IJRCS

International Journal of Robotics and Control Systems
Vol. 5, No. 3, 2025, pp. 1608-1624
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

ASCEE

A YOLO-Based Target Detection Algorithm for DJI Tello Drone

A'dilah Baharuddin [a,1], Mohd Ariffanan Mohd Basri [a,2,*]

[a] Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 81310 Johor Bahru, Johor, Malaysia
[1] adilahbaharuddin@gmail.com; [2] ariffanan@fke.utm.my
* Corresponding Author

ARTICLE INFO

Article history
Received April 30, 2025
Revised June 05, 2025
Accepted July 07, 2025

Keywords
Target Detection;
YOLO;
Real-Time;
Drone

ABSTRACT

The expansion of the application of drones has dispersed in wide range across military and civilian sectors. The application in such search and rescue missions are applicable with integration of computer vision and machine learning. A key feature of the drone for such applications is the capability to detect and locate objects and targets. Despite traditional methods perform excellently, deep-learning methods are the game changer in detection due to their better accuracy and robustness, rendering them ideal for real-time applications. The methods, including the YOLO series, are in continuous development to further enhance their performance. however, the regular issuance of updated and newer versions has intrigued curiosity regarding the potential superiority of the newer version over the previous versions in drone application. Hence, this paper has chosen the YOLOv8, YOLOv5u and YOLOv11 models for implementation on a DJI Tello drone to detect a custom target. A dataset for the target as a single class to be trained and validated is generated through images annotation. The target is required to be captured in the position of middle of the frame. However, the analysis upon performance metrics found that every model recorded high rates of precision, accuracy and recall. Yet, the simulations and experimentations displayed the ability of the model to accurately recognize the target. Thus, in order to evaluate the performance of each model thoroughly, it is advisable to ensure the data is sufficient and unbiased, while properly choosing the right setting parameters to the YOLO models.

1. Introduction

Emerging technology has piqued interest in enabling machines to imitate and replicate human capabilities. One of the exclusive capabilities of humankind is to process and understand visual data to recognize and distinguish things from one another. The ability is mirrored as object detection, an essential fundamental aspect of computer vision that enables machines to effectively analyse, interpret, and comprehend visual data by detecting and localizing objects inside images or video sequences [1], [2]. In general, object detection is categorized into traditional algorithms and deep-learning algorithms, where the traditional approaches rely on fundamental steps of image processing such as preprocessing, feature extraction, feature selection, classification, and post-processing [3]-[5].

Despite the traditional methods, such as Histogram of Gradient (HOG), Scale-Invariant Feature Transformation (SIFT), and Viola-Jones detectors have shown excellent performance, the emergence of deep learning has revolutionized the field of computer vision, especially in object detection,

particularly through deep convolutional neural networks (CNN) [6]-[12]. The deep-learning emergent is to facilitate detection methodologies to be more accurate and robust by functioning in a manner similar to how a human brain works in identifying the object by detecting the features of the images automatically [8]-[10], [13]. CNN's architecture plays crucial roles as their backbone's architecture in deep-learning methods, allowing the methods to extract feature maps from input images and preserve their spatial information [14], [15].

Based on the candidate regions produced by the frameworks of the detectors, the algorithms are categorized into two-stage and one-stage detectors, where the two-stage detectors adopt the fundamental idea of the traditional methods using convolutional features to enhance the feature expression ability significantly [3]. However, two-stage detectors demand a longer period to extract all proposals, as they first extract candidate regions before classifying and localizing these regions using CNN, as well as complex calculations due to the large number of nodes and layers of the network, making them unideal for real-time detection despite offering higher accuracy [5]. On the other hand, one-stage detectors offer simplicity and speed, as the algorithms utilize different proportions and sizes of predefined boxes to locate objects, effectively bypassing the candidates' regions extraction stage and directly predicting class probabilities simultaneously with feature extractions [5], [15]-[17].

Hence, to fulfil the demand of accurate and rapid processing in real-time applications, the constant development of one-stage algorithms, particularly the YOLO series, has enhanced the precision along with great processing time to levels with two-stage algorithms [11], [15], [18], [19]. As the YOLO series offers efficient deployment alongside accurate and quick target identification, it has become one of the most preferred and widely deployed algorithms especially for mobile devices, including drones [11], [20], [21].

Drones are initially utilized in military forces for target shooting practice, as can be perceived by the record of the real first drone; however, drones these days are actively integrated for various tasks from various sectors and industries, especially in agriculture, product delivery, transportation, telecommunications, and security and surveillance, where proficient vision-based capabilities are one of the demands [22]-[25]. At present, the employment of the advancing computerization and miniaturization technologies in the drones led to the reduction of production costs while extending the range of drone applications, making it an appealing alternative to human labor, particularly in high-risk tasks and confined workspaces, as drones can access unreachable locations with their agility and adaptability to nature [20], [26], [27]. With the possibilities of cooperating with computer vision and machine learning techniques, object or target detection has become one of the interests for the drone research community [28]. The search and rescue operation of Hurricane Katrina's survivors in 2005 remarks the first deployment of drones in performing non-military tasks [22], [23]. Despite the limitations of the drones, the utilization exemplified an innovative disaster's management, ultimately laying the foundation for various potential applications of drones.

With the choices of detection algorithms, including the YOLO series, the target or subject of interest, such as the survivors of Hurricane Katrina, can be differentiated from other things, easing the rescuers to find them instead of spending more time spotting them among the messiness of the tragedy site through the drones' streaming. Since June 2015, the YOLO community has been actively developing and launching updates with claims of improving the previous versions and solving any perceived issue. One of the mostly used iterations of YOLO is YOLOv5 released by team Ultralytics, which the model model has a smaller size with improved processing speed and accuracy compared to its predecessors, as well as its integration with being an open-source machine learning framework [29]-[31]. Later, the team released YOLOv8 with the aim to boost the performance of detection in various sizes, particularly smaller ones, while retaining largely similar backbone architecture of YOLOv5 [18], [32].

A study to compare the performance of YOLOv5 and YOLOv8 in detecting the tear film break-up based on ophthalmic videos is conducted by Z. Ruoyu et al. [33]. A total of 2808 labeled images are used as datasets, yielding the results of YOLOv5 with 51.4% is outperformed by YOLOv8 in

terms of accuracy at 54.2%. Living up to its main goal, YOLOv8, with 69.4% also outperformed YOLOv5 in accuracy with 47.6% in detecting smaller target. Despite scoring a slightly lower frame rate per second when executed on CPU, YOLOv8 remains at a commendable rate for real-time and even score higher on GPU. In another comparative study carried out by R.J. Iskandar et al. [34], both models are evaluated for detecting objects in low-light condition using a small pre-trained model. A total of 4132 images with three classes are used as datasets. The study also showed that YOLOv8 performed better than YOLOv5, with the accuracy rate of 70.4% over 59.3%, while mAP50 at 0.7019 by YOLOv8 over 0.5930 by YOLOv5 when trained with 100 epochs. On the other hand, A. Chen [18] found that in specific scenarios, YOLOv8 is outperformed by YOLOv5, such as in car detection where the precision, recall and accuracy rates of YOLOv8 fallen behind the rates of YOLOv5. The study found that the improvement of YOLO models is not in a linear path progression but rather trade-offs and tailored according to tasks.

As in January 2025, Ultralytics released another version of YOLOv11 upon the advancement of YOLOv8 with aim to boost the detection accuracy and processing speed, while in the meantime YOLOv5 which remain relevant still is updated to YOLOv5u by Ultralytics with improvement made to the backbone architecture [29], [32]. However, as the previous related works have concluded that YOLOv8 have tendencies of falling behind YOLOv5, the latest updated and released versions have intrigued question of whether the newer versions provide better target detection.

