128	3.775BF
6conv	64	1x1/1	160x160x128	160x160x64	0.419BF
7route	5			160x160x128	
8conv	64	1x1/1	160x160x128	160x160x64	0.419BF
9conv	64	1x1/1	160x160x64	160x160x64	0.210BF
10conv	64	3x3/1	160x160x64	160x160x64	1.887BF
11shortcut	8	wt=0	wn=0	160x160x64	0.002BF
12conv	64	1x1/1	160x160x64	160x160x64	0.210BF
13conv	64	3x3/1	160x160x64	160x160x64	1.887BF
14shortcut	11	c	wn=0	160x160x64	0.002BF
15conv	64	1x1/1	160x160x64	160x160x64	0.210BF
16route	15 6			160x160x128	
17conv	128	1x1/1	160x160x128	160x160x128	0.839BF
18conv	256	3x3/2	160x160x128	80x80x256	3.775BF
19conv	128	1x1/1	80x80x256	80x80x128	0.419BF
20route	18			80x80x256	
21conv	128	1x1/1	80x80x256	80x80x128	0.419BF
22conv	128	1x1/1	80x80x128	80x80x128	0.210BF
23conv	128	3x3/1	80x80x128	80x80x128	1.887BF
24shortcut	21	wt=0	wn=0	80x80x128	0.001BF
25conv	128	1x1/1	80x80x128	80x80x128	0.210BF
26conv	128	3x3/1	80x80x128	80x80x128	1.887BF
27shortcut	24	wt=0	wn=0	80x80x128	0.001BF
28conv	128	1x1/1	80x80x128	80x80x128	0.210BF
29conv	128	3x3/1	80x80x128	80x80x128	1.887BF
30shortcut	27	wt=0	wn=0	80x80x128	0.001BF
31conv	128	1x1/1	80x80x128	80x80x128	0.210BF
32conv	128	3x3/1	80x80x128	80x80x128	1.887BF
33shortcut	30	wt=0	wn=0	80x80x128	0.001BF
34conv	128	1x1/1	80x80x128	80x80x128	0.210BF
35conv	128	3x3/1	80x80x128	80x80x128	1.887BF
36shortcut	33	wt=0	wn=0	80x80x128	0.001BF
37conv	128	1x1/1	80x80x128	80x80x128	0.210BF
38conv	128	3x3/1	80x80x128	80x80x128	1.887BF
39shortcut	36	wt=0	wn=0	80x80x128	0.001BF
40conv	128	1x1/1	80x80x128	80x80x128	0.210BF
41conv	128	3x3/1	80x80x128	80x80x128	1.887BF
42shortcut	39	wt=0	wn=0	80x80x128	0.001BF
43conv	128	1x1/1	80x80x128	80x80x128	0.210BF
44conv	128	3x3/1	80x80x128	80x80x128	1.887BF
45shortcut	42	wt=0	wn=0	80x80x128	0.001BF
46conv	128	1x1/1	80x80x128	80x80x128	0.210BF
47route	46 19			80x80x256	
48conv	256	1x1/1	80x80x256	80x80x256	0.839BF
49conv	512	3x3/2	80x80x256	40x40x512	3.775BF
50conv	256	1x1/1	40x40x512	40x40x256	0.419BF
51route	49			40x40x512	
52conv	256	1x1/1	40x40x512	40x40x256	0.419BF
53conv	256	1x1/1	40x40x256	40x40x256	0.210BF
54conv	256	3x3/1	40x40x256	40x40x256	1.887BF
55shortcut	52	wt=0	wn=0	40x40x256	0.000BF
56conv	256	1x1/1	40x40x256	40x40x256	0.210BF

57conv	256	3x3/1	40x40x256	40x40x256	1.887BF
58shortcut	55	wt=0	wn=0	40x40x256	0.000BF
59conv	256	1x1/1	40x40x256	40x40x256	0.210BF
60conv	256	3x3/1	40x40x256	40x40x256	1.887BF
61shortcut	58	wt=0	wn=0	40x40x256	0.000BF
62conv	256	1x1/1	40x40x256	40x40x256	0.210BF
63conv	256	3x3/1	40x40x256	40x40x256	1.887BF
64shortcut	61	wt=0	wn=0	40x40x256	0.000BF
65conv	256	1x1/1	40x40x256	40x40x256	0.210BF
66conv	256	3x3/1	40x40x256	40x40x256	1.887BF
67shortcut	64	wt=0	wn=0	40x40x256	0.000BF
68conv	256	1x1/1	40x40x256	40x40x256	0.210BF
69conv	256	3x3/1	40x40x256	40x40x256	1.887BF
70shortcut	67	wt=0	wn=0	40x40x256	0.000BF
71conv	256	1x1/1	40x40x256	40x40x256	0.210BF
72conv	256	3x3/1	40x40x256	40x40x256	1.887BF
73shortcut	70	wt=0	wn=0	40x40x256	0.000BF
74conv	256	1x1/1	40x40x256	40x40x256	0.210BF
75conv	256	3x3/1	40x40x256	40x40x256	1.887BF
76shortcut	73	wt=0	wn=0	40x40x256	0.000BF
77conv	256	1x1/1	40x40x256	40x40x256	0.210BF
78route	77 50			40x40x512	
79conv	512	1x1/1	40x40x512	40x40x512	0.839BF
80conv	1024	3x3/2	40x40x512	20x20x1024	3.775BF
81conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
82route	80			20x20x1024	
83conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
84conv	512	1x1/1	20x20x512	20x20x512	0.210BF
85conv	512	3x3/1	20x20x512	20x20x512	1.887BF
86shortcut	83	wt=0	wn=0	20x20x512	0.000BF
87conv	512	1x1/1	20x20x512	20x20x512	0.210BF
88conv	512	3x3/1	20x20x512	20x20x512	1.887BF
89shortcut	86	wt=0	wn=0	20x20x512	0.000BF
90conv	512	1x1/1	20x20x512	20x20x512	0.210BF
91conv	512	3x3/1	20x20x512	20x20x512	1.887BF
92shortcut	89	wt=0	wn=0	20x20x512	0.000BF
93conv	512	1x1/1	20x20x512	20x20x512	0.210BF
94conv	512	3x3/1	20x20x512	20x20x512	1.887BF
95shortcut	92	wt=0	wn=0	20x20x512	0.000BF
96conv	512	1x1/1	20x20x512	20x20x512	0.210BF
97route	96 81			20x20x1024	
98conv	1024	1x1/1	20x20x1024	20x20x1024	0.839BF
99conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
100route	98			20x20x1024	
101conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
102conv	512	3x3/1	20x20x512	20x20x512	1.887BF
103conv	512	1x1/1	20x20x512	20x20x512	0.210BF
104max		5x5/1	20x20x512	20x20x512	0.005BF
105route	103			20x20x512	
106max		9x9/1	20x20x512	20x20x512	0.017BF
107route	103			20x20x512	
108max		13x13/1	20x20x512	20x20x512	0.035BF
109route	108 106	104 103		20x20x2048	

