REAL-TIME UAV-BASED VEHICLE DETECTION: A COMPARATIVE EVALUATION OF YOLO MODELS

Loi Nguyen

Faculty of Information Technology, University of Science, Ho Chi Minh City, Vietnam
Vietnam National University, Ho Chi Minh City, Vietnam
University of Information Technology, Ho Chi Minh City, Vietnam
Email: loinh@uit.edu.vn

Khanh-Duy Nguyen

University of Information Technology, Ho Chi Minh City, Vietnam
Vietnam National University, Ho Chi Minh City, Vietnam
Email: khanhnd@uit.edu.vn

Khang Nguyen

University of Information Technology, Ho Chi Minh City, Vietnam
Vietnam National University, Ho Chi Minh City, Vietnam
Email: khangnttm@uit.edu.vn

Abstract

With the rapid development of UAVs, vehicle detection from UAV-based aerial images has become important in intelligent traffic management and traffic safety monitoring systems. Recently, several studies have straight-forwardly adopted cutting-edge CNN-based object detection methods for this problem and demonstrated their effectiveness. High-accuracy methods, however, typically have a lot of parameters and a slow detection speed. The trade-off between accuracy and detection speed has not been well investigated. In this paper, we present a comparative evaluation of several real-time object detection models from the YOLO family, including YOLO-v5, YOLO-v6, YOLO-v7, and YOLO-v8, for aerial imagery. To compare the models' accuracy and complexity, experiments are conducted on the datasets VisDrone and Aeriau. The advantages and disadvantages of each model are also discussed.

***Keywords*: Vehicle Detection; YOLO; Aeriau Dataset; VisDrone2019.**

1. Introduction

Vehicle detection is a crucial component of an intelligent traffic management system. These systems conventionally use surveillance cameras for detecting vehicles. Because surveillance cameras have limited viewing angles, they cannot capture large traffic scenes. Compared with traditional surveillance cameras, unmanned aerial vehicles (UAVs) with moving cameras have several advantages, such as easy deployment, high mobility, and a large viewing range. To have a large viewing area, the UAVs need to fly at high altitudes, which pose many challenges for detecting small vehicles. Figure 1 demonstrates the vehicle detection problem based on UAVs.

Several cutting-edge CNN-based object detection methods can detect small objects, but they need to adopt a large backbone network to extract robust features. This makes the network inference time low and cannot work well for systems that require real-time processing. Recently, YOLOs have been proposed with high accuracy and real-time detection capability. Particularly, there are different YOLO models from v1 to v8, and they have some differences in both the backbone network and detection techniques. In this paper, we adopted different YOLO models for vehicle detection on UAV-based aerial images to analyze their accuracy and detection speed. Specifically, the contributions of our paper are twofold: firstly, we conduct a comprehensive evaluation of three YOLO models, including YOLO-v5 [1], YOLO-v6 [2], YOLO-v7 [3], and YOLO-v8 [4] for vehicle detection on two aeriau datasets, VisDrone [5] and Aeriau [6]; secondly, we show the trade-off between the evaluated models' accuracy, detection speed, and model complexity. The pros and cons of each model are also demonstrated by investigating the detection results.

2. Related works

2.1. *Object detection*

Object detection is one of the main tasks of computer vision. The goal is to classify and mark objects of interest with bounding boxes if they are present in the photo or video. This problem is widely applied across all fields and has many useful applications, such as security monitoring, people counting, traffic monitoring, pedestrian detection, self-driving cars, etc. Typical approaches to this problem include using machine learning techniques such as SVM [8] or deep CNN-based networks such as Faster-RCNN [9], and YOLO [10]. Figure 2 shows the architecture of a general CNN-based object detection method. The architecture consists of the following main parts: backbone, neck, dense prediction, and sparse prediction. Backbone is a CNN network that is utilized to extract features from input images. The architecture of this network significantly affects inference time because it carries the majority of the computational cost. Neck is used to combine low-level features and high-level semantic features, then build a pyramid feature map of all levels. The head part includes several convolutional layers to infer predictions based on the pyramid of features concatenated by the neck part. Single-stage object detection methods like YOLO [10] or RetinaNet [11] utilize dense prediction heads, while two-stage object detection methods like Faster R-CNN [9] use sparse prediction heads.

There are two main approaches for CNN-based object detection: one-stage models and two-stage models. In one-stage models [10], [11], the process of detecting and localizing objects takes place in a single step. This differs from the two-stage method, in which detection and localizing are performed in two separate steps. The advantage of these methods is that they are faster than two-stage methods and hence suitable for real-time applications. However, their ability to detect small and occluded objects may not be as accurate as two-stage methods.

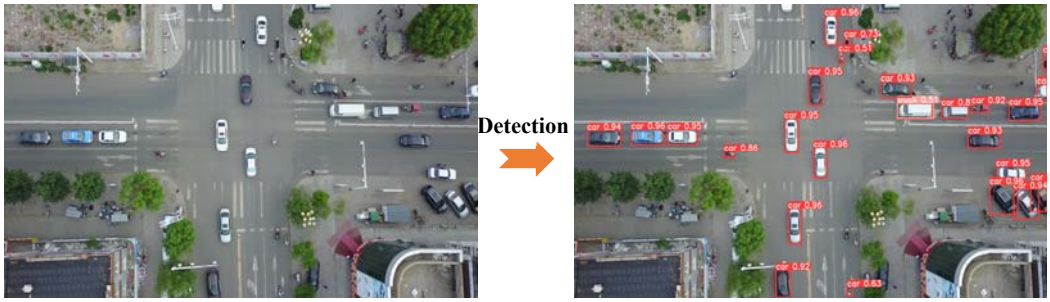


Fig. 1. Illustration of UAV-based vehicle detection.

