Pertanika J. Sci. & Technol. 32 (1): 365 - 394 (2024)

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Comparison Using Intelligent Systems for Data Prediction and Near Miss Detection Techniques

Lek Ming Lim[1], Saratha Sathasivam[1], Mohd. Tahir Ismail[1], Ahmad Sufril Azlan Mohamed[2], Olayemi Joshua Ibidoja[1] and Majid Khan Majahar Ali[1]*

[1]*School of Mathematical Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia*
[2]*School of Computer Sciences, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia*

ABSTRACT

Malaysia ranks third among ASEAN countries in terms of deaths due to accidents, with an alarming increase in the number of fatalities each year. Road conditions contribute significantly to near-miss incidents, while the inefficiency of installed CCTVs and the lack of monitoring system algorithms worsen the situation. The objective of this research is to address the issue of increasing accidents and fatalities on Malaysian roads. Specifically, the study aims to investigate the use of video technology and machine learning algorithms for the car detection and analysis of near-miss accidents. To achieve this goal, the researchers focused on Penang, where the MBPP has deployed 1841 CCTV cameras to monitor traffic and document near-miss accidents. The study utilised the YOLOv3, YOLOv4, and Faster RCNN algorithms for vehicle detection. Additionally, the study employed image processing techniques such as Bird's Eye View and Social Distancing Monitoring to detect and analyse how near misses occur. Various video lengths (20s, 40s, 60s and 80s) were tested to compare the algorithms' error detection percentage and test duration. The results indicate that Faster RCNN beats YOLOv3 and YOLOV4 in car detection with low error detection, whereas YOLOv3 and YOLOv4 outperform near-miss detection, while Faster RCNN does not perform it. Overall, this study demonstrates the potential of video technology and machine learning algorithms in near-miss accident detection and analysis. Transportation authorities can better understand the causes of accidents and take appropriate measures to improve road

ARTICLE INFO

Article history:
Received: 13 February 2023
Accepted: 21 July 2023
Published: 21 December 2023

DOI: https://doi.org/10.47836/pjst.32.1.20

E-mail addresses:
limlekming@gmail.com (Lek Ming Lim)
m.tahir@usm.my (Saratha Sathasivam)
saratha@usm.my (Mohd. Tahir Ismail)
sufril@usm.my (Ahmad Sufril Azlan Mohamed)
ojibidoja@fugusau.edu.ng (Olayemi Joshua Ibidoja)
majidkhanmajaharali@usm.my (Majid Khan Majahar Ali)
* Corresponding author

ISSN: 0128-7680
e-ISSN: 2231-8526

safety using these models. This research can be a foundation for further traffic safety and accident prevention studies.

Keywords: Bird's Eye View, intelligent systems, machine learning, near-miss, object detection, Social Distancing Monitoring, vehicle detection

INTRODUCTION

According to the World Health Organization (2020), the death rate from transportation ranked fifth in 2019. The majority of those killed in traffic accidents are teenagers. In 2030, the fatality rate from transportation is predicted to rise.

The main issue with transportation is a lack of accurate and reliable data. The manual data is the police data, also known as POL 37 data, collected after accidents, but some data would not be recorded. Therefore, the information could not be utilised to predict traffic conditions due to missing and inaccurate data in the statistical record. Majlis Bandar Pulau Pinang (MBPP) deployed 1841 Closed-circuit television (CCTV) cameras all around Pulau Pinang in 2019. CCTV is difficult to use to its full potential due to the lack of an algorithm capable of calculating and detecting vehicles, as well as the limited storage capacity, which can only store videos for one and a half months.

Furthermore, POL data is a hardcopy report. It cannot calculate vehicles automatically and does not record near misses due to the limitation of converting the manual report into a visual report. Aside from that, no autonomous algorithm can be utilised for near-miss counting. It is hard to measure near misses simultaneously in CCTV videos since no previous research exists.

Near misses are one of the transportation issues that must be addressed to reduce the likelihood of fatalities and accidents and meet the goals of the Penang 2030 mission. Since near misses cause accidents, near-miss reports are investigated to enhance road safety. Heinrich's 300-29-1 model demonstrated that the probability of 300 near misses can result in 29 minor injuries, and then these probabilities of minor injuries can result in one major injury. If 300 near-miss probabilities can cause a fatality or accident, it will emit carbon, harming the environment and contributing to atmospheric pollution. Therefore, reporting near misses is important to reduce the probability of accidents on specific roads and find out the root cause.

Near misses result from dangerous activities caused by human mistakes and situations produced by malfunctioning procedures or systems in Malaysian road traffic. According to Aldred (2016), cyclist behaviours influence near misses. A mixed traffic flow scenario contains a wide range of collision types, which contribute to near misses, too. According to Wang et al. (2020), the weather, which causes low-vision conditions, is another aspect that might lead to near misses in traffic flow. As a result, in the research, vision-based detection is used to detect objects and crash types in the monitoring system.

Vehicle detection is an algorithm-based computer vision technique used to detect vehicles, count vehicles and estimate the average speed of vehicles (Meng et al., 2020). Arinaldi et al. (2018) used visualisation in traffic monitoring systems. The research collected statistics on vehicle counts, vehicle type, predicted vehicle speed, and vehicle lane shift. These are all factors to consider in their study. Researchers who conducted prior investigations only used images and a linked-in algorithm and software method to conduct vehicle detection. The main reason is that the monitoring systems will show all details and information through visualisation reporting.

Appendix A summarises previous research and highlights gaps in vehicle detection, particularly in Penang. Limited studies employ various methods for detecting vehicles and identifying near misses, focusing on software and algorithm development in engineering. While image processing methods are commonly used, this study aims to improve accuracy by implementing Convolutional Neural Network (CNN) and Fast Region-based Convolutional Neural Network (RCNN). Many researchers connect their model or algorithm to software to enable visualisation in monitoring systems. Such systems aid in data analysis from image and video processing. In this study, Social Distancing Monitoring and Bird’s Eye View are used in vehicle identification to analyse images and identify near misses.

Models or software that recognise objects or track vehicles include the Faster RCNN, and You Only Look Once (YOLO). These models are not well-known from near-miss studies. Huang et al. (2020) detected vehicles in traffic utilising Faster RCNN, and You Only Look Once version 3 (YOLOv3) on 40-second basis CCTV recordings. Kumar et al. (2020) employed YOLOv3, and You Only Look Once version 4 (YOLOv4) for surveillance in traffic monitoring systems. Ammar et al. (2021) compare Faster RCNN, YOLOv3, and YOLOv4 for vehicle recognition from images. Faster RCNN, YOLOv3, and YOLOv4 were used by Sowmya and Radha (2021) to develop vehicle identification and classification algorithms for effective heavy vehicle traffic monitoring. Lim, Ali et al. (2022) experimented with different video quality in vehicle recognition using YOLOv3 and Faster RCNN.

In this experiment, YOLOv3 and YOLOv4 perform image processing on vehicle recognition in CCTV recordings using the approaches of Bird’s Eye View and Social Distancing Monitoring. The road condition may be monitored using vehicle detection to study the process of near misses and accidents. The data acquired may be utilised to forecast near misses in traffic flow and determine the causes of the issues.

MATERIALS AND METHODS

YOLO

Figure 1 shows the flow chart of YOLO in vehicle detection.

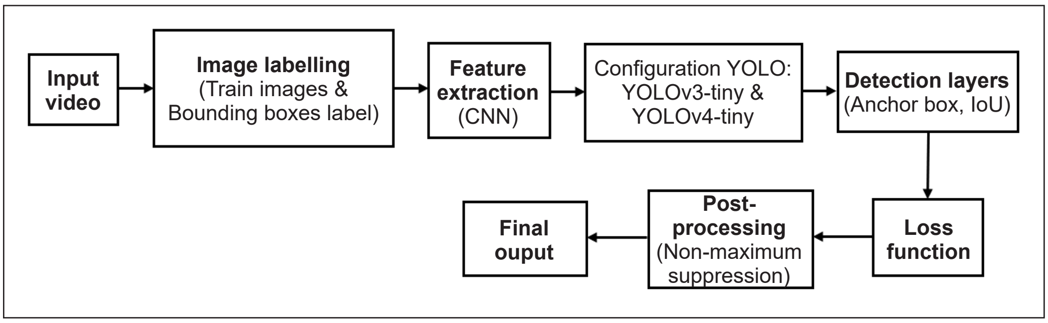

Input video
Image labelling (Train images & Bounding boxes label)
Feature extraction (CNN)
Configuration YOLO: YOLOv3-tiny & YOLOv4-tiny
Detection layers (Anchor box, IoU)
Loss function
Post-processing (Non-maximum suppression)
Final ouput

Figure 1. YOLO flow chart

Image Labelling

Training YOLO to identify an object involves the training image and the bounding box label. Collecting training images is the first step in initiating the object detector. Bounding box annotations are used to aid the object detector's learning. Each object detected by the detector is enclosed by a box and labelled with the object class forecasted by the detector. Image training in YOLO is depicted in Figure 2.

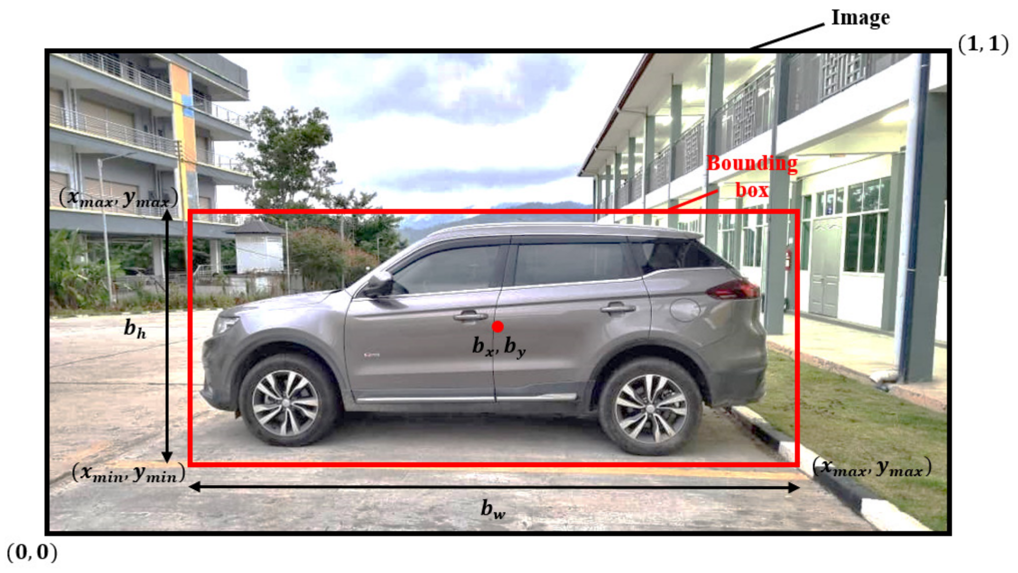


Figure 2. Train image in YOLO

Define W, H is the dimension of the original image (Equation 1) (AlKishri & Al-Bahri, 2021).

$$b_x = \frac{(x_{min} + x_{max})}{2 * W}, \qquad b_y = \frac{(y_{min} + y_{max})}{2 * H}$$

$$b_w = \frac{(x_{max} - x_{min})}{W}, \qquad b_h = \frac{(y_{max} - y_{min})}{H} \qquad [1]$$

where: b_x, b_y is the box locations; b_x, b_y, b_w, b_h is the width and height of the entire image; b_x, b_y, b_w, b_h is the bounding box prediction; $x_{min}, y_{min}, x_{max}, y_{max}$ is the image bounding box coordinate; and W, H is the width and height of the bounding box.

