SEMARAK ILMU PUBLISHING

Journal of Advanced Research in Applied Sciences and Engineering Technology

Journal homepage:
https://semarakilmu.com.my/journals/index.php/applied_sciences_eng_tech/index
ISSN: 2462-1943

Instance Segmentation Evaluation for Traffic Signs

Shi Heng Siow[1], Abu Ubaidah Shamsudin[1,*], Zubair Adil Soomro[1], Anita Ahmad[2], Ruzairi Abdul Rahim[2], Mohd Khairul Ikhwan Ahmad[3], Andi Adriansyah[4]

[1] Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia, Batu Pahat, Johor, Malaysia
[2] School of Electrical Engineering, Faculty of Engineering, Universiti Teknologi Malaysia, Skudai, Johor, Malaysia
[3] Cybersolution Technologies Sdn Bhd. Johor Bahru, Johor, Malaysia
[4] Universitas Mercu Buana, Jakarta, Indonesia

ARTICLE INFO

Article history:
Received 30 June 2023
Received in revised form 15 November 2023
Accepted 24 November 2023
Available online 8 December 2023

Keywords:
Instance Segmentation; Traffic Sign Recognition; YOLACT; Deep Learning Approach; Image Segmentation

ABSTRACT

This research paper focuses on developing a traffic sign recognition system based on the You Only Look At Coefficients (YOLACT) model, a one-stage instance segmentation model that offers high performance in terms of accuracy and reliability. However, the YOLACT performance was influenced by various conditions such as day/night and different angles of objects. Therefore, this study aims to evaluate the impact of different angles and environments on the system performance. The paper discusses the framework, backbone structure, prototype generation branch, mask coefficient, and mask assembly used in the system. ResNet-101 and ResNet-50 were used as the backbone structure to extract feature maps of objects in the input image. The prototype generation branch generates prototype masks using fully convolutional networks (FCN), and the mask coefficient branch generates the Mask assembly using the sigmoid nonlinearity. Two models, YOLACT and Mask-RCNN, were evaluated by mean precision (mAP) and frames per second (FPS) with the front view dataset. The results show that YOLACT outperforms Mask-RCNN in terms of accuracy and speed. For an image resolution of 550x550, YOLACT with Resnet-101 is considered the best model in this article since it achieves over 80% precision, recall, specificity, and accuracy in various conditions such as day, night, left and right, and forward-looking angles.

1. Introduction

Automation using autonomous systems is becoming increasingly popular across different industries, with its ability to execute programs or control operations on machines [1]. Autonomous driving, one of the most famous research areas, has become a widely used product worldwide. In this area, the level of autonomy is classified in five levels (1-5 levels) by the Society of Automotive Engineers (SAE) from driver assistance to full automation [2]. To improve the autonomous driving system, research and development (R&D) teams seek to achieve SAE level 5 [3]. They collect and store relevant sensory data, analyze data sets, and interpret the results as a production control

* *Corresponding author.*
E-mail address: ubaidah@uthm.edu.my

https://doi.org/10.37934/araset.34.2.327341

system to eventually reach a state where an autonomous car can sense information and accurately navigate the road.

Despite the advancement in self-driving cars that use algorithms to collect data from sensors, artificial intelligence (AI) could be integrated into the machine to achieve flexibility and effectiveness [4]. Deep learning approach is one of the most prominent research and development in vehicle autonomy, requiring continuous application and development of expertise in traffic signs. Using deep learning approaches such as YOLACT improves drivers' ability to comprehend road signs while driving.

This study aims to address the problem of drivers' inability to read and interpret road signs correctly. This is highlighted as it can lead to road accidents and fatalities. The parameters that cause such various factors include wear and tear, poor observation, and environmental conditions such as too much light or too much darkness [5-7]. To address this, recent research has shown that the integration of computer vision and deep learning can be used to assist drivers in recognizing traffic signs [8]. However, conventional deep learning algorithms have shown decent results, but they were sometimes time-consuming and discontinuous in following frames [9].

This study seeks to use YOLACT to increase the driver's ability to comprehend road signs while driving in the UTHM (Universiti Tun Hussein Onn Malaysia) and improve the recognition of road signs in adverse settings such as strong or dim illumination, wear, and deterioration. The research also intends to investigate the effects of variable features such as the visual brightness of objects in an image on YOLACT's performance. The objectives of this research are to develop instance segmentation utilizing the YOLACT at UTHM in Malaysia, compare Frame Per Second (FPS) and mean Average Precision (mAP) on the front traffic signs between the Mask-RCNN and YOLACT, and evaluate the reliability of the traffic sign right-left view, front view, day view, and night view provided by YOLACT.

2. Literature Review

The traffic sign recognition system generally includes two stages: detection and classification. Traffic sign detection is used to obtain inherent information (such as color or shape) and other natural information from images. After that, features are extracted from actual candidate regions. Using a recognition system, it plays a role in different types of traffic sign classification tasks. For example, the Histogram of Oriented Gradients (HOG) features for dimensionality with Iterative Nearest Neighbours-based Linear Projections (INNLP) and classification with Iterative Nearest Neighbours (INNC) reached 98.53% accuracy for traffic sign images [10]. Other machine learning algorithms used in traffic sign recognition include SVM classifiers with HOG feature [11], MLP (Multi-Layer Perceptron) with the radial histogram features [12], SVM with LIPID (local image permutation interval descriptor) [13], CNN etc. [14,15]. However, these methods could still be improved by data augmentation and the application of multiple CNNs, as these usually result in higher memory and computational costs.

In segmentation, the basic structure of object recognition is performed using a hierarchical method. In the concept of neural network, the basic structure is described as three layers, namely the input layer, hidden layer and output layer. The various methods used in segmentation all produce different results in terms of time to identify the objects in the image and structure in the architecture [16].

Object detection has many types of methods for object segmentation such as semantic segmentation [17] and instance segmentation [18]. Semantic segmentation treats multiple objects within a single category as one entity, and instance segmentation identifies individual objects within these categories. For example, instance segmentation can be used for any object such as a detected vehicle [18]. It is used in Artificial Intelligence for object recognition and is a usual method for object

detection in neural networks, a branch of Convolutional Neural Networks (CNN) with machine learning or deep learning. R-CNN (regions with convolutional neural networks) combines regional interest with CNN. There are several methods for generating recognition categories in detected objects' shadow areas. Regional suggestions include object [19], selective search [20], and category-independent object suggestions [21]. On the other hand, R-CNN is unable to comprehend the specific region proposal method. As a result, a selective search method is employed in order to achieve a comparison with previous object detection results. To further improve R-CNN, some functions can be combined to improve performance in detection or training speed, such as Mask R-CNN [22] and Fast R-CNN [23]. For example, Fast R-CNN selects the SoftMax with boundary box for the generating output step.

