Drone Detection and Classification using Machine Learning

Khurram Shafiq

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF APPLIED SCIENCE

in Electrical Engineering and Computer Science

Ottawa-Carleton Institute for Electrical and Computer Engineering

University of Ottawa

Ottawa, Canada

September 2023

Abstract

UAV (Unmanned Airborne Vehicle) is a source of entertainment and a pleasurable experience, attracting many young people to pursue it as a hobby. With the potential increase in the number of UAVs, the risk of using them for malicious purposes also increases. In addition, birds and UAVs have very similar maneuvers during flights. These UAVs can also carry a significant payload, which can have unintended consequences. Therefore, detecting UAVs near red-zone areas is an important problem. In addition, small UAVs can record video from large distances without being spotted by the naked eye. An appropriate network of sensors may be needed to foresee the arrival of such entities from a safe distance before they pose any danger to the surrounding areas.

Despite the growing interest in UAV detection, limited research has been conducted in this area due to a lack of available data for model training. This thesis proposes a novel approach to address this challenge by leveraging experimental data collected in real-time using high-sensitivity sensors instead of relying solely on simulations. This approach allows for improved model accuracy and a better representation of the complex and dynamic environments in which UAVs operate, which are difficult to simulate accurately. The thesis further explores the application of machine learning and sensor fusion algorithms to detect UAVs and distinguish them from other objects, such as birds, in real-time. Specifically, the thesis utilizes YOLOv3 with deep sort and sensor fusion algorithms to achieve accurate UAV detection.

In this study, we employed YOLOv3, a deep learning model known for its high efficiency and complexity, to facilitate real-time drone versus bird detection. To further enhance the reliability of the system, we incorporated sensor fusion, leading to a more stable and accurate real-time system, and mitigating the incidence of false detections. Our study indicates that the YOLOv3 model outperformed the state-of-the-art models in terms of both speed and robustness, achieving a high level of confidence with a score above 95%. Moreover, the YOLOv3 model demonstrated a promising capability in real-time drone versus bird detection, which suggests its potential for practical applications.

Keywords: YOLOv3, sensor fusion, deep neural network, real-time system, machine learning, and AI.

Acknowledgments

I would like to express my deepest gratitude to Dr. Iraj Mantegh from the National Research Council Canada (NRC) for leading this project and to NRC for funding this research project. Without NRC support, this study would not have been possible. I would also like to thank Dr. Varunkumar Mehta and Amer Alhalabi for their invaluable assistance with the software implementation of the real-time model.

I would like to express my heartfelt appreciation to my supervisor, Dr. Miodrag Bolic, for his guidance and support throughout the thesis writing process. His insightful feedback and advice have been instrumental in shaping this research work.

I would also like to extend my gratitude to my parents for their unwavering love, support, and encouragement. Their belief in me has been a constant source of motivation, and I cannot thank them enough for all that they have done.

Finally, I would like to thank all the team members who participated in this project. Their willingness to share their time and experiences has been critical to the success of this research project.

List of Abbreviations

UAV	Unmanned Airborne Vehicle
FOV	Field of view
CNN	Convolutional Neural Network
FPS	Frames Per Second
YOLO	YOU ONLY LOOK ONCE
ROI	Region of Interest
RCNN	Region-Based Convolutional Neural Networks
BBAF	Bird body axis filter
PODS	Probable observation data set
ML	Machine Learning
DL	Deep Learning
DVR	Digital Video Recorder
TCCA	Transport Canada Civil Aviation
ANN	Artificial Neural Network
IoU	Intersection over Union
mAP	Mean average precision.
TBD	Tracking by Detection
SIFT	Scale Invariant Feature Transform
HOG	Histogram of Oriented Gradients
RCN	Recurrent Correlation Network
FPS	Frame Per Second
CRF	Camera Radar Fusion
NRC	National Research Council
Dpi	Dots per inch
YOLO	You only look once.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

This Chapter explores the objectives of the thesis work by defining the problem statement and the milestones achieved by the thesis work. It also describes the research's motivation and the author's contributions in the provided research area.

1.1 Object Detection

Object detection is an essential branch of computer vision used to detect various instances of a particular object. The main objective of computer vision is to develop computational techniques and models to detect the required object at a particular time instance in a defined frame. Many advanced models have been developed to find the solution, and computer vision models have progressed drastically in the past 20 years.

From a research perspective, computer vision detection models have been divided into two major groups:

- General Object Detection: This aims to utilize aspects of different models to detect various categories of an object.
- Detection Applications: These are mainly focused on particular applications whose development is mainly concerned with specific models and certain types of objects.

The research is focused on the latter group in computer vision, where the main focus is drone detection and exploring different object detection frameworks to make the proposed model more robust and valuable under various rigorous scenarios.

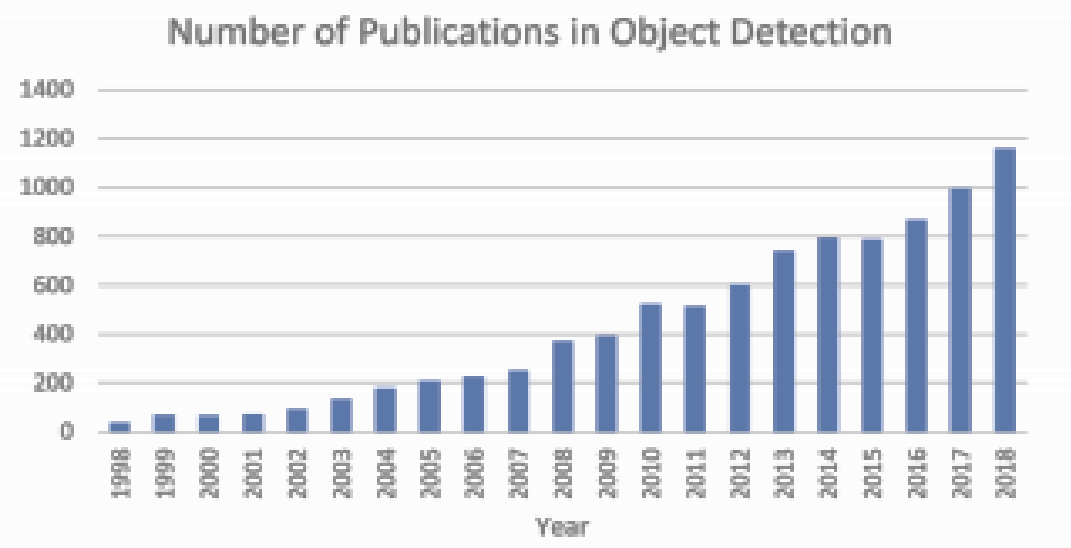


Fig 1- 1. The increasing number of publications in object detection from 1998 to 2018

Fig 1-1 shows the increasing number of publications on this subject matter in the past two decades. The data is retrieved from Google scholar advanced search titled “object detection” AND “detecting objects”. The state-of-the-art object detection systems from the early 1980s to 2019 drastically changed the scope of research. In recent years, a comprehensive analysis of these techniques helped many researchers investigate object detection models and improve further.

The progress was slow during the early 1980s, but there was a drastic change around 2012. The various state-of-the-art methods proposed are VJ Det. [1], [2], HOG Det. [3], DPM [4]–[6], RCNN [7], SPPNet [8], Fast RCNN [9], Faster RCNN [10], YOLO [11], SSD [12], Pyramid Networks [13], Retina-Net [14]. Today’s object detection is a field of artificial intelligence and deep learning. R. Girshick [9] proposed the R-CNN methodology in 2015 after a revolutionary invention of a Convolutional Neural Network (CNN). The introduction of CNN led to a significant development in object detection, and research work began to evolve rapidly.

1.2 Recent Developments in Object Detection Models

Convolutional Neural Network (CNN) is so far the most popular model to analyze images. Although image analysis is the most widespread use of CNN, it can also be used for various other purposes, such as data analysis and classification problems. The basic idea behind CNN is that it is a specialized deep neural network with various hidden layers called convolutional layers that identify patterns in a target image. This pattern detection capability makes CNN a helpful technique and is the basis for computer vision models.

On the other hand, R-CNN, a regional-based CNN, is a selective search technique that proposes candidate regions. This input is further reshaped and sent to convolutional layers. After R-CNN, significant improvements in the models were made, and we got faster object detection models like Fast R-CNN, Faster R-CNN, Fastest R-CNN, and various versions of YOLO models like YOLO-v1 to YOLO-v8. These are the significant improvements made on the total latency of Inference in CPU/GPU modes, average precision, frames per second retrieved, and many other result-based matrices.

1.3 Problem Statement

Unmanned airborne vehicles (UAVs) are omnipresent and gaining vast popularity in today's cutting-edge electronics for photography, delivery, and recreational industries. This increasing popularity and easy access will cause severe threats to some organizations and facilities such as airports. In this thesis, the author proposed an object detection models for drones using YOLO v3. The primary research questions for the research are:

1. Will the data augmentation technique help extend the dataset and methodically aid in improving the confidence score?

2. Will a deep sort algorithm help to locate objects with reasonable confidence under occlusion, fog, dust, and bad weather conditions?

3. Can sensor fusion logic be implemented for a limited dataset without having time synchronization between radar and optical sensors?

4. Will the distance between the sensor and the detected object affect the confidence scores during detection?

5. Will the object detection model (i.e., YOLOv3) be able to detect objects in near real-time without significant computational delays?

1.4 Motivation

Remotely controlled UAVs are gaining vast popularity because of their usefulness in society, as discussed briefly by Sanfridsson et al. [14]. Also, the drones' availability and accessibility for public recreational purposes could pose a risk of presence in some sensitive red zone areas.

Guvenc et al. [15] have described threats from unauthorized drones, problems caused by amateur drones, and the pros and cons of various sensors for drone detection. It was observed that optical sensors have a low cost of operation. Also, the availability of these optical sensors in different areas would allow one to take advantage of models such as object detection, etc. A few limitations of these optical sensors might be environmental conditions such as fog, dust, and occlusion, which require some maintenance.

Detection of UAVs is crucial to some authorities, but misclassification with birds could mislead the detection model and result in problems for the organization in question. Also, the maneuvers between birds and drones are not remarkably different and could pose a risk of false detections. The availability of radars with optical sensors will allow us to use sensor fusion logic to verify the detection of the required object with more precision than sole radar or sole optical detection.

A comprehensive review of drone detection and classification using machine learning is performed by Taha & Shoufan [15], discussing various modalities. One of the key points stated in this paper was the limitation of the available datasets to train models to perform drone detections. Also, not many papers discuss the detection range, the dataset used in their training, and the performance and utilization of the range while detecting the required object. Understanding these factors can be instrumental in improving detection methodology.

A detailed literature review is discussed in Chapter 2, but first, it is essential to have a high-level understanding of the subject matter, which tends to motivate the researcher to work in this particular research area. It has been observed that many papers have been published where the authors implemented drone detection utilizing the deep neural network methodology (i.e., convolutional neural network). However, these papers failed to share specific information regarding detection methodologies, such as features related to distance, range, resolution, etc. Also, although many different models have been implemented in this research to detect drones, there is a need to see which models will perform better in real-time. A few observations from the extensive literature review will provide an understanding of the problem statement. It was observed that the results from the literature review are mainly derived from minimal data. Additionally, the analysis was conducted on mostly simulated data.

Although UAVs have been a trendy and exciting research area, sufficient efforts have not been made to produce the results on a real-time dataset. Also, not many real-time applications have been deployed with high confidence. The solutions shared in the papers mentioned above seem unsatisfactory, and it is apparent that all these solutions do not apply to all kinds of drones. The reason is that not much data was shared, and the results obtained do not consider all varieties of birds and drones. Features like altitude, height, and range were also very limited in small samples of the dataset, and it was not taken into much consideration. The presented results by the previous researchers proved to be good enough with high accuracy. However, they were uncertain whether the models they implemented would be able to perform well in real-time. The reason is that few satisfactory results were provided with a wide variety of datasets in real-time.

1.5 Aims and objectives.

The following are some primary objectives of the thesis work:

- Distinguish drones v/s birds based on object detection algorithms in real-time.
- Implement a sensor fusion logic for non-synchronized data in real-time.
- Implement a real-time system to fuse estimates from different machine learning algorithms.

The following are the steps required and the contributions to meet the above objectives:

Data Collection & Augmentation:

- Prepare sufficient data for training and then utilize different data augmentation techniques to extend the dataset.
- Explore the steps needed to overcome the limitations in the dataset for deep neural networks.
- Manual annotations were required to localize the ground truth for training purposes.

Object Detection Model:

- Analyze limitations in object detection models (namely, the detection of drones mentioned in the literature review) and then suggest a suitable model for a real-time application.
- Implemented a trained YOLOv3 model for a custom dataset to classify drones and birds confidently.
- Implement a Faster R-CNN to compare the results obtained from YOLOv3.

DEEP-SORT (Appearance Matrix):

- Implement a deep sort algorithm and integrate it with YOLO v3 to improve the accuracy under occlusion and environmental conditions and to track the small-size objects in each frame.

Sensor Fusion:

- Implement a sensor fusion algorithm by fusing optical and radar confidence scores for non-synchronized and limited data.
- Implement a weight-based methodology, i.e., sensor fusion incorporating different features like range, height, etc., with high precision and accuracy.

Real-Time Detection Model:

- Develop a real-time system to accept and respond to the data using TCP socket streams.

- Optimize the performance of a deployed model to work near real-time.

1.6 Brief description of the proposed method

This subsection will provide a brief overview of the work that will be detailed in the following Chapters. The research explores the potential of detecting drones and birds using YOLOv3 with deep sort and sensor fusion algorithms in real-time. This work uses the YOLOv3 model, a highly efficient neural-based model with low complexity, to assist real-time drone vs. bird classification. Also, using sensor fusion, the system was made more robust than was possible with any individual sensors. It was observed that utilizing the model in sensor fusion made the real-time system more accurate.

The research will also look at recent developments in drone detection and provide a few scenarios and results in real-time systems. Fig 1-2 shows the proposed methodology and a traditional object detection approach. A few papers [16], [17] included the data augmentation approach in object detection algorithms but were limited to the results in real-time scenarios. Few researchers shared content about deep sort algorithms, including various matrices like distance and appearance for real-time drone detections. Also, there is a need to look at the approach where the training dataset is limited and is not synchronized with other sensors like radar and optical. There is a need to optimize traditional object detection algorithms to deal with real-time situations. A few weights have been added in various features like turn rate, optical resolution, and height in both radar and optical sensors while fusing scores from both sensors. In the end, looking at features like range, resolution, and their effects on final probability scores will be interesting. Many such features have been added to the final sensor fusion logic to improve the confidence score and make the model more robust.

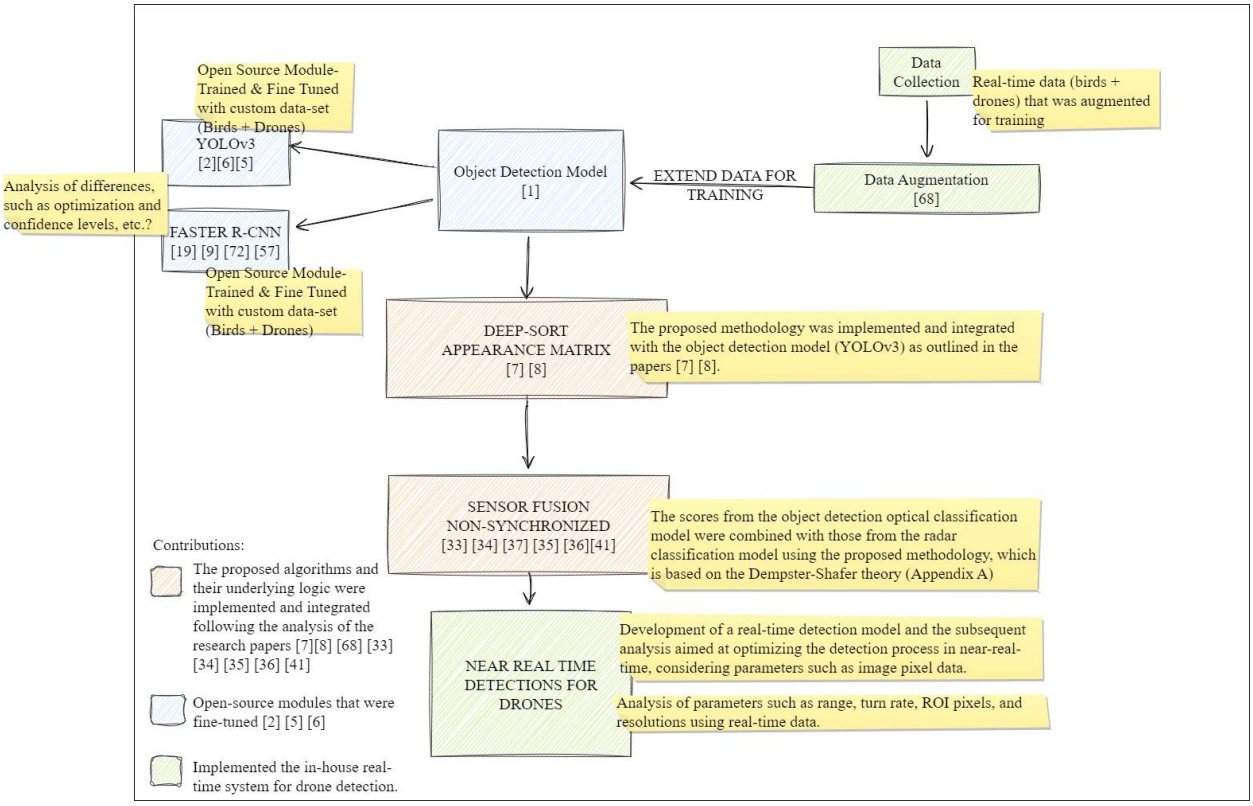


Fig 1- 2. The proposed methodology of traditional object detection models and the potential to improve the methodology with the addition of a few recent algorithms and approaches.

The researchers thoroughly compared convolution object detectors for real-time drone detection using PTZ-Camera. Six models were observed and compared with their detection accuracy and speed. Table 1-1 discusses the potential of each model mentioned in [27].

Table 1- 1. Computational requirements of object detection models [27]

Model	**Time taken per image**	**FPS**
SSD Mobile Net	0.048	20.8
SSD Inception V2	0.084	12.0
RFCN Resnet 101	0.320	3.1
FRCNN Resnet 101	0.423	2.4
FRCNN Inception Resnet	1.455	0.7

YOLOv2	0.077	13.0

One of the few challenges observed while comparing all these models [27] was detecting small drones, as drones located further may fall outside the camera's field of view. The reason is that the zooming power is insufficient to keep track of some drones beyond the 700m range.

It was observed that a deep neural-based network mentioned by Redmon [11] proves to be an excellent potential candidate based on its fast detection and accuracy. YOLOv3 with object detection performs well in most scenarios but has some limitations and requires further processing to make it perform well in real-time scenarios. It requires tracking in each frame which can help improve the overall score and the model. In the research, a deep sort algorithm [18] is used with Kalman filtering to help us overcome detection limitations and utilize prior information to track objects more accurately than traditional tracking algorithms such as optical flow, template matching and correlation-based tracking.

When dealing with real-time world scenarios, it might not be straightforward to deal with occlusion and blurriness with solely a distance matrix since it does not include information regarding the appearance of the object itself. Hence, we must introduce another factor to combine motion and appearance information.

The limited dataset was one of the major problems encountered while training the required deep neural network. Data augmentation plays a crucial role in such circumstances, which significantly helps improve object detection by increasing the volume of a dataset. A remarkable change was observed in accuracy after applying the data augmentation technique to the required dataset. Zoph et al. [19] explained some crucial data augmentation techniques in their proposed model.

In an image domain, data augmentation techniques translate images by a few pixels and adjust the contrast, exposure, gamma-bright, hue, saturation, brightness, and darkness. The techniques were also used to deepen the image pixels, and after applying a grey tone, it increased the dataset size. This increase in the dataset size assists our model in performing training in all possible scenarios given different augmentation levels. Also, it is noted that this increase in the dataset size will not increase the annotation task while training, as the required

object, was not rotated in a few augmentation methods. Only geometrical transformation would require one to update the annotation task, which can easily be taken care of without manual annotation.

In this thesis, the proposed model has acquired another sensor (radar) to do object detection utilizing Interacting multiple models (IMM) radar tracking [20]. Feature extraction was done to extract the potential region of interest for the optical model with high accuracy and precision. It is to be noted that the proposed model on the radar side is also providing the confidence score, which can be used to affirm the final accuracy and precision using sensor fusion. We have implemented the weighted scheme to incorporate fewer factors, i.e., turning angle (to complement radar), range, and bounding boxes (to complement vision sensor).

1.7 Thesis organization

This thesis is organized into six Chapters. Chapter 1 overviews the problem statement, limitations, primary objectives, and contributions. Chapter 2 contains an extensive literature review and insights into object detection algorithms. The chapter will highlight various limitations and propose a methodology for our application based on various factors. Chapter 3 contains a description of data collection and various data augmentation techniques developed to enhance the required dataset. Chapter 4 explains the methodologies employed in this research to answer the research questions posed in the problem statement earlier in this Chapter. It is followed by a system architecture and various algorithms and models incorporated to enhance the system's performance. It also includes the graphical user and system interface. Chapter 5 analyzes the performance of a defined model and problems with hardware. Also, it shares the results based on various flights and experiments conducted in different scenarios. Chapter 6 summarises the research, its contributions, challenges, and future directions.

Chapter 2. Literature review

2.1. Introduction

This section provides an introduction and a literature review of computer vision and deep learning for object detection algorithms. The research of a conventional and deep learning approach for the detection algorithms is presented. In addition, we will also give a brief introduction to the operational hardware used in the thesis work, related work on data collection for UAVs, performance metrics for object detection, and an overview of sensor fusion based on different levels.

2.2. PTZ Cameras

A pan-tilt-zoom camera (PTZ) is a specialized camera operated by a digital video recorder (DVR) that can change directions and zoom based on commands. PTZ cameras can cover a larger area and field of view than stationary cameras and can have more flexibility to cover 360 degrees while moving horizontally and vertically. The PTZ camera's protocol information is set using the DIP switches inside the camera. Table 2-1 shows several basic terms related to PTZ protocols that are important while working with these cameras. The camera's PTZ protocol information allows a DVR to communicate with it and control the camera's movement.

Table 2- 1. Terms related to the PTZ protocol.

ID	Each PTZ camera has a specific ID to be operated by DVR.
PROTOCOL	It is a language between a DVR and an optical sensor.
BAUD RATE	It is a frequency of communication.

2.2.1 Bosch camera

Bosch camera (Version: MIC IP STARLIGHT 7000I) has been used to conduct all experiments and demo flights. The Bosch camera is an advanced PTZ camera for robust and high-quality imaging needs. Moreover, the camera has an intelligent tracking speed of more than 0.2 degrees/second. It also covers 360 degrees of continuous rotations with high-definition resolution. The camera has many protocols, mainly UDP, TCP, etc. Also, the PTZ camera has 12x digital zoom and 30x motorized zoom to locate an object more than a km away, sufficient for the project's needs. The Bosch camera is shown in Fig 2-1.

Fig 2-1. BOSCH camera

2.2.2 Radar

Radar uses radio waves to determine the distance (range), elevation, angle, and velocity of an object (drone and birds). The initial experiments were conducted with QinetiQ radar, but due to several issues with the accuracy, the subsequent experiments were conducted using Echodyne radar.

The OBSIDIAN Radar collects two samples per second. Due to the low sampling rate and frequent losing the radar tracks, the QinetiQ data were considered unreliable for building the classifier. For this reason, the former radar has been replaced with the Echodyne radar. The Echodyne radar has a higher sampling rate of 10 samples per second and good target detection.

2.3. Related work on dataset collection for UAVs

Flying a drone is legal in Canada, according to Transport Canada Civil Aviation (TCCA), but you must follow several drone regulations listed on their official website. That gives a limitation in collecting dataset for researchers who are not able to fulfill all regulations. Experiments for data collection are also hampered by 'no drone zones', which are areas where it is either illegal or unsafe to fly drones.

To our knowledge, researchers did not show the resultant matrix for object detection with a distance (range) of 500 m and above other than collecting a UAV dataset for the sole purpose of image classification.

Zheng et al. [21] studied the problem of air-to-air UAVs object detection using monocular cameras. In this study, authors shared the dataset, i.e., Det-Fly containing more than 13000 images of a flying UAV. Images were taken from another flying UAV. The authors also contributed to analyzing different image attributes with UAV detection. The relative distance of the UAV from the optical camera was between 10 m to 100 m. So all results were obtained with a distance (range) below 100m. While evaluating the resultant matrix from different object detection algorithms, authors recorded the YOLOv3 performance on UAV detection to be 72.3%. Grid R-CNN performed the best, with an 82.4% score.