Feature Extraction

Gai et al. (2021) stated that other deep learning methods for object detection typically involve merging the feature extraction process with the classification prediction. It involves using a convolutional network to extract features and predict the objects' locations and classifications.

Detection Layers

Silva et al. (2020) stated that during model training, the convolutional network layer was configured to detect only one class: the car. The number of filters used in the layer was directly set in the configuration. The formula for determining the number of filters is in Equation 2:

$$\begin{aligned} Filters &= (classes + w + h + x + y + Confidence\ score) \times num \\ &= (classes + 1 + 1 + 1 + 1 + 1) \times num \\ &= (classes + 5) \times num \end{aligned} \qquad [2]$$

where 5 represents the conditions (x,y - position of boundary box, w,h - width and height of the image and C - the Confidence score), and c is the class probability. YOLO detect 3 boxes per grid cell (Equation 3), so the:

$$Filters = (classes + 5) \times 3 \qquad [3]$$

Therefore, there are 18 filters for 1 class. These 18 filters will be placed before two YOLO layers.

Figure 3 illustrates how YOLO performs detection. Initially, the input image is divided into S×S grids. If the centre of a ground truth object lies within a grid cell, the grid is assigned to detect that object. Each grid cell has B bounding boxes and corresponding confidence scores for those boxes. The bounding boxes and confidence scores are then multiplied with class probability maps. Finally, the results of the final detection are obtained.

The final detection takes place during inference using the formula derived from Figure 4, a tensor of size $S \times S * (B * 5 + c)$. In this formula, S represents the number of grids, B

is the bounding box, 5 denotes the four bounding box attributes and one object confidence per cell, c is the class probability, and tensor refers to the network output. Redmon et al. (2016) employed the PASCA VOC dataset, which consists of 20 object classes, resulting in $S = 7$, $B = 2$, $c = 20$. As a result, the YOLO model network structure produces a 7×7×30 tensor. YOLOv3-tiny and YOLOv4-tiny model network structures are used for feature extraction in the detection layers.

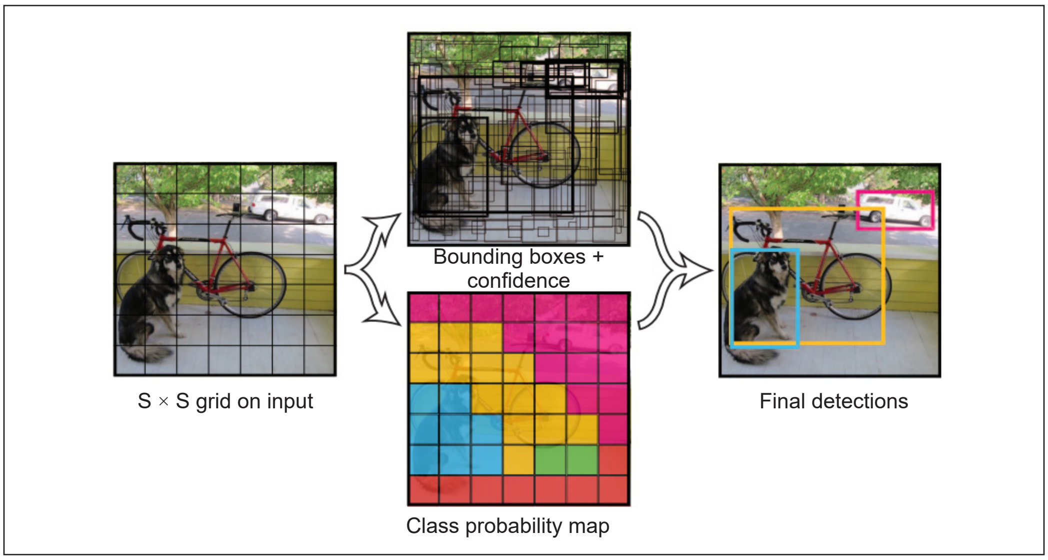


Figure 3. YOLO algorithm detection calculation (Redmon et al., 2016)

YOLO Model Training

The network was set up in model training to identify a car class. The number of filters is set directly in the convolutional network layer (Silva et al., 2020). The formula of the filter number is in Equation 4.

$$\begin{aligned} Filters &= (classes + w + h + x + y + Confidence\ score) \times num \\ &= (classes + 1 + 1 + 1 + 1 + 1) \times num \\ &= (classes + 5) \times num \end{aligned} \qquad [4]$$

where 5 represents the conditions (*x,y* - position of boundary box, *w,h* - width and height of the image and C - the Confidence score), and c is the class probability. YOLO detect 3 boxes per grid cell (Equation 5), so the

$$Filters = (classes + 5) \times 3 \qquad [5]$$

Therefore, there are 18 filters for 1 class. These 18 filters will be placed before two YOLO layers.

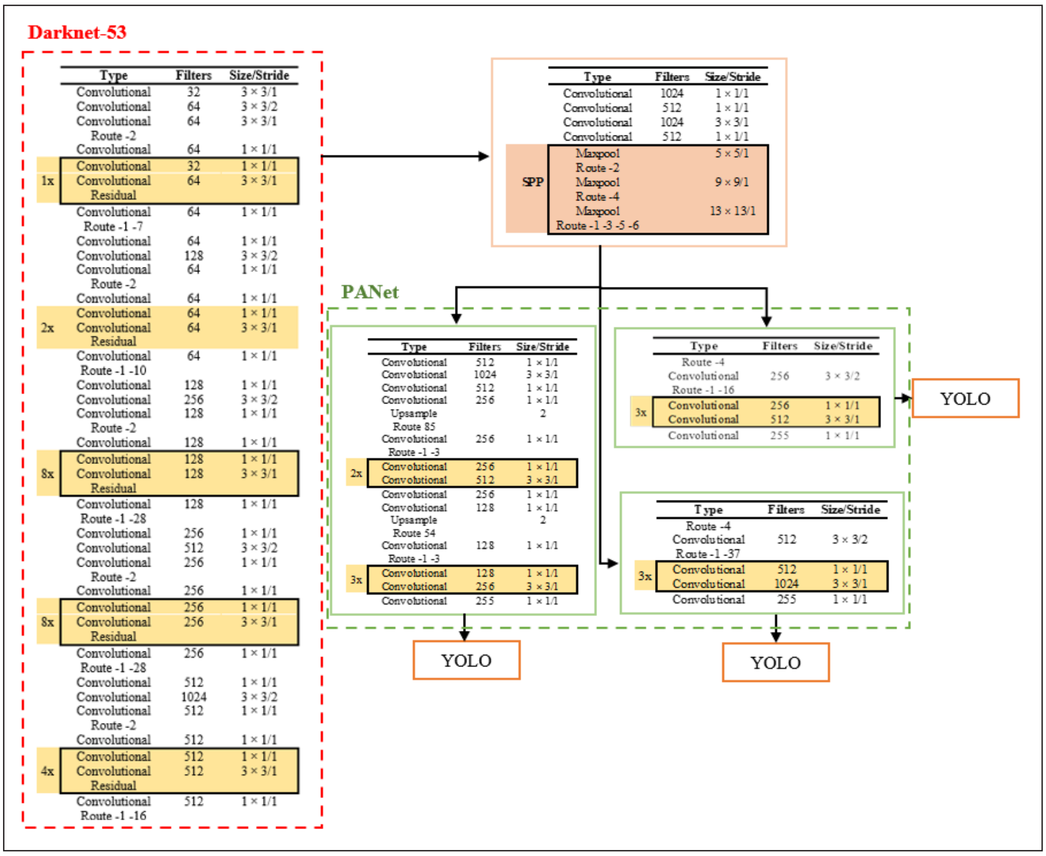
YOLOv3. The object detection algorithm YOLOv3 is in its third iteration of You Only Look Once. The accuracy has been greatly improved over earlier methods (Wang, 2021). Among these single-stage detectors, the YOLO family model may have the fastest object-detecting algorithm with the greatest achieved accuracy rate (Redmon et al., 2016). The real-time performance of YOLO series models reported in the literature is evaluated using a graphics processor unit (GPU) card with high-performance computational power (Redmon & Farhadi, 2018). YOLO predicts the class and object position using a forward convolutional network. With the creation of the basic network, YOLOv3, which uses Darknet-53 to replace the backbone network and uses multi-scale characteristics to determine the target, was proposed (Redmon & Farhadi, 2018).

YOLOv3 improves the network by introducing a residual module based on Darknet-19, which is You Only Look Once version 2's (YOLOv2) backbone and expands the network (Wang, 2021). Darknet-53, the enhanced network, contains 53 convolution layers. Table 1 shows the YOLOv3 network structure (Ezat et al., 2021).

Table 1
YOLOv3 network structure

	Type	**Filters**	**Size/Stride**	**Output**
	Convolutional	32	3 × 3/1	256 × 256
	Convolutional	64	3 × 3/2	128 × 128
	Convolutional	32	3 × 3/1	
1×	Convolutional	64	3 × 3/1	
	Residual			128 × 128
	Convolutional	128	3 × 3/2	64 × 64
	Convolutional	64	3 × 3/1	
2×	Convolutional	128	3 × 3/1	
	Residual			64 × 64
	Convolutional	256	3 × 3/2	32 × 32
	Convolutional	128	1 × 1/1	
8×	Convolutional	256	3 × 3/1	
	Residual			32 × 32
	Convolutional	512	3 × 3/2	16 × 16
	Convolutional	256	1 × 1/1	
8×	Convolutional	512	3 × 3/1	
	Residual			16 × 16
	Convolutional	1024	3 × 3/2	8 × 8
	Convolutional	512	1 × 1/1	
4×	Convolutional	1024	3 × 3/1	
	Residual			8 × 8
	Avgpool		Global	
	Convolutional	1000	1 × 1/1	8 × 8
	Softmax			

YOLOv4. YOLOv4 is a one-stage object detection algorithm that builds on YOLOv3 with various additional techniques and features. The YOLOv4 network structure is constructed of the Cross Stage Partial Darknet-53 (CSPDarknet53) backbone, the Spatial pyramid pooling (SPP) additional module, the Path Aggregation Network (PANet) path-aggregation neck, and the anchor-based YOLOv3 head (Bochkovskiy et al., 2020). Figure 4 shows the network structure of YOLOv4 (Abdurahman et al., 2021).



