ENHANCED KNOWLEDGE IN SCIENCES AND TECHNOLOGY
e-ISSN: 2773-6385

EKST

Vol. 5 No. 2 (2025) 188-198
https://publisher.uthm.edu.my/periodicals/index.php/ekst

License Plate Detector Application with Single Shot Detector Model

Gan Wei Hang[1], Lee Siaw Chong[1]*

[1] *Department of Mathematics and Statistics, Faculty of Applied Sciences and Technology, UTHM Kampus Cawangan Pagoh, Hab Pendidikan Tinggi Pagoh, KM 1, Jalan Panchor, 84600, Pagoh, Muar, Johor MALAYSIA*

*Corresponding Author: sclee@uthm.edu.my
DOI: https://doi.org/10.30880/ekst.2025.05.02.022

Article Info

Received: 30 December 2024
Accepted: 17 January 2025
Available online: 19 December 2025

Keywords

License Plate Detector, Machine Learning, Convolutional Neural Network, Single Shot Detector, Mean Average Precision

Abstract

Convolutional Neural Networks are one the best technologies used in license plate detection, specifically Single Shot Detector (SSD), which has great speed and accuracy. There is a lack of studies on the effects between epochs of batch size towards time and mAP in SSD research. Hence, a license plate detector using the SSD model is developed, and the optimal batch size and hyperparameters are identified. A dataset containing 7058 training images, 2048 validation images and 1020 testing images are used. We obtain the highest mAP of 66.13% when the batch size is 2. Generally, when batch size increases, the computation time and mAP decreases. An epoch of 2 has the lowest training time of 0.39 hours but trades off mAP of only 62.09%. The mAP gradually increases when the epoch increases to 8 epochs at 65.53% and stagnates at higher epochs. Thus, the optimal hyperparameters are batch size of 2 and 8 epochs. Then, an application based on the license plate detector is created. Using a small sample of images obtained from the Universiti Tun Hussein Onn campus, the application can detect completely grayscale images unaffected by the lack of colour. The application can best detect front license plates at 4 meters and rear license plates at 1 meter with confidence of 91% and 97%, respectively. The best angle detected has a confidence score of 99%.

1. Introduction

The License Plate Recognition (LPR) system was initiated by the police scientific development branch in the United Kingdom in 1976 [1]. This specific implementation of LPR back then was to reduce illegal drug trafficking. LPR has evolved to aid in various traffic applications such as law enforcement, toll collection, congestion control and targeted advertising. In recent years, the deployment of LPR systems across Malaysia has become more prevalent, especially in parking lots, to provide users with a more seamless and hassle-free parking experience since LPR can pinpoint the number plate on a vehicle and then analyse its license plate number for vehicle identification. With new techniques and hardware improvements, LPR has become more accurate even in less-than-ideal conditions such as lower image quality, rainy or foggy weather effects, image overexposure or non-standard license plates [2].

LPR consists of 4 main processes: image preprocessing, license plate detection, character segmentation and Optical Character Recognition (OCR). First, standard preprocessing techniques such as downscaling, greyscale or binarisation are applied to the image [3], improving efficiency for the later steps. Then, license plate detection is performed to extract only the license plate area from the image. Afterward, individual characters are segmented from the license plate through image segmentation. Finally, the characters themselves are recognised through

OCR. This research will focus on the license plate detection part of LPR. The application built in this project will be called a license plate detector.

One of the breakthroughs in license plate detection technology is the Convolutional Neural Network (CNN), which introduces the convolution technique. Convolution involves a matrix kernel that slides along the input image, also represented as a matrix, to produce an output [4]. CNNs usually consist of three layers: the convolutional layer, the pooling layer, and the fully connected layer. Convolutional layers help reduce computational load while improving feature extraction, pooling layers reduce the number of parameters to improve performance, and the fully connected layer enables a human to read the output [5]. CNNs first perform feature extraction with the convolutional and pooling layers, and then the fully connected layer performs classification based on the features obtained from the previous layers.

The Single Shot Detector (SSD) is one feed-forward convolutional network specialising in object detection. SSD is a one-stage detector, which means it does not resample pixels or features for bounding box hypotheses. Pixel resampling is done by two-stage detectors such as Faster R-CNN to obtain impressive detection accuracies [6]. However, these two-stage detectors are too computationally intensive even with today's hardware, and they are too slow for real-time applications. Previous attempts were made to build faster detectors but faced a significant drop in accuracy. SSD manages to escape this accuracy drop by adding multiple small improvements that improve accuracy significantly. This makes SSD the preferred model for real-time applications such as license plate detection for this research. The early layers of SSD consist of a standard CNN model, also known as a backbone, that is truncated before any classification layers. Therefore, only the feature extraction layers are kept. An auxiliary structure was added on top of the backbone, which contains additional convolutional layers and multi-scale feature maps to replace the fully connected layer. The most common backbone is Visual Geometry Group 16 (VGG16), the backbone of choice for this research. Other standard backbones include ResNet50 and MobileNetV3.