Dataset, i.e., Det-Fly, contains images at the top, bottom, and at front view in 4 different environmental backgrounds. Most of the dataset contains objects smaller than 5% of the total image size.

In this thesis, we will later show that detecting smaller objects (less than 10% of the image size) is a very challenging task. Thus, the proposed method is designed to handle this situation by fusing scores from radars. Results will be shared in the result section of Chapter 5. Authors in [22] also proposed a more comprehensive dataset to detect objects with drones other than

DJI Mavic. They also reviewed the results based on another dataset, i.e., MIDGARD in [23]. According to the authors, MIDGARD is the latest comprehensive dataset designed for deep-learning object detection algorithms. The relative distance recorded with MIDGARD was even less than 20m.

Svanstrom et al. [24] proposed a multi-sensor drone detection system. The dataset contains 650 annotated videos of birds, drones, airplanes, and helicopters. The relative distance from the sensor to the target was recorded as less than 200 m, split into close, medium, and distant bins. Authors trained YOLOv2, and precision was calculated at around 82% for UAVs.

Coluccia et al. [25] proposed three deep-learning algorithms for detecting birds and drones. Under the 2020 Drone v/s Bird Detection Challenge [25], a group of academic institutions collaborated to compile and release the following dataset. The training set contains 77 videos comprising 1384 frames of birds and drones. In 2020, 45 more videos were added at three different resolutions. The paper did not mention the relative distance from the target to the sensor. The author shared that a large section of the data containing objects with less than 5% of the image size, which is very challenging in detection tasks. There were 23 teams in the challenge; none shared the result matrix with a distance (range). Fig 2-2 shows several images from the dataset included in the challenge. It can be observed from Fig 2-2 that the images were not zoomed in on the object, and objects were 5% and less of the total image size, as discussed above. Also, there are some challenges associated with occlusion as few of the images have trees and clouds that need image masking, etc., to extract the objects in these scenarios before doing the object detection.

Fig 2- 2. Dataset provided in [25].

Pawelczyk & Wojtyra [26] created a real-world object detection dataset in real-time. The authors also pointed out that there is no available object detection dataset containing many UAVs. The UAV dataset created contains more than 50,000 images at a resolution of 640 x 480 in different types and sizes of UAVs under different environmental conditions. The relative distance from the sensor to the target was not discussed. They also implemented a custom artificial neural network and Haar Cascade in OpenCV for object detection. Also, an automatic labeling strategy was proposed in Fig 2-3 to save 50% of tagging time. The left side of the Figure shows the conventional approach to preparing the dataset for the detection model, where most of the time has been consumed (15-30 seconds per image) in annotating and labeling the dataset. The right side of the Figure shows an approach to save tagging time with automated labels. They used the artificial neural network (ANN) to automate this labeling

process by acquiring the labeled data, training it through the ANN, and then creating more labeled data using this ANN based on the inference of the resultant footage. The author claimed that this semi-auto labeling process reduced the workload by 293 hours.

2.3.1 Performance Metrics for Object Detection

This section proposes an overview of a set of metrics and algorithms for evaluating the performance of object detection and tracking systems. The object detection model has two main output parameters: the location of an object and the classification of the same object. Depending on the number of objects and classification, one can expect more than one bounding box with their classification accuracy. A good object detection model can localize all objects with good confidence scores.

There are various deep learning object detection algorithms like Fast RCNN, Faster RCNN, YOLO, SSD, and Mask RCNN. In these models, authors used various evaluation metrics. One such parameter is the concept of intersection over union (IoU). An IoU of 1 means that the predicted and ground-truth bounding boxes overlap perfectly. So if an object detection model is defined with a threshold value, say, 0.5, then during classification, the following will occur:

- All actual objects with IoU greater than and equal to 0.5 will be classified as true positives (TP).
- All actual objects with IoU less than 0.5 will be classified as a False positive (FP).
- If a model fails to classify the object with a presence of ground truth, it is classified as a False Negative (FN).
- All other objects without a ground truth are classified as True Negative (TN).

$$IntersectionOfUnion = \frac{AreaOfOverlap}{AreaOfUnion} \quad (2\text{-}1)$$

Precision and Recall are other evaluation parameters calculated using true positives, false positives, and false negatives.

$$Pr\,e\,cision = \frac{TruePositive}{TruePositive+FalsePositive} \quad (2\text{-}2)$$

$$Re\,c\,all = \frac{TruePositive}{TruePositive+FalseNegative} \quad (2\text{-}3)$$

Precision and recall are calculated for all the objects in the given frame. Confidence score, the probability of a particular object in a given frame, is one of the significant evaluation parameters after the softmax layer in a given object detection model. Mean average precision (mAP) is another parameter calculated over the entire dataset: the arithmetic means of the precision at each recall level in the testing dataset [27].

Many improvements in computer vision have come from various competitions hosted around conferences. The first significant object detection competition in computer vision was "The PASCAL Visual Object Classes Challenge 2007". In this competition, mean average precision was used to evaluate each project. Mean average precision was defined in that competition as the area under the precision-recall curve at 11 discrete recall values by taking a mean of the max precision value of all the classes. In 2012, another competition, "Visual Object Classes Challenge 2012 (VOC2012)," made a slight change in evaluating average precision at all unique recall intervals rather than just 11 discrete intervals used in the competition in 2007.

Redmon & Farhadi [28], for YOLOv3 and many other object detection algorithms, authors took motivation for evaluation metrics from coco dataset competitions [29]. In false detections, the average precision can be penalized through IoU, but it does not penalize the localization aspect if the prediction bounding boxes do not cover the whole object.

Fig 2- 3. Object detection: Red bounding box indicates the object is not entirely detected, while the blue indicates the complete tiger is detected. Source: https://en.wikipedia.org/wiki/Tiger

Take an example of a tiger as given in Fig 2-3; we can see the object detection model could not predict the whole region of the classifying object. The tail that was not predicted could be the potential region for the application. The model may be trying to classify different races of a particular tiger, and the tail would be an essential factor. In such cases, we should penalize the model's average precision. In coco competitions [29], another approach was introduced to calculate the correct average precision by changing the IoU over an interval and then averaging the outcome. AP (50) is computed over thresholds [0.5: 0.95: 0.05] and then averaged.

2.3.2 Localization Recall Precision

The approach shared above considers the localization aspect but does not fully penalize the average precision in case of false detections. In 2018, Okuzu et al. [30] proposed another metric to evaluate this matter called localization recall precision.

The approach introduced three components, i.e., LRP(FN), LRP(FP), and LRP(IoU).

$$LRP_{FN} = 1 - \mathrm{Re}\,call \tag{2-4}$$

$$LRP_{FP} = 1 - \mathrm{Pr}\,ecision \tag{2-5}$$

$$LRP = \frac{1}{N_{TP} + N_{FP} + N_{FN}} N_{TP} \frac{1}{1-\theta} LRP_{IoU} + N_{FP} LRP_{FP} + N_{FN} LRP_{FN} \quad (2\text{-}6)$$

Moreover, finally, LRP is the weighted normalized sum of all three components given below:

$$LRP = \frac{1}{N_{TP} + N_{FP} + N_{FN}} N_{TP} \frac{1}{1-\theta} LRP_{IoU} + N_{FP} LRP_{FP} + N_{FN} LRP_{FN} \quad (2\text{-}7)$$

LRP is computed for each confidence threshold from 1.00 to 0.00, and then a curve is plotted against each value.

Oksuz et al. [30] have suggested the optimum point using the LRP curves, the lowest point in the curve, giving the ideal confidence threshold score.

2.3.3 Overview of tracking algorithms in object detection

Object tracking is an essential domain of computer vision and image processing. It involves tracking an object in each frame across all sequential frames. The tracking algorithm works by detecting all possible objects in each frame. Each classifying object will be given an 'Id' corresponding to each frame. If an object stays within a given frame, the given id of an object will be compared with some metrics. And then, the tracking algorithm will decide whether it is the same object with the same id.

Object tracking is challenging, especially under occlusion, as depicted in Fig 2-4 by Xing et al. [31], scale change, background clutter, appearance change, and bad weather conditions. Also, when there are multiple classifying objects within a single frame, tracking might be a problematic task across a series of frames, as evident in Fig 2-5. The red bounding boxes in Fig 2-6 indicate the object detected and its unique ID represented by A, B, C, ….., H. The unique ID is provided for tracking the object later in the video frames. Fig 2-5 shows that the person having IDs F, G, and H are occluded as F is hidden by G and H.

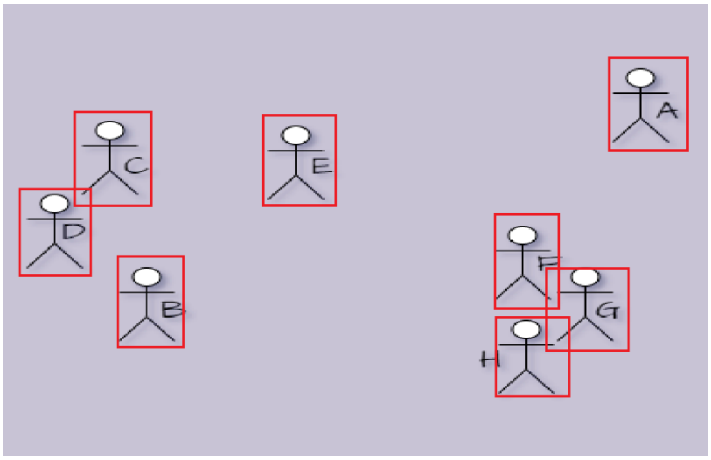


Fig 2- 4. Object occlusion source [31]

Khachatryan [32] implemented object tracking using the centroid-based assignment. It works by calculating centroids for each classifying object in a given frame. In the next frame, centroids are calculated for each bounding box again, and then based on a relative distance from the previous centroids, and ids were assigned to each object. The approach works well when objects are not too close together. Otherwise, it would switch IDs across objects. Fig 2-5 gives an overview of how the centroid-based algorithm works.

The centroid tracking algorithm is a multi-step process where the algorithm starts by accepting bounding boxes and computes centroids on each box. Step 1 accepts bounding box coordinates and computes centroids. Step 2 computes a Euclidean distance between new bounding boxes and existing objects. Step 3 will update existing objects' (x, y) coordinates. The new objects are registered in the last step, as shown in ID1, ID2, and ID3. Step 4 registers new objects.

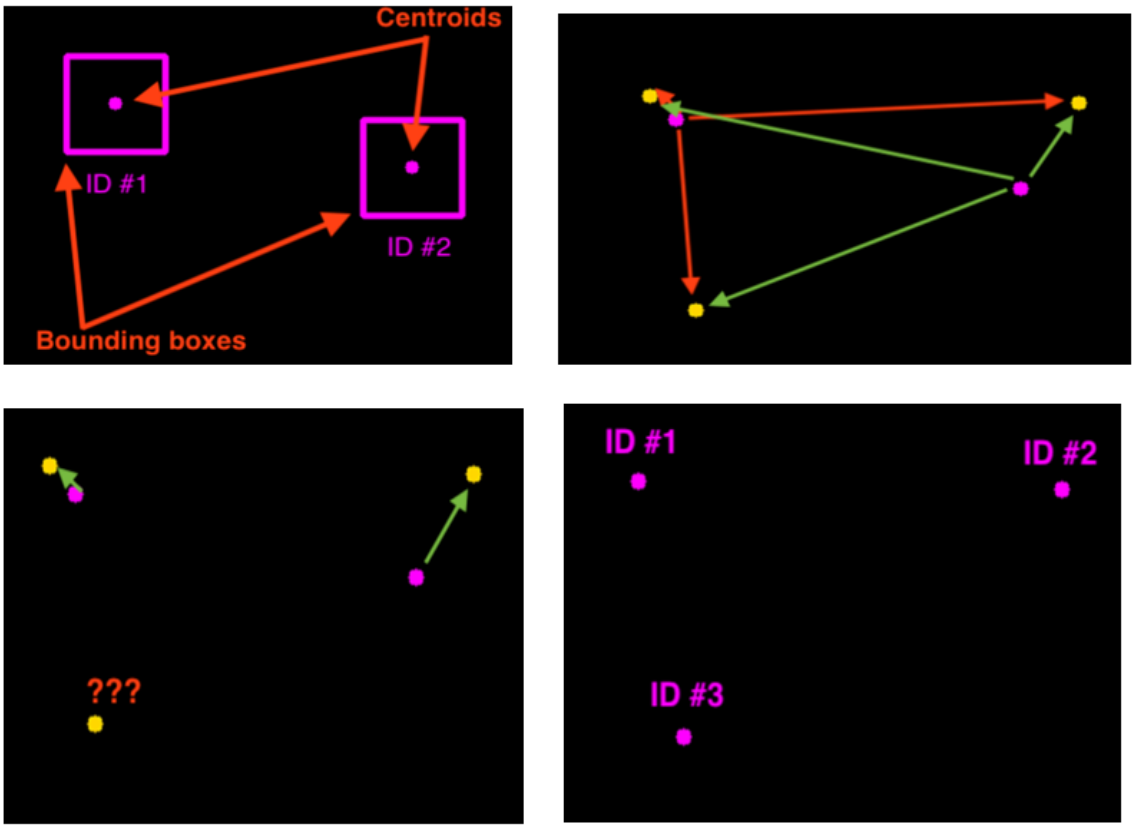


Fig 2-5. Centroid-based ID assignment, Source [32]

Kalman filtering is an improvement to the centroid Id assignment, which predicts the position of an object on the next frame based on the velocity and position of an object in a given frame. Bzarg [33] explained the Kalman filtering algorithm in detail. The author explained that the Kalman filtering algorithm models future positions and velocities using Gaussian distribution. The algorithm predicts measurements on each data sample and then updates itself based on correct readings. It takes into account the position and velocity of an object in real-time, providing better results than centroid-based assignments.

The deep sort algorithm [34] is one of the object tracking algorithms in which we track objects by using not only distance and velocity but also based on the appearance of a classifying object. Deep Sort implements deep appearance features for every object in a given frame. Then, using the similarity metrics between these features, we can introduce them to our tracking logic. Wojke et al. [18] trained an object detection model on a large set of human images and then extracted the 128-dimensional vector for each classifying object, which will

carry all the critical features of the given object. This approach will help if the objects are very close together or occluded, as shown in Fig 2-4.

One of the significant drawbacks of using the above tracking algorithms is the computational complexity which harms the number of tracked frames per second. Kaputa & Landy [35] proposed an efficient tracking algorithm for edge-based applications. The authors proposed that the most obvious way to do object tracking is through tracking by detection (TBD) on each frame. The author also concludes that this approach is inefficient, especially when an object is occluded. An algorithm like deep sort has been implemented to overcome these drawbacks, which takes another metric, i.e., appearance metrics, to track an object. These algorithms are computationally expensive for edge computing applications. YOLBO, an algorithm the author proposed, was initially run on 1070 TI GPU and then through TX2. The author included latency and the frame rate as primary evaluation metrics.

Moreover, the proposed algorithm uses ROI-to-centroid tracking and motion vector pre-calculation routines to reduce computational complexity. YOLO was used for object detection. A 2D vector represents points' motion from one frame to the next, and the optical flow was added to the methodology to account for this. The authors proposed the optical flow with an approach to look back on the previous information to reduce the latency rate by skipping frames. The proposed methodology has a few limitations. Thye work produced has a significant impact on accuracy. It was observed that accuracy was below 50% for nine skipped frames with a latency rate of 2.60 ms. This significant decrease in accuracy in the proposed algorithm might not be appropriate for some applications, but it can be employed in many edge computing devices with a trade-off between accuracy and latency rate.

2.3.4 Related Work on Object Detection Algorithms

Zheng et al. [1] proposed an experimental evaluation of deep-learning object detection algorithms for UAVs. The authors discussed that an existing UAV detection algorithm follows the following two streams.

- Using a traditional computer vision strategy by extracting features using Scale Invariant Feature Transform (SIFT) or Oriented FAST and rotated BRIEF (ORB). These extracted features are then sent to a machine learning algorithm to classify based on the application.

- Another approach is the deep learning-based approach, such as convolutional neural networks. In this approach, input is sent through an artificial neural network to output detections of a complex object.

2.3.5 Conventional Approaches to Detect UAVs

Gokce et al. [36] proposed a conventional computer vision approach to detect UAVs. The authors extracted the features based on Haar-like features, histograms of gradients, and local binary patterns, which were further classified using boosted classifiers. Near real-time detections on indoor and outdoor datasets were recorded using local binary patterns with cascaded classifiers.

Sapkota et al. [37] proposed an online detection of small UAVs in the 3d environment of a single moving camera. Object detection was done using HOG (Histogram of Oriented Gradients), which consists of ensemble learners via the AdaBoost algorithm. The proposed frameworks did detections with real-time tracking using the 2D information in the image plane, translated into the 3d positions and velocity of a target UAV.

Li et al. [38] use the optical flow matching mechanism to extract spatial and temporal features of a target UAV. The authors proposed the detection scheme by:

- Estimating the background motion using a perspective transformation model.
- Identifying the distinct points in the background.
- Extracting spatial and temporal features of the moving objects using the optical flow.
- Classifying the target regions based on their movement patterns.
- Boosting the model with the Kalman filter tracking.

Minaeian et al. [39] proposed a detection approach for UAVs. The camera motion in this paper was estimated using pyramidal Lucas-Kanade [40]. Foreground image segmentation was used with perspective homography to extract the features using the sliding window approach. The author concluded that the proposed algorithm would be efficient and robust for real-time applications. Opromolla et al. [41] proposed morphological transformations with

template matching for UAV detections. Their research estimated the results with an accuracy to the order of one pixel, which lies above 85%.

2.3.6 Deep Learning Approaches to Detect UAVs

Zheng et al. [1] stated that deep learning-based object detection models had made significant progress in object detection, but the area has not been explored much with UAV detections. In 2014, Girshick et al. [7] proposed an efficient object detection algorithm by extracting 2000 selective regions. These 2000 regions were generated using the selective search algorithm. These 2000 regions were then fed into a convolutional neural network to produce a 4096-dimensional feature vector as an output. The CNN acted as a feature extraction algorithm which was then passed through an SVM to classify the presence of an object. The proposed algorithm was very slow and took more than 47 seconds to classify 2000 proposed regions. Also, the selective search algorithm was not satisfactory for all applications.

The same author solved some drawbacks in the proposed methodology [9], and in 2015 another approach was published, Fast R-CNN. In this approach, instead of feeding proposed regions to CNN, an input image was passed through the CNN to generate a convolutional feature map. From the feature map, the authors identified the proposed regions, and then by using an ROI pooling layer, it was reshaped and passed through a fully connected layer. A softmax layer was used to predict the class of the proposed regions. The proposed approach was faster as only the input image has to be passed through a CNN layer once instead of all 2000 proposed regions.

R-CNN and Fast RCNN utilize selective search to generate proposed regions, which is a slow approach. In 2016, Ren et al. [10] proposed a Faster RCNN without using selective search algorithms. Instead of using selective search, a separate network was used to generate the proposed regions. The comparison of various algorithms based on test speed is given in Fig 2-6. From Fig 2-6, the Faster R-CNN outperforms other state-of-the-art algorithms by taking less computational time to perform inference.

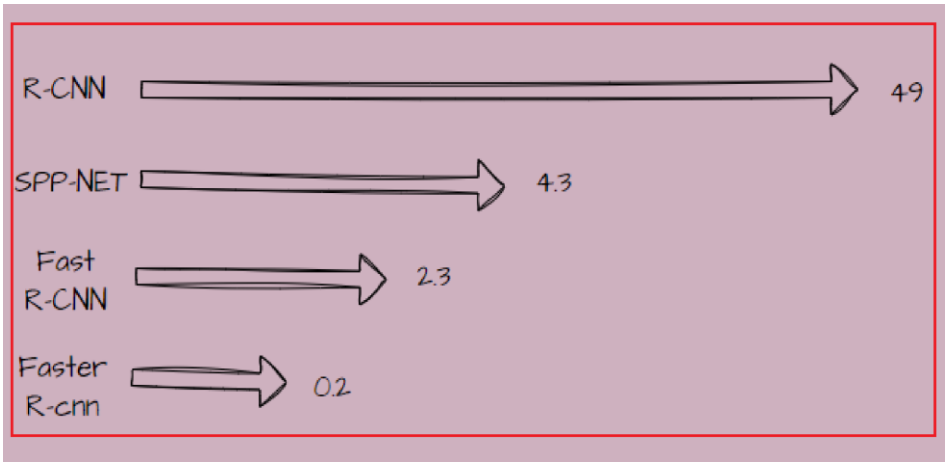


Fig 2-6. Comparison of test time speed of object detections [42]

These models used proposed regions to localize the objects within a given frame. In 2016, Redmon et al. [11] proposed YOLO (You Only Look Once) algorithm without using a separate network for the proposed regions. In this approach, a single CNN could predict the bounding boxes and the class probabilities. It splits the image into 'n' grids and 'm' bounding boxes. Each of these grids was then passed through a CNN, which predicts the class probabilities. Then, we can extract the required object using a simple filter approach on a given threshold. A-Max suppression layer was used to filter the close bounding boxes.

The more recent version of YOLO is YOLO v5. Many researchers are utilizing it for the detection of drones. Al-Qubaydhi et al. [43] presented an automated image-based drone-detection framework that guards against drone incursions in restricted areas by employing YOLOv5. The transfer learning to pre-train the model was employed because the dataset does not have enough samples to improve its performance. On top of that, the proposed model can identify the object in question across multiple images and label its bounding box by summing the results from each frame. The experiments produced remarkable results in terms of the drone's recall and precision, as well as its loss value and location detection.

Similarly, Jung & Choi [44] has proposed an enhanced version of YOLO v5 as an alternative to the standard YOLOv5 model. The dataset applied is based on photographs taken under various conditions, including those where the drone's altitude was altered, where there was no available light, and so on. The F11 4K PRO drone and the VisDrone dataset were used to capture the experimental data. The authors used the collected information to run the

YOLOv5 and the proposed YOLOv5 models and derive their respective performance metrics. The proposed YOLOv5 model improved upon the original YOLOv5 model in terms of precision, recall, F-1 score, and mAP (0.5), except for function loss. Also, the investigation was conducted to determine which object detection model performs best under different scenarios.

Zhan et al. [45] have proposed four strategies to enhance its small object detection accuracy based on YOLO v5. Moreover, the authors design these four methods with the impact on detection speed because it is essential for the model to be compact, quick, cheap, and simple to implement in real-world applications. The model incorporating all the enhanced techniques can increase detection precision and significantly lessens the hitch in detection time. Finally, the proposed model's 'mAP' is improved from 12.7% to 37.66% using VisDrone-2020, and the detection speed is up to 55 frames per second.

Liu & Luo [46] have proposed a YOLO v5-based algorithm for detecting multi-rotor drones. To begin, Efficientlite is used in place of Yolov v5's backbone to reduce the model's reliance on parameters. To compensate for the accuracy loss, adaptive spatial and spectral fusion is infused into the baseline model to enable the fusion of feature maps with varying spatial resolutions. Finally, an angle constraint is added to the initial regression loss function to speed up network convergence to prevent the prediction frame from deviating too much from the actual frame's orientation. The improved YOLOv5s show improved detection performance in experiments, providing a more effective method for detecting multi-rotor UAVs.

In recent years many other object detection approaches have been published. One recent idea was to transform an object into a set of points. Then, the detection was done by keypoint estimation. Law & Deng [47] proposed this approach in 2019. By detecting objects as a pair of critical points, the author suggested that there is no need to design a set of anchor boxes. Corner pooling was also introduced as a new pooling layer approach to localize corners. The authors were able to outclass all existing object detection models by evaluating the results on the COCO dataset. The average Precision was more than 42%. Another similar idea was used by Zhou et al. [48] in 2019. The authors detected the center point of a bounding box using a heat map. Regression was used to extract all other properties like size, 3D location, and

orientation of a bounding box. The model outperforms all existing object detection models with average precision as high as 52% on the COCO dataset.

Guvenc et al. [49] described some threats from unauthorized drones and prohibitions caused by amateur drones. Some advantages and disadvantages caused by different sensors were also discussed. It was observed that the optical sensors have the smallest cost of operation. The presence of cameras in different areas could help extract features at a meager cost. A few cons might be the impact on visibility, i.e., fog, dust, and occlusion, which require some maintenance.

