INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Boosting Vehicle Classification with Augmentation Techniques across Multiple YOLO Versions

Shao Xian Tan [a], Jia You Ong [a], Kah Ong Michael Goh [a,*], Connie Tee [a]

[a] *Faculty of Information Science & Technology (FIST), Multimedia University, Melaka, 75450, Bukit Beruang, Malaysia*

*Corresponding author: *michael.goh@mmu.edu.my*

Abstract*— In recent years, computer vision has experienced a surge in applications across various domains, including product and quality inspection, automatic surveillance, and robotics. This study proposes techniques to enhance vehicle object detection and classification using augmentation methods based on the YOLO (You Only Look Once) network. The primary objective of the trained model is to generate a local vehicle detection system for Malaysia which have the capacity to detect vehicles manufactured in Malaysia, adapt to the specific environmental factors in Malaysia, and accommodate varying lighting conditions prevalent in Malaysia. The dataset used for this paper to develop and evaluate the proposed system was provided by a highway company, which captured a comprehensive top-down view of the highway using a surveillance camera. Rigorous manual annotation was employed to ensure accurate annotations within the dataset. Various image augmentation techniques were also applied to enhance the dataset's diversity and improve the system's robustness. Experiments were conducted using different versions of the YOLO network, such as YOLOv5, YOLOv6, YOLOv7, and YOLOv8, each with varying hyperparameter settings. These experiments aimed to identify the optimal configuration for the given dataset. The experimental results demonstrated the superiority of YOLOv8 over other YOLO versions, achieving an impressive mean average precision of 97.9% for vehicle detection. Moreover, data augmentation effectively solves the issues of overfitting and data imbalance while providing diverse perspectives in the dataset. Future research can focus on optimizing computational efficiency for real-time applications and large-scale deployments.

***Keywords*— Vehicle detection; vehicle classification; object detection; YOLO; computer vision.**

Manuscript received 5 Dec. 2022; revised 9 Jul. 2023; accepted 25 Sep. 2023. Date of publication 31 Mar. 2024.

I. Introduction

The increase in transportation has become a critical issue due to population growth, urbanization, and increased motor vehicle ownership. The high traffic density and increasing motor vehicle ownership inevitably cause numerous vehicle traffic problems, including but not limited to traffic congestion, public safety, environmental impact, illegal driving, and road planning issues. Therefore, Intelligent Transportation Systems (ITS) have drawn considerable attention and have become essential traffic aids for managing traffic issues. For example, traffic surveillance is one of the critical components in ITS, which can extract helpful traffic information from traffic image analysis and traffic flow control such as vehicle tracking and counting [1], license plate recognition [2], vehicle velocity [3], wrong-way vehicle detection [4] and so on. Besides, traffic surveillance can be derived as a traffic impact assessment system [5] to inform people about traffic activities and road conditions in any monitored area to solve traffic problems.

Computer vision applications for the categorization of autonomous vehicles have a long history. Cameras are frequently deliberately positioned at vantage locations for computer vision-based systems to gather vehicle information in the region below from a top-to-bottom perspective, such as next to traffic signals, beneath flyovers, or on taller poles. In this case, a weatherproof and high-resolution surveillance camera is needed to capture images of the waiting area in different weather conditions. Low-resolution images and image distortion may lead to a wrong judgment in computer vision systems, like a human may have myopia. Dongbin Zhao [6] proposed a feature miming human visual attention by formulating a visual attention module with focused images with clear vital regions. A focused picture is created by emphasizing one area of an image while weakening the others using a visual attention-based image processing module. Although many different models can perform image classification and achieve good results, but the emergence of YOLO has defeated many traditional models by achieved faster speed and higher [7], [8], [9], [10]. Hence, this article chooses to use YOLO as the main model architecture for better understand the improvements brought by different YOLO versions to image classification performance and different data enhancement techniques.

Alamgir et al. [11] conducted a compresence of tests on yolov3 and yolov4 architecture to find out the most suitable architecture for traffic analysis and vehicle detection in Thailand. The authors emphasize that the performance of deep learning algorithms largely depends on the quality of the dataset, street type diversification and maintaining a balance of the labelled can effectively improve model performance. In another study conducted by Zuraimi et al. [12], a vehicle detection and tracking system was developed using YOLOv3 and YOLOv4 but it can only classify 4 different types of vehicles. Neupane et al. [13] developed a real-time vehicle detection and tracking system that could classify 7 types of cars using YOLOv3 and YOLOv5. This study demonstrated that the trained YOLOv5l model remained stable and maintained high accuracy levels even under different image quality and noise conditions. YOLOv3, YOLOv4 and YOLOv5 were selected to develop a vehicle detection system and study the detection performance of each YOLO version [14], [15]. The author state that YOLOv5 take longest training time and show unpredictable results in non-uniform lighting and occluded objects. An experimental study has evaluated the performance of YOLOv5 and YOLOv7 in sandy weather environments [16]. The result showed that YOLOv7 have achieved 94% mAP on the augmented dataset.

The study conducted by Terven et al. [17] provides a broad and systematic overview of different YOLO variant architectures and their evolution. As highlighted in a comprehensive review by Diwan et al. [18], this study compared two-stage detectors and one-stage detectors and the challenges in the field of computer vision. Small object detection has always been a difficult problem in computer vision, Gunawan et al. [19] proposed to use YOLOv3 and yolov5 to detect images taken by drones to verify the performance of the yolo series for small object detection. Gillani et al. [20] use YOLOv5, YOLO-X, YOLO-R and YOLOv7 to do the model performance comparison and architectures evaluation. The findings show that while YOLOv7 performs well in terms of accuracy compared to other YOLO variants, it lags behind in terms of FPS rate. In comparison, YOLOv5 provides a good balance with moderate accuracy and the highest FPS rate, which makes it a better choice for real-time detection. Besides, data augmentation was found to be a cost-effective way to improve the accuracy and performance of YOLO. [21], [22], [23] have indicate that data augmentation can help YOLO to improve the accuracy and help the model to learn more about the target object. Data augmentation allows the model to learn more image features and improve the overall performance by increasing diversity and variability in the training data.

This paper chooses YOLO, the most popular model architecture in object detection and computer vision, to develop the vehicle detection model for performance evaluation. The advantages of the YOLO are that it can directly train on full images compared with classifier-based approaches, and it pushes the state-of-the-art in real-time object detection [24]. For example, YOLO will make predictions based on the entire picture information, while other sliding window detection frameworks can only make inferences based on local picture information. YOLO is a single-stage object detector that can solve object detection as a simple regression problem and takes much less inference time than other algorithms. Since YOLO significantly outperforms other algorithms, this study investigates the accuracy of different versions of YOLO under different environments, lighting conditions and viewing perspective. Hence, the objective of this YOLO research work is to compare which version of YOLO has the highest accuracy in vehicle detection and develop a local vehicle classification system to realize the ITS in the future to ensure road safety in Malaysia.