The VGG16 model consists of 16 layers with 13 convolutional layers and three fully connected layers. Its unique characteristic is the usage of very small convolution filters (3 × 3), and consistent filter sizes and pooling layers across the model. This helps VGG16 to perform excellently in image classification tasks and is often used for transfer learning [7].

[8] used SSD with a VGG16 model pre-trained on Imagenet as a backbone for traffic sign recognition. A dataset was augmented to compare the SSD performance using the original and the augmented dataset. Data augmentation is done to increase the number of images from an existing dataset. The data augmentation techniques used in this study involve affine transformation, changing the brightness and contrast, and adding motion blur. All testing is done on a batch size of 7 over a single epoch. The results showed that using only the augmented dataset was not enough to improve the performance of SSD. However, using both original and augmented datasets improved the SSD results.

A comparative analysis was performed with SSD using different CNNs as feature extractors for vehicle detection in an unconstrained road scenario [9]. The CNNs used are InceptionNetV2, MobileNetV2 and MoBileNetV1. The results showed that SSD MobileNetV1 had the lowest classification time of 4.4 seconds but had the lowest Mean Average Precision (mAP) of 21%. SSD InceptionNetV2 had the best overall performance with a classification time of 4.75 seconds and mAP of 24%. This showed that the feature extractor plays a significant role in SSD performance. Image size is another factor in SSD performance [10]. A decrease in image resolution by a factor of two primarily results in a similar reduction in mAP, especially since SSD is less optimal for detecting small objects. We can see that some hyperparameters used in CNNs, especially SSD, such as epochs and batch size are not mentioned in the above studies. These hyperparameters are important since they can impact the performance of the license plate detector. Hence this motivates us to study their effects on SSD.

There are three objectives in this research. An SSD model with a VGG16 backbone specialising in license plate detection using the PyTorch machine-learning library will be developed. The dataset used for training is the License Plate Recognition Computer Vision Project created by Roboflow Universe Projects obtained from [11]. This study then aims to identify the optimal hyperparameters between epochs and batch size to decrease the time taken during training while maintaining a mAP of at least 65%. Lastly, a user-friendly application will be created to detect unseen license plate images and evaluate the strengths and weaknesses of the license plate detector.

2. Materials and Methods

The basic layers in a Convolutional Neural Network (CNN) are convolutional layers, pooling layers, and fully connected layers. Convolutional layers apply filters to create feature maps. Pooling layers reduce spatial dimensions and complexity. Fully connected layers classify features by converting the final output into a vector for prediction. Various activation functions are used within these layers to detect desirable features.

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 188-198

2.1 VGG16 Architecture

Visual Geometry Group 16 (VGG16) is a specific CNN architecture with 13 convolutional layers and three fully connected layers [12]. The input is a red, blue and green (RGB) image that forms an input matrix of 224 × 224 × 3. This image is passed through a stack of convolutional layers. All the convolutional layers in VGG16 use a small kernel size of 3 × 3 with stride, $s = 1$ and zero padding, $p = 1$ so the spatial resolution is preserved after the convolution. Every convolutional layer is followed by the Rectified Linear Unit (ReLU) activation function. The input matrix is then passed through the following layers, represented in Fig. 1. A block containing 2 or 3 convolutional layers always ends with a max pool layer that uses a 2 × 2 kernel with $s = 2$ which reduces the horizontal dimensions by a factor of 2. Table 1 shows the specific size and number of kernels of the convolution layers in each block.