Thus, this paper aims to compare the performance of real-time target detection by a DJI Tello drone using YOLOv8, YOLOv5u and YOLOv11. The models are trained using a dataset of the custom designed object that acts as the target for the detection by the drone on a Python platform. Each model is tested with a DJI Tello drone's camera, with condition of target need to always be in the center of the drone's view. The performance of each model is evaluated with several standard fundamental metrics to quantify the consistency, accuracy and sensitivity of each model.

This paper is structured into five sections, with Section 2 elaborating on the fundamentals and the working principles of YOLO models, while Section 3 outlines the methods of utilizing the models, comprising the dataset preparation, model training, evaluation metrics and experiments. The results and performance data are analyzed and discussed in Section 4. Finally, Section 5 concludes and summarizes the paper, along with a hint of future work progression.

2. YOLO Series

YOLO, an acronym for 'You Only Look Once', is a detection algorithm that was first developed and issued in 2015 by its original team, and as the YOLO community grows and actively develops, the series has released several versions throughout the years [29], [35]. Fig. 1 shows the timeline of the models released from 2015 until January 2025. The initial team developed YOLOv1 through YOLOv3, while the later versions are developed by other active teams in the YOLO community, with YOLOv7 and YOLOv9 are developed by one team, whereas YOLOv5, YOLOv8 and YOLOv11 are developed by Ultralytics teams [29].

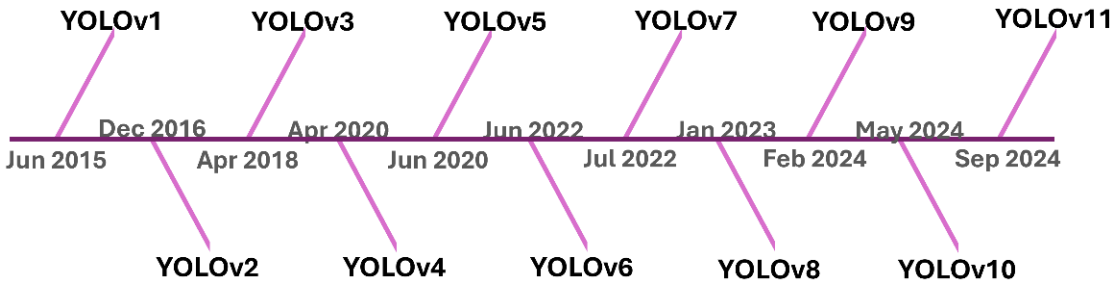

Fig. 1. The timeline of the YOLO series detector

Genuinely, YOLO integrates the detection paradigm of the region proposal stage with classification stage into unified neural network, ignoring the original method of executing each stage sequentially as, hence significantly reduces computational time [2], [11], [14]. The goals of YOLO

development are to aim to accurately and quickly detect objects, achieving good real-time performance [37]. Due to the high level of generalizability, YOLO is prone to less failure in unexpected situations and novel domains application [38]. In general, YOLO algorithms are made up of three elements: the backbone, the neck and the head [37], [39]. Fig. 2 shows the general architecture of YOLO [40].

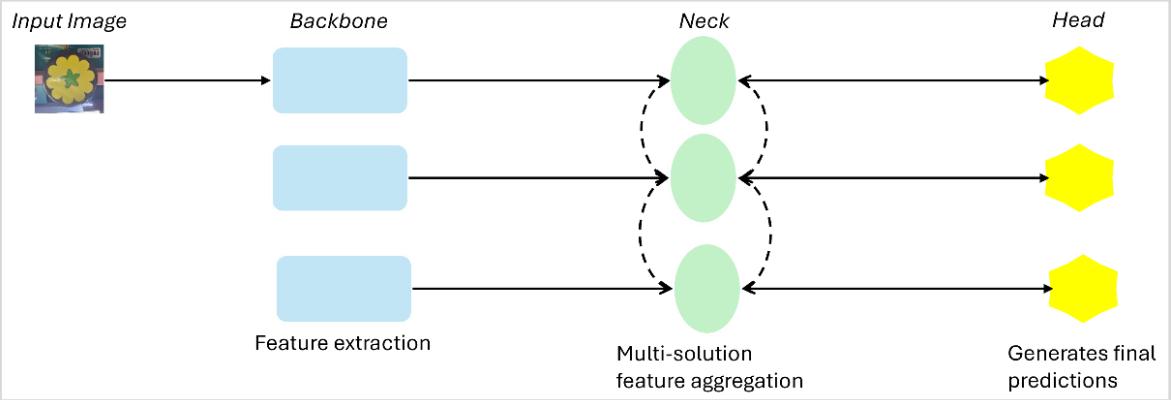

Fig. 2. General architecture YOLO [40]

The YOLO network is made of a backbone that encodes image features into multi-scales feature maps, a neck that processes these feature maps into integrated and refined representations, and a head that uses these processed features to generate prediction the bounding box and labels for classes [36], [39]. The backbone is the component that captures features from various scales and extract the characteristics from the input image via a trained CNN, before generalizing feature maps [39], [40]. The feature maps from the various layers of the backbone are then combined and merged in the neck of the algorithm and forwarding them to the head to make predictions according to confidence score, bounding boxes and the classification score [39], [40]. As the same general network is utilized by every model, the approaches taken by YOLO to detect the object are generally the same in every model.

Fundamentally, the input image is analyzed in its entirety in a single pass using residual blocks, in which the image is constructed into 'S x S' grid cells, with each cell obligated to predict the bounding boxes, the probability of an object existing within the underlying grid, and specific class probabilities [14]. With the employment of bounding box regression, the model is able to predict the position of the object as well as estimate the parameters of the bounding boxes, which comprise the coordinates of the center, the height, and the width of the box, as well as the class of the objects, as the model refines the bounding boxes during the training with the dataset to enhance the accuracy in prediction [1].

In ensuring the predicted bounding boxes overlap and align with the actual boundary of the object, the Intersection over Union, *IoU*, as defined in equation (1) is used as a normalized measure to effectively measure the localization performance, as the metric focuses on comparing the areas of the compared objects, while taking account the relevant parameters [14], [19], [30]. The ratio result should be obtained from (1) should be in between 0 and 1, where 0 represent no intersection and 1 represents the perfect intersect. Commonly, the prediction is considered as successful if the *IoU* calculated resulted in ratio exceeding 0.5 [1].

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{1}$$

Among the released iterations, YOLOv5 has been the most used iteration, as the model has a smaller size with improved processing speed and accuracy compared to its predecessors, as well as its integration with being an open-source machine learning framework [29]-[31]. The enhancement in the iteration is attributable to the adaptation of cross-stage partial network (CSPNet) which represented by C3 module into the backbone, which enhances the gradient combinations of the

network by dividing the feature maps of the base layer into two distinct parts through cross-stage hierarchy [9], [31].

Meanwhile, YOLOv8 is constructed upon the framework of YOLOv5, retaining largely similar backbone architecture, with aim to boost detection performances of objects in various sizes, particularly smaller ones [9], [18]. In order to enhance the efficiency of features extraction, the backbone is refined by replacing the C3 module with C2f module, whereas an anchor-free mechanism with additional heads is adopted into the model tasks including instance segmentation, pose key point detection, oriented bounding box (OBB) detection and classification tasks to enhance the detection precision especially the small objects [9], [29]. As the aim of YOLOv8 is achieved, the anchor-free split head feature is then adopted to the YOLOv5u, an updated version of YOLOv5, to refine the architecture of the model as an alternative to enhance the detection and processing speed of the model [41].

Notably, YOLOv11 is built upon the advancement of YOLOv8, featuring the enhancements to the backbone network to improve the processing speed without sacrificing the performance by substituting the C2F module with C3K2 module [9], [29]. To optimize the configuration across datasets and boost the detection accuracy, an adaptive anchor box mechanism is refined in YOLOv11 [18]. By preserving and extending the supports of various tasks by YOLOv8, YOLOv11 become one of the most capable and versatile detectors [29].