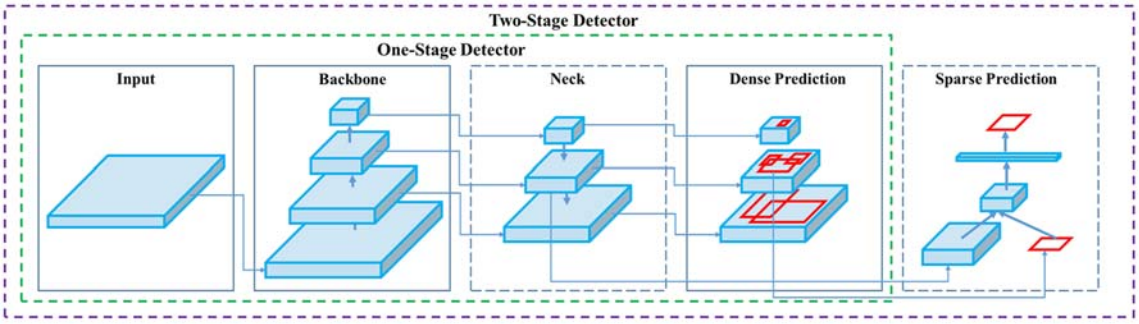


Fig. 2. Architecture of the object detection model [7].

2.2. *Vehicle detection from UAV-based aeriau images*

Several studies have recently used deep networks to tackle UAV-based vehicle detection challenges. Faster R-CNN has been successfully employed by Xu et al. [12] to detect cars from low altitude. Similar to this, Wang et al. [13] examined various backbone networks and developed an improved Faster R-CNN for detecting cars from UAV images. Additionally, Raza et al. [14] utilized RetinaNet, a well-known object detector, which boosted the accuracy of small vehicle detection. Other effective object detectors were also used in several studies, like TridentNet [15] or DetNet [16]. The accuracy of the best methods remains limited when detecting small objects at high altitudes. Only about 30% AP is achieved by the best algorithms on the VisDrone-DET 2018 dataset [17].

Several studies used fast object detection networks to address real-time drone-based vehicle detection. Tang et al. [18] examine the detection accuracy and speed of YOLO-v1 [10], and YOLO-v2 [19]. The YOLO-v2 model is the fastest and most accurate according to experiment results. Later, Faster R-CNN [9], YOLO-v3 [20], and YOLO-v4 [7] were compared by Ammar et al. [21]. Among the evaluated methods, YOLO-v4 had the fastest detection speed and the best accuracy. The YOLO family has recently significant upgrades, including YOLO-v5 [1], YOLO-v6 [2], YOLO-v7 [3], and YOLO-v8 [4]. Nevertheless, there aren't many papers that compare these

cutting-edge models for UAV images. To demonstrate the effectiveness of these models, we conducted thorough evaluations for this study.

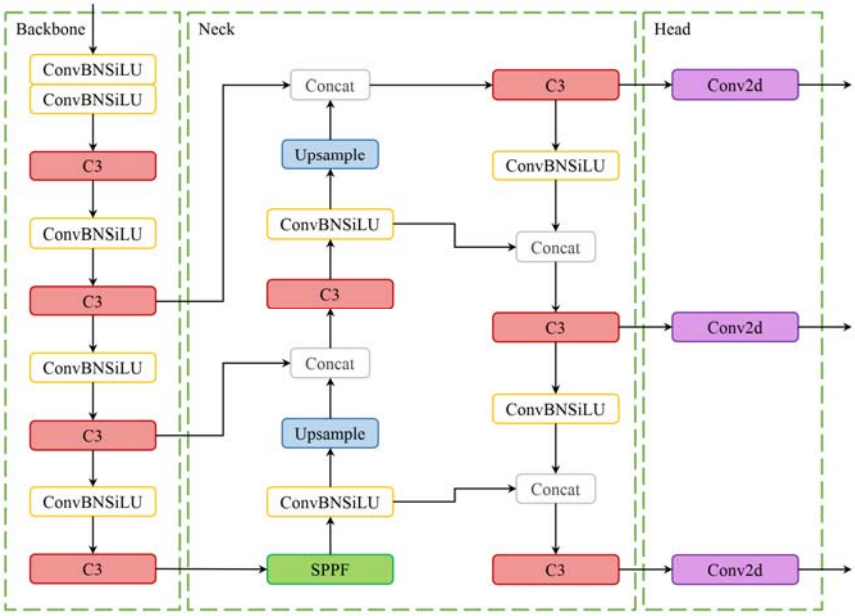
3. Network architectures of Yolo-v5, Yolo-v6, and Yolo-v7

YOLO-v5: YOLO-v5 [1] was presented in 2020 as an improvement of YOLO-v4 and YOLO-v3. Figure 3(a) demonstrates the YOLO-v5's network. Regarding the network architecture, the author replaced CSPResidualBlock in the backbone of YOLO-v4 [7] with a C3 module. The neck part consists of two parts: spatial pyramid pooling fast (SPPF) and PAN. The head part retains the same architecture as YOLO-v3. However, to determine the target coordinates for the bounding boxes, YOLO-v5 utilized a different equation from the previous version, as follows:

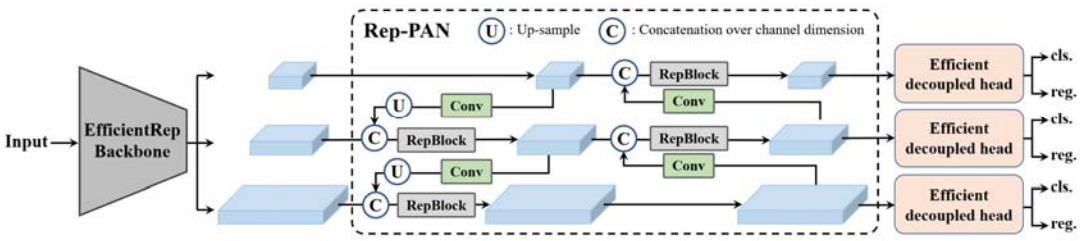
$$\begin{cases} b_x = (2 * \sigma(t_x) - 0.5) + c_x \\ b_y = (2 * \sigma(t_y) - 0.5) + c_y \\ b_w = t_w * (2 * \sigma(t_w))^2 \\ b_h = t_h * (2 * \sigma(t_h))^2 \end{cases} \quad (1)$$

where b_x, b_y, b_w, b_h are coordinates of bounding boxes and t_x, t_y, t_w, t_h are the the relative offsets compared with a particular anchor box.

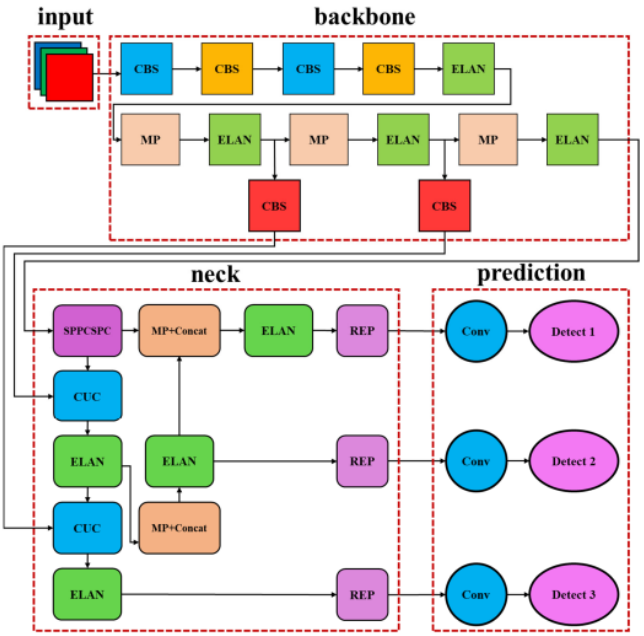
In addition, the authors propose a number of new data enhancement techniques, including fragment enhancement, copy-paste enhancement, random affine transformation, and mixing enhancement. The neck block's PAN has three outputs to detect objects at three different scales. At each scale, the effect on object loss is different, so a scaling factor is added for the object's loss to the loss function. YOLO-v5 [1] uses the k-means algorithm to group the actual bounding boxes into clusters and then uses the centroids of the clusters as the bounding box. This bounding box is called the dynamic bounding box. This allows bounding boxes to be aligned to better match the shape and size of detected objects. In the training stage, the weight update process will follow the exponential moving average formula.



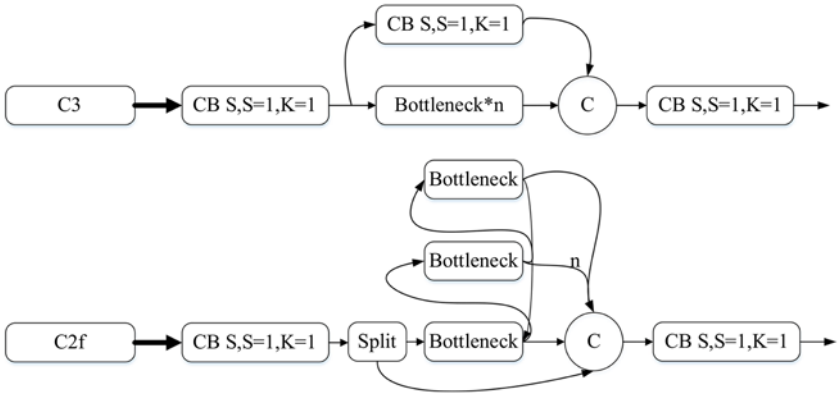
(a) The YOLO-v5's network architecture [22].



(b) The YOLO-v6's network architecture [2].



(c) The YOLO-v7's network architecture [23].



(d) C2f block module in YOLO-v8's network architecture [24].

Fig. 3. Network architectures of YOLO-v5, YOLO-v6, YOLO-v7 and YOLO-v8.