Computer vision shares various methodologies for efficient object detection. Various factors play a critical role in detection accuracy; for example, the field of view (FOV) as a wider FOV is generally used for broader coverage, and a narrower one is used for better detection and increased accuracy, so the wise decision of one of the above is needed while deploying the sensor. Optical sensors also depend on many other factors, which should be considered before deploying them for operations.

A very comprehensive review of drone detection and classification using machine learning was presented by Taha & Shoufan [15]. This paper was published in 2019 and summarized 50 references in a structured way. One of the critical factors shared in this paper was the limitation of the available dataset to train models for detection. The author also shared that not many papers discussed the detection range. The performance of a detection model with increased detection range can be a very potent future research area.

Jahangir & Baker [50] show the potential and need for radars for drone detection. 3d-Holographic radar was used in the paper to detect drones at a more extended range of around 1 km. It was observed that radar was able to detect drones but was generating a large number of false negatives while detecting birds and other targets. The author improved the detections while invoking many other features like height, acceleration, radial velocity, etc. Table 2-2 gives a quick overview of classification accuracy collected from a few papers on Radars. The columns present a reference, the type of radar used, the type of class, and the final classification scores.

Table 2- 2. Quick overview of classification accuracy based on different radars collected from a few papers.

Work From	**Test Flight**	**Radar**	**Class**	**Classification Accuracy**
[51]	45 Sample Flights	S-band Pulse Radar	Three categorized classes on payload	90%
[50]	300s flight	Holographic Radar	Drone/No-Drone	88%
[52]	Simulated Data	Pulsed Radar	UAV/Birds	100%
[53]	720 Samples	CW Radar	3 Classes-Drones	94.7%
[54]	10000 images	FMCW Radar	2 Classes-Drones	94.7%
[55]	-	Pulsed Radar	2 Classes-Drones	70%-100%
[56]	8000 Long Trails	S-Band Radar	4 Classes 2 Drones-2 Birds	100%

A few of the observations were collected from the review. The results conducted above were mainly on minimal data and mostly simulated. Even though the work on countermeasures for UAVs has been a trendy and exciting research area, no reasonable efforts have been made to evaluate the results on real datasets.

Taha & Shoufan [15] extracted the conclusion based on all the reviewed papers for detecting drones via radar. The author's final comments were “though the solutions shared in different papers seem to be satisfactory. It is not evident to utilize all these solutions and make them generic with all kinds of drones. Not much data was shared, and the results did not consider all varieties of birds and drones.” The extracted features like altitude, height, and range were minimal with small dataset samples. The author also stated that the results provided were insufficient to evaluate a detection model's performance in real-time. Not many satisfactory results were provided with a wide variety of datasets.

Various challenges were faced while detecting objects in real-time. Although radar is very efficient in target identification, it requires trained personnel that can interpret visual outcomes or people with good decision-making. Also, radar might be limited to a few scenarios, which some other sensors should complement. Many researchers made good progress in localizing drones using computer vision and implemented a methodology to increase the accuracy of the overall system. Rozantsev et al. [57] proposed several approaches to detect drones based on the implementation of visual detection with learned features. The authors implemented a CNN detection model, and the drones' average precision was around 0.850. Table 2-3 summarizes the literature on object detection algorithms using deep learning.

Table 2- 3. Overviews of the methodology used for object detection using deep learning.

RESEARCH	**CLASS**	**Methodology**	**Classification Accuracy**
E.UNLU [58]	Drone/Air-Craft	Generic Fourier Descriptor	85.5%
S. K. Boddhu [59]	UAV	Geographical Distributed Data Points	88%
D. Lee [60]	UAV	Haar & Learned Features	89.5%

J. Peng [61]	UAV	Faster R-CNN	60.5%
M. Saqib [62]	UAV	D-CNN	90.5%
C. Aker [63]	UAV	YOLOv2	90%
R. Yoshihashi [64]	UAV	Recurrent Correlation Network (RCN)	86%

It was observed that most of the detections were made on the simulated dataset. Also, it was observed that authors were not confident about sharing the results based on different features like range, pixels, and resolution. That provides an excellent launching point for investigation, as researchers can try out a wide range of features in controlled lab conditions to see which ones fail to produce the desired outcomes.

It has been observed that many papers have been published where the authors implemented drone detection utilizing the deep neural network methodology, i.e., convolutional neural network (CNN). One such research in 2017 [65] presented an overview of the detection models based on accuracy and speed. It was also concluded that YOLOv2 outperformed all deep neural networks, i.e., CNN, based on accuracy and speed. The paper also shared the performance metrics based on the training time and speed, i.e., the frame per second (FPS).

Many researchers [66]– [68] shared similar results and concluded that Yolov2 detected objects with a reasonable tradeoff between accuracy and speed. YOLOv3, in recent years, is considered way faster than Yolov2 and is considered a good detection model when implementing a real-time system. Unlu et al. [69] implemented the results based on YOLOv3. Based on these facts, the proposed methodology included YOLOv3 in this thesis work. It should also be noted that this decision was made after a comprehensive literature review,

considering the results based on each optical model used in other research conducted for object detection.

When implementing the real-time system, one primary concern is detecting the object without any delays due to the processing time. Also, since we are fusing radar and optical signals, it is essential not to have any computational delays that can lead to late detections or missing frames. Since radar provides us with a region of interest, optical with YOLOv3 gives the final confidence utilizing sensor fusion. It is important to note that even a few seconds of processing and computational delay on radar delay result in incorrect positioning of our object, and then the optical detection model will fail in the last layer of detection. More detail about our real-time system will be provided in a separate Chapter.

None of the Yolo-based research so far provided any results based on the sensor to target distance to the best of our knowledge. None so far provided any relative information on how distance might be a crucial factor while detecting drones. This thesis will present a few results based on these features, which can represent an essential addition to the object detection field. The two sensors that will be incorporated in our thesis work are radar and optical.

Overview of Sensor Fusion

Radars, lidars, lasers, IR, and cameras have complementary characteristics that naturally lead to the desire for sensor fusion. Object detection using cameras and radars standalone can have inherent limitations that can lead to misleading results and should be addressed by each of these sensors. The benefits of integrations come from the fact that features/data from these sensors excel on different tasks that can be matched, verified, and fused to achieve significantly improved accuracy and reliability overall.

From Tables 2-4 and 2-5, the comparative study shows a few of the limitations addressed by each sensor.

Table 2- 4. Comparative study of the properties of radar and camera

Sensor	**Radar**	**Camera**
Weather	Adverse weather, such as rain, snow, and fog, does not affect them.	Susceptible to illumination variations and adverse weather.
Data density	Sparse with point-wise reflection.	Dense with appearance information.
Object boundary	No	Yes
Object speed	Yes	No
Multiclass classification	Radar features such as range and velocity are used to discriminate between classes.	Discriminatory against a more significant number of classes.

Table 2- 5. Advantages of fusion in different tasks

Perception	**Camera**	**Radar**	**Fusion**
Distance	Fair	Excellent	Excellent
Angle	Excellent	Fair	Excellent
Velocity(Radial)	Limited	Excellent	Excellent
Velocity(Lateral)	Fair	Limited	Excellent
Boundary	Excellent	Limited	Excellent
Obstacle	Fair	Excellent	Excellent
Classification	Excellent	Fair	Excellent
Weather/Lightening/Dirt	Limited	Excellent	Excellent

Table 2-5 shows the performance of radars and cameras under various conditions, i.e., weather conditions, classification, obstacles/occlusion, and many more. From Table 2-5, it is depicted that fusion has the potential to improve the performance of both sensors by taking the best of both sensors. The table is based on the works of [49], [70]– [78]

2.3.7 Detection Level Fusion

The thesis work presented in [75] presented a comparative study and a few limitations of each sensor when fusing the results for decision-level detection. Fredrik Svanström utilizes the class output and confidence scores of the included sensors by giving each sensor a particular weight scale based on the limitations. It was evident that training the network on the feature level requires extensive data. The work done by Svanström [75] on sensor fusion was preliminary and just introduced the basics of sensor fusion but can be used to address the limitations of each of the sensors.

Dempster [79] uses the evidence theory to clarify confusing data from disparate sensors is possible using a common framework of interpretation.

By Bar-Shalom and Tse (1975) and Baig (2012) in [70] present a Bayesian approach in laser and radar and for a dynamic autonomous vehicle environment. His work included the Bayesian approach to get a combined position from laser and radar.

2.3.8 System Level Fusion

Baig et al. [71] proposed object detection on the system-level fusion between laser and stereo vision. The author employed the same approach as above to calculate the position of an object by fusing both sensors. The results obtained on data sets of INTERSAFE-2 showed that this fusion has improved data association and track management steps. A few other methods were highlighted in that paper, like low-level fusion. The fusion is done by constructing an occupancy grid for each sensor, and fusion is performed to get a fused occupancy grid at the object detection level. The output of each sensor is processed to extract lists of objects (or obstacles) in the environment, and then information about corresponding objects in these lists is fused to get a fused list of objects.

Chadwick et al. [72] proposed an approach for autonomous vehicles that explains the fusion methodology between radar and vision to calculate the speeds between two vehicles. SSD object detection framework with ResNet blocks was constructed by projecting the radar scans on an image plane and by utilizing video recording for vehicles and pedestrians. Around 50,000 images were used for training the neural network model. This approach can also be applied in our project, but we lack continuous synchronized data from optical sensors. Getting this kind of data to train our model is beyond this project's scope.

Zhong et al. [73] used the same methodology as above by constructing the target ROIs (i.e., region of interest) with associated information such as radial distance, angle, and radial velocity from radars and then fused it with camera data by CRF (Camera Radar Fusion) blocks which performs fusion. To accommodate the difference in space between 2 spaces, the author of the paper suggested a fusion framework that tracks sensed objects on the 3D and 2D planes illustrated in Fig below. It is modeled in the 3D space (i.e., 3D positions and velocity) for each object and on the 2D image plane as 2D velocities and bounding boxes.

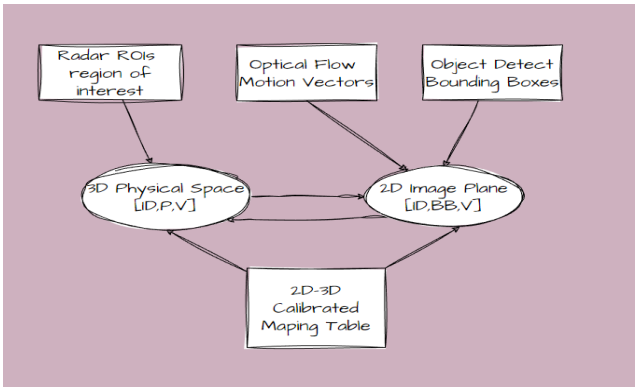


Fig 2- 7. Proposed object-level fusion framework [73]

As depicted in Fig 2-7, the fusion framework ensures the association between these models so that they correspond to the same object in real-time by coupled model updating. Once the association is done, we can train the detection model using YOLO or some other algorithm and then implement it in real-time.

Chapter 3. Methods & Material

3.1. Introduction

This Chapter describes the proposed methodology for object detection, i.e., the UAV-BIRD detection model. This section starts with an overview of the proposed modeling leading to the system architecture and, at the end, real-time software.

During the literature review, it became apparent that many researchers neglected to highlight critical features of drone detection in their work and that little effort was made to implement a real-time system. Furthermore, many researchers avoided reporting critical features such as the range and distance of the proposed methodology. This research will try to fill the identified research gaps.

This Chapter will also describe the hardware used in building the current project. The following topics play a crucial importance in the proposed methodology:

- Computer Vision
- Deep Neural Networks
- YOLOv3
- TensorFlow
- Machine Learning
- Faster RCNN Detection Models
- Image Processing

3.2. Proposed Methodology

An efficient drone detection model should be able to detect the required drone with incredible speed and with reasonable confidence. It is noted that newer versions of drones can fly very fast. Even a fewer-second delay will fail the required modeling, especially in a restricted area. Based on the observations, a model is required to address the above concerns. Also, one sensor might not be sufficient to detect those fast-moving objects, and various

sensors might be needed to complement each other, which can lead us to track those moving objects in the frame of the visual sensor. Various methodologies have already been built for object detection; however, choosing the right system for the application is critical if good results are needed.

Also, pre-processing and post-processing are very crucial in such models. The literature review done in Chapter 2 gives us an overview of all models, and we decided to choose some models based on our specific application. In this thesis, we are working with YOLOv3 mainly because of its fast detection reputation with reasonable confidence. Also, a comparison was made with Faster RCNN. The model is updated with a deep sort algorithm to track the given object in each frame to improve confidence.

To be able to put together such a firm system architect, another sensor, '*Radar,'* is introduced in the sensor fusion to improve the confidence and to complement each other in the case when the drone is not able to be detected in limited cases by one sensor. The thesis employed the supervised machine learning classification technique with the YOLOv3 architect, where a dataset based on the data augmentation technique is trained with an annotated ground truth. The model is trained with two classes with a sufficient annotated dataset on two ongoing classes to detect objects with high enough efficiency.

On the radar side, detection methodology is done with the help of an Interacting multi-model filter for target tracking, feature extraction, and machine learning classification technique. When tracking a moving target, it is impossible to use a static kinetic motion model that accounts for every possible scenario. With the Interacting Multiple Model filters, multiple models accurately depict the target's complex behavior. The IMM tracking is used to enhance the tracking performance and adopt them for the target classification at ranges beyond 500m.

The undergoing sections describe more details about the vision modeling, and detailed system architecture for the radars is beyond the scope of this thesis work.

3.3. System Architecture

The overall system architecture is divided into four major blocks: data acquisition, feature extraction (optical/radar), AI classifier and fusion and is presented in Fig 3-1**.**

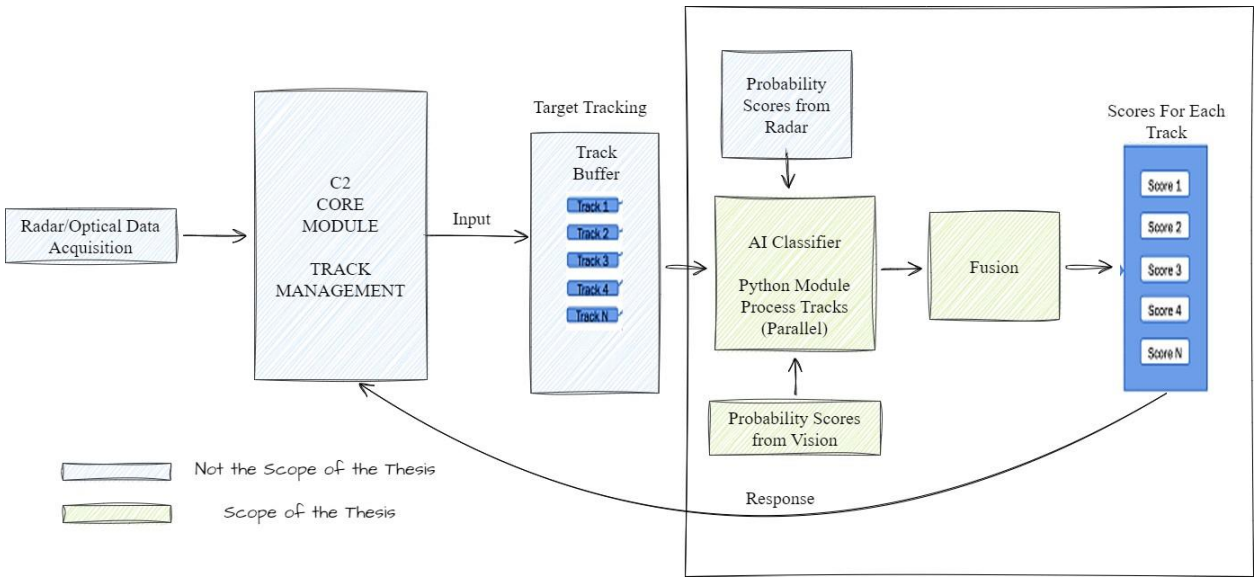


Fig 3- 1. System architecture

The system used a Bosch Camera as an optical sensor for detection and classification purposes. QinetiQ radar was used at the start of the experiments, but due to several issues with the data, the subsequent experiments were conducted using Echodyne radar and Bosch camera. This thesis will not focus on the sub-module, the C2 communication module that was developed by the industrial partner on the project. The C2 communication module will send a request to the AI classifier, which acknowledges it and then send the defined fused results. The TCP protocol establishes connections between the AI classification and C2 communication modules. Between iterations of real-time processing, the C2 module will keep tabs on the status of the active classifier module and maintain the tracks and connection. The proposed modules include the various models and algorithms, such as the python module, feature probability from vision, classifier, and score/probability of each track.

QinetiQ's obsidian system is at the heart of the system, and it features a single high-definition dual camera that combines color and a low-light camera sensor. The Palisade

system alerts this camera whenever it detects a drone, and humans can then review the footage. The Palisade is a toolset that provides advanced encryption and security capabilities to protect data and computations even in untrusted environments that are used under the C2 communication module. Additionally, the Palisade user interface allows manually cueing the camera onto any tracked contact by selecting it.

Bosch Camera is built on the most efficient and powerful H.264, and H.265/HEVC encoding platforms, and its rugged construction makes it ideal for use in various outdoor applications, including traffic monitoring. The dual-lens technology in the camera is intended to provide better image quality in different lightening conditions. The camera can stream high-definition video with minimal strain on the network.

3.3.1 Deep Neural Network

YOLOv3 [28] is the state-of-the-art algorithm used in the current methodology, which is further updated with deep sort to keep track of an object. YOLO is the fastest real-time object detection algorithm, which was revolutionary in the computer vision industry. YOLO was first introduced in 2015 by Joseph Redmon, which immediately brought the intention of many researchers in computer vision research.

YOLO

To understand YOLO, one should understand how a convolution neural network (CNN), called ConvNet, works. CNN takes an image as an input parameter, assigns several weighting parameters to different aspects of an image, like edges and outlines, and then provides an output label. Fig 3-2 shows the architecture of the CNN [3].

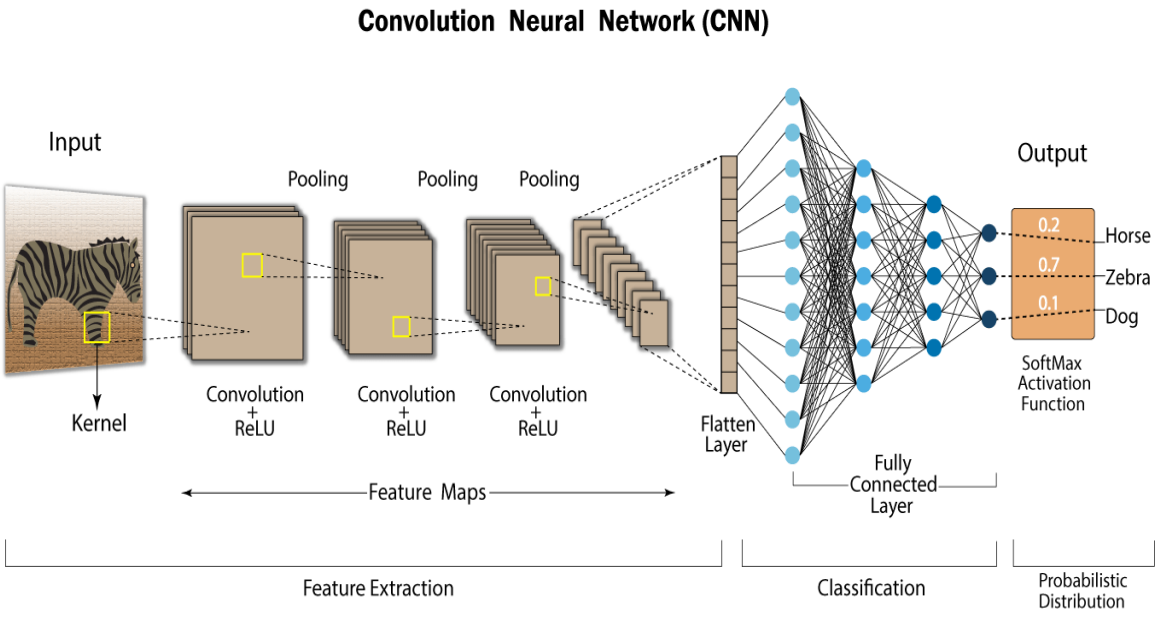


Fig 3- 2. Convolution Neural Networks

Convolution neural network contains several layers, also called convolutional layers, that can be optimized, i.e., pooling and many more, based on the particular application or the required dataset to make the required model more efficient. The input image is convolved using several filters or kernels. After processing the image, each filter in a stack forwards its outputs to the next filter in the stack. Each layer acquires the knowledge necessary to recognize a unique set of characteristics and the operations that must be performed and replicated across dozens, hundreds, or even thousands of layers. The conclusion is that CNN can recognize the complete object because it has processed all the image data and moved through each level.

Let us define an image with a small matrix 5x8, as shown in Table 3-1; this input will be passed to the convolutional layer, which has various filters per the depicted nodes. These filters will convolve across the image pixels of a 3x3 block of image pixels, slide over to each of the cells, and provide an output result that will translate the whole input image into another matrix based on the filter. The filter is given in Table 3-2. In other words, we can also refer to it as a dot product between the image pixels and the filter. CNN will continue the dot product process, known as convolutional operation, through all layers. Various operations like this can

be done on each convolutional layer to optimize the performance of a detection model. The required methodology will be able to extract the temporal and spatial features in an image.

Table 3- 1. Matrix of pixels for an image

1	2	3	5	6	7	8	9
1	0	2	58	9	6	48	8
22	55	66	141	55	665	4	55
66	33	21	23	332	11	23	5
55	55	88	44	01	2	35	55

Table 3- 2. 3x3 filter

1	2	5
8	7	5
8	7	8

Architecture of YOLO

Object detection is complex compared to image classification, as we have seen the methodology in a convolutional neural network to identify the patterns in each image; the complexity lies in detecting objects within the same image as the localized position is not as simple to depict.

YOLO divides the image into sections and then using image classification and probability calculations to find the target bounding boxes in each section. YOLO gained its reputation mainly because of its high accuracy and fast computing power than all the existing models in the object detection research and industry. The algorithm works fast compared to other

models, i.e., Fast R-CNN, Faster RCNN, and many more, because it requires just one forward propagation to make predictions. The non-max suppression layer is added to the model to avoid the same bounding boxes. So, the model is trained on a specific dataset that divides the testing image into different regions, and then CNN does the rest of the job.

YOLO efficiently extracts different features on an object and then learns the generalizable representation of objects on various classes to predict required boxes. YOLOv3 is the updated version of the original YOLO versions with 53 more convolutional layers, an ability to predict the score for each bounding box using logistics regression, and then the loss function, i.e., cross-entropy, to increase the model predictions.

YOLOv3 has 106 convolutional layers and is a variant of the darknet, as depicted in Fig 3-3 [82], which shows the overall system architecture for YOLOv3.

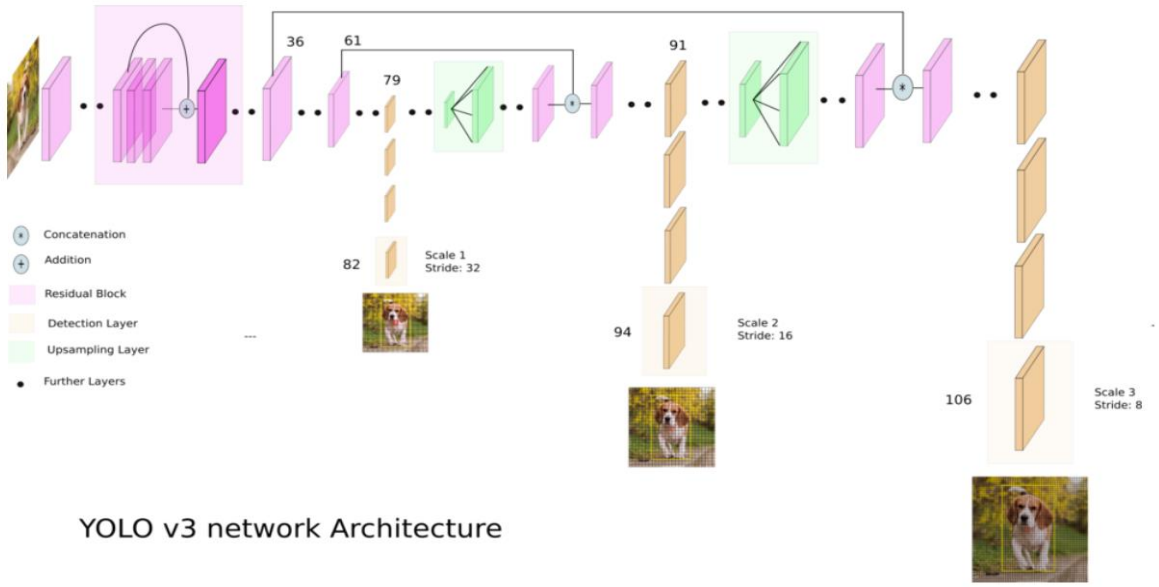


Fig 3- 3. YOLOv3 network architecture

YOLOv3 makes predictions at three different scales by applying a kernel of 1 x 1 at different network sections. The shape of our kernel is specific for each model and specific dataset, as in the above architecture [82]. There are two classes, so the defined kernel is 1 x 1 x 21. Each detection comes with box coordinates, the probability of the presence of an object in a defined image, and the class scores. YOLOv3 makes predictions at three different scales of a defined input image by downsampling the required input image. At each scale, each grid

of an image specifies three boxes using three anchors, so there are nine anchor boxes for three scales. The detection kernel is applied through the input image, and it transforms the required input in a feature map before sending it to another convolutional layer, followed by the fully connected layer in the end. Moreover, it predicts the probabilities of the required classes based on the logistics regression by specifying a threshold on each class score to detect multiple labels for an object. It is noted that SoftMax has been replaced with a multilabel classification approach in YOLOv3.

