Cihan University-Erbil Scientific Journal (CUESJ)

Research Article

Real-Time Vehicles Detection Using a Tello Drone with YOLOv5 Algorithm

Shalaw M. Abdallah

Department of Software Engineering, Koya University, Kurdistan Region, Iraq

ABSTRACT

Incorporating Unmanned Aerial Vehicles (UAVs) within Artificial Intelligence systems has given rise to an essential and academically significant approach in the domain of vehicle detection. This study introduces a real-time vehicle detection framework leveraging the you only look once (YOLO) algorithm for precise identification of vehicles, employing the camera on the DJI Tello drone. The research is underpinned by a rich dataset encompassing approximately 2000 images, meticulously annotated with the respective vehicle angles. The framework's innovation lies in its comprehensive training regimen, encompassing all angles of vehicles: Vehicle-front, vehicle-rear, vehicle-above, and vehicle-sides. This holistic approach aims to yield a model capable of accurately identifying and tracking vehicles from a multitude of viewing angles. The choice of the YOLO algorithm, further enhanced by Ultralytics HUB, ensures the robustness and accuracy required for the detection of moving objects. The model's capability to effectively track objects is a testament to the algorithm's efficacy. In assessing the framework's performance, we employed a comprehensive set of evaluation parameters, including mean average precision, precision, recall, and F1 score. This research not only underscores the practicality of UAVs in the field of artificial intelligence but also highlights the excellence achieved in real-time vehicle detection.

Keywords: Tello drone, you only look once v5, Roboflow, intersection over union, ultralytics

INTRODUCTION

The use of drones has become increasingly popular, especially in the military and transportation sectors. We can teach drones how to locate and recognize targets and gather the necessary information about them. We can make drones more advanced and smarter using artificial intelligence-trained models such as you only look once (YOLO) and tensorflow. The purpose of this article is to train a YOLO model that can recognize all angles of a vehicle for military and traffic use. By training drones to recognize all angles of a vehicle, we can enhance their surveillance capabilities and improve situational awareness in both military operations and traffic management. This advanced technology has the potential to revolutionize the way we gather information and make informed decisions in these sectors.

The YOLO framework has emerged as a popular open-source deep learning tool for various machine learning applications. Initially introduced in 2016, YOLO employs a single neural network to perform object identification, categorization, and localization. Over the years, YOLO has undergone several iterations, with the most recent versions, YOLOv5 and YOLOv6/v7, released in 2020 and 2022, respectively, and offering significant enhancements. It's important to note that versions 6 and 7 were independently developed by different teams and don't belong to the core YOLO series. As of now, YOLOv5 remains the most widely used among these iterations. YOLOv5 stands out for several key advantages over its predecessors. These include a reduced model size, increased processing speed, improved accuracy, and seamless integration with the popular PyTorch open-source machine learning framework. Furthermore, YOLOv5 introduces ten new types not found in earlier versions, expanding its versatility.[1] In the realm of data science projects, the YOLOv5 algorithm finds significant utility in categorizing freely available datasets accessible from Internet repositories like Roboflow.[2] Roboflow, a free tool, simplifies the annotation and labeling of images, addressing the complexities of managing large datasets. Beyond labeling, the platform streamlines the assignment of images to appropriate classes, project organization, and numerous other workflow tasks, providing comprehensive solutions for data scientists.

Corresponding Author:
Shalaw M. Abdallah, Department of Software Engineering, Koya University, Kurdistan Region, Iraq. E-mail: shalaw.mshir@gmail.com

Received: August 09, 2023
Accepted: November 21, 2023
Published: January 20, 2024

DOI: 10.24086/cuesj.v8n1y2024.pp1-7

At the core of YOLO's model is its employment of a single neural network to analyze an entire image. This network partitions the image into grid sections, estimating probabilities and bounding boxes for each section. These bounding boxes are weighted according to their expected probability, allowing the model to consider the global context when evaluating the entire image during testing.[3] In addition, YOLO delivers efficient predictions through a single pass, as opposed to repeated analyses required in other systems. It employs three methods for detecting which are as below:

1. Blocks of residual (making an S * S grid from the image)
2. Regression of bounding box (predict the class, center, width, and height of objects)
3. Intersection over union (IOU) - For checking the way that Bbox overlaps and achieves the greatest fit.

