International Research Journal of Multidisciplinary Scope (IRJMS), 2025; 6(3):914-931

Original Article | ISSN (O): 2582-631X DOI: 10.47857/irjms.2025.v06i03.04370

Next-Generation Aerial Threat Detection Using Yolov5

Ayush Kumawat[1], Anand Jawdekar[2]*, Vicky Gupta[3], Shivam Kumar Upadhyay[4], Sandeep Wadekar[2], Chandra Shekhar[2]

[1]Indian Space Research Organization, Space Application Centre, Ahmedabad, Gujarat, India, [2]Parul Institute of Engineering and Technology, Parul University, Vadodara, Gujarat, India, [3]Jaypee Institute of Information Technology, Noida Uttar Pradesh, India, [4]Parul Institute of Technology, Parul University, Vadodara, Gujarat, India. *Corresponding Author's Email: anand.cs2007@gmail.com

Abstract

Unmanned Aerial Vehicles (UAVs), commonly known as drones, have emerged as versatile tools with applications spanning surveillance, aerial photography, agriculture, and disaster response. However, their increasing presence raises security concerns, necessitating robust detection systems. This research explores the development of a real-time drone detection system using the state-of-the-art YOLOv5 algorithm. This paper presents a detailed methodology, comparative analysis, and results demonstrating the efficacy of our approach in enhancing security measures against UAV threats. Current detection technologies encompass a range of approaches, including computer vision, machine learning, radar systems, and acoustic sensors. Traditional methods often rely on rule-based algorithms or handcrafted features, exhibiting limited scalability and adaptability to dynamic environments. In this research paper, we present a novel approach to drone detection utilizing the YOLOv5, a powerful object detection algorithm, with OpenCV, a versatile computer vision library. The heart of our system lies in a meticulously curated dataset containing 1440 images, showcasing a diverse array of drones. Each image tells a unique story, helping our system learn to recognize drones of different types and sizes. Our methodology involves a detailed process of training YOLOv5 using the dataset, carefully splitting the data into training, validation, and testing sets, and setting up a real-time detection system using OpenCV. The system not only identifies drones but also issues warnings when a drone is detected within or near a specified area.

Keywords: Computer Vision, Custom Dataset, Drone Detection, Real-Time Detection, YOLOv5 Algorithm.

Introduction

In our rapidly advancing technological landscape, the soaring popularity of drones brings both excitement and challenges. Drones, or Unmanned Aerial Vehicles (UAVs), have become increasingly accessible to individuals and organizations worldwide. While these flying gadgets offer incredible possibilities in various fields, such as photography, agriculture, and delivery services, their misuse poses potential threats to privacy, safety, and security. Additionally, many Commercial-off-the-Shelf (COTS) drones come equipped with cameras, facilitating First-Person View (FPV) capabilities. This feature enables live video streaming from the drone's camera to the controller or a separate viewing device. However, the ease of access to such live video feeds raises concerns about potential privacy infringements. To address these issues, the Federal Aviation Administration (FAA), the civil aviation regulatory body of the U.S. government, introduced a set of policies on January 15, 2021. These policies, known as Remote ID, establish guidelines for the identification requirements of drones (1). Numerous drone detection solutions are available in the market, but their high costs often render them inaccessible to many individuals (2, 3). Furthermore, some of these solutions are designed to detect only specific models of drones, thereby limiting their effectiveness in detecting a wide range of UAVs (4). Recognizing these limitations, affordable drone detection solutions are increasingly being developed through intensified research efforts. These efforts primarily focus on detecting unauthorized drones entering restricted airspace or private areas. In recent years, methods for detecting drones by analyzing their network traffic patterns have been explored in several studies (5, 6). This approach leverages insights from network data to identify drones, offering a promising avenue for cost-effective drone detection solutions. However, these approaches may fail to differentiate drones from other moving radio sources. In response to these pressing concerns, our research endeavors to pioneer an

(Received 09th March 2025; Accepted 07th July 2025; Published 26th July 2025)

innovative solution aimed at detecting drones in real-time. By harnessing the latest advancements in computer vision and machine learning, our system is designed to swiftly identify and alert against unauthorized drone activities. Through meticulous development and rigorous testing, we aim to provide a robust and reliable defense mechanism against potential threats posed by drones in various settings, ranging from critical infrastructure protection to public safety enforcement. Our commitment to leveraging cutting-edge technologies underscores our dedication to safeguarding privacy, security, and airspace integrity in the face of evolving UAV challenges. With the proliferation of drones, there's a growing need for robust detection mechanisms to safeguard public spaces, events, and critical infrastructure. The existing UAV detection methods are mainly divided into the following categories: audio signal-based detection methods, radar-based detection methods, radio frequency-based detection methods, and image and video-based detection methods. Traditional methods of monitoring, relying on human surveillance alone, are becoming impractical due to the sheer volume and agility of drones. Our solution aims to automate the detection process, reducing reliance on manual oversight and providing a timely response to potential threats.

Our research leverages two powerful tools, YOLOv5 (You Only Look Once, version 5) and OpenCV. YOLOv5 is a state-of-the-art object detection algorithm, allowing computers to rapidly and accurately identify objects in images or video frames. OpenCV, an open-source computer vision library, serves as the backbone for implementing our detection system. Together, they empower our model to quickly analyze real-time video feeds and pinpoint the presence of drones.

Our system boasts several features that set it apart:

Real-time Detection

The ability to identify drones instantly as they appear in the video feed.

Interactive Rectangle Control

Users can define and adjust a detection area interactively.

Warning Mechanism

Immediate warnings are triggered when a drone is detected within or near the defined area.

The benefits of our system extend to enhanced security, automated monitoring, and adaptability to diverse environments. Furthermore, our approach is customizable, open-source, and cost-effective. As drones become more accessible to individuals and organizations alike, the need for effective detection systems has never been more critical. Ensuring that drones are used responsibly and within legal boundaries is vital for safeguarding public spaces, events, and critical infrastructure.

Traditional surveillance methods, relying heavily on human observation, struggle to keep pace with the agility and prevalence of drones. To address this challenge, our research focuses on developing an advanced drone detection system that leverages state-of-the-art technologies in computer vision and machine learning.

In the past, various techniques, such as radar, were used to detect drones. However, it is very difficult for radar to do so, due to the low levels of electromagnetic signals that drones transmit. Similarly, other techniques, such as acoustic and radio frequency-based drone detection, are costly and inaccurate. Recently, machine learning-based drone detectors, such as SVM and artificial neural network classifiers, have been used to detect drones, achieving better success than radar and acoustic drone detection systems (7).

Different types of drone images are collected to build a dataset (8). The images are then annotated in the YOLO format for training a YOLOv3 model. An NVIDIA GeForce GTX 1050 Ti GPU was used to train the dataset with chosen parameter values, such as a learning rate of 0.0001, batch size of 64, and 150 total epochs. The best mAP value was 0.74. PyTorch, an open-source machine learning programming language, was used to train and test the YOLOv3 model.

YOLOv4 was used to automatically detect drones with the aim of integrating the trained model into a CCTV camera, thereby reducing the need for manual monitoring. The authors collected their dataset from public resources such as Google images, opensource websites, etc. The images then converted into the YOLO format using free and paid image annotation tools. They fine-tuned the YOLOv4 architecture by customizing filters, max batches, subdivisions, batches, etc. After training the YOLOv4 model for 1300 iterations, the researchers achieved a mAP of 0.99. Though their mAP value was very high, they trained only 53 images and did not address model overfitting, resulting in a greater improvement scope.

An approach based on YOLOv5 was presented, utilizing a dataset of 1359 drone images. They fine-tuned the model on a local system with an 8 GB NVDIA RTX2070 GPU, 16 GB of RAM, and a 1.9 GHz CPU. They employed a 60:20:20 split of the dataset for training, testing, and validation. They trained the model on top of COCO pre-trained weights and obtained a precision of 94.70%, a recall of 92.50%, and a mAP of 94.1%.

In their study, drones were suggested to be detected by analyzing their First-Person View (FPV) streams (9). They propose comparing the bit rate of an FPV stream from a drone with that of a previously recorded FPV data stream. However, they note that the bit rate of an FPV stream tends to increase as the scene becomes more dynamic. This means that a drone capturing a highly dynamic scene may not always match a specific set of FPV bit rates accurately. Additionally, the authors argue that the Received Signal Strength Indicator (RSSI) of FPV channels differs from other WiFi video streaming services due to the movement of drones. However, they did not consider the possibility of other moving video streaming devices, such as Voice over Internet Protocol (VoIP) applications on smartphones.