II. Materials and Method

From related work, the previous researcher's detection of vehicle types was limited and only covered a few distinct categories which resulting in a lack of comprehensiveness. Furthermore, most of the current research on vehicle detection using YOLO focuses on a single version, and does not fully consider the impact of different YOLO versions on vehicle detection. Moreover, the local researcher conducted in Malaysia is more focused on speed estimation, traffic sign board recognition and number plate recognition while lacks of local vehicle detection system. Consequently, there is a need to generate a local vehicle detection system for Malaysia which have the capacity of detecting vehicles manufactured in Malaysia, adapting to the specific environmental factors in Malaysia and accommodating varying lighting conditions prevalent in Malaysia. The proposed solution aims to expand the range of detectable vehicle types on the road, analyses the effectiveness of different YOLO versions for vehicle detection, and take into account Malaysia's unique environmental and lighting conditions.

A. Data Collection

Data collection is important for ensuring the accuracy of training for a machine learning model because the model is only as good as the data it is trained on. Models can also be inaccurate if the data used to train it is incomplete, biased, or inaccurate. So, it is important to have a diverse and representative dataset of images that includes all the different types of vehicles that the model should be able to recognize such as vehicle in occluded or non-occluded images. Not only that, but it is also essential to gather photos of vehicles from various angles to provide the trained model with a wide field of view and the ability to recognize the vehicle from different perspectives shown as Fig. 1. This is because if the model is only trained on photos taken from a single direction, it will only be able to recognize objects from that angle.

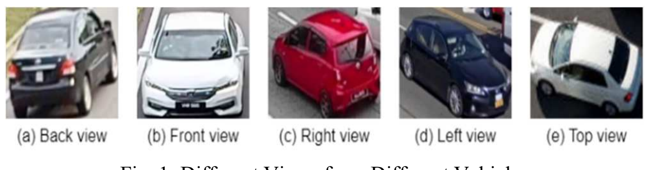


Fig. 1 Different Views from Different Vehicles

The data for this study was sourced from videos of two highway segments provided by highway companies. To generate the dataset, this study used FFmpeg to extract frames every 30 seconds apart from 24 videos of highway segments. The video length for each video in the highway segment is approximately 50 minutes and the selected video encompasses various lighting conditions including morning, afternoon, and night to capture a comprehensive range of environmental scenarios. The original dataset collected consisted of approximately 2,416 samples before applying any data augmentation techniques. Nevertheless, the dataset size has increased to a total of 5,878 samples after implementing data augmentation techniques and adding online open sources data. However, there is a limitation where the videos are captured from a top-down perspective of the road so that the dataset will only show a single view, as shown as Fig. 2. In this case, data augmentation techniques were employed to diversify the dataset and overcome the limitation of a single perspective. Next, a data partitioning method called a train-test split is used to separate the dataset in this project before training the models. In this project, the proportion of the training data set is 60%, the test data set is 25% and the validation data set is 15%. This dataset also includes 8% of background images that with no object labels to reduce false positives.

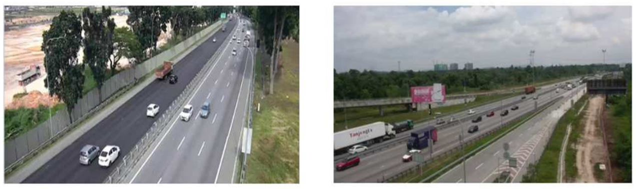


Fig. 2 Example Image of the Datasets

The decision to use a Malaysia local vehicle dataset for training was based on the unique environmental factors of Malaysia compared to other countries and the different culture and concept of car ownership. For example, the midday sunlight of Malaysia is much stronger than in other countries which can cause high exposure issues since Malaysia is an equatorial country. Additionally, Malaysia has locally produced vehicles brand which are relatively rare in other countries and the culture of car ownership in Malaysia is small-sized vehicles are more prevalent compared to larger vehicles. However, it is more common to own larger vehicles such as SUV or pickups in foreign countries because of better protection and better cargo capacity due to differences caused by cultural differences. According to a survey conducted by Eichelberger et al. [25], another reason of larger vehicle more popular in other country is because the majority of the parents believe that larger vehicle like SUV or pickup have better collision protection performance for their safety and prioritize as a suitable choice for their or their children's vehicles in the UK. From the two pictures of Malaysia and foreign vehicles obtained from the Internet, it can be clearly seen that the proportion of large vehicles will be more than Malaysia and Malaysians more prefer small vehicles as shown as Fig. 3. By focusing on local conditions, the detection system can better address the particular problems posed by the environment and reliably classify smaller vehicles that are more commonly seen in Malaysia. Therefore, the presence of locally produced car brands in Malaysia further justifies the use of local datasets for training. These locally produced vehicles may have unique design features and characteristics that differ from foreign vehicles and it is important that detection systems can accurately identify and classify them.

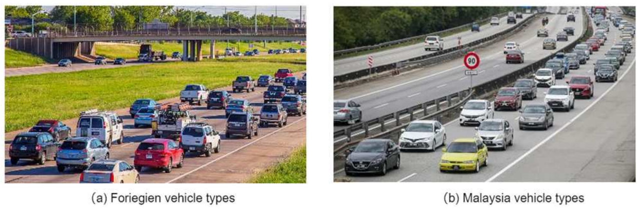


Fig. 3 Example of Foreign and Malaysia Vehicle Types

B. Data Labelling

After collecting many images related to the vehicles traffic, labelling images is crucial in object detection for identifying the specific areas within an image that contain the desired object (known as the region of interest or ROI). It is typically designated by four coordinates shown as Fig. 4: the x and y represent as the coordinate of the starting point of the ROI, the width of the object of interest and the height of the object of interest. Annotation of images was performed using a free annotation tool called Labeling which allows image annotation and saving in YOLO format. In this study, a total of 6 different vehicle types were labeled according to their characteristics as follows: C1 for small cars, C2 for large cars, C3 for vans and trucks, C4 for large trucks, c5 for buses and C6 for motorcycles. For more details to the vehicle category, C1 vehicles are primarily small-sized passenger cars that suitable for urban commuting or personal use. C2 is the abbreviation of pickup truck or SUV type vehicle which is a civilian vehicle with an open rear cargo compartment and usually used to transport personal items. Besides, C4 are heavy-duty vehicles with more than four wheels and often employed to deliver huge cargo across long distances.

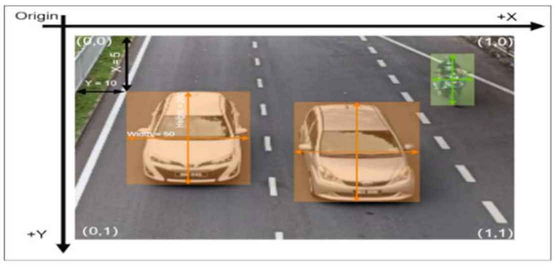


Fig. 4 ROI of an Interest Object

There are some important considerations when labeling ROI of objects in images. If an occluded object is more than 50% visible and can be easily identified by a human, it can be marked as fully visible even if it is occluded by another object. Otherwise, the object should not be labeled as it is not possible to recognize the category due to occlusion by another object. On the other hand, objects that are partially out of the frame and still can recognize to one of the vehicle classes should typically be labeled as fully visible. Additionally, it is generally recommended to include a small margin of non-object pixels around the object being labeled to avoid cutting off any part of the object with a rectangle label.

C. Data Imbalance

Since the dataset collection focused on highway environment, there are some limitations within the dataset including low number of occurrences of certain vehicle types, especially bus and motorcycle. There are many different methods that have been invented to solve the problem of data imbalance and improve the accuracy of machine learning models. In this project, data augmentation and adding online open-source data for certain categories are the main methods to solve the data imbalance. These added open-source datasets, which focus on buses and motorcycles, were obtained from Roboflow and Google.

