CLASSIFICATION AND DETECTION OF AERIAL TARGETS

A Deep Learning Project using YOLOv8

By Süreyya Yıldırım and Ahsen Dursun

June 2025

Contents

1. Introduction

1.1 Problem Definition and Background

In today's world, the use of airspace has expanded and diversified more than ever, ranging from civil aviation to military operations, logistics, and even recreational activities. Especially with the widespread adoption of Unmanned Aerial Vehicles (UAVs), the integration of traditional aircraft (such as airplanes and helicopters) has made the monitoring and securing of airspace a significantly complex problem. This diversity has created a growing need for autonomous systems capable of rapidly and accurately detecting and classifying airborne targets for applications such as airport security, military surveillance, protection of strategic infrastructure, and prevention of illegal activities.

1.2 Project Objective and Scope

The main goal of this project is to develop and evaluate a high-performance, deep learning-based computer vision system capable of real-time detection and classification of various aerial vehicles.

Within the scope of this project:

- **Target Classes:** A total of 27 unique classes were targeted, including Unmanned Aerial Vehicles (UAVs), passenger airplanes, civil helicopters, as well as 15 different types of military aircraft and 9 types of military helicopters.
- **Core Technology:** The project utilizes YOLOv8, a state-of-the-art object detection architecture known for its balance between speed and accuracy.
- **Outcome:** In addition to developing a robust model, a web-based user interface was also built to demonstrate the practical usage of the model.

1.3 Overview of Methodology

To achieve the project objectives, a data-driven and iterative methodology was followed. Initially, a comprehensive raw dataset was compiled from sources such as Roboflow and YouTube. This dataset was iteratively refined, cleaned, and restructured based on performance-driven approaches including experimental trainings and error analysis. Throughout the project, both YOLOv8s and YOLOv8m models were trained at different stages, and their performances were carefully evaluated using standard metrics such as mAP, precision, and recall.

1.4 Significance and Contribution of the Project

The significance of this study goes beyond applying an existing model to a defined problem. Its core contribution lies in the creation of a comprehensive dataset from scratch that includes diverse and challenging aerial targets, the systematic refinement of this dataset using a scientific methodology, and the development of a high-performing detection model. The project presents an end-to-end solution that encompasses the full workflow—ranging from data collection and model training to a final deployable application accessible to end users.

2. Literature Review

2.1. Introduction to Object Detection

Object detection is a fundamental and challenging task in the field of computer vision. Its primary goal is to determine the locations of specific objects within an image or video frame using bounding boxes and to assign these objects to their correct classes. While traditional methods employed techniques like Histogram of Oriented Gradients (HOG) for feature extraction, the field has shifted dramatically toward models based on Convolutional Neural Networks (CNNs) with the rise of deep learning.

2.2. Deep Learning-Based Object Detection Architectures

Deep learning-based object detection models are primarily categorized into two main approaches:

- **Two-Stage Detectors:** In this approach, the model first identifies potential regions of interest (region proposals) where an object might be located. Subsequently, these regions are passed through a classifier to determine the object's class and precise location. The R-CNN, Fast R-CNN, and Faster R-CNN series are the most well-known examples in this category. Although they typically offer high accuracy rates, their two-stage structure makes them slower and not always suitable for real-time applications.
- **One-Stage Detectors:** These models bypass the region proposal step, predicting the object's location and class in a single pass through the neural network. This structure provides them with a significant speed advantage. The YOLO (You Only Look Once) and SSD (Single Shot Detector) families are pioneers of this category. Given our project's goal of real-time detection, models from this category were chosen.

2.3. The YOLO Architecture and Its Evolution

Introduced by Redmon et al. in 2016, YOLO revolutionized the field by framing object detection as a single regression problem. Over the years, YOLO architecture has continuously evolved. Versions such as YOLOv3, YOLOv4, and YOLOv5 have progressively improved the balance between speed and accuracy by introducing different "backbone" networks, more efficient feature fusion layers in the "neck" (e.g., FPN, PANet), and optimized loss functions.

YOLOv8, which is central to this project, was released by Ultralytics in early 2023. It builds upon successful concepts from its predecessors (such as the C3 module from YOLOv5 and ideas from YOLOv7's ELAN) and introduces a new C2f module. Furthermore, its anchor-free design makes it simpler and more efficient. These features establish YOLOv8 as one of today's most powerful and flexible object detection models.

2.4. Academic Studies on Aircraft Detection

Aerial vehicle detection has gained significant attention recently, especially with the rise of UAVs. One of the primary challenges is accurately detecting **small objects** in aerial imagery. Zhou et al. (2024) introduced *Small Target–YOLOv5*, which enhances YOLOv5 with a weighted bidirectional feature pyramid network (BiFPN) and multi-head attention modules, achieving a **12.4% improvement in mAP** over the baseline on the VisDrone2021 dataset [1]. Similarly, a recent Nature paper proposed *PARE-YOLO*—a YOLOv8based model—highlighting enhanced feature fusion, a lightweight small-object head, and a new -EMAGIoU- loss, resulting in a **5.9% mAP boost** on VisDrone2019 [2] .

Other researchers focused on transformer-inspired and attentional enhancements. A study termed *HSP-YOLOv8* introduced a tiny prediction head and Space-to-Depth convolution, which improved mAP by **11% and 9.8%** on VisDrone2019 [3] There are also UAVspecific YOLOv8- variants—for example, *UAV-YOLOv8* integrates WIoU v3 loss, BiFormer attention, and a FasterNet block, reporting a **7.7% mAP gain** compared to the baseline [4] .

Attention mechanisms have also shown performance benefits. For instance, the SEBYOLOv8s model enhances UAV detection using SPDConv, AttC2f, and BRA modules, outperforming YOLOv8s on the Anti-UAV data-set [5] -. Additionally, in a PLOS ONE study, ASG-YOLOv5—enhanced with spatial gating and contextual attention—delivered a real-time **86 FPS** and improved mAP on VisDrone2021 [6] .

These studies consistently demonstrate that integrating **attention modules**, **small-object detection heads**, and **feature fusion networks** into YOLO-based architectures significantly boosts detection performance in aerial scenarios. This justifies our project’s focus on using YOLOv8 and employing customized enhancements for aerial vehicle detection.

2.5. Research Gap and Project's Contribution

A review of the existing literature indicates that studies tend to focus either on architectural improvements on public benchmark datasets or on a single type of vehicle (e.g., only drones).

The primary contributions of this project to the literature are as follows:

1. **A Comprehensive and Diverse Dataset:** A custom, large-scale dataset featuring 27 unique classes—including UAVs, civil/military helicopters, and over 15 different types of military aircraft—was created.
2. **A Data-Centric Refinement Methodology:** Instead of focusing solely on modifying the model architecture, this project presents a robust methodology

based on iteratively cleaning and improving the data through performance analysis. The case studies on the Airplane Passenger and Military Aircraft datasets demonstrate the success of this approach.

3. **An End-to-End Solution:** This work does not stop at training and evaluating a model but delivers a holistic workflow, from the data collection phase to a final web application accessible to end-users.

3. Methodology

This section provides a detailed explanation of the systematic approach followed in developing the aerial target detection and classification system. The overall workflow from data collection to model training, evaluation, and eventual integration with user interfaces-summarized in *Figure 3.1*. The subsequent subsections elaborate on each stage illustrated in this pipeline diagram.

3.1. Overall Pipeline

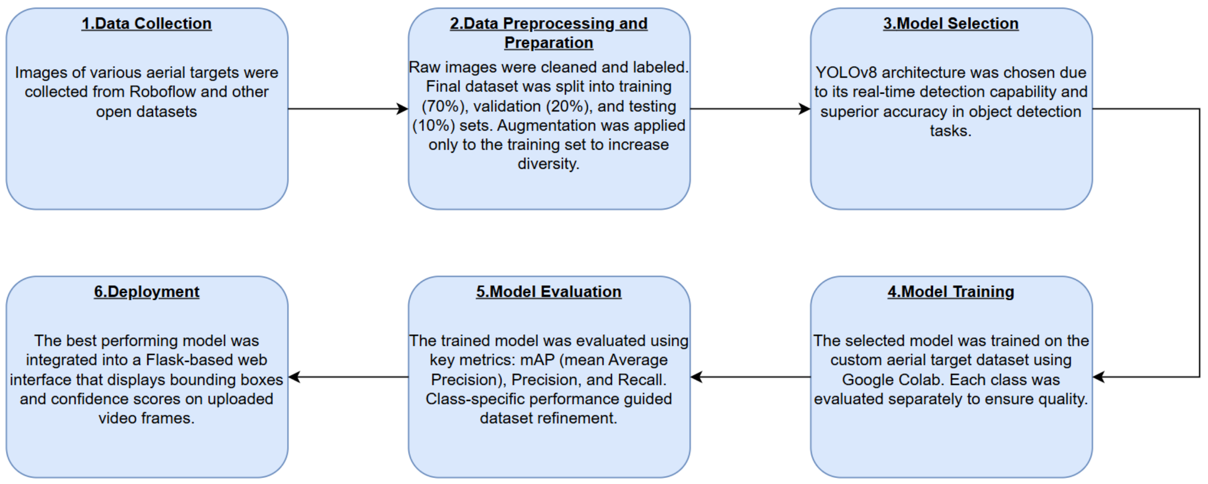

1.Data Collection
Images of various aerial targets were collected from Roboflow and other open datasets
2.Data Preprocessing and Preparation
Raw images were cleaned and labeled. Final dataset was split into training (70%), validation (20%), and testing (10%) sets. Augmentation was applied only to the training set to increase diversity.
3.Model Selection
YOLOv8 architecture was chosen due to its real-time detection capability and superior accuracy in object detection tasks.
4.Model Training
The selected model was trained on the custom aerial target dataset using Google Colab. Each class was evaluated separately to ensure quality.
5.Model Evaluation
The trained model was evaluated using key metrics: mAP (mean Average Precision), Precision, and Recall. Class-specific performance guided dataset refinement.
6.Deployment
The best performing model was integrated into a Flask-based web interface that displays bounding boxes and confidence scores on uploaded video frames.

Figure 3.1. *Overall Pipeline*

3.2. Data Acquisition and Preparation

This section outlines the processes involved in acquiring, processing, organizing, and finalizing the datasets used for model training.

3.2.1. Data Sources

The primary source for acquiring aerial vehicle images was the **Roboflow Universe** platform, which hosts a variety of open-source datasets commonly used in the field of computer vision. Relevant images and annotations were obtained from existing projects such as *"Military Aircraft Detection"* [8], which aligned with the scope of this study. However, it was observed that the available data for certain categories—such as **passenger aircraft**—was insufficient on the Roboflow platform. To address this limitation and enhance dataset richness, high-resolution video footage from **YouTube** was utilized. Video frames corresponding to the relevant classes were extracted and then manually annotated before being added to the dataset. This hybrid approach contributed to both the diversity and the comprehensiveness of the final dataset.

3.2.2. Raw Datasets

As a result of the data collection phase, the following raw datasets were constructed, categorized by aerial vehicle type:

- **Unmanned Aerial Vehicle (UAV):** A single-class dataset labeled as "UAV," consisting of approximately *4164* images.
- **Airplane Passenger:** A single-class dataset was constructed, comprising *4543* images extracted from both Roboflow and YouTube videos.
- **Airplane Military:** A multi-class dataset containing *36* categories such as fighter jets and F-Series, with approximately *4163* labeled images in total.
- **Helicopter Civil:** A single-class dataset labeled as "civil_helicopter," including approximately *2283* images.
- **Helicopter Military:** A multi-class data set representing various types of helicopters (e.g., attack helicopters, utility helicopters), composed of *10* classes and approximately *6768* images.

3.2.3. Data Preprocessing and Structuring

Before merging the raw datasets, a series of preprocessing, augmentation, and structuring steps were applied to improve data quality, increase data quantity, and enhance overall model performance. This process was carried out iteratively by conducting experimental training sessions for each dataset separately, followed by performance analysis and targeted adjustments. In all preliminary experiments, the datasets were split using a standard 70% training, 20% validation, and 10% test ratio.

A. Data Augmentation

To enrich datasets with relatively few samples such as **UAV** and **Civil Helicopter** and to enhance the model's generalization capabilities, various data augmentation techniques were applied via the Roboflow platform.

- **UAV Dataset**: Each training image was augmented to create 3.644 new images, using:
    - Horizontal Flip
    - Rotation between -15° and +15°
    - Saturation adjustment: ±13%
    - Brightness adjustment: ±15%
    - Blur up to 1 pixel
- **Civil Helicopter Dataset**: Each training image was augmented to create 2.283 new images, using:
    - Horizontal Flip
    - Saturation adjustment: ±14%
    - Brightness adjustment: ±15%

B. Experimental Training and Performance-Driven Cleaning

After initial preprocessing and augmentation, each dataset was independently trained using the YOLOv8 architecture to identify weak and strong components. Based on metrics such as *mAP, precision, and recall*, targeted cleaning and restructuring were conducted:

- **Military Helicopter Dataset**: Several images caused frequent misclassifications. These problematic samples were removed, resulting in a cleaner **v2** version of the dataset.
- **Military Aircraft Dataset**: Classes that consistently showed high error rates due to visual similarity or insufficient examples were eliminated. Additionally, low-quality images (blurred, misaligned, etc.) were filtered out.

This iterative, performance-based refinement significantly improved the quality of the final dataset. The following case studies on the Airplane Passenger and Military Aircraft datasets provide detailed illustrations of this process in action.

C. Case Study 1: Airplane Passenger Dataset

This dataset provided one of the most insightful experiments of the project. Two separate versions were explored:

Version 1 (v1): Multi-Class (Airbus vs Boeing)

- **Objective**: Test the model’s ability to differentiate passenger aircraft based on brand.
- **Dataset**: 4,543 raw images (Roboflow + YouTube) with 3x augmentation.
- **Result**: YOLOv8s trained for 20 epochs achieved a low mAP@50 of 0.573. While recall reached 0.935, precision was significantly lower, indicating frequent confusion between classes. This was attributed to the visual similarity between Airbus and Boeing aircraft.

Version 2 (v2): Single-Class (airplane_passenger)

- **Adjustment**: Airbus and Boeing labels were merged into a single class.
- **Dataset**: Augmentation was skipped to isolate base model performance; YOLOv8’s internal augment=True parameter was used during training.
- **Result**: A 30-epoch training with YOLOv8s achieved an outstanding mAP@50 of 0.994, validating the effectiveness of the single-class approach.

D. Case Study 2: Military Aircraft Dataset

- **Initial Condition**: The original dataset contained 36 classes, with a baseline mAP@50 of 0.601.
- **Adjustment**: Classes with poor performance or high confusion were removed, reducing the total to 22 classes. The dataset was expanded to 9,824 high-quality images.
- **Result**: With YOLOv8m trained for 75 epochs, mAP@50 improved to 0.761—demonstrating the benefit of pruning and balancing classes.

3.2.4. Final Dataset Consolidation

After cleaning and validation, all refined individual datasets were merged into a unified dataset for final training.

Global Class ID Mapping:
Each dataset originally assigned class IDs starting from 0, which caused ID conflicts upon merging. A **global class map** was created, assigning unique class IDs (e.g., UAV = 0, F-16 = 1, Boeing 747 = 2...) across all categories. This mapping was managed through a central data.yaml configuration file.

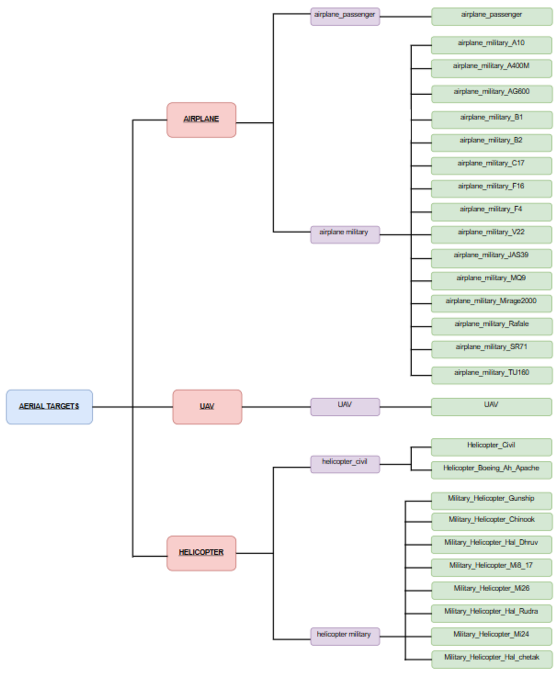


Figure 3.2. *Class ID Mapping*

Final Train-Validation-Test Split:
The unified dataset was split as follows to ensure optimal learning and generalization:

- **Training Set**: 80% (~X images)
- **Validation Set**: 10% (~Y images)
- **Test Set**: 10% (~Z images)

This structure ensured a sufficiently large training set while maintaining reliable validation and test subsets.

3.3. Model Architecture and Selection

This section outlines the deep learning model selected for this project, the rationale behind the selection, and the core architectural features that make it suitable for aerial object detection tasks.

3.3.1. Model Selection Rationale

The primary objective of this project is to detect and classify aerial vehicles both **rapidly and accurately**—a critical requirement for real-time applications. Given this need, we prioritized **single-stage object detectors**, which offer a strong balance between speed and accuracy, over **two-stage detectors** (e.g., the R-CNN family), which, although highly accurate, are typically slower and computationally intensive.

Among the single-stage detectors, the **YOLO (You Only Look Once)** family stands out in literature for its high performance and practical deployment capabilities. Therefore, YOLO-based models were selected as the backbone of the detection pipeline in this study.

3.3.2 Why YOLOv8?

YOLO has undergone numerous iterations since its initial release, with each version introducing substantial improvements. For this project, we adopted the most recent and advanced version available **YOLOv8**, developed by Ultralytics. The decision to use YOLOv8 was based on the following advantages:

- **High Performance:** YOLOv8 demonstrates significant improvements in both speed and accuracy (measured by mAP) compared to earlier versions, positioning it as one of the top-performing models in the field.
- **Flexibility and Scalability:** The architecture is available in multiple sizes—**nano (n), small (s), medium (m), large (l), and extra-large (x)**—allowing the model to be tailored to the complexity of each dataset used in the project.
- **Modern Architecture:** YOLOv8 incorporates cutting-edge innovations such as an **anchor-free detection mechanism** and an updated **loss function**, which simplify the detection process and improve overall model generalization.
- **Ease of Use:** With extensive documentation and streamlined workflows provided by the Ultralytics ecosystem, YOLOv8 simplifies the training, validation, and deployment phases of deep learning projects.

3.3.3. Model Versions Used in the Project

The selection of YOLOv8 model variants was guided by the trade-off between computational cost and detection performance. *Table 3.1* compares the core specifications of each YOLOv8 variant. The project employed YOLOv8s for lightweight tasks and YOLOv8m for more complex datasets with high class diversity.

Model Variant	Parameters (M)	GFLOPs	Speed (ms/img)*	Suitable For
YOLOv8n	3.2	8.7	~4.4	Edge devices, fast inference
YOLOv8s	11.2	28.8	~6.3	Small datasets, prototyping
YOLOv8m	25.9	78.9	~8.9	Mid-size datasets, better accuracy
YOLOv8l	43.7	165.2	~11.2	High-accuracy offline tasks
YOLOv8x	68.2	257.8	~14.5	Most accurate, resource-heavy

Table 3.1 *Comparison of YOLOv8 Model Variants*

Note: Inference speed varies depending on GPU (measured on Tesla V100). Source: Ultralytics YOLOv8 documentation.

As shown in Table 3.1, the **YOLOv8s** model—with 11.2 million parameters and an inference speed of approximately 6.3 ms per image—offered an ideal starting point for prototyping and handling smaller-scale datasets. Therefore, it was selected during the early experimental stages of the project and for less complex tasks such as **UAV** and **Airplane Passenger** detection.

For the more demanding task of training on the **Airplane Military** dataset, which contained 22 distinct classes, the project adopted the **YOLOv8m** model. With 25.9 million parameters, this medium-sized variant provided greater learning capacity compared to YOLOv8s, enabling improved accuracy in tasks involving high class diversity. This selection was based on a trade-off strategy, accepting a modest increase in inference time (~8.9 ms per image) in exchange for a significant gain in detection performance.

3.4. Model Training Process

3.4.1. Hardware and Software Environment

Training deep learning models requires significant computational resources. All training sessions and experiments in this project were conducted using Google Colab, a cloud-based platform that provides access to high-performance GPUs such as NVIDIA T4 and V100*,* without any additional hardware cost.

- The software stack and libraries used in the project are listed below:

- **Programming Language:** Python 3.x
- **Deep Learning Framework:** PyTorch
- **Object Detection Framework:** Ultralytics YOLOv8
- **Supporting Libraries:**
- OpenCV for image processing
- Matplotlib & Seaborn for visualization
- NumPy for numerical operations

3.4.2. Training Hyperparameters

The success of model training is closely linked to the correct selection of hyperparameters. Table 3.2 summarizes the key hyperparameters used during both experimental and final training stages. These settings were particularly applied for training the final model.

Parameter	Value	Description
Model Architecture	YOLOv8m	Medium-sized model used for final training
Epochs	100	Number of complete passes through the training dataset
Batch Size	16	Number of images processed per iteration
Image Size	640×640	Resolution to which all images are resized
Optimizer	AdamW	Optimization algorithm for weight updates
Learning Rate (lr0)	0.01	Initial step size for model learning
Augmentation	True	Built-in augmentations (mosaic, rotation, cutout, etc.) enabled
Device	CUDA (GPU)	Training executed on GPU rather than CPU

Table 3.2 *– Hyperparameters Used for Final YOLOv8m Model Training*

3.4.3. Training Monitoring and Evaluation

The training process was initiated using either the **command-line interface (CLI)** provided by Ultralytics or via **custom Python scripts**. At the end of each epoch, the model's performance was automatically evaluated on the validation set.

The following metrics and indicators were monitored throughout the training:

- **Loss Values:** Metrics such as box_loss, cls_loss (classification loss), and dfl_loss (distribution focal loss) were used to measure how far the model's predictions deviated from the ground truth. A decreasing loss trend indicated successful learning.
- **Performance Metrics:**

  Precision and **Recall** were used to measure the model's ability to correctly detect and classify objects.

  The most critical metric was **mean Average Precision (mAP)**, including:

  - **mAP@50**: Average precision at an Intersection-over-Union (IoU) threshold of 0.50.
  - **mAP@50–95**: Averaged over multiple IoU thresholds (0.50 to 0.95), providing a more comprehensive performance measure.

At the end of training, the **automatically generated result plots**—including loss curves, mAP trends, and sample predictions from the validation set—were reviewed to conduct a thorough performance analysis

4. Experimental Results and Analysis

This section presents the results of individual training experiments conducted prior to final data set consolidation. These results supported key decisions regarding data cleaning and restructuring**.**

4.1. Experimental Training Results

4.1.1. UAV Dataset Results

The first experimental training was conducted on the single-class **UAV dataset** to establish a baseline performance for the project.

- **Training Details**

As described in the methodology, the UAV training set was enriched using various augmentation techniques. The final dataset included **5,466 training**, **1,560 validation**, and **782 test** images. A 15-epoch preview training was performed using the **YOLOv8s** model.

- **Performance Metrics**

Table 4.1 summarizes the model's performance on the validation set after 15 epochs.

Metric	***Value***
Precision	0.952
Recall	0.868
mAP@0.50	0.928
mAP@0.50:0.95	0.738

Table 4.1 *– UAV Dataset Training Results (15 Epochs)*

- **Visual Results and Analysis**

The training and validation loss curves, as well as performance trends over epochs, are shown in **Figure 4.2**. These graphs indicate a consistent decrease in loss values and steady improvement in precision, recall, and mAP scores throughout the training process. Sample detection outputs on the validation set are presented in **Figure 4.3**, illustrating the qualitative performance of the model.

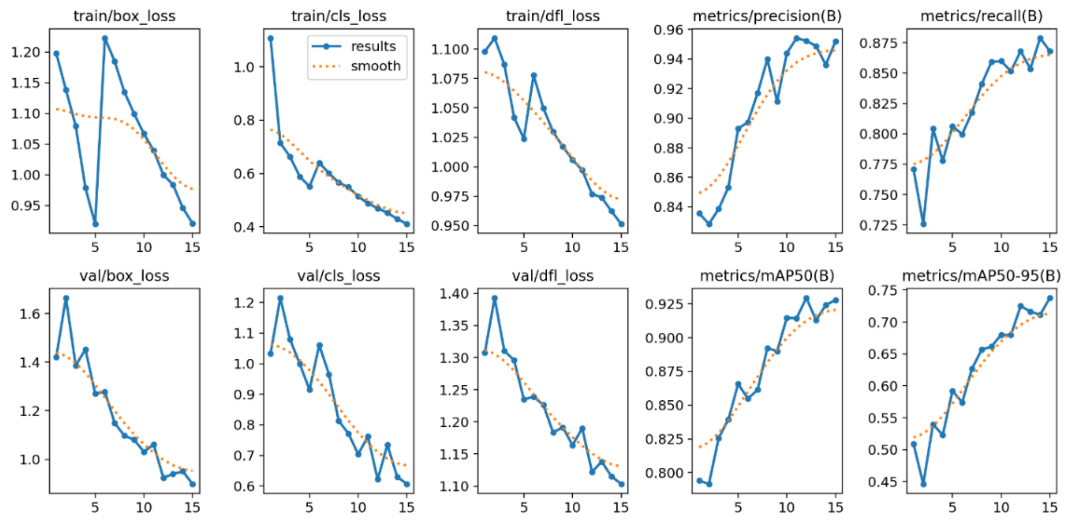


***Figure 4.2** – UAV Model Training and Validation Graphs*

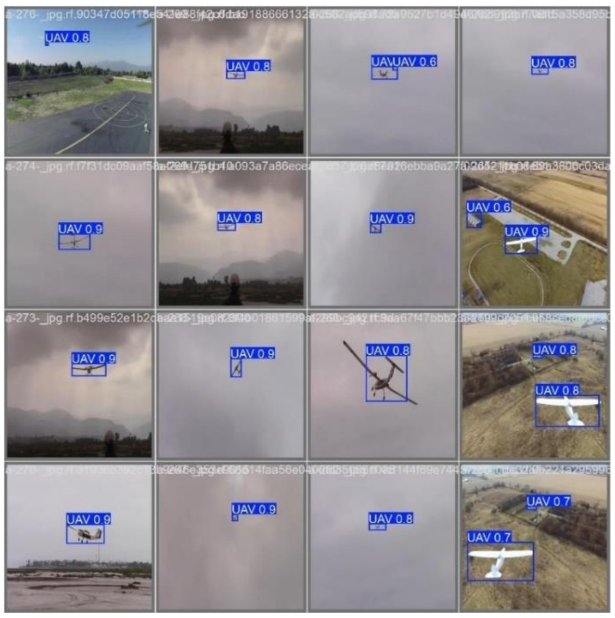


***Figure 4.3** – Sample Detection Results for the UAV Dataset*

- **Evaluation**

The high **mAP@0.50 of 0.928**, achieved after only 15 epochs, demonstrates the model's strong capability in detecting UAVs. This result confirms both the quality of the dataset and the suitability of the YOLOv8s model for this task. Due to the already satisfactory performance, no additional training or cleaning was deemed necessary for this dataset.

4.1.2. Airplane Passenger Dataset Results

The **Airplane Passenger** dataset underwent a two-phase experimental process, which led to one of the most important methodological decisions of the project. This experiment offered valuable insights into the model's limitations and guided optimization of the data strategy.

Version 1 (v1): Multi-Class Approach (Airbus vs. Boeing)

- **Objective:**
  The goal of this experiment was to evaluate the model's ability to distinguish between **Airbus** and **Boeing** aircraft by treating them as two separate classes.

- **Training Details:**
  A total of **4,543 raw images** were collected from Roboflow and YouTube, then tripled via data augmentation. The resulting dataset included **5,970 training**, **1,702 validation**, and **851 test** images. The model used was **YOLOv8s**, trained for **20 epochs**.

- **Performance Metrics:**
  The results are presented in **Table 4.2**.

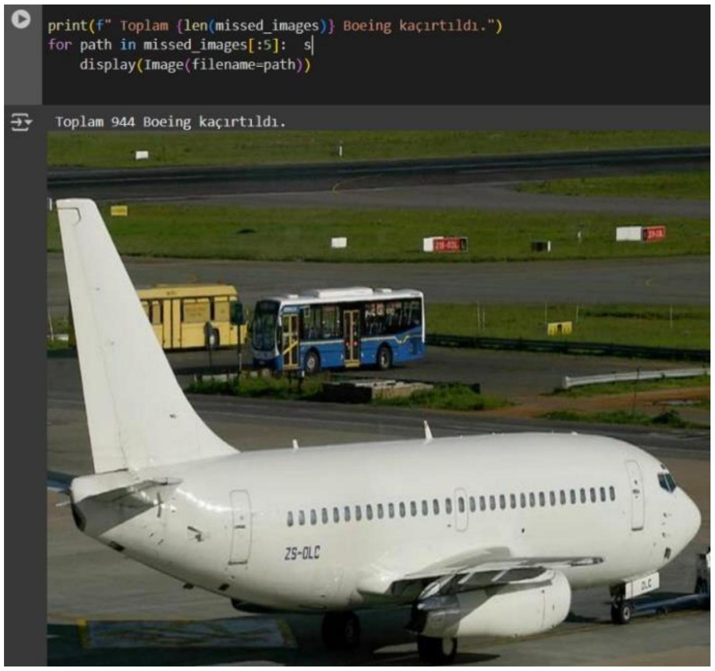
Metric	Value
Precision	0.502
Recall	0.935
mAP@0.50	0.573
mAP@0.50:0.95	0.513

Table 4.4 *– Airplane Passenger v1 (Multi-Class) Training Results*

- **Analysis and Evaluation:**

These results clearly indicated that the multi-class approach was ineffective. As shown in Table 4.4, although the **recall** was relatively high (0.935), the **precision** dropped significantly to 0.502, meaning nearly half of the detections were misclassified. Upon manual inspection, the false negatives were not due to labeling errors or poor image quality. Instead, the problem was attributed to the **strong visual similarity between Airbus and Boeing** aircraft, which the model struggled to distinguish. The fine-grained class boundaries proved to be too subtle for the model to reliably capture.

Figure 4.5 illustrates one of the 944 Boeing aircraft instances that the model failed to detect (false negative), despite the image being properly labeled and of high visual quality.



```
print(f" Toplam {len(missed_images)} Boeing kaçırtıldı.")
for path in missed_images[:5]:  s
    display(Image(filename=path))
```

```
Toplam 944 Boeing kaçırtıldı.
```

Figure 4.5 *– Example of a Boeing Aircraft Missed by the v1 Model*

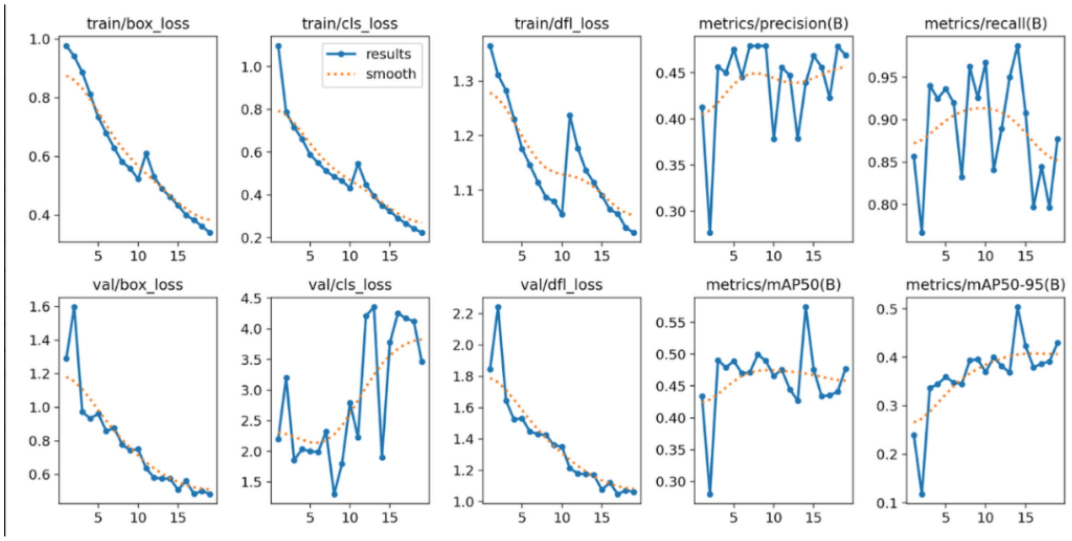


Figure 4.6 *– Airplane Passenger v1: Model Training and Validation Graphs*

Version 2 (v2): Single-Class Approach (airplane_passenger)

- **Methodological Adjustment:**
  Based on the findings from v1, Airbus and Boeing labels were merged into a single **airplane_passenger** class.

- **Training Details:**
  To better observe the base model performance, no external augmentation was applied. Only YOLOv8's internal augment=True setting was used. The new dataset included **3,451 training**, **990 validation**, and **494 test** images, trained with **YOLOv8s** for **30 epochs**.

- **Performance Metrics:**
  The dramatic improvement is summarized in **Table 4.3**.

Metric	**Value**
Precision	0.992
Recall	0.987
mAP@0.50	0.994
mAP@0.50:0.95	0.965

Table 4.7 *– Airplane Passenger v2 (Multi-Class) Training Results*

- **Analysis and Evaluation:**

The results were exceptional. The **mAP@0.50** jumped from **0.573 to 0.994**, clearly demonstrating how an accurate problem diagnosis followed by a strategic methodological shift can dramatically boost performance. This experiment laid the foundation for a broader strategy: in tasks involving highly similar subclasses, **merging them into more general and robust categories** may yield more reliable results.

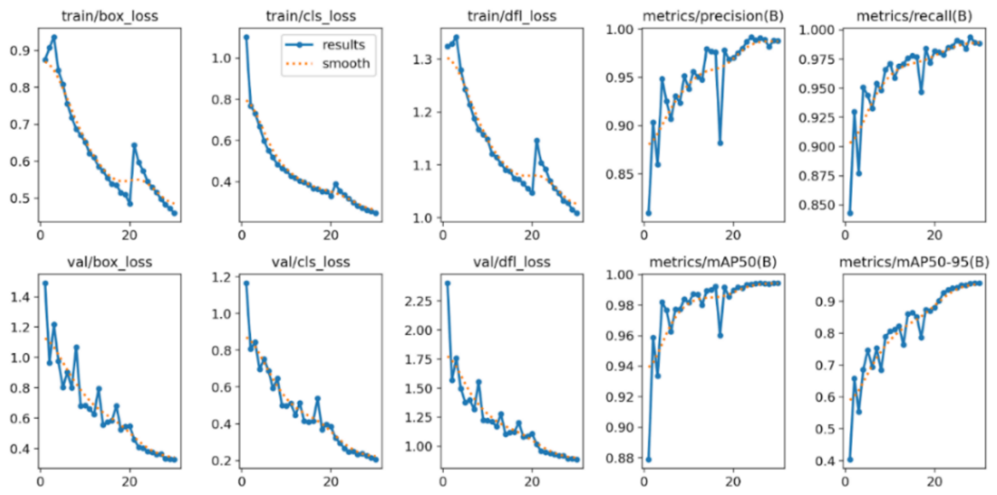


Figure 4.8 *– Airplane Passenger v2: Model Training and Validation Graphs*

4.1.3. Military Aircraft Dataset Results

This experiment focused on managing and improving a complex dataset containing a large number of military aircraft classes.

- **Initial Performance**

At the start of the project, the dataset included **36 distinct military aircraft classes**. The initial training experiments using this dataset yielded unsatisfactory results, with an **mAP@0.50 of only 0.601**.

- **Error Analysis and Confusion Matrix**

To understand the root causes of this low performance, a detailed class-level error analysis was conducted. The primary tool used in this analysis was the **normalized confusion matrix** shown in **Figure 4.5**.

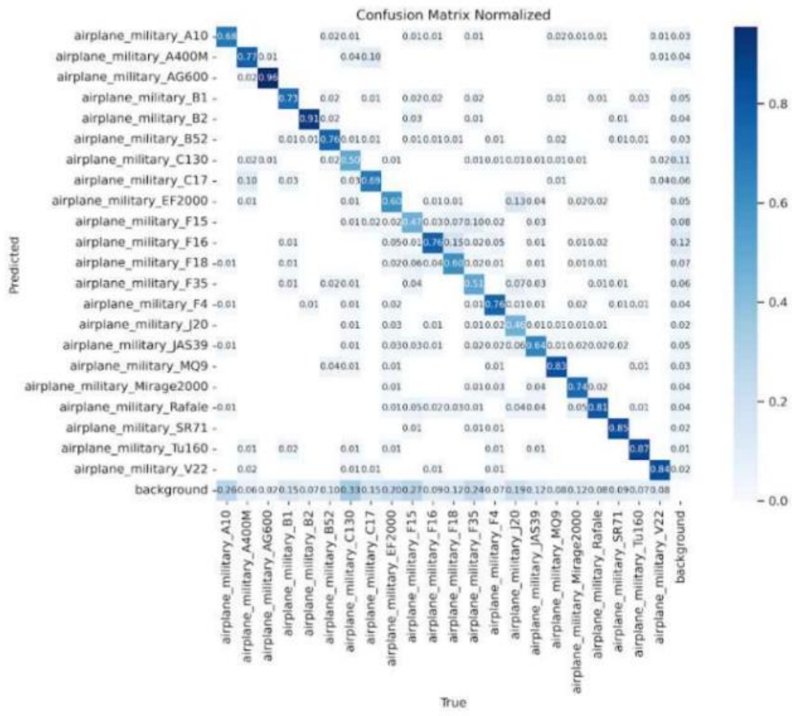


Figure 4.5 *– Confusion Matrix for the Initial 36-Class Military Aircraft Model*

In the matrix, the diagonal cells represent correct predictions for each class, while the brighter off-diagonal cells highlight frequent misclassifications. The analysis revealed that **visually similar classes** or **those with fewer training examples**—such as *F-15*, *F-18*, and *C-130*—were often confused by the model, leading to poor accuracy for those classes.

- **Methodological Improvement and New Results**

Based on the confusion analysis, the dataset was refined by removing classes with the highest misclassification rates and lowest individual accuracy. The **number of classes was reduced from 36 to 22**, and the updated dataset included **9,824 high-quality images**. To handle the increased complexity of the refined dataset, the **YOLOv8m** model—offering greater learning capacity—was used and trained for **75 epochs**.

The results obtained with the restructured dataset are presented in **Table 4.10.**

Metric	Value
Precision	0.772
Recall	0.708
mAP@0.50	0.761
mAP@0.50:0.95	0.683

***Table 4.10** – Military Aircraft (22-Class) Training Results*

- **Evaluation**

As a result of reducing class count and improving dataset quality, the **mAP@0.50 increased from 0.601 to 0.761**. This significant improvement validates the effectiveness of performance-driven data cleaning and class pruning strategies.

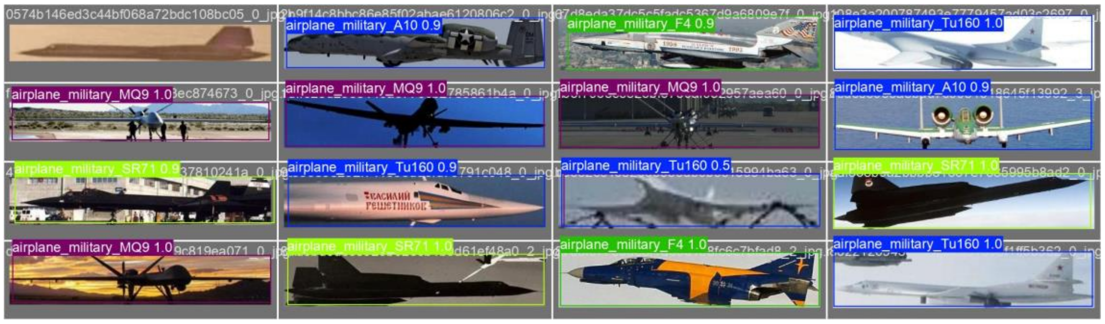

airplane_military_A10 0.9
airplane_military_F4 0.9
airplane_military_Tu160 1.0
airplane_military_MQ9 1.0
airplane_military_MQ9 1.0
airplane_military_MQ9 1.0
airplane_military_A10 0.9
airplane_military_SR71 0.9
airplane_military_Tu160 0.9
airplane_military_Tu160 0.5
airplane_military_SR71 1.0
airplane_military_MQ9 1.0
airplane_military_SR71 1.0
airplane_military_F4 1.0
airplane_military_Tu160 1.0

***Figure 4.11** – Sample Detection Results for the Refined Military Aircraft Dataset*

4.1.4. Civil Helicopter Dataset Results

This experiment aimed to evaluate the performance of the single-class **Civil Helicopter** dataset.

- **Training Details**

As outlined in the methodology, the training set for this dataset was augmented by a factor of **2x** to enhance the model's generalization capability. The final dataset consisted of **1,607 training**, **464 validation**, and approximately **232 test** images. The model used was **YOLOv8s**, trained for **30 epochs**.

- **Performance Metrics**

The model's high validation performance is summarized in **Table 4.12**.

Metric	Value
Precision	0.974
Recall	0.972
mAP@0.50	0.990
mAP@0.50:0.95	0.755

Table 4.12*– Civil Helicopter Dataset Training Results*

- ### Evaluation

The results indicate that the model performed exceptionally well in detecting civil helicopters. The **mAP@0.50 of 0.990** suggests near-perfect alignment between the predicted bounding boxes and ground truth targets. Similar to the UAV dataset, this strong performance confirms both the **quality of the dataset** and the **effectiveness of the selected methodology**, eliminating the need for any additional improvement steps in this category.

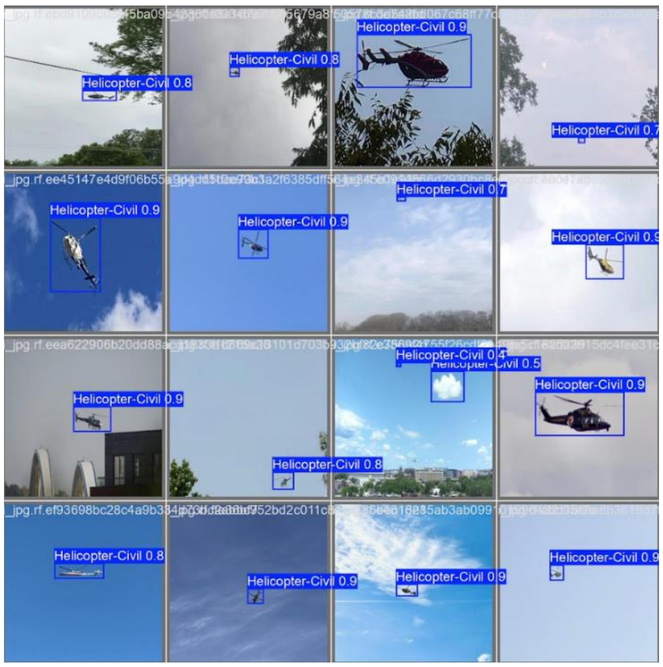


Figure 4.13 *– Civil Helicopter Model Detection Examples*

4.1.5. Military Helicopter Dataset Results

The multi-class **Military Helicopter** dataset underwent a two-phase improvement process to address challenges such as intra-class similarity and varying data quality.

Version 1 (v1): Initial Evaluation

- **Training Details:**
  The first version of the dataset contained **10 classes**, with **5,020 training** and **1,414 validation** images. Given the dataset's complexity, the **YOLOv8m** model was selected and trained for **50 epochs**.

- **Performance Metrics:**
  The overall training results are summarized in **Table 4.14**.

Metric	Value
Precision	0.866
Recall	0.876
mAP@0.50	0.897
mAP@0.50:0.95	0.690

Table 4.14*– Military Helicopter v1 Training Results*

- **Analysis and Evaluation:**

  While the overall **mAP@0.50 of 0.897** was promising, a class-wise breakdown revealed a significant issue: the *Hal Prachand* class had an unusually low **mAP@0.50 of 0.647**. Further inspection showed that the model frequently confused this class with *Mi-24*, resulting in degraded performance for both categories.

Version 2 (v2): Post-Cleaning Evaluation

- **Methodological Adjustment:**
  Based on the v1 findings, a targeted cleaning operation was conducted—especially focused on the *Hal Prachand* class. Low-quality and mislabeled images were removed across the dataset. As a result, the training set was reduced to **4,768** and the validation set to **1,334** images.

- **Training and Performance:**
  The refined dataset was again trained with the **YOLOv8m** model for **50 epochs**. Results are provided in **Table 4.15**.

Metric	**Value**
Precision	0.875
Recall	0.864
mAP@0.50	0.912
mAP@0.50:0.95	0.700

Table 4.7 – Military Helicopter v2 (Cleaned) Training Results

- **Per-Class Performance Overview:**
  A breakdown of the class-wise performance metrics is provided in **Table 4.16**, further justifying refinement decisions.

Class	***Images***	***Instances***	***Precision***	***Recall***	***mAP@0.50***	***mAP@0.50:0.95***
Military_Helicopter_Boeing_Ah_Apache	*196*	*200*	*0.926*	*0.990*	*0.989*	*0.784*
Military_Helicopter_Chinook	*193*	*225*	*0.956*	*0.869*	*0.975*	*0.660*
Military_Helicopter_Gunship	*71*	*91*	*0.727*	*0.857*	*0.897*	*0.649*
Military_Helicopter_Hal_Dhruv	*60*	*60*	*0.931*	*1.000*	*0.977*	*0.924*
Military_Helicopter_Hal_Prachand	*60*	*70*	***0.540***	***1.000***	***0.729***	***0.378***
Military_Helicopter_Hal_Rudra	*52*	*52*	*0.939*	*0.923*	*0.928*	*0.742*
Military_Helicopter_Hal_chetak	*200*	*268*	*0.956*	*0.731*	*0.933*	*0.667*
Military_Helicopter_Mi24	*377*	*394*	*0.966*	*0.624*	*0.835*	*0.686*
Military_Helicopter_Mi26	*49*	*49*	*0.818*	*0.918*	*0.905*	*0.864*
Military_Helicopter_Mi8_17	*76*	*76*	*0.995*	*1.000*	*0.995*	*0.646*

***Table 4.16**– Class-wise Performance Metrics for Military Helicopter Dataset (v2)*

***Note**: The Hal Prachand class is highlighted for its consistent underperformance.*

- **Final Evaluation and Decision**

  Despite moderate overall improvements in Version 2—where **mAP@0.50 increased from 0.897 to 0.912**—the *Hal Prachand* class continued to exhibit significantly lower performance across all metrics, particularly in **mAP@0.50:0.95** (0.378).

This outcome revealed that data cleaning alone was not sufficient for this class, suggesting that its visual features are inherently difficult for the model to distinguish.

To maximize overall model robustness, the *Hal Prachand* class was excluded from the final dataset. The remaining 9 well-performing military helicopter classes were retained and used in the final training phase.

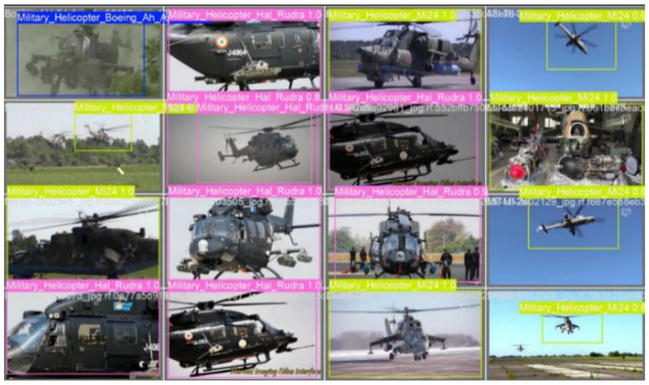


***Figure 4.17**– Sample Detection Results for Military Helicopter v2*

4.2. Final Model Performance

Following the experimental studies and dataset refinements detailed in the previous sections, the final dataset was constructed and used to train the project's ultimate model. This section presents the training details and a comprehensive performance analysis of the final model.

4.2.1. Final Dataset and Training Details

The final dataset was created by combining the cleanest and highest-performing datasets from all previous experiments. As a final optimization step, the **military**

aircraft category was revisited and further refined—classes with low performance and high confusion rates (e.g., *F-15*) were removed, reducing the number of military aircraft classes to **15**.

This resulted in a dataset with **27 unique classes**, including:

- **0:** UAV
- **1:** airplane_passenger
- **2:** helicopter_civil
- **3–11:** military_helicopter
- **12–26:** airplane_military

The dataset contained a total of **30,427 images**, split into:

- **24,341 training**
- **3,043 validation**
- **3,043 test**

The final model was trained using **YOLOv8m**, the most powerful model utilized in the project, for **100 epochs**.

4.2.2. Overall Performance Metrics

After 100 epochs, the model's overall performance on the test set is summarized in **Table 4.18**.

Metric	**Value**
Precision	0.932
Recall	0.902
mAP@0.50	0.939
mAP@0.50:0.95	0.834

***Table 4.18**– Final Model Overall Performance Metrics*

These results demonstrate that the final model performed **very well** in this complex multi-class task, achieving high accuracy across all key metrics.

4.2.3. Class-Wise Performance Analysis

In addition to overall metrics, understanding how the model performs across individual classes is critical to identifying strengths and weaknesses. **Table 4.19** provides a breakdown of precision, recall, and mAP scores for all 27 classes in the test set.

Class	Precision	Recall	mAP@0.50	mAP@0.50:0.95
UAV	0.996	0.994	0.994	0.857
airplane_passenger	0.986	0.985	0.992	0.939
Helicopter-Civil	1.000	0.993	0.995	0.895
Military_Helicopter_..._Apache	0.997	1.000	0.995	0.797
Military_Helicopter_Chinook	0.992	1.000	0.995	0.769
Military_Helicopter_Gunship	0.975	0.899	0.967	0.778
Military_Helicopter_..._Dhruv	0.984	0.989	0.990	0.917
Military_Helicopter_..._Rudra	0.951	0.950	0.982	0.821
Military_Helicopter_..._chetak	0.937	0.924	0.960	0.724
Military_Helicopter_Mi24	0.941	0.939	0.959	0.877
Military_Helicopter_Mi26	0.994	1.000	0.995	0.963
Military_Helicopter_Mi8_17	1.000	0.990	0.995	0.712
airplane_military_A10	0.882	0.859	0.871	0.783
airplane_military_A400M	0.857	0.924	0.973	0.936
airplane_military_AG600	0.992	1.000	0.995	0.979
airplane_military_B1	0.925	0.864	0.938	0.858
airplane_military_B2	0.970	0.852	0.912	0.859
airplane_military_C17	0.868	0.615	0.758	0.638
airplane_military_F16	0.781	0.902	0.898	0.816
airplane_military_F4	0.876	0.753	0.910	0.870
airplane_military_JAS39	0.916	0.725	0.823	0.764
airplane_military_MQ9	0.892	0.856	0.860	0.784
airplane_military_Mirage2000	0.867	0.783	0.876	0.801

Class	Precision	Recall	mAP@0.50	mAP@0.50:0.95
airplane_military_Rafale	0.943	0.823	0.917	0.868
airplane_military_SR71	0.820	0.913	0.899	0.819
airplane_military_Tu160	0.902	1.000	0.994	0.951
airplane_military_V22	0.923	0.815	0.916	0.734

Table 4.19 *– Class-Wise Performance of the Final Model*

Analysis

As seen in the table, general-purpose and visually distinct classes such as **UAV**, **airplane_passenger**, and **Helicopter-Civil** achieved near-perfect detection results with **mAP@0.50 values above 0.99**. Military vehicles with unique visual features (e.g., *AG600*, *Mi-26*, *Chinook*) also demonstrated excellent performance.

In contrast, classes like **airplane_military_C17** and **airplane_military_JAS39** showed relatively lower performance, likely due to a limited number of training samples or high visual similarity with other military aircraft

4.2.4. Qualitative Evaluation and Visual Results

In addition to quantitative metrics, visual assessment is essential for understanding the model's behavior and limitations.

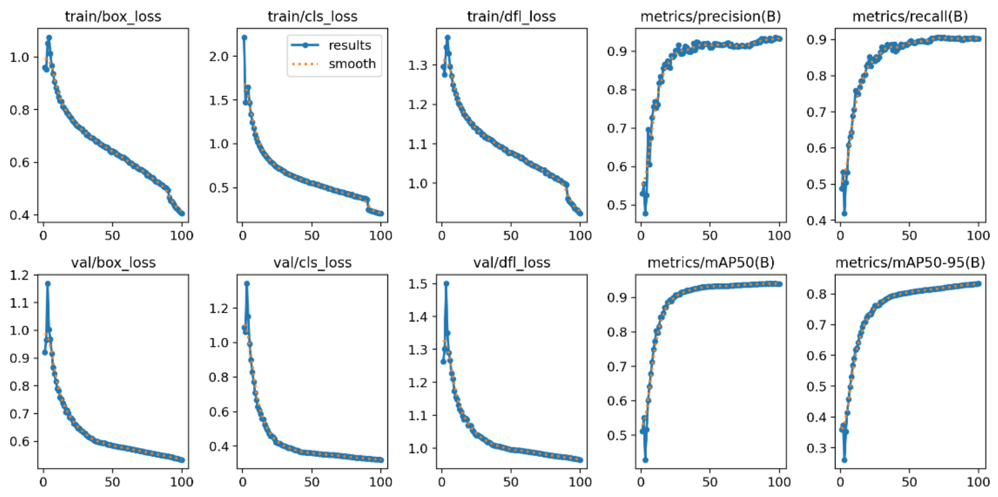


Figure 4.20*– Final Dataset Model Training and Validation Graphs*

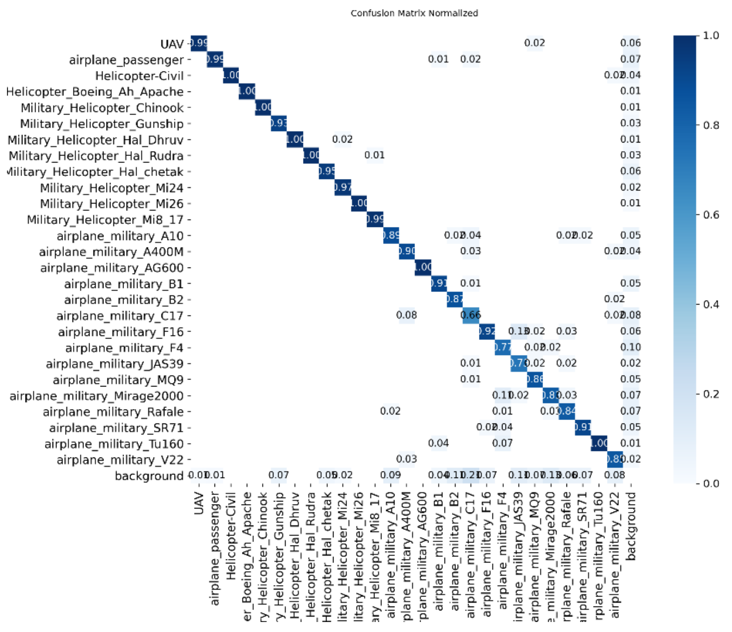


Figure 4.21– Confusion Matrix on the Final Test Set

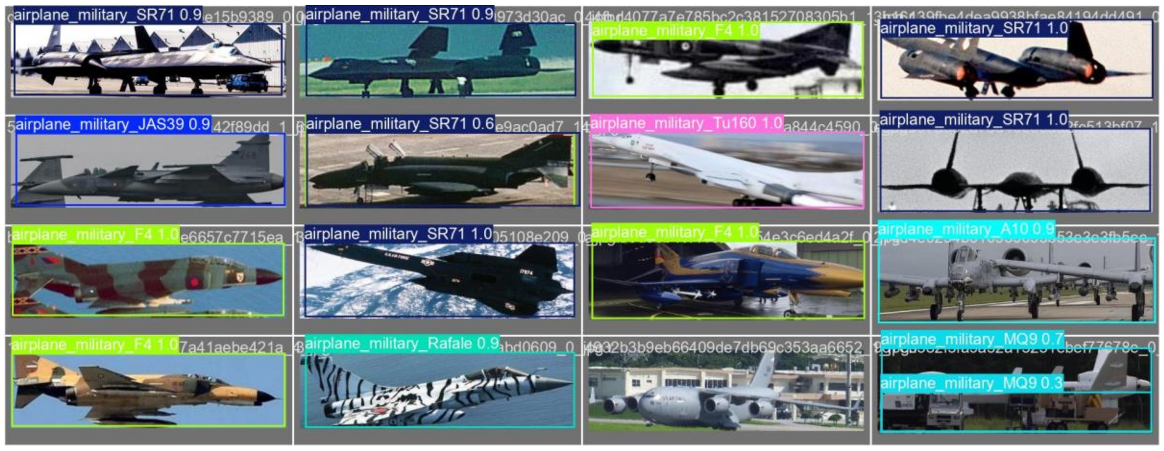

airplane_military_SR71 0.9
airplane_military_SR71 0.9
airplane_military_F4 1.0
airplane_military_SR71 1.0
airplane_military_JAS39 0.9
airplane_military_SR71 0.6
airplane_military_Tu160 1.0
airplane_military_SR71 1.0
airplane_military_F4 1.0
airplane_military_SR71 1.0
airplane_military_F4 1.0
airplane_military_A10 0.9
airplane_military_F4 1.0
airplane_military_Rafale 0.9
airplane_military_MQ9 0.7
airplane_military_MQ9 0.3

Figure 4.22– Example Detection Results of the Final Model

5.User Interface and Application Workflow

5.1. Purpose and Overview

One of the primary goals of this project is not only to achieve theoretical success with deep learning models but also to deliver a **practical and usable output**. To fulfill this goal, a **user-friendly web interface** was developed to visually demonstrate the model's detection capabilities.

The backend of the application is implemented using **Flask**, a lightweight Python-based web framework. The detection engine is powered by the **YOLOv8m model**, trained on the final dataset as detailed in previous sections.

5.2. Interface Features and Workflow

The developed interface, titled **“AirAI-Vision”**, features a clean and minimalist design that allows users to easily test the model. The main interface consists of a file upload section and an output display area. Key functionalities include:

- **Supported File Types**: Users can upload .jpg, .png image files or .mp4 video files.
- **Detection Engine**: All detection tasks are processed using the YOLOv8 model running in the backend.
- **Result Visualization**: For each detected object, the **class name** and **confidence score** are clearly displayed.

- **Application Workflow:**

  1. **File Upload**
     The user clicks the **“Choose File”** button to select an image or video from their local system.
  2. **Analysis**
     After selecting a file, clicking the **“Analyze”** button triggers the object detection process.
  3. **Result Presentation**

  - **For Image Files**:
    The output includes the original image with **bounding boxes** drawn around detected objects. Each box is labeled with the predicted class name and the model’s confidence score. Figure 5.1 presents an example detection result for the class airplane_military_MQ.

***Figure 5.1** – Example Image Detection Output from the Web Interface*

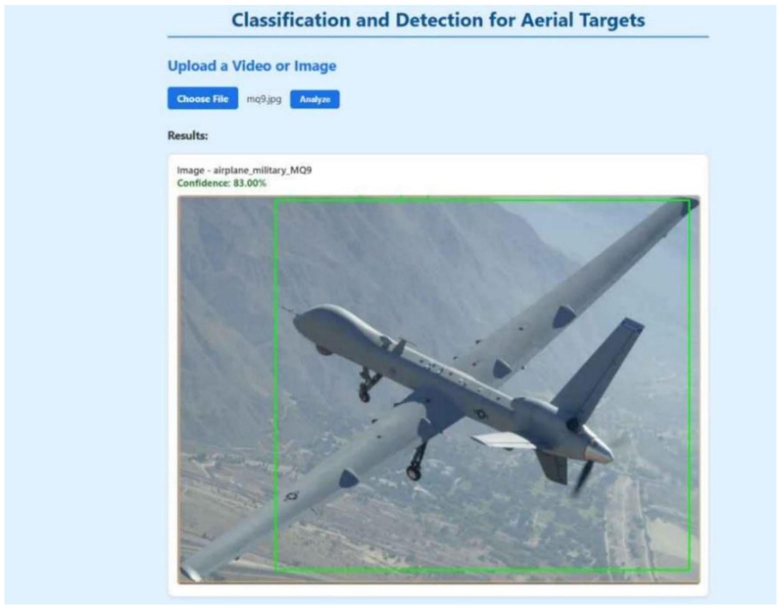


Image - airplane_military_MQ9
Confidence: 83.00%

- **For Video Files:**
  The system analyzes the video frame by frame, extracting detection results across the timeline. The results are displayed as a summary table, listing the detected objects, corresponding timestamps, and confidence scores. This allows the user to quickly focus on relevant moments without watching the entire video.

This interface translates the project's technical capabilities into a format that is both **accessible and interpretable for end-users**, serving as the tangible output of the entire system.

6.Conclusion

This study aimed to develop and evaluate a deep learning–based system for the real-time detection and classification of various aerial vehicles. With the growing need for autonomous systems in security and surveillance applications, the importance of such detection frameworks is rapidly increasing. The primary goal of this project was to build a model capable of accurately identifying a wide range of aerial targets—including UAVs, civilian/military aircraft, and helicopters—by leveraging the modern object detection architecture YOLOv8.

To achieve this objective, a data-centric and iterative methodology was adopted. Raw visual data collected from diverse sources such as Roboflow and YouTube were not simply aggregated but were thoroughly analyzed through separate experimental training sessions. These analyses led to key data refinement decisions, such as merging overlapping classes in the Airplane Passenger dataset and eliminating underperforming or mislabeled classes in the Military datasets. The final dataset, comprising **27 unique classes**, was used to train the **YOLOv8m** model for **100 epochs**. Additionally, a Flask-based web interface was developed to demonstrate the practical application of the trained model.

The experimental results validated the effectiveness of the methodology. For instance, merging the visually similar Airbus and Boeing classes into a single airplane_passenger class improved the mAP@0.50 from **0.573 to 0.994**, clearly highlighting the importance of data strategy. As a result of all refinement steps, the final model achieved **0.939 mAP@0.50** and **0.834 mAP@0.50:0.95** on the test set across 27 classes. Particularly strong performance was observed in visually distinct classes such as **UAV** and **Helicopter-Civil**, where detection was nearly flawless.

In conclusion, this project has demonstrated the power and flexibility of the YOLOv8 architecture when paired with a carefully curated and iteratively improved dataset for aerial object detection. Beyond producing a high-performance deep learning model, the project represents a comprehensive pipeline—from data acquisition to end-user deployment—bridging theoretical modeling with real-world application.

7. Future Work

This project has established a solid foundation for the detection and classification of aerial vehicles using deep learning. While the results achieved are promising, there are several directions in which the system can be expanded and enhanced. The following outlines potential improvements for future work:

7.1. Data and Model Enhancements

- **Dataset Expansion:**
  Collecting more diverse and representative images—especially for underperforming classes such as *airplane_military_C17*—can significantly improve class-level accuracy and model generalization.
- **Incorporation of Challenging Conditions:**
  Including images captured under diverse environmental conditions (e.g., night-time, fog, rain, different camera angles) would increase the model's robustness in real-world scenarios.
- **Exploration of Newer Architectures:**
  As technology evolves rapidly, future experiments may include testing newer architectures such as **YOLOv9** or other state-of-the-art detection models to potentially achieve even higher performance.

7.2. Application and Feature Improvements

- **Live Camera Support:**
  Integrating real-time webcam support into the web interface would significantly increase the application's practicality and demonstrate live detection capabilities.
- **Object Tracking Integration:**
  The current system performs detection on individual frames independently. A natural next step is to implement **object tracking algorithms** (e.g., BoT-SORT, ByteTrack) to assign unique IDs and follow aerial objects across frames in video streams.
- **Cloud Deployment:**
  Deploying the Flask application on a cloud platform such as **AWS**, **Google Cloud**, or **Heroku** would allow public access and remote demonstration of the system.

7.3. Performance Optimization

- **Model Optimization and Edge Deployment:**
  Converting and optimizing the final YOLOv8 model using tools like **TensorRT** or **ONNX** could enable its deployment on **resource-constrained edge devices** or

embedded systems, expanding its usability in real-time, low-latency environments.

These future directions aim not only to enhance the technical performance of the system but also to increase its accessibility, scalability, and applicability in real-world operational contexts.

8. References

[1] Wei, X., Li, Z., & Wang, Y. (2025). *SED-YOLO based multi-scale attention for small object detection in remote sensing*. *Scientific Reports, 15*, 3125.

[2] Zhang, H., Xiao, P., Yao, F., & Gong, Y. (2025). *Fusion of multi-scale attention for aerial images small-target detection model based on PARE-YOLO*. *Scientific Reports, 15*, 4753.

[3] Zhang, H., Sun, W., Sun, C., He, R., & Zhang, Y. (2024). Hsp-yolov8: Uav aerial photography small target detection algorithm. *Drones*, *8*(9), 453.

[4] Wang, G., Chen, Y., An, P., Hong, H., Hu, J., & Huang, T. (2023). UAV-YOLOv8: A small-object-detection model based on improved YOLOv8 for UAV aerial photography scenarios. *Sensors*, *23*(16), 7190.

[5] Fang, A., Feng, S., Liang, B., & Jiang, J. (2024). Real-time detection of unauthorized unmanned aerial vehicles using SEB-YOLOv8s. *Sensors*, *24*(12), 3915.

[6] Shi, H., Yang, W., Chen, D., & Wang, M. (2024). ASG-YOLOv5: Improved YOLOv5 unmanned aerial vehicle remote sensing aerial images scenario for small object detection based on attention and spatial gating. *Plos one*, *19*(6), e0298698.

[7] Zhou, J., Su, T., Li, K., & Dai, J. (2024). *Small Target–YOLOv5: Enhancing the algorithm for small object detection in drone aerial imagery based on YOLOv5*. Sensors, 24(1), 134.

[8] Military Helicopter Detection https://universe.roboflow.com/shruti-bxzas/military-aircraft-detection-irqxh