	Type	Filters	Size/Stride
	Convolutional	32	3 × 3/1
	Convolutional	64	3 × 3/2
	Convolutional	64	3 × 3/1
	Route -2		
	Convolutional	64	1 × 1/1
1x	Convolutional	32	1 × 1/1
	Convolutional	64	3 × 3/1
	Residual		
	Convolutional	64	1 × 1/1
	Route -1 -7		
	Convolutional	64	1 × 1/1
	Convolutional	128	3 × 3/2
	Convolutional	64	1 × 1/1
	Route -2		
	Convolutional	64	1 × 1/1
2x	Convolutional	64	1 × 1/1
	Convolutional	64	3 × 3/1
	Residual		
	Convolutional	64	1 × 1/1
	Route -1 -10		
	Convolutional	128	1 × 1/1
	Convolutional	256	3 × 3/2
	Convolutional	128	1 × 1/1
	Route -2		
	Convolutional	128	1 × 1/1
8x	Convolutional	128	1 × 1/1
	Convolutional	128	3 × 3/1
	Residual		
	Convolutional	128	1 × 1/1
	Route -1 -28		
	Convolutional	256	1 × 1/1
	Convolutional	512	3 × 3/2
	Convolutional	256	1 × 1/1
	Route -2		
	Convolutional	256	1 × 1/1
8x	Convolutional	256	1 × 1/1
	Convolutional	256	3 × 3/1
	Residual		
	Convolutional	256	1 × 1/1
	Route -1 -28		
	Convolutional	512	1 × 1/1
	Convolutional	1024	3 × 3/2
	Convolutional	512	1 × 1/1
	Route -2		
	Convolutional	512	1 × 1/1
4x	Convolutional	512	1 × 1/1
	Convolutional	512	3 × 3/1
	Residual		
	Convolutional	512	1 × 1/1
	Route -1 -16		

	Type	Filters	Size/Stride
	Convolutional	1024	1 × 1/1
	Convolutional	512	1 × 1/1
	Convolutional	1024	3 × 3/1
	Convolutional	512	1 × 1/1
SPP	Maxpool		5 × 5/1
	Route -2		
	Maxpool		9 × 9/1
	Route -4		
	Maxpool		13 × 13/1
	Route -1 -3 -5 -6		

	Type	Filters	Size/Stride
	Convolutional	512	1 × 1/1
	Convolutional	1024	3 × 3/1
	Convolutional	512	1 × 1/1
	Convolutional	256	1 × 1/1
	Upsample		2
	Route 85		
	Convolutional	256	1 × 1/1
	Route -1 -3		
2x	Convolutional	256	1 × 1/1
	Convolutional	512	3 × 3/1
	Convolutional	256	1 × 1/1
	Convolutional	128	1 × 1/1
	Upsample		2
	Route 54		
	Convolutional	128	1 × 1/1
	Route -1 -3		
3x	Convolutional	128	1 × 1/1
	Convolutional	256	3 × 3/1
	Convolutional	255	1 × 1/1

	Type	Filters	Size/Stride
	Route -4		
	Convolutional	256	3 × 3/2
	Route -1 -16		
3x	Convolutional	256	1 × 1/1
	Convolutional	512	3 × 3/1
	Convolutional	255	1 × 1/1

	Type	Filters	Size/Stride
	Route -4		
	Convolutional	512	3 × 3/2
	Route -1 -37		
3x	Convolutional	512	1 × 1/1
	Convolutional	1024	3 × 3/1
	Convolutional	255	1 × 1/1

Figure 4. YOLOv4 network structure (Abdurahman et al., 2021)

Anchor Box and Intersection over Union

According to the study by Ezat et al. (2021), the Intersection over Union (IoU) metric is utilised to evaluate object detection accuracy by comparing the overlap of two boxes. The calculation of IoU for various configurations of bounding boxes is illustrated in Figure 5.

Loss Function

According to the study of Cepni et al. (2020), the loss function is calculated using a sum squared error. The loss function consists of three parts: localisation error, confidence error and classification error (Equation 6).

$$loss = \sum_{i=0}^{S^2} coordError + iouError + classError \quad [6]$$

where S is the number of grids, $coordError$ is the localisation error, $iouError$ is the confidence error, and $classError$ is the classification error.

Localisation Error. In the equation, localisation error is represented by $coordError$. The localisation error is used to calculate the accuracy of the estimated bounding boxes. Equation 7 uses error to measure the position and dimensions of bounding boxes (Zhang et al., 2019).

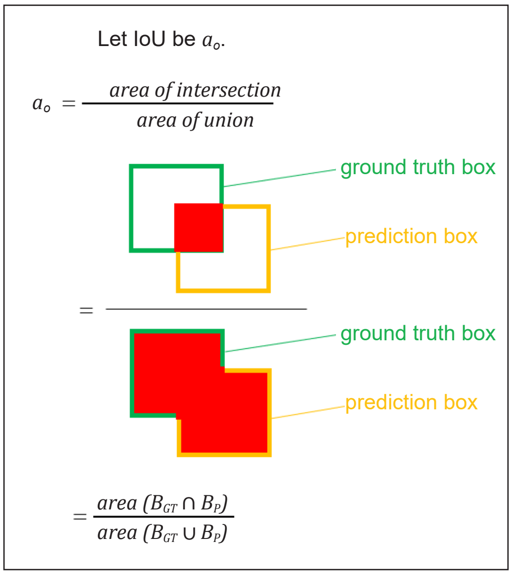

Let IoU be a_o.
$a_o = \frac{area\ of\ intersection}{area\ of\ union}$
ground truth box
prediction box
ground truth box
prediction box
$= \frac{area\ (B_{GT} \cap B_P)}{area\ (B_{GT} \cup B_P)}$

Figure 5. IoU calculation (which B_{GT} refers to the ground truth box represented by a green square, while B_P refers to the prediction box represented by an orange square. The IoU value can be calculated as the area of overlap between these two boxes, which is denoted by a_o)

$$coordError = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] \quad [7]$$

where S is the number of grids, B is the boundary box, x and y are the centre of the box relative to the bounds of the grid cell (coordinate system), and w and h are the width and height of the entire image (scale of the original image), λ_{coord} increase the weight of loss in the coordinate of the box and its value is fixed as $\lambda_{coord} = 5$, $1_{ij}^{obj} = 1$ if the jth boundary box in cell i is responsible for detecting the object, otherwise 0.

Confidence Error. The confidence error is represented by $iouError$. It is used to measure differently by the presence or absence of the object and bounding box (Jiang et al., 2020). The first line of Equation 8 represents the object's presence and is detected by the bounding box, while the second line of Equation 8 represents the object's absence and is not detected by the bounding box.

$$iouError = \sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{obj}\left(C_i - \hat{C}_i\right)^2 + \lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} 1_{ij}^{noobj}\left(C_i - \hat{C}_i\right)^2 \qquad [8]$$

where S is the number of grids, B is the boundary box, $\hat{C}_i$ is the prediction confidence score, λ_{noobj} is the weights to prevent unbalanced classes because most of the bounding boxes do not detect objects or is a background, and its value is fixed as $\lambda_{noobj} = 0.5$ to prevent the loss be tilted toward negative values or objects, 1_{ij}^{noobj} is the supplement of 1_{ij}^{obj}.

Classification Error. The classification error is represented by $classError$. It calculates the correctness of the estimated object (Equation 9) (Jiang et al., 2020).

$$classError = \sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{c \in classes}\left(p_i(c) - \hat{p}_i(c)\right)^2 \qquad [9]$$

where S is the number of grids, 1_{ij}^{obj} if an object is present, otherwise 0, $\hat{p}_i(c)$ is the prediction class probability for class c.

Post Process

According to Adarsh et al. (2020), Non-Maximum Suppression (NMS) is an additional step after object detection to remove redundant predictions and choose the most accurate prediction for each object in an image. This technique involves computing the IoU between every pair of predictions and eliminating the prediction with the lower confidence score.

Final Output

The monitoring system will identify the object and display it in the image. The final output is shown in the final detection in Figure 6.

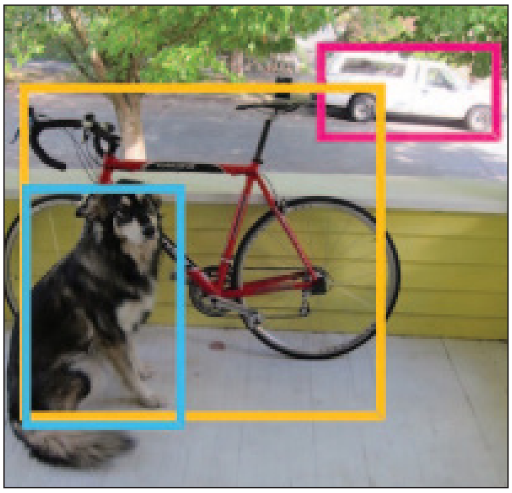


Figure 6. YOLO final detection output (Redmon et al., 2016)

Faster RCNN

In the research of Girshick (2015), Faster RCNN was developed. Figure 12 depicts the two stages in faster RCNN. It is a hybrid of Fast RCNN and the Region Proposal Network (RPN). Figure 7 shows the architecture of Faster RCNN.

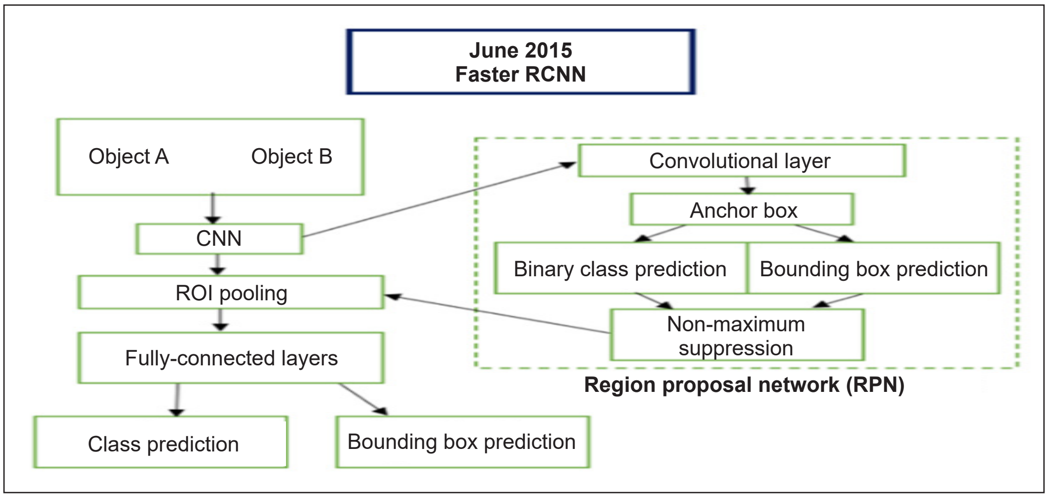


Figure 7. Faster RCNN architecture

RPN. Figure 8 shows the first stage of Faster RCNN, known as RPN. Faster RCNN will replace the external algorithm, selective search, with an RPN, which will share many parameters with Fast RCNN. So, to share the convolutional layers, utilise this concept of parameter sharing, a region proposal network, and Fast RCNN around the network (Zhao et al., 2020).