Data augmentation is a strategy aimed at making the entire database more robust and applied to solve the problem of small datasets. It can improve the depth and quantity of the total images in the dataset by applying techniques such as flip, mix-up and color jitter to the dataset. This helps alleviate the problem of data type monotony by generating more diverse image views from original images, as more diverse datasets with different viewing angles and different environment condition can improve the overall detection accuracy of the model. Wong et al. [26] show that data augmentation can significantly improve performance and reduce overfitting when the data is known and correct.

Mix-up is a data augmentation technique that uses weighted averages to generate new datasets by combining two existing examples. This study extracted images containing four types of lesser vehicle annotations (bus, truck, large truck, and motorcycle) for mix-up augmentation. Besides, horizontal flip augmentation was used to diversify the dataset by introducing another angle of the vehicle. Color jitter data augmentation was used to change the brightness of an image to simulate nighttime or exposure environment to enhance the robustness of the model. In this case, data augmentation allows to address the imbalanced data by creating more diverse views of images of fewer vehicle types as shown in Fig. 5.

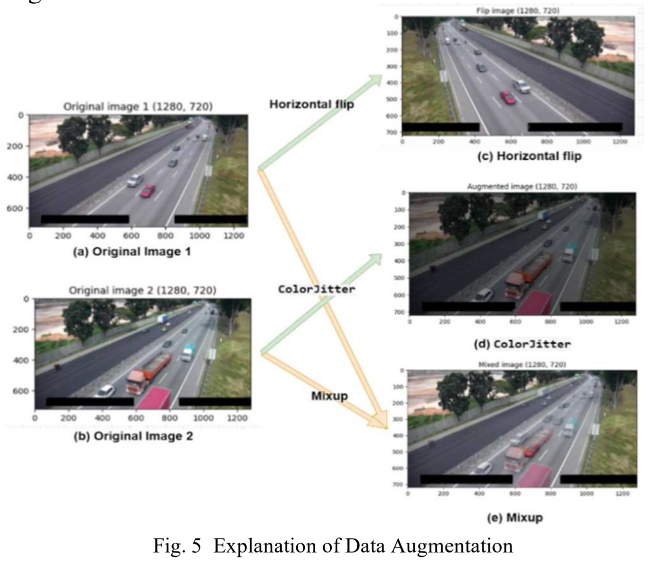


Fig. 5 Explanation of Data Augmentation

D. YOLO Variants

Joseph Redmon and his team first released YOLO in 2016 [24]. The release of the first version of YOLO has aroused great interest and research in the object detection community because it significantly reduced computing time and achieved high real-time performance and accuracy. It served as the foundation for subsequent versions of YOLO and spurred the creation of additional cutting-edge object identification frameworks. This study selected YOLOv5, YOLOv6, YOLOv7, and YOLOv8, relatively mature and modern versions of YOLO variants, to compare and experiment.

YOLOv5 [27] was the first version created in the PyTorch framework, which simplified configuration and improved the performance compared to earlier versions developed using the Darknet research framework. YOLOv5 architecture consists of three components: Neck, Backbone, and Head. First, the backbone of YOLOv5 utilizes the cross-stage partial networks (CSP) to extract essential features from the input images but increases training time. Next, the feature pyramid model PANet is used as the Neck component to improve the detection performance of objects of different scales. Lastly, the head performs final detection to generate output vectors for each predicted object in the input image using bounding boxes, object scores, and class probabilities.

YOLOv6 [28] extensively upgrades YOLOv5 regarding the accuracy, training time, and inference performance. A re-parameterizable and more efficient backbone network, EfficientRep Backbone and Rep-PAN Neck are designed based on the RepVGG style to replace the CSP network in YOLOv5. The author optimizes the design of a more concise and compelling Efficient Decoupled Head, which further reduces the additional delay caused by the general decoupled head while maintaining accuracy. The author adopts the Anchor-free paradigm in the training strategy, supplemented by the SimOTA label allocation strategy and the SIoU bounding box regression loss to improve detection accuracy. Besides, quantization-aware training (QTA) and post-training quantization (PTQ) are applied into YOLOv6 to improve inference speed without significantly reducing network performance. In summary, YOLOv6 reduces hardware latency, significantly improves algorithm accuracy, and makes detection networks faster.

YOLOv7 [29] is a widely used model in computer versions and machine learning due to its high detection capability. YOLOv7 integrated several techniques to improve the performance and efficiency of trained models. Extended Efficient Layer Aggregation Network (E-ELAN) replaced the original Backbone to control the gradient path to improve the learning and convergence. YOLOv7 combines various bag-of-freebies techniques to enhance the model performance further. These techniques include using reparametrized convolutions (RepConv) without identity connections (called RepConvN), coarse label assignment for the auxiliary head, and acceptable label assignment for the bootstrap head. These improvements ensure that YOLOv7 can perform real-time detection while maintaining high frames per second (FPS).

The same author of YOLOv5 has released YOLOv8 from the Ultralytics team [30]. YOLOv8 is an anchor-free detection model with a decoupled head that can independently handle abjectness, classification, and regression tasks. This dramatically reduces the number of frame predictions and speeds up the processing of Non-Max Suppression. YOLOv8 uses a modified backbone CSPDarknet53 feature extractor like YOLOv5, where the original C3 block is changed to a newly created cross-stage partial bottleneck with two convolutional (C2f) blocks. In the new C2f block, all the outputs of the Bottleneck are connected, while the C3 block uses only the last production of the Bottleneck. In the neck, feature concatenation occurs without enforcing the same channel size, which reduces parameters and overall tensor size. The author also packs the latest YOLOv8 into a python package to reduce the trouble of creating and installing a python environment suitable for running YOLO. Nevertheless, YOLOv8 not only provides object detection, but also offers object segmentation, classification and pose estimation capabilities.

III. Result and Discussion

A. Experiment Setup

This project primarily utilizes Python as the programming language because it is relatively easy to use and has a robust development environment, making it an excellent choice for machine learning tasks. Next, the YOLO framework plays a crucial role in this project for identifying and categorizing various types of vehicles. Therefore, this project uses a git clone to obtain multiple versions of the YOLO release in GitHub, such as YOLOv5, YOLOv6, YOLOv7, and YOLOv8. Furthermore, Anaconda's virtual environment feature is utilized in this project to create separate environments with specific runtime conditions and dependencies required by different versions of YOLO. Since the platform used in this experiment is RTX 3090, CUDA 11.7 is installed to drive it to maximize performance and solve the problem of time-consuming CPU training.

Hyperparameter tuning can be recognized as a significant tool for deep learning to determine the optimal configuration of hyperparameters to improve the performance of neural network models. Additionally, hyperparameters, including learning rate, batch size, number of layers, activation functions, and regularization techniques, can be considered predefined parameter settings used to influence the behavior and architecture of a neural network. Different hyperparameter tuning techniques were relentlessly experimented with to experiment with different YOLO variants to optimize the model's performance. For all model training, batch sizes of 24 and 400 epochs were set for optimal resource utilization and reduced the training time. These parameter settings provide enough iterations for the model to learn complex patterns and converge to an optimal solution.