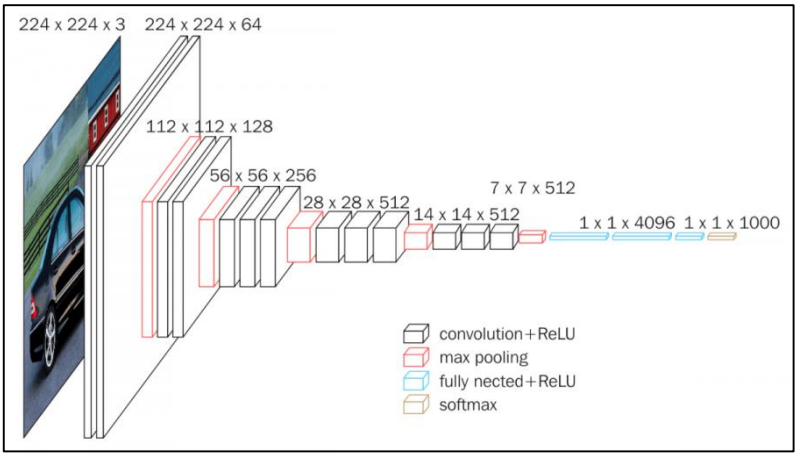

Fig. 1 *VGG16 architecture [13]*

Table 1 Specifications of Convolutional Layers in VGG16

Block number	Number of convolutional layers	Number of kernels in each convolutional layer	Convolutional layer size
1	2	64	224 × 224 × 64
2	2	128	112 × 112 × 128
3	3	256	56 × 56 × 256
4	3	512	28 × 28 × 512
5	3	512	14 × 14 × 512

After the convolutional layers in the fifth and final block, the last max pool layer decreases the output feature map size to 7 × 7 × 512. The output feature map is then flattened into a vector of size 25088 before entering the first fully connected layer. Each fully connected layer is followed by the ReLU activation function. The first fully connected layer has an input size of 25088 and an output size of 4096. The second fully connected layer has an input size of 4096 and an output size of 4096 to further refine the output without losing information, followed by the third fully connected layer with an input size of 4096 and an output size of 1000 to match the 1000 classes VGG16 is designed to detect. The SoftMax activation is applied to the final vector of size 1000 to perform classification over 1000 classes.

2.2 SSD Architecture

In this research, the Single Shot Detector (SSD) uses Visual Geometry Group (VGG) 16 as a base network for high-quality image classification [6]. The layers of VGG16 used for SSD are from the beginning of the network until conv5_3, which means the third convolutional layer in the fifth block of the VGG16 network, as shown in Fig. 2. The rest of VGG16 is truncated and replaced by an auxiliary structure that includes convolutional feature layers that progressively decrease in size to allow predictions and detections at multiple scales. Using an input image with 300 × 300, the size of each feature layer is 38 × 38, 19 × 19, 10 × 10, 5 × 5, 3 × 3, and 1 × 1. This forms different-sized feature maps, which are represented as a matrix. The large feature map in the front detects small objects and vice versa.

Each feature map cell has a set of default bounding boxes that tile the feature map in a convolutional manner to fix the bounding box position relative to the corresponding cell. Each feature map cell contains multiple default bounding boxes with different sizes and aspect ratios. At each cell, SSD predicts c class scores, where c is the

number of classes to be detected and four offsets relative to the original default box shape which produces $c + 4$ filters for each default box. Thus, given there are k default boxes from a given cell, SSD produces $(c + 4)k$ filters for each cell. With an $m \times n$ feature layer, there will be a total of $(c + 4)kmn$ outputs.

SSD directly uses convolution to extract detection results from the different feature maps instead of a fully connected layer. These convolutional filters are shown on top of the arrows in Fig. 2. A feature layer with a size of $m \times n$ with p channels (feature maps) can use a $3 \times 3 \times p$ sized kernel to form a predicted score for a category or a shape offset relative to the default box coordinates. This is used for potential detections. This process will produce 8732 bounding boxes for each class. The breakdown of the formation of these bounding boxes is shown in Table 2.

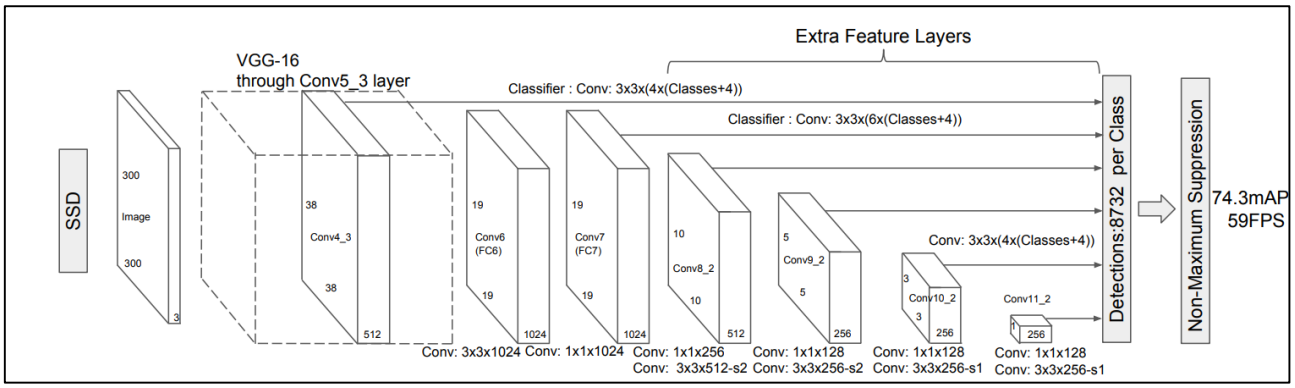

Fig. 2 SSD architecture [6]