In general, YOLOv5 is designed to balance between the detection precision and processing speed, which contributes to its popularity for real-time applications, whereas YOLOv8 is tailored for real-time applications to predict the bounding boxes and class probabilities of the object with exceptional processing speed and accuracy, and YOLOv11 is built to enhance the feature extraction while achieving superior efficiency and precision [42].

3. Methodology

These three iterations released by Ultralytics vary into five different variants. The first variant is the ‘n,’ which is a compact yet the smallest and fastest model, while the next variant is the ‘s,’ which is an appropriate model for inference tasks. Another variant is the ‘m,’ a versatile choice for applications with a wider range and datasets, where it is a mid-size variant that offers the best trade-off between computational efficiency and precision. Due to the employment of a more complex feature extraction process, variant ‘l’ is meant for applications that require higher precision. Lastly, the largest and most complex model is the variant ‘x,’ in which it takes longer processing time and requires high-end processors for real-time inference but becomes the best choice for applications where the accuracy cannot be compromised [1], [35], [36], [43]. As this paper aims to perform simple target detection, the ‘n’ variants of YOLOv5u, YOLOv8, and YOLOv11 are used for the application

3.1. Custom Dataset

Dataset is important in training a machine to learn to recognize the target. The YOLO series accommodates several open-source datasets with the Common Objects in Context (COCO), Visual Object Classes (VOC) and Roboflow datasets being commonly used for general and unspecified objects or targets, such as humans, animals and vehicles. For the uses of non-general custom objects as a target, a custom dataset should be prepared to be used for training [44]. A total of 1050 images of the target under varying light intensity, different angles and backdrops are captured and partitioned into training and validating datasets adhering to 1000 images allocated for training and the remaining 50 images for validation [45], [46]. (The data is available on: https://drive.google.com/drive.

The images are annotated with CVAT, an open-source annotation tool to extract information of the images in the form of quantitative elements, which are the numerical values of each bounding boxes, and the form of qualitative element which categorized the of the target and the look-alikes. Fig. 3 illustrates the target and the impostors that possess almost the same features as the target, while Fig. 4 shows the annotation process, wherein the targets are bounded with red colored boxes labeled as ‘Target’.

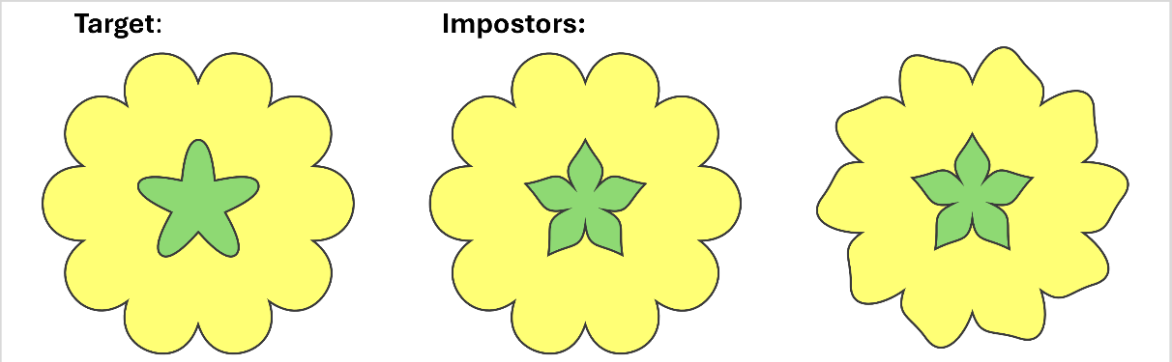

Fig. 3. The illustrations of targets and several look-alikes

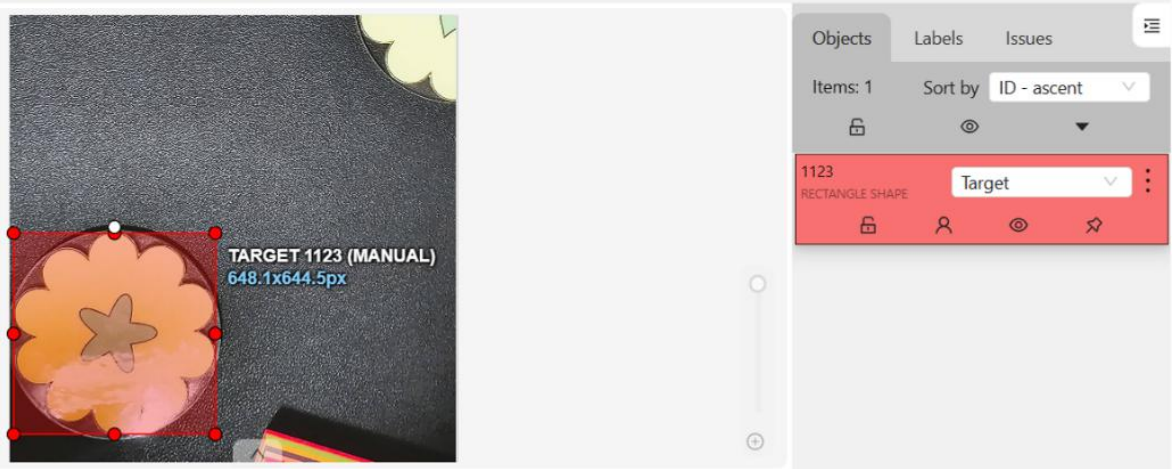

Fig. 4. Image annotation process

Each box in the annotation process produced a text line respectively, consisting of the class of the object and the normalized coordinates of the box, formatted as shown in Fig. 5. The data listed in Fig. 5 are examples of the text lines generated from the annotation process of the image in Fig. 4. The data are stored in a text file, named identically to each image. To enable each model to access these files during the training process, a YAML-formatted configuration file that specifies the dataset's directory and the class labels is created, as written in the snippet shown in Fig. 6. The numbers listed in Fig. 5 under <class> column, are defined as names in the configuration file, as shown in Fig. 6, which also referred to as the labels of the bounding boxes [46], [47].

<class>	<center x>	<center y>	<width>	<height>
0	0.513832	0.358403	0.220070	0.342083

Fig. 5. Data extracted from annotation process

```
path: C:\PROJECTS\PYTHON\JURNAL\YOLOS\V5C1\data     #dataset's directory
train: train\images                                 #dataset for training
val: valid\images                                   #dataset for validation

names:
  0: Target
```

Fig. 6. Configuration file for training

3.2. Model Training

To ensure consistency and comparability, the hyperparameters are conFig.d in a manner that is standardized across all models [42]. Table 1 summarizes the specifications of the hyperparameters used in the training sessions. The pre-trained weights, which are yolov5nu.pt, yolov8n.pt and yolo11n.pt are employed during the training phase to accelerate convergence process and improve detection accuracy [42]. These pre-trained weights are summoned in each model's training accordingly, as well as the previously created configuration file. In order to train and use YOLO, Ultralytics package is installed in the python environment. Fig. 7 represents the snippet of training commands for YOLOv5u, and the same commands are also used in other models.

Table 1. Specification of hyperparameters for training [42]

Hyperparameters	Description	Value
Epochs	Number of complete training dataset passes	300
Batch Size	Processed sample before model updates parameters	16
Momentum	Weight updates smoother & convergence accelerator	0.937
Weight Decay	Model overfitting reducer	0.0005
Warmup Epochs	Initial epochs	3.0

```
1  from ultralytics import YOLO                          #Importing YOLO's library from Ultralytics package
2
3  # LOADING THE MODEL
4  model = YOLO('yolov5nu.pt')                           #Model downloaded automatically upon commands run
5
6  # TRAIN THE MODEL
7  results = model.train(data="V5C2config.yaml", epochs=150)   #Calling configuration file & setting the epochs size
```

Fig. 7. YOLO models' training commands

3.3. Evaluations Metrics

In order to evaluate the target detection of each model, in which trained with annotated dataset, the standard foundational metrics are used to quantify the predicted bounding boxes relative to the ground truth and measure the models' performance. The models are evaluated metrics including mean Average Precision (mAP), recall, precision, box loss, class loss and object loss upon the completion of training and validation phases.