In their study, classical features similar to those employed in previous research were utilized to distinguish First-Person View (FPV) drones from other devices, thereby enabling drone detection (10-12). They demonstrate that their framework achieves high accuracy in detecting drones by analyzing features extracted from packet samples exchanged between the drone and its controller, with a minimum of 50 packets. However, the authors acknowledge two unresolved issues: the recognition of new types of unmanned aerial vehicles (UAVs) and the detection of modified video patterns. The findings of previous research were built upon by enhancing detection accuracy and addressing the challenges associated with recognizing new UAV types (13). At the forefront of our research are two key tools: YOLOv5 (You Only Look Once) and OpenCV (Open Source Computer Vision Library). YOLOv5 is an advanced object detection algorithm known for its rapid processing speed and accuracy. Paired with OpenCV, a versatile toolkit for image and video analysis, our drone detection system gains the capability to process real-time video feeds and identify drones swiftly and accurately.

Methodology

In our pursuit of creating an effective drone detection system, we followed a methodical process that combined precision in data preparation, leveraging advanced machine learning algorithms, and employing robust real-time processing techniques. Our research endeavors to create an effective and reliable real-time drone detection system, employing a carefully crafted methodology. The process involves several pivotal stages, each contributing to the development of a robust model.

Dataset Collection and Organization - Diverse Drone Images

Before our computer can recognize drones, it needs to learn from lots of pictures showing different types of drones. Imagine you're teaching a friend what a drone looks like—you'd show them many pictures of drones in various situations. The dataset is curated from open-source platforms such as Kaggle and GitHub, incorporating drones of varying models, sizes, and colors. Scenes included forests, urban skylines, clear and cloudy skies, and varied lighting conditions to ensure robustness and generalization. This diverse dataset simulates real-world complexity, improving the model’s resilience in different operational environments.

Collection of Images: Building a Photo Album for Our Computer

We created a big collection of images, like a massive photo album, to train our computer. This album had 1440 pictures of drones doing all sorts of things. We made sure to include different types of drones, in different sizes, and in various backgrounds. It's like having a diverse set of examples so that the computer learns to recognize all kinds of drones.

Labelling of Images: Making Sure Each Photo Tells a Clear Story

Just like in a photo album, each picture had to tell a clear story. We carefully labeled each image to let the computer know where the drones are and what they looked like. This labeling process is a bit like putting captions on photos so that our computer understands exactly what it's seeing.

Formatting: Getting the Data Ready for Training

Once our photo album is ready, we needed to organize it in a way that our computer could understand. Think of it like arranging your photos in a neat order. We used a special tool called Roboflow to help us with this. It made sure our images are in a format that our computer, YOLOv5, could easily use for learning.

Checking and Double-Checking for Accuracy

We wanted to be super sure that our photo album is top-notch. So, we double-checked everything. We looked at each image, made sure the labels matched what is in the picture, and verified that our dataset covered a wide range of drone scenarios. This attention to detail is crucial to ensure our computer learns accurately.

Creation of Dataset: Exporting Our Photo Album for YOLOv5

Once we are confident that our photo album is perfect, we exported it into a format that YOLOv5 could understand. It's like turning our photo album into a special book that YOLOv5 can read. This step is crucial for the computer to learn from the images and become really good at spotting drones.

Rigorous Curation with Roboflow

To prepare our dataset for training, we harnessed the capabilities of a remarkable tool known as Roboflow. Picture it as a wizard guiding us through the intricacies of dataset organization and structuring, ensuring our computer vision model, YOLOv5, comprehends the data seamlessly. Roboflow played a pivotal role in managing the nitty-gritty details, assuring that our dataset achieved the optimal shape for training our model. In this process, Roboflow served as a dedicated assistant, streamlining the organization of our images and associated information. Its role extended to formatting our images in a manner aligned with YOLOv5's requirements. Each image underwent meticulous labeling, providing annotations that would serve as a guide for YOLOv5 to understand the nuances of drone characteristics.

One of the striking aspects of Roboflow is its adaptability to different models and algorithms. It ensures that our dataset conforms to the specific preferences of YOLOv5, eliminating potential roadblocks during the training phase. Essentially, Roboflow acts as a translator, ensuring our dataset speaks the same language as YOLOv5, facilitating a seamless learning process.

Moreover, Roboflow boasts a user-friendly interface, making it accessible to both researchers and developers. Even without an extensive coding background, Roboflow guides users through the necessary steps, ensuring that the dataset is meticulously prepared for effective model training. Its automation of certain steps and workflow optimization significantly contributes to efficiency, a critical factor when dealing with large datasets containing numerous images.

As an additional feature, Roboflow empowers us to generate an API for our model to facilitate the training process. Utilizing a code snippet provided by Roboflow, we seamlessly export our dataset.

In summary, the second step of our methodology involves leveraging Roboflow not only for the organization, labeling, and formatting of our dataset but also for the generation of a convenient API. This step is pivotal, laying the foundation for a successful and efficient training process for our drone detection model, YOLOv5.

Dataset Splitting for Training, Validation, and Testing

We embarked on our journey by curating a rich dataset of 1440 images showcasing various drone scenarios. Each image is labeled with precision, indicating where the drones are located. This data collection phase is meticulous, ensuring diversity and accuracy in our dataset. To prepare our dataset for effective training, we engaged in data preprocessing. This involved refining and organizing the images, making sure they are in a standardized format. To streamline the data preparation process, we leveraged the power of Roboflow. It acted as our data organizing assistant, helping us import our meticulously prepared dataset and standardize it for optimal use in training our model. Roboflow served as a bridge between our curated images and the training process, ensuring that everything is in order.

Now, for effective model training and evaluation, we split our dataset into three sets: a training set (1018 images), a validation set (270 images), and a test set (152 images). Figure 1 is from the Roboflow dashboard showing dataset splitting this division is crucial to ensure our model learns well from a variety of images, fine-tunes its performance, and gets rigorously tested on new, unseen data.

With our data neatly organized and standardized, we proceeded to split it into three distinct sets to facilitate effective model training and evaluation as seen in Figure 2.

Training Set (71%)

A significant portion of our dataset, constituting 71%, is earmarked for training our model. This set, containing 1018 images, played a crucial role in teaching our computer vision system to recognize and understand the nuances of different drone scenarios. It's like giving our model a vast library of examples to study from.

Testing Set (19%)

To rigorously evaluate the performance of our trained model, we allocated 19% of the dataset for testing. This set, containing 270 images, represented unseen scenarios for the model. Testing on this set provided insights into how well our model could handle real-world situations beyond those encountered during training and validation.

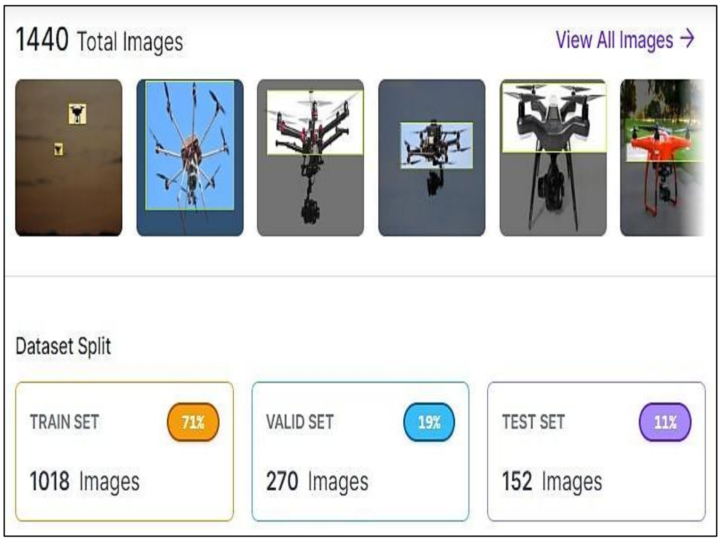


Figure 1: Data Split into Training, Testing and Validation

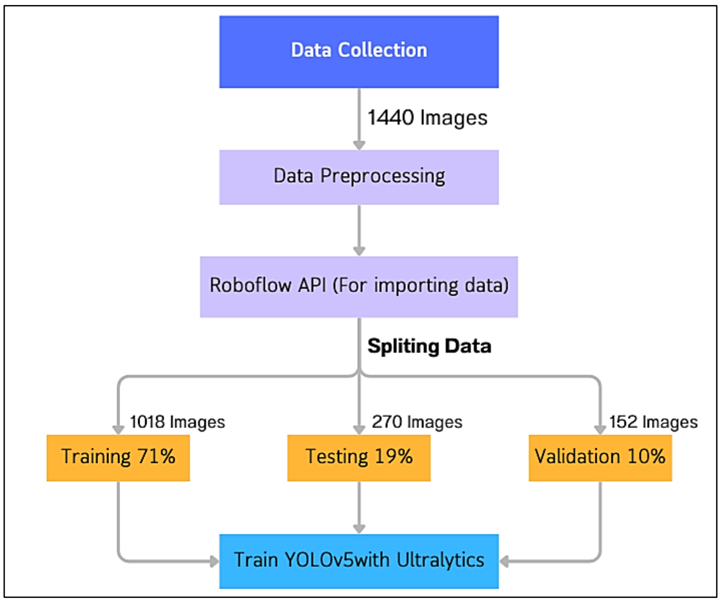


Figure 2: Process Flow of Training YOLOv5

we initiated the training of our model using YOLOv5, a state-of-the-art object detection algorithm. Ultralytics, a platform supporting YOLOv5, played a key role in this process (14). The training involved multiple epochs, allowing YOLOv5 to learn from our diverse dataset and fine-tune its ability to recognize drones.

