# Licence Plate Object Detection and Character Recognition

Christoporus Putratama Muhammad Amrullah Abdul Rukua Muhammad Al-Filambany
University of Southampton
{ cdp1g24, ia2u24, awr1u24, mgaf1n24}@soton.ac.uk

## Abstract

*This document reports the performance of deep learning models such as Faster-RCNN, RetinaNet, and YOLOv5 to detect licence plates. Using the transfer learning method, we used the Kaggle licence plate dataset to train these models. The best result is to use YOLOv5-small, which achieves mAP =* ***74.2%****. The bounding boxes output from these models are used for plate character recognition using PaddleOCR, which achieves* ***85.85%*** *on the single character recognition accuracy.*

## 1. Introduction

Licence plate detection is crucial to surveillance, especially in traffic monitoring, law enforcement, and parking systems. However, plate detection presents a unique challenge due to variation in plate formats, vehicle position, lighting, and orientation.

Research on deep learning and object detection to detect licence plates has made significant advances over the years. In this document, we trained several deep learning architectures such as Faster-RCNN [10], RetinaNet [6], and YOLOv5 [13]. We trained and evaluated the plate detection models using a licence plate dataset from Kaggle [4]. Also, this document reports the accuracy of character recognition on licence plates using PaddleOCR [3].

## 2. Methodology

This study aims to identify the best object detection model for accuracy and precision for detecting car licence plates based on images. This experiment compares four models: YOLOv5, Faster R-CNN, and RetinaNet.

### 2.1. YOLOv5

YOLOv5 is an improved version of the YOLO [8] object detector, a single-stage detector treated as a regression problem. [15] YOLO processes the image with a single forward pass through the network, making it a fast detector, rather than breaking it into multiple stages (bounding boxes, classification, and region proposals).

YOLOv5 is based on YOLOv3 [9] and YOLOv4 [1], which use anchor boxes as an improvement over Faster R-CNN and automatically select their anchor size. Before the training stage, a K-means clustering is performed to determine suitable anchors from the training data. YOLOv5's architecture contains three main parts:

- **Backbone**: YOLOv5 uses CSPDarknet53 [14] as the main body of the network.
- **Neck**: The part that connects the backbone and the model head. YOLOv5 uses the Spatial Pyramid Pooling-Fast (SPPF), which is the enhancement of SPP [5], and the Path Aggregation Network (PANet) [7].
- **Head**: The part that generates the final output, adopted from the YOLOv3 head.

To improve the training result of YOLOv5, several dataset augmentation techniques are applied for this experiment, such as: Mosaic Augmentation, HSV Augmentation, and Random Horizontal Flip.

To train YOLOv5, the loss computed is a combination of three individual losses:

- **Classes Loss**: It is a Binary Cross-Entropy (BCE) loss to measure the error of the classification task.
- **Objectness Loss**: BCE loss to measure error and check if an object is present in a particular cell.
- **CIoU Loss**: Complete IoU loss to measure the error of object localisation inside the grid cell.

The total loss is combined into:

$$Loss = \lambda_1 L_{\text{cls}} + \lambda_2 L_{\text{obj}} + \lambda_3 L_{\text{loc}} \quad (1)$$

Where $\lambda_1$, $\lambda_2$, and $\lambda_3$ are hyper-parameters settings to balance the weight of each loss in the training process.

## 2.2. RetinaNet

RetinaNet is an object detection model designed to address the accuracy gap between one-stage and two-stage object detection models [6]. There used to be a trade-off when choosing an object detection model. Two-stage detectors, such as Faster R-CNN, can be more accurate but slower. Meanwhile, one-stage detectors like YOLO are faster, but less accurate.

The primary reason for the accuracy gap is the extreme class imbalance that occurs when training with one-stage detectors. When the model tries to detect objects, many parts of the image are background, while only small parts of the image are the object of interest. Focal loss is the key innovation in RetinaNet that has solved this issue.

The Focal Loss is defined as:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \tag{2}$$

Where $\alpha_t$ is a balancing factor for addressing class imbalance.