The accuracy of the detection is evaluated via precision, P in which a higher value measured indicates the ability of the model to minimize incorrect identification of the target, and demonstrating high accuracy [30]. The precision is defined by equation (2), where the True-Positive, TP represents the detection and classification are appropriately identified by the model in consistent with the annotation, while False-Positive, FP represents the model incorrectly detects non annotated object or incorrectly labels the class of the object [30], [42], [45], [48].

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

The sensitivity of a model is reflected by recall, R as the metric measures the ability of model identifying all positive samples or relevant objects within the annotate dataset. The metric focuses on minimizing failure in detecting the object that existed in the annotations. Recall is defined by equation (3) where False-Negative, FN denotes the inaccurate classification of the object predicted [30], [42], [49].

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

The efficiency of the model in accurately identifying the positive cases which are represented by recall, R and correct cases which represented by precision, P is accessed through the F1 score, as defined in equation (4), which the measurement ranges from 0 to 1. A lower F1 score, which calculated as harmonic mean between a high recall rate and low precision rate, or the other way around, signifies suboptimal performance of the model [30], [45], [50].

$$F1\ score = 2(P \times R)/P + R \tag{4}$$

The average precision, AP is the area under precision-recall curve, that evaluates the ability of the model to identify relevant things and abandon those that are irrelevant. In order to evaluate the detection performance across all classes, the mean average precision, mAP is calculated, with a certain degree of mistakes related to the placement of object and the dimensions of bounding box are allowed [31], [42], [51], [52]. Both AP and mAP are defined as in equation (5) and equation (6), respectively where N represents the numbers of object classes.

$$AP = \int_0^1 P(R)\delta R \tag{5}$$

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N} \tag{6}$$

3.4. Simulation

Each version of YOLO trained in the study is tested in real-time through webcam simulation. The streaming window of the real-time simulation is divided and partitioned into nine-sections as shown in Fig. 8.

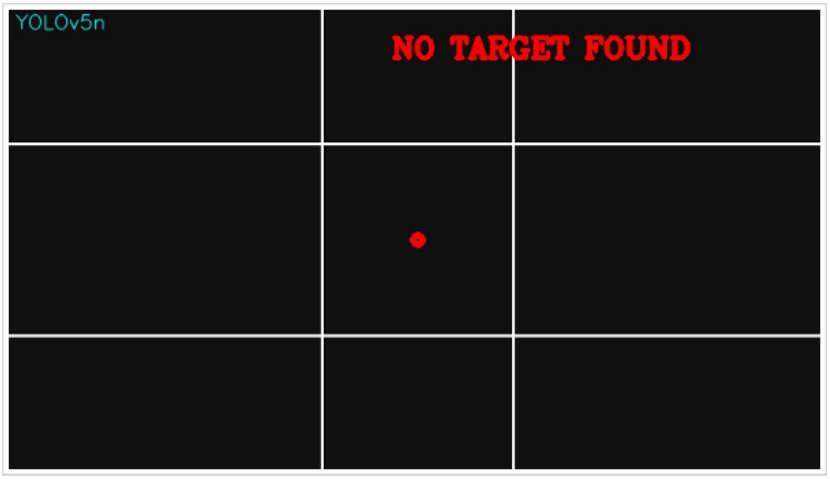

Fig. 8. The partitioned streaming window

The object to be detected needs to be in the desired position, which is the center of the frame. The same requirement will be applied to the detection with DJI Tello drone. The example shown in Fig. 8 represents the streaming window's view when the target is not detected. For the case the target is detected in any partition, a message or two of required action to be done will be displayed on the window.

3.5. Experiment

DJI Tello drone is utilized to perform real-time detection with drone. Tello drone is a small quadrotor that capable of providing stable flights, that it is expected to able to fly in challenging weather conditions and reaching 100-meter distance with maximum speed of 8m/s in a wind free environment [53], [54]. The drone offers a visual positioning system that allows it to hover in place and altitude holding function despite being compact and cost friendly [55]-[57]. The features are the result of the combination of IMU data with the mages taken by the built-in optical flow sensor at the bottom of the drone that analyzes the longitudinal and lateral movements across the time [53], [58]. For this study, the nose-mounted onboard camera is utilized to detect the target in real-time. The camera is capable of capturing 5MP images and streaming a 720pixel video and by featuring an electronic image stabilization (EIS), the images captured, and videos streamed are more stable even though vibrations occurred [59], [60].

In order to connect with the drone, djitellopy package is installed along with opencv-py package that processes the images and conducts the detection [59]. The drone real-time streaming is turned on and broadcasted to the connected laptop for observation and analysis process. The frame captured by the drone is also partitioned into nine-section as shown in Fig. 8 in the previous section. To ensure the target is captured in the center of the frame, the adjustment commands are made for the drone to adjust and position itself accordingly.

4. Results and Discussion

Each model of the chosen YOLO is trained with 300 epochs at a learning rate of 0.01 with a total of 16 batches in a time frame of around 20 to 24 hours. The training and validation process yields metrics that can be used for comprehensive analysis to assess the accuracy and consistency of detection and the efficiency of computation. Both the target and the target look-alike are annotated in order to test the models' ability to distinguish between the two. The evaluations are analyzed and discussed based on the performance of the models across all classes. As YOLOv5u is the updated version of YOLOv5, YOLOv8 becomes the predecessor of both YOLOv5u and YOLOv11. This is due to the fact that the base architecture of YOLOv11 is derived from the architecture of YOLOv8, while the architecture of YOLOv5u merges the elements from both the original YOLOv5 and YOLOv8 architectures.

Fig. 9 shows the captures of simulation and experimentation conducted on each version of YOLO. Each detection shows a successful detection with bounding box and label upon each detection. In the simulation, the messages are displayed in the frame as indicator the action needs to be taken to position the target in the center of the frame. Despite placing both target and impostor in the frame together, each version of YOLO tested successfully detect the target. Even though the target in each capture is in the center partition, every target does not fully fill the partition, thus a message of command is displayed for the user to move the target accordingly.

The target is also successfully detected by each version of YOLO during the real-time detection using DJI Tello drone. The message displayed on the frame upon detection is an indicator of the drone moving and adjusting it position to meet the requirement of capturing the target in the center of the frame and fully filled the partition. However, as the movement of the drone is initiated according to the preset command during the programming, the drone is having difficulties in repositioning itself. The movements of the drone during repositioning process are out of order and eventually cause the drone to lose the target from the frame.

Therefore, it is significant for the drone to be integrated with a robust controller such as PID controller, or intelligence algorithm controller to ensure that the movement initiated during repositioning process is in orderly mannered. In addition, it is also advisable for the system to be integrated with a prediction algorithm, such as Kalman filter. With the ability to predict, the system can make a prediction of the target location when the drone loses the target in a sudden during the repositioning process.

As the simulation and the experiment outputs are in visual data, the versions of YOLO utilized are evaluated with the data acquired from the training and validating process. Fig. 10 shows the result graph of training and validation performance of every model, where the loss functions and performance metrics are measured and visualized. The results in Fig. 10 demonstrate that the bounding box regression loss (box_loss), classification loss (cls_loss), and distributed focal loss (dfl_loss) curves exhibited a gradual decline and approached stability in every model. These patterns indicate that the features of the objects have been acquired more precisely by the models during training, resulting in a progressive enhancement of accurate position and classification prediction capabilities in detecting the objects.