When an image is captured, it is passed through convolutional layers, beginning and ending at a particular feature map (Ren et al., 2017). Due to the external algorithm's need to avoid providing object proposals, all of these actions are shared. It should be replaced rather than removed entirely. Proposals with dense anchor boxes are replaced for each

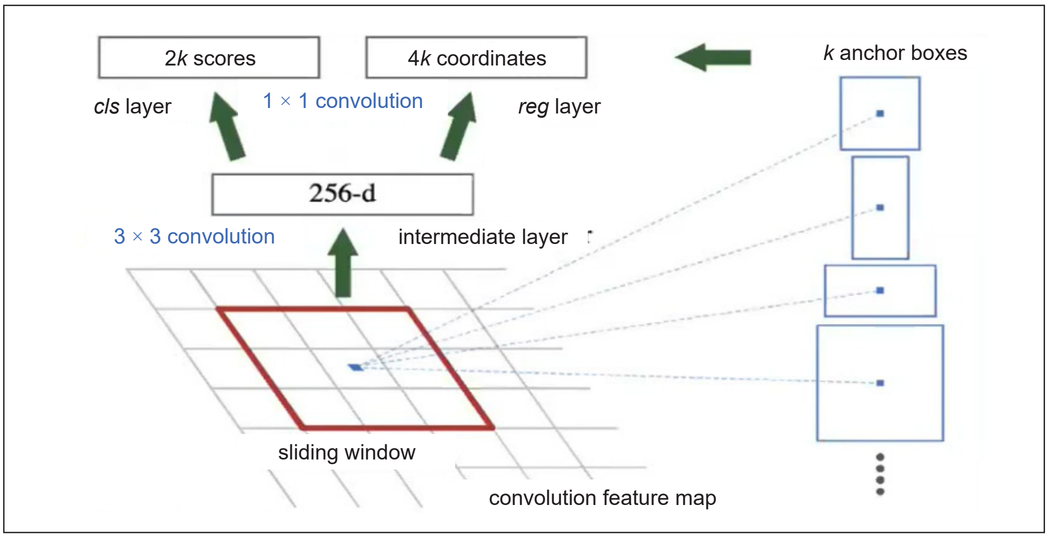


Figure 8. RPN (Ren et al., 2017)

pixel in the convolutional feature map, which considers k anchor boxes. Then, there are several anchor boxes in the pixels.

Following the study of Wang et al. (2019), w times h, which is the feature map's resolution, multiplied by k of those anchor boxes. Then, make any necessary adjustments to the anchor boxes. The Region Proposal Network will detect the object's existence. So, for each anchor, there are two classes: one for adjusting the anchor boxes. According to the research of Gou et al. (2019), the Region Proposal Network is led by two heads. 2k scores represent the object's existence. Then, $4k$ coordinates denote the top, bottom, left, and right from the centre point to the bounding box. The pixels are denoted by W and H. The red square in the illustration above is the 3×3 convolution in the section of the translation-invariant anchor. As a result, it contains nine anchors with three scales and three aspect ratios. It uses three sizes for anchors, with box areas of 128×128, 256×256, and 512×512 pixels and aspect ratios of 1:1, 1:2, and 2:1.

The Region Proposal Network is trained on a good loss function about classification and regression. Then, there is the problem of properly labelling data and observing the intersection over the union with ground truth. The equation below shows the loss function in the Region Proposal (Sekar & Perumal, 2021). The loss function comprises the loss function for the classifier, $L(C_i)$ and the loss function for the regressor, $L(R_i)$ (Equation 10).

$$L\{p_i, t_i\} = [L(C_i) + L(R_i)] \qquad [10]$$

where: i represents the selected proposal region index; p_i is the probability that the candidate box i is the object; t_i is the position coordinates of the predicted box.

The loss function of the classifier, $L(C_i)$ (Equation 11) (Jiang & Shi, 2021).

$$L(C_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i{}^*) \qquad [11]$$

where: C_i is the classifier in candidate box i; N_{cls} is the number of anchors available in mini-batch, is normalised and weighted by a parameter λ; i represents the selected proposal region index; L_{cls} is classification loss, which is a log function for classifying object or not object; p_i is the value of predicted probability in the candidate box i; p_i*, also called a ground truth label, which is used to mark the anchor positive when p_i* = 1 or negative when p_i* = 0;

The classification loss, L_{cls}, is the logarithm loss for two categories which are object and non-object (Equation 12) (Wang et al., 2019).

$$L_{cls}(p_i, p_i{}^*) = \begin{cases} -\log p_i & if\ p_i{}^* = 1 \\ -\log(1 - p_i) & if\ p_i{}^* = 0 \end{cases} \qquad [12]$$

The loss function of the regressor, $L(R_i)$ (Equation 13) (Jiang & Shi, 2021).

$$L(R_i) = \lambda \frac{1}{N_{reg}} \sum_i L_{reg}(t_i, t_i^*) \qquad [13]$$

where: R_i is the regressor in candidate box i; λ is a constant value or a balancing factor; N_{reg} is the number of anchor locations normalised and weighted by a parameter λ; i represents the selected proposal region index; L_{reg} is regression loss; t_i is the position coordinates of the predicted box; t_i* is the coordinate vector of the corresponding object bounding box.

The regression loss, L_{reg}, is the loss of the ith region given by the Equations 14 and 15 (Wang et al., 2019).

$$L_{reg}(t_i, t_i^*) = smooth_{L1}(t_i, t_i^*) \qquad [14]$$

$$smooth_{L1}(m) = \begin{cases} 0.5m^2, & |m| < 1 \\ |m| - 0.5, & |m| \geq 1 \end{cases} \qquad [15]$$

where $smooth_{L1}\ (m)$ is a smooth function of function m.

The parameter of regression expression, $\{t_i\}$ and the region values of x, y, w and h are shown in Equation 16 (Jiang & Shi, 2021).

$$\begin{aligned} t_x &= \frac{(x - x_a)}{w_a}, \quad t_y = \frac{(y - y_a)}{h_a} \\ t_w &= \log\left(\frac{w}{w_a}\right), \qquad t_h = \log\left(\frac{h}{h_a}\right) \\ t_x{}^* &= \frac{(x^* - x_a)}{w_a}, \quad t_y{}^* = \frac{(y^* - y_a)}{h_a} \\ t_w{}^* &= \log\left(\frac{w^*}{w_a}\right), \qquad t_h{}^* = \log\left(\frac{h^*}{h_a}\right) \end{aligned} \qquad [16]$$

where x, y are the predicted bounding box coordinates; x^*, y^* is the ground truth bounding box coordinates; w, h is the width and height of the predicted bounding box; w^*, h^* is the width and height of the ground truth bounding box; t_x, t_y is the regression value of x and y of the predicted bounding box coordinates; t_x*, t_y* is the regression value of x and y of the ground truth bounding box coordinates; t_w, t_h is the regression value of the width and height of the predicted bounding box; t_w*, t_h* is the regression value of the width and height of the ground truth bounding box; $t_i = \{t_x, t_y, t_w, t_h\}$ is a vector-prediction parametrised candidate frame coordinate; $t_i^* = \{t_x^*, t_y^*, t_w^*, t_h^*\}$ is the coordinate vector of real boundaries.

The following step is an alternate training process (Ren et al., 2017). To begin, transfer learning from an ImageNet pre-trained network to the proposed network for the region. Second, repeat the transfer learning from ImageNet to Fast RCNN. The Region Proposal Network proposes objects for Fast RCNN to consider for object locations. This detector network is then used to initialise RPN. Before fine-tuning the final layers, transfer learning from Fast RCNN to an RPN network and fix the convolutional layers. The neural network's head then performs the same thing. Finally, fine-tune the faster scene while keeping the shared convolutional layers alone. It is how things will eventually become completely convolutional.

Fast RCNN. The second stage contains a detection network that employs Fast RCNN (Ren et al., 2017). The RPN and shared convolutional features produce proposals, which are then input into the Region of Interest pooling (RoI) pooling layer, followed by the remaining layers of the backbone CNN to predict the class and class-specific box refinement for each proposal (Lokanath et al., 2017).

Social Distancing Monitoring

Social Distancing Monitoring is an approach that employs CCTV footage to observe and calculate the distance between vehicles. Figure 9 depicts the flowchart for Social Distancing Monitoring that recognises each vehicle in videos and shows the distance between vehicles through different coloured bounding boxes (Lim, Sadullah et al., 2022).

Bird's Eye View

Bird's Eye View is a method for detecting vehicles and converting them into points. It shows the distance between the points in the specifically drawn area. The various coloured points show the distance between vehicles. Figure 10 illustrates the flowchart of the Bird's Eye View on vehicle detection (Lim, Ali et al., 2022).