Table 2 Formation of Bounding Boxes

Feature map	Convolutional layer involved	Size, $m \times n$	Number of default bounding boxes per cell, k	Number of default bounding boxes in the convolutional layer, $m \times n \times k$
1	Conv 4_3	38×38	4	5776
2	Conv7	19×19	6	2166
3	Conv8_2	10×10	6	600
4	Conv9_2	5×5	6	150
5	Conv10_2	3×3	6	54
6	Conv11_2	1×1	6	6
			Total:	8732

Since a forward pass of SSD produces many boxes (8732), most of the default boxes will be negative, and they will be used as an example for the background class. A significantly larger number of negative examples than positive ones will cause a major imbalance. Therefore, a method called hard negative mining is used. Only the negative examples with the highest confidence loss are kept, and the ratio between negatives and positives is maintained at 3:1. This ensures a more stable and faster optimisation during the training process.

As for testing, there will be many bounding box predictions, each with its confidence scores. We use non-maximum suppression (NMS) with an initial confidence threshold of 0.01 to filter out most boxes. The leftover boxes are then sorted by confidence score so that the boxes with the highest confidence score are kept. A different threshold, intersection over union (IoU) = 0.45, is used to apply NMS again to remove redundant, overlapping boxes so a final single bounding box is produced. A visualisation of this process is shown in Fig. 3.

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 188-198

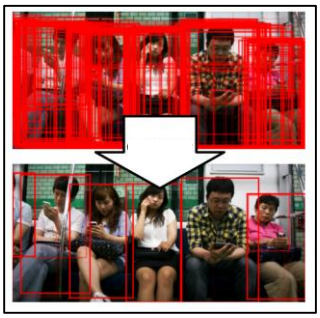

Fig. 3 Example of non-maximum suppression [14]

2.3 Mean Average Precision (mAP)

Mean Average Precision (mAP) is a widely used metric to measure the accuracy of object detection models. The main concepts involved in calculating mAP are the intersection over union (IoU), precision, and recall.

For each bounding box, IoU is obtained by measuring the overlap between the predicted bounding box and the ground truth bounding box. The obtained IoU is then compared with a given IoU threshold. For example, if the IoU threshold is 0.5, and the IoU value produced during prediction is 0.5 or higher, the prediction is classified as a true positive. IoU is calculated using the following equation,

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}. \tag{1}$$

Precision is the ratio of true positives (correctly predicted positive observations) to the total predicted positive observations, represented by the following equation,

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}. \tag{2}$$

Recall is the ratio of true positives to all observations that are actually positive, as shown in this equation,

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{3}$$

We then obtain the average precision (AP) by measuring the area under the precision-recall curve for a single class, the area is shown in Fig. 4. The mAP is the mean of the sum of APs across all classes. In this research, the threshold value used is mAP [0.5,0.95]. This means an IoU threshold from 0.5 to 0.95 in increments of 0.05 is used sequentially to calculate many precision-recall curves, which are averaged out to form mAP.

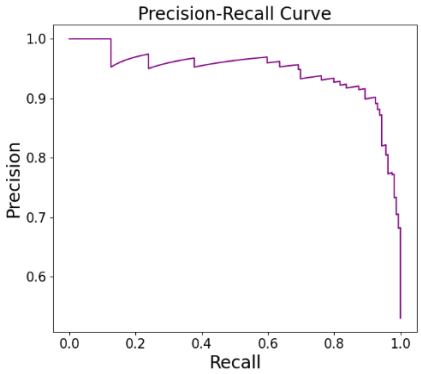

Fig. 4 Precision-Recall Curve Example

2.4 Implementation of License Plate Detector

Using Python, several important libraries are imported to build the license plate detector. Starting with Torchvision, which is a subset of Pytorch, a Single Shot Detector (SSD) model with Visual Geometry Group (VGG) 16 backbone pre-trained on the COCOV1 dataset will be imported to be the base of the license plate detector. The module Torchmetrics is used to measure the model's Mean Average Precision (mAP), and the model optimiser is

also imported from PyTorch. OpenCv is imported to draw a visible bounding box for the user on unseen data based on the predictions made by the model.