In the meantime, the curves of performance metrics in both test and validation sets of every model shown in Fig. 10 are observed with an upward trend. This pattern is a validation of the effect of learning and generalization capabilities of the models. To further analyze the performance of the

models, the performance metrics are broken into several assessments focusing on the accuracy of the detection, the consistency of the detection, the computational efficiency, and the classification performance. The details of performance metrics, as reflected in the curves shown in Fig. 10, which includes the precision, recall, and mean average precision at the threshold of 0.5 or higher (*mAP@0.5*) for all classes, are organized accordingly in Table 2.

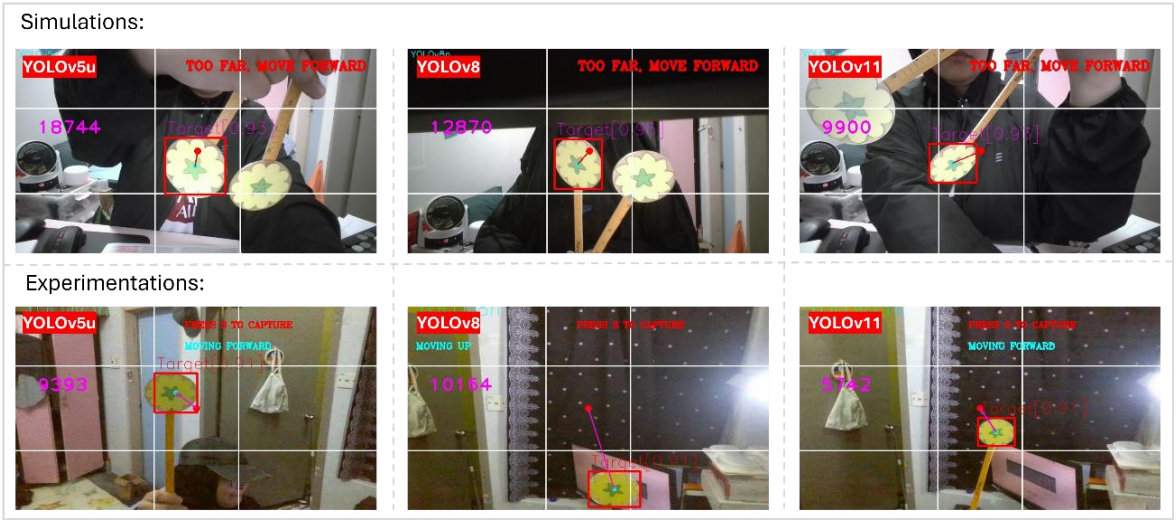

Fig. 9. The result graph of loss functions and performance metrics

Table 2. Performance metrics resulted from the training and validation process

Performance Metrics	Model (variant)		
	YOLOv8(n)	YOLOv5u(n)	YOLOv11(n)
Precision, *P*	0.974	0.964	0.960
Recall, *R*	0.982	0.980	0.990
mAP@0.5	0.990	0.973	0.973

The data listed in Table 2 reflected very small differences of rate in precision, recall and accuracy. The high rate of precision is an indicator of lower possibility of false negative detection where a detection with fewer incorrect recognitions and labels increases the probability of accurate label predictions, as well as the identification of class and location. Meanwhile, the rate of recall represents the false-negative detection, where the model erroneously fails to recognize and labels. From the rates record in this study, the possibility of false positive and false negative detection is concluded to be unlikely to happen as every version scored almost perfect accuracy and precision in detection. Such high rates are recorded as a result form the training for each version where only a single class with a total of 4645 instances is defined and annotated for the model training.

Further analysis is conducted upon the F1-score, where the graph reflects a model's ability to precisely capture positive cases, which involve the precision and recall of the model, in a balanced and maintained manner. On the other hand, the mean average precision (*mAP*) is evaluated to quantify the capability of the models in identifying the relevant things and excluding the erroneous ones. The F1-score reflects a model's ability to precisely capture positive cases, which involve the precision and recall of the model, in a balanced and maintained manner. The curves in Fig. 11 demonstrated that all models achieved and maintained high F1-scores across the wide range of confidence, which indicate and prove the robustness of the models.

The ability of the model to capture positive cases is mapped out in confusion matrices, which provide details of classification performance and error patterns of the models. The matrices present the true-positive rate of the detections in diagonal elements and the erroneous classification in off-diagonal elements. The confusion matrices are reorganised into Table 3. As the study define only one class of object, the confusion matrices data is formed by recognizing target from background, a class automatically generated by the YOLO framework.