Different network parts can detect tiny objects like UAVs by down sampling the input image.

The loss function in the YOLOv2 is given by Equation 3-1 [11]. The updated YOLOv3 used the cross-entropy loss function and replaced the last three terms of the given loss function.

$$L=\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{i\,j}^{obj}[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2] \tag{3-1}$$

$$+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{i\,j}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right]$$

$$+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{i\,j}^{obj}\left(C_i-\hat{C}_i\right)^2$$

$$+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{i\,j}^{noobj}\left(C_i-\hat{C}_i\right)^2$$

$$+\sum_{i=0}^{S^2}\mathbb{1}_{i\,j}^{obj}\sum_{c\,\epsilon\,classes}(p_i(c)-\hat{p}_i(c))^2$$

Localization loss is given in Equation 3-2.

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{i\,j}^{obj}[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2] \tag{3-2}$$

This term measures how well the network is able to predict the coordinates of the bounding boxes. λ_{coord} constant that controls the weight of the localization loss. S is the size of the output grid, B is the number of bounding boxes predicted by each grid cell, $\mathbb{1}_{i\,j}^{obj}$ is an indicator

variable that is 1 if the jth bounding box in the i[th] cell is responsible for detecting an object, and x_i , y_i are the predicted coordinates of the bounding box center, while $\hat{x}_i$, $\hat{y}_i$are the ground truth coordinates.

Size loss is given in Equation 3-3.

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i\,j}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\widehat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \tag{3-3}$$

This term measures how well the network predicts the width and height of the bounding boxes λ_{coord} is a constant that controls the weight of the size loss. $\sqrt{w_i}$, $\sqrt{h_i}$are the predicted square root of the width and height of the bounding box, while $\sqrt{\widehat{w}_i}$, $\sqrt{\hat{h}_i}$are the ground truth values.

Objectness loss is given in Equation 3-4.

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i\,j}^{obj} \left(C_i - \hat{C}_i \right)^2 \tag{3-4}$$

This term measures how well the network can predict whether an object is present in each grid cell. C_i is the predicted objectness score for the ith grid cell, while $\hat{\mathrm{C}}_\mathrm{i}$ is the ground truth score.

Background loss is given in Equation 3-5

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i\,j}^{noobj} \left(C_i - \hat{C}_i \right)^2 \tag{3-5}$$

This term measures how well the network is able to predict that there is no object in each grid cell. λ_{noobj} is a constant that controls the weight of the background loss. $\mathbb{1}_{i\,j}^{noobj}$ is an indicator variable that is 1 if the jth bounding box in the ith cell is not responsible for detecting an object.

Classification loss is given in Equation 3-6.

$$\sum_{i=0}^{S^2} \mathbb{1}_{i\,j}^{obj} \sum_{c \,\epsilon\, classes} \left(p_i(c) - \hat{p}_i(c) \right)^2 \tag{3-6}$$

This term measures how well the network is able to predict the class probabilities for each object. $p_i(c)$ is the predicted probability that the object in the ith cell belongs to class c, while $\hat{p}_i(c)$ is the ground truth probability. The sum is taken over all classes.

Deep Sort Algorithm

YOLOv3 with object detection performs well in most scenarios but has some limitations and requires further processing to make it perform well in real-time scenarios; it requires tracking in each frame which can be helpful to improve the overall score and improve the model. The proposed method utilizes deep sort with Kalman filtering to help us overcome detection issues [18]. Furthermore, the information is stated for each track, and the parameters to track and delete are based on the most recent successful tracks, as they would have gone out of frame. Also, to avoid any duplicate tracks, a threshold to prioritize a minimum number of detections is used.

While the Deep Sort algorithm has provided a new bounding box, we still need a matrix to link the updated predictions to the old ones (that is, to link track 'i' with a detected event 'k'). In this case, we use the covariance and a distance matrix to quantify the relationship, which is crucial when working with real-world situations. The Mahalanobis distance is given in Equation 3-7.

$$D^2 = (x - \bar{x})^T S^{-1} (x - \bar{x}) \quad (3\text{-}7)$$

Where T is the transpose of the matrix and S^{-1} is the inverse of the covariance matrix. x is a vector of feature values for a specific data point.

The Hungarian algorithm is used to minimize the cost. Since the problem is associating track "i" with incoming detection "k" with minimum distance, we need some algorithm to associate this problem with minimum cost (i.e., the minimum cost associated with distance matrices and appearance matrix).

When dealing with real-time world scenarios, it might not be straightforward to deal with occlusion and blurriness with just some distance matrix since it does not include the information for the object's appearance. So, there is a need to introduce one more factor to combine motion and appearance information. Thus, another matrix is presented based on the appearance, and the classifier is built over the dataset, which is trained, and then stripped of the last classification layer. After this process, the appearance factor will be a dense layer that

generates a single feature vector. The term "single feature vector" refers to a data point with only one feature vector or set of characteristics or attributes for which the category is unknown. Data points are typically classified in machine learning and pattern recognition by assigning them to one of several classes based on the values of their attributes. A class prediction is obtained by applying the model to the feature vector in classification. To what extent the model is stable, and the characteristics used to represent the data are exhaustive determines how trustworthy the category will be.

The Mahalanobis distance with critical value D_k is given in Equation 3-8.

$$D = Lambda * D_k + (1 - Lambda) * D_a \quad (3\text{-}8)$$

Where D_a is the cosine distance between the appearance feature vectors, and $Lambda$ is the weighting factor, $D_k = sqrt\ (chi^{2\_k}, \alpha)$ and denotes the critical value of the chi-squared test with k degrees of freedom. However, chi^{2_k} is the chi-squared statistic with k degrees of freedom at the significance level α.

Fig 3-4 shows the variables and methods of the deep sort algorithm.

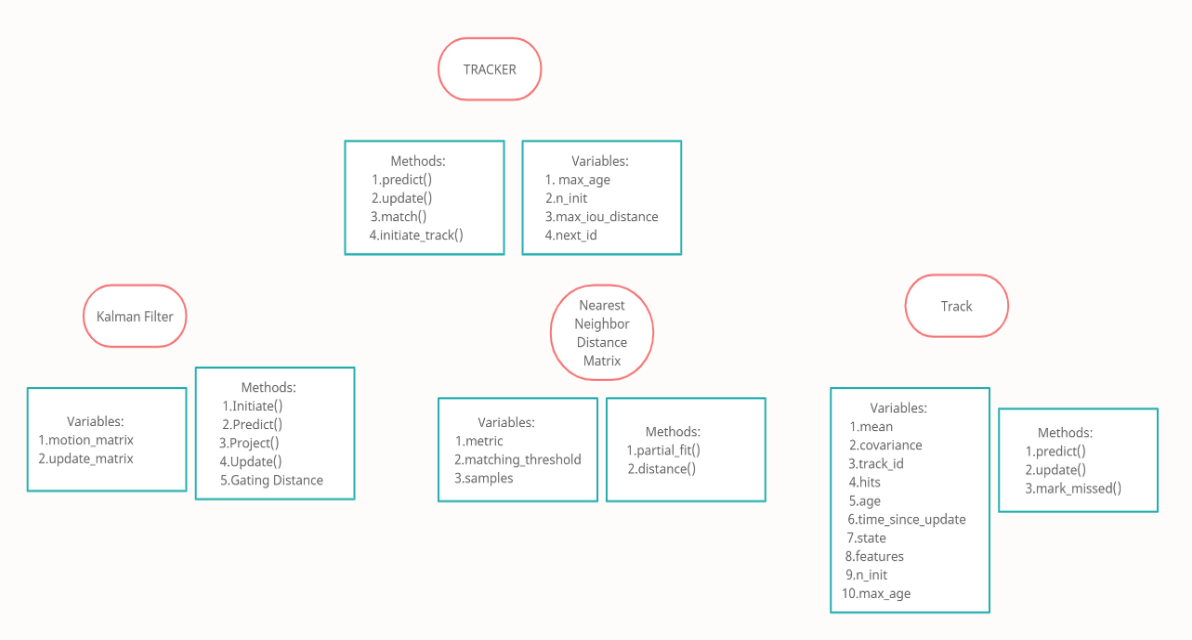


Fig 3- 4. Deep Sort with methods and variables

3.4 Sensor Fusion

3.4.1 Sensor Fusion

Data from two sensors working independently to perform object detection can be fused to improve accuracy. Also, from the literature review, it can be suggested that the model performance can be complemented in various scenarios where one model cannot detect the defined object because of its limitations.

Radars and cameras have complementary characteristics that naturally lead to the desire for sensor fusion. If we look at cameras & radars only, because of the project requirements and limitations, we can have inherent limitations that are being addressed by each of these sensors if they are used alone. The benefits of integrations come from the fact that features/data from these sensors excel on different tasks that can be matched, verified, and fused to achieve significantly improved accuracy and reliability.

The comparative study by John & Mita [74] shows a few limitations addressed by each of the sensors depicted in Table 3-3.

Table 3- 3. Limitations of the sensors

Sensor	**Millimeter wave Radar**	**Monocular Camera**
Weather	Not affected by adverse weather such as rain, snow and fog.	Susceptible to illumination variation and adverse weather.
Data Density	Sparse with point-wise reflection.	Dense with appearance information.
Object Boundary	No	Yes
Object Speed	Yes	No
Multi-Class Classification	Discriminative for limited classes using radar features such as depth and velocity.	Discriminative for the higher number of classes.

Real-Time Software

Fig 3-7 illustrates the interface connection between AI Classifier, i.e., (Radar and an optical sensor to perform the proposed detection methodology) and sub-modules, i.e., C2 Communication. The C2 communication module was built by the industrial partner on the project and will not be the scope of this thesis work.

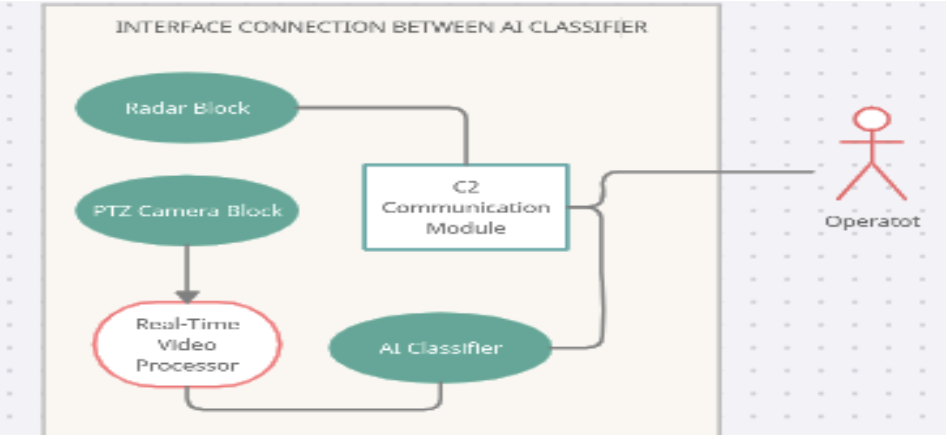


Fig 3- 5. Interface connections

The AI classification module relates to the C2 communication module using the TCP protocol. The C2 communication module will monitor the current classifier module's health and keep tracking the tracks and connection alive between real-time processing.

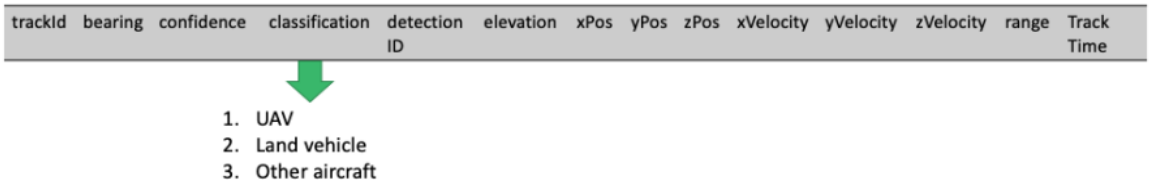
3.4.2 Graphical User & System Interface

The radar module provides track data that is further decoded to make it available for the C2 module, which then communicates with the AI Classifier module to process each track data and then make the predictions. The obsidian radar collects two samples per second.

The Track ID is a unique identifier for the track to which the track point belongs. The Track ID starts at 1 when the system database is empty and does not reset when the software is restarted.

The Detection ID is the unique identifier for the radar detection from which the track point was created. The Detection ID starts at 1 when the system database is empty and does not reset when the software is restarted. If the track point is predicted, then this field will be -1.

The X, Y, and Z measurements estimate the position of a track point in UTM coordinates, assuming a local UTM zone. Cross-zone working is not supported. The Z component is the altitude relative to the radar (or radar number 1 in the case of multiple radars). v_x, v_y, and v_z measurements estimate the velocity of target detection in meters per second within the X, Y, and Z coordinate spaces. The classification column classifies the target track into the class. Confidence is the probability of the classification.



The AI classification module processes the required tracks and then sends back the classification results in a packet containing the following:

- Radar Track ID
- Classification
- Confidence
- Classification Source
- Time Stamp

The classification source is either 0 or 1, defining radar classification results or fused results. The C2 communication module defines this parameter, and whenever a data packet is sent towards AI classification, this parameter is enclosed to have the defined results. Once the radar module detects the target, the camera will automatically slew towards the defined region of interest and classify based on the sensor fusion logic.

The C2 communication module will send a request to the AI classifier, which acknowledges it and then send the defined fused results. Classification results are expected to follow Table 3-4.

Table 3- 4. Defined class labels

UNKNOWN	The detected Object is *'UNKNOWN.'*
UAV	The detected Object is *'UAV.'*
BIRD	The detected Object is *'BIRD.'*
OTHER	The detected Object is other than a drone or bird.

The following algorithms illustrate the real-time software methodology.

Sensor Fusion logic has many check marks on each sensor. These check marks were made based on the extended analysis of the system's accuracy. The flow chart of the fusion logic is given in Fig 3-8.

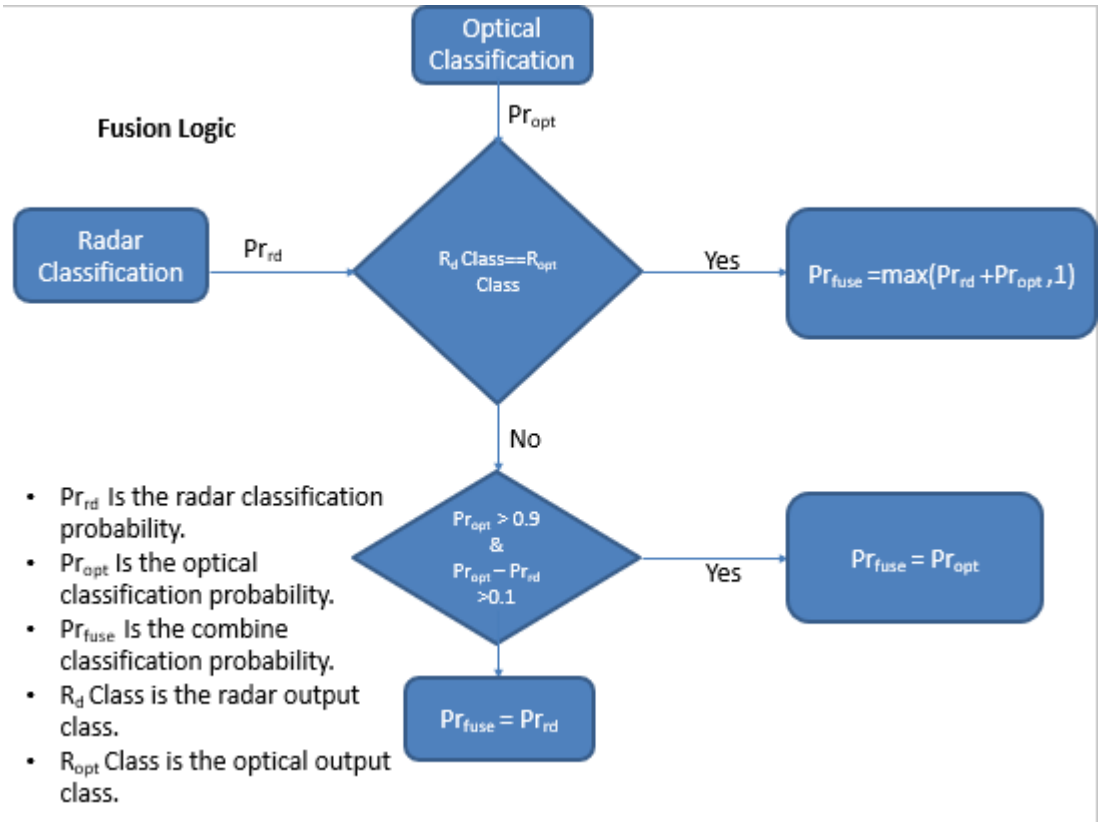


Fig 3- 6. Fusion logic

The purpose of these check marks was to complement the system's accuracy based on the limitations of each sensor in a specific limited environment.

Algorithm 1: Sensor Fusion Logic

```
while Video Is Enabled For That Particular TrackId do
    Sensor Fusion;
    if Both Radar And Optical Predict The Same Object;
     then
        Combine Scores Based On Yager's Rule of Combination
    if Optical Is More Accurate than 10 Percent;
     then
        Take The Optical Score
        if Define Region Of Interest Has Resolution Less Than Threshold then
            Do The Required Processing ;
        else
            Optical Score;
    if Optical Is Less Accurate than 10 Percent;
     then
        Take The Radar Score
        if Radar Packet Has Turn Rate Below The Threshold then
            Reduce The Radar Score Based On The Define Analysis;
        else
            Radar Score;
Radar Classification;
```

Fig 3- 7. Fusion logic-algorithm

As depicted from the fusion logic algorithm in Fig 3-9, it compares the radar and optical classes. In Appendix A, we have derived the equations for Dampster's and Yager's rules of combination for detection level fusion of both camera and radar sensors. In the case of ground targets and other aircraft, radar-based classifier outputs class 3. In this scenario, the Optical classifier is trained with two classes (UAV and Bird). It gives either UAV, Bird, or Unknown, which will bypass the optical classifier.

One more condition is introduced to check for class 3 fusion to tackle the current situation. If the radar detects the target as class 3, an additional check of the optical classification is done if it is a bird, drone, or unknown. The final class will be optical classification.

The following changes were made to the real-time system.

- Take the radar height from the ground (AGL) to mask the ground objects.
- Take the average target height from every track window instead of the initial window.

- When the detected target is tiny, do not tag it as another class. Just provide the UAV or Bird score of YOLO.

The flow chart for fusion logic with class 3 updates is given in Fig 3-10.

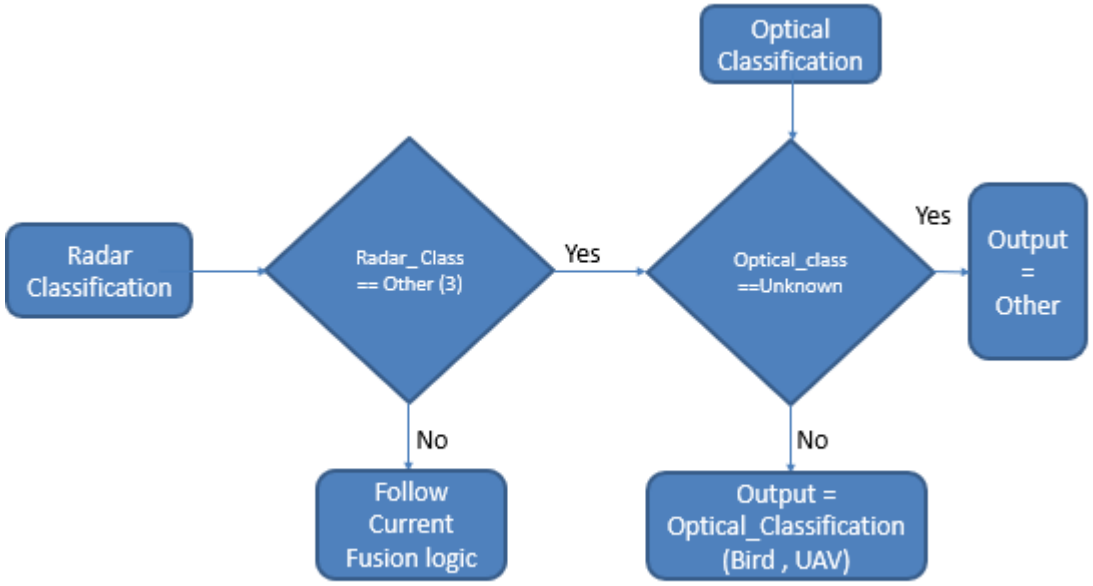


Fig 3- 8. Fusion logic with class 3 updates

Chapter 4. Data Collection & Data Augmentation Techniques

4.1. Data Collection

Machine learning (ML) algorithms require training data to generalize the features they learn from the data. Nevertheless, several other factors also contribute significantly to ML progress. Training data is the backbone of our object detection algorithm. The data quality and organized format are crucial in data collection as well. Typically, data gathered from various sources is made available in a disorganized format, which cannot be used for training. However, this data becomes useful for ML training after proper tagging and annotation.

Object detection models require an organized real-time dataset to train on different augmented situations so the model can ingest all features during real-time predictions. Collecting data is crucial for ML development, but it is not very economical based on various applications. For the research purpose, we have collected the data under *NRC (National Research Council)* collaboration with *CS Canada.*

There were about 6-7 experiments to collect data over different weather seasons in one year. Each experiment had around 15-20 flights with an average duration of 10-15 min per experiment. The initial experiments were conducted with QinetiQ radar, but due to several issues with the data, the subsequent experiments were conducted using Echodyne radar and Bosch camera. During the early data collection stage, it was observed that data collected through the PTZ camera was unsatisfactory. Objects were out of zoom, and each frame was blurry and useless for the training pipeline. To rectify the situation, Bosch Camera was introduced to enhance the quality of the data collection.

QinetiQ's OBSIDIAN high-definition dual camera system was used in one of the experiments conducted on Nov 28*th,* 2019; this camera has both color and low-light sensors. The Palisade system, which is a toolset that provides advanced encryption and security capabilities, signals this camera to look for drones; any identifications made by the camera are

then passed on to a human for final approval. Additionally, the Palisade user interface allows manually queuing the camera onto any tracked contact by selecting it.

The different categories of drones have been used in the dataset, including a small sized drone, high-performance drones like DJI Phantom 2, DJI Phantom 3, DJI Phantom 4, DJI Inspire, DJI Phantom 4 Pro, DJI Mavic, and medium-sized drones such as DJI Flame wheel. Several types of drones are shown in Fig 4-1.

The land location is an unoccupied piece of land within the Port of Montreal Contrecoeur Terminal, as depicted in Fig 4-2. The points on the map are J0L1C0 Lat/long: 45.8293, -73.2807. The location was in Contrecoeur, on the south shore of the St-Laurent River, east of Montreal, as shown on the map.

a) DJI Inspire b) DJI Phantom 4

Fig 4- 1. Categories of drones

DJI Phantom is a professional set of drones larger than any other traditional drones, with around 0.3m motor-to-motor side length.

During the flight, the following few points were taken into consideration:

- The drone was kept under the visual sight of the pilot.
- The drone was kept under the specified operational parameters, including range, elevation and velocity.

- The drone was kept at a safe elevation while considering the safety parameters of people and animals on the ground.
- The drone was following all the safety parameters. Also, the airport was nearby, so the pilot needed to follow all the regulations associated with drone flight.