Using the License Plate Recognition Computer Vision Project created by Roboflow Universe Projects obtained from [11], the images are in different sizes, so they will be resized to a fixed size of 640 ×640 before it is passed onto the backbone. 7058 images are allocated for training, 2048 images for validation and 1020 images for testing. We use stochastic gradient descent for the optimiser to reduce training loss at a learning rate of 0.0001. The number of workers used during training is 4. To reduce overfitting, we use data augmentation techniques imported from the Albumentations library so that the training data is slightly altered in each epoch. The augmentations used are blur, motion blur and median blur, all with a blur limit of 3 and probability of 0.1. Other augmentations are grayscale, brightness, contrast, colour jitter, and random gamma, all with a probability of 0.3. Some hyperparameters of the model will be adjusted to compare model performance. These hyperparameters are the number of epochs with values 2, 4, 8, 16, and 32 and batch size with values 1, 2, 4, 8, 16, 32, 48. The number of epochs determines the number of times the training data is passed in the model, while the batch size is the number of training data used in one iteration of model training. Note that one epoch can contain many iterations.

The performance of the license plate detector is done on a Lenovo LOQ 15IAX9 with an Intel core i5-12450HX, NVIDIA GeForce RTX 4050 Graphics Processing Unit and 20GB DDR5 RAM. The outline of the training and evaluation process of the license plate detector is shown in Fig. 5.

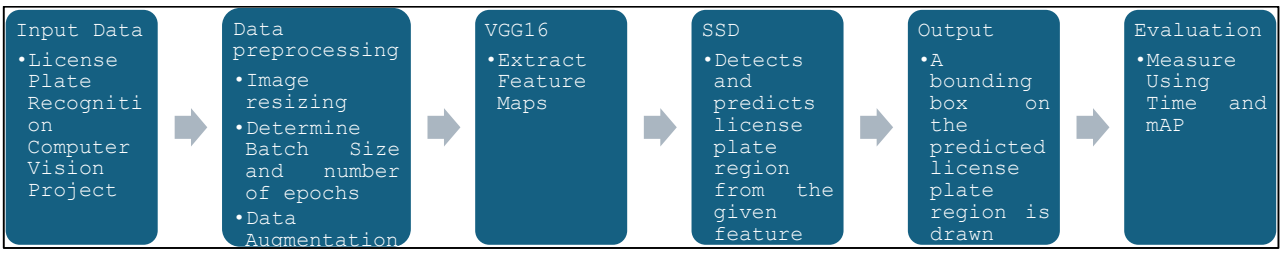

Fig. 5 General architecture of the license plate detector

3. Results and Discussion

In this section, we evaluate the effects of batch size and epochs towards time taken and Mean Average Precision (mAP) to find the optimal balance between batch size and epochs that results in low computation time and high mAP. The batch sizes tested are 1, 2, 4, 8, 16, 32 and 48, and the epochs tested are 2, 4, 8, 16 and 32. Then, the license plate detector application is created, and we use it to test the application's strengths and weaknesses towards different sample images.

3.1 Effects of Batch Size

The test in this section is done with 8 epochs. The Graphics Processing Unit (GPU) used in this research has 6GB of memory. During training, as seen in Table 3, an increase in batch size correlates to an increase in GPU memory usage. For batch size 48, the required GPU memory exceeds the available GPU memory in this research, resulting in memory overflow and severely slowing down the training process.

Table 3 GPU Memory Usage

Batch Size	GPU Memory (GB)	Batch Size	GPU Memory (GB)
1	0.6	16	2.6
2	0.7	32	5.0
4	1.0	48	Overflow
8	1.4		

According to Fig. 6, as batch size increases, the computation time decreases up to a batch size of 32. When the batch size is 48, the time taken is more than doubled compared to the previous value due to the memory overflow shown in Table 4. As for mAP, there is a noticeable decrease as batch size increases; the batch size of 2 achieved the highest mAP at 66.13%, while the largest batch size (48) achieved the lowest mAP at 64.61%.