Training the Model: Teaching the Computer to Recognize Drones

The core of our research involves the intricate process of training our computer vision model, YOLOv5. Training is akin to teaching the model to recognize drones by exposing it to a curated dataset of labeled images. These images serve as the educational material for YOLOv5, guiding it to distinguish drones from their surroundings. The steps we took in training are specific and detailed:

Image Size Definition (img): We defined the input image size for our training, ensuring consistency. The input image resolution of 640x640 is selected to balance computational efficiency with the need to detect small-scale aerial targets. YOLOv5's auto-anchor feature is leveraged to tailor anchor boxes to the specific dimensions of drone instances in our dataset. Loss functions, including CIoU loss for bounding box regression and BCE loss for classification and objectness, are retained due to their robustness in handling scale and aspect ratio variations typical in drone imagery.

Batch Size Determination (batch): We specified the batch size, determining how many images YOLOv5 processes in each training step.

Training Epochs (epochs): We decided on the number of training epochs, essentially how many times YOLOv5 would go through our entire dataset during training.

Training Data Location (data): We provided the location of our training dataset, ensuring YOLOv5 knows where to find our labeled images.

Starting with Pretrained Weights (weights): We initiated our training using pretrained weights from a generic COCO dataset, giving YOLOv5 a head start in learning.

Cache for Faster Training (cache): We used caching to speed up the training process, making it more efficient.

The culmination of our model training methodology involved the execution of a meticulously crafted training command. The Python script executed, encapsulated the entire training configuration. This command orchestrated the learning process, orchestrating the adjustments of internal parameters, optimizing model weights, and iteratively refining the model's ability to discern drones.

Our custom training configuration incorporated the vital aspects necessary for effective model learning:

Data Specification (data): The location of our curated training dataset is specified using the data parameter. This ensured that YOLOv5 knew precisely where to access our labeled images for learning.

Pretrained Weights Initialization (weights): We kickstarted the training process by initializing YOLOv5 with weights pretrained on a generic COCO dataset. This strategic choice provided our model with foundational knowledge, accelerating the learning curve.

Caching for Efficiency (cache): To enhance training efficiency, we implemented caching, a mechanism that stored images in memory for faster access during subsequent training iterations. This optimization contributed to a streamlined and resource-efficient learning process.

The training process unfolded over multiple epochs, with YOLOv5 iteratively analyzing the dataset, adjusting its internal parameters, and incrementally improving its ability to accurately identify drones. The model's progress is closely monitored, and adjustments are made as needed to address any challenges encountered during the learning journey.

Combined Detection Process Flow

To visualize how our detection process works, we have a handy flowchart given at Figure 3. First, we divide the video frames. Then, we predict the target's bounding box and category, essentially telling our computer, "Hey, where is the drone, and what kind is it?" Next, we use a confidence rating method. It's like giving a grade to the computer's prediction. If the confidence is high (which means the computer is really sure about its prediction), we simply draw a bounding box around the drone and annotate its type and confidence score. But, if the confidence is not that high, we apply a technique called non-maximum suppression. It's like refining the computer's guess by filtering out less certain predictions. Finally, we output the target bounding box, annotate the target type, and show the confidence score. This process ensures that our computer is not only accurate but also cautious, double-checking its predictions when needed.

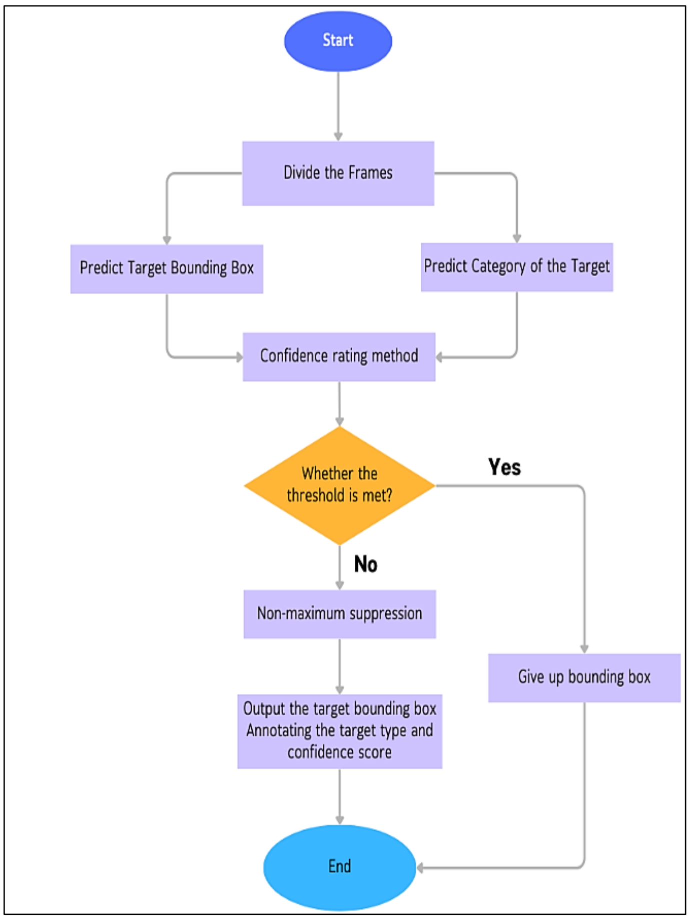

Start
Divide the Frames
Predict Target Bounding Box
Predict Category of the Target
Confidence rating method
Whether the threshold is met?
Yes
No
Non-maximum suppression
Give up bounding box
Output the target bounding box Annotating the target type and confidence score
End

Figure 3: Flow of Detection Process

Real-Time Drone Detection Implementation with Warning System Integration

Video Feed Processing: In the real-time implementation of our drone detection system, we utilized the OpenCV library to process video feeds. OpenCV served as the backbone, allowing us to capture and analyze each frame from the video source seamlessly.

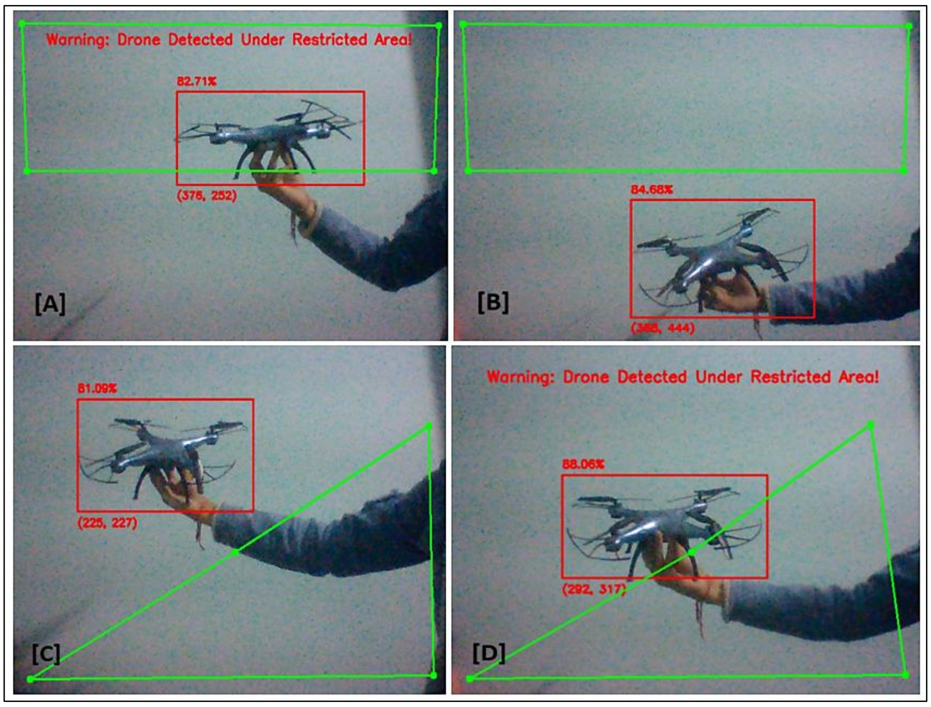
YOLOv5 Object Detection: With each frame processed, we engaged our trained YOLOv5 model to perform object detection. YOLOv5, having learned from our curated dataset, swiftly identified potential drones within the video frames. The algorithm not only recognized the presence of drones but also provided precise bounding box coordinates around the detected objects.