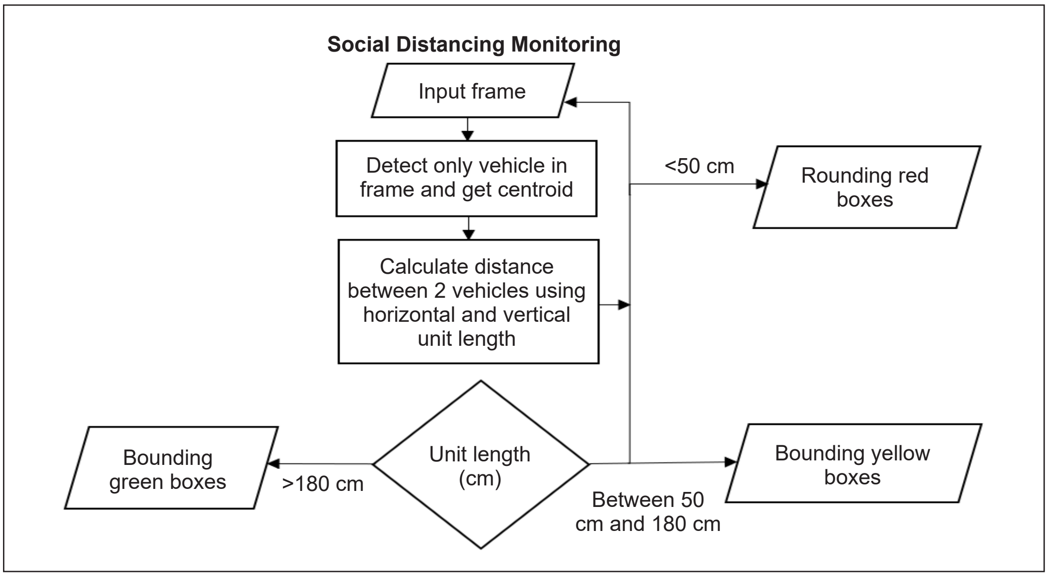

Social Distancing Monitoring
Input frame
Detect only vehicle in frame and get centroid
Calculate distance between 2 vehicles using horizontal and vertical unit length
<50 cm
Rounding red boxes
Unit length (cm)
Bounding green boxes
>180 cm
Between 50 cm and 180 cm
Bounding yellow boxes

Figure 9. Social Distancing Monitoring Algorithm (Lim, Ali et al., 2022)

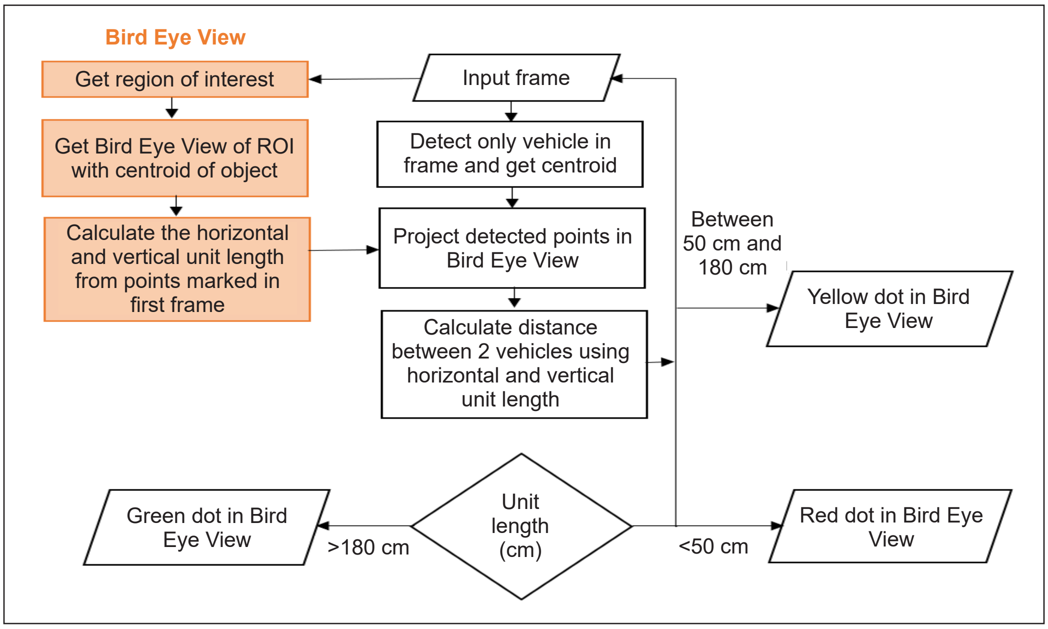

Bird Eye View
Get region of interest
Get Bird Eye View of ROI with centroid of object
Calculate the horizontal and vertical unit length from points marked in first frame
Input frame
Detect only vehicle in frame and get centroid
Project detected points in Bird Eye View
Calculate distance between 2 vehicles using horizontal and vertical unit length
Between 50 cm and 180 cm
Yellow dot in Bird Eye View
Unit length (cm)
Green dot in Bird Eye View
>180 cm
<50 cm
Red dot in Bird Eye View

Figure 10. Bird's Eye View algorithm (Lim, Sadullah et al., 2022)

RESULTS AND DISCUSSION

YOLOv3 Results

In YOLOv3, vehicle and near-miss detection were conducted together in Social Distancing Monitoring and Bird's Eye View. The Social Distancing Monitoring technique displays vehicle detection in monitoring and closeness in videos (Vinitha & Velantina, 2020), whereas the Bird's Eye View presents cars as points in a specified drawn box and displays the risk level. The algorithms described above are employed in YOLOv3 to observe and calculate real-world near misses (Ong, 2020). Only cars will be detected in this study.

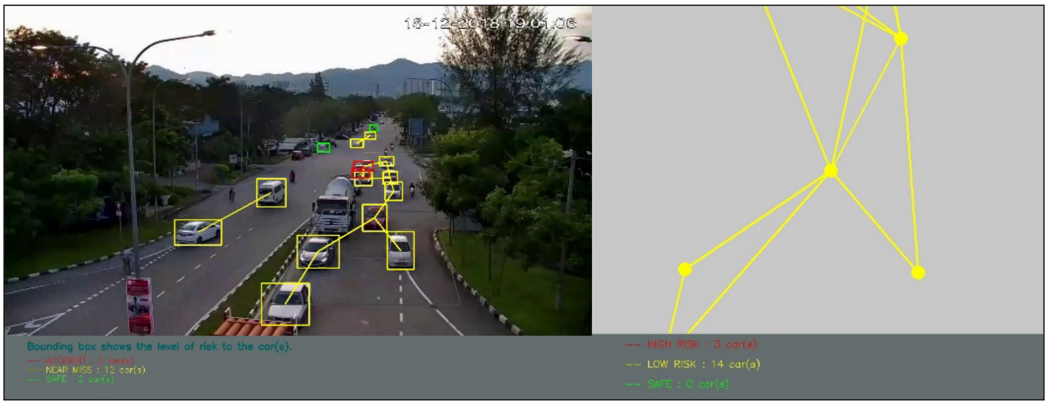


Figure 11. Near miss detection result in YOLOv3

Figure 11 exclusively demonstrates that YOLOv3 conducts vehicle detection to detect the likelihood of accidents and near misses between cars. The Lebuhraya Tun Dr Lim Chong Eu video was taken from 18/12/2018, 19:00:00 to 18/12/2018, 19:01:19. Only cars within the specified rectangle area will be displayed using the Bird's Eye View method. Because it is in a poor perspective, the results outside the rectangle region will be ignored, and only cars within the drawn rectangle zone will be counted.

Table 2 compares different durations of the same video (in the 20s, 40s, 60s, and 80s). In the research of Sonnleitner et al. (2020), three videos with a 10-minute duration and 30 frames per second were acquired online to carry out the study. When the length of the video is increased, the running time also increases. It is a reason for this study's choice of a video duration of 20 seconds since the computing time is too long for the computational procedure when conducting the image process using YOLOv3 compared to an 80-second video. Even though the data displayed in the 80-second video is more dependable than the data shown in the 20-second video, counting data manually is a huge undertaking for recording data. The longer the video length, the higher the number of frames obtained.

The videos show that the percentage of near-miss detection is high. It demonstrates that the vehicles in the videos are too near to one other while coming to a halt in front of a traffic signal or when stuck in traffic. So, the percentage of near-miss detection increased from 35.56% to 53.27%.

In the paper by Huang et al. (2020), the researchers gathered data from three distinct videos, each lasting 40 seconds. The videos contain different weather and scenarios. The YOLOv3 algorithm is used for video traffic monitoring. They also mention the vision field and the location of the video collection. The research shows missed detection when there is a large video data collection. If the field of view is too large, it will cause error detection. In

Table 2
Comparison of various length videos in YOLOv3

CCTV video time		19:00:00 – 19:00:19	19:00:19 – 19:00:39	19:00:39 – 19:00:59	19:00:59 – 19:01:19
Duration of video (s)		20	40	60	80
Computational time (s)		612	1223	1835	2447
Total number of frames, D		599	1199	1799	2399
Error detection	Number of frames, A	6	166	315	316
	Percentage, A/D x 100%	1 %	13.85 %	17.51 %	13.17 %
	Object detected	Motorcycle	Motorcycle, lorry, cement truck	Motorcycle, lorry, cement truck	Motorcycle, lorry, cement truck
Near miss detection	Number of frames, B	213	478	826	1278
	Percentage, B/D x 100%	35.56 %	39.87 %	45.91 %	53.27 %

this study, the duration of the 20s, 40s, 60s, and 80s videos were tested in the experiment. The vision-blocked view and the place of cameras might be factors in future work.

YOLOv4 Results

YOLOv4 is applied to conduct Social Distancing Monitoring and Bird's Eye View to simultaneously conduct vehicle and near-miss detection. Object detection and near-miss detection are displayed in the monitoring system.

Figure 12 displays car detection and near-miss detection by using YOLOv4. The cameras were set up at Lebuhraya Tun Dr Lim Chong Eu, and the footage was captured between 18/12/2018, 19:00:00, 18/12/2018, and 19:01:19. The image on the right resulted from using Bird's Eye View. The vehicles in the drawn boxes, or RoI, are displayed in the results.

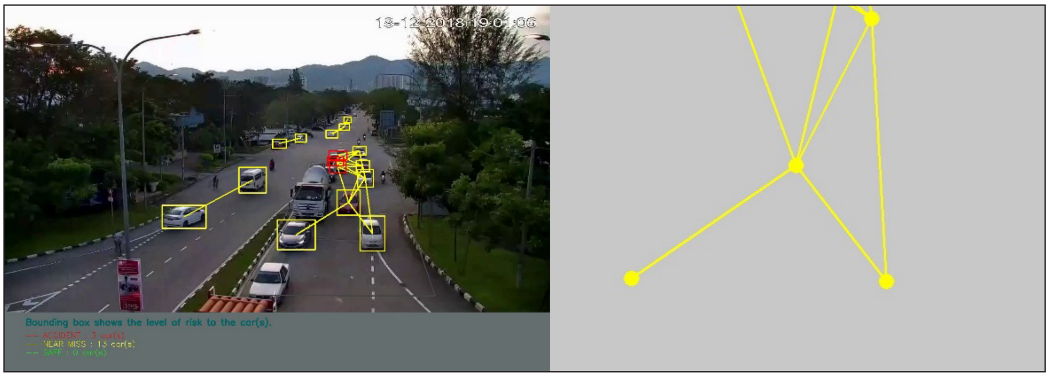


Figure 12. Near miss detection result in YOLOv4

Table 3 analyses the duration of 20s, 40s, 60s, and 80s videos by using the YOLOv4. In this experiment, a 20-second video was selected since the computation for the image process with YOLOv4 was much too long compared to an 80-second video. Even though the data in the 80-second video is more reliable than in the 20-second video, physically gathering data is a significant amount of labour for data recording. As the duration of the video rises, so does the total frame rate.