A fundamental element of YOLO is its use of residual blocks to construct an S * S grid from the input image. This grid allows the image to be divided into a set of cells, each responsible for predicting specific class probabilities and bounding boxes. The adoption of residual blocks enables YOLO to analyze the entire image in a single pass, making it well-suited for real-world object recognition applications.[4]

For object position prediction, YOLO employs bounding box regression. This involves estimating the height, width, center, and class of objects. During training, the model adjusts the bounding boxes based on the training data to improve predictions. This fine-tuning enhances the model's accuracy in predicting object positions in new images.

The next critical concept is IOU, also known as the Jaccard index. IoU is a prominent evaluation metric for object detection and is commonly used to measure the similarity between two shapes. It provides a normalized measure that focuses on the areas (or volumes) of the compared objects, taking into account their parameters, such as width, height, and placement of bounding boxes. IoU is scale-invariant, making it a valuable metric for various performance assessments in segmentation, object detection, and tracking.[5-7] As previously mentioned, each bounding box (Y = [pc, bx, by, bh, bw, c]) in the image comprises features such as the bounding box center (bx, by), class label (c), height (bh), and width (bw). A regression of a single bounding box is employed to predict the class, center, width, and height of objects Figure 1.

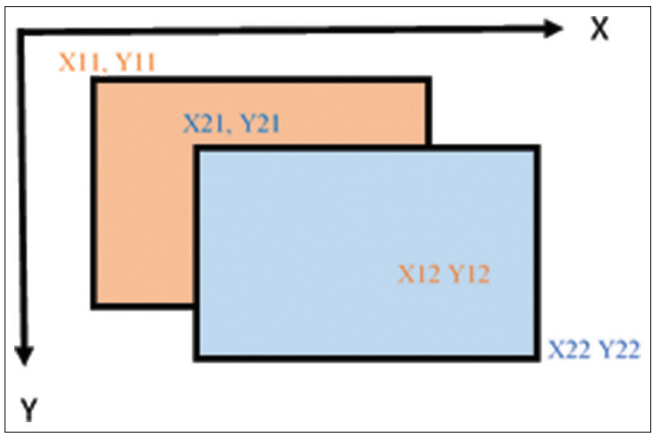

Figure 1: Example of applying Bbox to a grid class

IOU attends by below (1):

$$IoU = \frac{Inter\,sec\,tion\_Arear\,(IA)}{Uion\_Area\,(UA)} \quad (1)$$

$UA = (x22–x21) * (y22–y21) + (x21–x11) * (y12–y11) – IA$
$Inter\,sec\,tion\_height\ (Ih) = \min\,(y12, y22) – \max\,(y11, y21)$
$Inter\,sec\,tion\_width\ (Iw) = \min\,(x12, x22) – \max\,(x11, x21)$
$IA = Ih * Iw$

The calculation of IOU is represented by (1), where we determine the overlap area between the ground-truth bounding box and the predicted bounding box in the intersection area. The union area covers both the predicted and ground-truth bounding boxes. The resulting IOU score, when >0.5, signifies a successful object detection, while values below indicate detection failure.[8]

The Architecture of YOLOv5

The network architecture of YOLOv5 is presented in Figure 2. Choosing YOLOv5 as our initial learner has three causes.