The instance segmentation approach used in object recognition has two major architectures: one-stage instance segmentation and two-stage instance segmentation. The instance segmentation approaches have been widely used in object detection systems. For example, two-stage segmentation [24] works in two steps. The first step generates the bounding box by using a regional proposal, and the second step classifies each of the region proposals by using a classifier such as the ResNet . In addition, the one-stage segmentation [25] is lightweight and uses many samples to classify various images at different scales and ratios. These provide functionality such as object localization and classification.

Compared to two-stage segmentation, one-stage segmentation generally has higher speed but lower accuracy. For example, Mask-RCNN [25] is a two-stage segmentation method that achieves higher accuracy than one-stage methods. However, two-stage segmentation has limitations in terms of speed, making it difficult to achieve real-time segmentation (<30FPS). On the other hand, one-stage segmentation is lighter and faster, and is suitable for linear combination. YOLACT [26] is a lightweight option that achieves real-time instance segmentation at 30FPS.

Different methods are used in one-stage segmentation to improve accuracy, such as location-sensitive pooling [27,28] or methods that combine semantic segmentation and direction prediction logic [29]. However, the limitation of one-stage segmentation still lies in its lower accuracy compared to two-stage segmentation. The structure, layout, and results of one-stage segmentation can vary depending on the method used in the first-level segmentation.

According to Figure 1, compared with other methods (such as Mask R-CNN) used in the same device and MS COCO [30], YOLACT is the fastest frame per second (FPS). For the fps and mean Average Precision (mAP) results in the article by Bolya, Daniel, *et al.,* [31] YOLACT obtained 29.8 mAP and 33.5 fps. For YOLACT, it is possible to achieve real-time instance segmentation in a one-stage. The YOLACT system structure is shown in Figure 2. It starts with the FPN method, which uses a different backbone for constructing feature extraction. These are used to generate prototype generation branches (protocols) with ReLU [32] and mask coefficients, and then merge them for mask assembly. Then, many parameters used on the score threshold filter the image, and these parameters become the output image and are displayed as segmented regions in the object.

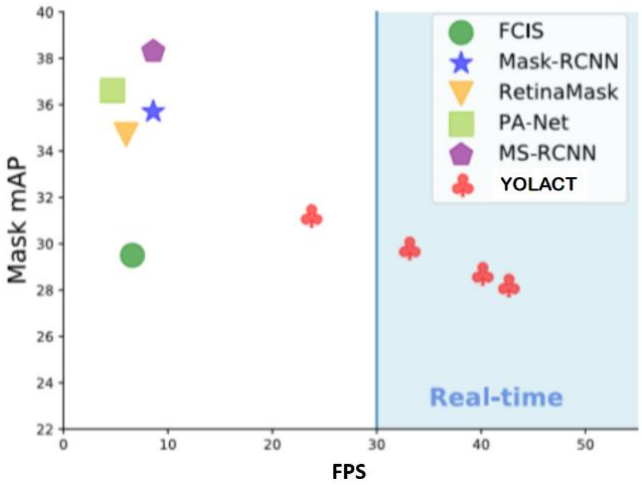


Fig. 1. Comparison methods in term of frame per second (FPS) [14]

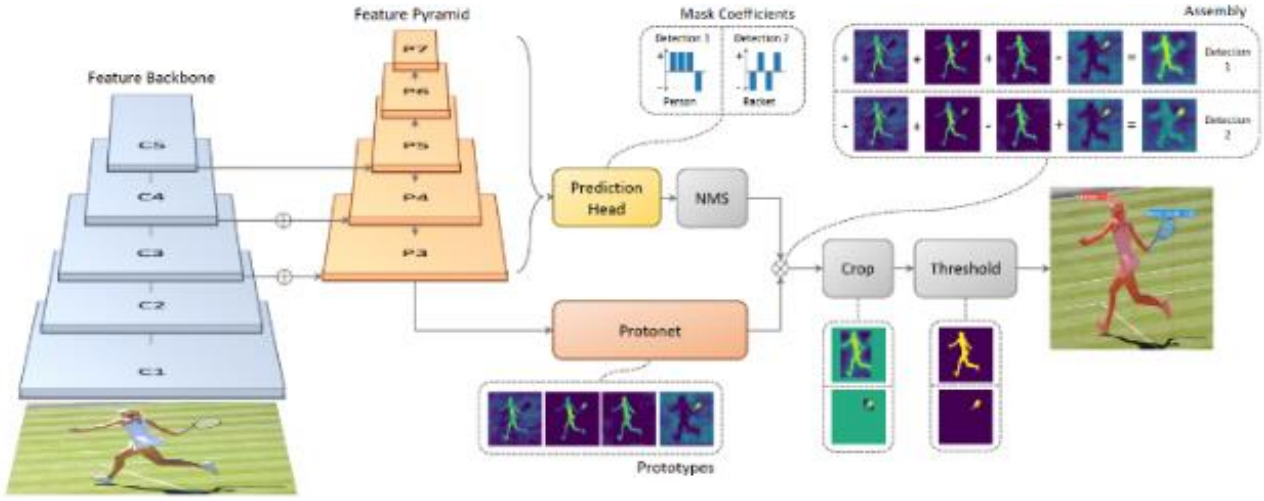


Fig. 2. YOLACT architecture

A real-time instance segmentation algorithm, in different domains has been utilized such as marine situational awareness, metal screw defect detection, teeth and facial landmark segmentation, railroad track inspection, railway catenary fastener detection, and instance segmentation accuracy improvement.

In marine situational awareness [33,34], YOLACT was used to recognize ships' classes and geographic locations in real-time from static oblique view images. The results showed that YOLACT had a faster FPS than DetectoRS, but a lower performance of overall mAP. In metal screw defect detection, YOLACT was used to detect surface defects of screw heads, and the results showed high accuracy and reliability with a detection result of 40.11 mAP box and 41.37 mAP mask.