110conv	512	1x1/1	20x20x2048	20x20x512	0.839BF
111conv	512	3x3/1	20x20x512	20x20x512	1.887BF
112route	111 99			20x20x1024	
113conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
114conv	256	1x1/1	20x20x512	20x20x256	0.105BF
115up-sample	2x		20x20x256	40x40x256	
116route	79			40x40x512	
117conv	256	1x1/1	40x40x512	40x40x256	0.419BF
118route	117 115			40x40x512	
119conv	256	1x1/1	40x40x512	40x40x256	0.419BF
120conv	256	1x1/1	40x40x256	40x40x256	0.210BF
121route	119			40x40x256	
122conv	256	1x1/1	40x40x256	40x40x256	0.210BF
123conv	256	3x3/1	40x40x256	40x40x256	1.887BF
124conv	256	1x1/1	40x40x256	40x40x256	0.210BF
125conv	256	3x3/1	40x40x256	40x40x256	1.887BF
126route	125 120			40x40x512	
127conv	256	1x1/1	40x40x512	40x40x256	0.419BF
128conv	128	1x1/1	40x40x256	40x40x128	0.105BF
129up-sample	2x		40x40x128	80x80x128	
130route	48			80x80x256	
131conv	128	1x1/1	80x80x256	80x80x128	0.419BF
132route	131 129			80x80x256	
133conv	128	1x1/1	80x80x256	80x80x128	0.419BF
134conv	128	1x1/1	80x80x128	80x80x128	0.210BF
135route	133			80x80x128	
136conv	128	1x1/1	80x80x128	80x80x128	0.210BF
137conv	128	3x3/1	80x80x128	80x80x128	1.887BF
138conv	128	1x1/1	80x80x128	80x80x128	0.210BF
139conv	128	3x3/1	80x80x128	80x80x128	1.887BF
140route	139 134			80x80x256	
141conv	128	1x1/1	80x80x256	80x80x128	0.419BF
142conv	256	3x3/1	80x80x128	80x80x256	3.775BF
143conv	128	1x1/1	80x80x256	80x80x128	0.419BF
144conv	256	3x3/1	80x80x128	80x80x256	3.775BF
145conv	18	1x1/1	80x80x256	80x80x18	0.059BF
146yolo					
147route	141				
148conv	256	3x3/2	80x80x128	40x40x256	0.944BF
149route	148 127			40x40x512	
150conv	256	1x1/1	40x40x512	40x40x256	0.419BF
151conv	256	1x1/1	40x40x256	40x40x256	0.210BF
152route	150			40x40x256	
153conv	256	1x1/1	40x40x256	40x40x256	0.210BF
154conv	256	3x3/1	40x40x256	40x40x256	1.887BF
155conv	256	1x1/1	40x40x256	40x40x256	0.210BF
156conv	256	3x3/1	40x40x256	40x40x256	1.887BF
157route	156 151			40x40x512	
158conv	256	1x1/1	40x40x512	40x40x256	0.419BF
159conv	512	3x3/1	40x40x256	40x40x512	3.775BF
160conv	256	1x1/1	40x40x512	40x40x256	0.419BF

161conv	512	3x3/1	40x40x256	40x40x512	3.775BF
162conv	18	1x1/1	40x40x512	40x40x18	0.029BF
163yolo					
164route	160				
165conv	512	3x3/2	40x40x256	20x20x512	0.944BF
166route	165 111			20x20x1024	
167conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
168conv	512	1x1/1	20x20x512	20x20x512	0.210BF
169route	167			20x20x512	
170conv	512	1x1/1	20x20x512	20x20x512	0.210BF
171conv	512	3x3/1	20x20x512	20x20x512	1.887BF
172conv	512	1x1/1	20x20x512	20x20x512	0.210BF
173conv	512	3x3/1	20x20x512	20x20x512	1.887BF
174route	173 168			20x20x1024	
175conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
176conv	1024	3x3/1	20x20x512	20x20x1024	3.775BF
177conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
178conv	1024	3x3/1	20x20x512	20x20x1024	3.775BF
179conv	18	1x1/1	20x20x1024	20x20x18	0.015BF
180yolo					

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Overview

This section discusses the results of models' accuracy and performance on the car license plate dataset CCPD as mentioned in chapter 3, i.e., EfficientDet-d6, EfficientDet-d0, CenterNetResNet50-512x512, SSDResnet50-640x640, FasterR-CNNResNet50-640x640, YOLOv4-csp-640x640, YOLOv5x, and YOLOv5s. This chapter will then continue with presenting the effect of image preprocessing on bad images to increase the overall accuracy of the models. Finally, a slightly improved model architecture is then trained on the same CCPD dataset to show the possible improvement that can be achieved through model architecture modifications.

In addition to the above results, this section also addresses a small issue regarding the calculation of 70% IoU, i.e., the 69% IoU problem where there are a significant number of objects bounding boxes being rejected as false negatives despite being detected by the model hence lowered the overall accuracy.

4.2 Model Training Loss

The following models were trained on CCPD training dataset, i.e., EfficientDet-d6, EfficientDet-d0, CenterNetResNet50-512x512, SSDResnet50-640x640, FasterR-CNNResNet50-640x640, YOLOv4-csp-640x640, YOLOv5x, and YOLOv5s. Below shows their training loss graph respectively.