Sowmya and Radha (2021) proposed an algorithm's classifier (YOLOv4) that is tested using a computer vision approach. It uses a custom vehicle dataset that includes 3,500 images of trucks and buses. According to experimental data, the proposed technique has the best detection accuracy among various YOLO applications. Kumar et al. (2020) show that YOLOv4 investigates object detection in video and object detection in grayscale video. Objects are detected frame by frame in a video for various object categories in the sample image of video with a duration of 2 minutes and 22 seconds for multiple object detection and the sample image of grayscale video with a duration of 9 seconds for multiple object detection, which is input for multiple object detection.

Table 3
Comparison of various length videos in YOLOv4

CCTV video time		19:00:00 – 19:00:19	19:00:19 – 19:00:39	19:00:39 – 19:00:59	19:00:59 – 19:01:19
Duration of video (s)		20	40	60	80
Computational time (s)		671	1343	2015	2687
Total number of frames, D		599	1199	1799	2399
Error detection	Number of frames, A	11	174	326	434
	Percentage, A/D x 100%	1.87 %	14.51 %	18.12 %	18.09 %
	Object detected	Motorcycle	Motorcycle, lorry, cement truck	Motorcycle, lorry, cement truck	Motorcycle, lorry, cement truck
Near miss detection	Number of frames, B	354	835	1353	1913
	Percentage, B/D x 100%	59.1 %	69.64 %	75.21 %	79.74 %

Table 3
Comparison of various length videos in YOLOv4

CCTV video time		19:00:00 – 19:00:19	19:00:19 – 19:00:39	19:00:39 – 19:00:59	19:00:59 – 19:01:19
Duration of video (s)		20	40	60	80
Computational time (s)		671	1343	2015	2687
Total number of frames, D		599	1199	1799	2399
Error detection	Number of frames, A	11	174	326	434
	Percentage, A/D x 100%	1.87 %	14.51 %	18.12 %	18.09 %
	Object detected	Motorcycle	Motorcycle, lorry, cement truck	Motorcycle, lorry, cement truck	Motorcycle, lorry, cement truck
Near miss detection	Number of frames, B	354	835	1353	1913
	Percentage, B/D x 100%	59.1 %	69.64 %	75.21 %	79.74 %

Faster RCNN Results

The Faster RCNN only detected cars in the experiment data. There are four different lengths of footage: 20s, 40s, 60s, and 80s. Figure 13 only shows cars identified by the Faster RCNN. The green boxes represent spotted vehicles and display the percentage similarity of vehicles in training datasets.

Table 4 shows the results of comparing four different video durations. The time required to compute the algorithm increases as the video length increases. Due to the time taken by Faster RCNN being too long, this study used a 20-second video instead of an 80-second video. The number of frames taken increases, as does the video length. As a result, Faster RCNN takes longer to compute than YOLOv3 and YOLOv4 (Alganci et al., 2020). The accuracy results in Faster RCNN are higher than YOLO (Dixit et al., 2019). The research of Dixit et al. (2019) shows that the precision of Faster RCNN is the highest, but the speed of object detection is the slowest compared to other object detection models.

Faster RCNN has error-detected motorcycles, a lorry and a cement truck as cars. The percentage of error detection increased from 1.34% to 59.29%, correspondingly to 20s video until 80s video. It also does not detect small objects (cars) in further places.

Harianto et al.'s (2021) research employed the Google Open Image Dataset v6 and four different classes of vehicles. The study collected 1000 images from each class, with 800 photos utilised as training data and the rest as research data. The vehicle detection was then performed using Faster RCNN on the Tesla K80 GPU. According to Tariq et al. (2021), the data was collected independently. The collected dataset contains five different colours of cars and four different views of cars. Five hundred images were distributed to each class. Therefore, the researchers obtained 2500 images from 15 high-quality camera streams with a 20-minute duration.

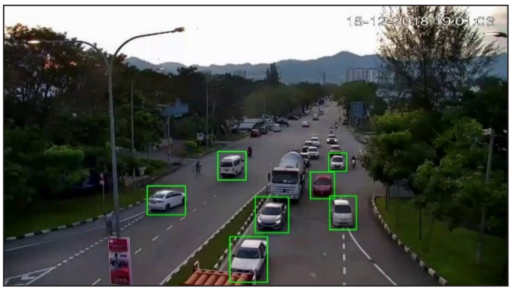


Figure 13. Result of vehicle detection in Faster RCNN

Table 4
Comparison of various length videos in Faster RCNN

	CCTV video time	19:00:00 – 19:00:19	19:00:19 – 19:00:39	19:00:39 – 19:00:59	19:00:59 – 19:01:19
	Duration of video (s)	20	40	60	80
	Computational time (s)	1534	3072	4608	6144
	Total number of frames, D	599	1200	1800	2400
Error detection	Number of frames, A	8	253	903	1423
	Percentage, A/D x 100%	1.34 %	21.08 %	50.17 %	59.29 %
	Object detected	Motorcycle	Motorcycle, lorry, cement truck	Motorcycle, lorry, cement truck	Motorcycle, lorry, cement truck

Comparison Between YOLOv3, YOLOv4 and Faster RCNN

The videos have been experimented with in YOLOv3, YOLOv4 and Faster RCNN. Table 5 compares YOLOv3, YOLOv4 and Faster RCNN in the videos. The following comparison is based on Tables 2, 3 and 4.

Table 5
Comparison detections in three systems

Difference	YOLOv3	YOLOv4	Faster RCNN
Method	Social Distancing Monitoring and Bird's Eye View	Social Distancing Monitoring and Bird's Eye View	Vehicle detection
Speed	Fast	Faster	Slow
Error detection	Higher than Faster RCNN	Higher than Faster RCNN	Low
Near miss detection	Very high	Lower than YOLOv3	null
Accident detection	Low	Lower than YOLOv3	null

The quality of the footage given by MBPP varies based on their position. They are not in the same place and have various levels of video quality. In 2015, MBPP deployed 534 CCTV cameras and another 1841 in 2019. Some places already installed CCTV cameras in 2015, although some were of poor quality compared to other locations.

Finally, it may be concluded that the comparison between Faster RCNN, YOLOv3 and YOLOv4. It requires more computing time to run Faster RCNN compared to YOLOv3 and YOLOv4 in videos because YOLOv3 and YOLOv4 have a simplified construction and Faster RCNN has been trained to do categorisation and regression of bounding boxes at the same time.

Faster RCNN displayed more precise results when compared to YOLOv3 and YOLOv4 in the footage of Lebuhraya Lim Chong Eu. Faster RCNN required more dataset samples depending on images or videos to train the algorithm, while YOLOv3 and YOLOv4 did not need to train the algorithm because their dataset trained it. Both algorithms have detection errors in vehicle detection. YOLOv3, YOLOv4 and Faster RCNN detected motorcycles, trucks, and lorries as vehicles in the Lebuhraya Lim Chong Eu videos.

From Table 5, the Faster RCNN does not have data for detecting near misses and accidents. Social Distancing Monitoring and Bird's Eye View techniques are unavailable at Faster RCNN.

Figure 14 depicts the test time in the videos utilising YOLOv3, YOLOv4 and Faster RCNN. YOLOv4 uses more computational time than YOLOv3. As revealed by Wang (2021), the detection efficiency of YOLOv3, YOLOv3-tiny, YOLOv3-SPP3, YOLOv4 and YOLOv4-tiny was comparable. When compared to YOLOv4 models, YOLOv3 models require less testing time. Faster RCNN has the highest computational time compared to YOLOv3 and YOLOv4 in both quality videos because it is a two-stage detector (Soviany & Ionescu, 2018).

Figure 15 displays the detection of errors in the videos using YOLOv3, YOLOv4 and Faster RCNN. Error detection happened in several cases, such as identifying motorcycles, cement trucks, and lorries as cars in the videos. When the cars are too close to one another, and at the traffic signal or junction, YOLOv4 detects more errors than YOLOv3.

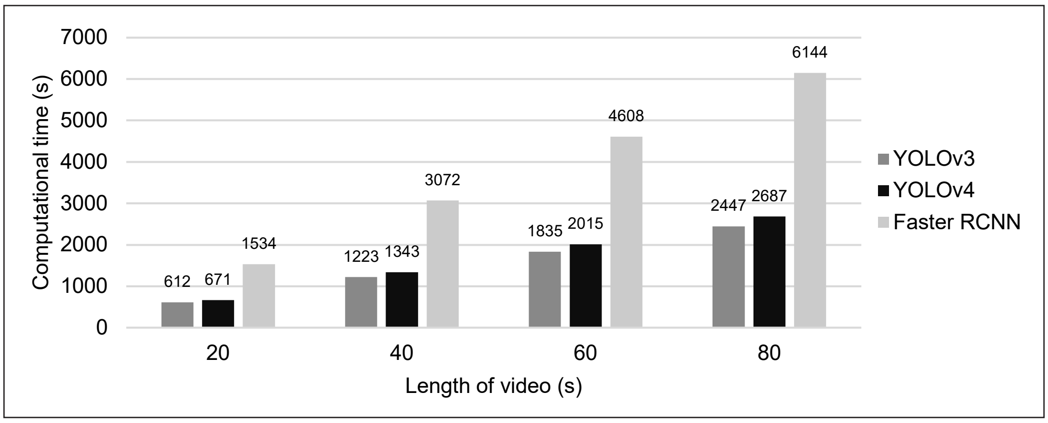


Figure 14. Comparison of computational time in three systems

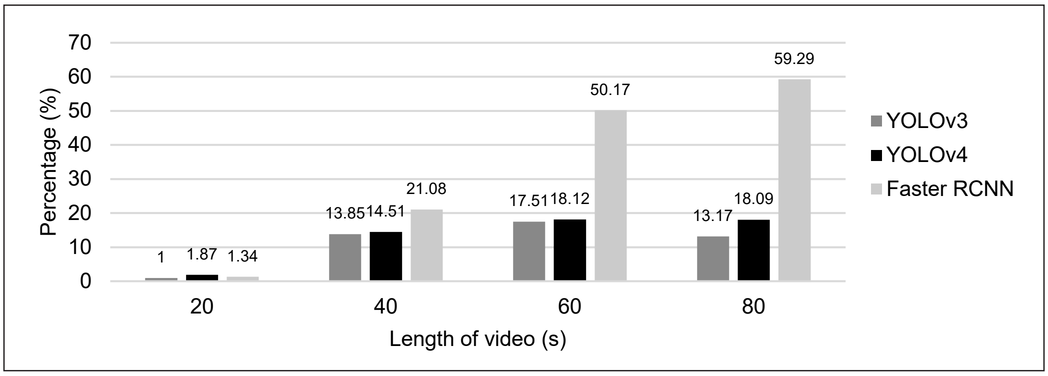


Figure 15. Comparison of error detection in three systems

Through the experiment, YOLOv3 shows higher accuracy than YOLOv4 and Faster RCNN. YOLOv3 and YOLOv4 can detect the car in the further part of the videos. Besides that, the experimental performance of YOLOv3 and YOLOv4 in identifying small and large vehicles is demonstrated by Nepal and Eslamiat (2020). In the research of Sowmya and Radha (2021), Faster R-CNN is not suited for usage in real-time videos since the system depends on the precision ratio to ensure detection speed.