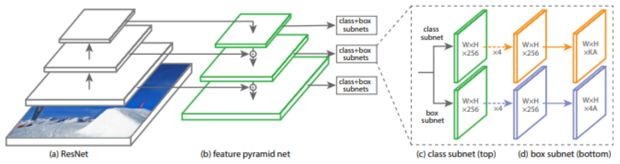

Figure 1. Illustrations of RetinaNet architecture. Reproduced from [6].

There are three main components in RetinaNet: the backbone network, the classification subnet, and the regression subnet. The illustration can be seen from figure 2. The normalised input image will be passed into the backbone network first, typically a ResNet with a Feature Pyramid Network (FPN). ResNet will extract the features at different levels to produce feature maps with many resolutions. FPN will combine high-resolution features and low-resolution features, enabling plate detection at multiple levels. RetinaNet will then create various points, or anchors, in each feature map with a predefined size, according to the object, it tries to detect. In the case of plates, the shape will be rectangular. The function of this point is to create a set of potential detection locations across the image.

In the classification subnet, a small fully connected network (FCN) will be applied to each level in the feature map. In this part, each anchor will use focal loss to predict whether it contains a licence plate. Finally, in the regression subnet, a small FCN will run in parallel with the classification subnet. Each anchor will predict four values representing the offset to adjust the anchor box ($\Delta$x, $\Delta$y, $\Delta$width, $\Delta$height). After RetinaNet obtains the classification score, non-maximum suppression (NMS) is applied to remove overlapping detections.

## 2.3. Faster R-CNN

Faster R-CNN (Faster Region-Convolutional Neural Network) is a Region-Based Convolutional Network that utilises region proposal algorithms and CNN. [10] The architecture consists of three main parts that is the CNN Backbone, Region Proposal Network (RPN) and Fast R-CNN Detector.

### 2.3.1 CNN Backbone

The CNN Backbone is used to extract the input image into a feature map. This feature map is used in both the RPN process and the Detector process.

### 2.3.2 RPN

The RPN receives input of feature map from the CNN Backbone, then it outputs the Anchor Generation and Proposal Scoring. The Anchor Generation are anchor boxes at each point on the feature map. The proposal scoring is the probability the anchor contains an object vs. background and a bounding box adjustments. Unlike previous methods that use Selective Search, this method uses neural networks to generate the score.

### 2.3.3 Detector

The last step is a Detector. The detector receives inputs of the proposed region from RPN and the feature map from CNN Backbone. It then takes each proposed region (RoI) and crops/resizes its features from the backbone's feature map into a fixed size. It also passed the pooled feature through a Fully Connected Network to predict classification.

## 2.4. Text Recognition with PaddleOCR

PaddleOCR is an open source OCR system that supports multiple languages and is used in various applications, including licence plate recognition [3]. The overall pipeline of PaddleOCR consists of three main steps: text detection, direction classification, and text recognition. In the first stage, text detection locates textual regions in an image using differentiable binarisation (DB). Next, a direction classifier determines the orientation of the detected text boxes, which are then rotated into horizontal rectangles. In the final stage, text recognition is performed using a CRNN combined with CTC loss, converting aligned text regions into digital characters with high accuracy.

### 2.5. Metrics and Evaluation

To evaluate the performance of the licence plate detection model, we used mean Average Precision (mAP) on IoU = 0.5, and mAP on IoU from 0.5 to 0.95 [10], which is also used on COCO and Pascal VOC metrics.

## 3. Experiment

### 3.1. Dataset

For this experiment, we are using the License Plate Dataset [4] from Kaggle. This dataset consists of 1,695 colour car licence images, divided into two folders: train and val. The images are in different sizes and have diverse times, lighting, and locations of cars. This makes this dataset suitable for real-world case licence plate detection.