YOLO-v6: YOLO-v6 [2] was introduced in 2022 by Meituan. Figure 3(b) shows the network architecture of YOLO-v6. YOLO-v6 proposed an efficient backbone network based on RepVGG called EfficientRep, which uses parallel computing. The neck part is a PAN topology, also known as Rep-PAN, using PAN augmented with RepBlocks (small models) or CSPStackRepBlocks (large models). Additionally, it simplifies the decoupled head to make it more efficient, titled Efficient Decoupled Head. In YOLO-v6, the authors utilize a label assignment strategy based on the Task Alignment Learning method. In addition, YOLO-v6 also improves the quantization scheme by using RepOptimizer and channel-wise distillation to help reduce model size and improve detection in both speed and accuracy. The loss function in object detection includes classification loss, box regression loss, and optional object loss. The classification loss function uses VariFocal loss [25], while the regression loss uses SIoU/GioU [26]. A self-distillation strategy based on KL-divergence is also employed to help the network automatically adapt knowledge from teachers. The knowledge distillation loss can be computed as follows:

$$L_{KD} = KL\,(p_t^{cls} \,||\, p_s^{cls}) + \; KL\,(p_t^{reg} \,||\, p_s^{reg}), \tag{2}$$

where p_t^{cls} and p_s^{cls} are predicted class, and p_t^{reg} and p_s^{reg} are predicted bounding box of the teacher model and the student model respectively

YOLO-v7: YOLO-v7 [3] was proposed in 2022 by WongKinYiu and Alexey Bochkovskiy. The network architecture of YOLO-v7 is shown in Figure 3(c). The main contribution of this model is the optimization of the training process and network architecture. Specifically, the author designs several bag-of-freebies methods for training that include modules and optimization methods, which significantly improve detection accuracy but do

not increase the cost of inference. The network architecture uses the E-ELAN strategy, which enables a deep model to learn and converge more effectively by controlling the longest and shortest gradient paths. In order to enhance the learned features, the E-ELAN method combines the features of the groups together by shuffling and merging schemes. It also reduces the number of parameters and the calculation cost. In addition, YOLO-v7 introduces a number of other training bag-of-freebies including: (1) Utilizing Connection-Aware RepConv (RepConvN) to design the architecture of planned re-parameterized convolution; (2) Adopting a new labeling method that guides both auxiliary head and lead head; (3) Normalizing data in conv-bn-activation topology. (4) Combining both addition and multiplication in convolution on a feature map. (5) Applying the EMA model as the final inference model.

YOLO-v8: The latest, YOLO-v8, featuring several additional improvements, was introduced by Ultralytics in 2023. The C2F module carries the first significant part. Figure 3(d) illustrates the C2f module's architecture. Via the utilization of YOLO-v5's CSP backbone, YOLO-v8 replaced the C3 module with the C2f module. Next, for classification and regression tasks, YOLO-v8 subsequently adopts an anchorfree technique with task alignment learning. Furthermore, it eliminates the objectness branch from the network and uses the decoupled head for regression and classification. VFL loss and DFL loss are adopted for classification and regression tasks, respectively. By using an asymmetric weighting technique, VFL is able to resolve data imbalances between positive and negative samples. The formula is presented as follows:

$$VFL(y,\hat{y}) = \begin{cases} -\hat{y} * (\hat{y} * \log(y) + (1-\hat{y}) * \log(1-y)), \hat{y} > 0 \\ -\alpha * y^{\lambda} * \log(1-y), \hat{y} = 0 \end{cases} \quad (3)$$

where y is the label and yˆ is the predicted value. DFL is a form of cross entropy which optimizes the probability of the two positions that are closest to the label. In this way, the network can focus on the target's nearby area quickly. The formula is presented as follows:

$$DFL(S_i, S_{i+1}) = -((y_{i+1} - y)\log(S_i) + (y\text{-}y_i)\log(S_{i+1}) \quad (4)$$

4. Datasets

4.1. *Visdrone-2019 dataset*

VisDrone-2019 [5] is a well-known aerial imagery dataset created by the AISKYEYE team. The images are taken from 14 different cities in China, in different environments (urban and rural) with various congested levels. The ground-truths are provided for five computer vision tasks: object detection on images (VisDrone-DET-2019), object detection on video, single object surveillance, multi-object surveillance, and crowd counting. For our evaluation, we use the the VisDrone-DET2019 which consists of 8,629 images with 10 classes (pedestrian, people, bicycle, car, van, truck, tricycle, awning-tricycle, bus, motor) with 457,066 labeled objects. In particular, the training set has 6,471 images (343,205 labeled objects), the validation set has 548 images (38,759 labeled objects), and the test set has 1,610 images (75,102 labeled objects).

4.2. *Aeriau dataset*

The Aeriau dataset [6] was created in 2020 by Chung et al. This dataset is a combination of three aerial imagery datasets: VisDrone-2018, KIT AIS, and Aerial Open Source. The data set includes 1,658 images with 4 classes (car, truck, bus, motor) and 63,185 labeled objects. In particular, the training set contains 1,179 images (46,502 labeled objects), the validation set contains 295 images (11,581 labeled objects), and the test set contains 184 images (5,102 labeled objects).

5. Experiments and results

5.1. *Experimental settings*

The experiments were conducted on an Ubuntu 20.04.1 LTS operating system machine with 4 GeForce GTX 2080ti 12GB GPUs. All the models are trained with 300 epochs. We set the batch size to 8. Other hyper-parameters of the YOLO models are kept as originally.