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 188-198

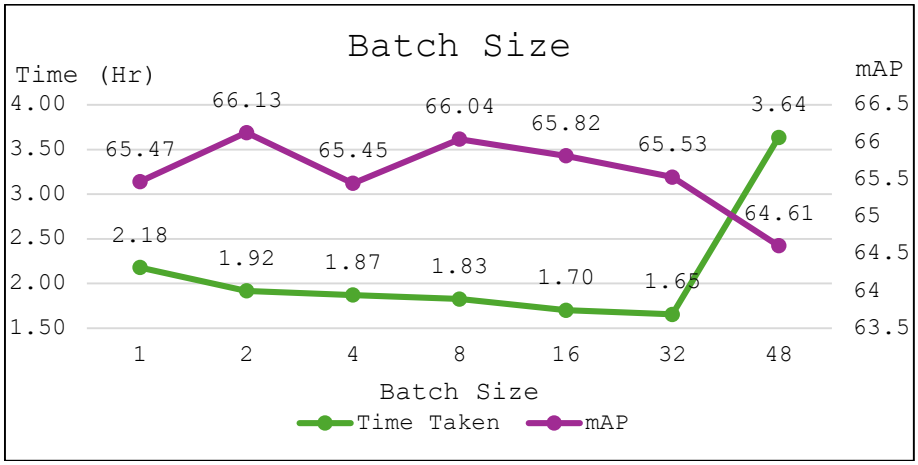

Fig. 6 Line chart of batch size against time (hr) and mAP

3.2 Effect of Epochs

The test in this section is done with batch size of 32. From Fig. 7, an increase in epochs correlates to both an increase in time and mAP. The increase in time taken is proportional to the increase in epochs, where the doubling of epochs also about doubles the time taken. Moving on to mAP, there is an increase in mAP from epochs = 2 to epochs = 8. A higher number of epochs gives the model more chances to learn the training data, giving it more opportunities to provide more accurate detections. However, the mAP stagnates from epochs = 8 to epochs = 32. This implies that in these epochs, the model suffers from overfitting by being too specialised in training data and performing poorly in unseen data.

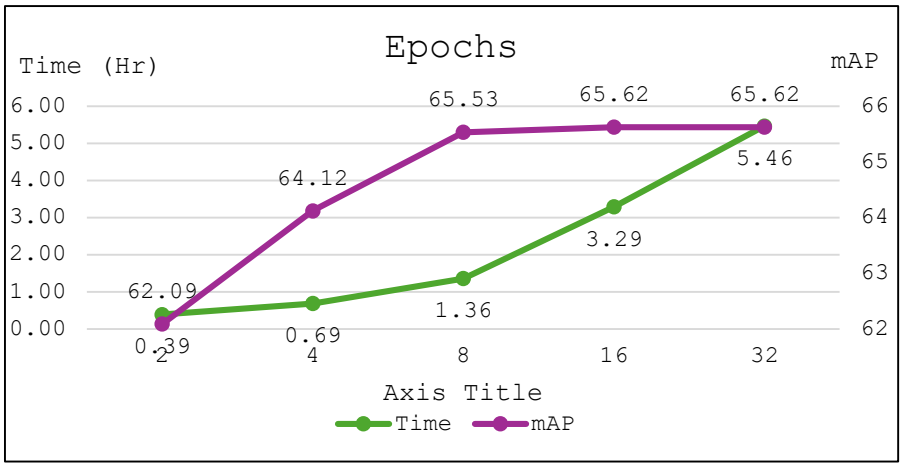

Fig. 7 Line chart of epochs against time (hr) and mAP

3.3 License Plate Detector Application

After multiple training sessions from the previous section, we used the model with the best mAP; the model with a batch size of 2 and 8 epochs was used to create the application. The application mainly uses tkinter for the user interface and cv2 to draw the bounding box on the license plate images. As seen in Fig. 8, the user first chooses images to be used to detect license plate images from the *Select Image(s)* button in the bottom left. The license plate detector then processes the selected images and draws the bounding box along with the confidence value in percentage (72%). This confidence value determines how confident the model thinks it has made a positive detection. The user can compare the input images on the left and the output images on the right. If multiple images are selected, the user can scroll down the interface to see the rest of the images shown in Fig. 9. Lastly, the user can save the output images in a folder of their choice. Fig. 9 shows further examples of the application's output, involving rear view and multiple cars. The first image in Fig. 9 is detected with 100% confidence, while the confidence values in the second image from the leftmost car to the rightmost car are 74%, 54% and 49% respectively.