First, YOLOv5, much like YOLOv4, introduces the cross-stage partial network (CSPNet) into the darknet framework, forming the foundation known as CSPDarknet. The incorporation of CSPNet is primarily designed to enhance the network's gradient combinations, all while demanding fewer computational resources. This is achieved by dividing the base layer's feature map into two distinct parts and skillfully integrating them using a suggested cross-stage hierarchy. Notably, CSPNet significantly reduces computational overhead, leading to improved speed and accuracy during inference.[9,10]

Second, YOLOv5 leverages the path aggregation network (panet) as its "neck" to enhance information flow. Panet adopts a novel feature pyramid network (fpn) topology with an augmented bottom-up pathway, facilitating the propagation of low-level features. It further employs adaptive feature pooling, effectively linking feature grids with various levels of features. This design choice enables vital information from every level of features to promptly propagate to the subsequent sub-network. Panet amplifies the utilization of precise localization cues in lower layers, thereby enhancing the accuracy of object localization.[11,12]

Third, the YOLOv5 "head," encompassing the YOLO layer, is responsible for producing multi-scale predictions. It generates three distinct feature map sizes, namely 72 × 72, 36 × 36, and 18 × 18, enabling the model to effectively handle objects of various scales, encompassing those that are large, medium, and small.[12]

Moreover, one of the most captivating features of YOLOv5 is its versatility in offering ten different types, with the most common ones being categorized into small, medium, large, and extra-large variations. These classifications are distinguished primarily by the number of feature extraction modules and convolution kernels they incorporate. This distinction carries substantial practical implications when working with YOLO models, as it directly impacts the model's capabilities and resource requirements.

Moreover, these variations in YOLOv5 also translate into variations in the quantity of neural network parameters.

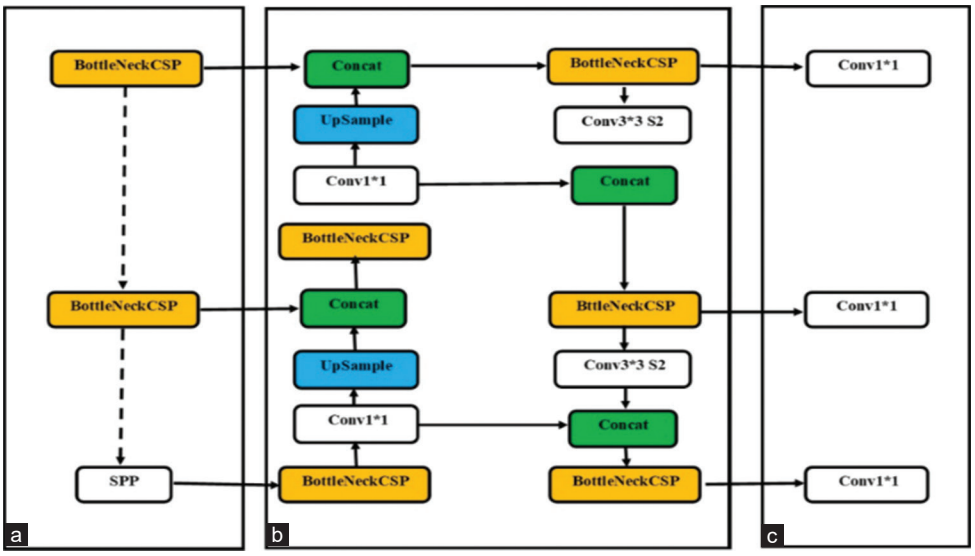

Figure 2: The network structure of you only looks once (YOLO). (a) CSPDarknet. (b) panet neck. (c) YOLO head

Notably, the size of these parameters varies across YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. The file sizes for these respective versions are approximately 14 mb, 41 mb, 89 mb, and can extend up to 166 mb. These differences in size and complexity provide users with a range of options to suit their specific project requirements and computational resources.[12,13]

MATERIALS AND METHODS

The methodologies and approaches used in this study to locate the top outcome in the selected dataset are presented in this section. The datasets are additionally described in segment 3.1. In segment 3.2, experimenting with the YOLOv5 on the dataset is given. In segment 3.3, the YOLO-trained model is fed to the Tello drone.