5.2. *Evaluation metrics*

We adopted the Mean Average Precision (mAP) metric, which is employed in conventional object detection benchmarks, e.g MS-COCO [27], to evaluate the accuracy of all methods. This metric is calculated using the following formula:

$$mAP = \frac{1}{n}\sum_{i=1}^{n} APc_i \; ; \; APc_i = \frac{1}{m}\sum_{j=1}^{m} Pc_{ij} \quad (5)$$

where mAP is mean Average Precision of all classes, n is the number of classes, and m is the number of images. AP_{c_i} is the Average Precision (AP) of the class c_i . $P_{c_{ij}}$ is the precision for class c_i on the jth image, which is computed as: $P_{c_{ij}} = (TP_{c_{ij}})/(TP_{c_{ij}} + FP_{c_{ij}})$. $TP_{c_{ij}}$ denotes the true positive bounding boxes and $FP_{c_{ij}}$ denotes the false positive bounding boxes. It is necessary to select an IoU threshold to determine the true/false positive bounding boxes. We can use a single value, e.g 50 (abbreviated as mAP_{50}^{test}), or a range, e.g, from 50 to 95 in step 5 (abbreviated as $mAP_{(50:95)}^{test}$), then calculate the mAP for this range and get the mean value. We evaluated the models by both measures mAP_{50}^{test} and $mAP_{(50:95)}^{test}$.

5.3. *Results*

Table I shows the results of the evaluated models on the Aeriau dataset. We find that the YOLO-v6L model obtains the best results, scoring 67.1% and 47.9% in terms of AP_{50}^{test} and $AP_{(50:95)}^{test}$, respectively. Additionally, with mAPs of 87.5% and 23.3%, the bus and motorbike classes exhibit the best AP_{50}^{test} scores for this model. With mAPs of 94.1% and 75.2%, the YOLO-v7x model gave the best AP_{50}^{test} scores for the car and truck classes. The newest member of the YOLO family, YOLO-v8, does not perform as well as YOLO-v6 and YOLO-v7. Even though the YOLO-v6L model has the greatest mAP values on the Aeriau data set, its accuracy on the bus and motorbike object classes is lower than that of the YOLO-v7x model.

The detection results for each image in the test set of the Aeriau dataset are shown in Figure 4. Despite the fact that all three YOLO models correctly detected and located almost all of the automobiles, trucks, and buses, there is still uncertainty in the classification of these three object classes. The YOLO-v5x, YOLO-v6L, and YOLO-v8x models either failed to detect or incorrectly classified the motorbikes, whereas the YOLO-v7x model detected them more accurately.

According to the results on Table II, the YOLO-7x model outperforms the YOLO-v5x, YOLO-v6L, and YOLO-v8x models across all metrics on VisDrone dataset. Additionally, compared to large objects (cars, trucks, and buses), the detection accuracy for small objects (bikes, people) is lower. Note that the test set of the VisDrone dataset (1,610 images) is larger than the Aeriau dataset (184 images), and its images contain many complex situations taken from cameras in weather and lighting conditions. Additionally, this dataset has a lot of congested scenes, which causes objects to overlap, resulting in missed detections or incorrectly matched bounding boxes, as seen in Figure 5. The YOLO-v5x, YOLO-v6L, YOLO-v7x, and YOLO-v8x models are able to detect pedestrians, cars, trucks, and motorbikes. Nevertheless, drivers of motorbikes are considered as well as objects belonging to the people or pedestrian class, as shown in Figure 6. This yields the incorrect detection results.

Methods	Parameters (M)	FLOPs (G)	Detection speed (FPS)	AP_{50}^{test} classes (%)				mAP_{50}^{test}	$mAP_{50:95}^{test}$
				Car	Truck	Bus	Motor		
YOLO-v5x	86.19	203.8	33	90.7	60.6	86.8	23.0	65.3	47.3
YOLO-v6x	59.54	150.51	115	93.6	65.3	**87.5**	**23.3**	**67.4**	**48.0**
YOLO-v7x	70.80	188.0	74	**94.1**	**75.2**	65.0	16.5	62.7	43.5
YOLO-v8x	68.12	257.4	43	86.2	41.6	51.2	19.7	49.7	36.5

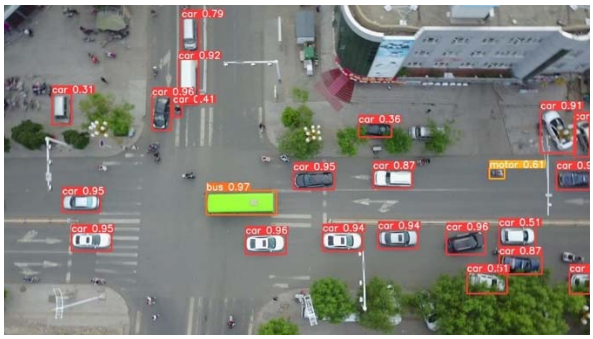
TABLE I. Experimental results on the test set of the Aeriau dataset.

Methods	AP_{50}^{test} classes (%)										mAP_{50}^{test}	$mAP_{50:95}^{test}$
	pedes.	people	bike	car	van	truck	tricycle	awn-tricycle	bus	motor		
YOLO-v5x	30.5	17.7	10.9	71	36.2	41.1	13.9	15.4	58.5	29	32.4	18.8
YOLO-v6x	29.1	14.1	13.1	73.8	40.8	48.4	21.1	21.0	62.8	32.3	35.7	21.0
YOLO-v7x	**40.7**	**28.6**	**17.8**	**79.5**	**42.5**	**54.7**	**28.7**	**25.5**	**63.8**	**43.1**	**42.5**	**24.0**
YOLO-v8x	34.2	18.1	14.8	74.9	41.5	50.0	20.7	23.1	63.3	37.6	37.8	22.6

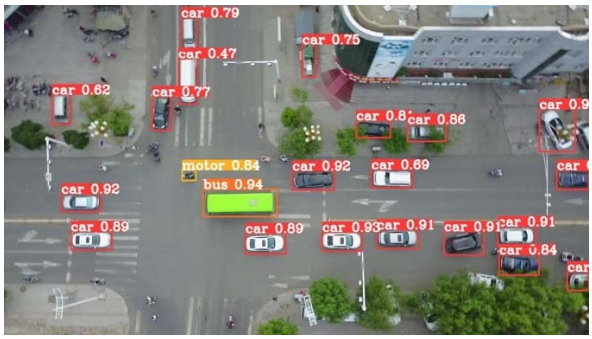
TABLE II: Experimental results on the test set of the VisDrone dataset.