In the domain of teeth and facial landmark segmentation [35,36], YOLACT was evaluated with Mask R-CNN, MS R-CNN, and D2Det, and it obtained a 0.50 mAP and 0.55 mAR. However, MS R-CNN achieved the highest performance at 0.67 mAP and 0.69 mAR.

In railroad track inspection [37], YOLACT was used with ResNet-50, ResNet-101, Res2Net-50, and Res2Net-101, and Mask R-CNN for real-time instance segmentation of railway tracking components. The results showed that Mask R-CNN obtained the highest mask coefficient at 63.9mAP, while Yolact-Res2Net-101 obtained the highest bbox at 59.9mAP.

In railway catenary fastener detection [38], YOLACT was used with Reinforcement Learning and F-RCNN to perform semantic segmentation for the part of looseness of fasteners. The YOLACT obtained the best mask mAP of 0.975 and less than 1.1 FPS compared with YOLACT++ for segmentation performance.

Finally, in instance segmentation accuracy improvement [39], YOLACT was designed with multiple feature extraction and FPN to enhance instance segmentation. The results showed that ResNetSt50 has 49.66 mAP higher than other feature extraction methods.

Traffic signs on the road have a significant impact on drivers' decision-making during driving. According to Bazire *et al.,* [40], most road traffic accidents occur due to non-compliance with traffic signs. Under normal circumstances, the frequency of traffic volume during the day is higher than at night, but there are more road accidents at night than during the day. This is due to reasons such as reduced light levels [41].

3. Methodology

This section describes the main framework, backbone structure, Prototype Generation branch, mask coefficient and mask assembly.

3.1 Main Framework

This paper presents a traffic sign recognition system based on YOLACT, a real-time one-stage instance segmentation model. YOLACT's instance segmentation architecture is similar to that of Mask-RCNN and Faster-RCNN, but it processes pixel-level information directly on the image, without using the repooling operation.

YOLACT performs two main tasks: (1) generating prototype masks and (2) predicting the masking coefficient of each instance in each layer. The architecture starts with Feature Pyramid Networks (FPN) and feature backbones to extract object features from images (Figure 3). Then, the prototype generation branch (protonet) generates prototype masks using fully convolutional networks (FCN) at a fixed image size and uses the largest image size layer.

In addition, YOLACT predicts masking coefficients between each layer of FPN using a prediction head and non-maximum suppression (NMS) method, where each anchor encodes the representation of the instance in the prototype space. Finally, the protonet and mask coefficient branches are combined to form a Mask assembly.

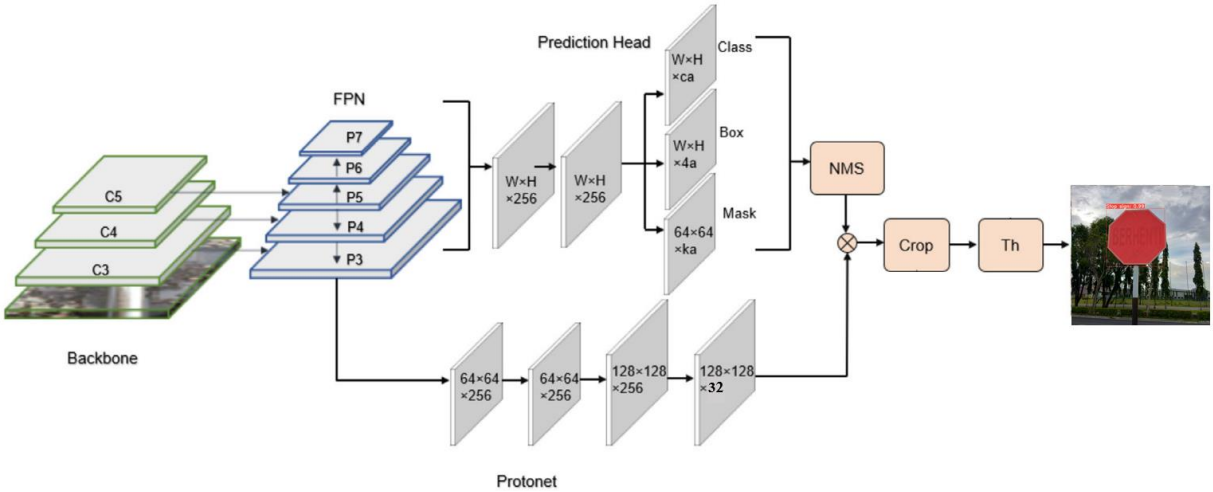


Fig. 3. The main structure of YOLACT [25]

The most significant step in the development of the traffic sign segmentation was image preparation. It consists of image acquisition, resizing, and identification which affects the accuracy of objects and the reliability of data analysis. The traffic sign images were captured in different conditions, including day, night, frontal, and right-left view in Universiti Tun Hussien Onn Malaysia (UTHM) premise. Capture an image from the same smartphone camera with 1.5 m distance, same angle and adjust its size to 1:1 for all the images. The captured image displayed the resolution as 3456 x 3456 pixels. The number of traffic signs collected for traffic signs totaled 512 images.

3.2 Backbone Structure

The backbone structure acts as a feature extractor to obtain feature maps of objects in the input image. After modifying or tuning the model, each structure is developed to meet specific requirements for detection accuracy and efficiency. The accuracy of models has greatly improved with the use of ResNet in the backbone structure. In lightweight architectures, MobileNet [42] can be used as in this article [43]. In YOLACT, ResNet structures were employed to enhance the performance of the models, including YOLACT-ResNet-50 and YOLACT-ResNet-101.

ResNet is a convolutional neural network backbone structure. Each ResNet series has a unique set of convolutional layers, and the performance of each ResNet series is determined by the number of these layers. For example, ResNet-50 has 49 convolutional layers between conv1 and conv5_x, as well as one average pooling layer, while ResNet-101 has 100 convolutional layers between conv1 and conv5_x, and one average pooling layer. In YOLACT, ResNet-50 was found to be more accurate than other backbone architectures, such as VGG-16 [43]. ResNet consists of five stages: Conv1, Layer1, Layer2, Layer3, and Layer4, corresponding to C1 to C5. This experiment employed two ResNet methods: ResNet-50 and ResNet-101, both of which were obtained from YOLACT.