4.2.1 EfficientDet-D6

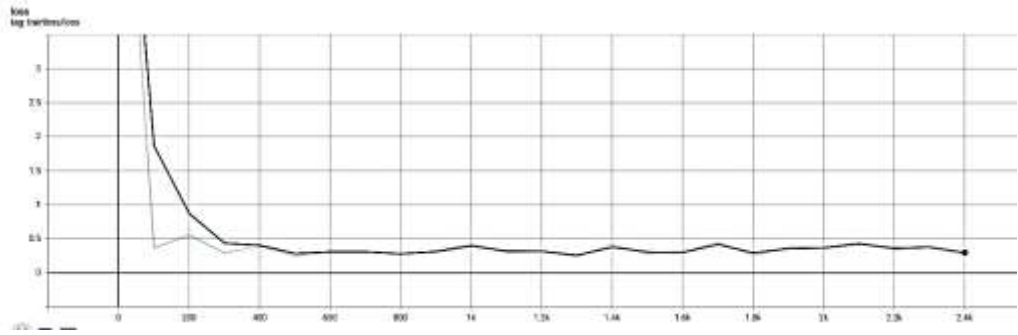


Figure 4.1 EfficientDet-D6 Training Loss

4.2.2 EfficientDet-D0

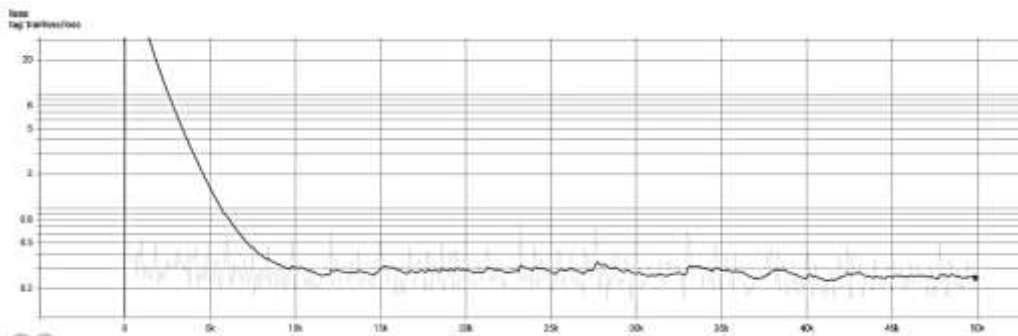


Figure 4.2 EfficientDet-D0 Training Loss

4.2.3 CenterNetResNet50-512x512

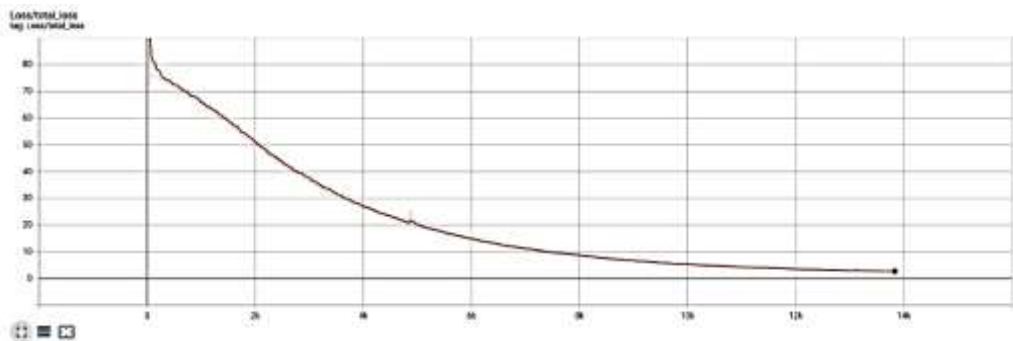


Figure 4.3 CenterNetResNet50-512x512 Training Loss

4.2.4 SSDResNet50-640x640

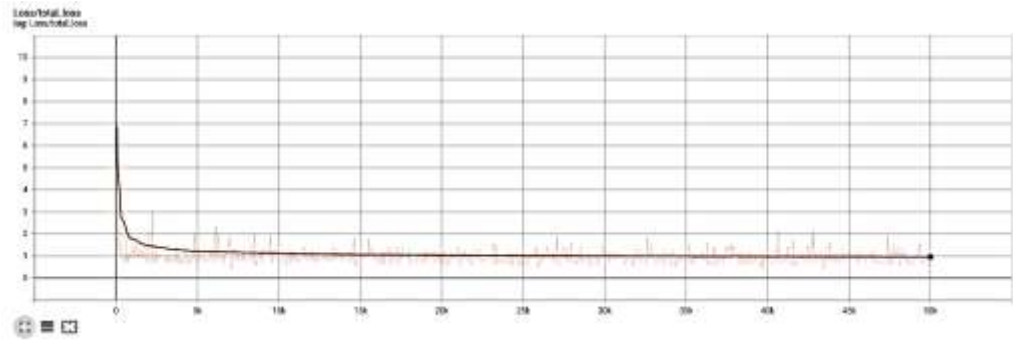


Figure 4.4 SSDResNet50-640x640 Training Loss

4.2.5 FasterR-CNNResNet50-640x640

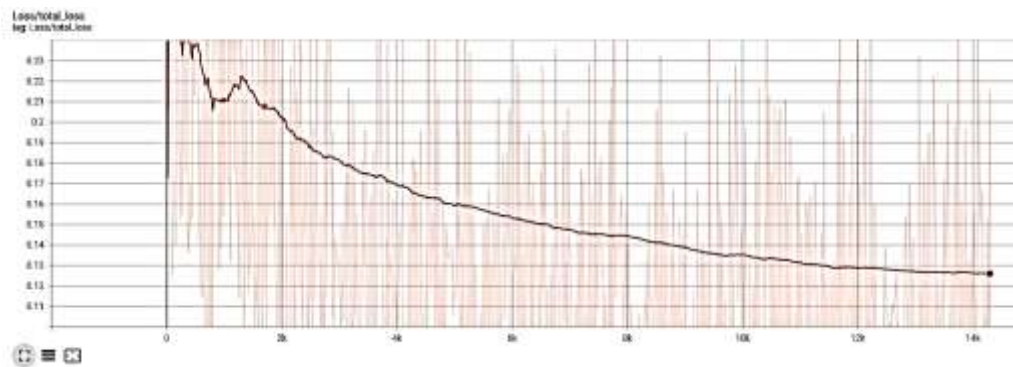


Figure 45 FasterR-CNNResNet50 Training Loss

4.3 Result of Models Training

A series of steps of training the chosen models and evaluating them has been performed. Below shows the results.