Dataset of Experimental

In the dataset, we have meticulously curated four distinct vehicle classes: "vehicle_above," "vehicle_rear," "vehicle_front," and "vehicle_sides." Our source of data is the official Roboflow Universe website, ensuring data reliability. Comprising a total of 2000 images, each class is thoughtfully represented with an equal share of 500 images, contributing to a well-balanced dataset. All images have been thoughtfully curated and collected from the Roboflow Universe.[14] To enhance the dataset quality and uniformity, we've harnessed the power of Roboflow's preprocessing capabilities. This crucial step guarantees consistency across the dataset, especially in image dimensions, making it ideal for model training, testing, and validation. The class annotations are meticulously represented as bounding boxes, a common practice in object detection tasks. These bounding boxes have been skillfully converted to the YOLOv5 format, all within the Roboflow Universe. Furthermore, each image in the dataset has been enriched with bounding box annotations using Roboflow Annotate. These annotations act as the "answer key" for our computer vision model during training, enabling it to learn and understand each class with precision. As we continue to add more annotated images to the dataset, our model gains access to richer and more diverse data, further improving its performance. Figure 3 provides a visual example, shedding light on the dataset's composition and structure.

Experimenting the YOLOv5 on the Dataset

In this experimental setup, each YOLOv5s model architecture was diligently trained using the T4 GPU accelerator, a resource generously provided by the Google Colab platform, readily accessible online. For those interested in replicating this experiment, the entire process is made accessible and straightforward. You can access the notebook code for YOLOv5 with a free GPU on this link: Google Colab YOLOv5 Notebook.

Here are the key steps to train your custom model:

1. Cloning YOLOv5: Begin by cloning the YOLOv5 repository from Ultralytics on GitHub.
2. Cloning Requirements: Clone all the necessary dependencies and libraries, such as OpenCV-Python and Matplotlib.
3. Dataset Upload: Upload your dataset using the code provided by Roboflow. This code facilitates the seamless retrieval and organization of the dataset from the Roboflow account where it was created.
4. Model Training: With the dataset in place, your model is now poised for training. Specify the YOLOv5 variant and set the number of epochs for training. In this particular experiment, 100 epochs were employed, encompassing both small and extra-large types of YOLOv5.

Table 1 records the time taken for each model to complete training after 100 epochs:

The results suggest that the YOLOv5s model achieved the quickest training time, taking less than an hour, while YOLOv5x training proved to be the most time-intensive, spanning approximately 9 h. After the successful completion of training, the trained model is saved in the.pt format, ready to be loaded and utilized with PyTorch for further applications.

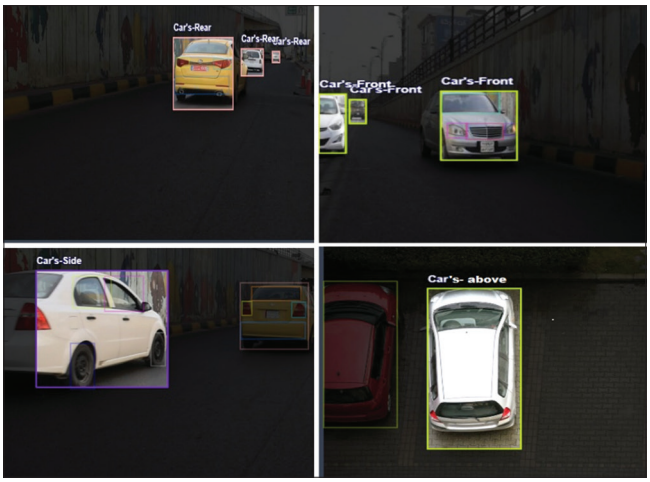

Figure 3: The dataset example of vehicle detection

Tello Drone Feeding

The initial step in connecting with the Tello drone involves the integration of another library called "djitellopy." This library serves as the gateway to interfacing with the drone's sensors and facilitates control over its movements.[15] The following code snippet demonstrates how to establish this connection:

```
import djitellopy as tello
kp.init()
me = tello.Tello()
me.connect()
print(me.get_battery())
```