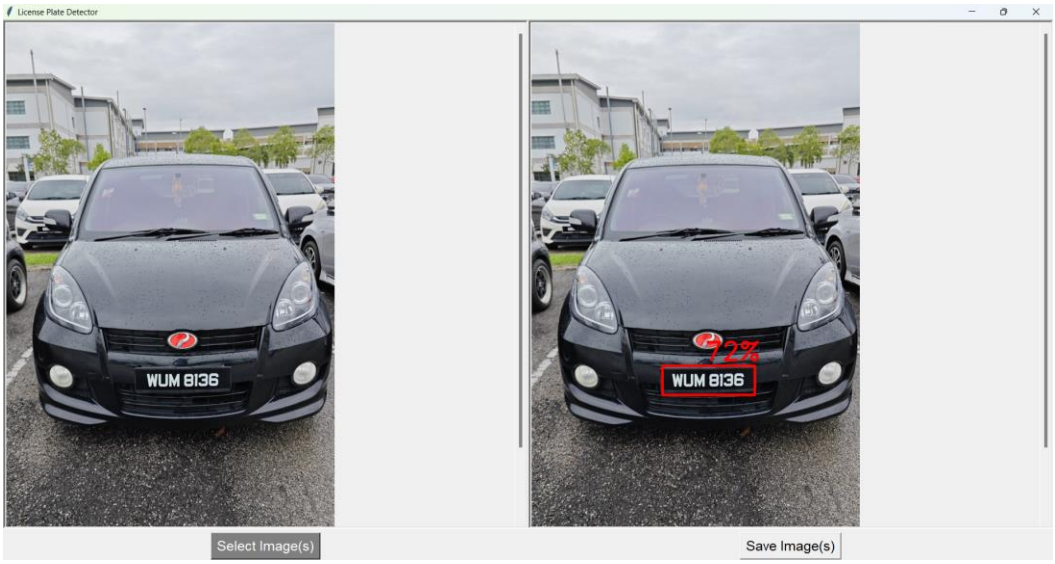

Fig. 8 Example image of license plate detector application

Fig. 9 Additional examples of license plate detector application

3.4 Strengths and Weaknesses of License Plate Detector

The license plate detector application is tested with a small sample of images taken near the Universiti Tun Hussein Onn Malaysia campus. Certain aspects of the images, which are colour, distance and angle, are changed to test their effects on the confidence score of the application. Due to the small sample size, the results shown here are not representative of the general performance of the license plate detector.

3.4.1 Colour

The license plate detector does not require colour information to process its data. Hence, the reduction of colour does not affect its accuracy and confidence, as shown in Fig. 10.

Enhanced Knowledge in Sciences and Technology Vol. 5 No. 2 (2025) p. 188-198

(a) (b)

Fig. 10 (a) unaltered photo; (b) grayscale photo

3.4.2 Distance

According to Table 4, if the license plate detector detects the car's front license plate, it prefers distances between 2 to 4 meters. In other distances, the confidence percentage is noticeably lower. On the other hand, the license plate detector works best for short distances for the rear license plate. The Single Shot Detector model is generally unsuitable for detecting very small objects [6]. This results in a drastic decrease in confidence at further distances.

.

Table 4 Relationship between distance and confidence score

Distance from the front of the car (m)	Confidence (%)	Distance from the rear of the car (m)	Confidence (%)
1	39	1	97
2	78	2	85
3	86	3	76
4	91	4	62
5	64	5	71
6	46	6	21

3.4.3 Angle

The results in this section are based on the photos shown in Fig. 11. The license plates are taken at increasing horizontal angles. The confidence scores are shown in Table 6. The confidence increases as the angle increases until it reaches the best score in Fig. 11(c). The confidence scores then fall, and the license plate detector fails to recognise the license plates at extreme angles, as shown in Fig 11. (e) and Fig 11. (f).

(a) (b) (c)

(d) (e) (f)

Fig. 11 Photos of license plates taken at different angles

Table 5 Confidence scores

Figure Used	Confidence (%)	Figure Used	Confidence (%)
10 (a)	80	10 (d)	91
10 (b)	87	10 (e)	Not detected
10 (c)	99	10 (f)	Not detected

4. Conclusion

The license plate detection technology used in this research is based on the Single Shot Detector model with Visual Geometry Group 16 backbone from PyTorch. Using data containing 7058 training images, 2048 validation images and 1020 testing images, this research analysed the effects of batch size and epochs towards time and Mean Average Precision (mAP). The batch size of 2 has the highest mAP at 66.13%. The time taken decreases as batch size increases, with a batch size of 32 taking the least training time of 1.65 hours while consuming 5GB of Graphics Processing Unit (GPU) memory. On the other hand, a batch size of 48 causes GPU memory to overflow and takes the most time of 3.64 hours. For epochs, a low number of epochs has a very low training time at the expense of mAP, with an epoch of 2 only taking 0.39 hours but having mAP of only 62.09%. The mAP gradually increases with an increase of epochs until 8 epochs, where the mAP reaches 65.53% with 1.36 hours of time taken. The mAP then stagnates at 16 and 32 epochs but with significantly higher computation time of 3.29 hours and 5.46 hours, respectively. With this, we conclude that the best model in this research is a model with a batch size of 2 and 8 epochs. A license plate detector application is built based on the selected hyperparameters. The application can accept multiple images of front and rear views and multiple cars.

Using the small number of images obtained from the Universiti Tun Hussein Onn Malaysia campus, we observe that the application is unaffected by greyscale value. The application can best detect the front license plate at 4 meters with 91% confidence, whereas the back license plate is best at 1 meter with 97% confidence. Following that, the angle of the license plate had the highest confidence of 99% as shown in Fig. 11(c). However, the small sample size does not reflect the overall confidence of the license plate detector application. Future studies could use different training data and backbones, such as ResNet, ImageNet and InceptionNet, to obtain a more comprehensive analysis of the license plate detector.

Acknowledgement

The authors would like to thank the Faculty of Applied Sciences and Technology, Universiti Tun Hussein Onn Malaysia, for its support.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Gan Wei Hang, Lee Siaw Chong;* ***data collection:*** *Gan Wei Hang;* ***analysis and interpretation of results:*** *Gan Wei Hang, Lee Siaw Chong;* ***draft manuscript preparation:*** *Gan Wei Hang, Lee Siaw Chong. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] S. Jitenda,S.P. Khushboo, and S.P. Amit. Comparative Study of Different Techniques for License Plate Recognition. *Journal of Advance Computing and Communication Techonologies,* vol. 1, no. 2, pp. 1-5, Dec. 2013

[2] M. Abdullah, S.M. Al-Nawah,O. Husna, and J. Jasrina. "License Plate Recognition Techniques: Comparative Study," *Malaysian Journal of Computer Science*, vol.1, no.9 pp. 94-105, 2021.
[3] A. Dahich, *Practical Computer Vision: Extract Insightful Information from Images using Tensorflow, Keras, and Opencv*. Packt Publishing, Limited, 2018.
[4] S. Ozbay, and E. Ercelebi. "Automatic Vehicle Identification by Plate Recognition," *World Academy of Science, Engineering and Technology,* vol. 9 pp. 222-225, 2005.
[5] P.D. Smith. *Hands-on Artificial Intelligence for Beginners: An Introduction to AI Concepts, Algorithms, and Their Implementation*. Packt Publishing, Limited, 2018.
[6] W. Liu *et al*., "SSD: Single Shot Multibox Detector," *Lecture Notes in Computer Science*, pp. 21-37, Sep. 2016, https://doi.org/10.1007/978-3-319-46448-0_2.
[7] B. Sahithi, and S. Vigneshwari, "Exploring the Performance of VGG16 and Efficient Net Models for Plant Disease Classification: A Comparative Approach," in *2023 First International Conference on Cyber Physical Systems, Power Electronics and Electric Vehicles (ICPEEV)*, pp. 1-6, https://doi.org/10.1109/ICPEEV58650.2023.10391936.
[8] R. Manikandan, "Sign Recognition - How Well Does Single Shot Multibox Detector Sum Up? A Quantitative Study," in *2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*, pp. 1-13, https://doi.org/10.1109/AIPR.2018.8707409.
[9] J. Sehgal, M. Sharma, J. Chatterjee, and A. Mehra, "Comparative Study of Deep Learning Models for Vehicle Detection in an Unconstrained Road Scenario," In *2020 International Conference on Communication and Signal Processing (ICCSP)*, pp. 1076-1080. https://doi.org/10.1109/ICCSP48568.2020.9182346.
[10] J. Huang *et al*, "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 3296-3297, https://doi.org/10.1109/CVPR.2017.351.
[11] Roboflow Universe Projects, Dec. 2022, "License Plate Recognition Computer Vision Project", Roboflow Universe. [Online] Available: https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e
[12] K. Simonyan, and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognitio," In *The 3rd International Conference on Learning Representations (ICLR2015), Apr.* 2015.
[13] J. Wang, X. Yan, and Y. Cong, "Research on Computer Classification Algorithm of Concrete Crack Based on Deep Learning," *Journal of Electrical Systems*, vol. 20, no. 2, pp.1392-1402, 2024, https://doi.org/10.52783/jes.1376
[14] J.H. Hosang, R. Benenson, and B. Schiele, "Learning Non-Maximum Suppression," In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6469-6477.