Methods	Input image size	mAP_{50}^{test}	$mAP_{50:95}^{test}$
YOLO-v5x	640x640	32.4	18.8
YOLO-v6x		35.7	21.0
YOLO-v7x		42.5	24.0
YOLO-v8x		37.8	22.6
YOLO-v5x	1280x1280	37.9	22.4
YOLO-v6x		45.5	26.9
YOLO-v7x		47.8	27.2
YOLO-v8x		43.1	26.2

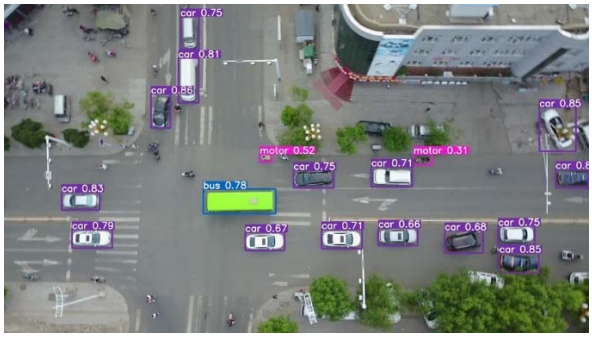
TABLE III: Accuracy comparison of YOLO models for varying input image sizes on VisDrone dataset.



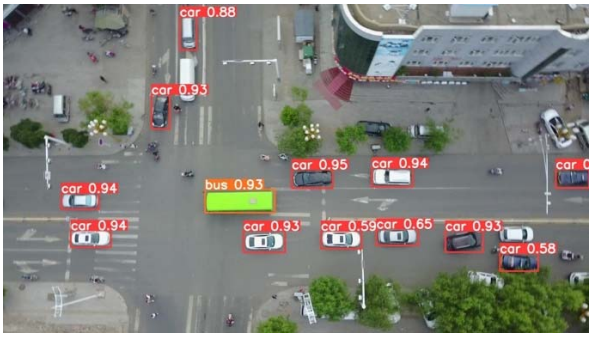
(a) The YOLO-v5x model detected a motorcycle but made lots of mistakes in classifying the truck as a car.



(b) The YOLO-v6L model detected a motorcycle but made lots of mistakes in classifying the truck as a car.



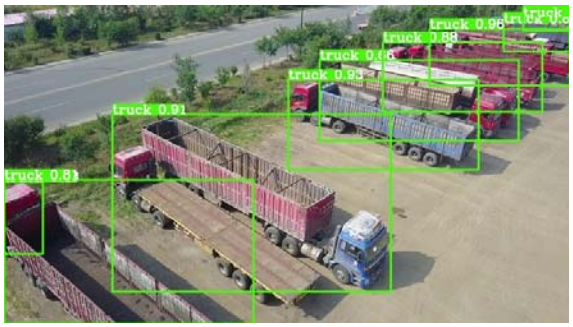
(c) Two motorcycles were found using the YOLO-v7x model, but many more remain undetected. It still struggles to distinguish a truck from a car.



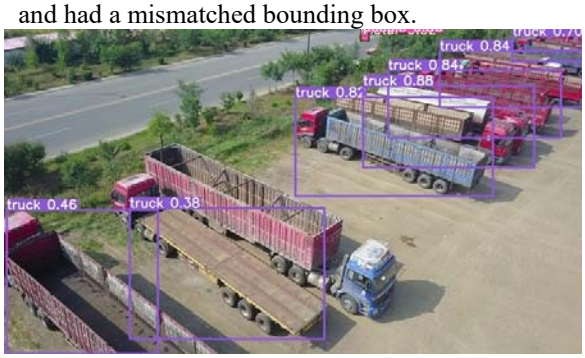
(d) YOLO-v8x model failed to detect most motorcycles and mis-detected four cars.

Fig. 4: Visualization of images of YOLO models on the Aeriau dataset [6].

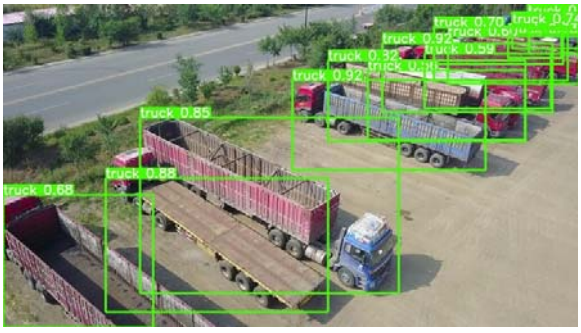
(a) The YOLO-v5x model mis-detected two trucks and had a mismatched bounding box.



(b) The YOLO-v6L model mis-detected a truck.