With this connection in place, the drone is linked exclusively to the host device, which, in most cases, is a laptop, through a Wi-Fi connection. Simultaneously, to effectively load and run the YOLO models on any device, we need the PyTorch framework.[15] PyTorch necessitates access to the Internet too; therefore, there will be confusion between loading models and feeding models to the drone. Resolving this compatibility challenge involves a specific sequence of actions. The model must be loaded while the laptop is still connected to the Internet. Once the model is operational, the Internet connection can be interrupted, and the laptop is subsequently connected to the drone. An intriguing aspect to note is that the model remains cached in memory, serving as temporary storage and enabling us to retrieve the model without requiring Internet access. This cached model can be effortlessly loaded using the PyTorch Hub. This code below is used to load the model by torch (torch: Model = torch. hub.load ("ultralytics/YOLOv5," "custom," path_or_model= "model.pt")

With the model successfully loaded, the next step involves utilizing specific commands to control the drone's activities.

The following commands in Table 2 could be employed to commanding the drone and controlling the movements.

Note**:** In this step, PyCharm serves as the software of choice for model testing. Utilizing PyCharm, we can effectively evaluate the model's performance. To initiate the testing process, the YOLOv5 model must be obtained by cloning or downloading it from the repository accessible through this link: YOLOv5_Drone Repository. Once obtained, the model files can be uploaded to PyCharm. Particularly, the.pt file, derived from the final stage of model training, is paramount in this section as it is instrumental in conducting comprehensive testing and assessment.

Table 1: The training times for each YOLOv5 model after 100 epochs

Symbol	Model	Time (h)
V5x	YOLOv5x	8.6
V5s	YOLOv5s	0.73
V5	YOLOv5l	2.1
V5m	YOLOv5m	1.4

Table 2: Commands used to controlling the drone

Effect	Command
Take off	e
Land	q
Move down	s
Move up	w
Move backward	down arrow
Move forward	up arrow
Move right	right arrow
Move left	left arrow

RESULTS AND DISCUSSION

Our YOLOv5 model embarked on a rigorous training regimen involving 100 epochs, fed by a dataset comprising 2000 images. This arduous training journey extended over approximately 12 h, with YOLOv5x, our heavyweight contender, claiming the most substantial share of that time. The colossal YOLOv5x model boasts a formidable 166 MB of parameters. In our pursuit of exceptional precision, speed was a secondary consideration, ensuring that our primary focus remains on achieving the highest levels of accuracy. For testing purposes, we exclusively scrutinized the YOLOv5x model, aligning with our precision-driven objectives. However, for a comprehensive comparative analysis, we present the results for all four model variants across the dataset. The performance evaluation of our YOLO model extends to an array of critical metrics, including accuracy, mean average precision (mAP), average precision (AP), F1 score, recall, and precision.[16]

Precision, one of the fundamental metrics, is derived by dividing the total count of samples classified as positive (regardless of correctness) by the count of positive samples correctly identified. Precision serves as a valuable gauge of the model's proficiency in accurately classifying samples as positive.[17] The below (2) define the precision:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

TP is assigned to true positive, when the model accurately predicts the positive class, the result is a true positive. FP

assigned to false positive, when the model forecasts the positive class inaccurately, the result is called a false positive.[17]

The second metric is recalled, recall is determined by dividing the total number of positive samples by the number of positive samples that were accurately categorized as positive. The recall gauges how well the model can identify positive samples. Positive samples are found in greater numbers the higher the recall. To defining recall (3) is used:

$$\mathrm{Re}call = \frac{TP}{TP + FN} \tag{3}$$

Here FN false negative, when the model forecasts the negative class inaccurately, the result is known as a false negative.