Fig 4- 2. Location of the experiment

All data were collected in daylight to keep the drone in the visual sight of the pilot. An effort was made to augment data under various lighting conditions, which will be helpful for better predictions under low-light conditions. Also, weather conditions differed from a cold, cloudy day to sunny clear visual sight. Many other strategies were imposed that will be discussed later. Fig 4-3 depicts the weather variations while acquiring the data.

Fig 4- 3. Weather variations

The bird databank that was open-sourced, i.e., Google images etc., also has a considerable variation, and the following birds were included in the database.

- Black-bellied whistling-duck
- Canada goose
- Ruffed grouse
- Laughing gull
- Northern fulmar
- Double-crested cormorant, and many more.

The databank has an enormous variety of features; images/videos were captured using Bosch Camera ranging from 0–1000-meter distance in different test cases as depicted in Fig 4-3 with two classes as will be shared in the following sections. The two classes of the dataset are given in Table 4-1.

Table 4- 1. Class defined for an Optical model.

(0-1000 METERS)	CLASS	
DATA-SET	UAV	BIRD

4.2. Statistics on the collected/augmented data

The total collected dataset contains around 10k images of birds and drones. More than 50% of the data collected count was improved with data augmentation. A tool, PhotoScape, was used for pre-processing, i.e., data augmentation and many more. 'LabelImg' [80], as shown in Fig 4-4, was used to manually label/annotate all images. It supports the YOLO format and is written in Python.

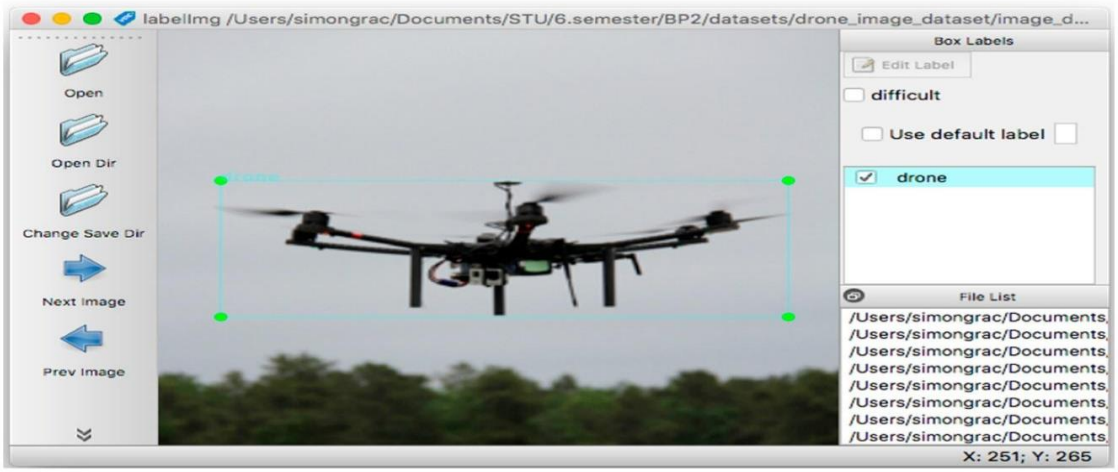


Fig 4- 4. LabelIMG (Annotation Tool)

Figures 4-5 and 4-6 show a few of the images collected through real-time experiments.

Fig 4- 5. Real-Time Data Collected for Drones

Images were taken in different weather conditions and orientations to train the model better with vast features. A few of the images collected through open source are as follows.

Fig 4- 6. Open-source data collected for drones & birds

Fig 4-7 below shares the statistics of the dataset collected over width and height. Most of the training samples collected are categorized in smaller bins (orange in color), i.e., width and height lower than 20% of the total number of pixels in an image. The area of the circular region defines how many samples of that training data are in a particular width or height. After looking at the statistics shared, we can observe that most samples have a width of 128 and a height of around 148 pixels.

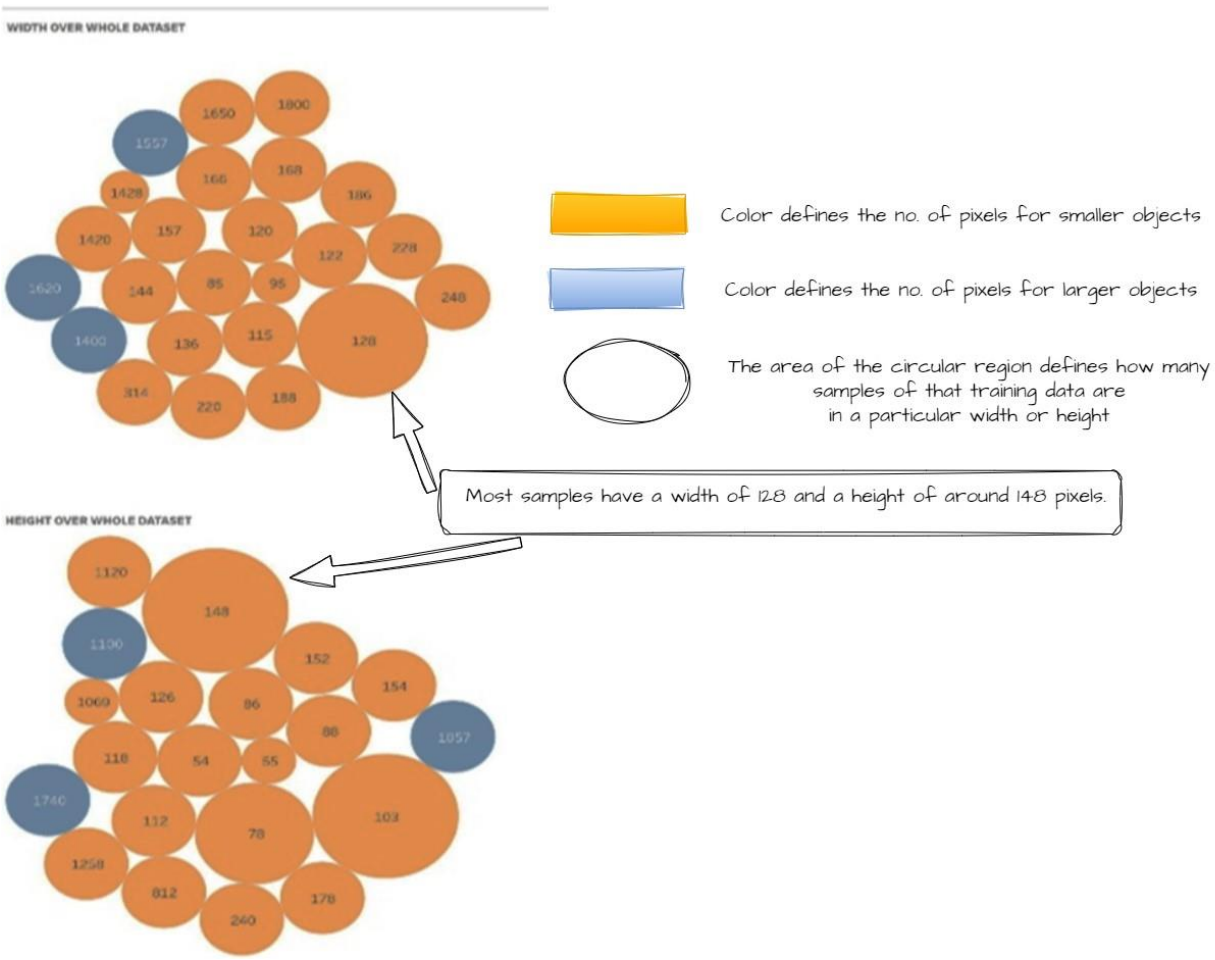


Fig 4- 7. Statistics on dataset over height/width

Dpi (Dots per inch) is another criterion defining an image's quality. The better the quality of an image, the better the extracted features will be. That eventually will result in a better model to train these extracted features. Fig 4-8 depicts that the training samples are mainly in a range of dpi 300.

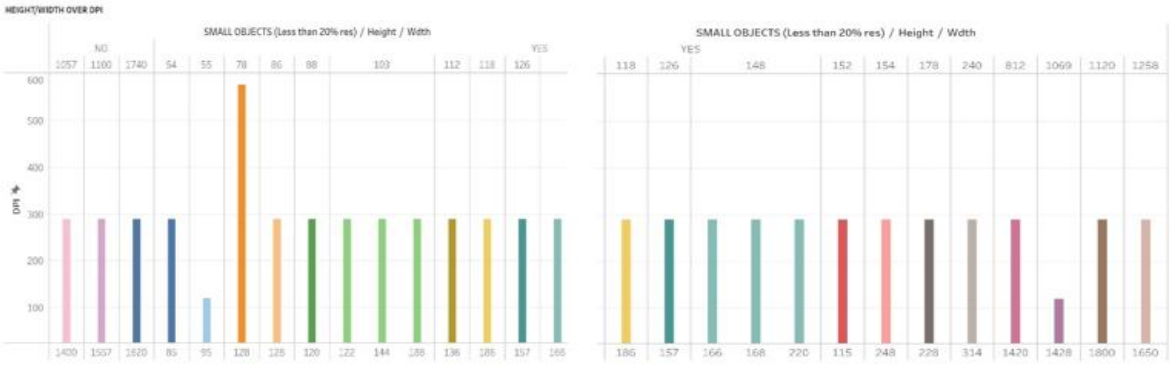


Fig 4- 8. Statistics on dataset over height/width vs. DPI

Fig 4-9 gives more insight into height/width over collected training data. 5k images have a width of around 128 pixels and 86 pixels in height.

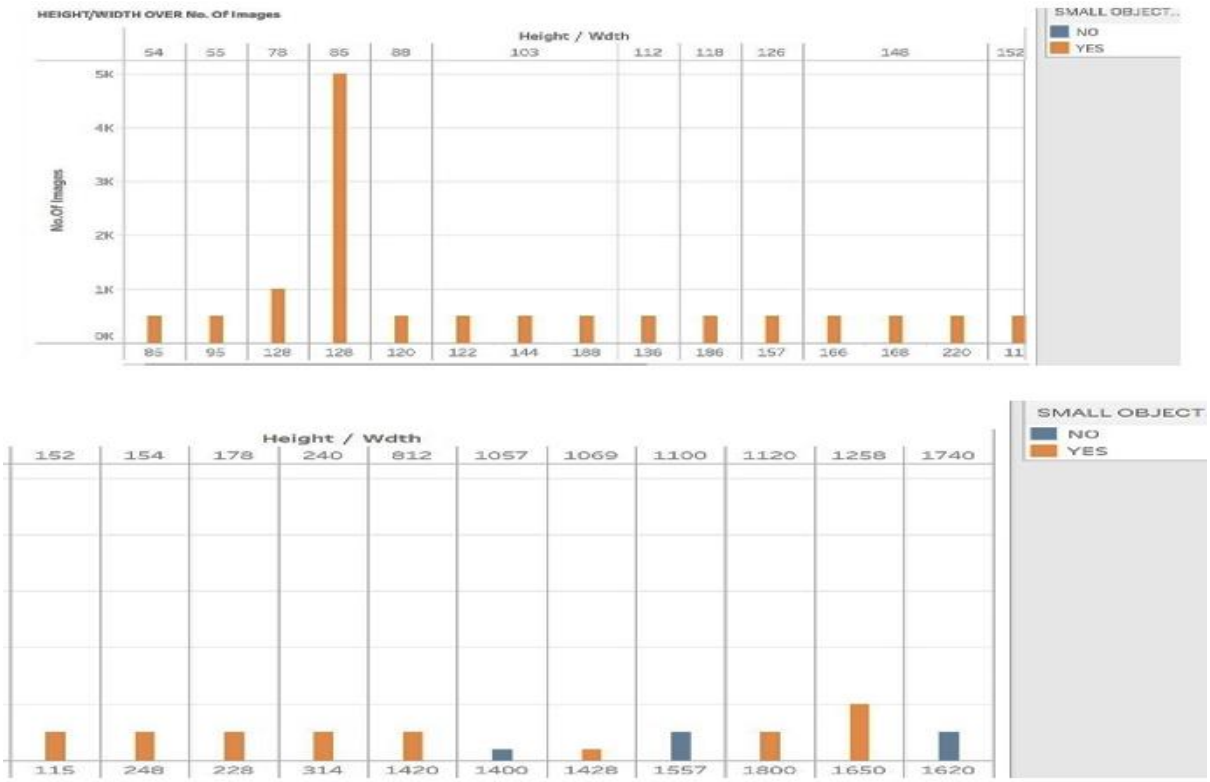


Fig 4- 9. Statistics on dataset over height/width

Fig 4-10 below shares the statistics over different features. The total number of images in our training samples is 17,400. Statistics share the number of images that were processed on different features on augmentation.

Following are the augmented features and the defined columns:

- Blur
- Geometrical transformation (0-90-180-270 degrees)
- Morphological transformations (Dilation, Erosion, and many more.)
- Multiple Objects (By mirroring the original image)
- Noise
- Occluded images (Not an augmented feature but the characteristics of an original image)
- Open-Source (Collected images from the open-source)
- Real-time collected samples (From NRC test experiments)
- Simulated samples.
- Small-Object (object is 20% and smaller than the total number of pixels in an image)
- Translation pixels
- Zoom-in
- Zoom-out.

Fig 4- 10. Statistics on dataset over different features

Fig 4-11 shares more insight into the collected features with both classes and the total count of images.

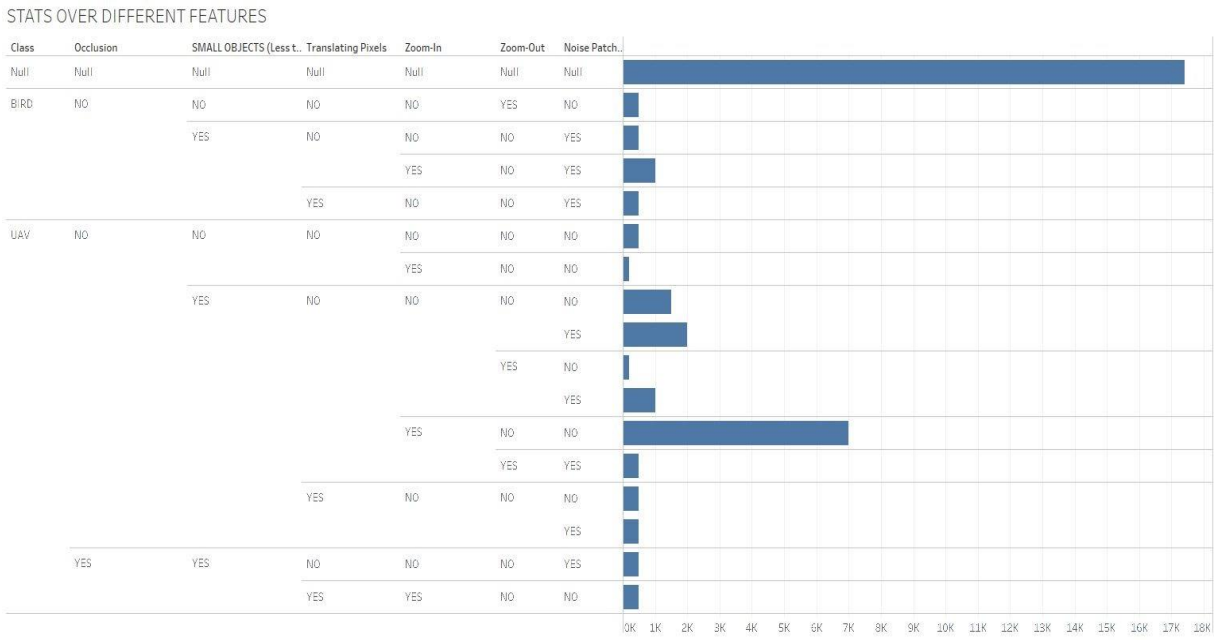


Fig 4- 11. Statistics on dataset over classes (Bird/UAV)

4.3. Test cases

Following are a few test cases on the real-time experiments:

- Test case-1 Maintaining a constant altitude while flying more than 500 meters directly towards and away from the sensor and radar.
- Test case-2 Straight flight with increasing altitude away from the radar and decreasing altitude towards the sensor at a distance of more than 500m.
- Test case-3 Flying towards the sensor for a few seconds, hovering around for some time, and taking a return flight.
- Test case-4 Vertical loops.
- Test case- 5 Horizontal loops.
- Test case -6 Smooth gliding up and down.
- Test Case -7 Horizontal + Vertical loops with varying altitude
- Test case -8 Elevation flight.
- Test case -9 Flying as a bird

- Test case -10 Moving around the target.

4.4. Data Augmentation

Data augmentation can artificially generate new information from the current samples by using a variety of feature enhancements. It is accomplished by applying techniques specific to the training samples' domain. It is helpful for both expanding the available data and producing new variants of the existing training samples. There are different policies in this research area:

- Data augmentation operations for image classification.
- Specialized data augmentation policies for detection models.

Both these policies have cons and pros, with some computational expenses that demotivate many researchers to utilize them in the detection models.

The need for data augmentation is the limited dataset that will result in learning. Deep neural networks need big data to overcome underfitting and train a model with good confidence scores.

Data augmentation strategies for image classification are particular to each dataset; for example, the Modified National Institute of Standards and Technology (MNIST) dataset requires distortion, translational and rotational data augmentation strategies. Also, image mirroring is even used in some research. Image cropping and mirroring of an image are commonly employed strategies of data augmentation in the image domain. Also, rotational and geometrical transformations play crucial importance in data augmentation strategies. Also, adding noise, grey tones, or contrast features can add to data augmentation strategies.

Given recent advances in computer vision, many researchers are attempting to automate data augmentation techniques to avoid the data set's specific nature. A few examples will be to automate the generation of this dataset by merging images based on the same class. Also, few researchers are looking to reduce the tedious task of manual annotation based on the presence of these objects in the exact location if translational features have not changed in the operational images.

In the image domain, a few standard augmentation techniques are rotating the image, translating the image with a few pixels, changing levels of contrast, exposure, gamma-bright, hue, saturation, brightness, darkness, & deepen, and applying grey tone. It was also observed that the model could detect with increased confidence after applying all these policies with a new set of datasets.

A few data augmentation strategies on the image are presented next. Fig 4-12 shows the original image.

Fig 4- 12. Original input image

As shown in Fig 4-13, the backlight illuminates the subjects from the back. In PhotoScape, we can illuminate the object at several levels to train the model at different levels of illumination. It will also help in coping with under-fitting.

Using PhotoScape, we can also increase the contrast and blur level. Fig 4-14 shares the computer graphic effect at a high dynamic range.

Fig 4- 13. Original Image illuminated with additional back light.

..

Fig 4- 14. The original image at bloom mode level 92, blur level 2.6 & contrast level 75

Figures 4-15 and 4-16 depict a series of images that have been transformed from the original image using different techniques. The first image in each series has been brightened, the second image has been given higher contrast, the third image has been darkened, and the fourth image has been created using deeper color ratios. Exposure has also been added at different levels. These different operations will increase the lightening and will help in extracting the required target object.

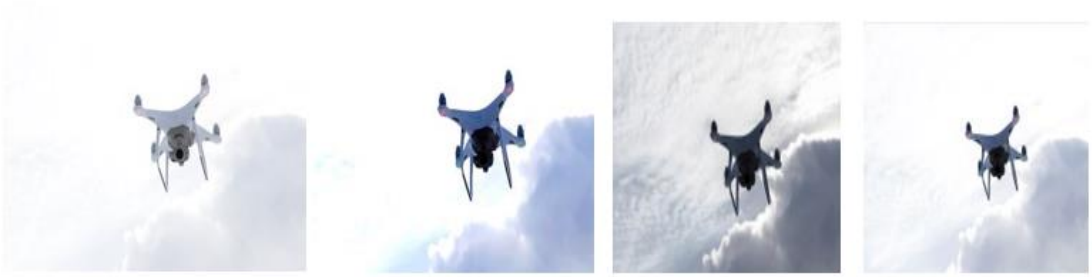


Fig 4- 15. The original image was subjected to different transformations, with a brightening level of 100, a contrast level of 56, a darkening level of 199, and a deepening level of 100.

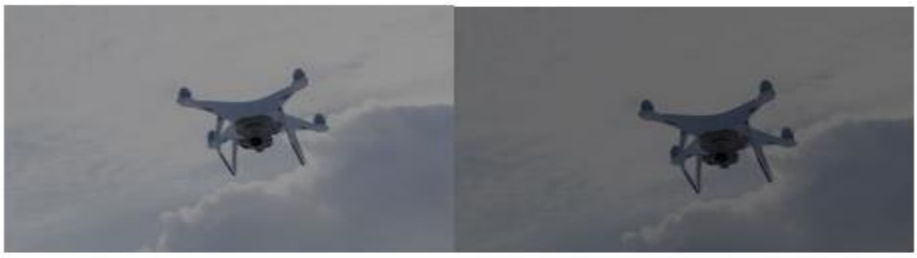


Fig 4- 16. The original image transformed at exposure level 0.86 & exposure level 1.40

The film effect shown in Fig. 4-17 aims to change the texture of a background image so that the object detection model can be trained at different conditions to predict in real-time scenarios.

Fig 4- 17. Film effect applied to the original image.

Figures 4-18, 4-19, 4-20, and 4-21 apply different lightening effects and morphological transformations to increase/decrease the white noise and reduce/enhance black thickness at various levels.

The texture will enhance the background and help extract more information while training the model considering different noise levels. Also, we made some geometrical transformations at different rotations, i.e., 0-90-180-270, as shown in Fig 4-20.

Fig 4- 18. Image was transformed with gamma correction, hue adjustment, and longer length.

Fig 4- 19. Geometrical orientation transformations at 90°, 180°, and 270°.

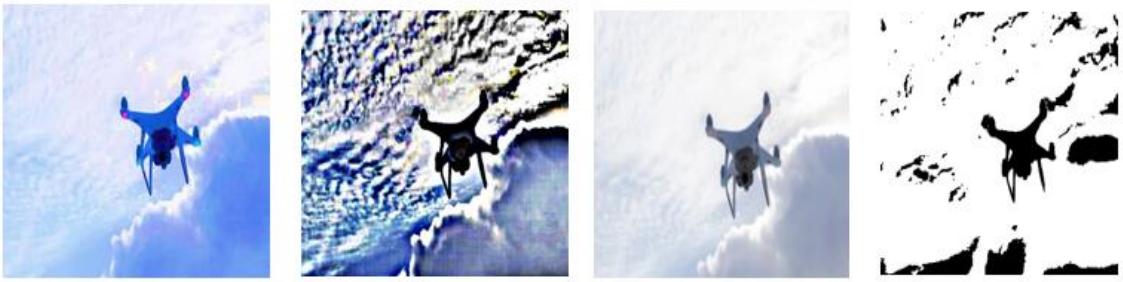


Fig 4- 20. Image transformed by adjusting its saturation (level 206), sharpness (radius 7.5), pixel translation (W=500 H= 359) & threshold masking (level 118)

Some computer vision transformations like erosion and dilation were also introduced using the same tool to introduce more variations.

Fig 4- 21. Image transformed by applying dilation & erosion

Noise patches depicted in Fig 4-22 were introduced in the original image on different levels. In neural networks, these noises play a crucial role in misleading the original model. Adversarial networks will be a good example where just a few layers of noise patches can mislead the network. The newsprint effect is a technique that simulates the halftone printing process used in newspapers and other printed media. The resulting image has a distinctive vintage look that is often associated with old newspapers and magazines. Although these noise patches do not make a difference in visual human eye inspection, if introduced to the detection models can result in different results. In many cases, these models cannot identify the given object class.

Fig 4- 22. Image transformed at different noise levels (172 & 230) with newsprint effect

In a few images, the reflection of the original image is introduced, as shown in Fig 4- 23. The motivation comes from extracting an image's intrinsic components to steer the learning process, which will help reduce the over-learning of a model.

These image-filtering techniques and layer separation tasks enhance the dataset variations, eventually making the model more robust while training. Also, it helps to identify objects under various lighting conditions.

Fig 4- 23. Image is transformed by adding a reflection effect

Also, using the filter feature in PhotoScape, we can introduce the illusion as shown in Fig 4-24 and introduce multiple objects. It will help train the model to identify multi-objects in real-time scenarios.

Fig 4- 24. Image transformed with an optical illusion effect

In real-time experiments, the camera was out of zoom in various test cases. Also, the weather conditions are not favorable enough, and sometimes we do not have a clear view of a drone. Given this, a defined model should be trained with different blur levels.

At earlier stages of detection model development, it was observed that blurriness was crucial in false detections. Fast R-CNN could not identify various situations where the object in the target image got blurred. It gives the potential to train the model at various levels of blurred images, as shown in Figures 4-25 and 4-26, to help the detection model identify when the camera is out of zoom.