Bounding Box Drawing and Confidence Score Display: Upon detecting a drone with confidence above a predefined threshold, our system drew a bounding box around the identified object within the frame. This bounding box served as a visual representation of the detected drone, enhancing the interpretability of the system's output. Simultaneously, the confidence score, denoting the algorithm's certainty about the detection, is displayed alongside the bounding box.

Interactive Rectangle Creation and Adjustment: In parallel, we implemented an interactive rectangle feature as shown in Figure 4 using mouse event handling in OpenCV. Users could define and adjust a rectangular area of interest on the video feed. This area acted as a virtual restricted zone, enabling the system to monitor drone activity within or near the specified region.

Warning System Integration: To enhance the security capabilities of our system, we integrated a warning mechanism. If a detected drone intersected with or is inside the user-defined rectangle, a warning message is promptly displayed on the video feed. This immediate alerting feature ensures timely responses to potential drone threats.

User Interaction and System Display: To ensure user-friendly interaction, we facilitated the creation and adjustment of the detection rectangle through mouse events. Users could also adjust the detection area interactively, providing them with control over the monitoring process. The system displayed the live video feed with overlaid bounding boxes, confidence scores, and warning messages, providing real-time feedback to the user.



Warning: Drone Detected Under Restricted Area!
82.71%
(378, 252)
[A]
84.68%
(368, 444)
[B]
81.09%
(225, 227)
[C]
Warning: Drone Detected Under Restricted Area!
88.06%
(292, 317)
[D]

Figure 4: Demonstration of Real-Time UAV Detection System (A-D)

Hardware and FPS Performance Benchmarking

To ensure the practicality of our system in real-world deployments, we evaluated performance across multiple hardware platforms:

- Intel Iris Xe (Integrated GPU): ~12 FPS
- Jetson Nano (Edge Device): ~7 FPS (optimized with TensorRT)
- Raspberry Pi 4 (Minimum Viable Device): ~5–6 FPS using quantized model with OpenCV backend
- The minimum configuration to run our drone detection program smoothly includes:
- Raspberry Pi 4 (4 GB RAM or above)
- Raspbian OS / Ubuntu 20.04 ARM
- Python 3.8+, OpenCV 4.5+, PyTorch (Lightweight version or TorchScript)
- CSI/USB Camera with 15 FPS minimum capture rate
- These results confirm that the model is deployable even on lightweight IoT edge devices such as Raspberry Pi 4, commonly used in real-time surveillance and smart security systems. The system performs reliably with 15 FPS camera input, which is sufficient for drone detection due to their relatively slow movement in localized areas.

Results and Discussion

In the realm of drone detection, the results and evaluation section serve as the compass guiding the success of our research. This critical phase unveils the performance and capabilities of our real-time drone detection system, showcasing how effectively it discerns drones from its surroundings. Through a series of meticulously crafted charts and graphs, we illuminate the model's learning journey, unraveling its capabilities in object localization, label attribution, and precise bounding box predictions. These visual representations, such as loss charts and mAP progression, offer a transparent narrative of the model's evolving accuracy throughout the training process. The ensuing F1-Confidence Curve, Precision-Recall Curve, and other insightful charts encapsulate the delicate balance between precision and recall, essential for robust drone detection. As we delve into numerical metrics, the summary reveals an impressive mAP of 97.23%, coupled with precision and recall rates of 95.82% and 96.42%, respectively. These metrics underscore the system's adeptness in identifying drones while maintaining a high level of precision, validating the efficacy of our approach in enhancing security and automation in drone surveillance scenarios.

Model Evaluation

The success of any innovation lies in its ability to perform reliably and consistently. In this section, we rigorously evaluate the effectiveness of our real-time drone detection model. Through a series of metrics, charts, and comprehensive analyses, we assess how well our system has learned to identify and distinguish drones from the surrounding environment.

Precision

Understanding how our model's precision evolves over the course of training is crucial for gauging its accuracy in identifying drones. Precision is a measure of the accuracy of our model, indicating how many of the identified drones are drones (15). The formula for precision is given at equation 1:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives} \quad [1]$$

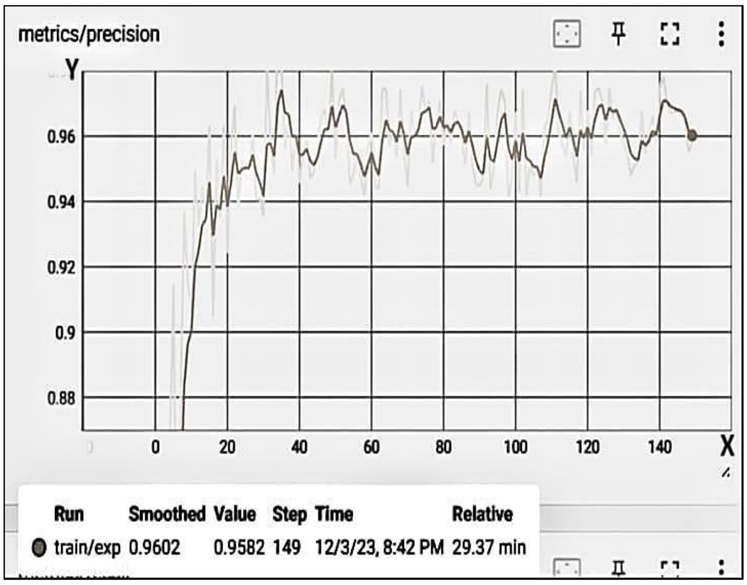


Figure 5: Outlines the Journey of Precision Values across 150 Training Epochs

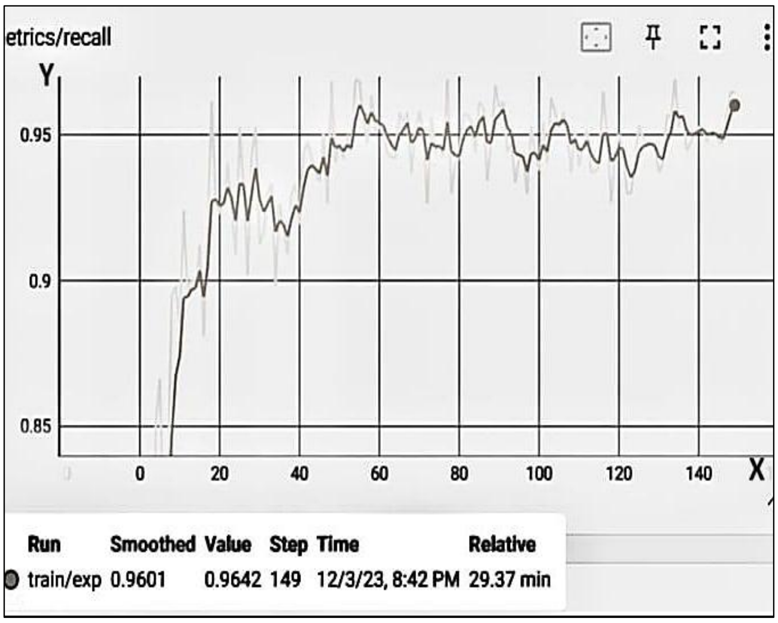
In Figure 5, the x-axis represents the training epochs, and the y-axis indicates the precision values. Precision, in simple terms, is a measure of how accurate our model is when it claims to have identified a drone.

In the initial epochs, precision tends to be relatively low. This is expected as the model is just starting to learn and may make some errors in identifying drones. As the model undergoes more epochs, precision steadily improves. The model learns from its earlier mistakes and refines its ability to distinguish drones from other objects (16), resulting in a noticeable increase in precision. In the latter epochs, precision approaches its peak, reaching a commendable value of 0.95 on the 150th epoch. This signifies that our model has become highly accurate in identifying drones, with only a minimal margin of error.

Recall

Understanding how our model improved in recalling drones is essential for assessing its effectiveness. The Recall chart vividly illustrates this journey over the course of training, revealing notable patterns and milestones. Recall measures the model's ability to correctly identify all instances of drones in the dataset. The formula for recall is given at equation 2:

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad [2]$$



Run	Smoothed	Value	Step	Time	Relative
● train/exp	0.9601	0.9642	149	12/3/23, 8:42 PM	29.37 min

Figure 6: Outlines the Journey of Recall Metrics across 150 Training Epochs

As seen in Figure 6, in the initial stages, the Recall is relatively modest, highlighting the model's learning curve. It started at a lower value, there is a notable surge in Recall around 0.93 till 20th epoch, indicating that the model is struggling to capture all instances of drones in the early training phases. Between epochs 20 and 40, the recall stabilizing at values between 0.90 and 0.95. This phase suggests that the model began to grasp more intricate features of drones, leading to a more consistent and reliable detection of these objects (17). Post the 40th epoch, there is a gradual and steady increase in Recall. This progressive climb is indicative of the model's continued learning and adaptation to diverse scenarios. As the training journey approached its culmination at the 150th epoch, the Recall surpassed 0.95 and reached 0.96. This signifies that the model achieved its peak performance, demonstrating a high level of proficiency in identifying drones across various conditions.

MAP (Mean Average Precision)

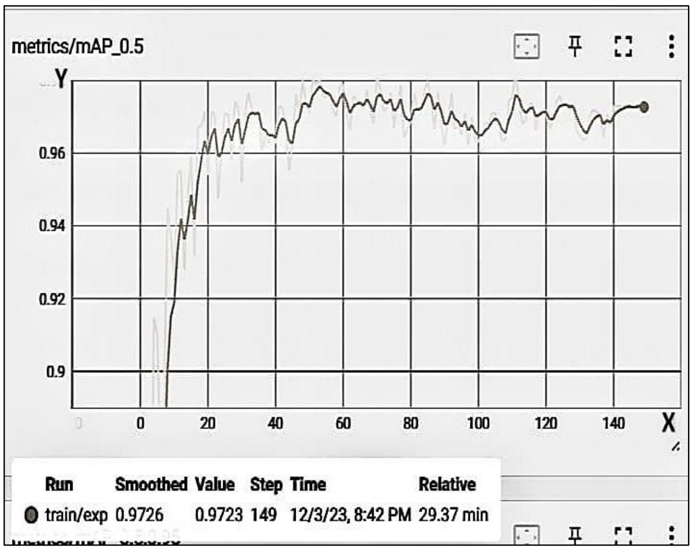
The mAP_0.5 chart provides a fascinating insight into how our model's performance evolved over the course of training (18), specifically concerning the Mean Average Precision (MAP) at a confidence threshold of 0.5. The mAP is calculated by taking the average of precision values at different recall levels as given in equation 3.

$$mAP = \frac{1}{C}\sum_{i=1}^{C} AP_i \qquad [3]$$

Where:

- C is the total number of classes.
- AP_i, is the average precision for class

Mean Average Precision at a 0.5 threshold is a critical metric. It essentially quantifies how well our model balances precision and recall, emphasizing accuracy in detecting drones while minimizing false positives and false negatives.



Run	Smoothed	Value	Step	Time	Relative
● train/exp	0.9726	0.9723	149	12/3/23, 8:42 PM	29.37 min

Figure 7: Outlines the Journey of Mean Average Precision at 0.5 Confidence Threshold

As shown in Figure 7, in the initial stages of training, our model demonstrated a rapid surge in mAP, reaching an impressive 0.96 within the first 20 epochs. This early success indicates that the model quickly grasped essential patterns and features related to drone detection. As the training progressed, the mAP_0.5 curve exhibited a gradual and steady climb, showcasing the model's continuous learning and refinement. A notable highlight of the mAP_0.5 chart is the consistent progression observed up to the 150th epoch, where the mAP reached a commendable 0.97. The surge and subsequent steady climb in the mAP_0.5 values affirm the robustness of our drone detection system. This suggests that our model achieved a balance where it confidently identifies drones with a high level of accuracy across various situations.

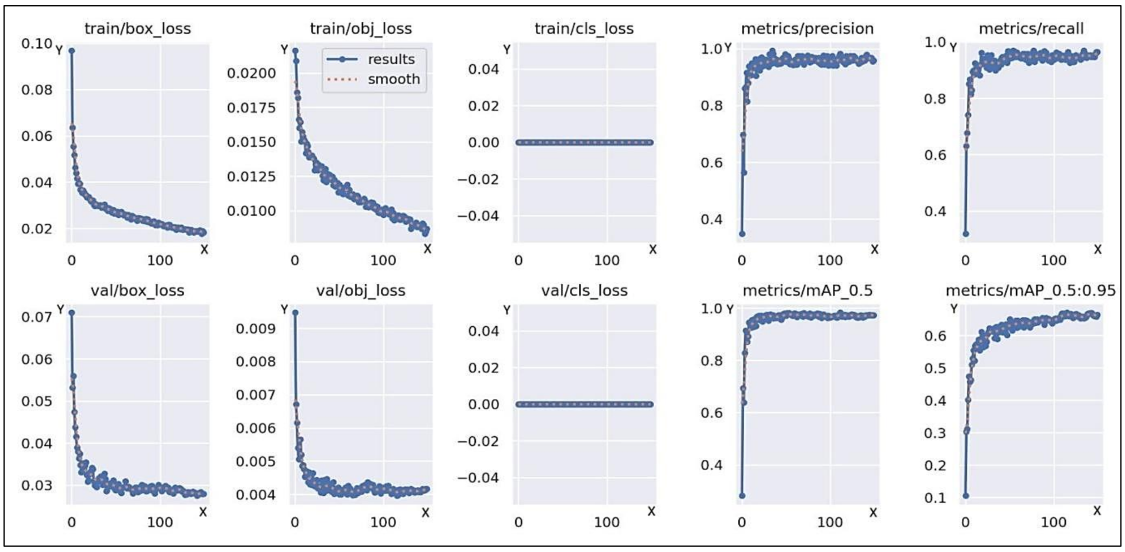


Figure 8: Metrics Charts Showing Model's Progress

Figure 8 shows the overall training summary of the model. The loss curves indicate a downward trend, meaning that during training, the losses are minimized both for training and validation. The metrics curves show upward trends, meaning the performance of the model improved over the iterations during training.

Table 1: Overall Evaluation Metrics Results

Class	Precision	Recall	MAP
Drone	95.82 %	96.42 %	97.23 %

Table 1 summarizes the core evaluation metrics. The drone class achieved a precision of 95.82%, indicating that false positives are minimal, while the recall of 96.42% shows that most actual drones are correctly detected.

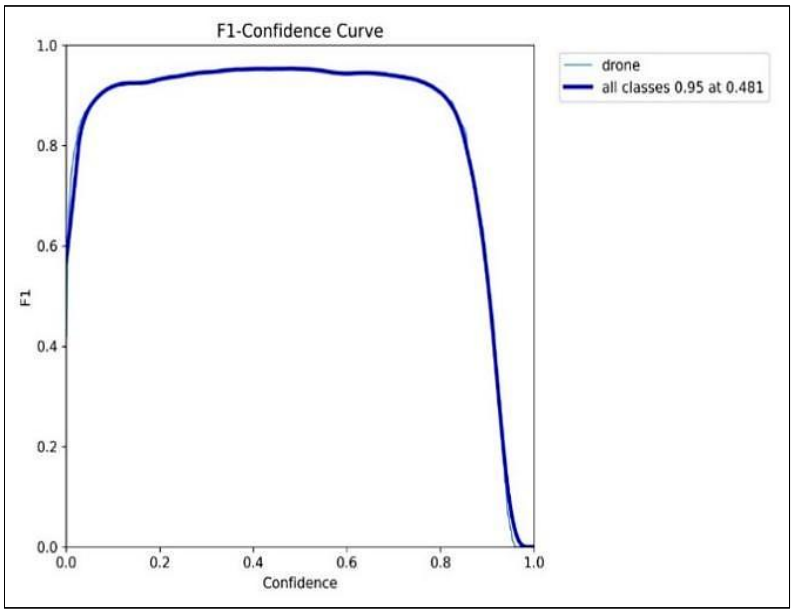


Figure 9: F1-Confidence Curve Graph

The Figure 9 F1-Confidence Curve illustrates the balance between precision and recall at different confidence thresholds. Finding an optimal threshold ensures a good balance between minimizing false positives and false negatives, contributing to effective drone detection (19).

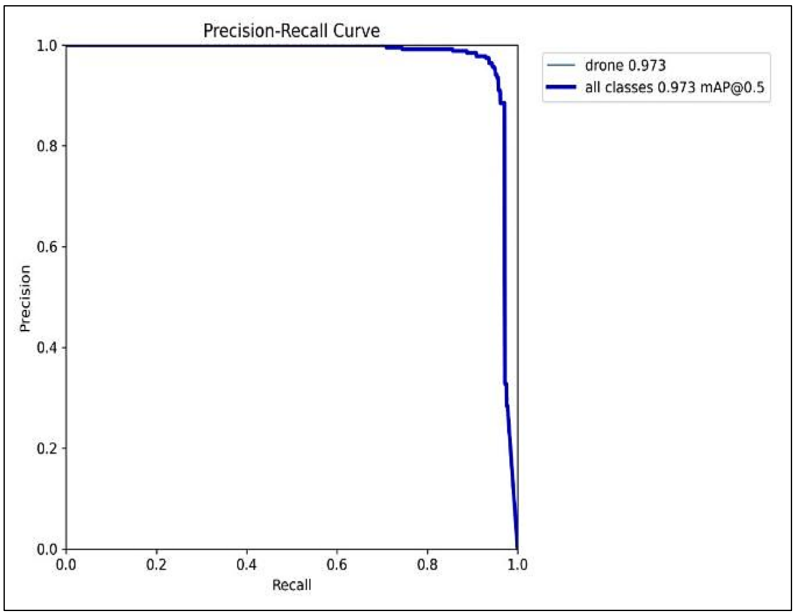


Figure 10: Precision Recall Curve Graph

The given Precision-Recall Curve in Figure 10 showcases the trade-off between precision and recall at different classification thresholds (20). A curve that hugs the top-right corner indicates a model with high precision and recall, offering confidence in its detection capabilities.

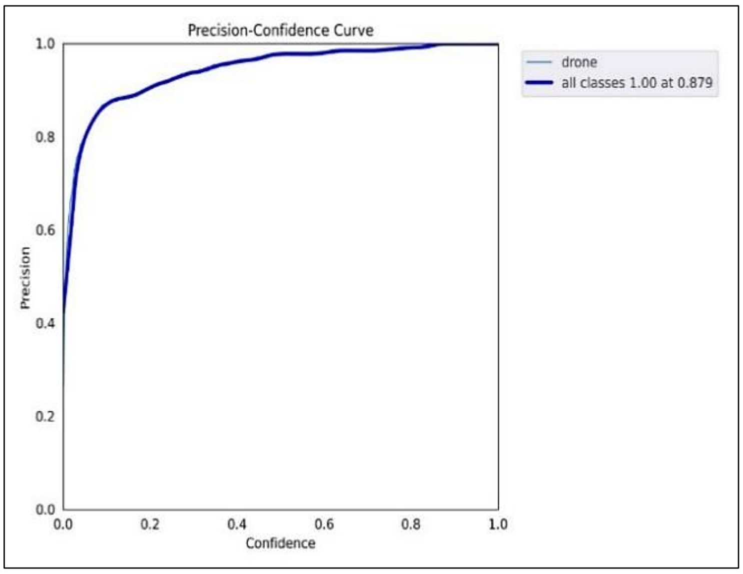


Figure 11: Precision Confidence Curve Graph

In Figure 11 the Precision Confidence curve illustrates how precision varies with different confidence thresholds. Identifying an appropriate threshold ensures that the system maintains a high level of precision in detecting drones (21).

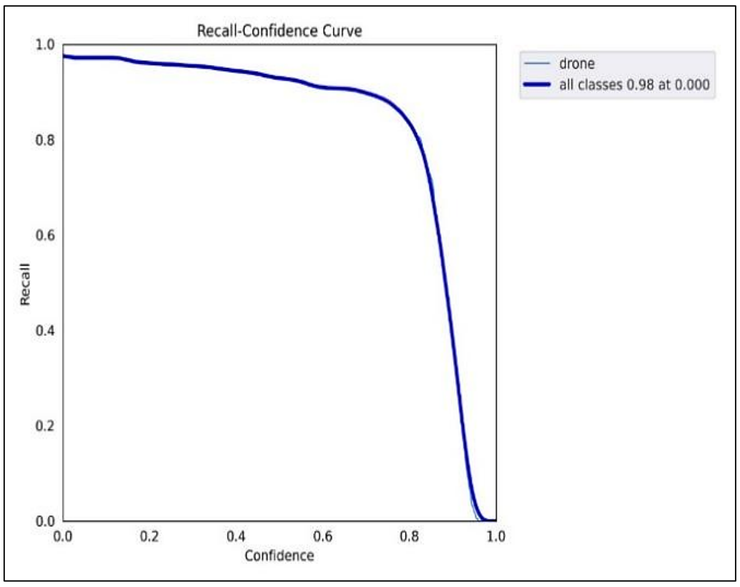


Figure 12: Recall Confidence Curve Graph

In Figure 12 the Recall-Confidence Curve depicts how recall changes with different confidence thresholds. Striking a balance between recall and precision is essential for an effective and reliable drone detection system (22).

Visual Results

Our detailed methodology, from data collection to real-time implementation, ensures a comprehensive understanding of our drone detection system. Our drone detection system, a combination of meticulous training and implementation, demonstrated significant efficiency in real-time scenarios. The results showcase the system's ability to accurately identify and locate drones, providing a critical layer of security in various environments.

Detection Accuracy

The above image illustrates the system's detection accuracy, with bounding boxes accurately outlining the detected drones as shown in Figure 13. Each box represents a successful identification, demonstrating the model's ability to precisely locate drones within the video frame.

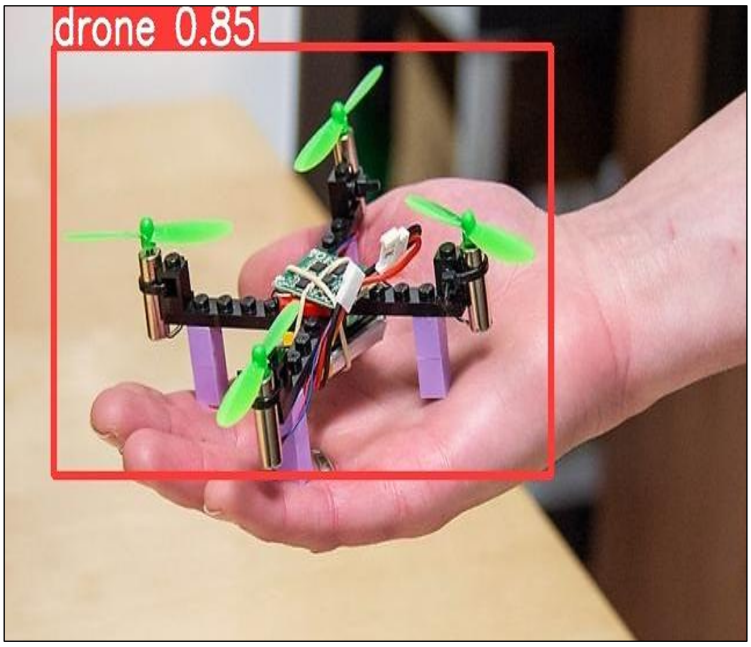


Figure 13: Model Showing Confidence Score on Un-Trained Images

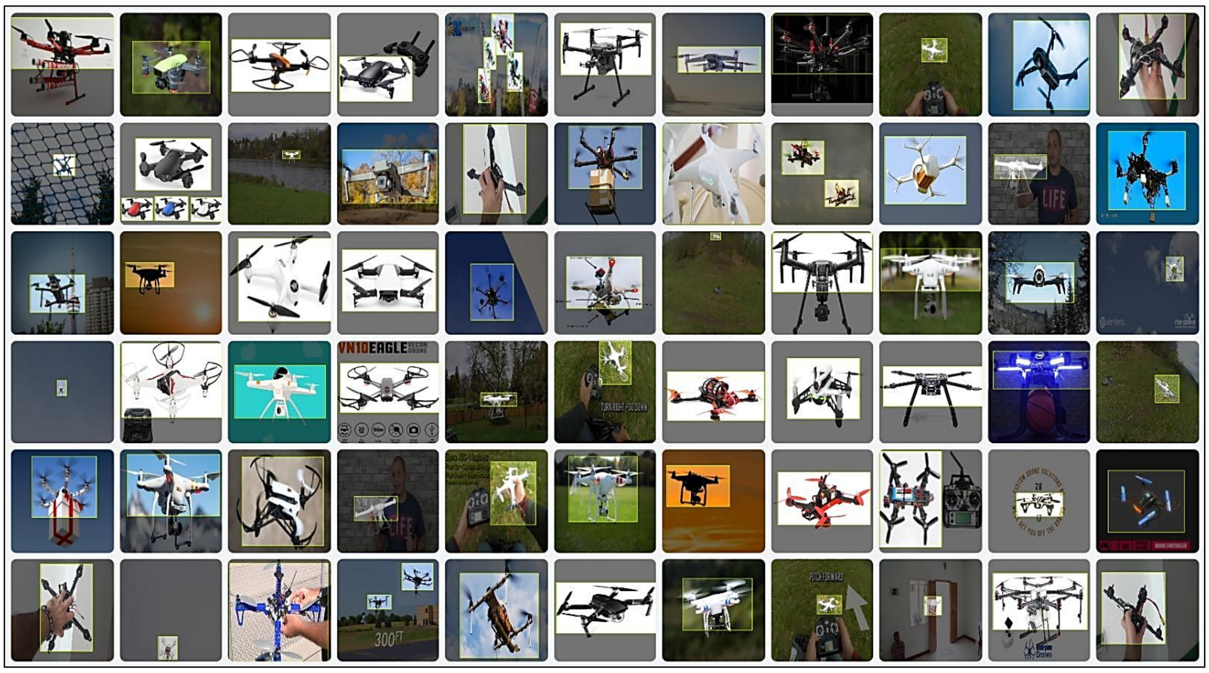


Figure 14: Multiple Result Images

System Robustness

To assess the system's robustness, we evaluated its performance under varying environmental conditions, such as changes in lighting and weather (23). The model exhibited resilience, maintaining high accuracy across diverse scenarios as seen in Figure 14.

Real-Time Warning System

In this example, our system triggers a real-time warning message upon detecting a drone within or near a predefined area as seen in Figure 15. This feature enhances the system's proactive capabilities, providing immediate alerts in potential security-sensitive situations.

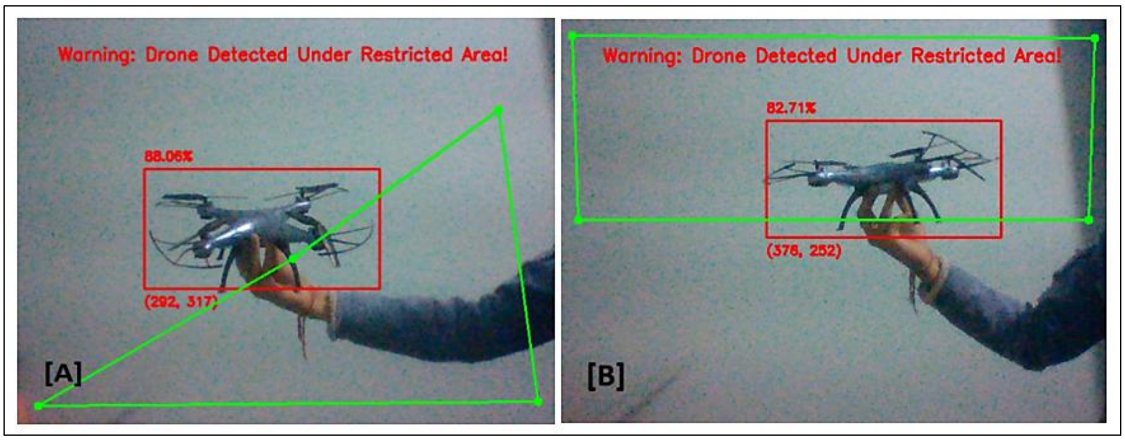

Warning: Drone Detected Under Restricted Area!
88.06%
(292, 317)
[A]
Warning: Drone Detected Under Restricted Area!
82.71%
(376, 252)
[B]

Figure 15: Real-Time Detection Program with Custom Field of View (FoV) (A, B)

Comparative Analysis

To contextualize our system's performance, we conducted a comparative analysis with existing state-of-the-art drone detection methods (24, 25). The results consistently demonstrated competitive precision, recall, and mAP, affirming the effectiveness of our approach.

Table 2 highlights the performance improvement of our YOLOv5 model over YOLOv4 and previous YOLOv5 implementations. The significant boost in mAP from 0.90 to 0.97 reflects the benefits of improved data diversity and optimized training settings.

Table 2: Comparison between Previous and Proposed Model's Performance

Models	Precision	Recall	MAP
Previous YOLOv4 (2)	0.950	0.680	0.743
Previous YOLOv5 (1)	0.918	0.875	0.904
Proposed YOLOv5	**0.958**	**0.964**	**0.972**

Table 3: Comparison between Other Methods and Proposed Mode's Precision and Proposed Mode's Precision

Reference	Method	Data Size	Results (Precision)
6	CNN	712	95.0%
6	SVM	712	88.0%
7	KNN	712	80.0%
7	YOLOv2	215	90.0%
8	MaskRCNN	1359	93.0%
Proposed	**YOLOv5**	**1440**	**95.8%**

Unlike traditional CV approaches relying on handcrafted features and classifiers (e.g., SVM, KNN), which require significant feature engineering and struggle with scalability, YOLOv5 offers end-to-end detection with real-time performance. Compared to recent detectors like Mask-RCNN, YOLOv5 achieves higher speed while maintaining competitive precision as seen in Table 3. Although newer models like YOLOv8 exist, YOLOv5 remains computationally efficient and is well-supported for deployment, justifying its selection. In a comparative analysis of existing methods and our proposed approach, our YOLOv5-based drone detection system outperforms established models. Notably, while traditional CNN achieved a precision of 95%, SVM and KNN lagged with precisions of 88% and 80%, respectively. YOLOv2, though efficient with a precision of 90%, utilized a smaller dataset (215 images). The advanced MaskRCNN exhibited a commendable precision of 93% with a substantial dataset of 1359 images. In contrast, our proposed YOLOv5 method excels with a precision of 95.8%, leveraging a diverse dataset of 1440 images, showcasing its superior accuracy in real-world drone detection scenarios.

Error and Misclassification Analysis

Despite the complexities of real-world environments, our model demonstrated minimal misclassification. Owing to the high precision (95.82%) and a carefully set confidence threshold, the system is conservatively tuned to avoid mislabeling. Instead of falsely classifying other objects (e.g., birds, clouds, or kites) as drones, the model opts to withhold detection unless sufficiently confident, thus prioritizing reliability over over-detection.

This approach significantly reduces false positives in cluttered or natural backgrounds. However, detection failures may occur under certain conditions:

- Poor lighting (e.g., dusk or backlighting).
- Low camera resolution or frame **rate** (especially <15 FPS).
- Tiny drone sizes occupying very few pixels in the frame (typically under 20×20 px).

These situations may cause the drone to be missed—not misclassified—highlighting the importance of proper deployment conditions (e.g., 1080p camera, stable lighting). Future improvements may include integrating multi-frame fusion or thermal vision to overcome these edge cases.

Conclusion

In the pursuit of advancing drone detection methodologies, our research has culminated in the development and validation of a real-time drone detection system. The systematic approach, encompassing data preparation, model training, and practical implementation, has yielded promising results with far-reaching implications for security and surveillance applications.

Integration with Threat Response Frameworks

Our system is well-aligned with the requirements of broader civil aviation and defence threat response mechanisms. It incorporates a real-time warning system, which activates when a drone enters a user-defined restricted zone—a region that can be dynamically adjusted based on the surveillance perimeter or sensitive area boundaries (e.g., airports, government facilities, or military installations).

When an intrusion is detected, the system:

- Displays a visual warning on-screen.
- Can trigger an audio or network-based alert (e.g., via siren or notification system) to immediately notify security personnel.
- Outputs bounding box coordinates, which can be used to calculate the approximate location of the drone within the field of view.
- These features form the basis of a modular real-time response pipeline. The system can be extended to:
- Integrate with radar or RF-based drone jammers.
- Send automatic alerts to command centers via HTTP/REST endpoints or MQTT.
- Feed data to tracking systems or external PTZ cameras for continuous monitoring.
- This modularity ensures the system's suitability for civil aviation zones under Remote ID policies as well as for defense-grade installations requiring scalable drone threat management solutions.

While our system performed exceptionally well, it is essential to acknowledge its limitations. Future iterations will focus on addressing challenges related to crowded scenarios, occlusions, and improving the system's adaptability to evolving drone technologies.

Our primary objective is to design a robust drone detection system capable of real-time identification and warning in diverse environments (26, 27). Through meticulous dataset curation, leveraging the power of YOLOv5, and integrating OpenCV for real-time processing, we addressed this objective with a comprehensive and practical approach. The system's detection accuracy, illustrated through visual results, showcases its ability to precisely locate drones within video frames. The integration of a real-time warning system further enhances its utility, providing immediate alerts in response to potential security threats. These achievements underscore the system's significance in bolstering security measures in sensitive areas, such as airports, public events, and critical infrastructure.

Quantitative metrics, including precision, recall, and mean Average Precision (mAP), underscore the high performance of our system. Comparative analyses against existing methods validate its competitive standing within the realm of drone detection technologies (28). The successful development and validation of our drone detection system hold significant implications for the field of security and surveillance. The system's real-time capabilities, coupled with its high accuracy, position it as a valuable asset in safeguarding public spaces, critical infrastructure, and events where unauthorized drone activities pose potential threats.

In conclusion, our research contributes to the field of drone detection with a practical and effective system. The amalgamation of advanced computer vision techniques, machine learning, and real-time processing heralds a new era in the proactive identification of drone threats. As the technological landscape continues to evolve, our system stands poised to play a pivotal role in fortifying security measures.

In the dynamic landscape of drone technology, our research serves as a stepping stone, contributing not only to the current state of the art but also laying the groundwork for future innovations in the realm of drone detection and security.

Abbreviations

CNN: Convolutional Neural Network, KNN: K-Nearest Neighbors, mAP: Mean Average Precision, MaskRCNN: Mask Regional Convolutional Neural Network, OpenCV: Open Source Computer Vision Library, SVM: Support Vector Machine, YOLO: You Only Look Once,

Acknowledgement

None.

Author Contributions

The authors have collaborated equally on all aspects of this study, including research design, data analysis, manuscript preparation, and revisions. They collectively accept responsibility for the integrity of the work and have given their approval for the final version.

Conflict of Interest

The authors declare that they have no competing interests.

Ethics Approval

Not Applicable.

Funding

This research was conducted independently by the authors and did not receive any external funding or grant support.

References

1. Aydin B, Singha S. Drone detection using YOLOv5. Eng. 2023;4:416–433.
2. Behera DK, Raj AB. Drone detection and classification using deep learning. In: Proceedings of the 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS); Madurai, India. 2020 May 13–15:1012–1016. https://ieeexplore.ieee.org/abstract/document/9121150/
3. Mishra A, Panda S. Drone detection using YOLOv4 on images and videos. In: Proceedings of the 2022 IEEE 7th International Conference for Convergence in Technology (I2CT); Mumbai, India. 2022 Apr 7–9:1–4. https://ieeexplore.ieee.org/abstract/document/9825244/
4. Mahdavi F, Rajabi R. Drone detection using convolutional neural networks. In: Proceedings of the 2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS); Mashhad, Iran. 2020 Dec 23–24:1–5. https://ieeexplore.ieee.org/abstract/document/9349620/
5. Al-Qubaydhi N, Alenezi A, Alanazi T, Senyor A, Alanezi N, Alotaibi B, Alotaibi M, Razaque A, Abdelhamid AA, Alotaibi A. Detection of unauthorized unmanned aerial vehicles using YOLOv5 and transfer learning. Electronics. 2022 Aug 26;11(17):2669.
6. Ulzhalgas Seidaliyeva, Lyazzat Ilipbayeva, Kyrmyzy Taissariyeva, et al. Advances and Challenges in Drone Detection and Classification Techniques: A State-of-the-Art Review. Sensors, vol. 24. 2023 Dec 26.
7. Aker C, Kalkan S. Using deep networks for drone detection. In: Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS); Lecce, Italy. 2017 Aug 29:1-6. https://ieeexplore.ieee.org/abstract/document/8078539/
8. Wu Q, Feng D, Cao C, Zeng X, Feng Z, Wu J, Huang Z. Improved mask R-CNN for aircraft detection in remote sensing images. Sensors. 2021 Apr 8;21(8): 2618.
9. Case EE, Zelnio AM, Rigling BD. Low-cost acoustic array for small UAV detection and tracking. In: 2008 IEEE National Aerospace and Electronics Conference; IEEE. 2008:110–113. https://ieeexplore.ieee.org/abstract/document/4806528/
10. Cheng Y, Ji X, Lu T, *et al.* DeWiCam: Detecting hidden wireless cameras via smartphones. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security. 2018:1–13. https://dl.acm.org/doi/abs/10.1145/3196494.3196509

11. Sciancalepore S. Detecting drones status via encrypted traffic analysis. In: Proceedings of the ACM Workshop on Wireless Security and Machine Learning; 2019:67–72. https://dl.acm.org/doi/abs/10.1145/3324921.3328791
12. Sciancalepore S, Ibrahim OA, Oligeri G, Di Pietro R. PiNcH: An effective, efficient, and robust solution to drone detection via network traffic analysis. Computer Networks. 2020 Feb 26;168:107044.
13. Conti M, Rigoni G, Toffalini F. ASAINT: A spy app identification system based on network traffic. In: Proceedings of the 15th International Conference on Availability, Reliability and Security. 2020:1–8. https://dl.acm.org/doi/abs/10.1145/3407023.3407076
14. DJI. AeroScope: Drone Detection System Shenzhen (China): DJI. https://www.dji.com/aeroscope
15. Ezuma M, Erden F, Anjinappa CK, *et al.* Micro-UAV detection and classification from RF fingerprints using machine learning techniques. In: 2019 IEEE Aerospace Conference. IEEE. 2019:1–13. https://ieeexplore.ieee.org/abstract/document/8741970/
16. Alipour-Fanid A, Dabaghchian M, Wang N, Wang P, Zhao L, Zeng K. Machine learning-based delay-aware UAV detection and operation mode identification over encrypted Wi-Fi traffic. IEEE Transactions on Information Forensics and Security. 2019 Dec 16;15: 2346-60.
17. Ganti SR, Kim Y. Implementation of detection and tracking mechanism for small UAS. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE. 2016:1254–1260. https://ieeexplore.ieee.org/abstract/document/7502513/
18. Kershner I. Israel builds a laser weapon to zap threats out of the sky. The New York Times. 2022 Jun 3. https://www.nytimes.com/2022/06/03/world/middleeast/israel-laser-rockets.html
19. Li W. Drone profiling through wireless fingerprinting. In: 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER). IEEE. 2017:858–863. https://ieeexplore.ieee.org/abstract/document/8446096/
20. Liu T, Liu Z, Huang J, Tan R, Tan Z. Detecting wireless spy cameras via stimulating and probing. In: Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services. 2018:243–255. https://dl.acm.org/doi/abs/10.1145/3210240.3210332
21. Trinh H, Calyam P, Chemodanov D, Yao S, Lei Q, Gao F, Palaniappan K. Energy-aware mobile edge computing and routing for low-latency visual data processing. IEEE Transactions on Multimedia. 2018 Aug 17;20(10):2562-77.
22. Bisio I, Garibotto C, Lavagetto F, Sciarrone A, Zappatore S. Improving WiFi statistical fingerprint-based detection techniques against UAV stealth attacks. In2018 IEEE Global Communications Conference (GLOBECOM).IEEE. 2018 Dec 9:1-6). https://ieeexplore.ieee.org/abstract/document/8647288/
23. Zhu P, Wen L, Bian X, Ling H, Hu Q. Vision meets drones: A challenge. arXiv preprint arXiv:1804.07437. 2018 Apr 20. https://arxiv.org/abs/1804.07437
24. Nassi B. Drones' cryptanalysis – smashing cryptography with a flicker. In: 2019 IEEE Symposium on Security and Privacy (SP). IEEE. 2019:1397–1414. https://ieeexplore.ieee.org/abstract/document/8835328/
25. Seufert M, Schatz R, Wehner N, *et al.* QUICker or not? – An empirical analysis of QUIC vs TCP for video streaming QoE provisioning. In: 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN). 2019:7–12. https://ieeexplore.ieee.org/abstract/document/8685913/
26. Soyata T, Muraleedharan R, Funai C, Kwon M, Heinzelman W. Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In2012 IEEE symposium on computers and communications (ISCC). IEEE. 2012 Jul 1 :000059-000066. https://ieeexplore.ieee.org/abstract/document/6249269/
27. Robin Radar Systems B.V. Robin Radar Systems: drone and avian radar solutions. Noordwijk (The Netherlands): Robin Radar Systems. https://www.robinradar.com
28. US Department of Transportation, Federal Aviation Administration. Remote Identification of Unmanned Aircraft. Final rule, 14 CFR Part 89. Federal Register. 2021 Jan 15. https://www.faa.gov/uas/getting_started/remote_id