The number of F1 score is from 0 to 1; the greater the value, the better the accuracy of identifying an item. The F1 score indicates how well the model balances its capacity to capture positive cases (recall) and be correct with the cases it captures (precision). The F1 score is calculated as the harmonic mean of precision and recall. It is a widely used statistic for classification models since it assesses the model's capacity for both recall and precision and yields reliable findings even for unbalanced datasets.[18] For this reason, F1 can assess a model's recall and precision because of the following derivation (4) method:

$$F1\_Score = \frac{2 * (precision * \mathrm{Re}call}{(\mathrm{Pr}ecision + \mathrm{Re}call)} \tag{4}$$

The final metric is a mAP, computer vision model performance is measured using mAP. The mean of the average precision metric over all classes in a model is equal to mAP. mAP can be used to compare different iterations of the same model as well as various models on the same job. The range of mAP measurement is 0 to 1. Before showing the equation of mAP, there are specific metrics that affect the calculating in mAP. The metrics involve IoU, recall, precision, and confusion matrix.[19] In the previous we mention IoU, recall, and precision. Let us go into the confusion matrix, basicly this elements are explained in Table 3. The following four elements are essential for the confusion matrix:[20]

True positives (Tp): The label was correctly predicted by the model and it matched the ground truth. The appropriate object type and location have been identified by the model.

True negatives (Tn): Neither does the model predict the label nor is it a part of the ground truth. The incorrect class was accurately predicted by the model.

False positives (Fp): The model predicted a label, but it is not included in the ground truth. The model has either given the incorrect label or recognized an object that is not there.

False negatives (Fn): Although a label is a part of the ground truth, it is not predicted by the model. The model hasn't picked up on the object.

Table 3 shows the confusion matrix's states and labels:

Now the calculation of the mAP will be like (5):

$$mAP = \frac{1}{C}\sum_{i=1}^{C} APi \tag{5}$$

Here the C refers to the number of the class used in the model, and the APi is the average precision of the certain class. AP computes the precision at each threshold as a weighted average, where the weight represents the increase in recall from the previous threshold.[19] The AP will be calculated like (6):

$$Ap = \sum \left(Rn - R(n+1)\right) * Pn \tag{6}$$

Now that we have a comprehensive understanding of how the model calculates its metrics, let's delve into the results for the various YOLOv5 types. Table 4 shows the result of YOLOv5s or the small type. V5s get a good result; 100 epochs are used here with 416 * 416 image sizes. Note: The epochs and the image size are standard but they can be changed according to the model.

Table 5 indicates the final training of v5m, which scores the result that we expected from the training. The model needs around 1 h and 40 min to finalize the training. As a model v5s, here we use a 416 * 416 image size and 100 epochs.

Table 6 represents the outcome of v5l or the large. Here we got an interesting result, but we still need a more

Table 3: Confusion matrix elements

Matrix name	State	Label	Label
	Positive	TP (true positive)	FN (false negative)
Confusion matrix	Negative	FP (false positive)	TN (true negative)

Table 4: Indicator of YOLOv5s

Model	Class	Precision	Recal	F1 Score	mAP (0.5)
	Vehicle_above	0.57	0.70	0.60	0.59
YOLOv5s	Vehicle_rear	0.84	0.55	0.66	0.71
	Vehicle_front	0.88	0.70	0.78	0.75
	Vehicle_sides	0.87	0.57	0.68	0.60

Table 5: Indicator of YOLOv5m

Model	Class	Precision	Recal	F1 score	mAP (0.5)
	vehicle_above	0.91	0.70	0.85	0.88
YOLOv5m	vehicle_rear	0.66	0.55	0.76	0.87
	vehicle_front	0.76	0.70	0.84	0.80
	vehicle_sides	0.72	0.57	0.70	0.70