Table 4.1: Initial Models' Test Accuracy (mAP@0.70) and Speed (FPS) on Each Test Set

Model	FPS	DB	Blur	FN	Rotate	Tilt	Challenge	Average
efficientdet-d6	8.2	67.0	79.25	80.5	92.1	85.0	89.1	82.16
efficientdet-d0	34.53	53.5	70.6	62.9	91.0	86.3	78.3	73.77

centernet-resnet50-512x512	28.84	50.14	64.48	70.20	90.59	74.40	70.87	70.11
ssd-resnet50-640x640	24.85	45.24	64.06	47.28	92.46	83.79	72.72	67.59
faster-rcnn-resnet50-640x640	18.73	52.31	68.82	61.74	91.62	76.32	80.85	71.94
yolov4-csp-640x640	50.9	64.54	67.58	29.46	81.75	65.76	66.41	62.58
yolov5s	80.0	50.5	74.0	11.5	75.8	51.6	83.1	57.75
yolov5x	40.0	44.4	68.2	10.7	56.3	35.0	79.4	49.00

From this table, the best score in terms of average mAP is achieved by the model EfficientDet-d6, but it is a network larger than the others in terms of parameters and FLOPs (d0 - 3.9M, 2.54B) v.s. (d6 - 51.9M, 325B) (Tan et. al., 2019), therefore is an unfair comparison.

Mean Average Precision of Different Models (%)

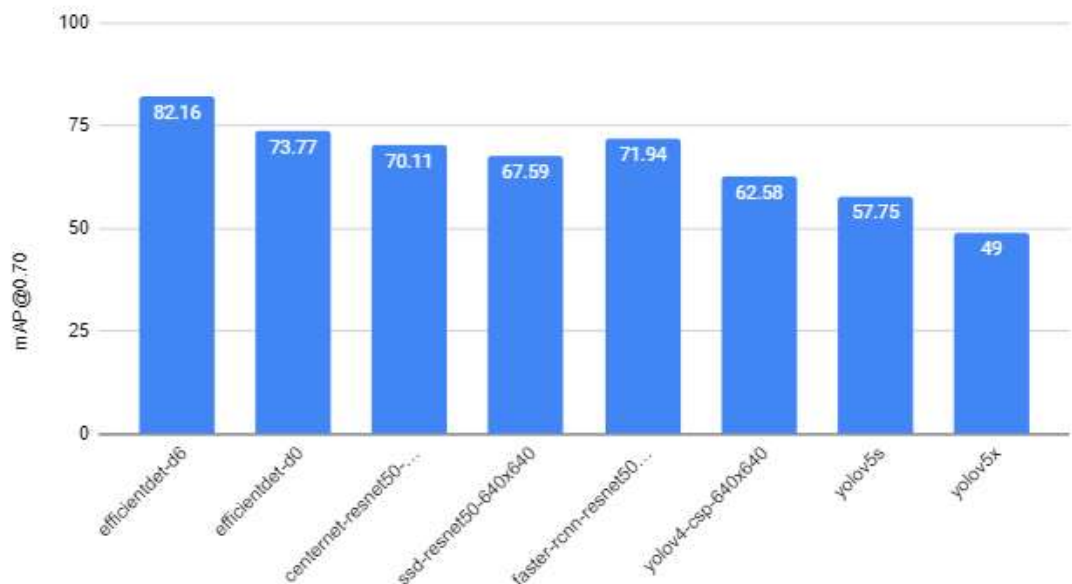


Figure 4.6 Mean Average Precision for All Models for CCPD

Dataset

Speed of Different Models (FPS)

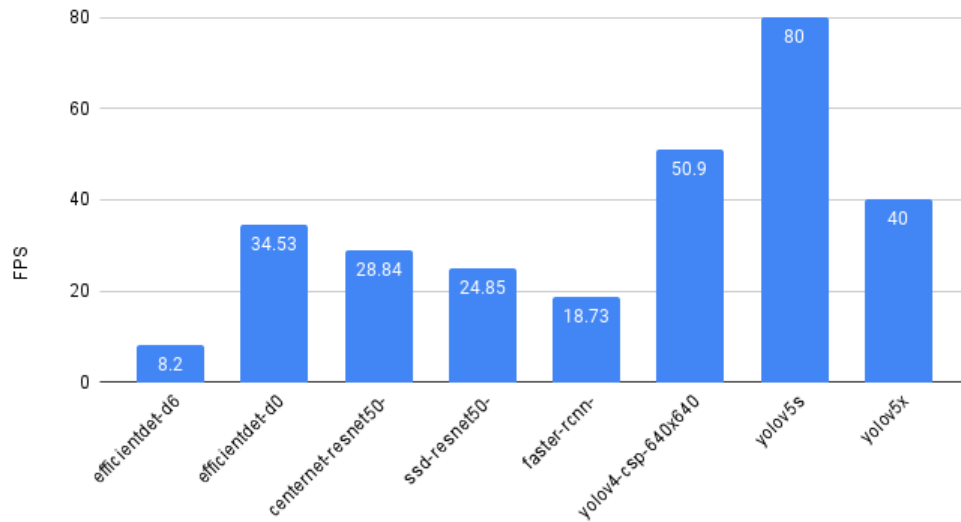


Figure 4.7 Speed for All Models for CCPD Dataset

It has been observed from Table 4.1 that Efficientdet-d6 (82.16) scores highest followed by Efficientdet-d0 (73.77). However, the FPS has dropped tremendously from 34.53 FPS to only 8.2 FPS which is not a very good model to be perform in real-time applications. Comparing both models, Efficientdet-d6 outperforms Efficientdet-d0 in all test dataset (DB – 67.0 vs. 53.5), (Blur – 79.25 vs. 70.6), (FN – 80.5 vs. 62.9), (Rotate – 92.1 vs. 91.0), and (Challenge – 89.1 vs. 78.3) dataset, except for (Tilt – 85.0 vs. 86.3) dataset.

If Efficientdet-d6 is not included, then the highest scores will be Efficientdet-d0 (73.77) followed by Faster R-CNN (71.94). Comparing the processing speed between these two models, Efficientdet-d0 performs faster (34.53) than Faster R-CNN (18.73). Comparing the accuracy between Efficientdet-d0 and Faster R-CNN, Efficientdet-d0 performs better in all test sets (DB – 53.5 vs. 52.31), (Blur – 70.6 vs. 68.82), (FN – 62.9 vs. 61.74), (Tilt – 86.3

vs. 76.32) except for the , (Rotate – 91.0 vs. 91.62) dataset and (Challenge – 78.3 vs. 80.85) dataset.

YOLOv4-CSP model performed very badly in the FN dataset (29.46) compared to the best model in the category Centernet-Resnet50 (70.20). This shows that Centernet-Resnet50 is very good at detecting very small objects in the image whilst YOLOv4-CSP is having difficulty to detect small objects. The other model that performed like YOLOv4-CSP (29.46) is SSD-Resnet50 (47.28). More analysis and improvement will be carried out on the FN dataset and the result will be discussed later in this chapter.