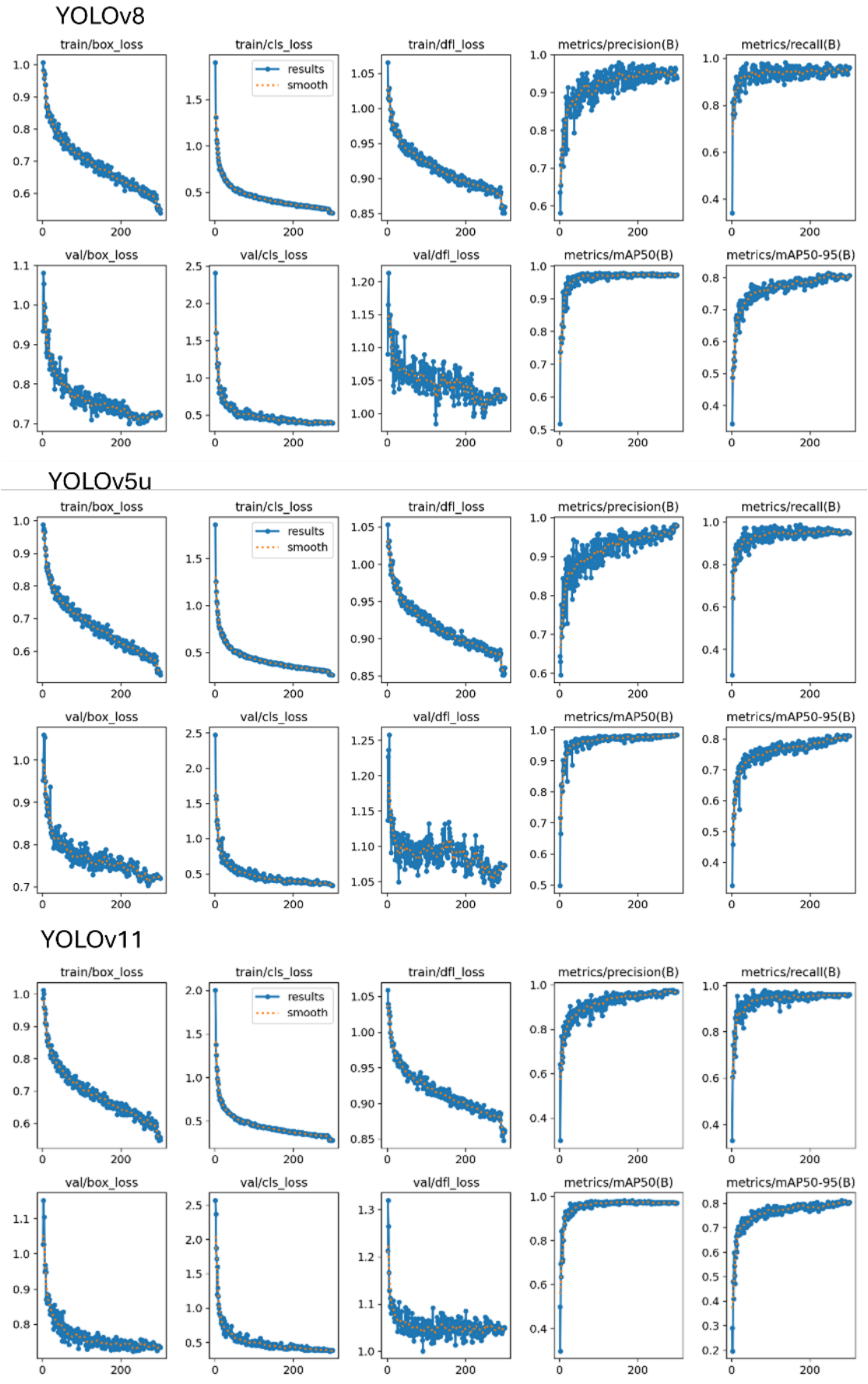

Fig. 10. The result graph of loss functions and performance metrics

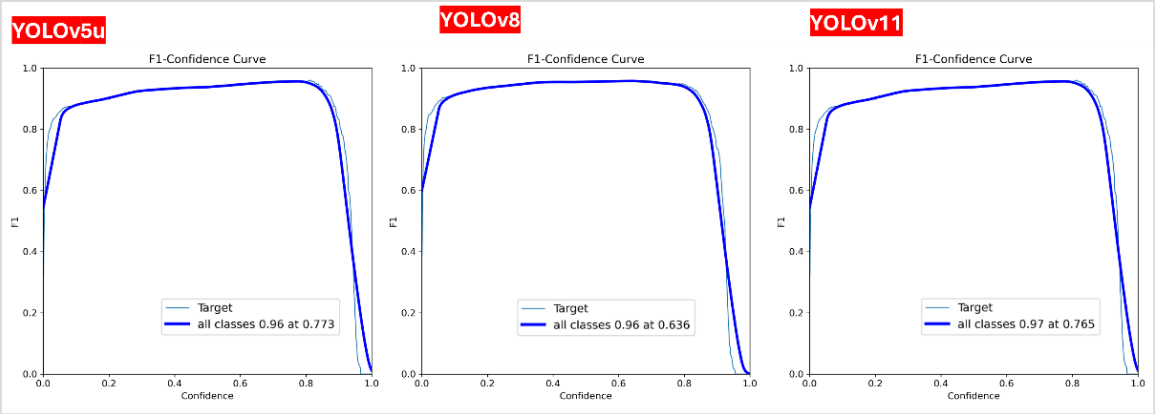

Fig. 11. The result graphs of F1-confidence

Table 3. Confusion matrices normalized

Pattern	Model (variant)		
	YOLOv8 (n)	YOLOv5u (n)	YOLOv11 (n)
True-Positive	0.98	0.97	0.96
True-Negative	0.02	0.03	0.04
False-Positive	1.00	1.00	1.00

From the data of the confusion matrices, every model reflects the high accuracy detection via the high rates of true-positive detection, and almost no true-negative cases recorded. However, the false-positive rates are at the maximum for each model. The false-positive detection is the result of the models implicit learn from the training, any false-positive, where the non-present target's frame is detected as target-present in the frame, is misclassified as background. This misclassification can be overcome with optimization of classes interest by removing any blank annotation data from the dataset.

In addition, even though the high training accuracy is desirable, a very high accuracy rate is also an indicator of the overfitting case, especially when the validation or test accuracy is much lower. In contrast, the lost graph of validation shows no sign of overfitting as can be observed from Fig. 10, where every model's training and validation errors are decreasing throughout the sessions. For the case of overfitting, the validation errors' graphs will eventually be increasing upon reaching overfitting iterations. Hence, the training sessions of the YOLO versions in this study are unlikely to be overfitting despite having a very high accuracy rate by each version. Nevertheless, the high rates of accuracy are plausibly due to data biases. The study defined only one class of object to be detected, where a total of 4645 instances is trained for single class detection. In addition, the high rates of accuracy are also plausibly due to the perfect fit of the data to the training model. Hence, to avoid such unclear evaluation, it is recommendable to train more than one class with perfectly chosen model parameters setting, including the epoch size.

Equally important, the computational efficiency of these YOLO models also recorded and analyzed. In large-scale real-time applications, computational efficiency plays a critical role in evaluating detection methods. Even though the scale of the study conducted in this paper can be considered small scale, the computational efficiency of each model is evaluated as the applications of detection models are employed in real-time. Efficiency reflects the ability of the model to immediately analyze and respond, which is the most important demand in real-time detections. Table 3 lists the processing times obtained from training reports of the models.

The preprocessing and post processing time for every model shows very small differences to one another. Due to the pattern of preprocessing and post-processing times of all models being relatively similar, the inference time of the models becomes the determiner of computational efficiency. The processing time indicates the time taken by the models to process the input image throughout their framework. The data in Table 4 shows that YOLOv11 took a slightly longer time to process the input

image as compared to YOLOv5 and YOLOv8. This is plausibly due to the complex architecture and framework of YOLOv11 that require more time for thorough decision-making processes. However, the difference can affect the performance of real-time applications that require very high-speed decision making as the model has the tendencies of slower detection. Hence, a proper parameter tuning or properly chosen additional algorithm can be integrated into the model to overcome the differences.

In a whole this study is unable to compare the performance of each model as the evaluations of each model reflects uncertainty of either the model is trained with perfect fit that contributes to a very high accuracy rate and speedy processing, or the model is trained with data biases as only a single class is defined and annotated for the training and evaluation.

Table 4. Processing time of YOLO versions

Efficiency Metrics	Model(variant)		
	YOLOv8(n)	YOLOv5u(n)	YOLOv11(n)
Preprocessing Time, *ms*	1.40	1.30	1.30
Inference Time, *ms*	54.9	54.0	64.4
Postprocessing Time, *ms*	0.40	0.40	0.50

5. Conclusion

This paper studies and compares the performance of target detection for a drone using a custom dataset. The YOLOv5u, YOLOv8 and YOLOv11 models are trained and validated with a dataset prepared by annotating the images of the target. YOLOv8 is defined as the oldest out the models, as both YOLOv5u and YOLOv11 adopt the model's architecture.

The evaluation of the accuracy, consistency, and efficiency of each model is conducted through the results of training and validating processes. However, each model recorded high rates of precision, recall and accuracy. Even so, no sign of overfitting is observed from any model. The high rate of training accuracy is likely due to the data bias occurrence or an indicator to a perfect fitness of the training as the study defined a single class of object. Thus, it is recommended to define more than one class of object to be trained with the models for better insight of the accuracy evaluations.

In the meantime, YOLOv11 took the longest time to process with 10ms slower than others. Despite being slower with a very small amount of difference, the performance of the model in the real-time application could be affected. Thus, it is recommendable to fine tune the parameters of the model in order to reduce the processing time. In conclusion, this study is unable to point out a model with the most excellent performance, to solve the question of whether the later version outperforming its predecessor. In the future, the study will be enhanced with thorough training session by properly chosen parameters setting and defined classes of object

The simulation and experimentation with DJI Tello drone also show that the drone has no issues in detecting the target with each model's integration. However, in order to position the target in the center frame of the drone, a robust and intelligence controller is required to ensure the drone able to capture a stable frame with a stable flight. The model with the most convincing performance will be chosen to be integrated with a properly tuned controller in a study of target tracking and following drone.

Author Contribution: All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding: This research was funded by Fundamental Research Grant Scheme of Ministry of Higher Education (MOHE) Malaysia, grant number (FRGS/1/2021/TK0/UTM/02/56).

Acknowledgment: The authors express gratitude to the Ministry of Higher Education (MOHE) Malaysia through Fundamental Research Grant Scheme (FRGS/1/2021/TK0/UTM/02/56) and Universiti Teknologi Malaysia through UTMFR (Q.J130000.3823.22H67) for the supporting of this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] M. Yaseen, "What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector," *ArXiv*, 2024, https://doi.org/10.48550/arXiv.2408.15857.

[2] M. Sohan, T. Sai Ram, and C. V. Rami Reddy, "A review on yolov8 and its advancements," *International Conference on Data Intelligence and Cognitive Informatics*, pp. 529-545, 2024, https://doi.org/10.1007/978-981-99-7962-2_39.

[3] M. Zhou, "Research Advanced in Deep Learning Object Detection," *2022 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)*, pp. 1318-1322, 2022, https://doi.org/10.1109/TOCS56154.2022.10016018.

