TẠP CHÍ KHOA HỌC ĐẠI HỌC HẠ LONG
Scientific Journal of Ha Long University
http://uhl.edu.vn/

# PHÁT HIỆN VẬT THỂ BAY THEO THỜI GIAN THỰC BẰNG YOLO V8,9,10

**Đoàn Trung Sơn[1*], Nguyễn Thị Khánh Trâm[1]**

[1]*Trường Công nghệ thông tin, Đại học Phenikaa*

*Email: son.doantrung@phenikaa-uni.edu.vn*

*Ngày nhận bài: 31/07/2025 Ngày nhận bài sửa sau phản biện: 17/10/2025 Ngày chấp nhận đăng: 24/10/2025*

## TÓM TẮT

Bài báo trình bày một mô hình chung để phát hiện vật thể bay theo thời gian thực có thể được sử dụng cho việc học chuyển giao và nghiên cứu sâu hơn, cũng như một mô hình tốt để phát hiện vật thể bay. Nghiên cứu đã thực hiện huấn luyện một tập dữ liệu chứa 40 loại vật thể bay khác nhau với dữ liệu được tinh chỉnh và xử lí một số trường hợp thực tế như: vật thể ở khoảng cách xa do vị trí đặt camera và vật thể bay thường ở trên bầu trời hoặc vật thể thường rất giống nhau như thủy phi cơ AG600 và US-2 khiến việc nhận dạng trở nên khó khăn. Bài báo tập trung vào việc sử dụng mô hình YOLO với các phiên bản mô hình mới nhất là 8, 9, 10. Kết quả thực nghiệm cho thấy mô hình YOLOv8-N đạt hiệu năng vượt trội so với YOLOv9-T và YOLOv10-N, với các chỉ số mAP50 đạt 91,1% và mAP50-95 đạt 87,3%. Ngoài ra, bài báo cũng chỉ ra mô hình YOLOv10-N khi đếm số lượng vật thể trong khung được tùy chỉnh theo yêu cầu cho trước đảm bảo kiểm soát được số lượng vật thể trong khung. Kết quả nghiên cứu cho thấy tiềm năng ứng ụng của mô hình cho kiểm soát sân bay hoặc an toàn an ninh hàng không.

***Từ khóa:*** *phát hiện vật thể bay, YOLO v8, YOLO v9, YOLO v10.*

# REAL-TIME FLYING OBJECT DETECTION WITH YOLO V8,9,10

## ABSTRACT

This paper presents a general model for real-time flying object detection that can be used for transfer learning and further research, as well as a refined model that achieves state-of-the-art results in detecting flying objects. The study trains a dataset containing 40 different types of flying objects with fine-tuned data to address several practical challenges, such as objects appearing at long distances due to camera placement, flying objects frequently observed against sky backgrounds, and visually similar objects (e.g., AG600 and US-2 seaplanes), which make accurate identification difficult. The paper focuses on the application of the YOLO model using its latest versions, including versions 8, 9, and 10. Experimental results demonstrate that the YOLOv8-N model outperforms YOLOv9-T and YOLOv10-N, achieving an mAP50 of 91.1% and an mAP50–95 of 87.3%. In addition, the study shows that the YOLOv10-N model can be effectively customized for object counting tasks under predefined constraints, enabling reliable control of the number of detected objects within a frame. These results indicate the potential applicability of the proposed approach to airport monitoring and aviation security and safety.

***Keywords:*** *Flying Object Detection, YOLO v8, YOLO v9, YOLO v10.*

## 1. INTRODUCTION

Although research suggests drone use will increase significantly, current detection technology still does not provide reliable and accurate results. Drones and small unmanned aerial vehicles (UAVs) are stealthy and can avoid detection by most modern radar systems because their electromagnetic signals are tiny. In conflict zones, real-time object detection is deployed to track the movements of soldiers and military vehicles. However, current systems are ineffective at detecting drones, which may explain why airborne threats go undetected. Detecting drones in this environment is difficult due to the complex terrain and distances over which these drones operate. The farther away from the detector, the more difficult it is to identify and classify the drone, as the object will transmit less signal in the input space of the model.