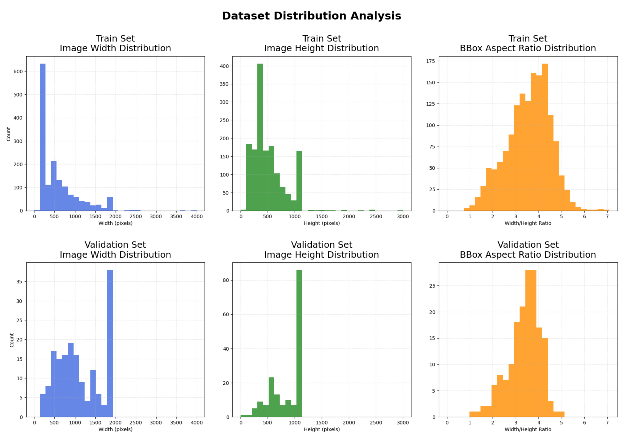

Figure 2. The histogram of Image properties for both training and validation dataset.

In the training dataset, the image dimensions range from the largest size of 4,000 × 3,000 pixels (width × height) to the smallest size of 78 × 87 pixels. For the val, the largest dimension ranges from 1920 x 1080 pixels to 22 x 8. The average dimension for the training dataset is 603 x 521 pixels. For the validation, the average dimension is, 1119 x 853 pixels. The bounding box aspect ratio for the training data consists of 70.4% of the standard licence plate aspect ratio (2.5-4.5). The bounding box aspect ratio for the val data consists of 84.6% of the standard plate aspect ratio.

Each image in the dataset has also been manually annotated with bounding box coordinates for the licence plates in YOLO format. The dataset was then combined and randomly divided into train, validation, and test using 7:2:1 setting to make the evaluation objective.

### 3.2. Pre-processing

For pre-processing, each model has a different approach to ensure the best result for each model.

#### 3.2.1 YOLOv5

For YOLOv5 [13], the following values are the probabilities that the preprocessing will be applied randomly, on a scale of 0 to 1. The parameters are as follows: the hue, the saturation, and the brightness shifts are 0.015, 0.7, and 0.4, respectively. For hue and saturation, the shifts range from -0.5 to 0.5. This means that a value of -0.5 and +0.5 is the same, since they are circular values. The translation and scale are 0.1 and 0.5, which means that the images are translated between -10% to +10% of their initial position and -50% to +50% of their original size. The horizontal flip augmentation probability is 0.5, and the mosaic augmentation is always used with a probability of 1.

#### 3.2.2 RetinaNet

In RetinaNet, the images are resized to 320x320 to standardise their format. The images were then normalised using the ImageNet [2] statistics (mean = [0.485, 0.456, 0.406], std [0.229, 0.224, 0.22]). This number is applied because the architecture backbone ResNet50 is trained on ImageNet. Data augmentation is applied during model training by flipping the image with a 50% probability and adjusting the brightness or contrast with a 20% probability by -0.2 to +0.2. A negative value will make the image darker and flatter, while a positive value will make the image brighter and have more pronounced contrast between light and dark areas. The YOLO annotation was also converted into PASCAL VOC format [x1, y1, x2, y2] where x1, y1 is the coordinate of the top-left corner and x2, y2 is the coordinate of the bottom-right corner.

#### 3.2.3 Faster-RCNN

The images in the training dataset are transformed with random horizontal flip with a probability of 50%, this is to prevent overfitting as well artificially increases the size of the training dataset.

## 4. Experiment

### 4.1. YOLO-v5

The model used for this experiment is YOLOv5-small. The YOLOv5-small pretrained model was used to train this model faster using the transfer learning method. No parameters were frozen during the training process. We converted the number of class parameters from 80 to 1 to accommodate a single class detector that detects only the licence plate.

The hyperparameters for YOLO-v5 training are as follows: the initial learning rate is 0.01, the final learning rate is 0.01 of the initial learning rate, the momentum for the SGD optimiser is 0.937, and the L2 regularisation, also known as weight decay, for the optimiser is 0.0005. At the initial training, a warm-up phase was set to stabilise the model training. The warm-up epochs, warm-up momentum, and warm-up bias parameter are 3.0, 0.8, and 0.1, respectively. Regarding training losses, the weights for class loss ($\lambda_1$), object loss ($\lambda_2$), and localisation loss ($\lambda_3$) are set at 0.5, 1, and 0.05, respectively. An IoU threshold of 0.2 is used, and the anchor ratio for the bounding box is set to 4.0. The training is conducted up to 10 epochs, and the training batch size is 32.