The bottleneck structure depicted in Figure 4 was designed for ResNet-50 and ResNet-101. As shown in Figure 4, the bottleneck structure begins with a 1x1 convolutional layer in the initial stage and ends with another 1x1 convolutional layer for inferring image resolution.

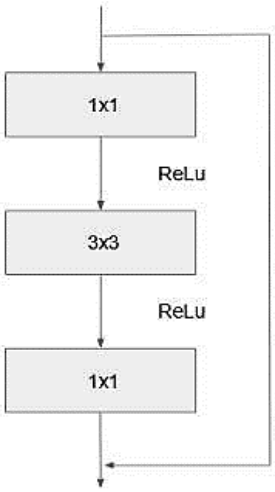


Fig. 4. Bottleneck of structure [9]

3.3 Prototype Generation Branch

The prototype generation branch (protonet) was a fully connected network attached to the P3 layers in the feature Pyramid Networks (FPN) [44]. The FPN received information from the output layers of the backbone structure to generate protonet and prediction head. Corresponding to the FPN and backbone structure, the output between C3 to C5 was used to analyze data for [P3, P4, P5]. In YOLACT type models, P5 was upsampled to P6 and P7 with one-fourth dimensions to increase detection performance on small objects, while P2 was omitted. In the last layer of the FCN, k prototype masks were generated for the entire image. Each of the k channels represented a prototype.

Unlike RetinaNet, NMS performs mask coefficients between each layer (P3 to P7) and this system produces three branches in the predicted head: one branch was the predicted class confident as c, one of the branches was four bounding box regressors, and one branch was mask coefficients as k prototype. Therefore, the equation forms to 4+c+k.

To generate mask prototypes, k channels were used to attract the layers P3, as shown in Figure 3. The value of k=32 channels was used for the last layers of the FPN to perform instance segmentation. The Rectified Linear Unit (ReLU) nonlinear activation function was used to unbind the prototype net and generate prototypes to make them more interpretable. Furthermore, the number of prototype masks was independent of the number of categories, distributing the representation of the generated prototypes.

3.4 Mask Coefficient and Mask Assembly

Based on the protonet branch and predicted head branch in Figure 3, these were combined to generate the mask assembly. The generated prototypes were removed by using the tanh activation function. The masks were created by linearly combining the protonet output and mask coefficients. The final masks were created using the sigmoid nonlinearity. This process is illustrated by Eq. (1):

$$M = \sigma P C^T \tag{1}$$

where M is the mask assembly, P denotes an h × w × k matrix of prototype masks, and C denotes an n × k matrix of mask coefficients for n instances that survived NMS and score thresholding.

Feng Gou et al., [45] describes how YOLACT trained different backbone weights using various hyperparameters. The initial learning rate, momentum, iteration, and batch size were kept constant throughout when training models with different backbones. Two image resolutions were used: 512x512 pixels and 550x550 pixels. The batch size used was 2. Table 1 displays the hyperparameters considered and their default values during the different phases.

Table 1
Considered hyper-parameters and the default

Parameters	Default value
Initial learning rate	10-3
Momentum	0.9
Iterations	10k

4. Result and Discussion

In this section, an experiment was conducted to evaluate the applied method. Firstly, two models were trained and tested on the same dataset, and their performances were assessed. Then, the performance of the two adopted methods, YOLACT and Mask-RCNN, were compared using the training sets of these datasets to illustrate their usefulness. To evaluate the performance of the two models, mean Average Precision (mAP) and frames per second (FPS) were calculated. Additionally, YOLACT was used to test four different conditions, including front view, day, night, and right-left view conditions.

4.1 Dataset

This segment provides a brief overview of the dataset used in YOLACT, including the resizing, labeling, and file format.

The COCO format is widely used for custom datasets in large-scale object detection and segmentation evaluations. For this dataset, a ratio of 0.6:0.2:0.2 was used to split the dataset into training, validation, and testing sets [46]. For the front view dataset, 320 images were split into 192 for training, 64 for validation, and 64 for testing. The same number of untrained images were used for testing for the front view, daytime, left-right view, and night-time datasets.

To prepare the training and validation datasets for both models, images were resized to a resolution of 550x550 pixels and 512x512 pixels before completing the annotations file. The Labelme tool was used to produce the COCO format in a JSON file for the training, validation, and testing datasets. The background category was labeled as 0, Parking Lot sign as 1, No Entry sign as 2, Stop sign as 3, and Name sign as 4.

4.2 Labeling Tool

To understand the masking process of the images used, object labeling was performed. The Labelme software was executed in Ubuntu 18.04 operating system where a Graphical User Interface appears to selectively label each object, as shown in Figure 5. Figure 6 illustrates the original jpg image before and after the instance label visualization process on the Labelme tool.

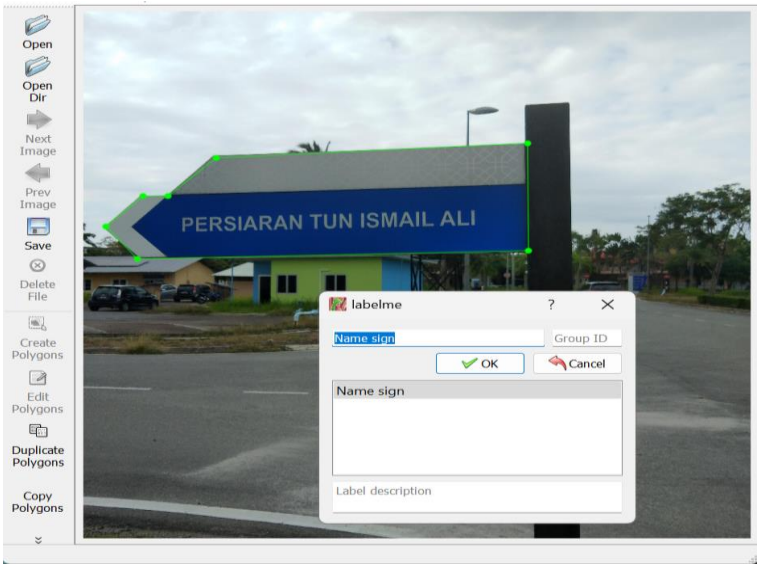


Fig. 5. Label different types of road signs

(a) (b)

Fig. 6. Original jpg image process on LABELME tool: (a) before instance label visualization; (b) after instance label visualization

4.3 Implementation Details

The implementation was carried with the YOLACT and Mask-RCNN models executed on an Nvidia GeForce 950m GPU and an Intel Core i7-7700 CPU. For the YOLACT model, the hyperparameters were set as described above, except for the batch size which was set to 2 to fit within the GPU's memory capacity. The momentum and iteration were set to 0.9 and 10000, respectively, and the initial learning rate was 0.001. The input size was varied between 550x550 pixels and 512x512 pixels. The PyTorch library provided by Facebook, along with the CUDA and CuDNN packages developed by NVIDIA, were used to configure, and accelerate the model training process. For the Mask-RCNN, the default settings were used, with a resolution of 512x512 pixels.