There have been reports of assassination attempts via drones with small explosive payloads, drug deliveries to state prisons, and surveillance of the United States (U.S) Border Patrol by smugglers (Tohidi, 2022) to exploit weaknesses. While research indicates that drone usage is expected to increase exponentially, detection technology has yet to provide reliable and accurate results. Drones and mini unmanned aerial vehicles (UAVs) are also small, highly maneuverable, and emit low noise. This, along with the ease of access, provides a natural incentive for drones to remain an integral part of modern warfare and illegal activities. While methods such as radio and acoustic detection have been proposed as solutions, they are currently known to be inaccurate (Roboflow, 2025). This motivates the integration of a visual detector in any such detection system. The U.S. Border Patrol implements real-time object detection from digital towers to monitor people and motor vehicles (Al-Qubaydhi et al., 2022). Still, it is not currently known to implement drone detection, which may explain the recent undetected illegal patrolling. Drone detection in this environment is challenging due to the cluttered desert background and the distance that drones survey from (Assalim, 2025).

Recently, there have been several studies related to identifying flying objects from real-time surveillance camera systems. When YOLOv8 was developed, there was research on the application for the problem of detecting flying objects and achieved a 99.1% mAP50, 98.7% Precision, and 98.8% Recall with 50 fps inference speed on the 3-class data set (drone, plane, and helicopter), surpassing models generated from previous research to a significant extent (Reis et al,2023).

Aydin & Singha (2023) proposed a YOLOv5 instance that achieved 90.40% mAP50, 91.8% Precision, and 87.5% Recall with 31 fps inference speed trained on a data set containing only drones and birds. Dai Duong Nguyen et al. (2024) proposed using an FPGA-SoC implementation of YOLOv4 for flying-object detection. Al-Qubaydhi et al. (2022) proposed a model utilizing the YOLOv5 framework that achieves an impressive 94.1% mAP50, 94.7% Precision, and 92.5% Recall on a dataset containing only one class of drones. In the study of Aote et al. (2023), a hybrid model based on the combination of a convolutional neural network (CNN) and long short-term memory (LSTM) was introduced to detect and classify UAVs. Bayesian optimization is used to tune the hyperparameters, making the results of the proposed hybrid CNN-LSTM model more promising when compared with other state-of-the-art algorithms such as YOLO, R-CNN, Faster R-CNN, SGD, and CNN.

However, research often focuses on certain flying objects; the accuracy for difficult cases is still very low, and cases of similar objects, obscured objects, and distances are too far.

The main goal of the paper is to identify flying objects in real-time based on the YOLO model with the latest versions. The inherent challenges of detecting flying objects, such as large variations in size, aspect ratio, speed, and occultation, require a sophisticated approach. By combining the latest advances in YOLOv8, YOLOv9, and YOLOv10, we strive to achieve an optimal balance between inference speed and detection accuracy. In this article, we will use and delve into the specific architecture and functions of the following versions: YOLOv8-N, YOLOv9-T, and YOLOv10-N to provide detailed information about their suitability for Real-time detection of flying objects and comparison of model improvements.

The distinctive aspect of this work compared to others is that it is the first to apply the latest version of YOLO for flying object detection. By training on a data set with 40 different classes of diverse flying objects to apply aviation service management, comparing the results, and customizing to manage traffic in and out of the management area proposed to use the current best model for this problem.

## 2. OVERVIEW

In this paper, we focus on some common types of flying objects, analyze some difficult real-life cases, and use the YOLO model to solve this problem.

### 2.1. Dataset

In this article, because flying objects are a very special class of objects, often used in military and aviation, it is very difficult to collect in practice. Therefore, we use the "*flying_object_dataset*" data set as presented in Figure 1.

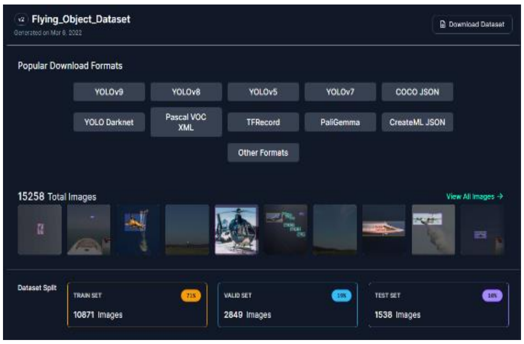


**Figure 1. The dataset**

In this article, we focus on handling some of the following common classes of flying objects:

*P-airplane*: transport or commercial aircraft.

*Drone*: a type of unmanned aircraft (UAV), controlled remotely or automatically.

*C-helicopter*: is a type of helicopter; it can be a military or civil helicopter.

*Bird*: is the object class for birds.

*Other types of combat aircraft*: Airplanes and UAVs in general.

We focus on describing and processing the data set in Part III Experimental Setup.

### 2.2. Difficulty

Many factors can affect the accuracy and the ability to detect flying objects in real-time:

– The distance from the Camera to the object is not fixed, the object size is tiny in the frame due to the very far distance, the object moves continuously, making it difficult to detect and easily confused with other flying objects

– *Lighting conditions:* In many situations, lighting is not good enough, for example, when flying objects are tracked in darkness or low light conditions. This causes the image to become blurry and noisy, affecting the input quality of the detection system (Figure 2).

**Figure 2. In low-light conditions, images are blurred and noisy**

– *Flying objects* are often obscured: Flying objects can be obscured by surrounding objects or move quickly so that the camera cannot detect them (Figure 3).

**Figure 3. The flying object is partially obscured by other objects**

– *Flying objects with similar shapes and characteristics*: Some aircraft, such as the AG600 and US-2 (Figure 4), are all seaplanes with high-mounted wing designs and bodies capable of taking off and landing on water.

**Figure 4. The AG600 and US-2 seaplanes have similar designs**

### 2.3. YOLO Model

YOLO (*You Only Look Once*) is an algorithm that uses neural networks to provide real-time object detection. This algorithm is popular because of its speed and accuracy. This model has been used in a variety of applications to detect traffic signals, people, parking meters, and animals. When launched, YOLO showed outstanding speed. So far, YOLO has developed into many different versions, including YOLO – v1, v2, v3, v4, v5, v6, v7, v8, v9, and the latest is v10 (Figure 5).

YOLO has been proven to outperform other machine learning algorithms in image recognition performance, especially in the image segmentation problem. The following is an overview of the YOLO version 8 and 9 models that the research will experimentally apply, improve, and evaluate.

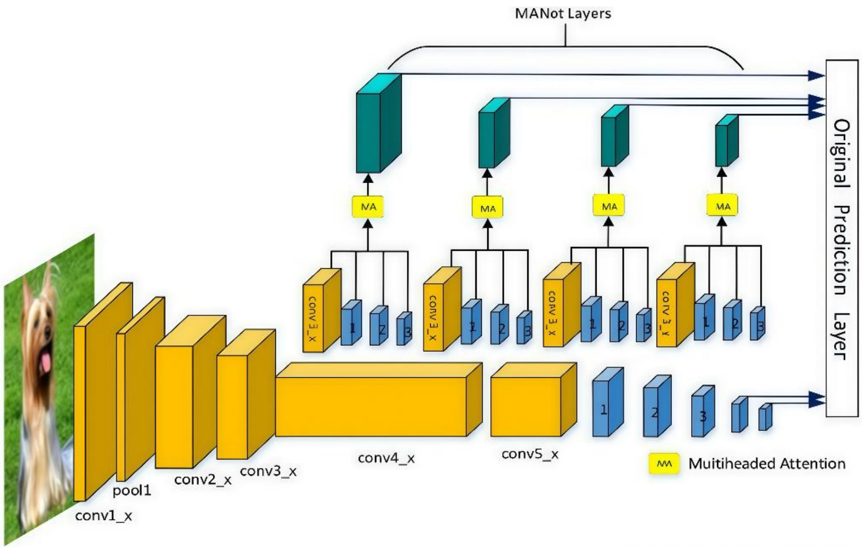


**Figure 5. Single Shot Detection Architecture**

**YOLO-V8**

YOLO version 8_segment is an advanced audience segmentation model based on the latest YOLO algorithm from Ultralytics, providing outstanding performance in speed and accuracy. YOLOv8 offers state-of-the-art performance and comes in the smallest to largest versions to suit a variety of applications: YOLOv8-n (Nano), YOLOv8-s (Small), YOLOv8-m (Medium), YOLOv8-l (Large), YOLOv8-x (Extra Large). YOLOv8 Nano is the fastest and smallest, and conversely, YOLOv8 Extra Large is the most accurate but slowest.

***Architecture:*** *Backbone CSPNet*, enhanced feature extraction capabilities; *Neck*: FPN and PAN, combining multi-dimensional information; *Head*: Single stage, performs prediction in one data transfer.

***Improvement:*** Anchor-free detection eliminates the anchor box mechanism, reducing parameters and improving accuracy. New C3 convolutional layer in the neck, enhancing feature extraction. Mosaic: A data augmentation technique to improve generalization. Multi-language support: Python, C++, C#.

YOLO-V8-segment has a fast processing speed and high accuracy, outperforming other object segmentation models (Figure 6): flexibility (different versions meet diverse user needs, from mobile devices to high-end computers), ease of use (provides a simple API, making model deployment a breeze).

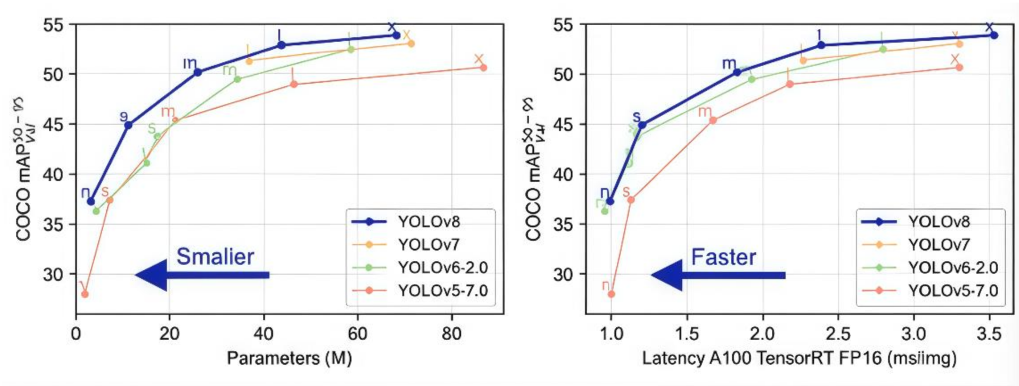


**Figure 6. YOLOv8 has outstanding performance (Al-Qubaydhi et al., 2022)**

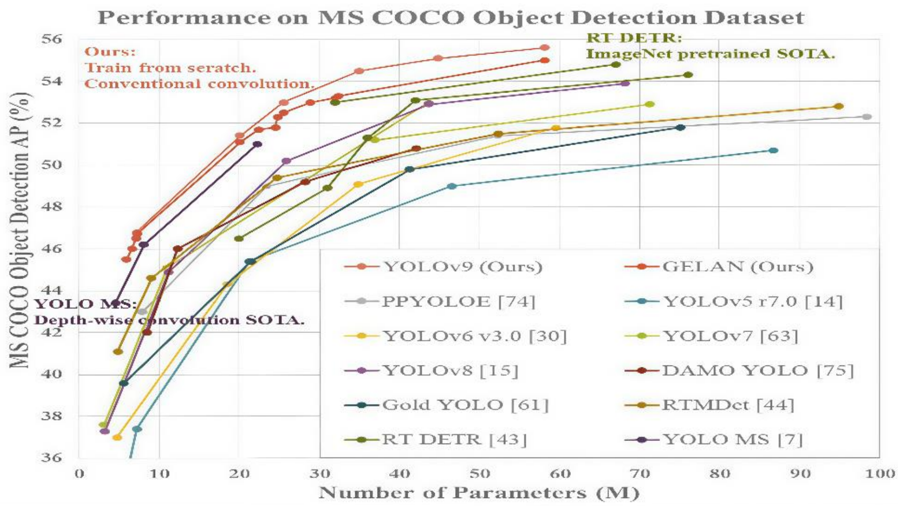


**Figure 7. A comparison of YOLOv9 with other models (Al-Qubaydhi et al., 2022)**

**YOLO-V9**

YOLOv9 is the most recent version of the YOLO model family, released in February 2024, bringing significant improvements in accuracy and performance. YOLOv9 versions include: YOLOv9-t (Tiny), YOLOv9-s (Small), YOLOv9-m (Medium), YOLOv9-c (Compact), YOLOv9-e (Enhanced).

Compared with YOLOv8_seg, YOLOv9_seg applies advanced deep learning techniques such as PGI (Programmable Gradient Information) and GELAN (Generalized Efficient Layer Aggregation Network) to improve performance (Figure 7).

***Architecture:*** *Backbone GELAN* (Global Enhanced Local Attention Network) improves information circulation and effective learning. *Neck:* FPN and PAN, combining multi-dimensional information. *Head:* Single-stage, performs prediction in one data transfer.

***Improvement:*** Programmable gradient information (PGI) preserves important information during detection. Optimization: Optimized for real-time, reducing computational and memory costs.

**YOLO-V10**

YOLOv10 is the latest version in the YOLO model line, developed and introduced in May 2024. YOLOv10 brings many outstanding improvements compared to previous versions, with versions: YOLOv10-n (Nano), YOLOv10-s (Small), YOLOv10-m (Medium), YOLOv10-l (Large), YOLOv10-x (Extra Large).

***Architecture:*** *Backbone* combines CSPNet, PANet, and SwiftNet, increasing processing speed and minimizing the number of parameters. *Neck:* FPN and PAN, combining multi-dimensional information. *Head:* Single stage performs prediction in one data transfer.

***Improvement:*** Dual labeling strategy combines one-to-many and one-to-one labeling methods, improving accuracy and efficiency. The consistent concordance index governs the supervision between the two labeling strategies, ensuring high prediction quality. Achieve outstanding performance on multiple benchmark data sets.

When comparing the models, YOLOv10 achieves the best balance between several parameters and accuracy, being highly accurate and resource-efficient for the problems (Figures 8 and 9).

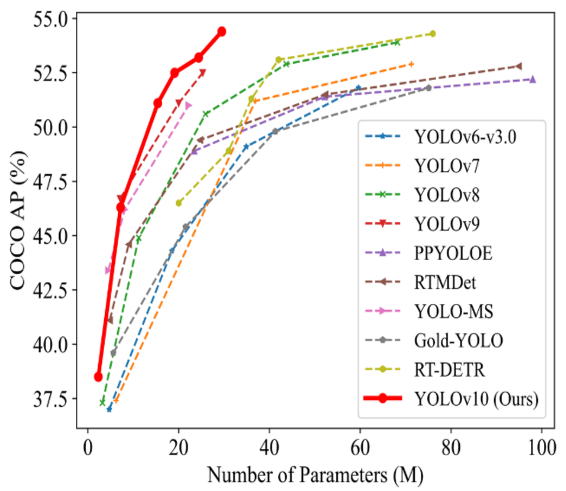


**Figure 8. Comparison chart of accuracy (COCO AP%) and Number of Parameters (M)**

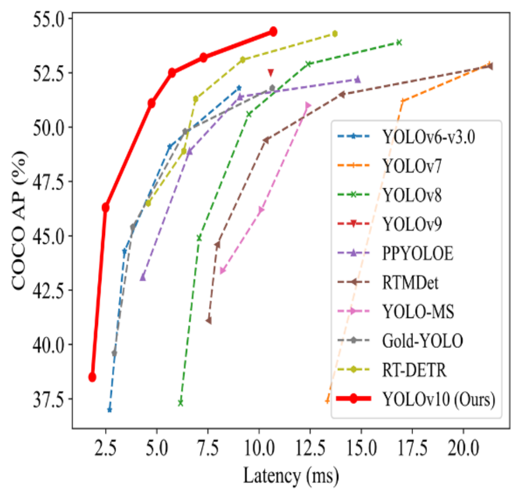


**Figure 9. Comparing accuracy (COCO AP%) and latency (Latency)**

## 3. EXPERIMENTAL SETUP

### 3.1. Data Collection and Pre-Processing

Flying objects are very special objects, and it is difficult to collect data from reality, so we used the flying_object_dataset publicly available at Roboflow.

### 3.2. Describe and process data

Data is used to detect flying objects from images or videos and deploy applications such as counting the number of objects to support traffic management, security, surveillance, and military processes.

During data preprocessing, we performed an Auto-Orient: An applied step that adjusts the image's orientation based on metadata, ensuring all data has the same standard orientation. This helps the model process and analyze data consistently, increasing accuracy in object detection and segmentation. Helps ensure consistency and increase data accuracy. And the data set during data preprocessing or model training does not have any enhancement techniques applied.

The *"flying_object_dataset"* dataset includes 15,258 images. Number of captions: 24,769 captions, with an average of 1.6 captions per image. Average image size: 0.17 megapixels (mp), ranging from 0.03 mp to 24.00 mp. Average image ratio: 416x416 pixels. Number of layers: 40 different object layers. Number of images missing captions: 0. Number of null examples: 59. From this, we can see that this dataset is very diverse in image size and scale, with a large number of classes and captions.

The data set is divided into 3 parts: Train, Valid, and Test (Table 1). Divided according to the ratio (71%-19%-10%) with a total of 10,871 images were used for training, 2,849 images for validation, and 1,538 images were used for testing.

**Table 1. Data division**

| Class | All | Train | Val | Test |
|---|---|---|---|---|
| A10 | 238 | 175 | 46 | 17 |
| A400M | 131 | 88 | 31 | 12 |
| AG600 | 106 | 69 | 28 | 9 |
| B1 | 218 | 150 | 39 | 29 |
| B2 | 220 | 160 | 21 | 39 |
| B52 | 174 | 118 | 38 | 18 |
| Be200 | 151 | 111 | 33 | 7 |
| bird | 1,373 | 1,010 | 227 | 136 |
| C130 | 1,407 | 1,060 | 211 | 136 |
| C17 | 268 | 188 | 56 | 24 |
| C5 | 151 | 104 | 29 | 18 |
| c-helicopter | 141 | 112 | 15 | 14 |
| drone | 5,637 | 3,925 | 1,133 | 579 |
| E2 | 131 | 91 | 25 | 15 |
| EF2000 | 119 | 83 | 22 | 14 |
| F117 | 93 | 70 | 14 | 9 |
| F14 | 123 | 78 | 25 | 20 |
| F15 | 284 | 204 | 54 | 26 |
| F16 | 328 | 229 | 64 | 35 |
| F18 | 252 | 166 | 50 | 36 |
| F22 | 157 | 114 | 28 | 15 |
| F35 | 287 | 198 | 68 | 21 |
| F4 | 129 | 92 | 24 | 13 |
| J20 | 106 | 76 | 21 | 9 |
| JAS39 | 144 | 94 | 33 | 17 |
| Mig31 | 91 | 67 | 13 | 11 |
| Mirage2000 | 106 | 78 | 17 | 11 |
| MQ9 | 93 | 62 | 20 | 11 |
| p-airplane | 1,753 | 1,305 | 269 | 179 |
| Rafale | 155 | 110 | 35 | 10 |
| RQ4 | 108 | 73 | 23 | 12 |
| SR71 | 102 | 79 | 13 | 10 |
| Su57 | 114 | 81 | 25 | 8 |
| Tu160 | 105 | 71 | 22 | 12 |
| Tu95 | 87 | 68 | 12 | 7 |
| U2 | 89 | 65 | 19 | 5 |
| US2 | 234 | 161 | 47 | 26 |
| V22 | 375 | 263 | 79 | 33 |
| XB70 | 82 | 53 | 23 | 6 |
| YF23 | 67 | 45 | 13 | 9 |
| null | 59 | 48 | 4 | 7 |

There are 15,199 images with objects and 59 images without objects. The purpose is to add images and process cases without objects, avoiding misidentification.

Each image can appear as one or more objects of different sizes, so each image can have more than one label. The total number of images in the table can be more than 15,258 images. The dataset for each object is diverse in type, color, shape, and size, and is made of different materials; a variety of special cases, such as multiple perspectives, blurry images, noise, low resolution, poor lighting, small objects, long distances, and single point cases.

### 3.3. Training Model

In this step, the author chooses a training model included in Table 2.

**Table 2. Models and parameters**

| Model | Version |
|---|---|
| YOLOv8 | YOLOv8-N |
| YOLOv9 | YOLOv9-T |
| YOLOv10 | YOLOv10-N |

The process of installing and testing the model is carried out through the following steps:

– Use the pre-trained model in Table 2.

– Install and configure the development environment on Visual Studio Code.

– Take advantage of the NVIDIA GeForce RTX 3070 Laptop GPU to speed up the training and experimentation process.

– Select Python 3.12.4 as a programming language.

– RAM 32GB.

The model training parameters are presented in Table 3.

**Table 3. Parameters**

| Argument | Parameters | Meaning |
|---|---|---|
| Model | YOLOv8n.pt<br>YOLOv9t.pt<br>YOLOv10n.pt | Specify a pre-trained model file for training |
| Data | flying_object_dataset-2/data.yaml | Path to the dataset configuration file |
| Epochs | 50 | The number of times the entire data set is repeated during the process |
| Batch | 16 | Number of data samples processed simultaneously |
| Image size | 640 | Training image size |
| Device | 0 | Computing equipment for training |

### 3.4. Evaluation Measurement

**MAP** (Mean Average Precision): This is a measure of the aggregate results of many queries applied to the search system. To calculate, we must have AP *(Average Precision)* as the average of the precisions at the threshold points returned by each correct result, written with the following formula:

$$AP = \sum_{k=0}^{k=n-1}[Rs(k) - Rs(k+1)] * Ps(k) \quad (1)$$

with recalls(n)=Rs(n)= 0, precisions(n)= Ps(n)= 1, n = threshold coefficient. Once AP is available, the formula for mAP is written as follows:

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k \quad (2)$$

*$AP_k$ = AP value of class k, n = Number of layers.*

We use an average accuracy of *$mAP^{bbox}$* (IoU=0.5 and IoU=0.95) along with mask *mAP* in forms (S-small, M-medium, C-Compact).

### 3.5. Model training detection results

The training results for YOlv8-N, YOLOv9-T, and YOLOv10-N with important parameters mAP50, mAP50-95, Precision, and Recall is presented in Tables 4 and 5:

**Table 4. Experimental Performance Results**

| Model | mAP 50 | mAP 50-95 | Precision | Recall | Training time (h) | Memory (MB) |
|---|---|---|---|---|---|---|
| Yolov8-N | 91.1 | 87.3 | 93.9 | 98.5 | 6.083 | 6.2 |
| Yolov9-T | 88.8 | 80.1 | 87.7 | 87.6 | 8.56 | 4.7 |
| Yolov10-N | 89.3 | 80,6 | 90.7 | 89.3 | 4.646 | 5.8 |

**Table 5. Experimental Model Results**

| Model | Preprocess (ms) | Inference (ms) | Postprocess(ms) | Process time | Layers | Parameter | Gflops |
|---|---|---|---|---|---|---|---|
| Yolov8-N | 0.3 | 1.9 | 2.5 | 4.7 | 168 | 3013448 | 8.1 |
| Yolov9-T | 0.4 | 7.4 | 3.6 | 11.4 | 486 | 1978584 | 7.7 |
| Yolov10-N | 0.3 | 2.4 | 0.5 | 3.2 | 285 | 2710016 | 8.4 |

From the results in Table 4, we can see that YOLOv8-N has the highest accuracy as well as the fastest processing speed, surpassing YOLOv9 and YOLOv10 in limited training time, and small model size due to limited resources and Hardware.

Results in Table 5 show that YOLOv9-T has the highest number of layers but the fewest parameters, which helps reduce the possibility of overfitting. However, the high number of layers causes increased training and inference time. YOLOv8 has high parameters that affect computational efficiency and memory consumption, but it can help the model learn more complex features from data. YOLOv10 shows the best balance between performance and resources, maintaining good performance for learning from the model. Gflops (8.4) shows the strongest computation among the models, which helps improve Processing speed and performance with more demanding computing tasks. The accuracy of the learning model was not optimized because the data set is quite large and has many different classes along with the uneven distribution of data. This leads to performance differences between classes. Accuracy can be compared with the results when focusing on certain classes, specifically classes like “drone”, "c-helicopter", and "p-airplane" have high accuracy and sensitivity due to a large number of samples and rich data:

– *drone*: Precision = 0.858, Recall = 0.976, mAP = 0.992

– *c-helicopter:* Precision = 0.875, Recall = 0.957, mAP = 0.982

– *p-airplane*: Precision = 0.825, Recall = 0.876, mAP = 0.908

In contrast, classes like “EF2000”, “Mirage2000”, and “Rafale” have worse performance due to fewer samples or uneven data:

– *EF2000*: Precision = 0.32, Recall = 0.214, mAP = 0.196

– *Mirage2000*: Precision = 0.156, Recall = 0.119, mAP = 0.123

– *Rafale*: Precision = 0.172, Recall = 0.116, mAP = 0.125

Therefore, to improve and enhance the accuracy of the model, it is necessary to focus on collecting and processing more data for classes with a small number of samples and irregular data.

### 3.6. Results in special cases

We conducted experiments with high-precision classes in a number of different cases. The results are presented in Figures 10 to 15.

YOLOv8-N YOLOv9-T YOLOv10-N

**Figure 10. Recognition results when there is only 1 object**

YOLOv8-N YOLOv9-T YOLOv10-N

**Figure 11. Recognition results when there are many objects**

YOLOv8-N YOLOv9-T YOLOv10-N

**Figure 12. Recognition results when there is no object**

We see that the results are good, the YOLOv10 model has the highest accuracy of 90-99% compared to YOLOv8 (85-94%) and YOLOv9. Next, we will experiment with special cases that can affect model accuracy, such as long distances, not fixed, objects obscured due to rotation angle, and objects with similar characteristics.

YOLOv8-N YOLOv9-T YOLOv10-N

**Figure 13. Recognition results when the object is too far away and small**

In the case of distant objects, very small in size, in the picture is a c-helicopter, YOLOv9-T detected the best of the 3 models with an accuracy of 74% (compared to the strongest model YOLOv10 54%).

**YOLOv8-N YOLOv9-T YOLOv10-N**

**Figure 14. Recognition results in bad conditions, partially obscured**

With the drone in bad conditions, with blurred and noisy images, YOLOv9-T continues to have good accuracy (54%) compared to the other 2 models (44% with v8 and 30% with v10). Similarly, when compared in the case of hidden objects, YOLOv9 also has the highest accuracy (91%).

**YOLOv8-N YOLOv9-T YOLOv10-N**

**Figure 15. The results identify two similar objects**

In this case, the shapes of two objects are seaplanes with similar shapes, but the versions do not confuse. The accuracy is over 90% and YOLOv8 has the highest accuracy. In general, the results in normal conditions are very good and have improved detection performance in bad conditions. YOLOv10-N recognizes best in normal conditions, accuracy and recognition speed are fast, however, in the case of small objects, low light, and obscured objects, etc., the YOLOv9-T model gives better results.

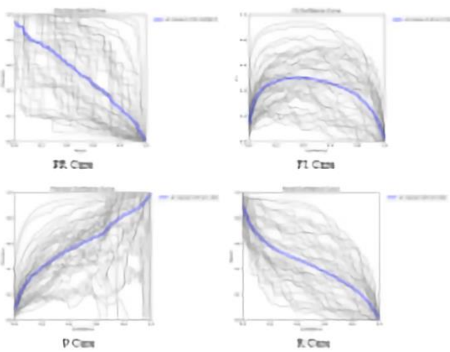

**Figure 16. Performance graph of the YOLOv8-N model**

Performance graphs based on F1-Confidence Curve, Precision-Recall Curve, Precision-Confidence Curve, and Recall-Confidence Curve (Figure 16), show that the model has high accuracy at high confidence levels and key object recognition ability body. The overall performance is quite good with mAP reaching 0.518; however, the performance drops sharply when increasing Recall, showing that the model may miss some objects when trying to cover more objects (Figure 17).

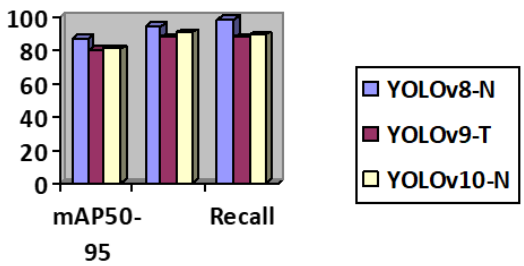


**Figure 17. Comparison of three versions of YOLO**

## 4. CONCLUSION

This research focuses on training a real-time flying object detection system using YOLO's latest and most advanced models, including YOLOv8, YOLOv9, and YOLOv10 to detect flying objects in images and provide information or warnings depending on the field of use. We used a fairly diverse dataset with 40 object classes such as aircraft, fighter jets, drones, helicopters, and birds. The results of the study showed that YOLOv8-N had the best overall performance with the highest mAP50, mAP50-95, and Precision values, although Recall ranked second. This model is suitable for applications requiring high precision. YOLOv10-N has good performance, preferred for use in applications that require the detection of more objects. YOLOv9-T demonstrates accurate object recognition under special conditions. This result demonstrates that the detection and tracking of flying objects is greatly improved when using advanced detection models as in this study. This serves as a basis for future research and development of methods to detect and track flying objects…

## ACKNOWLEDGMENT

The author gratefully acknowledges the support of Phenikaa University for this research.

## REFERENCES

Al-Qubaydhi, N., Alenezi, A., Alanazi, T., Senyor, A., Alanezi, N., Alotaibi, B., Alotaibi, M., Razaque, A., Abdelhamid, A. A., & Alotaibi, A. (2022). Detection of unauthorized unmanned aerial vehicles using YOLOv5 and transfer learning. *Electronics*, *11*(17), 2669.

Aote, S., Wankhade, N., Pardhi, A., Misra, N., Agrawal, H., & Potnurwar, A. (2023). An improved deep learning method for flying object detection and recognition. *Signal Image and Video Processing*, *18*(1), 1–10.

Assalim, J. (2025). *Weapons and knives detector with YOLOv8*. Retrieved July 29, 2025, from https://github.com/JoaoAssalim/Weapons-and-Knives-Detector-with-YOLOv8

Aydin, B. & Singha, S. (2023). Drone detection using YOLOv5, *MDPI*, *4*(1), 416–433.

Duong, N. D., Tuan, N. D., Thuy, L. M., & Cuong, N. Q. (2024). FPGA-SoC implementation of YOLOv4 for flying-object detection. *Journal of Real-Time Image Processing*, *21*(63). DOI: 10.1007/s11554-024-01440-w

Reis, D., Kupec, J., Hong, J., & Daoudi, A. (2023). *Real-time flying object detection with YOLOv8, archivePrefix.* DOI: 10.48550/arXiv.2305.09972

Roboflow. (2025). Flying object dataset. Retrieved July 29, 2025 from https://universe.roboflow.com/new-workspace-0k81p/flying_object_dataset

Tohidi, A. (2022). *Detection of pistol by deep learning with YOLOv5*. Retrieved July 29, 2025, from https://github.com/amin-tohidi/Detection-of-pistol-by-deep-learning-With-YOLO_v5