Table 6: Indicator of YOLOv5l

Model	Class	Precision	Recal	F1 score	mAP (0.5)
	vehicle_above	0.90	0.80	0.85	0.88
YOLOv5l	vehicle_rear	0.94	0.87	0.90	0.91
	vehicle_front	0.94	0.88	0.91	0.92
	vehicle_sides	0.93	0.88	0.90	0.93

Table 7: Indicator of YOLOv5x

Model	Class	Precision	Recal	F1 Score	mAP (0.5)
YOLOv5x	Vehicle_above	0.97	0.88	0.94	0.93
	Vehicle_rear	0.94	0.94	0.92	0.93
	Vehicle_front	0.96	0.96	0.96	0.96
	Vehicle_sides	0.94	0.95	0.94	0.95

accurate outcome. The time spent on training was 2 h and 10 min. The image size and epochs were the same as in previous models.

Table 7 appears for the model of v5x or the xlarg. The train achieves what we need here in this article, even though it takes us nearly 8 h to finalize the train. We chose v5x as the final model to be used with the drone for detecting vehicles. As a previous type, we used a 416 * 416 image size and 100 epochs.

CONCLUSION

To summarize what has been discussed in this article, we present a system that can detect vehicles from multiple sides. The sides are classified in the dataset that was collected from the Roboflow Universe. First, we classified images into four sections (vehicle_above, vehicle_rear, vehicle_front, and vehicle_sides), then annotated all images. Next, we performed the pre-processing and saved our dataset in the format of our model, which is YOLOv5 through Roboflow, which plays the main role in this study.

YOLOv5 is one of the versions of YOLO that you only look at. V5 has three components: first the CSP-Darknet53 backbone, second the neck PANet model, and third the head. Each component has a specific process. V5 has four main types (v5s, v5m, v5l, and v5x), each of which has different parameters. In this article, we train all of them and show the results of the training. The eye camera was a DJI Tello Drone that can be connected with YOLO models using the Pytorch framework. Tello can detect and classify the vehicles correctly based on the YOLO model. In the future, the Tello drone will be integrated with a TensorFlow model and a comparative analysis will be conducted to evaluate its performance against YOLOv5. The objective is to determine which model is better suited for drone applications. In addition, there are plans to extend the testing of these models to other drones for a more comprehensive assessment.