4.4 Evaluation of YOLACT and Mask-RCNN

The performance of YOLACT and Mask-RCNN models was evaluated based on their accuracy in detecting traffic signs for the front view dataset using the COCO mAP (mean average precision) metric. Before evaluating the mAP, the intersection over union (IOU) and average precision (AP) were calculated using a confusion matrix that included True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) values. The IOU measured the overlap between the ground truth and predicted boundary, and a prediction was considered accurate if the IOU was greater than 0.5. The AP was calculated by averaging precision across all recalls that ranged from 0 to 1. Precision, recall, specificity, and accuracy were calculated based on the confusion matrix using Eq. (2), (3), (4), and (5), respectively. The YOLACT model used ResNet-101 backbone weights, while the Mask-RCNN used ResNet-50 backbone weights.

$$Precision = TP/(TP + FP) \tag{2}$$

$$Recall = TP/(TP + FN) \tag{3}$$

$$Specificity = TN/(TN + FP) \tag{4}$$

$$Accuracy = (TP + TN)/(TP + TN + FP + FN) \tag{5}$$

For the Average Precision, it calculated the equation for the value of the area of the precision-recall curve. The precision-recall (PR) curve was the plot of the precision value as a function of the recall value. It shows the trade-off between the two metrics for varying confidence values for the model detections in the Eq. (6) and has shown below.

$$AP_a = \int_0^1 PdR \tag{6}$$

where P was precision and R was recall. Each AP was calculated individually for each class. The values of the AP averaged to become the mean Average Precision. It was the mean of the average precision overall class and has shown below Eq. (7).

$$mAP = \frac{1}{n}\sum_{i=n} AP_i \tag{7}$$

where the n was the number of classes. Meanwhile the frames per second in this model use the following Eq. (8).

$$FPS = 1/(Time\ average\ process\ for\ each\ image) \tag{8}$$

Figure 7 presents a concise view of the results, with YOLACT outperforming Mask-RCNN in terms of speed. In terms of performance on the training and validation datasets, Mask-RCNN achieves higher performance compared to YOLACT, obtaining 100% mAP. However, YOLACT achieves a different range of performance between 0 and 4% for each category. For the validation and testing datasets, all models show a drop in performance, ranging from 0% to 12.5%. Mask-RCNN experiences the highest drop, dropping from 100% to 87.5% between the validation and testing datasets. On the other hand, YOLACT-ResNet-50 at 512x512 pixels has the lowest drop of 0.26% between the validation and testing datasets. For the test dataset performance, YOLACT-ResNet-101 at 512x512 pixels has the highest mAP performance of 96.52%, while the lowest mAP of 87.50% is observed for Mask-RCNN-ResNet-50. With a ResNet-50 backbone and an image size of 550x550 pixels, YOLACT achieves the highest FPS of 5.06 and a mAP of 92.53% in the test dataset. This performance is 4% better for FPS than the 512x512 pixel resolution but 0.66% lower for mAP. With a ResNet-101 backbone and an image size of 550x550 pixels, YOLACT achieves the highest FPS of 3.25 and a mAP of 94.99% in the test dataset. In comparison to the 512x512 pixel resolution, it performed with a 2% increase in FPS but a decrease of 1.53% in mAP. In contrast, Mask-RCNN's results in Table 2 are the lowest, with 0.78 FPS and 0.68 FPS for ResNet-50 and ResNet-101, respectively.