Another highlight worth mentioning is that in the dark_bright dataset, YOLOv4-CSP performs better (64.54) than other models (45.24 ~ 52.31). This shows YOLOv4-CSP is less sensitive to the brightness of the image instead the other features e.g., shape allow it to detect an object at a higher chance.

Regarding lower scores achieved by YOLOv5, this model is still under development, and no record of a published paper regarding the algorithm is available. However, it is worth testing out its performance. So

In terms of speed, YOLOv4-CSP achieved the highest score (50.9) compared to the second highest Efficientdet-d0 (34.53). YOLOv4-CSP will be selected to further improve its performance to achieve the objective in this project mentioned at the start, which is to improve the speed and accuracy of

license plate detection tasks using deep learning models through image processing, training, and modifying existing model architecture.

4.4 Preprocessing

The first intuitive way to improve the overall result is to perform simple image preprocessing for the test dataset. The test set is very challenging for any model to perform object detection as it is composed of images from various difficult conditions. Below shows the result for preprocessing, i.e., enlargement, sharpening, gamma correction, CLAHE, and non-local means denoising.

Table 4.2 Preprocessing Result for YOLOv4-CSP model

	Blur	db	fn	rotate	tilt	challenge
Dataset Images	20,611	10,132	20,967	10,053	30,216	50,003
mAP (0.70) (Before) (100 Imgs)	63.29	64.20	43.90	88.90	66.64	62.08
No. of Selected	100	100	100	100	100	100
TP-FN Ratio	70-30	69-31	52-48	92-8	77-23	67-33
FP	45	28	75	9	26	61
Human Eye TP	90	91	86	100	99	88
No. of Preprocess	6	4	14	-	-	12
Redrawn GT Box	-	-	-	8	23	-
mAP (0.70) (After) (100 Imgs)	68.26	67.09	57.39	96.40	77.94	71.11
Increment in mAP	4.97	2.89	13.49	7.50	11.30	9.03
TP-FN Ratio	75-25	73-27	66-34	97-3	85-15	76-24
FP	43	28	68	4	18	56
Legend						
TP – True Positive						
FN – False Negative						

FP – False Positive GT – Ground Truth

Bounding boxes are denoted as TP when it has more than 70% IoU with the ground truth box. When the model failed to detect the ground truth in an image, it will denote as FN. Lastly, when the bounding box has less than 70% IoU with the ground truth box, it will be denoted as False Positives. Note that there can be more than one FP for an image i.e., the model detected multiple incorrect bounding boxes. There can also be less FP than FN in a category, this occurs in the db test set because the image is so dark that the model does not detect any bounding boxes for the image. According to the mAP formula, FP is a variable that contributes to the score. False Positives occur very frequently in the above experiment due to the 0.69 IoU problem. This will further elaborate on later in this chapter.

In the above experiment In Table 4.2, 100 images out of each category are selected. The algorithm then uses YOLOv4-CSP model to perform object detection on these 100 images. At the same time, TP, FN, and FP are used to calculate the mAP. When deciding which image to preprocess, it is natural to select all FN images. However, due to the 0.69 problem, only a few images are selected. The table above shows mAP score before preprocessing and after preprocessing.

Six images were preprocessed for the blur test set. Five out of six images successfully convert from FN to TP. The mAP score increased from 63.29 to 68.26. Four images were preprocessed for the db test set. All four images successfully converted from FN to TP. The mAP score increased from 64.20 to

67.09. 14 images were preprocessed for the fn test set. All 14 images successfully converted from FN to TP. The mAP score increased from 43.90 to 57.39. 12 images were preprocessed for the challenge test set. Nine out of 12 images successfully convert from FN to TP. The mAP score increased from 62.08 to 71.11.

A different scenario happens to the rotate test set and the tilt test set. These test sets do not have an imbalance of light condition, blurry or distance and size issues. On human eye viewing, all 100 images are successfully detected on the rotate test set, and 99 images on the tilt test set. Image preprocessing will not improve the score for these test sets. However, there were still 8 FN and 23 FN on the rotate and tilt test set. To eliminate the FN, a further investigation has been carried out to tackle the issue.

Upon investigation, there is a problem with the annotation of the ground truth box by the dataset author. Refer to the figure 4.8. below:

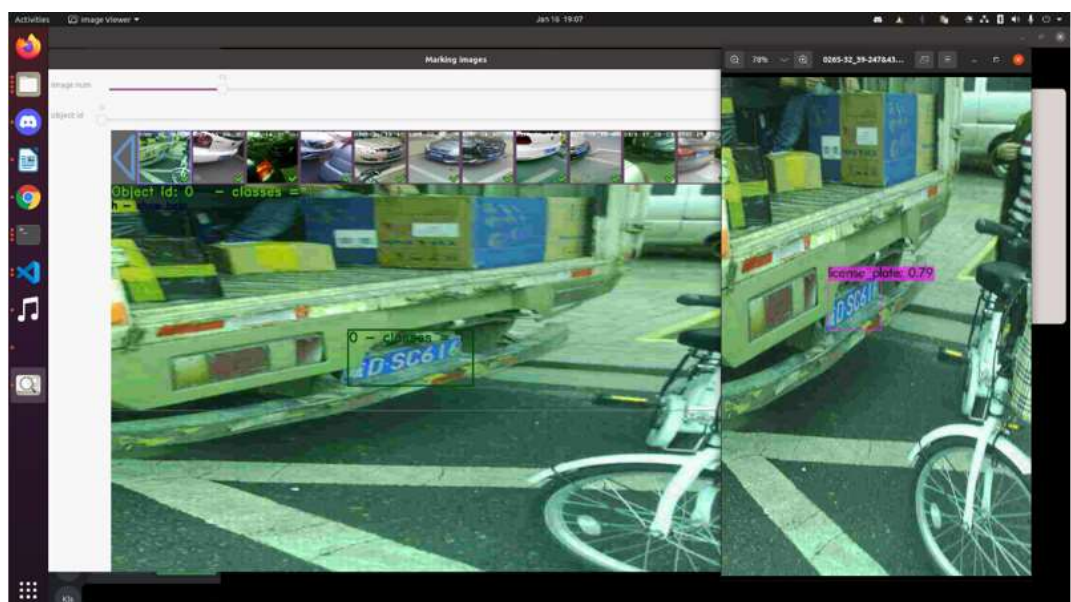


Figure 4.8 Ground Truth Box (Left) vs Model Prediction (Right)

As shown in the picture, the model successfully detected the object from the input. However, the box is denoted as a false positive due to the IoU with the ground truth box does not exceed 70%.