Figure 16 displays the near-miss detection in the videos by applying YOLOv3, YOLOv4 and Faster RCNN. YOLOv4 shows a higher percentage of near misses than YOLOv3 in the videos. Since Faster RCNN does not have an algorithm to conduct the near-miss detection, there is no data for it.

This data lends weight to the idea that near misses occur during the busy hours in the black spot. Besides driving behaviour, the primary cause of near misses is also a major problem. According to Matsui et al. (2013), to reduce the number of deaths and serious injuries in Japan, the construction of driving safety systems requires precise functioning of the interaction scenario between the automobile and the pedestrian.

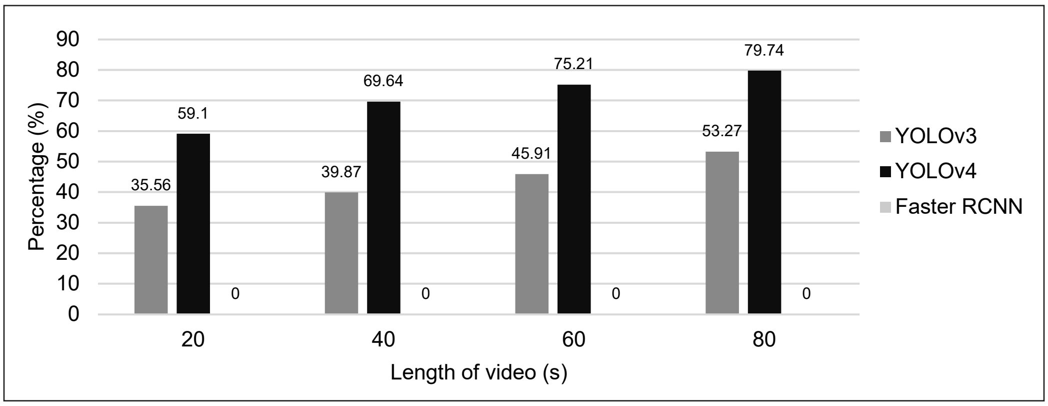


Figure 16. Comparison of near miss detection in three systems

Due to real-world accidents and a shortage of statistics, the researchers focused on near misses. As an outcome, the near miss is spotted utilising the methods of Bird's Eye View and Social Distancing Monitoring in the video recorded from the black spot position. Faster RCNN cannot detect near misses because of the failure of the Social Distancing Monitoring technique and Bird's Eye View method.

The reliability of this data is impacted by the video given by the Penang state government. The video is reliable and can be recorded as historical data for future usage. According to Calles et al. (2017), near misses warn drivers and prevent accidents. Driver behaviours and drivers' experience can contribute to accidents among young people. Future studies should collaborate with hospitals and insurance companies for more complete data and reduce the data flaws.

CONCLUSION

The ultimate purpose of this inquiry is to learn about near misses in Penang Island so that fatalities and air pollution can be reduced in the city. This project can achieve the goals mentioned in the Sustainable Development Goals (SDGs) and is concerned with the Penang 2030 mission. The time (18/12/2018, 7:00:00 p.m. to 7:01:19 p.m.) and Lebuhraya Tun Dr Lim Chong Eu black spot location are chosen as the experiment data to apply vehicle detection techniques YOLOv3, YOLOv4, and Faster RCNN. YOLOv3 and YOLOv4 were faster and more accurate than the Faster RCNN in vehicle detection. YOLOv3 performed better than YOLOv4 in precision and speed. The probability of near misses is high, as demonstrated by Lebuhraya Tun Dr Lim Chong Eu, which YOLOv3 and YOLOv4 calculated. The quality of CCTV video and the angle of cameras need to be enhanced in future work. The length of videos can be longer if the algorithm can calculate it automatically. Near misses and accidents can be predicted by traffic flow. It can also forecast future seasonal variations, assuring the success of the Penang 2030 objective.

ACKNOWLEDGEMENTS

The authors express their gratitude to the team of MBPP CCTV, Pulau Pinang, Malaysia, for their assistance in supplying important data, as well as the School of Mathematical Sciences, Universiti Science Malaysia, for the financing.

REFERENCES

Abdullah, A., & Ootthariasamy, J. (2020). Vehicle counting using deep learning models: A comparative study. *International Journal of Advanced Computer Science and Applications, 11*(7), 697-703. https://dx.doi.org/10.14569/IJACSA.2020.0110784

Abdurahman, F., Fante, K. A., & Aliy, M. (2021). Malaria parasite detection in thick blood smear microscopic images using modified YOLOV3 and YOLOV4 models. *BMC Bioinformatics, 22*(1), Article 112. https://doi.org/10.1186/s12859-021-04036-4

Aldred, R. (2016). Cycling near misses: Their frequency, impact and prevention. *Transport Research Part A, 90*(1), 69-83. https://doi.org/10.1016/j.tra.2016.04.016

Adarsh, P., Rathi, P., & Kumar, M. (2020, March). YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. In *2020 6th international conference on advanced computing and communication systems (ICACCS)* (pp. 687-694). IEEE Publishing. https://doi.org/10.1109/ICACCS48705.2020.9074315

Alganci, U., Soydas, M., & Sertel, E. (2020). Comparative research on deep learning approaches for airplane detection from very high-resolution satellite images. *Remote Sensing, 12*(3), Article 458. https://doi.org/10.3390/rs12030458

AlKishri, W., & Al-Bahri, M. (2021). Object recognition for organizing the movement of self-driving car. *International Journal of Computation and Applied Sciences, 10*(1), 1-8. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3828692

Ammar, A., Koubaa, A., Ahmed, M., Saad, A., & Benjdira, B. (2021). Vehicle detection from aerial images using deep learning: A comparative study. *Electronics, 10*(7), Article 820. https://doi.org/10.3390/electronics10070820

Arinaldi, A., Pradana, J., A., & Gurusniaga, A., A. (2018). Detection and classification of vehicles for traffic video analytics. *Procedia Computer Science, 144*, 259-268. https://doi.org/10.1016/j.procs.2018.10.527

Bochkovskiy, A., Wang, C., Y., & Liao, H., Y., M. (2020). *YOLOv4: Optimal speed and accuracy of object detection*. ArXiv Preprint. https://doi.org/10.48550/arXiv.2004.10934

Calles, M., B., Nelson, T., & Winters, M. (2017). Comparing crowdsourced near-miss and collision cycling data and official bike safety reporting. *Transportation Research Record: Journal of the Transportation Research Board, 2662*(1), 1-11. https://doi.org/10.3141/2662-01

Cepni, S., Atik, M. E., & Druan, Z. (2020). Vehicle detection using different deep learning algorithms from image sequence. *Baltic Journal of Modern Computing, 8*(2), 347-358. https://doi.org/10.22364/bjmc.2020.8.2.10

Ciberlin, J., Grbic, R., Teslic, N. and Pilipovic, M. (2019). Object detection and object tracking in front of the vehicle using front view camera. In *IEEE 2019 Zooming Innovation in Consumer Technologies Conference (ZINC)* (pp. 27-32). IEEE Publishing. https://doi.org/10.1109/ZINC.2019.8769367

Ding, X., & Yang, R. (2019). Vehicle and parking space detection based on improved YOLO network model. *Journal of Physics: Conference Series, 1325*, 1-7. https://doi.org/10.1088/1742-6596/1325/1/012084

Dixit, K. S., Chadaga, M. G., Savalgimath, S. S., Rakshith, G. R., & Kumar, M. N. (2019). Evaluation and evolution of object detection techniques YOLO and R-CNN. *International Journal of Recent Technology and Engineering (IJRTE), 8*(2S3), 824-829. https://doi.org/10.35940/ijrte.B1 154 07 0782S319

Ezat, W., A., Dessouky, M., M., & Ismail, N., A. (2021). Evaluation of deep learning YOLOv3 algorithm for object detection and classification. *Menoufia Journal of Electronic Engineering Research (MJEER), 30*(1), 52-57. http://dx.doi.org/10.21608/mjeer.2021.146237

Gai, J., Zhong, K., Du, X., Yan, K., & Shen, J. (2021). Detection of gear fault severity based on parameter-optimized deep belief network using sparrow search algorithm. *Measurement, 185*, Article 110079. https://doi.org/10.1016/j.measurement.2021.110079

Girshick, R. (2015). *Fast R-CNN*. ArXiv Preprint. https://doi.org/10.48550/arXiv.1504.08083

Gou, C., Peng, B., Li, T., & Gao, Z. (2019). Pavement crack detection based on the improved faster-RCNN. In *2019 IEEE 14th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)* (pp. 962-967). IEEE Publishing. https://doi.org/10.1109/iske47853.2019.9170456

Gulati, I., & Srinivasan, R. (2019). Image processing in intelligent traffic management. *International Journal of Recent Technology and Engineering (IJRTE), 8*(2S4), 213-218.

Harianto, R. A., Pranoto, Y. M., & Gunawan, T. P. (2021). Data augmentation and faster RCNN improve vehicle detection and recognition. In *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)* (pp. 128-133). IEEE Publishing. https://doi.org/10.1109/eiconcit50028.2021.9431863

Huang, Y. Q., Zheng, J. C., Sun, S. D., Yang, C. F., & Liu, J. (2020). Optimized YOLOv3 algorithm and its application in traffic flow detections. *Applied Sciences, 10*(9), Article 3079. https://doi.org/10.3390/app10093079

Jiang, Z., G., & Shi, X., T. (2021). Application research of key frames extraction technology combined with optimized faster R-CNN algorithm in traffic video analysis. *Complexity, 2021*, Article 6620425. https://doi.org/10.1155/2021/6620425

Jiang, Z., Zhao, L., Li, S., & Jia, Y. (2020). *Real-time object detection method for embedded devices*. ArXiv Preprint. https://doi.org/10.48550/arXiv.2011.04244

Kumar, B. C., Punitha, R., & Mohana. (2020). YOLOv3 and YOLOv4: Multiple object detection for surveillance applications. In *2020 Third international conference on smart systems and inventive technology (ICSSIT)* (pp. 1316-1321). IEEE Publishing. https://doi.org/10.1109/icssit48917.2020.9214094

Lim, L. M., Ali, M. K. M., Ismail, M. T., & Mohamed, A. S. A. (2022). Data safety prediction using bird's eye view and social distancing monitoring for Penang roads. *Pertanika Journal of Science & Technology, 30*(4), 2563-2587. https://doi.org/10.47836/pjst.30.4.15

Lim, L. M., Sadullah, A. F. M., Ismail, M. T., Awang, N., & Ali, M. K. M. (2022). Penang data safety prediction using database development and alternative road identification using Dijkstra approach. *AIP Conference Proceedings, 2465*, Article 040007. https://doi.org/10.1063/5.0078252

Lokanath, M., Kumar, K. S., & Keerthi, E. S. (2017). Accurate object classification and detection by faster-RCNN. *IOP Conference Series: Materials Science and Engineering, 263*, Article 052028. https://doi.org/10.1088/1757-899x/263/5/052028

Mahto, P., Garg, P., Seth, P., & Panda, J. (2020). Refining Yolov4 for vehicle detection. *International Journal of Advanced Research in Engineering and Technology (IJARET), 11*(5), 409-419.