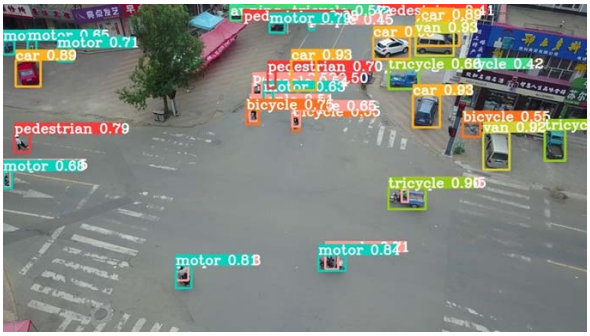
(c) The YOLO-v7x model mis-detected a truck and predicted two mismatched bounding boxes.



(d) Many mismatched bounding boxes are predicted by YOLO-v8.

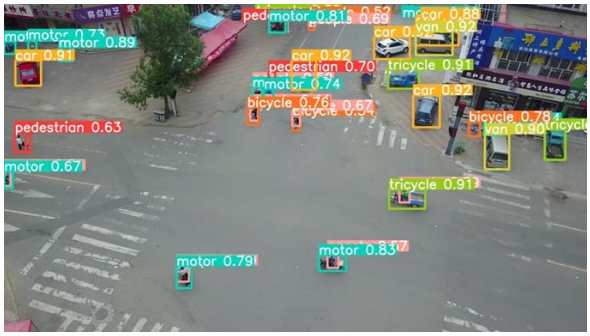
Fig. 5: On the VisDrone dataset, overlapping objects can lead to mistakenly predicted bounding boxes or being unrecognized by the YOLO models.

(a) Detection results of YOLO-v5x model.



(b) Detection results of YOLO-v6L model.

c) Detection results of YOLO-v7x model.



(d) Detection results of YOLO-v8x model.

Fig. 6: Visualization of images predicting whether the vehicle driver is a pedestrian or human when testing YOLO models on the VisDrone dataset [5].

We also calculated the number of parameters (in million), floating-point operations per second (in FLOPS), and detection time (in frames per second, or FPS) for each model in order to compare the accuracy and detection speed of YOLO models. Table I shows the results. We observe that the YOLO-v6L model, which has 59.54M parameters and can detect vehicles at a speed of 115 frames per second, is the lightest model. This is significantly faster than YOLO-v5x and YOLO-v7x, by five times and nearly two times, respectively. It should be noted that YOLO-v6L also achieves the highest accuracy on the Aeriau dataset and the second-highest accuracy on the VisDrone dataset. On the VisDrone dataset, the YOLO-v7x model has the highest accuracy, but because it requires a large number of parameters, its detection speed is substantially slower than the YOLO-v6L model's. Figure 7 visualizes the number of parameters, detection speed, and accuracy of four models on the VisDrone dataset.

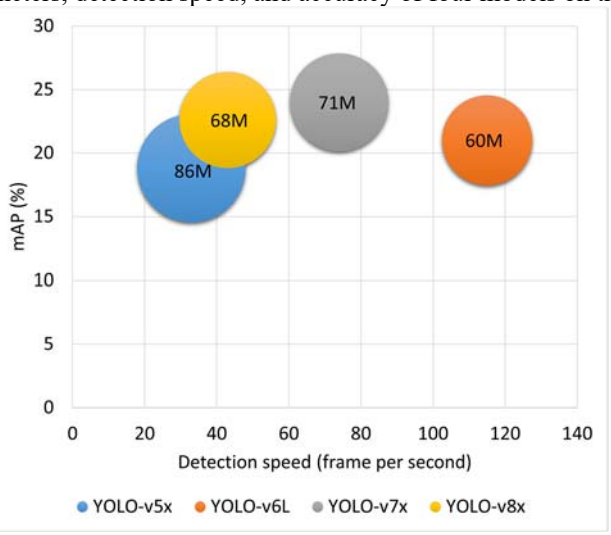


Fig. 7: Visualization of accuracy, detection speed, and the number of parameters for three YOLO models.

YOLO models detect and localize objects using an input image of a fixed size. This size is important, particularly when detecting small vehicles from drones. As a result, we further investigate in this work how different input sizes affect the effectiveness of all YOLO models. The results from Visdrone datasets with two input image sizes-640×640 pixels and 1280×1280 pixels-are presented in Table III. In terms of mAP_{50}^{test} , the results of all models on the larger input image size (1280x1280) are much better than those on the smaller input

image size (640x640), ranging from 5.3% (YOLO-v7x) to 9.8% (YOLO-v6L). Therefore, just using a large input image size is necessary for aerial imagery-based vehicle detection.

6. Conclusions

In this study, we compared the effectiveness of the four YOLO models, YOLO-v5, YOLO-v6, YOLO-v7, and YOLO-v8. Based on our evaluations using the VisDrone and Aeriau datasets, we prove that YOLO-v6 and YOLO-v7 had the highest accuracy. The model with the fewest parameters and smallest FLOPs, YOLO-v6, is the fastest. This should be the best choice for highly accurate real-time vehicle detection. The newest member of the YOLO family, YOLO-v8, does not perform as well on aerial photography as earlier versions, such as YOLO-v6 and YOLO-v7. The disadvantage of the evaluated models is that while they are good at detecting large objects like cars, trucks, and buses, their precision for small objects is still quite poor. The classification of several object classes, such as cars and trucks, is still incorrect. The accuracy of YOLO models for the remaining issues identified above will be our primary focus for future studies.

Acknowledgement

This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number B2023-26-01.