Observe that the ground truth box drawn is not ‘tight’ to the license plate object. This is the main problem with the dataset. Many bounding boxes were wrongly drawn during the data preparation stage. However, deep learning models are very good at adjusting and correcting their errors during the training phase. When the model is trained for a long period, it will slowly learn to adjust itself to be closer and closer to the actual object. Table 4.3 shows the example of the 0.69 IoU problem.

Table 4.3 0.69 IoU Problem

Image Name	IoU
0056-15_17...27.png	0.698895
0220-19_36...41.png	0.566831
0265-32_39...32.png	0.648916
0273-15_29...27.png	0.657966
0305-21_19...180.png	0.662354
0326-17_29...16.png	0.637488
0330-19_22...56.png	0.679596
0353-19_22...43.png	0.000000
0369-34_34...164.png	0.699994
0401-37_31...85.png	0.696482
0419-19_33...70.png	0.000000
0419-38_43...156.png	0.689906
0432-22_18...130.png	0.659566
0436-22_33...115.png	0.688282
0480-17_37...141.png	0.658595
...	...

4.5 Modify YOLOv4-CSP (Proposed Model)

The second approach to improve the overall performance is to modify the model architecture in terms of layers, filters, etc. In this experiment, six new layers have been added to the model. Refer to table 4.4.

Table 4.4 YOLOv4-CSP-Modified Architecture

Layer	filters	Size/strd(dil)	input	output	bflops
0conv	32	3x3/1	640x640x3	640x640x32	0.708BF
1conv	64	3x3/2	640x640x32	320x320x64	3.775BF
2conv	32	1x1/1	320x320x64	320x320x32	0.419BF
3conv	64	3x3/1	320x320x32	320x320x64	3.775BF
...
.
.
.
...
136conv	128	1x1/1	80x80x128	80x80x128	0.210BF
137conv	128	3x3/1	80x80x128	80x80x128	1.887BF
138conv	128	1x1/1	80x80x128	80x80x128	0.210BF
139conv	128	3x3/1	80x80x128	80x80x128	1.887BF
140route	139 134			80x80x256	
141conv	128	1x1/1	80x80x256	80x80x128	0.419BF
142conv	256	3x3/1	80x80x128	80x80x256	3.775BF
143conv	128	1x1/1	80x80x256	80x80x128	0.419BF
144conv	256	3x3/1	80x80x128	80x80x256	3.775BF
145conv	18	1x1/1	80x80x256	80x80x18	0.059BF
146yolo					
147route	141				
148conv	256	3x3/2	80x80x128	40x40x256	0.944BF
149route	148 127			40x40x512	
150conv	256	1x1/1	40x40x512	40x40x256	0.419BF
151conv	256	1x1/1	40x40x256	40x40x256	0.210BF
152route	150			40x40x256	
153conv	256	1x1/1	40x40x256	40x40x256	0.210BF
154conv	256	3x3/1	40x40x256	40x40x256	1.887BF
155conv	256	1x1/1	40x40x256	40x40x256	0.210BF
156conv	256	3x3/1	40x40x256	40x40x256	1.887BF
157route	156 151			40x40x512	
158conv	256	1x1/1	40x40x512	40x40x256	0.419BF
159conv	512	3x3/1	40x40x256	40x40x512	3.775BF
160conv	256	1x1/1	40x40x512	40x40x256	0.419BF
161conv	512	3x3/1	40x40x256	40x40x512	3.775BF
162conv	18	1x1/1	40x40x512	40x40x18	0.029BF
163yolo					
164route	160				
165conv	512	3x3/2	40x40x256	20x20x512	0.944BF
166route	165 111			20x20x1024	
167conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
168conv	512	1x1/1	20x20x512	20x20x512	0.210BF
169route	167			20x20x512	
170conv	512	1x1/1	20x20x512	20x20x512	0.210BF
171conv	512	3x3/1	20x20x512	20x20x512	1.887BF
172conv	512	1x1/1	20x20x512	20x20x512	0.210BF
173conv	512	3x3/1	20x20x512	20x20x512	1.887BF
174route	173 168			20x20x1024	
175conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
176conv	1024	3x3/1	20x20x512	20x20x1024	3.775BF

177conv	512	1x1/1	20x20x1024	20x20x512	0.419BF
178conv	1024	3x3/1	20x20x512	20x20x1024	3.775BF
179conv	18	1x1/1	20x20x1024	20x20x18	0.015BF
180yolo					

The intuition behind adding new layers is that they can learn new additional features thus improving the accuracy. The model is then retrained using the training dataset. Below shows the training graph.