[4] A. Baharuddin and M. A. Mohd Basri, "A Color-Based Real-Time Target Detection Using OpenCV for DJI Tello Drone," *2025 21st IEEE International Colloquium on Signal Processing & Its Applications (CSPA)*, pp. 77-82, 2025, https://doi.org/10.1109/CSPA64953.2025.10933104.

[5] Y. Zhou, "Research Advanced in Object Detection Based on Deep Learning," *2022 4th International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM)*, pp. 694-698, 2022, https://doi.org/10.1109/AIAM57466.2022.00141.

[6] Z. Luo, "The Application and Development of Image Recognition Technology in the Field of Computer Vision Highlights in Science," *Engineering and Technology,* vol. 85, pp. 909-914, 2024, https://doi.org/10.54097/g63vgv77.

[7] T. Ahmad, Y. Ma, M. Yahya, B. Ahmad, S. Nazir, and A. U. Haq, "Object detection through modified YOLO neural network," *Scientific Programming,* vol. 2020, no. 1, pp. 1-10, 2020, https://doi.org/10.1155/2020/8403262.

[8] X. Luo, K. Quan and Y. Liu, "Ball-Type Small Objection Algorithm Based on YOLOv8," *2024 IEEE/ACIS 24th International Conference on Computer and Information Science (ICIS)*, pp. 139-144, 2024, https://doi.org/10.1109/ICIS61260.2024.10778368.

[9] Z. He, K. Wang, T. Fang, L. Su, R. Chen, and X. Fei, "Comprehensive performance evaluation of YOLOv11, YOLOv10, YOLOv9, YOLOv8 and YOLOv5 on object detection of power equipment," *arXiv*, 2024, https://doi.org/10.48550/arXiv.2411.18871.

[10] A. Alhardi, M. A. Afeef, "Object Detection Algorithms & Techniques," *International Conference on Innovative Academic Studies (ICIAS)*, pp. 391-399, 2024, https://www.researchgate.net/profile/Anas-Alhardi/publication/379120594_Object_Detection_Algorithms_Techniques/links/65fc1e2ad3a08551423a1238/Object-Detection-Algorithms-Techniques.pdf.

[11] S. G. Verma, S. Maurya, H. Pant, A. K. Yadav, S. T. Varghese and R. Garg, "YOLO: Redefining Real-Time Object Detection Accuracy and Efficiency," *2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC)*, pp. 1123-1129, 2024, https://doi.org/10.1109/AIC61668.2024.10731055.

[12] S. K. Jaiswal and R. Agrawal, "A Comprehensive Review of YOLOv5: Advances in Real-Time Object Detection," *International Journal of Innovative Research in Computer Science and Technology*, vol. 12, no. 3, pp. 75-80, 2024, https://doi.org/10.55524/ijircst.2024.12.3.12.

[13] P. K. Hebbar and P. K. Pullela, "Deep Learning in Object Detection: Advancements in Machine Learning and AI," *2023 International Conference on the Confluence of Advancements in Robotics, Vision and Interdisciplinary Technology Management (IC-RVITM)*, pp. 1-7, 2023, https://doi.org/10.1109/IC-RVITM60032.2023.10435048.

[14] R. Padilla, S. L. Netto and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 237-242, 2020, https://doi.org/10.1109/IWSSIP48289.2020.9145130.

[15] W. Li, X. S. Feng, K. Zha, S. Li, and H. S. Zhu, "Summary of Target Detection Algorithms," *Journal of Physics: Conference Series,* vol. 1757, no. 1, p. 012003, 2021, https://doi.org/10.1088/1742-6596/1757/1/012003.

[16] K. Li and L. Cao, "A Review of Object Detection Techniques," *2020 5th International Conference on Electromechanical Control Technology and Transportation (ICECTT)*, pp. 385-390, 2020, https://doi.org/10.1109/ICECTT50890.2020.00091.

[17] R. Varghese and S. M., "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness," *2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, pp. 1-6, 2024, https://doi.org/10.1109/ADICS58448.2024.10533619.

[18] A. Chen, "Comparative Analysis of YOLO Variants Based on Performance Evaluation for Object Detection," *ITM Web of Conferences*, vol. 70, p. 03008, 2025, https://doi.org/10.1051/itmconf/20257003008.

[19] 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, 2023, https://doi.org/10.1007/s11042-022-13644-y.

[20] Y. Liu and X. Zhou, "Object Detection Algorithm for UAV Aerial Image Based on Improved YOLOv8," *2024 5th International Conference on Electronic Communication and Artificial Intelligence (ICECAI)*, pp. 789-793, 2024, https://doi.org/10.1109/ICECAI62591.2024.10675035.

[21] C. Y. Wang and H. Y. M. Liao, "YOLOv1 to YOLOv10: The fastest and most accurate real-time object detection systems," *APSIPA Transactions on Signal and Information Processing*, vol. 13, no. 1, p. e29, 2024, https://doi.org/10.1561/116.20240058.

[22] B. Maassarani, J. Epps, K. Garanger, M. T. Shahab, O. Wali, and E. Feron, "Tetrahedral and Dodecahedral UASs, Structured Designs," *Unmanned Aerial Vehicles Applications: Challenges and Trends*, pp. 3-41, 2023, https://doi.org/10.1007/978-3-031-32037-8_1.

[23] B. Marton, "History, types, application and control of drones," *Safety and Secuirty Science Review,* vol. 4, no. 3, pp. 1-13, 2022, https://biztonsagtudomanyi.szemle.uni-obuda.hu/index.php/home/article/view/222/202.

[24] O. Hall and I. Wahab, "The Use of Drones in the Spatial Social Sciences," *Drones*, vol. 5, no. 4, p. 112, 2021, https://doi.org/10.3390/drones5040112.

[25] I. Guvenc, F. Koohifar, S. Singh, M. L. Sichitiu and D. Matolak, "Detection, Tracking, and Interdiction for Amateur Drones," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 75-81, 2018, https://doi.org/10.1109/MCOM.2018.1700455.

[26] A. Barisic, M. Car and S. Bogdan, "Vision-based system for a real-time detection and following of UAV," *2019 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED UAS)*, pp. 156-159, 2019, https://doi.org/10.1109/REDUAS47371.2019.8999675.

[27] S. M. Abdallah, "Converting a DJI Tello Quadcopter into a Face-follower Machine using the Haar Cascade with PID Controller," *Cihan University-Erbil Scientific Journal*, vol. 7, no. 2, pp. 54-59, 2023, https://doi.org/10.24086/cuesj.v7n2y2023.pp54-59.

[28] A. Ramachandran and A. K. Sangaiah, "Areview on object detection in unmanned aerial vehicle surveillance," *International Journal of Cognitive Computing in Engineering,* vol. 2, pp. 215-228, 2021, https://doi.org/10.1016/j.ijcce.2021.11.005.

[29] T. Jiang and Y. Zhong, "ODverse33: Is the New YOLO Version Always Better? A Multi Domain benchmark from YOLO v5 to v11," *ArXiv*, 2025, https://doi.org/10.48550/arXiv.2502.14314.

[30] S. M. Abdallah, "Real-Time Vehicles Detection Using a Tello Drone with YOLOv5 Algorithm," *Cihan University-Erbil Scientific Journal,* vol. 8, no. 1, pp. 1-7, 2024, https://journals.cihanuniversity.edu.iq/index.php/cuesj/article/view/1010.

[31] T. Saidani, "Deep learning approach: YOLOv5-based custom object detection," *Engineering, Technology & Applied Science Research,* vol. 13, no. 6, pp. 12158-12163, 2023, https://doi.org/10.48084/etasr.6397.