References

[1] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, Y. Kwon, K. Michael, J. Fang, Z. Yifu, C. Wong, D. Montes et al., "ultralytics/yolov5: v7. 0-yolov5 sota realtime instance segmentation," Zenodo, 2022.
[2] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie et al., "Yolov6: A single-stage object detection framework for industrial applications," arXiv preprint arXiv:2209.02976, 2022.
[3] C. Wang, A. Bochkovskiy, and H. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arxiv 2022," arXiv preprint arXiv:2207.02696, 2022.
[4] J.-H. Kim, N. Kim, and C. S. Won, "High-speed drone detection based on yolo-v8," in ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2023, pp. 1–2.
[5] D. Du, P. Zhu, L. Wen, X. Bian, H. Lin, Q. Hu, T. Peng, J. Zheng, X. Wang, Y. Zhang et al., "Visdrone-det2019: The vision meets drone object detection in image challenge results," in ICCV workshops, 2019.
[6] 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," in RIVF. IEEE, 2020, pp. 1–3.
[7] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.
[8] K.-D. Nguyen, D.-D. Le, and D. A. Duong, "Efficient traffic sign detection using bag of visual words and multi-scales sift," in ICONIP. Springer, 2013, pp. 433–441.
[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," Advances in neural information processing systems, vol. 28, 2015.
[10] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in CVPR, 2016, pp. 779–788.
[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for ´ dense object detection," in ICCV, 2017, pp. 2980–2988.
[12] Y. Xu, G. Yu, Y. Wang, X. Wu, Y. Ma et al., "Car detection from low-altitude uav imagery with the faster r-cnn," Journal of Advanced Transportation, vol. 2017, 2017.
[13] L. Wang, J. Liao, and C. Xu, "Vehicle detection based on drone images with the improved faster r-cnn," in 2019 11th International Conference on Machine Learning and Computing, 2019, pp. 466–471.
[14] M. A. Raza, H. Bint-e Naeem, A. Yasin, and M. H. Yousaf, "Birdview retina-net: Small-scale object detector for unmanned aerial vehicles," in 2021 16th International Conference on Emerging Technologies (ICET). IEEE, 2021, pp. 1–6.
[15] Y. Li, Y. Chen, N. Wang, and Z. Zhang, "Scale-aware trident networks for object detection," in ICCV, 2019, pp. 6054–6063.
[16] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, "Detnet: Design backbone for object detection," in ECCV, 2018, pp. 334–350.
[17] P. Zhu, L. Wen, D. Du, X. Bian, Q. Hu, and H. Ling, "Vision meets drones: Past, present and future," arXiv preprint arXiv:2001.06303, vol. 1, no. 2, p. 8, 2020.
[18] T. Tang, Z. Deng, S. Zhou, L. Lei, and H. Zou, "Fast vehicle detection in uav images," in 2017 International Workshop on Remote Sensing with Intelligent Processing (RSIP). IEEE, 2017, pp. 1–5.
[19] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in CVPR, 2017, pp. 7263–7271.
[20] J. Redmon and F. Ali, "Yolov3: An incremental improvement," arXiv preprint arXiv:1804.02767, 2018.
[21] A. Ammar, A. Koubaa, M. Ahmed, A. Saad, and B. Benjdira, "Aerial images processing for car detection using convolutional neural networks: Comparison between faster r-cnn and yolov3," arXiv preprint arXiv:1910.07234, 2019.
[22] M. Jani, J. Fayyad, Y. Al-Younes, and H. Najjaran, "Model compression methods for yolov5: A review," arXiv preprint arXiv:2307.11904, 2023.
[23] B. R. Chang, H.-F. Tsai, and C.-W. Hsieh, "Accelerating the response of self-driving control by using rapid object detection and steering angle prediction," Electronics, vol. 12, no. 10, p. 2161, 2023.
[24] B. Xiao, M. Nguyen, and W. Q. Yan, "Fruit ripeness identification using yolov8 model," Multimedia Tools and Applications, pp. 1–18, 2023.
[25] H. Zhang, Y. Wang, F. Dayoub, and N. Sunderhauf, "Varifocalnet: An iou-aware dense object detector," in CVPR, 2021, pp. 8514–8523.
[26] Z. Gevorgyan, "Siou loss: More powerful learning for bounding box regression," arXiv preprint arXiv:2205.12740, 2022.
[27] [T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft coco: Common objects in ´ context," in ECCV. Springer, 2014, pp. 740–755.

Conflict of Interest: The author has no conflicts of interest to declare.

Authors Profile

Loi Nguyen, received the B.Sc. from the University of Science, VNUHCM, in 2014, respectively. Currently, he is a Master Student with the Faculty of Information Technology, University of of Science - VNUHCM. Since 2023, he has been with the University of Information Technology-VNUHCM. His research interests include computer vision and deep learning.

Khanh-Duy Nguyen, received the Ph.D. degree from the University of Information Technology, VNU-HCM, Vietnam in 2021. He was a Project Postdoc Researcher with the Echizen Laboratory, National Institute of Informatics from 2022 to 2023. He is currently a Researcher with the University of Information Technology. His research interests include computer vision, multimedia analysis, and deep learning.

Khang Nguyen, received the B.S. and M.S. degrees in computer science and the Ph.D. degree from the University of Science, VNUHCM, Vietnam, in 1990, 1995, and 2012, respectively. Currently, he is the Vice-President of the University of Information Technology-VNUHCM. His research interests include artificial intelligence and computer vision.