B. Evaluation Metrics

Mean Average Precision (mAP) is a widely recognized metric for evaluating the performance and accuracy of object detection models. This evaluation index will be the standard for judging the model's performance throughout this study. The mAp metric is calculated by computing the mean of average precision (AP) for each object category based on the specific threshold called IoU. IoU is a measure of overlap between predicted and ground-truth bounding boxes, which is used to observe the actual label and trained model label.

$$Interest\ of\ union\ (IoU) = \frac{Area\ of\ overlap}{Area\ of\ union} \quad (1)$$

Here, the area of overlap and area of the union in equation (1) respectively represent the intersection and combined region between ground truth bounding boxes and predicted bounding boxes. Each class of true positive (TP), false positive (FP), and false negative (FN) could be calculated based on the IoU and the confidence score of predicted bounding boxes. These 3 metrics can then be used to calculate the precision and recall of the model by following equations (2) and equations (3):

$$Percision = \frac{TP}{TP+FP} = \frac{TP}{Total\ positive\ results} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} = \frac{TP}{Total\ relevant\ samples} \quad (3)$$

AP measures the precision-recall balance of a model at different IoU thresholds to obtain an overall measure of the model's overall accuracy and robustness. AP can be calculated by summing the recall values at 11 equidistant levels [0, 0.1, ..., 0.9, 1.0] from the recall curve by equation (4):

$$AP = \frac{1}{11}\sum_{r=0}^{1} P_{\max(r), r\in\{0.0, 0.1, \ldots, 0.9, 1.0\}} \quad (4)$$

Then, mAP is calculated by dividing the AP values of each class by the total number of classes. Generally, higher mAP values indicate better accuracy and robustness of the model. The mAP metric is typically measured using mAP@0.5, which evaluates performance at an IoU threshold of 0.5, and mAP@0.5:0.95, which represents the average mAP over various IoU thresholds from 0.5 to 0.95.

C. Model Benchmark

In model benchmarking, the training phase follows the dataset allocation method and parameter settings described in the section above. Training iterations for different YOLO versions were conducted, and the results are recorded in Tables I and II. These tables show the training results before and after using data augmentation techniques, including the training loss, accuracy, and training time. These recorded results are a comprehensive reference for evaluating training progress and monitoring model performance. These tables help to compare the performance of different YOLO versions and assess the impact of data augmentation on training results. Table III records the pre-trained checkpoints used in this paper and the models' frames per second (FPS).

TABLE I
Model performance before applying data augmentation techniques

Model	Precision	Recall	mAP @0.5	mAP@0 .5:0.9	Training time (hours)
YOLOv5	0.756	0.803	0.818	0.649	2.839
YOLOv6	0.718	0.71	0.746	0.57	3.214
YOLOv7	0.809	0.765	0.811	0.651	3.722
YOLOv8	0.824	0.725	0.818	0.666	2.437

TABLE II
Model performance after applying data augmentation techniques

Model	Precision	Recall	mAP @0.5	mAP@0 .5:0.9	Training time (hours)
YOLOv5	0.968	0.952	0.977	0.91	11.44
YOLOv6	0.945	0.9	0.967	0.836	7.847
YOLOv7	0.941	0.927	0.961	0.859	9.866
YOLOv8	0.964	0.947	0.979	0.928	11.93

TABLE III
Model pretrain checkpoints and FPS

Model	Parameters (m)	FLOPS (b)	Weights (MB)	FPS
YOLOv5x	86.2	203.9	173.1	62.89

Model	Parameters (m)	FLOPS (b)	Weights (MB)	FPS
YOLOv6l	59.54	150.51	119.7	37.6
YOLOv7x	70.81	188.1	142.1	62.5
YOLOv8x	68.13	257.4	136.7	56.49

According to observations in Table I, all models can only achieve about 80% of mAP@0.5 before applying data augmentation. This means that the models can most accurately identify objects in images, but they struggle to achieve higher performance and falsely detect objects. Even so, the loss curve of the training process also shows that most models show signs of overfitting. This happens when the models perform well on the training data but poorly on the validation data. Interestingly, both YOLOv8 and YOLOv5 exhibit early stopping, suggesting that the training dataset may not have sufficiently diverse examples to improve the model's performance further. Hence, data augmentation techniques have been used in this study to solve the data imbalance, increase the dataset size, and overcome overfitting. From the observation of Table II, YOLOv5 achieves the highest precision and recall among the models, which are 96.8% and 95.2%, respectively, after applying data augmentation. These two-evaluation metrics show that YOLOv5 can detect and classify objects accurately with relatively few false positives and false negatives. In addition, YOLOv8 also shows its strong detection capabilities and good performance, with a precision rate of 96.4% and a recall rate of 94.7%. In this case, YOLOv6 and YOLOv7 will perform slightly worse than YOLOv5 and YOLOv8. Regarding mAP evaluation indicators, YOLOv8 achieved the highest mAP@0.5 and mAP@0.5:0.9 values reaching 97.9% and 92.8%, respectively. This obtained result demonstrates YOLOv8 superior performance in object detection over a wide range of confidence thresholds. YOLOv5 also performs well in terms of mAP with values of 97.7% at 0.5 IoU and 91% from 0.5 to 0.9 IoU. The YOLO variants vary in model weight size, with YOLOv5 having the largest size (173.1 MB), followed by YOLOv7 (142.1 MB), YOLOv8 (136.7 MB), and YOLOv6 (119.7 MB). Model weight size is an important consideration when deploying models in resource-constrained environments. However, a balance must be struck between reducing the model size and maintaining satisfactory performance.

YOLOv8 required the longest training time of 11.929 hours with the highest FLOPs (257.4 billion), followed by YOLOv7 with 9.866 hours of training time. YOLOv5 has a training time of 11.439 hours and the second-highest FLOPs (203.9 billion). YOLOv6 has the shortest training time of 7.847 hours and the lowest FLOPs (150.51 billion). The number of parameters has a similar trend, with YOLOv8 having the highest number (68.13 million), followed by YOLOv5 (86.21 million), YOLOv7 (70.81 million), and YOLOv6 (59.54 million). The inference test results for each trained model show that YOLOv5 had the highest Frames Per Second (FPS) with a high rate of 62.89, and YOLOv7 did pretty well with a high FPS of 62.5. This means that YOLOv5 and YOLOv7 are capable of processing images in the YOLO family. YOLOv8 had a slightly lower FPS of 56.49, while YOLOv6 had the lowest FPS of 37.6, which means it is not good at quickly processing images. The FPS calculation of this part is based on the recommendation of the author of YOLOv5, which is 1000/inference times.

In summary, the experimental results show different performance characteristics of different YOLO variants. Despite higher computational requirements and longer training time, YOLOv8 demonstrates its strong detection capabilities and achieves the highest mAP values among other models. Before the release of YOLOv8, YOLOv5 demonstrated its leadership in object detection with its high precision, recall, and mAP scores. It is worth noting that the tradeoff between model complexity, training time, and performance should be carefully considered when choosing a YOLO variant. YOLOv6 offers a viable option for faster training times and lower computational requirements, albeit with slightly lower performance than YOLOv5 and YOLOv8. However, the final choice of the YOLO variant should depend on the project's specific requirements, including the ideal balance between accuracy, training time, model size, and available computing resources.