[32] Z. He, K. Wang, T. Fang, L. Su, R. Chen, and X. Fei, "Comprehensive Performance Evaluation of YOLOv11, YOLOv10, YOLOv9, YOLOv8 and YOLOv5 on Object Detection of Power Equipment," *arXiv*, 2024, https://doi.org/10.48550/arXiv.2411.18871.

[33] R. Zhou *et al*., "A Comparison of YOLOv5 and YOLOv8 in the Context of Tear Film Break-Up Detection Based on Ophthalmic Videos," *2023 IEEE International Conference on Electrical, Automation and Computer Engineering (ICEACE)*, pp. 303-306, 2023, https://doi.org/10.1109/ICEACE60673.2023.10442470.

[34] R. J. Iskandar, C. Fatichah and A. Yuniarti, "Object Detection in Low-Light Conditions: A Comparison using YOLOv5 and YOLOv8," *2024 4th International Conference of Science and Information Technology in Smart Administration (ICSINTESA)*, pp. 558-563, 2024, https://doi.org/10.1109/ICSINTESA62455.2024.10748090.

[35] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," *ArXiv*, 2024, https://doi.org/10.48550/arXiv.2410.17725.

[36] R. Khanam and M. Hussain, "What is YOLOv5: A deep look into the internal features of the popular object detector," *ArXiv*, 2024, https://doi.org/10.48550/arXiv.2407.20892.

[37] R. H. Hasan, R. M. Hassoo, and I. S. Aboud, "Yolo Versions Architecture," *International Journal of Advances in Scientific Research and Engineering,* vol. 9, no. 11, p. 73, 2023, https://doi.org/10.31695/IJASRE.2023.9.11.7.

[38] M. G. Ragab *et al*., "A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023)," *IEEE Access*, vol. 12, pp. 57815-57836, 2024, https://doi.org/10.1109/ACCESS.2024.3386826.

[39] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, "YOLOv8 to YOLO11: A Comprehensive Architecture In-depth Comparative Review," *ArXiv*, 2025, https://doi.org/10.48550/arXiv.2501.13400.

[40] A. Soni and A. Rai, "YOLO for Medical Object Detection (2018–2024)," *2024 IEEE 3rd International Conference on Electrical Power and Energy Systems (ICEPES)*, pp. 1-7, 2024, https://doi.org/10.1109/ICEPES60647.2024.10653506.

[41] W. Wu *et al*., "Application of local fully convolutional neural network combined with YOLOv5 algorithm in small target detection of remote sensing image," *PLOS One*, vol. 16, no. 10, p. e0259283, 2021, https://doi.org/10.1371/journal.pone.0259283.

[42] R. Khanam, T. Asghar, and M. Hussain, "Comparative performance evaluation of YOLOv5, YOLOv8, and YOLOv11 for solar panel defect detection," *Solar*, vol. 5, no. 1, p. 6, 2025, https://doi.org/10.3390/solar5010006.

[43] Z. Zou, K. Chen, Z. Shi, Y. Guo and J. Ye, "Object Detection in 20 Years: A Survey," *Proceedings of the IEEE*, vol. 111, no. 3, pp. 257-276, 2023, https://doi.org/10.1109/JPROC.2023.3238524.

[44] A. Vijayakumar and S. Vairavasundaram, "YOLO-based Object Detection Models: A Review and its Applications," *Multimedia Tools and Applications,* vol. 83, no. 35, pp. 83535-83574, 2024, https://doi.org/10.1007/s11042-024-18872-y.

[45] H. Chang and Z. Wang, "UAV vehicle detection system based on YOLOv8," *Journal of Physics: Conference Series*, vol. 2872, no. 1, p. 012019, 2024, https://doi.org/10.1088/1742-6596/2872/1/012019.

[46] P. G. J. T, P. J, V. K, J. S. B, S. B and S. K V B, "Object Sensing for Visually Impaired Using Machine Learning," *2024 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*, pp. 1-7, 2024, https://doi.org/10.1109/ICPECTS62210.2024.10780148.

[47] L. Zhao and S. Li, "Object detection algorithm based on improved YOLOv3," *Electronics*, vol. 9, no. 3, p. 537, 2020, https://doi.org/10.3390/electronics9030537.

[48] I. Bisio, C. Garibotto, H. Haleem, F. Lavagetto and A. Sciarrone, "A Systematic Review of Drone Based Road Traffic Monitoring System," *IEEE Access*, vol. 10, pp. 101537-101555, 2022, https://doi.org/10.1109/ACCESS.2022.3207282.

[49] H. Sugiharto, Y. Silviana, and Y. S. Nurpazrin, "Unveiling document structures with YOLOv5 layout detection," *ArXiv*, 2023, https://doi.org/10.48550/arXiv.2309.17033.

[50] Z. Zhang, "Drone-YOLO: An efficient neural network method for target detection in drone images," *Drones,* vol. 7, no. 8, p. 526, 2023, https://doi.org/10.3390/drones7080526.

[51] L. Zhou, Y. Dong, B. Ma, Z. Yin, and F. Lu, "Object detection in low-light conditions based on DBS-YOLOv8," *Cluster Computing,* vol. 28, no. 1, p. 55, 2024, https://doi.org/10.1007/s10586-024-04829-1.

[52] S. A. Mostafa *et al.*, "A YOLO-based deep learning model for Real-Time face mask detection via drone surveillance in public spaces," *Information Sciences,* vol. 676, p. 120865, 2024, https://doi.org/10.1016/j.ins.2024.120865.

[53] G. Ghazi and J. Voyer, "Use of a DJI Tello Drone as an Educational Platform in the Field of Control Engineering," *Espace ÉTS*, 2024, https://espace2.etsmtl.ca/id/eprint/29458/.

[54] C. Lane, O. Jones, C. Miloro and M. Frye, "Development of Semi-Autonomous Flight & Detection Systems Using Small Drones," *2022 IEEE/AIAA 41st Digital Avionics Systems Conference (DASC)*, pp. 1-9, 2022, https://doi.org/10.1109/DASC55683.2022.9925805.

[55] P. García, L. Arribas, P. Pueyo, L. Riazuelo and A. C. Murillo, "Exploring DJI Tello as an Affordable Drone Solution for Research and Education," *2024 7th Iberian Robotics Conference (ROBOT)*, pp. 1-6, 2024, https://doi.org/10.1109/ROBOT61475.2024.10796954.

[56] M. Pawlicki, K. Hulek, A. Ostrowski and J. Możaryn, "Implementation and analysis of Ryze Tello drone vision-based positioning using AprilTags," *2023 27th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 309-313, 2023, https://doi.org/10.1109/MMAR58394.2023.10242452.

[57] O. Pohudina, M. Kovalevskyi and M. Pyvovar, "Group Flight Automation Using Tello EDU Unmanned Aerial Vehicle," *2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT)*, pp. 151-154, 2021, https://doi.org/10.1109/CSIT52700.2021.9648704.

[58] A. Madhani, K. Bingi, M. Omar, R. Korah, G. Kumar and B. R. Prusty, "Real-time object detection on DJI Ryze Tello quadrotor drone with YOLO and SSD algorithms," *International Conference on Green Energy, Computing and Intelligent Technology 2024 (GEn-CITy* 2024, pp. 66-71, 2024, https://doi.org/10.1049/icp.2025.0235.

[59] S. Jha, C. Seo, E. Yang, and G. P. Joshi, "Real-time object detection and tracking system for video surveillance system," *Multimedia Tools and Applications*, vol. 80, no. 3, pp. 3981-3996, 2021, https://doi.org/10.1007/s11042-020-09749-x.

[60] M. Iskandar, K. Bingi, B. R. Prusty, M. Omar and R. Ibrahim, "Artificial intelligence-based human gesture tracking control techniques of Tello EDU Quadrotor Drone," *International Conference on Green Energy, Computing and Intelligent Technology (GEn-CITy 2023)*, pp. 123-128, 2023, https://doi.org/10.1049/icp.2023.1770.