Manne, G., & Raghuwanshi, N. (2017). Review on vehicle detection techniques. *International Journal of Science, Engineering and Technology Research (IJSETR), 6*(4), 482-486.

Maqbool, S., Khan, M., Tahir, J., Jalil, A., Ali, A., & Ahmad, J. (2018). Vehicle detection, tracking and counting. In *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)* (pp. 126-132). IEEE Publishing. https://doi.org/10.1109/SIPROCESS.2018.8600460

Matsui, Y., Hitosugi, M., Doi, T., Oikawa, S., Takahashi, K., & Ando, K. (2013). Features of pedestrian behavior in car-to-pedestrian contact situations in Near-Miss incidents in Japan. *Traffic Injury Prevention, 14*(1), 58-63. https://doi.org/10.1080/15389588.2013.796372

Meng, C., Bao, H., & Ma, Y. (2020). Vehicle detection: A review. *Journal of Physics: Conference Series CISAT 2020, 1634*, Article 012107. https://doi.org/10.1088/1742-6596/1634/1/012107

Moutakki, Z., Ouloul, I. M., Afdel, K., & Amghar, A. (2018). Real-time system based on feature extraction for vehicle detection and classification. *Transport and Telecommunication, 19*(2), Article 93. https://sciendo.com/article/10.2478/ttj-2018-0008

Nepal, U., & Eslamiat, H. (2020). Comparing YOLOv3, YOLOv4 and YOLOv5 for autonomous landing spot detection in faulty UAVs. *Sensors, 22*(464) 1-15. https://doi.org/10.3390/s22020464

Nousi, P., Triantafyllidou, D., Tefas, A., & Pitas, I. (2019). Joint lightweight object tracking and detection for unmanned vehicles. In *2019 IEEE International Conference on Image Processing (ICIP)* (pp. 160-164). IEEE Publishing. https://doi.org/10.1109/ICIP.2019.8802988

Ong, Y. (2020). *Near Miss Vehicle Collisions Estimation using YOLO*. (Unpublished master's thesis). Universiti Sains Malaysia, Malaysia.

Pawar, B., Humbe, V., & Kundnani, L. R. (2017). Morphological approach for moving vehicle detection. *IOSR Journal of Computer Engineering (IOSR-JCE), 3*(1), 73-80.

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An incremental improvement.* ArXiv Preprint. https://doi.org/10.48550/arXiv.1804.02767

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788). IEEE Publishing. https://doi.org/10.1109/cvpr.2016.91

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 39*(6), 1137-1149. https://doi.org/10.1109/tpami.2016.2577031

Rin, V., & Nuthong, C. (2019). Front moving vehicle detection and tracking with Kalman Filter. In *IEEE 4th International Conference on Computer and Communication Systems* (pp. 304-310). IEEE Publishing. 10.1109/CCOMS.2019.8821772

Sekar, A., & Perumal, V. (2021). Automatic road crack detection and classification using multi-tasking faster RCNN. *Journal of Intelligent & Fuzzy Systems, 41*(6), 6615-6628. https://doi.org/10.3233/jifs-210475

Silva, L. A., Sanchez San Blas, H., Peral García, D., Sales Mendes, A., & Villarubia González, G. (2020). An architectural multi-agent system for a pavement monitoring system with pothole recognition in UAV images. *Sensors, 20*(21), Article 6205. https://doi.org/10.3390/s20216205

Sonnleitner, E., Barth, O., Palmanshofer, A., & Kurz, M. (2020). Traffic measurement and congestion detection based on real-time highway video data. *Applied Sciences, 10*(18), Article 6270. https://doi.org/10.3390/app10186270

Soviany, P., & Ionescu., R., T. (2018). *Optimizing the trade-off between single-stage and two-stage deep object detectors using image difficulty prediction*. ArXiv Preprint. https://doi.org/10.48550/arXiv.1803.08707

Sowmya, V., & Radha, R. (2021). Comparative analysis on deep learning approaches for heavy-vehicle detection based on data augmentation and transfer-learning techniques. *Journal of Scientific Research, 13*(3), 809-820. https://doi.org/10.3329/jsr.v13i3.52332

Tariq, A., Khan, M. Z., & Khan, M. U. G. (2021). Real time vehicle detection and colour recognition using tuned features of faster-RCNN. In *2021 1st International Conference on Artificial Intelligence and Data Analytics (CAIDA)* (pp. 262-267). IEEE Publishing. https://doi.org/10.1109/caida51941.2021.9425106

Uus, J., & Krilavičius, T. (2018). Vehicle detection and classification in aerial imagery. *IEEE International Conference on Image Processing, 2410*(1), 80-85.

Vançin, S., & Erdem, E. (2018). Detection of the vehicle direction with adaptive threshold algorithm using magnetic sensor nodes. *Journal of Polytechnic, 21*(2), 333-340.

Vinitha, V., & Velantina, V. (2020). Social distancing detection system with artificial intelligence using computer vision and deep learning. *International Research Journal of Engineering and Technology (IRJET), 7*(8), 4049-4053.

Wang, C., Dai, Y., Zhou, W., & Geng, Y. (2020). A vision-based video crash detection framework for mixed traffic flow environment considering low-visibility condition. *Hindawi, Journal of Advanced Transportation, 2020*, Article 9194028. https://doi.org/10.1155/2020/9194028

Wang, G., Guo, J. M., Chen, Y., Li, Y., & Xu, Q. (2019). A PSO and BFO-based learning strategy applied to faster R-CNN for object detection in autonomous driving. *IEEE Access, 7*, 18840-18859. https://doi.org/10.1109/access.2019.2897283

Wang, S. (2021). Research towards yolo-series algorithms: Comparison and analysis of object detection models for real-time UAV applications. *Journal of Physics: Conference Series, 1948*(1), Article 012021. https://doi.org/10.1088/1742-6596/1948/1/012021

World Health Organization. (2020, December 9). *The Top 10 Causes of Death*. World Health Organization. https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death

Zhang, Y., Shen, Y., & Zhang, J. (2019). An improved tiny-YOLOv3 pedestrian detection algorithm. *Optik, 183*, 17-23. https://doi.org/10.1016/j.ijleo.2019.02.038

Zhang, Z., Trivedi, C., & Liu, X. (2018). Automated detection of grade-crossing-trespassing near misses based on computer vision analysis of surveillance video data. *Safety Science, 110*(Part B), 276-285. https://doi.org/10.1016/j.ssci.2017.11.023

Zhao, J., Wei, S., Xie, H., & Zhong, H. (2020). Image recognition of small UAVs based on faster RCNN. In *Proceedings of the 2020 4th International Conference on Vision, Image and Signal Processing* (pp. 1-6). ACM Publishing. https://doi.org/10.1145/3448823.3448844

Zhao, X., Pu, F., Wang, H., Chen, H. and Xu, Z. (2019). Detection, tracking, geolocation of moving vehicle from UAV using monocular camera. *IEEE Access, 7*(1), 101160-101170.

Zohra, A., F., Kamilia, S., & Souad, S. (2018). Detection and classification of vehicles using deep learning. *International Journal of Computer Science Trends and Technology (IJCST), 6*(3), 23-29.

APPENDIX A

No.	Researchers	Type of paper		Application data		Type of method				Objects detected			Remarks
		Review	Research	Image process	Empirical data	Survey/ Questionnaire	Model	Software	Others	Vehicles	Pedestrians	Motorcycles/ Bicycles	
1	Manne and Raghuwanshi (2017)		✓	✓				✓		✓			- MATLAB - India
2	Pawar et al. (2017)		✓	✓			✓	✓					- Edge Detection - Top-Hat Processing - Masking Operation - India
3	Arinaldi et al. (2018)		✓	✓			✓	✓		✓	✓		- Region Convolutional Neural Networks (RCNN) - Support Vector Machine (SVM) - Indonesia
4	Maqbool et al. (2018)		✓	✓			✓	✓					- Gaussian Mixture Model - Blob detection - Hungarian Algorithm - Kalman Filter - Pakistan
5	Moutakki et al. (2018)		✓	✓			✓	✓		✓		✓	- Background subtraction - Filtering and contour detection - North Africa
6	Vançin and Erdem (2018)		✓	✓			✓			✓			- Adaptive threshold detection algorithm (ATDA) - Turkey

No.	Researchers	Type of paper		Application data			Type of method			Objects detected			Remarks
		Review	Research	Image process	Empirical data	Survey/ Questionnaire	Model	Software	Others	Vehicles	Pedestrians	Motorcycles/ Bicycles	
7	Zhang et al. (2018)		✓	✓			✓			✓			- Histograms of Oriented Gradients - AdaBoost algorithm - Image segmentation - Edge detection - Kalman filter - China
8	Zohra et al. (2018)		✓	✓			✓	✓		✓			- CNN - North Africa
9	Ciberlin et al. (2019)		✓	✓				✓		✓			- Viola-Jones algorithm - YOLOv3 - Median Flow tracking - Correlation tracking method - Croatia
10	Ding and Yang (2019)		✓	✓				✓		✓			- YOLOv3 - China
11	Gulati & Srinivasan (2019)	✓		✓					✓				- Comparison between detector systems and algorithms of image processing - India
12	Nousi et al. (2019)		✓	✓			✓	✓		✓			- Kernelised Correlation Filter - YOLOv2 - Greece
13	Rin and Nuthong (2019)		✓	✓			✓			✓			- Kalman Filter - Thailand
14	Uus and Krilavičius (2019)		✓	✓				✓		✓			- YOLOv3 - Lithuania

No.	Researchers	Type of paper		Application data		Type of method				Objects detected			Remarks
		Review	Research	Image process	Empirical data	Survey/ Questionnaire	Model	Software	Others	Vehicles	Pedestrians	Motorcycles/ Bicycles	
15	Zhao et al. (2019)		✓	✓			✓	✓		✓			- YOLOv3 - Kernelised Correlation Filter - China
16	Abdullah and Ootthariasamy (2020)		✓	✓			✓	✓		✓			- YOLOv3 - KL, Malaysia
17	Cepni et al. (2020)		✓	✓			✓	✓		✓			- YOLOv3 - Turkey
18	Mahto et al. (2020)		✓	✓			✓	✓		✓			- CNN -YOLOv4 - India