Besides that, this study also observed and compared the curves on the train and loss graphs to investigate the impact of data augmentation on YOLO. This observation mainly focuses on the presence of overfitting and the subsequent performance improvement on the validation loss graph. Before applying data augmentation techniques, overfitting was observed in the validation phase among all YOLO versions; the accuracy also fluctuated up and down, and it didn't maintain a perfect continuous upward trend. These fluctuations in these metrics indicate inconsistencies in model predictions, exhibiting varying degrees of accuracy and completeness in detecting objects. These changes are particularly evident in obj_loss (confidence in object existence) and cls_loss (classification loss) across all YOLO variants. The training and validation curves for these different versions before applying data augmentation techniques are placed in Fig. 6-9. Since YOLOv6 does not provide training loss and validation loss during training, only mAP@0.5 and mAP@0.5:0.9 are displayed in the figure.

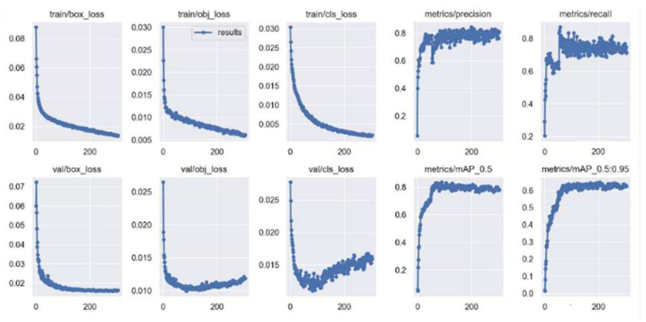


Fig. 6 Training and Loss Curves for YOLOv5 Before Applying Data Augmentation Techniques

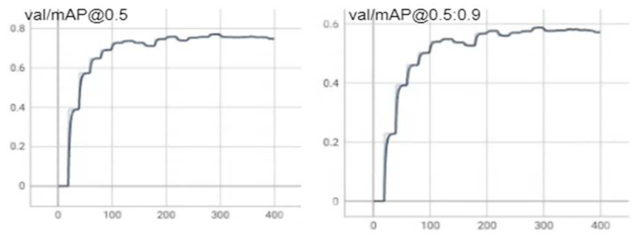


Fig. 7 Training and Loss Curves for YOLOv6 Before Applying Data Augmentation Techniques

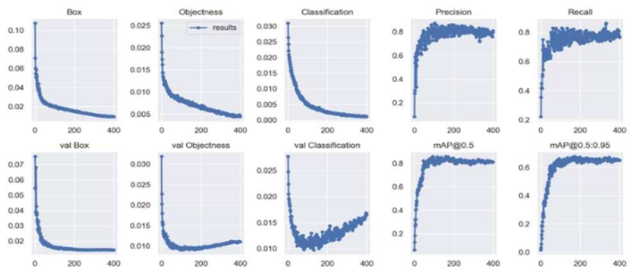


Fig. 8 Training and Loss Curves for YOLOv7 Before Applying Data Augmentation Techniques

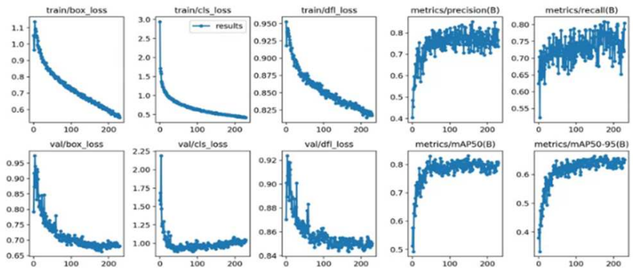


Fig. 9 Training and Loss Curves for YOLOv8 Before Applying Data Augmentation Techniques

In addition, the confusion matrix for different YOLO variants also reflects the percentage of misclassified classes. As can be seen from the confusion matrix of different YOLO variants in the figure below, the observation showed that only one or two categories have a high accuracy, and the other categories are kept below 80% before applying data augmentation techniques. Classes from C1 to C4 have the most misclassifications among the other classes, while C5 and C6 have the least misclassifications. The low error rate despite the small dataset may be due to the distinct and unique characteristics of the bus category represented by class C5 and the motorcycle category represented by class C6 compared with other classes. This makes it easier for the model to differentiate between these two classes even with a small and biased dataset. This study also performed validation and testing on the unseen dataset to ensure the model's generalization ability. The result showed that both C5 and C6 performed well. The common vehicles represented by C1 and C2 have some extremely similar shapes under the influence of the angle, so this may be a problem that makes the model difficult to distinguish. The worst model performance on object detection is YOLOv6 before applying data augmentation.

After applying data augmentation to the datasets, overfitting was significantly mitigated, and both training and validation loss consistently decreased, as shown in Fig. 10-13. The overfitting reduction and performance improvement observed on the resulting graph can be attributed to the introduction of data augmentation techniques. The trained model was exposed to a more diverse range of variations in the data by implementing horizontal flip, color jitter, and mix-up. This augmented dataset enables the model to learn more powerful and general object detection features, which results in decreasing validation loss. The trained model also tests and validates unseen data to ensure the model's performance. The result showed that the model performance after applying data augmentation has outperformed the trained model before applying data augmentation techniques. Nonetheless, the observation from the confusion matrix after applying the data augmentation techniques reflects a significant improvement with all YOLO variants achieving 86% and above on object detections. The misclassification of all YOLO variants has been decreased and the best result on object detection after applying data augmentation as shown in the confusion matrix in Fig. 14-17. In conclusion, this observation highlights the effectiveness of data augmentation in mitigating overfitting, improving the performance, and reducing the misclassified in the YOLO model. The reduction in validation loss demonstrates that the model improves generalization through augmentation techniques.

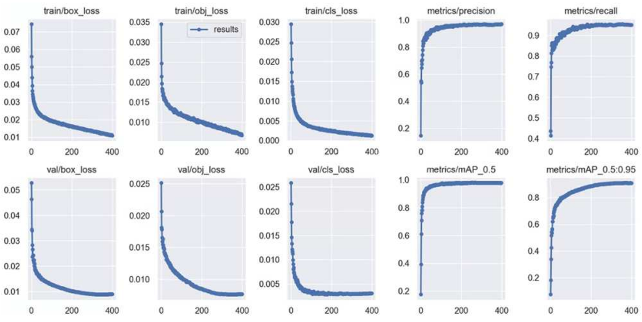


Fig. 10 Training and Loss Curves for YOLOv5 After Applying Data Augmentation Techniques

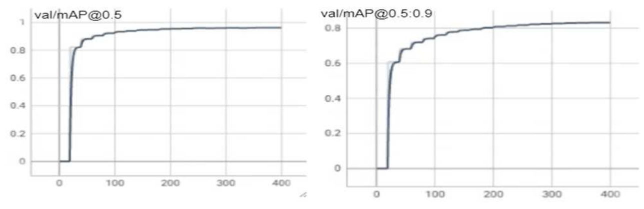


Fig. 11 Training and Loss Curves for YOLOv6 After Applying Data Augmentation Techniques