Fig 4- 25. Image transformed at blur level (3.6 & 8.0)

Fig 4- 26. Image transformed by applying zoom blur (feature 26) & zoom motion blur effect (feature 26)

Fig 4-27 shows a few results from YOLO v3, where the camera is out of zoom, but the model was able to identify the target object with more than 90% confidence. It gives a firm base for our object classification model.

All these data augmentation strategies were applied with the help of PhotoScape. A tool helps apply different image editions. Different learned data augmentation policies can be used, but they are not in the scope of this thesis work.

After employing all these data augmentation strategies, we increased the dataset by about 50% of the actual size. After utilizing these strategies, the model could identify drones under low inference time with high confidence scores.

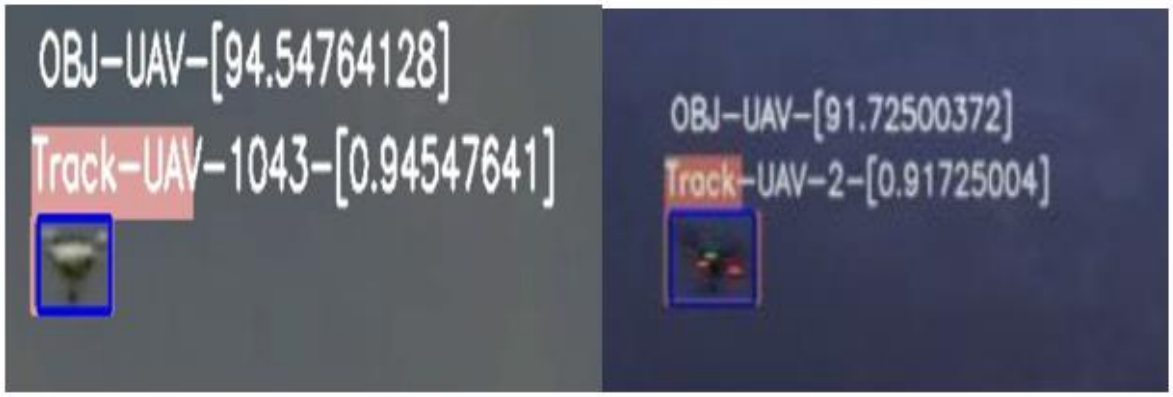


Fig 4- 27. Drone detection for blur images

It can be observed from Fig 4-27 that a drone is zoomed in, and blurred images could be identified with high confidence during real-time experiments.

Chapter 5. Results & Discussion

5.1. Introduction

This chapter starts with an evaluation of a training analysis of the optical model, which continues to a real-time model analysis. Also, the performance of the developed model was compared with two other models. Some limitation of the required sensor was recorded in the first few experiments, which is the motivation to have more sensors to evaluate the results.

The Chapter continues with a required analysis based on different features, i.e., evaluating the performance of an optical model based on the range and then evaluating the model on the system level, i.e., the overall results after the sensor fusion in a real-time system.

5.2. Training Accuracy

Training Deep Neural Networks like YOLOv3 requires a system to run the process, as it is expensive and can take a long time to train the model and update the weighting parameters. The system configuration shown in Fig 5-1 was utilized in the methodology.

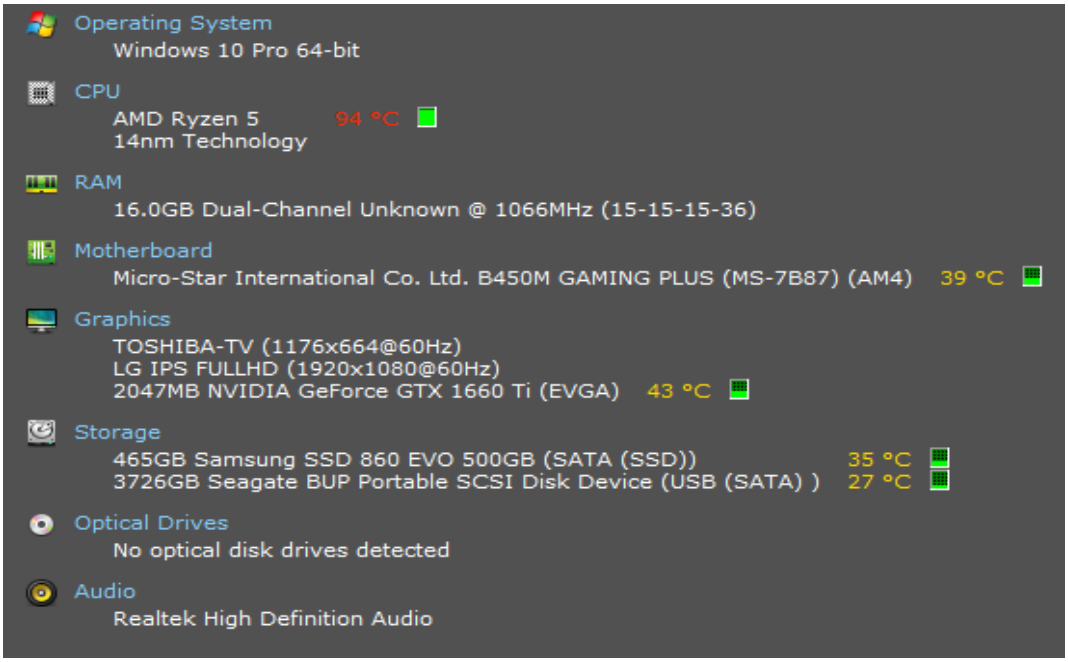


Fig 5- 1. System specification

GPU, 2047MB NVIDIA GeForce GTX 1660 Ti (EVGA) was utilized for the training purposes, which starts with a vast training loss (entropy-loss) from 30000, which gradually decreased to 1 after 1000 iterations and then to 0.025 after 6000 iterations. It took approximately 8 h to train the required model with two classes.

The parameters for training are given in Table 5-1.

Table 5- 1. Parameters used for training.

Parameter	Value
Batch	64
Subdivisions	16
Width	416
Height	416
Channels	3
Momentum	0.9
Decay	0.0005
Angle	0
saturation = 1.5	1.5
Exposure	1.5
Hue	0.1
Learning_rate	0.001
Burn_in	1000
Max_batches	6000
Policy	Steps
Steps	4800,5400
Scales	.1,.1

The curves shown in Fig 5-2 and 5-3 depict the real-time training loss with the help of a tensor board.

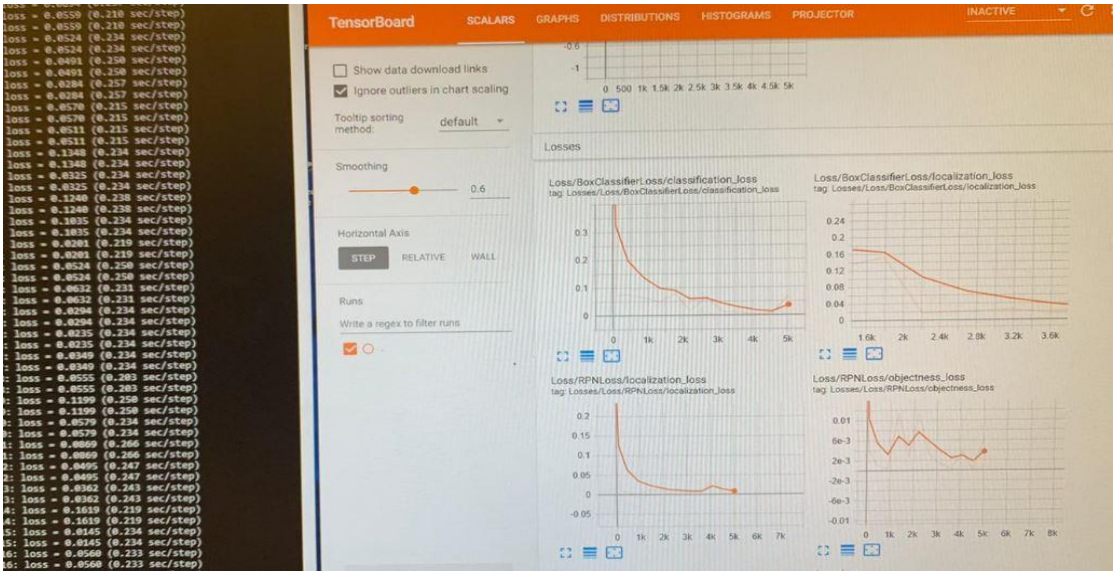


Fig 5- 2. Training loss curves

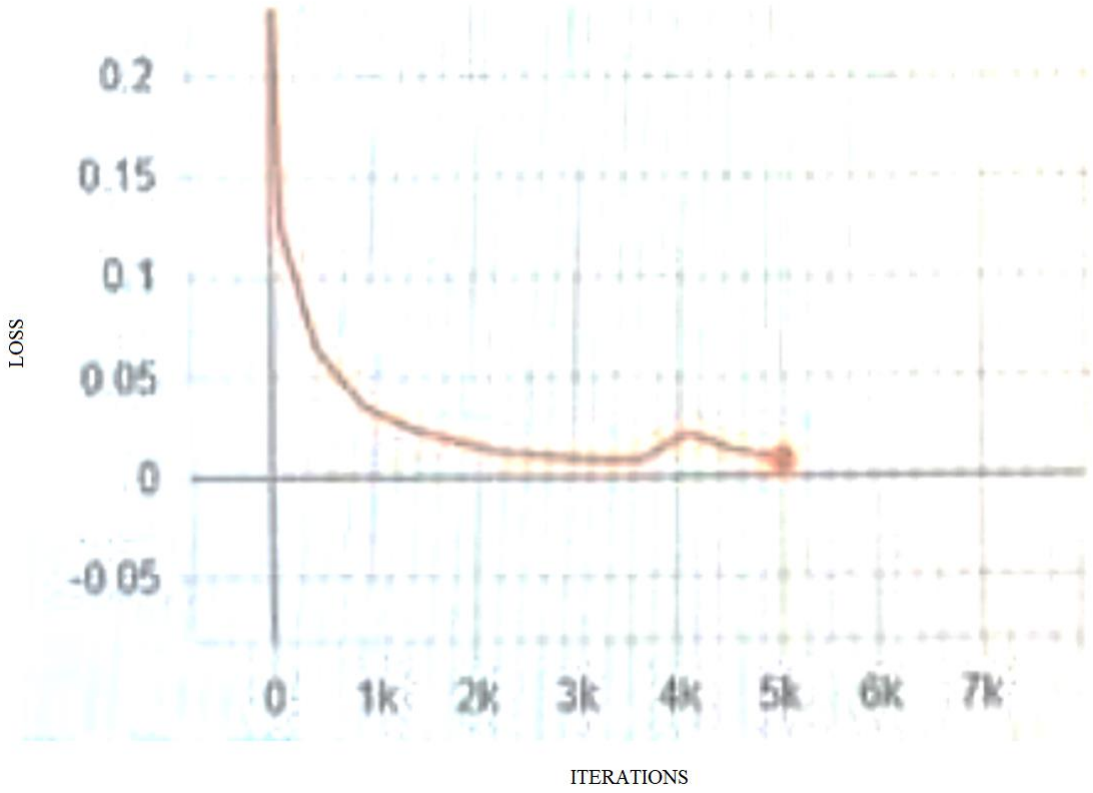


Fig 5- 3. Loss curve

5.3. Validation Accuracy

The dataset used is divided into validation and test set. Validation was performed on the minimal count of the dataset containing 268 images. The validation results are given in Table 5-2. Also, Fig 5-4 depicts the screenshot of the validation results.

Table 5- 2. Validation results

Performance metric	Value
Average Precision	89.94%
False Positive	14
True Positive	232
IOU threshold	50%
For Conf-Threshold	0.25
F-1 Score	0.91
Recall	0.87
Precision	0.94

```
 calculation mAP (mean average precision)...
268
 detections_count = 300, unique_truth_count = 266
class_id = 0, name = UAV, ap = 89.94%           (TP = 232, FP = 14)
class_id = 1, name = BIRD, ap = 0.00%           (TP = 0, FP = 0)

 for conf_thresh = 0.25, precision = 0.94, recall = 0.87, F1-score = 0.91
 for conf_thresh = 0.25, TP = 232, FP = 14, FN = 34, average IoU = 75.52 %

 IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
 mean average precision (mAP@0.50) = 0.449682, or 44.97 %
Total Detection Time: 244 Seconds
```

Fig 5- 4. Validation results

The comparison results were made with the YOLOv2 detector [65], and the F1-score was around 0.728. The proposed model was able to outclass the evaluation matrix.

5.4. Summary of Experiments and Results

Each experiment in the following sections will follow the flight description very closely, as shown in Table 5-3.

Table 5- 3. Flight Description

	Flight Description	Altitude (m)	Range (m)	Take Off (Montreal Time)	Landed (Montreal Time)
1	Radial And tangential	60	230	10 Am	10:15Am
2	Constant Velocity, with back and forth and right and left movement of the drone.	60	250	11:07 Am	11:24 Am
3	Ascent and descent, Varying altitude, range, and Velocity (0-10)	0-40	0-497	11:27 Am	11:41 Am
4	Birds were recorded during this flight	-	-	11:42 Am	11:50 Am

5	Birds were recorded during this flight	-	-	12:42 Pm	
6	Battery issues landed abruptly.	-	-	12:54 Pm	12:56 Pm
7	Constant Velocity, back and forth, and hovering around for a few minutes.	57m	500-900	1 Pm	1:23 Pm
8	Circular movements were recorded	60	500-900	1:32 Pm	1:39 Pm
9	Birds were recorded, perpendicular flight for the drone.	120	500-900	1:45 Pm	1:59 Pm
10	Birds were recorded, and the flight was back and forth	80	500-900	2:03 Pm	2:10 Pm
11	Zig Zag patterns and diving throughout the flight:		200m	2:12 Pm	2:25 Pm

12	Zig Zag patterns and diving throughout the flight:	60	200m	2:40 Pm	2:55 Pm
13	Sudden movements were recorded for this flight.	60	250m	3:21 Pm	3:37 Pm

Dataset 1:

There were around 137 videos, each around 11-40s in duration, corresponding to 11 tracks, i.e., 15 min each. Most of the videos were out of zoom, blurry, shaky, and not zoomed properly towards the object. There was no synchronization between radars and optical data. XYZ coordinates and time were not synchronized with the optical data. With this enormous data of around 137 videos, only 49 showed some objects within the frames. Weather conditions were cloudy and were not suitable for the experiment. YOLO object tracker did not lose the tracking object throughout the video, considering the UAV was within the frame. Also, the Confidence recorded was above 90%. The description of dataset 1 is given in Table 5-4. A few samples of dataset 1 are given in Fig 5-5

Table 5- 4. Description of dataset 1.

Date	28th November 2019
Location	Port of Montreal
Radar	Obsidian Radar, QinetiQ
Drone	DJI Phantom 3 Advanced Quadcopter
Total Number of videos	137

Training & Testing	Fewer frames were utilized for training and testing purposes
Results	Scores recorded on the testing set were above 90%.
FPS	14

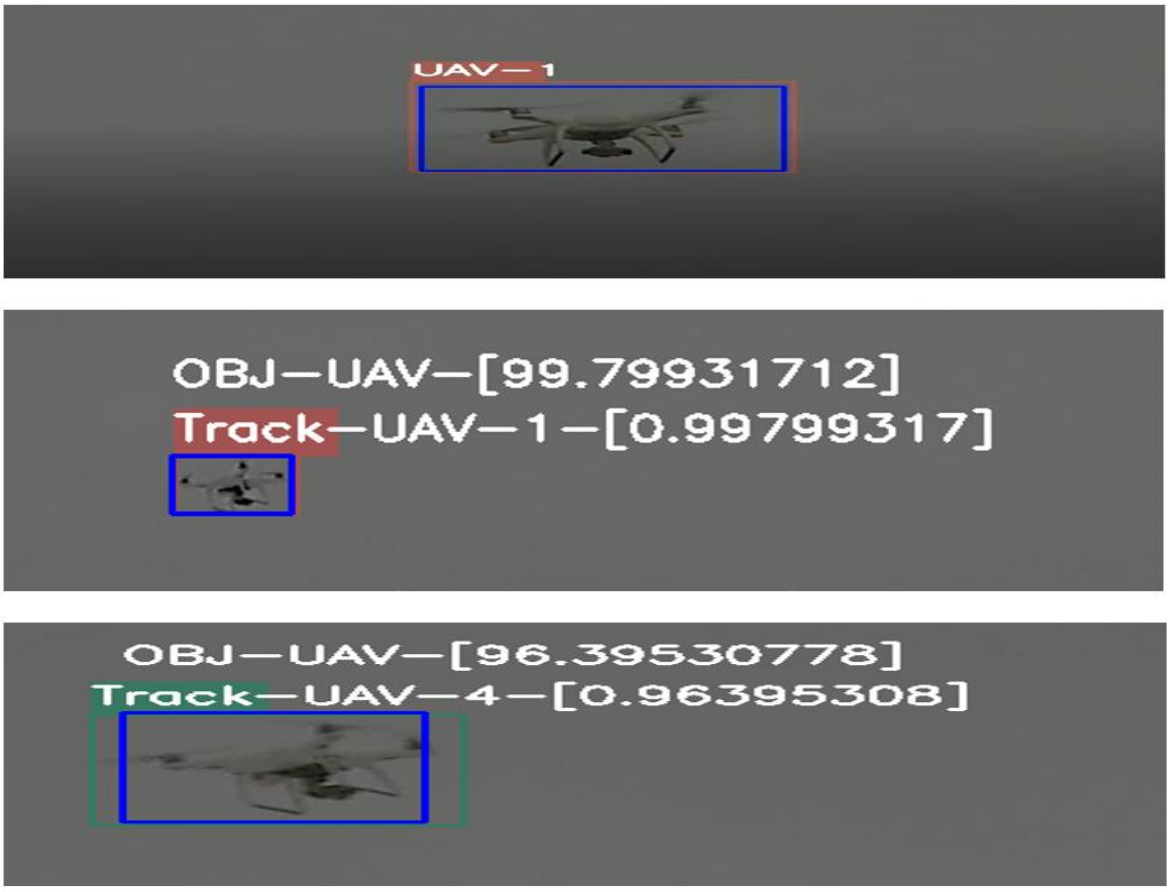


Fig 5- 5. Samples of dataset 1.

Dataset 2:

There was just one video around 14m long in dataset 2. It was shaky and out of zoom throughout the video, and there was no synchronization between radars and optical data. Additionally, XYZ Coordinates & time were not synchronized with the optical data, and only a few frames throughout the video were zoomed properly on drones. Also, only around 2 min

out of 14 had some information, and the Confidence recorded was above 90%. The description of dataset 2: System_test_1_Laval is given in Table 5-5. A few samples of dataset 2 are given in Fig 5-6.

Table 5- 5. Description of dataset 2

Date	12th May 2020
Radar	Obsidian Radar, QinetiQ
Drone	DJI Phantom 4 Pro
Total Number of videos	1
Training & Testing	Fewer frames were utilized for training and testing purposes
Results	Scores recorded on the testing set were above 90%.
FPS	14

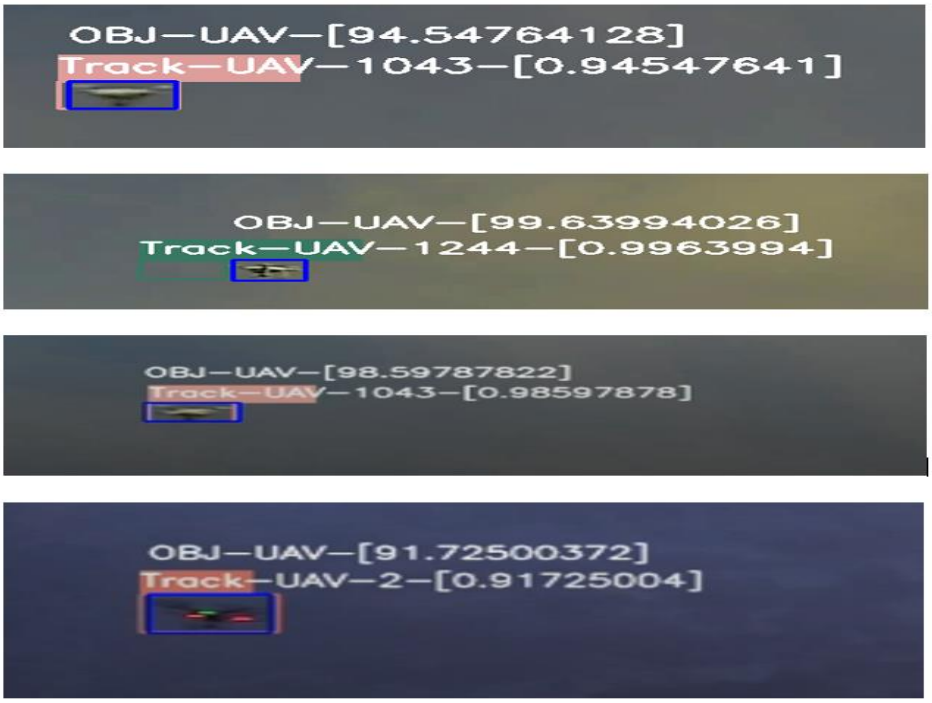


Fig 5- 6. Samples of dataset 2

Dataset 3: System_test_2_Laval

In dataset 3, there were around 37 videos, each around 30 sec-22 mins in duration. At the same time, most videos were out of zoom, blurry, shaky, and not zoomed properly towards the object. There is no synchronization between radars and optical data. XYZ Coordinates & time were not synchronized with the optical data. Moreover, only around seven videos had some objects shown within the frames. Also, fewer frames of around 5 min were correctly zoomed in and had some excellent information. Weather conditions were also cloudy. YOLO object tracker did not lose the tracking object throughout the video, considering the UAV was within the frame and the Confidence recorded was above 90%. The description of dataset 3 is given in Table 5-6. A few samples of dataset 3 are given in Fig 5-7.

Table 5- 6. Description of dataset 3

Date	9th June 2020
Radar	Obsidian Radar, QinetiQ
Drone	DJI Phantom 4 Pro

Total Number of videos	37
Training & Testing	Fewer frames were utilized for training and testing purposes
Results	Scores recorded on the testing set were above 90%.
FPS	14

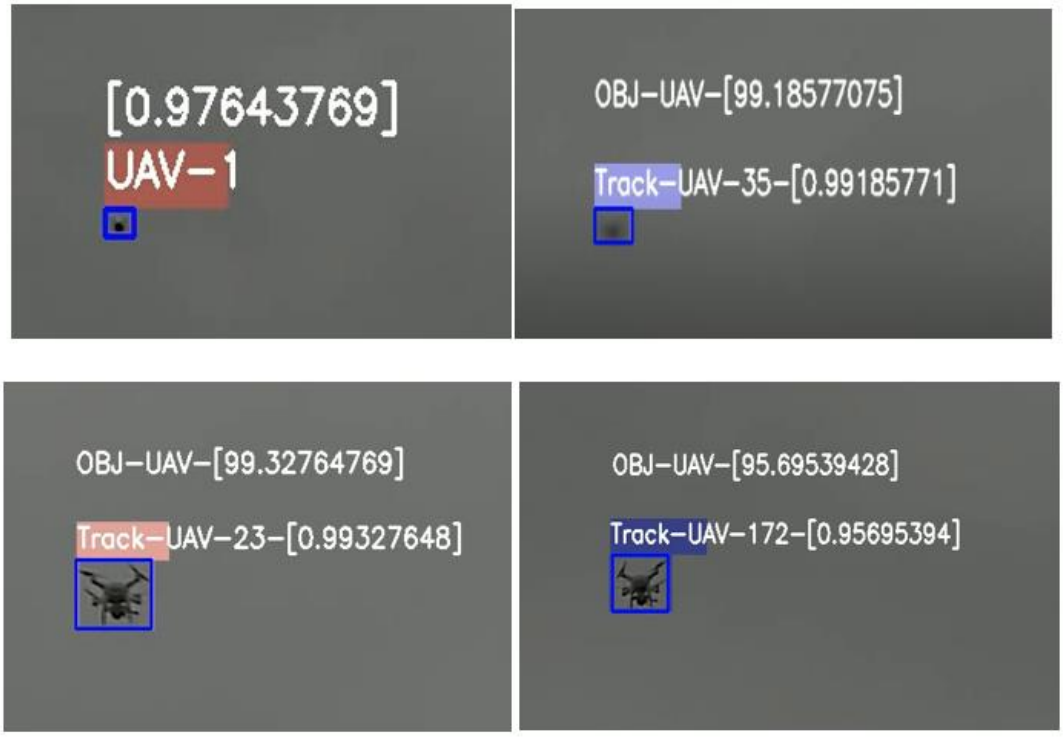


Fig 5- 7. Samples of dataset 3

Dataset 4: Drone-vs-Bird detection challenge at IEEE AVSS2017 [85]

The Videos contain the DataSet having UAVs and a few birds. Most of the videos are around 8s. The dataSet contained synchronized positions with videos. However, there are no radar coordinates. UAV was visible throughout the video, not shaky, and correctly zoomed in. YOLO object tracker did not lose the tracking object throughout the video. Also, the Confidence was above 90%. The description of dataset 4 is given in Table 5-7. A few samples of dataset 4 are given in Fig 5-8.