REFERENCES

1. J. Lu, C. Ma, L. Li, X. Xing, Y. Zhang, Z. Wang and J. Xu. A vehicle detection method for aerial image based on YOLO. *Journal of Computer and Communications*, vol. 6, pp. 98-107, 2018.
2. J. Nelson and J. Solawetz. *Responding to the Controversy about YOLOv5*. Roboflow Blog, 2020. Available from: https://blog.roboflow.com/yolov4-versus-yolov5 [Last accessed on 2023 Jan 10].
3. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu and A. C. Berg. SSD: Single Shot Multibox Detector. In*: Computer Vision-ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11-14, 2016, Proceedings, Part I 14*. Springer International Publishing, Berlin, pp. 21-37, 2016. Available from: https://arxiv.org/pdf/1512.02325.pdf%22source%22 [Last accessed on 2023 Jan 02].
4. J. Redmon, S. Divvala, R. Girshick and A. Farhadi. You Only Look Once: Unified, Real-time Object Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 779-788, 2016. Available from: https://arxiv.org/abs/1506.02640 [Last accessed on 2023 Jan 07].
5. J. Yu, J. Xu, Y. Chen, W. Li, Q. Wang, B. I. Yoo and J. J. Han. Learning Generalized Intersection Over Union for Dense Pixelwise Prediction. In: *International Conference on Machine Learning*, pp. 12198-12207, 2021. Available from: https://shorturl.at/jBGIN [Last accessed on 2023 Jan 14].
6. H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern recognition*, pp. 658-666, 2019. Available from: https://arxiv.org/abs/1902.09630 [Last accessed on 2023 Jan 14].
7. Z. Huang, H. Zhao, J. Zhan and H. Li. A multivariate intersection over union of SiamRPN network for visual tracking. *The Visual Computer*, Vol. 38, pp. 2739-2750, 2022.
8. H. K. Jung and G. S. Choi. Improved yolov5: Efficient object detection using drone images under various conditions. *Applied Sciences*, Vol. 12, no. 14, p. 7255, 2022.
9. I. Katsamenis, E. E. Karolou, A. Davradou, E. Protopapadakis, A. Doulamis, N. Doulamis and D. Kalogeras. TraCon: A novel dataset for real-time traffic cones detection using deep learning. In: *Novel and Intelligent Digital Systems Conferences*. Springer International Publishing, Cham, pp. 382-391, 2022. Available from: https://arxiv.org/pdf/2205.11830.pdf [Last accessed on 2023 Jan 12].
10. C. Y. Wang, H. Y. Mark Liao, Y. H. Wu, P. Y. Chen, J. W. Hsieh and I. H Yeh. CSPNet: A New Backbone that Can Enhance Learning Capability of CNN. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 390-391. 2020. Available from: https://arxiv.org/abs/1911.11929 [Last accessed on 2023 Feb 06].
11. K. Wang, J. H. Liew, Y. Zou, D. Zhou and J. Feng. Panet: Few-shot Image Semantic Segmentation with Prototype Alignment. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9197-9206, 2019.
12. M. Horvat, L. Jelečević and G. Gledec. A Comparative Study of YOLOv5 Models Performance for Image Localization and Classification. In: *Proceedings of theCentral European Conference on Information and Intelligent Systems*. Faculty of Organization and Informatics Varazdin, pp. 349-356, 2022. Available from: https://shorturl.at/hFPT1 [Last accessed on 2023 Feb 16].
13. D. Dlužnevskij, P. Stefanovič and S. Ramanauskaite. Investigation of YOLOv5 efficiency in iPhone supported systems. *Baltic Journal of Modern Computing*, Vol. 9, pp. 333-334, 2021.
14. S. Mshir. *Vehicle Detection Dataset*. Roboflow Universe, 2023. Available from: https://universe.roboflow.com/shalaw/vehicle-detection-rfmsje [Last accessed on 2023 Feb].
15. S. Karslioglu. *Saving and Loading Models Across Devices in PyTorch*. PyTorch Organization, 2023. Available from: https://pytorch.org/tutorials/recipes/recipes/save_load_across_devices.html [Last accessed on 2023 Feb 02].
16. P. K. Yadav, J. A. Thomasson, S. W. Searcy, R. G. Hardin, U. Braga-Neto, S. C. Popescu, D. E. Martin, R. Rodriguez, K. Meza, J. Enciso, J. S. Diaz and T. Wang. Assessing the performance of YOLOv5 algorithm for detecting volunteer cotton plants in corn fields at three different growth stages. *Artificial Intelligence in Agriculture*, Vol. 6, pp. 292-303, 2022.

17. J. Lowe. *Precision and Recall in Machine Learning*. Roboflow Blog, 2022. Available from: https://blog.roboflow.com/precision-and-recall [Last accessed on 2023 Feb 03].
18. F. Damien, M. U. Javaid, N. Posocco and S. Tihon. Anomaly Detection: How to Artificially Increase Your F1-Score with a Biased Evaluation Protocol. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, Cham, pp. 3-18, 2021. Available from: https://arxiv.org/pdf/2106.16020.pdf [Last accessed on 2023 Feb 12].
19. J. Solawetz. *What is Mean Average Precision (mAP) in Object Detection?* Roboflow Blog, 2020. Available from: https://blog.roboflow.com/mean-average-precision [Last accessed on 2023 Feb 14].
20. Q. H. Phan, V. T. Nguyen, C. H. Lien, T. P. Duong, M.T. K. Hou and N. B. Le. Classification of tomato fruit using Yolov5 and convolutional neural network models. *Plants*, Vol. 12, p. 790, 2023.