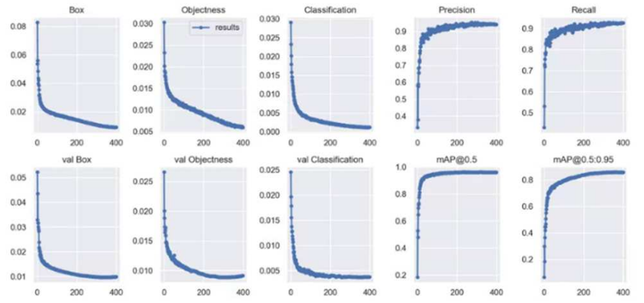


Fig. 12 Training and Loss Curves for YOLOv7 After Applying Data Augmentation Techniques

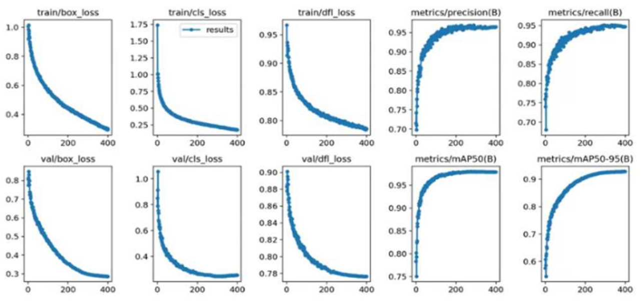
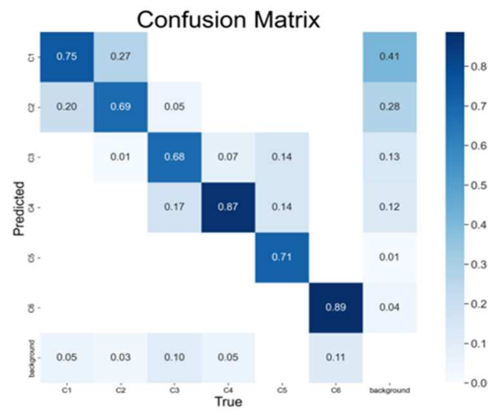
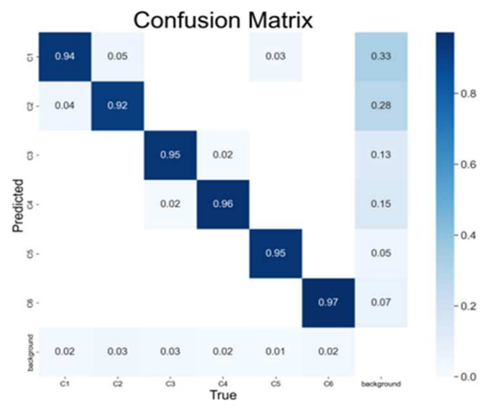


Fig. 13 Training and Loss Curves for YOLOv8 After Applying Data Augmentation Techniques