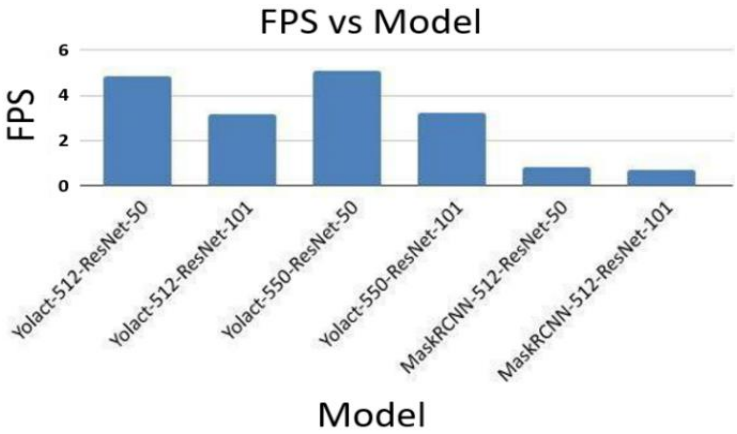


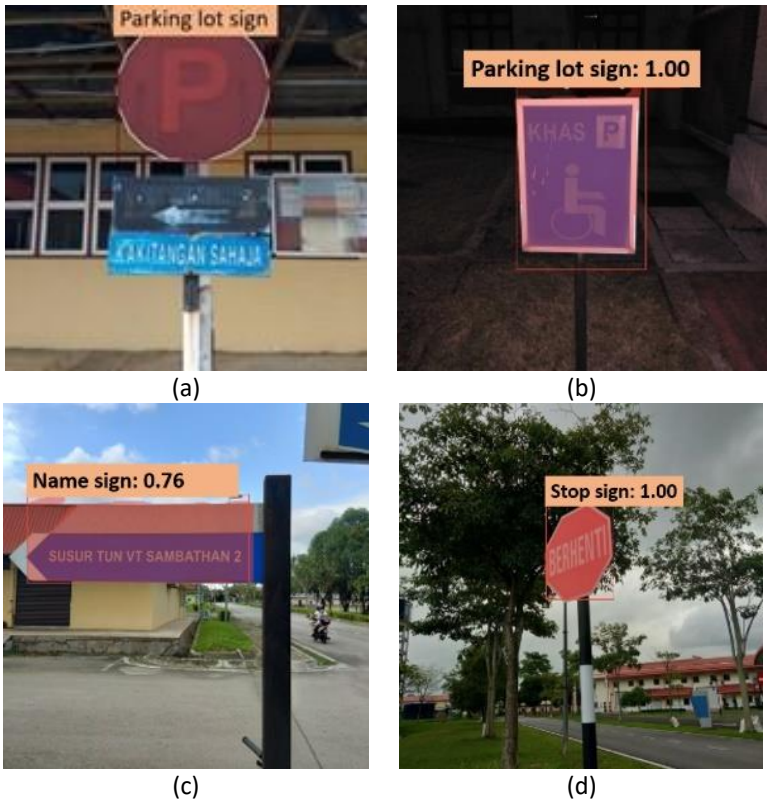
Fig. 7. The FPS against the model used

Table 2
YOLACT and MASK-RCNN for mask mAP on traffic sign's front view dataset

Models	Image resolution	FPS	Train (% mAP)	Validation (% mAP)	Test (% mAP)
Yolact-ResNet-50	512x512	4.84	93.45	93.45	93.19
Yolact-ResNet-101	512x512	3.18	99.50	99.50	96.52
Yolact-ResNet-50	550x550	5.06	93.21	93.21	92.53
Yolact-ResNet-101	550x550	3.25	99.15	99.15	94.99
Mask-RCNN-ResNet-50	512x512	0.78	100.00	100.00	87.50
Mask-RCNN-ResNet-101	512x512	0.68	100.00	100.00	93.75

4.5 Evaluation the Testing Datasets Between Front View, Days, Right-left View, and Night Conditions

Based on the dataset, tests were conducted to evaluate the performance of YOLACT-ResNet-50 and YOLACT-ResNet-101 on front, right-left, day, and night views using precision, recall, specificity, and accuracy. Tables 3 and 4 present the performance indices of the two models, computed using Eq. (2)-(5). Figure 8 shows the results for objects predicted with an IOU score above 0.5, and the performance of the confusion matrix is computed using four indices for each image.



(a) (b) (c) (d)

Fig 8. The example output of each condition (a) Front-view, (b) Night-time, (c) Days, and (d) Left-right view conditions

Table 3 presents the overall results for different conditions and image resolutions. The 512x512 pixel resolution performs better than the 550x550 pixel resolution in terms of overall precision, recall, specificity, and accuracy, except for precision. The 512x512 resolution achieves 92.31%, 80.81%, 99.48%, and 84.35% for precision, recall, specificity, and accuracy, respectively, while the 550x550 resolution obtains 97.45%, 72.92%, 99.35%, and 78.04% for these metrics. YOLACT-ResNet-50 exhibits different performances ranging from 0% to 7% for front and right-left views at both image

resolutions. In daytime conditions, the values range from 0% to 29% away from nighttime conditions. The recall performance is the lowest for ResNet-50, with the model having low sensitivity in recognizing objects in nighttime conditions, reaching 59.38% and 67.19% in 550x550 and 512x512 nighttime conditions, respectively. Overall, ResNet-50 with 512x512 pixels performs well in terms of the four metrics, except for nighttime conditions where precision, recall, and accuracy are low.

Table 3
YOLACT-ResNet-50 for four conditions

Image resolution	Condition	Front (%)	Right and Left view (%)	Days (%)	Nights (%)	Overall (%)
512x512	Precision	100.00	98.22	100.00	71.00	92.31
	Recall	87.50	85.73	82.81	67.19	80.81
	Specificity	100.00	99.49	100.00	98.44	99.48
	Accuracy	90.00	87.41	86.25	73.75	84.35
550x550	Precision	98.53	95.83	97.92	97.50	97.45
	Recall	81.25	75.00	76.04	59.38	72.92
	Specificity	99.48	98.96	99.48	99.48	99.35
	Accuracy	85.00	80.00	79.66	67.50	78.04

Table 4 presents the performance of YOLACT-ResNet-101, with a precision of 95.75%, recall of 90.99%, specificity of 98.44%, and accuracy of 92.24% at the 512x512 resolution. At the 550x550 resolution, the model achieves a precision of 94.73%, recall of 92.16%, specificity of 98.19%, and accuracy of 92.59%. YOLACT-ResNet-101 performs differently in front and right-left view conditions, with the four metrics differing by 0% to 7% at both image resolutions. The values range from 0% to 11% in daytime and nighttime conditions. The recall is the lowest for ResNet-101, with the model having low sensitivity in recognizing objects in nighttime conditions, reaching 85.94% in both image resolutions. Overall, YOLACT achieves more than 80% performance for both backbones, except for ResNet-50's recall and accuracy for 550x550 pixels in right-left view, day, and night conditions, ResNet-50's 512x512 pixel nighttime condition, and ResNet-50's precision for nighttime in 512x512 pixels.

Table 4
YOLACT-ResNet-101 for the four conditions

Image resolution	Condition	Front (%)	Right and Left view (%)	Days (%)	Nights (%)	Overall (%)
512x512	Precision	95.50	91.83	97.22	98.33	95.72
	Recall	92.19	89.06	96.77	85.94	90.99
	Specificity	98.44	96.88	98.96	99.48	98.44
	Accuracy	93.75	91.25	95.19	88.75	92.24
550x550	Precision	92.50	93.94	96.97	95.50	94.73
	Recall	89.06	95.21	98.44	85.94	92.16
	Specificity	97.40	97.94	98.97	98.45	98.19
	Accuracy	91.25	94.99	96.53	87.57	92.59

It was deduced that the best model was YOLACT-ResNet-101 with a 550x550 image resolution. This is because YOLACT prioritizes lightweight ability over performance. ResNet-50 struggled to recognize objects during nighttime conditions and had reduced recall as the resolution increased. In contrast, ResNet-101 achieved over 80% image resolution performance with YOLACT.

5. Conclusions

This study evaluates real-time instance segmentation models and their application to four conditions. The evaluated model consists of a fully convolutional model with a backbone, FPN, protonet, and prediction heads. The FPN structure detects objects of different scales, while the improved backbone structure extracts input features. Two concurrent processes create instance masks assembly, one performed by protonet and the other by the prediction head. The YOLACT model was evaluated with different backbone configurations, and YOLACT-ResNet-101 with 550x550 image resolution outperforms Mask-RCNN in terms of FPS, making it the preferred model for handling image resolution. Based on the results, for the four examined traffic sign conditions, it achieved better than 80% performance and worked well at night according to the four performance values. The study suggests improving the YOLACT system's performance by using advanced transfer learning models, developing general classification models, and using high-performance GPUs. Overall, YOLACT shows high performance in traffic sign detection and classification, and it is recommended for use in self-driving systems.

Acknowledgement

This research is supported by Universiti Teknologi Malaysia and Universiti Tun Hussein Onn Malaysia through TIER 1 vot(H194).

References

[1] Chen, Jerry, Maysam Abbod, and Jiann-Shing Shieh. "Integrations between autonomous systems and modern computing techniques: a mini review." *Sensors* 19, no. 18 (2019): 3897. https://doi.org/10.3390/s19183897

[2] Inagaki, Toshiyuki, and Thomas B. Sheridan. "A critique of the SAE conditional driving automation definition, and analyses of options for improvement." *Cognition, technology & work* 21 (2019): 569-578. https://doi.org/10.1007/s10111-018-0471-5

[3] Ackerman, Rebecca P. "Regulating autonomous vehicles in the United States." (2019).

[4] Shadrin, Sergey Sergeevich, Oleg Olegovich Varlamov, and Andrey Mikhailovich Ivanov. "Experimental autonomous road vehicle with logical artificial intelligence." *Journal of advanced transportation* 2017 (2017). https://doi.org/10.1155/2017/2492765

[5] Hassaballah, Mahmoud, Mourad A. Kenk, Khan Muhammad, and Shervin Minaee. "Vehicle detection and tracking in adverse weather using a deep learning framework." *IEEE transactions on intelligent transportation systems* 22, no. 7 (2020): 4230-4242. https://doi.org/10.1109/TITS.2020.3014013

[6] Geng, Keke, and Guodong Yin. "Using deep learning in infrared images to enable human gesture recognition for autonomous vehicles." *IEEE Access* 8 (2020): 88227-88240. https://doi.org/10.1109/ACCESS.2020.2990636

[7] Zhang, Yongtao, Zhishuai Yin, Linzhen Nie, and Song Huang. "Attention based multi-layer fusion of multispectral images for pedestrian detection." *IEEE Access* 8 (2020): 165071-165084. https://doi.org/10.1109/ACCESS.2020.3022623

[8] Zhu, Y., Yan, W.Q. Traffic sign recognition based on deep learning. Multimed Tools Appl 81, 17779–17791 (2022). https://doi.org/10.1007/s11042-022-12163-0

[9] Jiang, Yanhua, Shengyan Zhou, Yan Jiang, Jianwei Gong, Guangming Xiong, and Huiyan Chen. "Traffic sign recognition using ridge regression and Otsu method." In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 613-618. IEEE, 2011. https://doi.org/10.1109/IVS.2011.5940440

[10] Mathias, Markus, Radu Timofte, Rodrigo Benenson, and Luc Van Gool. "Traffic sign recognition—How far are we from the solution?." In *The 2013 international joint conference on Neural networks (IJCNN)*, pp. 1-8. IEEE, 2013. https://doi.org/10.1109/IJCNN.2013.6707049

[11] Greenhalgh, Jack, and Majid Mirmehdi. "Real-time detection and recognition of road traffic signs." *IEEE transactions on intelligent transportation systems* 13, no. 4 (2012): 1498-1506. https://doi.org/10.1109/TITS.2012.2208909

[12] Jiang, Yanhua, Shengyan Zhou, Yan Jiang, Jianwei Gong, Guangming Xiong, and Huiyan Chen. "Traffic sign recognition using ridge regression and Otsu method." In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 613-618. IEEE, 2011. https://doi.org/10.1109/IVS.2011.5940440

[13] Tian, Tian, Ishwar Sethi, and Nilesh Patel. "Traffic sign recognition using a novel permutation-based local image feature." In *2014 International Joint Conference on Neural Networks (IJCNN)*, pp. 947-954. IEEE, 2014. https://doi.org/10.1109/IJCNN.2014.6889629

[14] Guo, Tianmei, Jiwen Dong, Henjian Li, and Yunxing Gao. "Simple convolutional neural network on image classification." In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, pp. 721-724. IEEE, 2017. https://doi.org/10.1109/ICBDA.2017.8078730

[15] Boujemaa, Kaoutar Sefrioui, Ismail Berrada, Afaf Bouhoute, and Karim Boubouh. "Traffic sign recognition using convolutional neural networks." In *2017 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, pp. 1-6. IEEE, 2017. https://doi.org/10.1109/WINCOM.2017.8238205

[16] Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." *Advances in neural information processing systems* 28 (2015). https://doi.org/10.1109/TPAMI.2016.2577031

[17] Lin, Chien-Ming, Chi-Yi Tsai, Yu-Cheng Lai, Shin-An Li, and Ching-Chang Wong. "Visual object recognition and pose estimation based on a deep semantic segmentation network." *IEEE sensors journal* 18, no. 22 (2018): 9370-9381. https://doi.org/10.1109/JSEN.2018.2870957

[18] Zhang, Bo, and Jian Zhang. "A traffic surveillance system for obtaining comprehensive information of the passing vehicles based on instance segmentation." *IEEE Transactions on Intelligent Transportation Systems* 22, no. 11 (2020): 7040-7055. https://doi.org/10.1109/TITS.2020.3001154

[19] Bolya, Daniel, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. "Yolact: Real-time instance segmentation." In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9157-9166. 2019. https://doi.org/10.1109/ICCV.2019.00925

[20] Guo, Feng, Yu Qian, Yunpeng Wu, Zhen Leng, and Huayang Yu. "Automatic railroad track components inspection using real-time instance segmentation." *Computer-Aided Civil and Infrastructure Engineering* 36, no. 3 (2021): 362-377. https://doi.org/10.1111/mice.12625

[21] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016. https://doi.org/10.1109/CVPR.2016.90

[22] He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In *Proceedings of the IEEE international conference on computer vision*, pp. 2961-2969. 2017. https://doi.org/10.1109/ICCV.2017.322

[23] Girshick, Ross. "Fast r-cnn." In *Proceedings of the IEEE international conference on computer vision*, pp. 1440-1448. 2015. https://doi.org/10.1109/ICCV.2015.169

[24] Wong, Ching-Chang, Li-Yu Yeh, Chih-Cheng Liu, Chi-Yi Tsai, and Hisasuki Aoyama. "Manipulation planning for object re-orientation based on semantic segmentation keypoint detection." *Sensors* 21, no. 7 (2021): 2280. https://doi.org/10.3390/s21072280

[25] Bolya, Daniel, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. "Yolact: Real-time instance segmentation." In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9157-9166. 2019. https://doi.org/10.1109/ICCV.2019.00925

[26] Chen, Fuen, Xiaoming Liang, Longhan Chen, Baoyuan Liu, and Yubin Lan. "Novel method for real-time detection and tracking of pig body and its different parts." *International Journal of Agricultural and Biological Engineering* 13, no. 6 (2020): 144-149. https://doi.org/10.25165/j.ijabe.20201306.5820

[27] Dai, Jifeng, Kaiming He, Yi Li, Shaoqing Ren, and Jian Sun. "Instance-sensitive fully convolutional networks." In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14*, pp. 534-549. Springer International Publishing, 2016. https://doi.org/10.1007/978-3-319-46466-4_32

[28] Li, Yi, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. "Fully convolutional instance-aware semantic segmentation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2359-2367. 2017. https://doi.org/10.1109/CVPR.2017.472

[29] Chen, Liang-Chieh, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. "Masklab: Instance segmentation by refining object detection with semantic and direction features." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4013-4022. 2018. https://doi.org/10.1109/CVPR.2018.00422

[30] Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft coco: Common objects in context." In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pp. 740-755. Springer International Publishing, 2014. https://doi.org/10.1007/978-3-319-10602-1_48

[31] Bolya, Daniel, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. "Yolact: Real-time instance segmentation." In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9157-9166. 2019. https://doi.org/10.1109/ICCV.2019.00925

[32] Chieng, Hock Hung, Noorhaniza Wahid, Pauline Ong, and Sai Raj Kishore Perla. "Flatten-T Swish: a thresholded ReLU-Swish-like activation function for deep learning." *arXiv preprint arXiv:1812.06247* (2018). https://doi.org/10.26555/ijain.v4i2.249

[33] Carrillo-Perez, Borja, Sarah Barnes, and Maurice Stephan. "Ship segmentation and georeferencing from static oblique view images." *Sensors* 22, no. 7 (2022): 2713. https://doi.org/10.3390/s22072713

[34] Chen, Wei-Yu, Yu-Reng Tsao, Jin-Yi Lai, Ching-Jung Hung, Yu-Cheng Liu, and Cheng-Yang Liu. "Real-time instance segmentation of metal screw defects based on deep learning approach." *Measurement Science Review* 22, no. 3 (2022): 107-111. https://doi.org/10.2478/msr-2022-0014

[35] Oksuz, Kemal, Baris Can Cam, Fehmi Kahraman, Zeynep Sonat Baltaci, Sinan Kalkan, and Emre Akbas. "Mask-aware iou for anchor assignment in real-time instance segmentation." *arXiv preprint arXiv:2110.09734* (2021).

[36] Pfeil, Juliane, Alina Nechyporenko, Marcus Frohme, Frank T. Hufert, and Katja Schulze. "Examination of blood samples using deep learning and mobile microscopy." *BMC bioinformatics* 23, no. 1 (2022): 65. https://doi.org/10.1186/s12859-022-04602-4

[37] Guo, Feng, Yu Qian, Dimitris Rizos, Zhi Suo, and Xiaobin Chen. "Automatic rail surface defects inspection based on Mask R-CNN." *Transportation research record* 2675, no. 11 (2021): 655-668. https://doi.org/10.1177/03611981211019034

[38] Zhong, Junping, Zhigang Liu, Hongrui Wang, Wenqiang Liu, Cheng Yang, Zhiwei Han, and Alfredo Nunez. "A looseness detection method for railway catenary fasteners based on reinforcement learning refined localization." *IEEE Transactions on Instrumentation and Measurement* 70 (2021): 1-13. https://doi.org/10.1109/TIM.2021.3086913

[39] Lin, Shaodan, Kexin Zhu, Chen Feng, and Zhide Chen. "Align-Yolact: A one-stage semantic segmentation network for real-time object detection." *Journal of Ambient Intelligence and Humanized Computing* (2021): 1-8. https://doi.org/10.1007/s12652-021-03340-4

[40] Bazire, Mary, and Charles Tijus. "Understanding road signs." *Safety science* 47, no. 9 (2009): 1232-1240. https://doi.org/10.1016/j.ssci.2009.03.013

[41] Scott-Parker, Bridie, and Oscar Oviedo-Trespalacios. "Young driver risky behaviour and predictors of crash risk in Australia, New Zealand and Colombia: Same but different?." *Accident Analysis & Prevention* 99 (2017): 30-38. https://doi.org/10.1016/j.aap.2016.11.001

[42] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." *arXiv preprint arXiv:1704.04861* (2017).

[43] Bak, Hui-Yong, and Seung-Bo Park. "Comparative study of movie shot classification based on semantic segmentation." *Applied Sciences* 10, no. 10 (2020): 3390. https://doi.org/10.3390/app10103390

[44] Zhao, Yongqiang, Rui Han, and Yuan Rao. "A new feature pyramid network for object detection." In *2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS)*, pp. 428-431. IEEE, 2019. https://doi.org/10.1109/ICVRIS.2019.00110

[45] Guo, Feng, Yu Qian, Yunpeng Wu, Zhen Leng, and Huayang Yu. "Automatic railroad track components inspection using real-time instance segmentation." *Computer-Aided Civil and Infrastructure Engineering* 36, no. 3 (2021): 362-377. https://doi.org/10.1111/mice.12625

[46] Zhang, Xinxiang, Dinesh Rajan, and Brett Story. "Concrete crack detection using context-aware deep semantic segmentation network." *Computer-Aided Civil and Infrastructure Engineering* 34, no. 11 (2019): 951-971. https://doi.org/10.1111/mice.12477