Table 5- 7. Description of dataset 4

Total Number of videos	7
Training & Testing	Fewer frames were utilized for training and testing purposes
Results	Scores recorded on the testing set were above 90%.
FPS	14

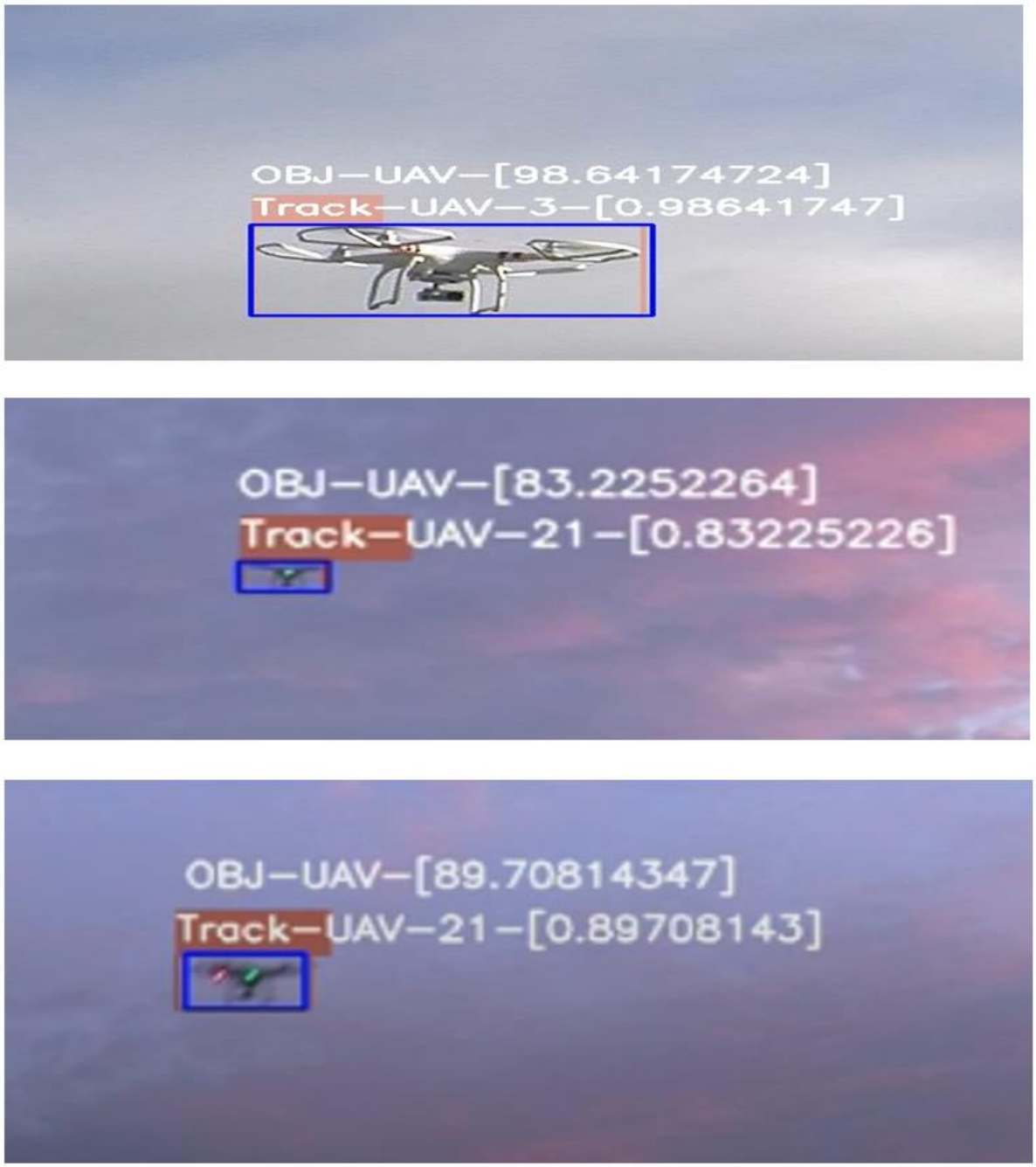


Fig 5- 8. Samples of dataset 4

Dataset 5: PTZ Bosch Camera

For dataset 5, the camera was shaky and moving very fast and was not steady in a few of the samples (maybe mainly because of operating manually), and it can be visible in the first few videos of the data shared. Only fewer frames throughout the video captured UAV

adequately. Many birds were recorded in the frame too. It seems to have a limitation of around 200 ft with a wide 1x range. A proper field of view beyond 500 m is not shown accurately.

Moreover, most frames were out of zoom, even with Tele30x. Also, there is no synchronization between radars and optical data. XYZ Coordinates & time were not synchronized with the optical data. Videos in the dataset are around 30 sec to 2 min range.

The description of the dataset is given in Table 5-8. A few samples of dataset 4 are shown in Fig 5-9.

Table 5- 8. Description of dataset 5

Date	9th June 2020
Total Number of videos	27
Digital Zoom	12x
Results	Scores recorded on the testing set were above 90%.
FPS	14

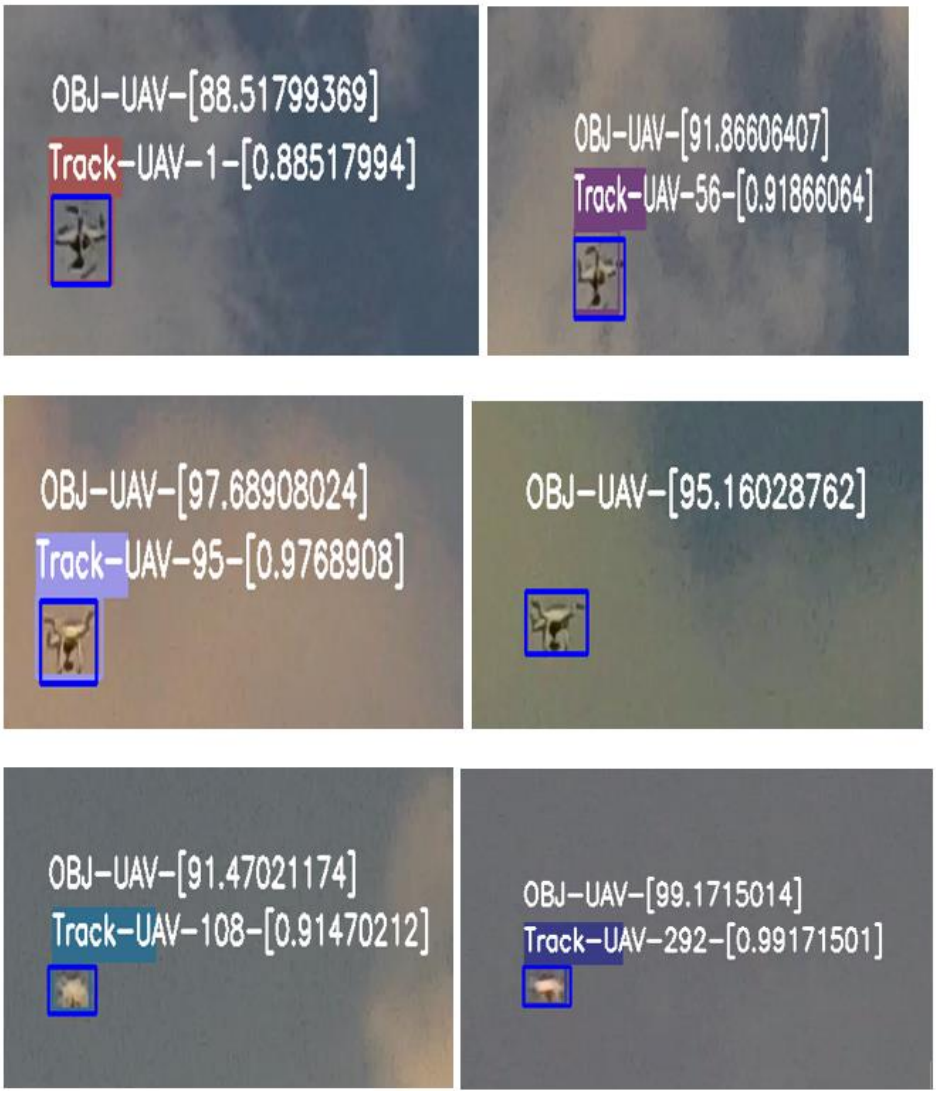


Fig 5- 9. Samples of dataset 5

Fig 5-10 provides the screenshot of experimental observations collected on different scenarios, i.e., horizontal, vertical, Straight flights, sharp turning angles, and hovering with distance beyond 0-1 km, and altitude was recorded from 0-120 m. The model detected drones and birds with an accuracy of around 98% and 88% in each scenario at varying altitudes and ranges. Time in detection was also recorded as an approximate value in seconds.

A model could detect even a tiny-sized drone with greater accuracy, which was not detected in a few comparisons in the next section. A few potential problems were occlusion and weather/lightning conditions that can be dressed in sensor fusion.

EXPERIMENTS	Video Length min:sec	Drone Detected on Screen (Sec) (Approx)	Drone Detected On YOLO (Sec) (Approx)	Score (%) (Approx)	Drone Not Detected On YOLO (Sec)	Time In Detection (Less Than) (Approx)	False Detections (Sec) (Approx)
Test1	06:12	36	36	99	0	1	0
Test2	12:40	247	244	98	3	1	0
Test3	15	12	12	99	0	1	0
Test4	08:39	22	17	97	5	1	0
Test5	10:57	153	142	98	11	1	0
Test6	07:15	387	387- (2s aircraft)	97	0	1	0
Test7	10:32	130	130	99	0	1	18 (10 Tower) (8 Bird)
Test8	14:00	233	228	99	5	1	0
Test9	05:05	36	36	98	0	1	0
Test10	04:02	42	42	98	0	1	0
Test11	05:10	205	205	98	0	1	0
Total	**84:78**	**1503**	**1479**		**24**		**18**

Fig 5- 10. Experimental Observations

5.5. Optical Object Detection (UAV) vs. Range

The following analysis was based on experimental results on datasets, i.e., systems Laval 1, 2 & 3 with radars (QinetiQ) and Optical (Bosch Camera) on YOLOv3 tracking and object detection. The radar dataset determined the optical dataset range, and the YOLOv3 score was plotted against this range, as shown in Fig 5-11. It should be noted that radar and optical are synchronized on the same clock where radar collected all tracks with UNIX time format, so we have converted this on local time (Laval Time) to see the detected object with optical data.

Also, it is essential to discuss that UAV covered 29% of the screen of the total video length, so there were many points where we could not see the object when radar points were synchronized with the optical. After analyzing all Laval experiments, a few results are as follows:

UAVs were detected mainly with (a greater than 90% score) where the distance from the radar was around 0-700 meters. It was realized that the range (distance from the radar) was limited to 0-200 meters in 80% of the overall experiment. The object size was zoomed to 5-10% of the screen size in all collected datasets. The Velocity of UAVs was around 3-5 m/s in all collected datasets. Fig 5-11 shows the Range vs. Score plot.

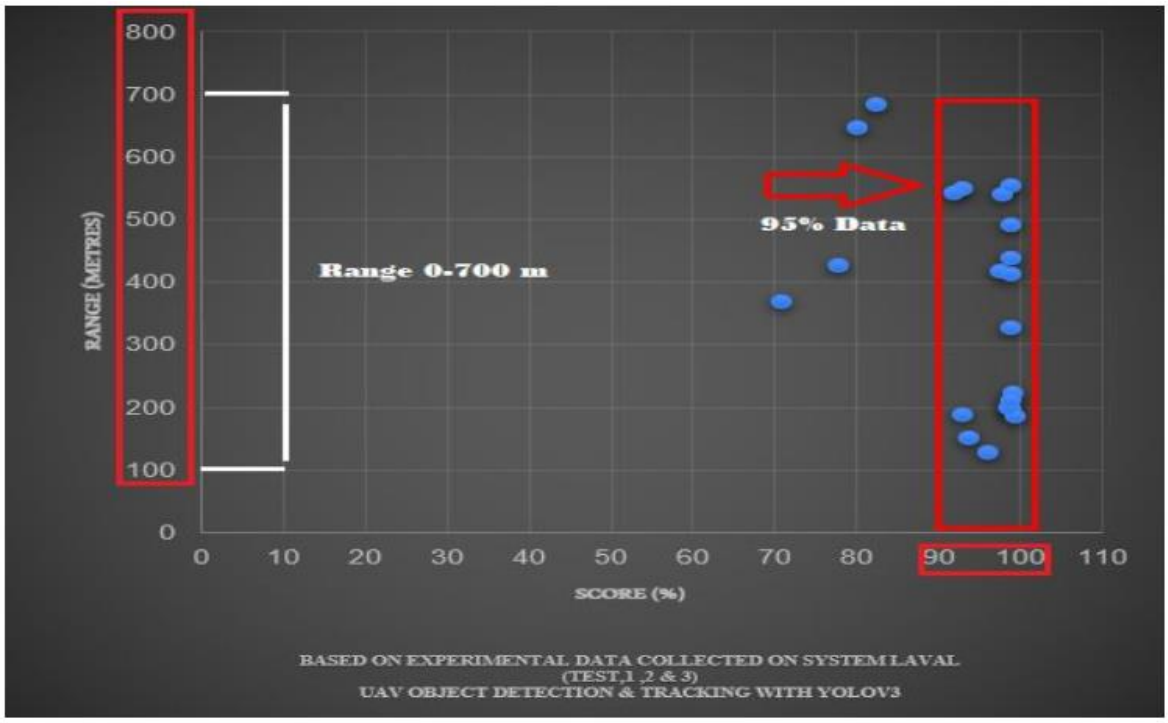


Fig 5- 11. Range vs. YOLOv3 scores

Comparison Between Faster RCNN Inception V2 With YOLOV3

RCNN is a region-based convolutional neural network that uses the selective search technique to extract regions from the image. Instead of classifying the whole image, this selective search technique will allow the classification of only the specified regions for increased computational power. The selective search technique proposes a candidate region, which utilizes the greedy algorithm to combine similar regions into larger ones with recursive

topology. Then, the final proposed regions to carry out the classification are acquired. In Faster RCNN, the proposed RCNN framework is updated to allow the network to learn those proposed regions instead of utilizing selective techniques to propose 2000 regions. A separate network is defined to produce the proper regions, which are further reshaped using a pooling layer and then sent for classification. IoU over here defines the presence of an object with some threshold value.

The proposed RCNN and faster RCNN framework do not help much in real-time applications, mainly because of the selective search regional base framework. However, YOLO, mainly designed for real-time applications, looks at the whole image on run time and then proposes predictions by the global context. RCNN requires thousands of proposed evaluation networks for a single image. YOLO suits real-time applications where all predictions are defined inside a single evaluation network. Fig 5-12 depicts the result of Faster RCNN. Fig 5-12 also indicates the probability score of the detected UAV; like in the first left image, two UAVs are detected with a score of 91% and 71%.

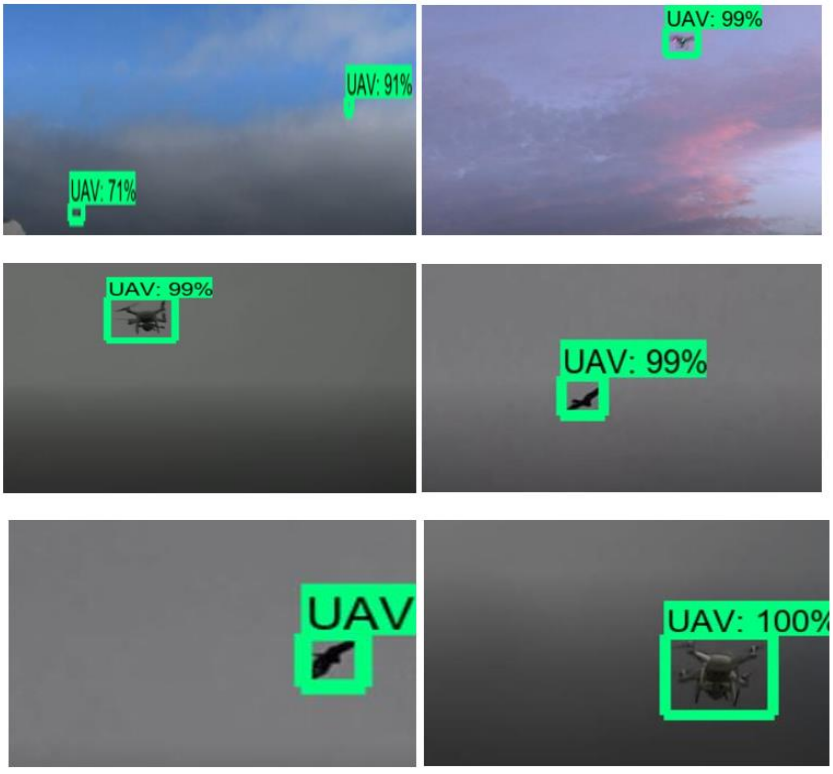


Fig 5- 12. Results of faster RCNN

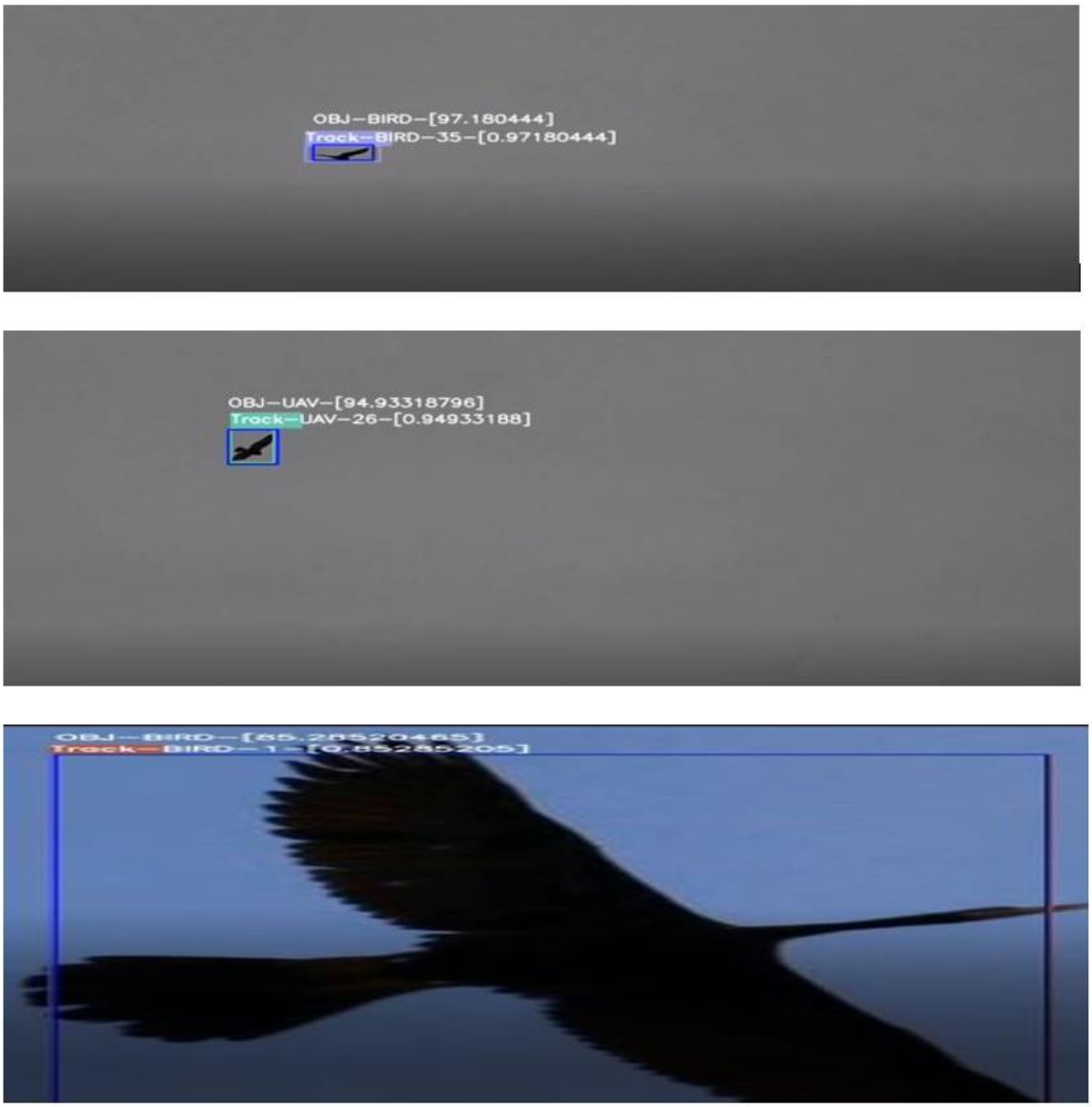


Fig 5- 13. Results of YOLOv3

The results of the YOLOv3 are depicted in Fig 5-13. It also provides the probability score of detection and the bounding box around the detected object. It detects both birds and UAVs.

One more short comparison was made using one another model, i.e., Fast R-CNN architecture via Keras-Retinanet Implementation. Faster R-CNN is ten times faster than Fast

R-CNN and unsuitable for real-time applications [84]. Also, the above model is unsuitable for small objects.

Figures 5-14 and 5-15 show a few events of a drone flying where Fast R-CNN was unable to detect, and in a few cases, it does make a detection with low Confidence of 39% and at 1 FPS (Frames/Second), respectively.

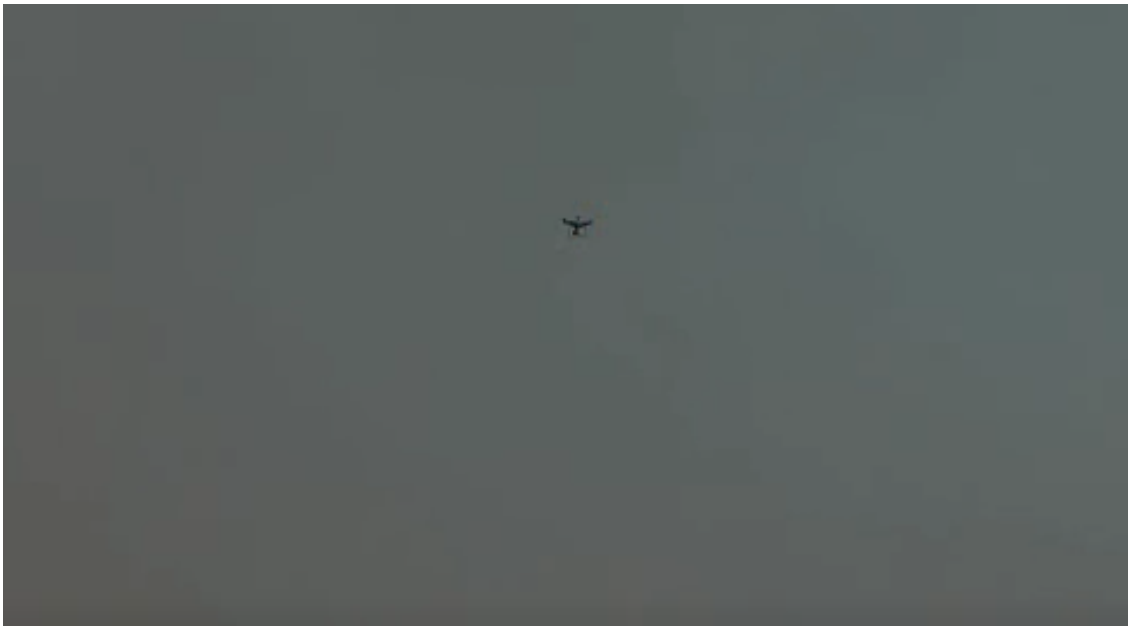


Fig 5- 14. No detection

Fig 5- 15. Detection made with low confidence of 0.393.

YOLOv3 tracker could detect and track all frames of a real-time scenario of a tiny drone at 14 FPS (Frames/Second), as depicted in Figures 5-16 and 5-17.

Fig 5- 16. YOLOv3 tracker made the detection with 90% confidence.