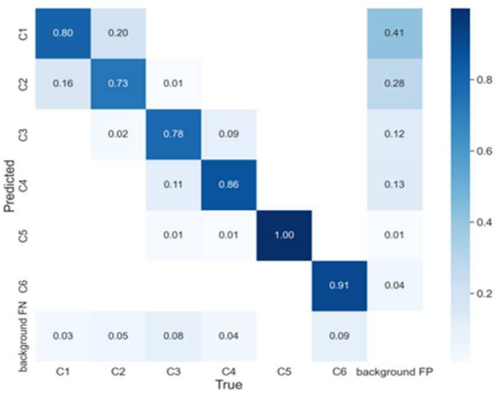


(a) Confusion Matrix for YOLOv5 Before Applying Data Augmentation Techniques

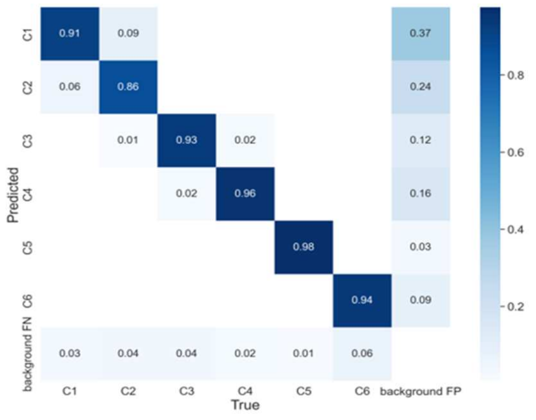


(b) Confusion Matrix for YOLOv5 After Applying Data Augmentation Techniques

Fig. 14 Confusion Matrix for YOLOv5

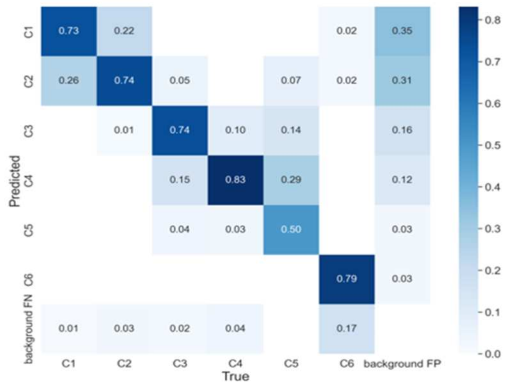


(a) Confusion Matrix for YOLOv7 Before Applying Data Augmentation Techniques

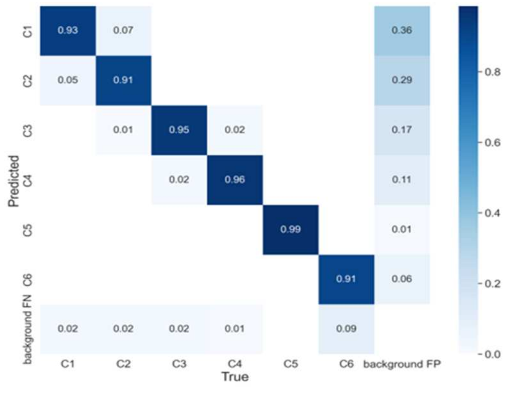


(b) Confusion Matrix for YOLOv7 After Applying Data Augmentation Techniques

Fig. 16 Confusion Matrix for YOLOv7

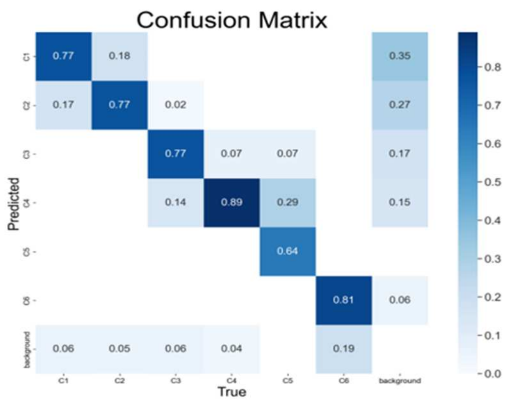


(a) Confusion Matrix for YOLOv6 Before Applying Data Augmentation Techniques

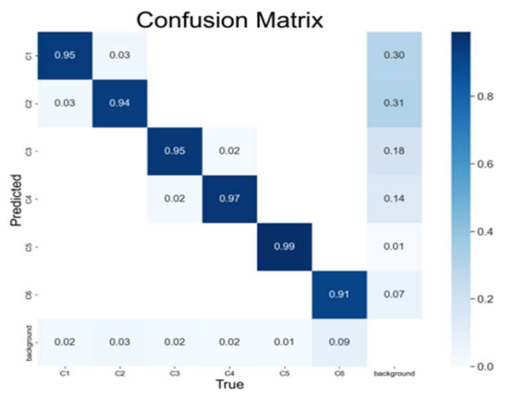


(b) Confusion Matrix for YOLOv6 After Applying Data Augmentation Techniques

Fig. 15 Confusion Matrix for YOLOv6



(a) Confusion Matrix for YOLOv8 Before Applying Data Augmentation Techniques



(b) Confusion Matrix for YOLOv8 After Applying Data Augmentation Techniques

Fig. 17 Confusion Matrix for YOLOv8

Fig. 18-21 below depicts the detection results of different versions of the YOLO model in terms of vehicle detection. After analyzing the results, the trained model can accurately identify and detect vehicles on the road. Moreover, there is a higher misclassification when detecting vehicle classes C1 and C2 while other vehicle types still perform satisfactorily. However, the model tends to perform at a higher misclassification rate when the vehicle is far away, especially in the latter half of the image. This limitation suggests that the model's ability to distinguish vehicle classes weakens when objects are far away. Still, it has proven effective in close-range scenarios, accurately identifying various vehicle types. Overall, the performance of these YOLO models for vehicle detection has both strengths and limitations. These observations emphasize the importance of considering the model's limitations and further optimizing it to improve its performance across all distances and vehicle classes.

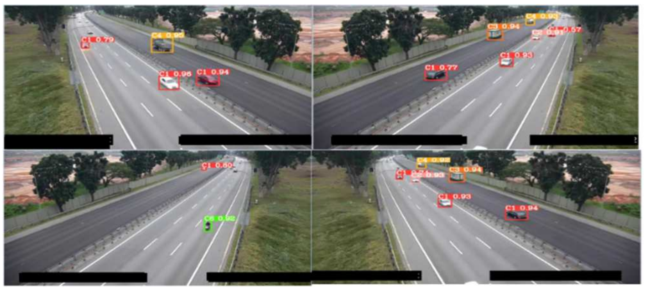


Fig. 18 YOLOv5 Detection Results

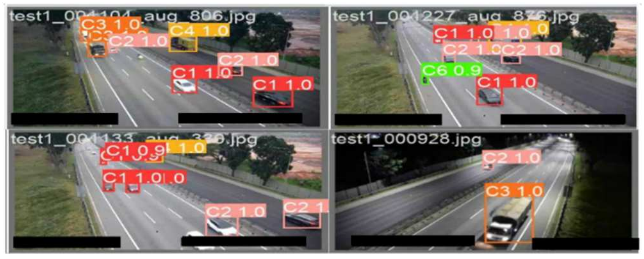


Fig. 19 YOLOv6 Detection Results

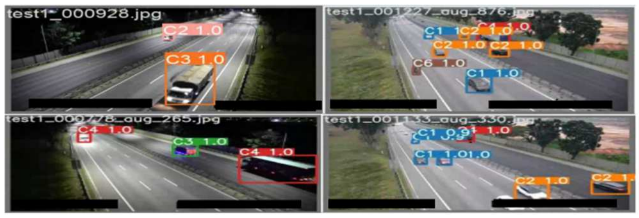


Fig. 20 YOLOv7 Detection Results

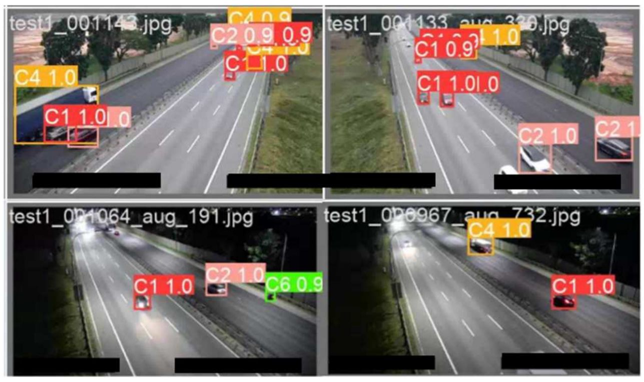


Fig. 21 YOLOv8 Detection Results

IV. Conclusion

This project mainly focuses on evaluating the performance of different variants of YOLO for vehicle detection. The proposed method addresses the challenges of data imbalance and data labeling. This project successfully addressed the data imbalance by employing data augmentation techniques and incorporating additional open-source data to fill in the imbalance categories. The experimental result has demonstrated that proper data augmentation can significantly improve the model's accuracy and enhance the depth of datasets. The hyperparameter tuning technique used also optimizes and improves the performance of the model to a large extent.

Undeniably, the performance might be improved when the larger datasets are combined with large models. However, considering the higher computational demands associated with such methods, it is necessary to investigate. Researchers can choose YOLO variants according to their specific experimental needs or industry application. YOLOv6 is an appropriate choice for those expecting to generate reasonably accurate models within limited resources and time constraints. In contrast, YOLOv5 and YOLOv8 are more suitable for scenarios that do not have limitations on computational resources and time constraints. These two models will perform better or result in accuracy, precision, and robustness. As a beneficial suggestion, YOLOv5, YOLOv7, or YOLOv8 are recommendable for the usage of real-time object detection. YOLOv8 performed well in object detection and surpassed other tested models with an accuracy of 97.9% mAP@0.5 and 92.8% mAP@0.5:0.9, but it required a relatively longer training time compared to other models.

Although this research has made encouraging results in vehicle detection, some problems remain to be solved, especially in accurately identifying fuzzy and tiny objects. The dataset used in this study primarily consists of top-down and far-angle views, which might contribute to the difficulty in identifying the vehicle types when the vehicles approach from a distance or appear small in the image. In the future, it would be beneficial to expand the dataset by incorporating more diverse data that includes occluded objects, varying angles, and different lighting conditions. This would enable the model to improve its accuracy and robustness in real-world scenarios. For example, local environmental factors such as rain or smog can be added to the dataset in the future to enhance the ability of the trained model to recognize objects in challenging scenes.

Furthermore, finding efficient strategies to handle the escalating computational demands becomes more crucial as datasets and models continue to grow and be complex. Exploring techniques such as distributed training or model compression would be valuable in optimizing the training process and reducing computational resources without sacrificing performance. Additionally, it would be helpful to explore transfer learning and domain adaptation techniques for creating more versatile and adaptable models. In this case, the performance of object detection systems can be enhanced in scenarios where the labeled data is limited, or the target domain differs from the training data by leveraging knowledge from related domains or pre-trained models. Lastly, exploring real-time optimization and deployment of object detection models on resource-constrained devices or in edge computing scenarios is another important avenue for future research. This would enable the deployment of efficient and accurate models in various applications, including autonomous vehicles, robotics, and surveillance systems.

Acknowledgment

This work was supported by the TM R&D Fund (Project no. RDTC/221054 and SAP ID: MMUE/220023) and MMU IR Fund (Project ID MMUI/220041).

References

[1] P. K. Bhaskar and S.-P. Yong, "Image processing based vehicle detection and tracking method," 2014 International Conference on Computer and Information Sciences (ICCOINS), Jun. 2014, doi:10.1109/iccoins.2014.6868357.