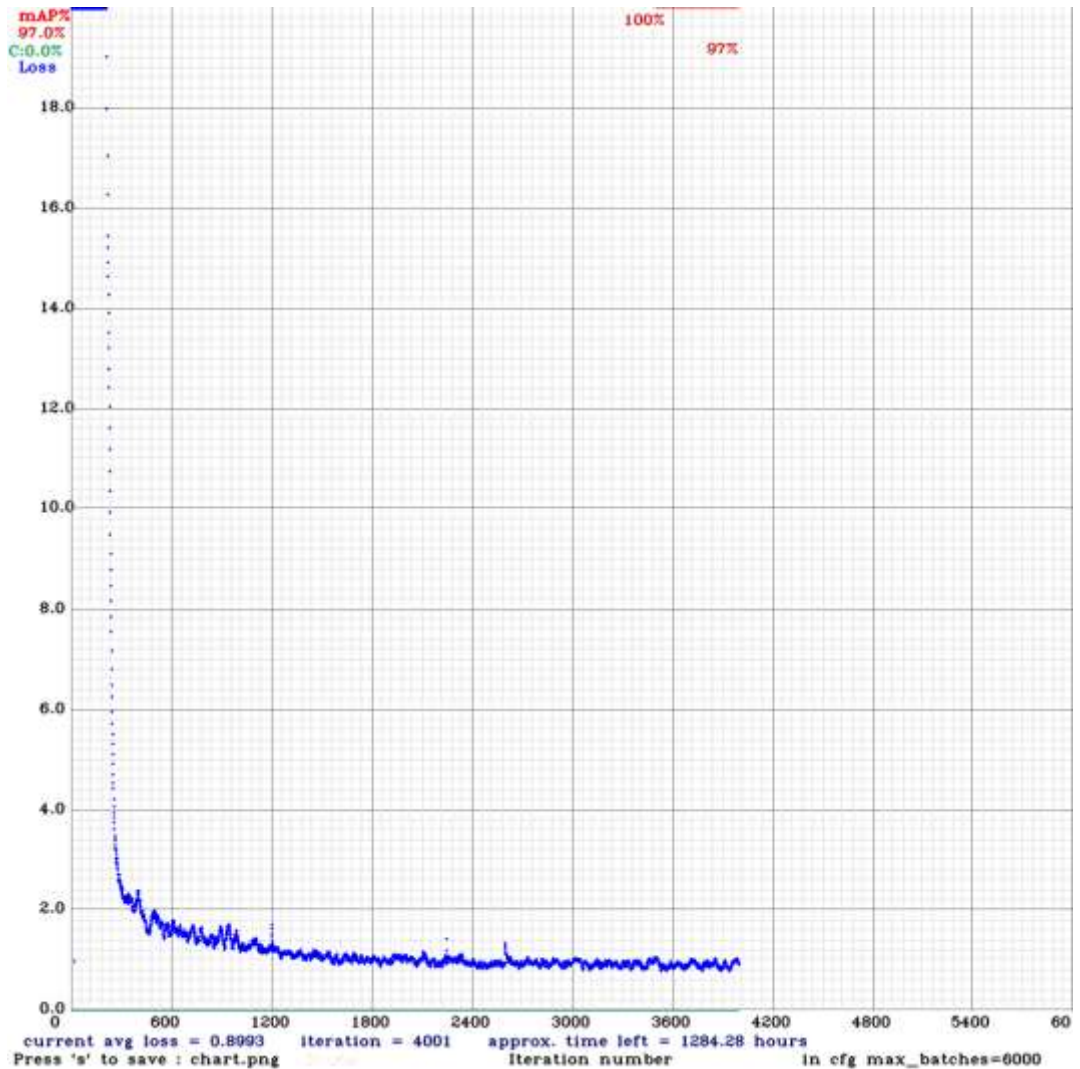


Figure 4.9 YOLOv4-CSP-Modified Training Loss Graph

The whole training process elapsed 10 hours, twice the duration of the original model (5 hours). This is due to the newly initialized layers are yet to learn meaningful weights from the training.

From the graph above, the average total loss has reached below 1.0 at around 1800 to 2400 iterations. However, the model is then left to train longer until very minor improvement can be seen till the 4000th iteration. Note that one iteration is one batch of 32 images. The machine is capable of training 6~7 iterations per minute.

The newly trained model is then be evaluated on the test set. Below shows the comparison between the original model and the new model with six additional layers.

Table 4.5 Accuracy (mAP@0.70) of Modified YOLOv4-CSP vs Original YOLOv4-CSP

Dataset	blur	db	fn	rotate	tilt	challenge	valid
No. of Images	20,611	10,132	20,967	10,053	30,216	50,003	99,996
YOLOv4-CSP	67.58	64.54	29.46	81.75	65.76	66.41	83.64
Modified YOLOv4-CSP	74.65	51.48	49.78	67.57	45.12	83.62	96.96
Increment	+7.07	-13.06	+20.32	-14.18	-20.64	+17.21	+13.32

The accuracy on the validation dataset improves from 83.64 to 96.96 which shows that adding six new layers can help the model to detect the objects under normal conditions more accurately. The problem where the model was having difficulty detecting small or far objects has also been improved from

29.46 to 49.78. This shows that the model can predict even tighter bounding boxes which match 70% of the ground truth boxes.

On the other hand, a decline in performance can be seen on the test set rotate, tilt, and db. This is due to the rotate and tilt test set containing non-rectangle objects therefore the drawn bounding boxes are loose. The model is having difficulty when deciding the real coordinates of the objects.

For the case of the db test set, the accuracy drops from 64.54 to 41.48. This is due to that this dataset mostly consists of too-bright and too-dark images. When the newly trained model is trained to draw a tighter bounding box, it will then have difficulty determining the real coordinates of the objects in such lighting conditions, resulting in a degraded performance in 0.70 mAP.

Overall, the new modified model can detect objects that have good visibility in terms of lighting and blurriness. In addition to that, it has also become very strict on objects that had failed to represent regular shapes in terms of skewness and angle.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

In this project we have designed an experiment to evaluate the performance of various state-of-the-art deep learning object detection models on a new dataset which is CCPD with transfer learning. Secondly, this project also shows two ways i.e., image preprocessing step and introducing additional model layers to improve the overall performance as stated in the objectives of the research. The image preprocessing step has shown that the visibility of the license plate can be improved in various conditions. The improved YOLOv4 model has achieved (96.96%) mean Average Precision $mAP@0.70$ on the validation dataset compared to the original model (83.64%) on the Chinese City Parking Dataset (CCPD). This shows that the six additional convolutional layers added have helped the model to predict the bounding boxes more accurately.

Deep learning is an exciting field of research, especially in computer vision. It has a continuous fast pace of development, and the accuracy is overtaken by new models every year. Now, a new type of object detection model known as ‘transformer’ which is derived from the natural language model can already surpass the current best model.

5.2 Future Work and Suggestions

There are many potential issues and use cases in real life as deep learning technology has just begun ten years ago. One suggestion to improve the model

performance is to further tweak the model's architecture such as layers number, convolutional filter sizes, hyperparameters and others. This project also faces the limitations of the processing power and memory sizes of a consumer grade hardware. A better hardware will allow a bigger size of model to be trained efficiently hence present a better result.