### 4.2. RetinaNet

The backbone we are using for the RetinaNet model is ResNet50 with Feature Pyramid Network (FPN). The anchor sizes we use in this model are 32, 64, 128, 256, and 512 pixels. The aspect ratios are 0.5, 1.0, and 2.0. This means that for each predicted location, the model will generate 15 different anchors (5 sizes x 3 aspect ratios). The sizes represent the square root of the anchor area. The aspect ratio represents the width-to-height ratio of the anchors. The gamma applied in focal loss is 2. Gamma in the focal loss function controls the rate at which the easy examples are down-weighted. For the hyperparameter, the optimiser used is Stochastic Gradient Descent (SGD) with a momentum of 0.9. The weight decay is 0.0005. The initial learning rate is 0.001 with a step scheduler decrease factor of 0.1 for every 8 epochs. The batch size is 2. For the first 1000 iterations, the learning rate increases linearly. The model is trained with 10 epochs, but the final model used for test results is based on the latest epochs with the lowest validation losses.

### 4.3. Faster-RCNN

With Faster-RCNN, the backbone model used is a pre-trained MobileNetV2 [11]. This model is chosen because of its size and fitness with the available GPU. the models are trained with an image size of 320x320, however Faster-RCNN can work with any image size input because of the sliding window operation. The model backbone output channel parameter is 576. The anchor size chosen is (16,32,64,128). Licence plate sizes are small, therefore a small anchor size will help in detecting the licence plate. The aspect ratios are (0.5, 1.0, 2.0) based on the EDA of the dataset. For the RoI pooling parameter, the pooling size chosen is (7, 7) which is standard used on the original implementation, as for the sampling.

### 4.4. PaddleOCR

In the initial experiment, we conducted an evaluation on our three models: Faster R-CNN, YOLOv5, and RetinaNet. Each model was tested both with and without padding, using PaddleOCR v4 with pre-trained weights. All evaluations were conducted on the test set. The objective was to analyse the distribution of detection confidence scores and to identify the number of licence plates that could not be successfully extracted. Based on the OCR performance results, the best-performing detection model was selected for further experimentation. Subsequent experiments focused on applying four distinct preprocessing techniques, motivated by [12]. The experiments were conducted under six conditions: (1) greyscale only, (2) contrast only, (3) with padding, (4) without preprocessing, (5) all preprocessing excluding inversion, and (6) all preprocessing. To assess the impact of preprocessing techniques, we used three metrics: plate-level accuracy (correctly recognised full plates), character-level accuracy (percentage of correct characters), and average edit distance (average number of edits needed to match OCR output to ground truth).

## 5. Results

### 5.1. Result of Licence Plate Detection

The detection result is shown in Figure 3, and the detection performance of this experiment is shown in Table 1.

| **Model** | **Size** | **mAP@IoU=0.5:0.95** | **mAP@IoU=0.5** |
|---|---|---|---|
| YOLOv5-s | 320x320 | **74.2%** | **98.9%** |
| RetinaNet | 320x320 | 69.4% | 94.1% |
| Faster-RCNN | 320x320 | 50.4% | 98.1% |

Table 1. The result of the licence plate detection models on the test set.

### 5.2. Performance of PaddleOCR

In the initial results without any preprocessing or padding, RetinaNet achieved the highest average OCR confidence at 82.39%, with 17 undetected licence plates. YOLOv5 followed with an average confidence of 81.25% and 19 undetected plates, while Faster R-CNN showed the weakest performance with 78.79% confidence and 23 undetected plates. In contrast, after padding was applied, all models demonstrated improved OCR performance. YOLOv5 achieved the highest average confidence

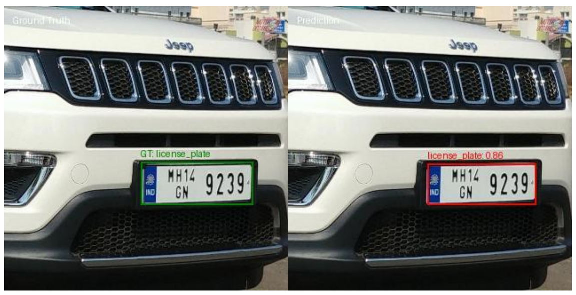

Figure 3. The visualisation of the bounding box result of the licence plate using RetinaNet (green: ground truth, red: RetinaNet).

| Preprocessing | Plate Acc (%) | Character Acc (%) | Edit Distance |
|---|---|---|---|
| No preprocessing | 41.76 | 77.54 | 2.37 |
| Padding Only | 41.76 | 83.93 | 1.82 |
| Grayscale | 43.53 | 83.60 | 1.85 |
| Contrast | 36.47 | 79.52 | 2.31 |
| No Invert | 15.29 | 62.81 | 4.03 |
| All Preprocessing | 21.18 | 63.78 | 3.91 |

Table 2. The performance of PaddleOCR on Different Preprocessing on YOLOv5-s.

at 85.85%, with only 9 undetected plates. RetinaNet recorded 84.96% confidence and 11 undetected plates, followed by Faster R-CNN with 81.60% confidence and 18 undetected plates. Based on these results, YOLOv5 was selected for further evaluation to assess the impact of various preprocessing techniques. As shown in Table 2, applying padding and greyscale conversion noticeably improved both plate-level accuracy and character-level accuracy compared to the baseline without preprocessing. In contrast, other preprocessing techniques tended to decrease overall accuracy, indicating that not all enhancements contribute positively to OCR performance.

## 6. Discussion

From the licence plate detection experiment, YOLOv5-s performed the best result (mAP = 74.2%) on the same input size as other architectures. YOLOv5-s used significant augmentations during the training process, such as random mosaics, which add more variance to the network. RetinaNet can compete with other models with mAP = 69.4%, lower than YOLOv5-s, but achieved a higher result in OCR without preprocessing. This result can still be improved by adding more epochs to training and adding more types of data training augmentation, like HSV colour shift, scaling, and mosaic augmentation, just like in YOLOv5-s, while Faster-RCNN performed the least with mAP = 50.4%.

Faster-RCNN requires a large GPU to train and has a large model size of 300 MB. However, it can receive any input size as long as it is more than the pooling size of $7 \times 7$. When combined with PaddleOCR, the detection models produced varying OCR accuracies. Although RetinaNet performed best without any preprocessing, YOLOv5 benefited more from post-detection enhancements such as padding.

## 7. Conclusion

We have successfully performed the licence plate detection and recognition. We used several object detection methods using deep learning to localise the plate and create a bounding box around a licence plate. From the bounding boxes, we can extract the region of the plate and perform character recognition on the plate, although the overall performance of the plate recognition is not good.

## References

[1] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[3] Yuning Du, Chenxia Li, Ruoyu Guo, Xiaoting Yin, Weiwei Liu, Jun Zhou, Yifan Bai, Zilin Yu, Yehua Yang, Qingqing Dang, and Haoshuang Wang. Pp-ocr: A practical ultra lightweight ocr system, 2020.

[4] Ronak Gohil. License plate dataset. `https://www.kaggle.com/datasets/ronakgohil/license-plate-dataset/data`, 2023.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[6] Tsung-Yi Lin, P. Goyal, R. Girshick, Kaiming He, and P. Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318 – 27, 2020/02/.

[7] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[8] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[9] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-CNN: Towards real-time object detection with region proposal networks. 39(6):1137–1149. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[11] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[12] Renato Augusto Tavares. Comparison of image preprocessing techniques for vehicle license plate recognition using ocr: Performance and accuracy evaluation, 2024.

[13] Ultralytics. YOLOv5: A state-of-the-art real-time object detection system. `https://docs.ultralytics.com`, 2021.

[14] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[15] Yu Zhang, Zhongyin Guo, Jianqing Wu, Yuan Tian, Haotian Tang, and Xinming Guo. Real-time vehicle detection based on improved yolo v5. *Sustainability (Switzerland)*, 14, 10 2022.