[2] Othman Omran Khalifa, Sheroz Khan, Md. Rafiqul Islam, and Ahmad Suleiman, "Malaysia Vehicle License Plate Recognition," *The International Arab Journal of Information Technology*, pp. 359–364, 2007, Accessed: Jan. 23, 2024. [Online]. Available: https://iajit.org/PDF/vol.4,no.4/10-Othman.pdf

[3] Cui, Wang, Wang, Liu, Yuan, and Wang, "Preceding Vehicle Detection Using Faster R-CNN Based on Speed Classification Random Anchor and Q-Square Penalty Coefficient," Electronics, vol. 8, no. 9, p. 1024, Sep. 2019, doi:10.3390/electronics8091024.

[4] Z. Rahman, A. M. Ami, and M. A. Ullah, "A Real-Time Wrong-Way Vehicle Detection Based on YOLO and Centroid Tracking," 2020 IEEE Region 10 Symposium (TENSYMP), 2020, doi:10.1109/tensymp50017.2020.9230463.

[5] J. J. Ng, K. O. M. Goh, and C. Tee, "Traffic Impact Assessment System using Yolov5 and ByteTrack," Journal of Informatics and Web Engineering, vol. 2, no. 2, pp. 168–188, Sep. 2023, doi:10.33093/jiwe.2023.2.2.13.

[6] D. Zhao, Y. Chen, and L. Lv, "Deep Reinforcement Learning With Visual Attention for Vehicle Classification," IEEE Transactions on Cognitive and Developmental Systems, vol. 9, no. 4, pp. 356–367, Dec. 2017, doi:10.1109/tcds.2016.2614675.

[7] R. Cheng, "A survey: Comparison between Convolutional Neural Network and YOLO in image identification," Journal of Physics: Conference Series, vol. 1453, no. 1, p. 012139, Jan. 2020, doi:10.1088/1742-6596/1453/1/012139.

[8] "Object Detection: YOLO vs Faster R-CNN," International Research Journal of Modernization in Engineering Technology and Science, Sep. 2022, doi:10.56726/irjmets30226.

[9] Viswanatha v., Chandana R K, and Ramachandra Ac, "Real Time Object Detection System with YOLO and CNN Models: A Review," *Journal of XI AN University of Architecture & Technology*, pp. 144–151, 2022, Accessed: Jan. 23, 2024. [Online]. Available: 10.37896/JXAT14.07/315415

[10] S. Srivastava, A. V. Divekar, C. Anilkumar, I. Naik, V. Kulkarni, and V. Pattabiraman, "Comparative analysis of deep learning image detection algorithms," Journal of Big Data, vol. 8, no. 1, May 2021, doi:10.1186/s40537-021-00434-w.

[11] R. M. Alamgir et al., "Performance Analysis of YOLO-based Architectures for Vehicle Detection from Traffic Images in Bangladesh," 2022 25th International Conference on Computer and Information Technology (ICCIT), Dec. 2022, doi:10.1109/iccit57492.2022.10055758.

[12] M. A. Bin Zuraimi and F. H. Kamaru Zaman, "Vehicle Detection and Tracking using YOLO and DeepSORT," 2021 IEEE 11th IEEE Symposium on Computer Applications &amp; Industrial Electronics (ISCAIE), Apr. 2021, doi:10.1109/iscaie51753.2021.9431784.

[13] B. Neupane, T. Horanont, and J. Aryal, "Real-Time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep Learning Network," Sensors, vol. 22, no. 10, p. 3813, May 2022, doi:10.3390/s22103813.

[14] G. S. A. Mohammed, N. Mat Diah, Z. Ibrahim, and N. Jamil, "Vehicle detection and classification using three variations of you only look once algorithm," International Journal of Reconfigurable and Embedded Systems (IJRES), vol. 12, no. 3, p. 442, Nov. 2023, doi:10.11591/ijres.v12.i3.pp442-452.

[15] K. Liu, H. Tang, S. He, Q. Yu, Y. Xiong, and N. Wang, "Performance Validation of Yolo Variants for Object Detection," Proceedings of the 2021 International Conference on Bioinformatics and Intelligent Computing, Jan. 2021, doi: 10.1145/3448748.3448786.

[16] N. Aloufi, A. Alnori, V. Thayananthan, and A. Basuhail, "Object Detection Performance Evaluation for Autonomous Vehicles in Sandy Weather Environments," Applied Sciences, vol. 13, no. 18, p. 10249, Sep. 2023, doi:10.3390/app131810249.

[17] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," Machine Learning and Knowledge Extraction, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, doi:10.3390/make5040083.

[18] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," Multimedia Tools and Applications, vol. 82, no. 6, pp. 9243–9275, Aug. 2022, doi:10.1007/s11042-022-13644-y.

[19] T. S. Gunawan, I. M. M. Ismail, M. Kartiwi, and N. Ismail, "Performance Comparison of Various YOLO Architectures on Object Detection of UAV Images," 2022 IEEE 8th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA), Sep. 2022, doi:10.1109/icsima55652.2022.9928938.

[20] I. S. Gillani et al., "Yolov5, Yolo-x, Yolo-r, Yolov7 Performance Comparison: A Survey," Artificial Intelligence and Fuzzy Logic System, Sep. 2022, doi:10.5121/csit.2022.121602.

[21] Q. M. Chung, T. D. Le, T. V. Dang, N. D. Vo, T. V. Nguyen, and K. Nguyen, "Data Augmentation Analysis in Vehicle Detection from Aerial Videos," 2020 RIVF International Conference on Computing and Communication Technologies (RIVF), Oct. 2020, doi:10.1109/rivf48685.2020.9140740.

[22] S.-Y. Lin and H.-Y. Li, "Integrated Circuit Board Object Detection and Image Augmentation Fusion Model Based on YOLO," Frontiers in Neurorobotics, vol. 15, Nov. 2021, doi:10.3389/fnbot.2021.762702.

[23] G. Dai, L. Hu, and J. Fan, "DA-ActNN-YOLOV5: Hybrid YOLO v5 Model with Data Augmentation and Activation of Compression Mechanism for Potato Disease Identification," Computational Intelligence and Neuroscience, vol. 2022, pp. 1–16, Sep. 2022, doi:10.1155/2022/6114061.

[24] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2016, doi:10.1109/cvpr.2016.91.

[25] A. H. Eichelberger, E. R. Teoh, and A. T. McCartt, "Vehicle choices for teenage drivers: A national survey of U.S. parents," Journal of Safety Research, vol. 55, pp. 1–5, Dec. 2015, doi:10.1016/j.jsr.2015.07.006.

[26] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding Data Augmentation for Classification: When to Warp?," 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), Nov. 2016, doi:10.1109/dicta.2016.7797091.

[27] Glenn Jocher *et al.*, "ultralytics/yolov5: Initial Release," *https://zenodo.org/records/3908560*, Jun. 2020, Accessed: Jan. 23, 2024, doi:10.5281/zenodo.4679653.

[28] C. Li *et al.*, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," Sep. 2022, Accessed: Jan. 23, 2024, doi:10.48550/arXiv.2209.02976

[29] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-yuan Mark Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Institute of Information Science, Academia Sinica, Taiwan*, Jul. 2022, pp. 1–11. Accessed: Jan. 23, 2024. doi:10.48550/arXiv.2207.02696.

[30] J. Glenn, C. Ayush, and Q. Jing, "YOLO by Ultralytics (Version 8.0.0). Computer software." Accessed: Jan. 23, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics.