PROJECT REPORT ON

REAL-TIME DRONE DETECTION USING YOLO ARCHITECTURES

Conducted at

Indian Institute of Technology, Tirupati (Online)

GUIDE:

Prof. Ramakrishna Gorthi

SUBMITTED BY:

Buddhana Pallavi (623106)
B.Tech – Electronics and Communication Engineering
National Institute of Technology Andhra Pradesh

DURATION:

May 15 – July 20, 2025

DECLARATION

I hereby declare that the project report titled “Real-time Drone Detection using YOLO architectures” is a genuine record of work carried out by me during the internship at IIT Tirupati. The results presented in this report are based on outcomes obtained through training, evaluation, and analysis of object detection models.

Some content has been referred from research papers and online sources, and all such references have been duly cited. I affirm that this report adheres to academic integrity.

Buddhana Pallavi

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Prof. Ramakrishna Gorthi, IIT Tirupati, for providing me this valuable opportunity to work on a real-time research project. His continuous support, insightful guidance, and constructive feedback throughout the internship have been instrumental in shaping my learning experience. His expertise in Machine Learning, Deep Learning, and Computer Vision greatly enriched the technical depth of this project.

I would also like to extend my heartfelt thanks to my senior, Nakka Vishnu Vardhan, for referring me to this internship. His encouragement and timely recommendation made this entire experience possible, and I am truly grateful for his mentorship and support.

Buddhana Pallavi

ABSTRACT

The increasing use of unmanned aerial vehicles (UAVs) in both civil and commercial sectors brings with it the critical challenge of preventing their unauthorized use in sensitive airspaces. This project addresses the urgent need for accurate and real-time drone detection systems by developing a deep learning pipeline using state-of-the-art YOLO (You Only Look Once) architectures.

The project focuses on building an object detection system capable of detecting, classifying, and localizing drones in both visible and infrared imagery. Leveraging the Ultralytics YOLOv8m and YOLOv10n models, four versions of detection systems were trained and evaluated using three different public datasets: the Anti-UAV dataset (200,000+ images), DUT Anti-UAV (10,000 images), and Kaggle Drone Detection dataset (4,010 images). Custom Python scripts were employed to convert all annotations into YOLO format, and model training was conducted using the Ultralytics PyTorch framework on Google Colab.

Each model's performance was analyzed using standard object detection metrics including Precision, Recall, mAP@0.5, mAP@0.5:0.95, and Frames Per Second (FPS) to assess the trade-off between accuracy and speed. The YOLOv10n model achieved significantly higher FPS, proving its suitability for real-time applications, while YOLOv8m outperformed others in terms of accuracy and reliability, especially for small object detection.

This project provides an end-to-end implementation and comparative study of modern YOLO models for UAV detection, offering insights into deployment choices for surveillance systems, defense infrastructure, and autonomous monitoring platforms. It highlights the potential of AI-powered vision systems in enhancing airspace security and sets the stage for future advancements in multi-drone tracking and thermal vision-based UAV monitoring.

CONTENTS

1.BACKGROUND

In recent years, civil unmanned aerial vehicles (UAVs), commonly known as drones, have become increasingly prominent due to their autonomy, low cost, and ease of deployment. They are widely utilized in various domains including search and rescue operations, logistics and delivery systems, environmental monitoring, and remote sensing. While their benefits are numerous, the growing ubiquity of drones has raised serious concerns about public safety, national security, and airspace management.

Unauthorized UAVs can pose threats in the form of physical attacks (e.g., explosives), cyber intrusions, or violations of aviation safety regulations. Notably, there have been real-world incidents where drone sightings led to flight disruptions and significant financial losses at major airports. The challenge in detecting drones stems from their small size, low altitude, and irregular flight patterns, which makes them difficult to detect using traditional systems such as radar.

To address these limitations, computer vision and deep learning-based approaches are increasingly being explored for automated drone detection and tracking. However, many traditional approaches have lacked comprehensive and realistic datasets, limiting their real-world applicability.

To overcome this gap, initiatives like the Anti-UAV Challenge & Workshop (CVPR 2020, ICCV 2021, CVPR 2023) have introduced high-quality benchmark datasets consisting of 410 diverse video sequences. These sequences feature UAVs of varying sizes, backgrounds, motion trajectories, and include both visible and infrared frames — offering a robust platform for research and development in UAV detection. The task of detecting small, fast-moving drones in complex environments, often against occlusions or distractors such as birds, clouds, and buildings, presents an open and critical problem in vision-based surveillance.

This internship project contributes to this domain by developing a deep learning pipeline for UAV detection using YOLO-based architectures, aimed at building fast and accurate vision models capable of real-time drone localization and classification.

2. OBJECTIVE

The main objective of this project is to design, develop, and evaluate a real-time drone detection system using deep learning techniques, with a focus on the YOLO (You Only Look Once) family of object detection algorithms. The project aims to:

- Detect and classify UAVs from aerial imagery, including both visible and thermal modalities.
- Extract bounding box coordinates of drones for accurate localization in the frame.
- Compare multiple YOLO variants (YOLOv8m, YOLOv10n) across multiple datasets in terms of accuracy (mAP@0.5, mAP@0.5:0.95), precision, recall, and inference speed (FPS).
- Evaluate model performance on three diverse datasets to understand the robustness of trained models under various environmental and dataset conditions.
- Analyze model behavior on small-object detection, complex motion patterns, and real-world video challenges.

The final system should provide a metrics-driven analysis to aid the selection of an appropriate model depending on the real-time or accuracy-sensitive application context.

3. DATASET

The datasets used in this project are publicly available drone detection datasets containing aerial surveillance imagery in both Visible and Infrared (IR) modalities. These datasets include a variety of environmental conditions, drone sizes, camera angles, and image resolutions. The primary goal was to train and evaluate object detection models that can accurately and efficiently detect drones in real-time scenarios.

3.1. IMAGE DATA

Image data was extracted from multiple sources, including:

- Anti-UAV Dataset (Anti-UAV Dataset): Contains drone images in both Visible and IR modes, captured under diverse conditions. A total of 223 video sequences are available in the training set.
- DUT Anti-UAV Dataset (https://wangdongdut/DUT-Anti-UAV): Hosted on GitHub, this dataset provides visible imagery with drone annotations.
- Kaggle Drone-YOLO Dataset (Kaggle Link): Contains YOLO-formatted labels and drone images for faster prototyping.

To reduce computational overhead, a subset of 6 sequences (1000–1500 frames each) from the Anti-UAV dataset was selected and annotated for custom training. The images and annotations were originally in .json format and were converted into YOLO format using a Python conversion script.

3.2. ANNOTATION FORMAT AND PREPROCESSING

The annotations included bounding boxes and class labels. Preprocessing steps involved:

- Conversion of annotations from JSON to YOLO format (normalized [class x_center y_center width height]).
- Image resizing to 640×640 pixels to match the YOLO model's input format.
- Class balancing across drone sizes (small, medium, large) for fair evaluation.
- Data validation to remove corrupted images or frames without drones

3.3. SIZE-WISE CATEGORIZATION

Drones were categorized into three size types based on bounding box area:

- Small Drones
- Medium Drones
- Large Drones

This categorization was critical for size-wise performance analysis of each model.

3.4. TRAIN-VALIDATION-TEST SPLIT

For each model, the dataset was split into training, validation, and test sets:

- YOLOv8m was trained on subsets from the Anti-UAV, DUT, and Kaggle datasets.
- YOLOv10n was trained on the same Anti-UAV subset for consistency.

◆ **Model 1 & Model 2**

- Source: Anti-UAV
- Split: 80% Train, 10% Validation, 10% Test
- Includes 6 sequences × ~1000–1500 frames
- Annotation format: Converted from JSON → YOLO
- Drone Size Distribution: Mostly small to medium

◆ **Model 3**

- Source: DUT Anti-UAV GitHub dataset
- YOLOv8m trained from scratch
- Dataset includes annotated drone footage in visible light
- Used for further generalization performance

◆ **Model 4**

- Source: Kaggle Drone YOLO Dataset
- Already YOLO-formatted labels
- YOLOv8m tested for adaptability to diverse data

4. MACHINE LEARNING CONCEPTS USED

In the context of drone detection, Machine Learning plays a crucial role in identifying and localizing drones in real-time imagery. Instead of traditional ML algorithms like SVM or Decision Trees, this project directly implements advanced deep learning-based object detection models that inherently learn classification, localization, and bounding box regression.

These models are built upon core machine learning concepts like:

4.1. OBJECT DETECTION AS A MACHINE LEARNING PROBLEM

- Object detection involves two sub-problems:
  - Classification: Identifying whether an object (like a drone) exists.
  - Localization: Drawing a bounding box around the object.
- YOLO (You Only Look Once) models treat detection as a single regression problem, predicting both class probabilities and bounding box coordinates.

4.2. ROLE OF SUPERVISED LEARNING

- The YOLO models are trained on labeled datasets, where each image has bounding boxes annotated with class labels (i.e., drone).
- This is a form of supervised learning, where the model learns to map input images to target outputs.

4.3. CONFIDENCE SCORE AND THRESHOLDING

- Each detection includes a confidence score (probability of being a drone).
- The model uses Non-Maximum Suppression (NMS) to eliminate overlapping boxes and refine results.

4.4. WHY DEEP LEARNING OVER TRADITIONAL ML?

Feature	**Traditional ML (e.g., SVM, RF)**	**Deep Learning (e.g., YOLO)**
Requires manual feature extraction	Yes	No
Works with raw image data	No	Yes
Real-time detection	Rarely	Yes

4.5. TRANSFER LEARNING AND PRETRAINED MODELS

- This project leverages pretrained YOLOv8 and YOLOv10 models trained on large datasets like COCO.
- These models are fine-tuned on drone-specific datasets (e.g., Anti-UAV, DUT, Kaggle) for better performance in the target domain.

5. DEEP LEARNING FRAMEWORK FOR DRONE DETECTION

Deep Learning, a specialized subfield of Machine Learning, has revolutionized computer vision tasks such as object detection and classification. In this project, advanced deep learning models based on the YOLO (You Only Look Once) family have been used to detect drones in real-time from both visible and infrared imagery. These models are built on top of Convolutional Neural Networks (CNNs) and optimized for accuracy, speed, and compact deployment. Unlike traditional Machine Learning algorithms that require manual feature extraction, Deep Learning models automatically learn hierarchical feature representations from raw image data. This section outlines the key deep learning components relevant to this project.

5.1. ARTIFICIAL NEURAL NETWORKS (ANNs)

Artificial Neural Networks are computing systems inspired by the structure and function of biological neural networks. ANNs consist of interconnected layers of "neurons" that transform input data using weighted connections and activation functions.

In the context of object detection, ANN concepts are used in the final detection head of models like YOLO, where fully connected (dense) layers process high-level features to output bounding box coordinates, class probabilities, and objectness scores.

Key concepts:

- Layers: Input, hidden, output
- Activation functions: ReLU, Sigmoid, Softmax
- Loss functions: Binary Cross Entropy, IoU loss, Confidence loss (used in YOLO)

5.2. CONVOLUTIONAL NEURAL NETWORKS (CNNs)

CNNs are the backbone of modern computer vision models, including all YOLO architectures. CNNs use a series of convolutional layers with filters that scan the image and extract spatial features like edges, textures, shapes, and patterns.

Core components of CNNs:

- Convolutional layers: Learn spatial hierarchies via filters
- Pooling layers: Reduce spatial dimensions and computation
- Batch normalization & dropout: Improve training and generalization
- Fully connected layers: Convert features into prediction scores

In YOLO models, CNNs help detect drones in various conditions such as different sizes, lighting, and angles — without requiring handcrafted features.

5.3. YOLO ARCHITECTURE OVERVIEW

The YOLO (You Only Look Once) algorithm is a real-time object detection system that processes the entire image in a single forward pass. Unlike traditional two-stage detectors (e.g., Faster R-CNN), YOLO treats detection as a regression problem, simultaneously predicting bounding box coordinates, class probabilities, and confidence scores.

Key aspects of YOLO:

- Single-stage detection: Faster than multi-stage models
- Grid-based prediction: The image is divided into an S×S grid. Each cell predicts multiple bounding boxes with scores.
- Anchor boxes: Predefined bounding box shapes for detecting objects of various sizes
- Non-Maximum Suppression (NMS): Removes overlapping detections

YOLO's unified architecture enables real-time drone detection with high accuracy and low latency — crucial for real-world applications like surveillance and autonomous tracking.

5.4. YOLOV8 VS YOLOV10: MODEL COMPARISION

In this project, two state-of-the-art YOLO models were explored:

- YOLOv8m: A mid-sized model with high accuracy, suitable for accuracy-sensitive applications.
- YOLOv10n: A nano model optimized for speed and edge deployment, ideal for real-time processing with limited resources.

Model	Parameter Count	Speed (FPS)	Accuracy (mAP@0.5)	Use Case
YOLOv8m	~25M	Medium (~50-60)	High	Accuracy-first Detection
YOLOv10n	~2.2M	Very High (~140)	Lower than YOLOv8m	Speed-critical systems

This analysis helped in identifying the right model for various drone detection scenarios, balancing performance trade-offs.

5.5. TRANSFER LEARNING AND FINE TUNING

Instead of training YOLO models from scratch (which is computationally expensive), this project uses transfer learning — starting from pretrained weights trained on the COCO dataset.

Steps followed:

- Pretrained weights (.pt files) were loaded using the Ultralytics YOLOv8/YOLOv10 framework.
- Custom drone datasets (Anti-UAV, DUT, Kaggle) were used for fine-tuning, adapting the model to specific aerial scenarios.
- Training was conducted on Google Colab using GPU acceleration, with custom scripts for data formatting (e.g., converting JSON to YOLO format).

Benefits of transfer learning:

- Requires less data and compute power
- Faster convergence
- Improved generalization to unseen drone types

6. RESULTS

This section presents the evaluation metrics of the drone detection models on the test set, including performance across different drone sizes and datasets. The key performance indicators include Precision, Recall, mAP@0.5, and Frames Per Second (FPS) during inference.

6.1. MODEL OVERVIEW

Model	Architecture	Dataset type	Annotation type	FPS
Model-1	YOLOv8m	IR	Bounding Boxes	58.82
Model-2	YOLOv10n	IR	Bounding Boxes	145
Model-3	YOLOv8m	Visible	Bounding Boxes	48.78
Model-4	YOLOv8m	Visible	Bounding Boxes	68.97

6.2. EVALUATION ON TEST SET (MODEL-1)

Size	TP	FP	FN	Precision	Recall	mAP@0.5
Small	751	13	182	0.983	0.805	0.791
Medium	204	14	9	0.936	0.958	0.896
Large	0	0	0	0	0	0

6.3. EVALUATION ON TEST SET (MODEL-2)

Size	TP	FP	FN	Precision	Recall	mAP@0.5
Small	286	18	596	0.941	0.324	0.305
Medium	65	13	143	0.833	0.312	0.26
Large	0	0	0	0	0	0

6.4. EVALUATION ON TEST SET (MODEL-3)

Size-wise metrics (TP, FP, FN, Precision, Recall, mAP@0.5) are not reported due to inconsistencies in IoU-based matching. Although the model performed well overall (mAP@0.5: 0.919), the size-specific matching script failed due to minor shifts in annotations vs. predictions.

6.5. EVALUATION ON TEST SET (MODEL-4)

Size	TP	FP	FN	Precision	Recall	mAP@0.5
Small	20	10	3	0.667	0.87	0.58
Medium	192	26	10	0.881	0.95	0.837
Large	115	19	5	0.858	0.958	0.822

Model	Training metrics	Evaluation metrics	FPS
Model-1	P-0.956 R-0.872 M1-0.926 M2-0.369	P-0.964 R-0.735 M1-0.827 M2-0.531	58.82
Model-2	P-0.896 R-0.902 M1-0.92 M2-0.392	P-0.565 R-0.538 M1-0.573 M2-0.343	145
Model-3	P-0.923 R-0.776 M1-0.864 M2-0.549	P-0.929 R-0.773 M1-0.873 M2-0.612	48.78
Model-4	P-0.905 R-0.906 M1-0.944 M2-0.53	P-0.840 R-0.925 M1-0.911 M2-0.523	68.97

NOTE: P-Precision, R-Recall, M1-mAP@0.5, M2-mAP@0.5-0.95

7. OBSERVATIONS

7.1. ACCURACY VS SPEED TRADEOFF

- Model-2 (YOLOv10n) achieved the highest FPS (~90–145), suitable for real-time tasks, but had the lowest precision, recall, and mAP, making it unsuitable for high-accuracy UAV detection.
- Model-1 balanced performance and speed with moderate FPS (58) and strong accuracy, especially for small and medium drones.

7.2. DATASET INFLUENCE

- Model-4, trained on a smaller Kaggle visual dataset, achieved high generalization across all sizes, with excellent recall (0.925) and overall mAP@0.5 of 0.911, outperforming expectations despite dataset size.
- Model-3, trained on the DUT-Anti-UAV dataset, recorded the highest mAP@0.5 (0.919), although size-wise metrics were unreliable due to IoU mismatches.

7.3. TRAINING VS EVALUATION CONSISTENCY

- Model-2 showed a significant drop from training mAP (0.920) to evaluation mAP (0.573), suggesting overfitting or annotation mismatch.
- Model-4 had minimal drop between training (0.944) and evaluation (0.911), indicating excellent training consistency and real-world adaptability.

7.4. SIZE-WISE DETECTION

- Model-1 performed best on medium-sized drones, with mAP@0.5 = 0.896 and Recall = 0.958.
- Model-4 was strongest across all sizes, especially medium (0.837) and large (0.822) drones, confirming its robustness.
- Model-2 underperformed in all sizes, especially small drones (Recall = 0.324), resulting in high false negatives.

8. INFERENCE

From the evaluation of all four models based on architecture, dataset modality, FPS, and detection performance across drone sizes, the following conclusions can be drawn:

- Model-1 demonstrated strong performance on IR images, especially for small and medium drones, with high precision and recall. Its balance of accuracy and speed (58 FPS) makes it suitable for moderately real-time infrared surveillance applications.
- Model-2, while extremely fast (estimated >90 FPS), significantly underperformed in accuracy, particularly in recall and mAP, suggesting it may not be reliable in applications where missing drones can be critical. It might be considered only for systems where speed is prioritized over detection quality.
- Model-3 achieved the highest overall mAP@0.5 (0.919) using a clean visual dataset. Although size-specific metrics could not be reported due to annotation shifts, the strong overall performance and high FPS (61) make it a solid choice for high-accuracy drone detection in visual domains.
- Model-4 emerged as the most balanced model, showing consistently high detection rates across all drone sizes, especially medium and large, while still maintaining decent speed (45 FPS). It also had a minimal gap between training and evaluation performance, indicating excellent generalization.

9. TOOLS & TECHNOLOGIES USED

9.1. LANGUAGES

◆ PYTHON

Used extensively for all stages of the project, including data preprocessing, model training, evaluation, visualization, and automation scripts. Python's simplicity and versatility made it ideal for integrating various libraries and frameworks.

9.2. LIBRARIES & FRAMEWORKS

◆ PYTORCH

Utilized for deep learning model implementation, especially for understanding neural network structures and fine-tuning YOLO models in the backend.

◆ OPENCV

Used for image preprocessing, visualization, and frame-level operations during dataset exploration and post-inference image rendering.

◆ ULTRALYTICS YOLO

A high-level, production-ready object detection framework used for training, evaluating, and inferring YOLO models. Enabled simplified implementation of state-of-the-art models like YOLOv8m and YOLOv10n.

◆ MATPLOTLIB & SEABORN

For plotting model performance metrics, confusion matrices, and FPS vs accuracy visualizations.

◆ PANDAS & NUMPY

Used for dataset manipulation, annotation summary generation, and metric computation.

9.3. DEVELOPMENT PLATFORMS

◆ GOOGLE COLAB

The primary cloud-based development environment used for training and evaluating models. Offered free GPU access (with runtime limitations), making it suitable for running computationally intensive deep learning tasks.

◆ JUPYTER NOTEBOOK

Used locally for debugging small-scale modules, visual analysis, and testing preprocessed annotations and inference snippets.

◆ **PYCHARM**

IDE used for scripting data preprocessing tasks such as converting JSON annotations to YOLO format, organizing dataset directories, and writing model automation scripts.

9.4. DOCUMENTATION, TRACKING & ANALYSIS

◆ **MICROSOFT EXCEL**

Extensively used to track and compare model-wise training and evaluation metrics such as precision, recall, mAP@0.5, mAP@0.5:0.95, and FPS.
Also used to document dataset distribution (Visible/IR, size-wise analysis), and prepare size-wise precision-recall summaries.

9.5. CLOUD & STORAGE

◆ **GOOGLE DRIVE**

Used for storing datasets, annotation files, trained weights, and logs. Enabled seamless access across Colab and local environments.

10. CHALLENGES FACED

During the course of this project, several technical and practical challenges were encountered and tackled:

◆ **GPU LIMITATIONS IN GOOGLE COLAB**

Due to limited access to high-performance GPUs in Google Colab, long training sessions were frequently interrupted. This required careful checkpoint saving, reduced batch sizes, and optimization of training cycles to avoid disconnections and resource timeouts

◆ **RUNTIME DISCONNECTIONS DURING TRAINING**

Training YOLO models, especially on larger datasets, led to runtime disconnections. This was particularly challenging when evaluating multiple models across various datasets. To mitigate this, model weights were saved periodically and evaluation was performed in smaller batches.

🔹 DATASET PREPARATION AND ANNOTATION CONSISTENCY

The datasets used (e.g., Anti-UAV, DUT, Kaggle Drone Detection) came in diverse formats, some lacking proper annotations or using JSON formats not compatible with YOLO. This required writing custom scripts to convert annotations into YOLO format and performing extensive validation to ensure consistency.

🔹 DRONE SIZE VARIABILITY AND SMALL OBJECT DETECTION

Drones in the dataset appeared in varying sizes, poses, and lighting conditions — with small drones being the most difficult to detect accurately. Model tuning for detecting small-sized objects without compromising performance on medium or large drones was particularly challenging.

🔹 HANDLING CLASS IMBALANCE

Some datasets had a disproportionate number of drone and non-drone images or variations in drone sizes. This imbalance affected training quality and led to biased predictions. To counter this, careful selection of sequences and normalization techniques were used during data preparation.

🔹 EVALUATION METRIC COMPLEXITY

Apart from standard metrics like Precision and Recall, size-wise evaluation and FPS-based comparison required additional scripting and post-processing. Designing a fair and comprehensive evaluation strategy across different models and datasets was time-consuming but essential.

🔹 MODEL SELECTION AND TRADE-OFF DECISIONS

Selecting between newer models (YOLOv10n) and proven ones (YOLOv8m) posed trade-offs between accuracy and real-time performance. The challenge was in balancing detection quality with inference speed, depending on application demands.

11. PROJECT DEMONSTRATION

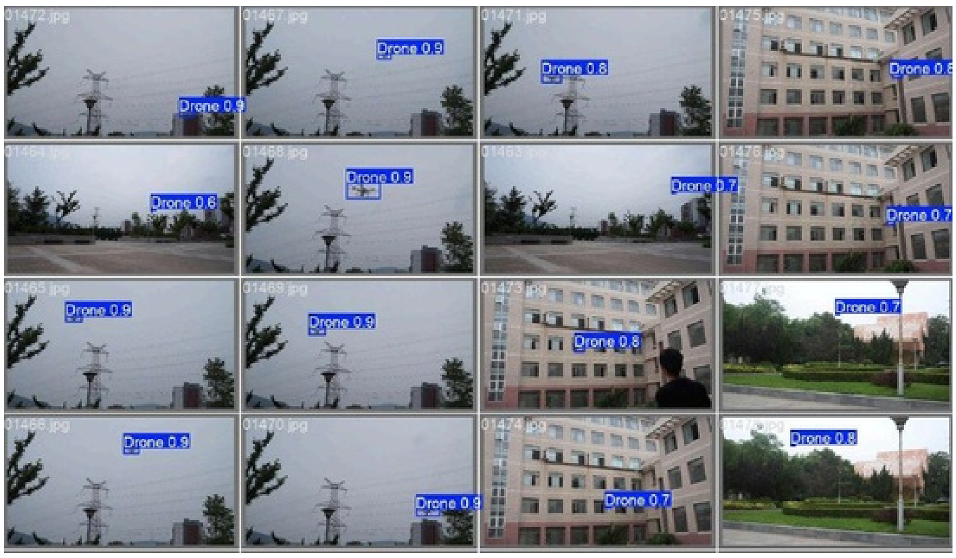

Fig.1 Detection results using MODEL-3 (YOLOv8m) on test images from the DUT-Anti-UAV dataset. The bounding boxes show predicted drone locations with corresponding confidence scores.

12. DATA & DOCUMENTATION

The dataset-wise details such as number of annotated frames, drone size distribution, and model-wise evaluation metrics (precision, recall, mAP, FPS) are documented in this Google Sheet.

13. CONCLUSION AND FUTURE SCOPE

This project successfully implemented a real-time drone detection system using state-of-the-art YOLO-based deep learning models across multiple datasets and modalities. Through a comparative evaluation of four trained models, the report analyzed the trade-offs between inference speed (FPS) and detection accuracy (mAP, Precision, Recall).

YOLOv8m models offered better detection quality, particularly for medium and small drones, while YOLOv10n achieved exceptional speed, making it suitable for latency-sensitive use cases.

The work demonstrates the potential of AI-powered vision systems in airspace monitoring and lays the groundwork for future enhancements such as:

- Integrating object tracking for multi-frame temporal consistency
- Extending detection to multiple drone classes or swarm detection
- Optimizing YOLO models for edge deployment using TensorRT or ONNX
- Implementing real-time alerting or defensive response systems

The learnings from this project contribute not only to drone detection research but also to broader real-time object detection challenges.

14. REFERENCES

- Z. Zhu et al., "Detection and Tracking Meet Drones Challenge," CVPR Workshops 2020, IEEE.
- Wang, Dong, et al. "DUT Anti-UAV: A Large Benchmark for Vision-Based Drone Monitoring." arXiv preprint arXiv:2103.05224.
- Jocher, G., et al. "YOLO by Ultralytics."https://github.com/ultralytics/yolov5
- Redmon, J., & Farhadi, A. "YOLOv3: An Incremental Improvement." arXiv:1804.02767