REFERENCES

- Anagnostopoulos, C.N.E. *et al.* (2006) ‘A license plate-recognition algorithm for intelligent transportation system applications’, *IEEE Transactions on Intelligent Transportation Systems*, 7(3), pp. 377–391. Available at: <https://doi.org/10.1109/TITS.2006.880641>.
- ANPR Cameras | ANPR accuracy test* (2019). Available at: <https://sensorable.io/articles/anpr-accuracy-test/index.html> (Accessed: 10 September 2022).
- Ashtari, A.H., Nordin, Md.J. and Seyed Mostafa Mousavi Kahaki (2011) ‘A new reliable approach for Persian license plate detection on colour images’, in *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*. IEEE, pp. 1–5. Available at: <https://doi.org/10.1109/ICEEI.2011.6021697>.
- Bay, H. *et al.* (2008) ‘Speeded-Up Robust Features (SURF)’, *Computer Vision and Image Understanding*, 110(3), pp. 346–359. Available at: <https://doi.org/10.1016/j.cviu.2007.09.014>.
- Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y.M. (2020) ‘YOLOv4: Optimal Speed and Accuracy of Object Detection’.
- Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2021). *GitHub - AlexeyAB/darknet: YOLOv4 / Scaled-YOLOv4 / YOLO - Neural Networks for Object Detection (Windows and Linux version of Darknet)*. Available at: <https://github.com/AlexeyAB/darknet> (Accessed: 10 September 2022)
- Calonder, M. *et al.* (2010) ‘BRIEF: Binary Robust Independent Elementary Features’, in, pp. 778–792. Available at: https://doi.org/10.1007/978-3-642-15561-1_56.
- Chang, S.L. *et al.* (2004) ‘Automatic License Plate Recognition’, *IEEE Transactions on Intelligent Transportation Systems*, 5(1), pp. 42–53. Available at: <https://doi.org/10.1109/TITS.2004.825086>.
- Creating TFRecords* (2021). Available at: https://keras.io/examples/keras_recipes/creating_tfrecords/ (Accessed: 17 October 2022).
- Duan, K. *et al.* (2019) *CenterNet: Keypoint Triplets for Object Detection*. Available at: <https://github.com/>.
- Faradji, F., Rezaie, A.H. and Ziaratban, M. (2007) ‘A Morphological-Based License Plate Location’, in *2007 IEEE International Conference on Image Processing*. IEEE, pp. I-57–I-60. Available at: <https://doi.org/10.1109/ICIP.2007.4378890>.

Girshick, R. *et al.* (2014) *Rich feature hierarchies for accurate object detection and semantic segmentation*. Available at: <http://arxiv>.

Girshick, R. (2015) *Fast R-CNN*. Available at: <https://github.com/rbgirshick/>.

Habeeb, D. *et al.* (2021) 'Deep-Learning-Based Approach for Iraqi and Malaysian Vehicle License Plate Recognition', *Computational Intelligence and Neuroscience*, 2021. Available at: <https://doi.org/10.1155/2021/3971834>.

Hendry and Chen, R.C. (2019) 'Automatic License Plate Recognition via sliding-window darknet-YOLO deep learning', *Image and Vision Computing*, 87, pp. 47–56. Available at: <https://doi.org/10.1016/j.imavis.2019.04.007>.

Hsu, G.S., Chen, J.C. and Chung, Y.Z. (2013) 'Application-oriented license plate recognition', *IEEE Transactions on Vehicular Technology*, 62(2), pp. 552–561. Available at: <https://doi.org/10.1109/TVT.2012.2226218>.

Hui J (2018) *SSD object detection: Single Shot MultiBox Detector for real-time processing* | by Jonathan Hui | Medium. Available at: <https://jonathan-hui.medium.com/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06> (Accessed: 10 September 2022).

Jacob, S. (2020) *EfficientDet for Object Detection*. Available at: <https://blog.roboflow.com/breaking-down-efficientdet/> (Accessed: 10 September 2022).

Jørgensen, H. (2017) *Automatic License Plate Recognition using Deep Learning Techniques*. Available at: <https://doi.org/http://hdl.handle.net/11250/2467209>.

Karol Majek (2018) (10) *SSD Mobilenet Object detection FullHD S8#001 - YouTube*. Available at: <https://www.youtube.com/watch?v=7p2XL8wApfo&t=430s> (Accessed: 10 September 2022).

Liu, W. *et al.* (2016) 'SSD: Single Shot MultiBox Detector', in, pp. 21–37. Available at: https://doi.org/10.1007/978-3-319-46448-0_2.

Malaysia - Flash report, Automotive sales volume, 2022 - MarkLines Automotive Industry Portal. (2022). Available at: https://www.marklines.com/en/statistics/flash_sales/automotive-sales-in-malaysia-by-month (Accessed: 9 February 2023)

Ng, P.C. (2003) 'SIFT: predicting amino acid changes that affect protein function', *Nucleic Acids Research*, 31(13), pp. 3812–3814. Available at: <https://doi.org/10.1093/nar/gkg509>.

Redmon, J. *et al.* (2016) *You Only Look Once: Unified, Real-Time Object Detection*. Available at: <http://pjreddie.com/yolo/>.

Ren, S. *et al.* (2015) *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Available at: <https://github.com/>.

Selmi, Z., ben Halima, M. and Alimi, A.M. (2017) ‘Deep Learning System for Automatic License Plate Detection and Recognition’, in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, pp. 1132–1138. Available at: <https://doi.org/10.1109/ICDAR.2017.187>.

Sharma, P., Karan, J. and Karan Sing, J. (2015) ‘Challenges and Overview of License Plate Character Segmentation’, *International Journal of Computer Science International Journal of Computer Science International Journal of Computer Science International Journal of Computer* [Preprint]. Available at: www.ijcaonline.org.

Tan, M., Pang, R. and Le, Q. v. (2019) ‘EfficientDet: Scalable and Efficient Object Detection’. Available at: <http://arxiv.org/abs/1911.09070>.

Tan, M. and Yu, A. (2020) *Google AI Blog: EfficientDet: Towards Scalable and Efficient Object Detection*. Available at: <https://ai.googleblog.com/2020/04/efficientdet-towards-scalable-and.html> (Accessed: 10 September 2022).

Uri Almog (2021) *CenterNet, Explained. CenterNet is an anchorless object... | by Uri Almog | Towards Data Science*. Available at: <https://towardsdatascience.com/centernet-explained-a7386f368962> (Accessed: 10 September 2022).

Viswanathan DG. (2011). *Features from Accelerated Segment Test (FAST)*.

Xu, Z. *et al.* (2018) *Towards End-to-End License Plate Detection and Recognition: A Large Dataset and Baseline*. Available at: <https://github.com/detectRecog/CCPD>.

Zang, D. *et al.* (2015) ‘Vehicle license plate recognition using visual attention model and deep learning’, *Journal of Electronic Imaging*, 24(3), p. 033001. Available at: <https://doi.org/10.1117/1.jei.24.3.033001>.

Zheng, D., Zhao, Y. and Wang, J. (2005) ‘An efficient method of license plate location’, *Pattern Recognition Letters*, 26(15), pp. 2431–2438. Available at: <https://doi.org/10.1016/j.patrec.2005.04.014>