Fig 5- 17. YOLOv3 tracker made the detection with 99% confidence of a tiny drone.

The observations on model performances for YOLOv3 and Faster R-CNN, based on different criteria such as speed, accuracy, training time etc., are summarized in Table 5-9.

Table 5- 9. Comparison between YOLOv3 vs. Faster R-CNN

Criteria	**YOLOv3**	**Faster R-CNN**
Speed	Fast	Slower than YOLOv3
Accuracy	Higher than Faster R-CNN	Lower than YOLOv3
Training time	Faster than Faster R-CNN	Slower than YOLOv3
Detection range	Good for small objects	Good for large objects
Multiple objects in the image	Performs well	Performs well
Non-maximum suppression	Incorporated in model	Separate post-processing step
Anchor boxes	Not used	Used to propose object locations
Model complexity	Less complex	More complex
Deployment	Light-weight model, easy to deploy	Heavier model, harder to deploy

5.6. Sensor Fusion at the System Level

While we have compared our optical classifier's performance with two other models that have gained a significant reputation and have a refined set of results that define their performance in various scenarios, it is now necessary to assess its performance at a system level by fusing both radar and optical data.

The classification is performed on the data acquired from the radar and optical sensors. The following Figures use the video frames used as input for classification. Fig 5-18, 5-19,5-20, and 5-21 are the few frames of the input video for which classification is performed.

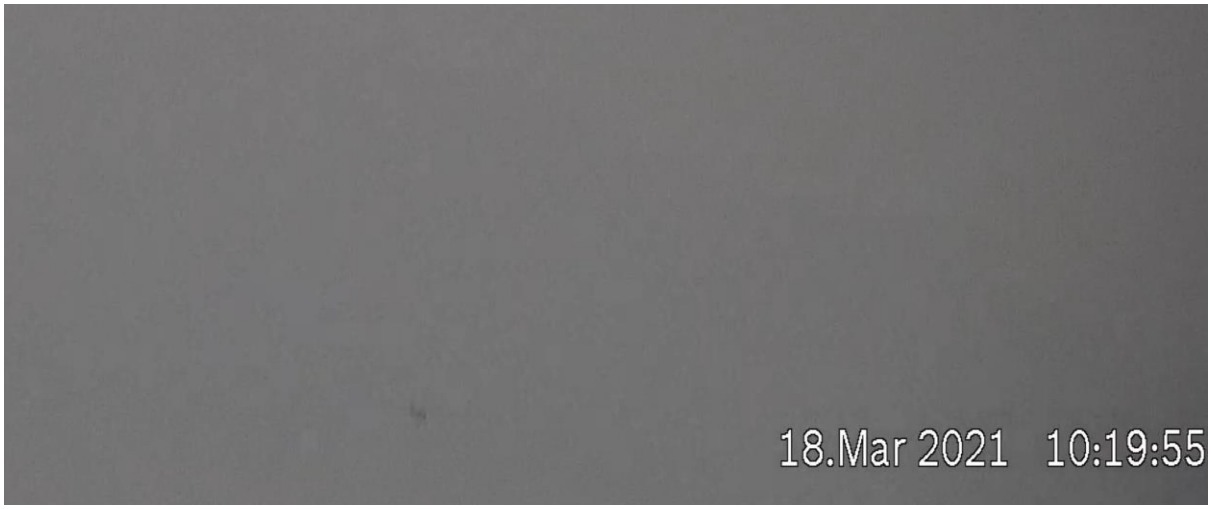


Fig 5- 18. Frame 1 of the input video

Fig 5- 19. Frame 2 of the input video

Fig 5- 20. Frame 3 of the input video

Fig 5- 21. Frame 1 of the input video

Fig 5-18 shows the slight view of the bird in the video frame. Even the tiny view is detected and classified as a bird with the system id 11. System id is the unique id provided to each detected and classified object. Fig 5-19 shows the UAV; thus, the proposed algorithms detect it and classify it as UAV with the system id 1. Fig 5-20 shows the person walking in the snow. During classification, it is detected and classified as a human with the system id 6. Fig 5-21 shows the UAV. It is detected and classified as UAV with the system id 101.

In real-time, three samples of a track show the input response from the radar fed through our radar and optical system to get fused predictions. The trackid is a unique id to identify which tracks were sent so that we can filter the corresponding track id when responses are received. Also, the deep sort will keep track of these track ids to identify if it is the same drone.

The various input features obtained through radar are fed to the proposed AI radar model to get predictions. These features are azimut, elevation, range, xCoordinate, yCoordinate, zCoordinate, xVelocity, yVelocity, m_zVelocity, and crossSect.

The input from the radar is given in Table 5-10.

Table 5- 10. Input from radar

header	trackID	azimut	elevation	range	xCoordinate	yCoordinate	zCoordinate	xVelocity	yVelocity	m_zVelocity	crossSect	timestamp
TRE	11	-1.98	-12.92	682.13	-22.984	-152.615	664.44	0.29	1.90	-6.216	-14.51	b'20210310T153651.298Z'
TRE	11	-1.96	-12.93	681.49	-22.76	-152.515	663.81	0.35	1.897	-6.24	-14.641	b'20210310T153651.401Z'
TRE	11	-1.96	-12.8	680.78	-22.73	-151.45	663.33	0.35	2.043	-6.24	-14.84	b'20210310T153651.504Z'

Fig 5-22 shows optical input response was taken in the real-time system for the same response.

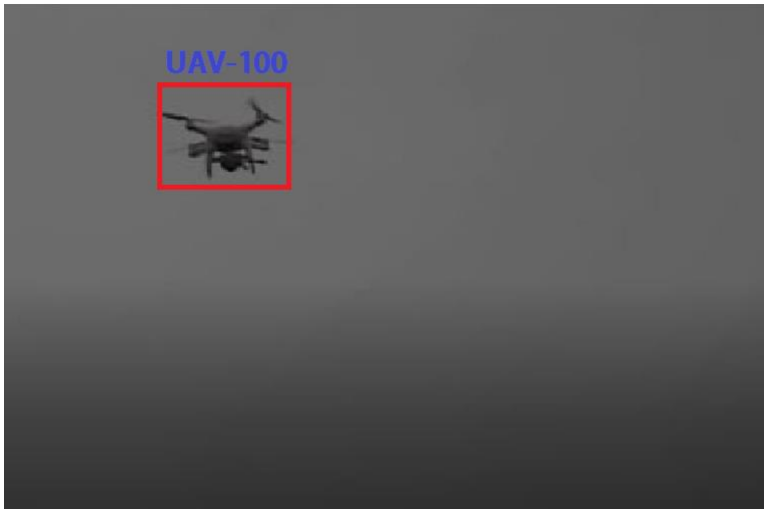


Fig 5- 22. Optical input response

Table 5-11 shows the output response/logs returning against the corresponding time sample with fused predictions, Classification into four categories UNKNOWN represented by 0, UAV represented by 1, BIRD represented by 2, and OTHER represented by 3. In this case, for trackID

11, logs representing the classification 1, i.e., UAV with confidence 100% and fused prediction as UAV.

The following table, 5-11, includes a header row that identifies the data in each column. The trackID column contains unique identifiers assigned to each object being tracked, while the classification column contains the label or category assigned to each object based on the algorithm's analysis, as explained above. The confidence column contains a measure of the fused predictions in its classification. The topLeftX, topLeftY, bottomRightX, and bottomRightY columns contain the coordinates of the bounding box around each object. The timestamp column indicates the time at which each object was detected and tracked. The ML_TRACK_PROB column contains a probability score that the algorithm is tracking the object correctly. Finally, the Fused_Classification column contains a classification label generated by combining scores from radar & optical. Overall, this table provides detailed information about the objects being tracked and classified by the machine learning algorithm.

Table 5- 11. Output response

header	trackID	classification	confidence	topLeftX	topLeftY	bottomRightX	bottomRightY	timestamp	ML_TRACK_PROB	Fused_Classification
RSP	11	1	100	351	150	376	162	b'20210310T153649.777Z'	None	UAV
RSP	11	1	100	351	150	376	162	b'20210310T153649.779Z'	None	UAV
RSP	11	1	100	351	150	376	162	b'20210310T153649.781Z'	None	UAV

Table 5-12 shows the input logs from radar for another track with track id 18. Each column in the header provides information about the detected objects from the radar, such as their unique identifier (trackID), horizontal and vertical angles relative to the radar system (azimut and elevation), distance from the radar system (range), and Cartesian coordinates in three-dimensional space (xCoordinate, yCoordinate, zCoordinate). Additionally, the radar system measures the velocity of the detected objects along each coordinate axis (xVelocity, yVelocity, zVelocity), as well as the radar cross section (RCS) of each object, which is a measure of how much radar energy the object reflects back. Overall, the radar system provides detailed information about the location, velocity, and size of detected objects, which is shared as the input response below.

Table 5- 12. Input from radar for track id 18

header	trackID	azimut	elevation	range	xCoordinate	yCoordinate	zCoordinate	xVelocity	yVelocity	zVelocity	crossSect	timestamp
TRE	18	-0.9668	-6.1556	44.769215	-0.7510834	-4.800622	44.504745	1.355479	0.3508082	-3.21617	-13.4317	b'20201218T152115.220Z'
TRE	18	-0.400	-5.9098	43.412033	-0.3017025	-4.4698248	43.180252	1.3805618	0.4756557	-3.4285	-13.6366	b'20201218T152115.529Z'
TRE	18	1.277	-5.8094	41.30801	0.9165764	-4.1812434	41.085629	1.827803	0.4964186	-3.6033	-13.4939	b'20201218T152116.044Z'
TRE	18	5.400	-7.8394	37.129131	3.4616089	-5.0643167	36.618881	1.36443	-0.2538513	-2.9990	-16.4055	b'20201218T152117.691Z'
TRE	18	6.14075	-8.6760	36.582516	3.8684998	-5.5183983	35.956398	1.05644	-0.4379734	-2.91606	-17.3132	b'20201218T152118.103Z'
TRE	18	-39.583	-17.014	619.78363	-377.64111	-181.35782	456.75845	-8.93480	-5.3811178	9.09475	-14.904	b'20201218T152302.207Z'
TRE	18	-39.650	-17.0506	621.20142	-378.96942	-182.14633	457.27026	-9.04431	-5.4083643	9.00378	-14.8941	b'20201218T152302.310Z'
TRE	18	-39.71	-17.101	622.57983	-380.21564	-183.0826	457.73621	-9.10761	-5.4501915	8.88288	-14.9825	b'20201218T152302.413Z'
TRE	18	-39.81	-17.0840	624.0459	-381.96262	-183.32828	458.17963	-9.32523	-5.3780107	8.76625	-15.0485	b'20201218T152302.517Z'
TRE	18	-39.87	-17.1020	625.44818	-383.25137	-183.92845	458.7746	-9.4015	-5.3703766	8.6823692	-15.0512	b'20201218T152302.620Z'

Optical Input:

Fig 5-23 shows the second screenshot that was taken at the point of a real-time system.

Fig 5- 23. Optical output response

Table 5-13 represents the output response/logs from the real-time system for trackID 18, which represents the classification 1, i.e., UAV at confidence 100% with fused classification as UAV.

Table 5- 13. Output response for track 18

header	trackID	classification	confidence	topLeftX	topLeftY	bottomRightX	bottomRightY	timestamp	ML_TRACK_PROB	Fused_Classification
RSP	18	1	100	1036	825	1090	855	b'20201218T152111.444Z'	None	UAV
RSP	18	1	100	1036	825	1090	855	b'20201218T152111.625Z'	[0.015, 0.945, 0.04]	UAV
RSP	18	1	100	1041	852	1084	886	b'20201218T152112.443Z'	None	UAV
RSP	18	1	100	944	809	996	841	b'20201218T152114.120Z'	None	UAV
RSP	18	1	100	913	784	958	819	b'20201218T152114.442Z'	None	UAV
RSP	18	1	99.68936443	1087	575	1123	603	b'20201218T152258.447Z'	None	UAV
RSP	18	1	99.68936443	1087	575	1123	603	b'20201218T152258.449Z'	None	UAV
RSP	18	1	99.68936443	1087	575	1123	603	b'20201218T152258.451Z'	None	UAV
RSP	18	1	99.68936443	1087	575	1123	603	b'20201218T152258.459Z'	None	UAV
RSP	18	1	99.82967973	1081	541	1119	570	b'20201218T152259.119Z'	[0.17,0.82, 0.01]	UAV

Fig 5-24 shows increasing range vs. confidence to see if our classifier can track the defined object. The x-axis shows the confidence scores, while the y-axis shows the range of UAVs in m. Observation shows the model performance in increasing range. The proposed model was able to identify UAVs with confidence scores above 99% at a range of 1km (approx). These observations are very crucial, as discussed in previous sections. This research proposes a novel solution, and a few papers have discussed the performance of an optical detection model based on key features like range and many more. The underlying evaluation will show the model performance up to approximately 1 km.

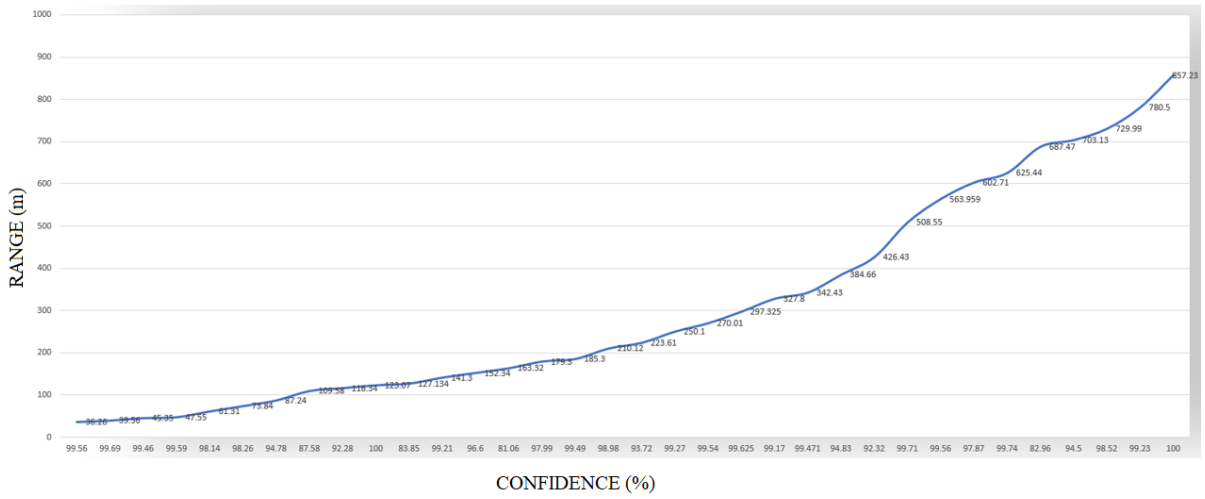


Fig 5- 24. Range vs. confidence

5.7. Optimization of real-time performance

It was observed during real-time experiments that the system had a delay of around a few seconds and needed some optimization before it could get ready for the final demo. Amer Alhalabi was working on the same project and assisted in optimizing the performance of the real-time system to 60% faster, so if the initial classification rate was 14 FPS, we could optimize the real-time classification performance to 24 FPS.

The proposed model optimized the performance of a real-time system to (approx.) 150 ms for an entire cycle of receiving and processing that includes the fused predictions and sending the response back through TCP sockets to the C2 communication module. The few observations and steps taken to optimize the performance are given below.

Removed unnecessary operations and functions in the optical module and cleaned the code to optimize it with a few milliseconds.

The cost of running different operations on the optical classifier in the code is given in the table below.

Table 5- 14. Cost estimation of running different operations on the optical classifier

Operation	Description	Time Taken (ms)
RGB Conversion	Converting the image from BGR to RGB	13@416, 10@256
Image Dimension Change	Expanding the dimensions of the image	13@416, 10@256
Image Transformation	Transforming the image to the desired size	13@416, 10@256
YOLO Prediction	Running YOLO to detect objects in the image	160@416, 100@256
Bounding Box Conversion	Converting bounding box coordinates to fit original image size	6@416, 2@256
Feature Extraction	Extracting features from the image using an encoder	6@416, 2@256
Detections Creation	Creating object detections from YOLO output	6@416, 2@256
Color Map Creation	Initializing color map for visualization	1
Non-Max Suppression	Running non-max suppression to filter detections	1
Tracker Predict/Update	Predicting and updating tracker using detections	Negligible
FPS Printing	Printing the frames per second on the image	2
Image Display	Displaying the image in a window or GUI	6
Writing Output	Writing output images and detections to a file	6
RGB Conversion	Converting the image from BGR to RGB	13@416, 10@256

These operations involve several steps, such as preprocessing the data and performing object detection and classification. While reducing the size might increase the system's process faster, it reduces accuracy. The rate of classifying each frame was also reduced, reducing the accuracy but making it more optimal for the real-time system.

Based on this analysis, the conclusion was drawn that changing the image size impacts the processing time, with larger images taking longer to process. However, reducing the image resolution also reduced the system's accuracy, meaning that it may not be able to classify all objects in the images correctly.

Despite the reduction in accuracy, the analysis found that reducing the image resolution resulted in a lower rate of classifying each frame, suggesting that the system could process more frames per unit of time. The trade-off between accuracy and processing speed may be considered acceptable since the fused predictions are crucial for real-time processing, and some degree of accuracy reduction can be tolerated. Furthermore, using fused predictions is likely to increase the model's trustworthiness, as it integrates multiple sources of information and reduces the impact of individual errors.

However, it is important to note that the specific trade-off between accuracy and processing speed may vary depending on the application and the specific requirements. It is necessary to carefully evaluate the system's performance under different scenarios and determine the optimal approach that balances the trade-off between accuracy and speed.

Chapter 6. Discussion & conclusion

6.1. Discussion

Drone detection is the primary focus of this research due to its potential importance for several applications. We adopted YOLOv3 with deep sort after applying the data augmentation technique on our custom dataset. We have combined the information from two sensors by using sensor fusion logic, which includes incorporating the work of a radar sensor into our actual system. We described testing procedures and weighting schemes to confirm the outcomes. The proposed sensor fusion logic has enhanced classification and detection results by eliminating false detection and providing an exquisite method of improving confidence based on the limitations of a single sensor.

While the system's performance is generally up to par, some exceptions exist. As discussed in the previous Chapter, there is a restriction on performing experiments on different scenarios mainly due to the limitations imposed on flying UAVs, making it challenging to evaluate the performance of an optical model. However, around ten real-world experiments were conducted to assess the effectiveness of an optical classifier. Due to the low bird population in the area where the installation was set up, the system could not reliably detect many birds during the experiments.

This thesis work aimed to locate drones and develop a reliable model capable of locating drones in real-time with high assurance. All results are included in the evaluation matrix described in the previous Chapter to demonstrate the efficacy of sensor fusion.

In this thesis work, we propose a new method of sensor fusion in which radar tracks provide the region of interest, the camera is whirled to focus on that area, and finally, predictions are made using a YOLO tracker. Another avenue of investigation in this line of study is the development of a deep neural network sensor fusion logic that can combine radar tracks and optical frames into a single data set to increase confidence levels. Synced radar tracks and optical frames need to be used to train the neural network. Because this particular kind of dataset is not readily available, we were compelled to use weighting schemes and various check marks as described in the previous Chapter.

It is possible to evaluate the performance over a limited range, which can be found in the thesis works. Experimenting with additional sensors, such as lidars, lasers, and many more, to extract even more features of these tiny objects and boost the model's performance is an exciting prospect.

Though many complex models have already been applied in this research area, not much has been done to deceive the network by using adversarial neural networks. Also, as we have seen, these models are case-specific and require a specific set of datasets to train the neural network. What happens if we are placed in an environment where these models may not function properly? For example, many commercially available drones are designed to mimic a bird's flight; some even look like birds and perform remarkably similar maneuvers, which can cause confusion and cause an optical model to fail.

The results of the experiments demonstrate many advantages of YOLOv3 over other detection methods. YOLOv3 is significantly quicker than RCNN. Therefore, the YOLOv3 is better suited for use in time-sensitive settings. In addition, fewer hyperparameters are available in the YOLOv3 version compared to the RCNN version. The result is that YOLOv3 has less of an impact on system memory than RCNN. Recognizing distant small objects is another strength of YOLOv3. The disadvantage of RCNN is that it requires a large amount of data to be trained effectively. Due to the advantages of YOLOv3, it outperforms RCNN by achieving the F1 score of 91%. The proposed model is evaluated using standard performance metrics like accuracy, recall, and mean absolute precision—using data from both the test and training sets. Compared to RCNN, YOLOv3 is fast and accurate in detecting small objects. Thus, the time required by YOLOv3 is less for detecting real-time objects than RCNN.

6.2. Future Scope

We intend to test our approach on a larger dataset to demonstrate and utilize the higher versions of YOLO, like YOLO v7, and its small, medium, and large versions. Also, we plan to add more difficult situations and constraints. Specifically, there is a high need for work to include various bird species, atmospheric noises, aerial vehicles, and drones. We will also try to include drones and other objects in ambiguous contexts. These new datasets might require

the development of more complex deep learning models and advanced sensor fusion algorithms.

Appendix A: Detection Level Fusion

The thesis work presented by Svanström in [75] performed a comparative study and a few of the limitations marked by each sensor to fuse the results on decision-level detection. Svanström utilizes the class output and confidence scores of the included sensors by giving each sensor a particular weight scale based on the limitations. The authors analyzed that it was evident that training the network on the feature level requires data, also with papers [70]–[74], [77]–[79], [83].

The work done by Svanström, as described above, on sensor fusion was preliminary and just introduced the basics of sensor fusion but can be used to address the limitations of each of the sensors. Dempster [79] utilizes the evidence theory to explain the ability to aggregate different sensors from different sources over the same frame of discernment which can be used for sensor fusion.

Thus, let's define some notations. Let Ω be the union of the frames of discernment X and Y, i.e., $\Omega = X \cup Y$. The frame of discernment defines the space of possible outcomes or alternatives that we are interested in or concerned with. It is a way of defining the boundaries or scope of the problem, and it determines what types of hypotheses or propositions we can entertain and reason about. In the scope of sensor fusion, frame of discernment, refers to the range of sensor modalities and data sources that are being integrated in order to provide a more complete and accurate understanding of the frame of discernment.

A is the subset of Ω, that represents the proposition or hypothesis under concern.

The Dempster-Shafer rule of combination [79] tells us that to compute the combined belief function m, we need to consider all possible ways that the propositions or hypotheses in X and Y can combine to form the proposition or hypothesis A. Specifically, we need to consider all pairs of subsets Xi of X and Yj of Y such that their intersection equals A, that is, $Xi \cap Yj = A$. X and Y can be considered as range of sensor modalities i .e., camera and radar as in our case. For each such pair (Xi, Yj), we compute a mass or weight of evidence, given by m1(Xi) m2(Yj). This weight of evidence represents the degree of support or evidence for the proposition or hypothesis A, based on the available evidence in the two frames of discernment X and Y.

For each such pair (Xi, Yj), we compute a mass or weight of evidence, given by m1(Xi) m2(Yj). This weight of evidence represents the degree of support or evidence for the proposition or hypothesis A, based on the available evidence in the two frames of discernment X and Y. We then sum up the weights of evidence over all pairs (Xi, Yj) that satisfy the condition Xi ∩ Yj = A. This gives us the numerator of the formula for m(A), which is:

$$\sum_{X_i \cap Y_j = A} m_1(X_i) m_2(Y_j) \qquad \text{(A-1)}$$

Finally, we need to compute a normalization constant K, which is the sum of all possible weights of evidence over all pairs (Xi, Yj) that satisfy the condition Xi ∩ Yj ≠ ∅ (that is, the intersection is non-empty). This gives us:

$$\text{Where } K = \sum_{X_i \cap Y_j = \emptyset} m_1(X_i) m_2(Y_j), m(\emptyset) = 0$$

Combining K above, with equation A-1, we can compute the belief function m(A) for a proposition or hypothesis A in the Dempster-Shafer theory [79], given two input belief functions m1 and m2 defined on two frames of discernment X and Y, respectively as:

$$m(A) = \frac{\sum_{X_i \cap Y_j = A} m_1(X_i) m_2(Y_j)}{1-K} \; for \; A \subset \Omega, \qquad \text{(A-2)}$$

Yager [76] refined these equations as given in Equation A-3. To derive Yager's rule of combination from the Dempster's equation for m(A), we can start by re-expressing the equation A-2 as:

$$m(A) = \sum_{B \cap C = A} m(B) m(C) \,/(1 - K)$$

Where m(B) and m(C) are the belief functions for propositions B & C.

Let's now define a new belief function $m_a(A)$ by considering only the basic belief assignments m(B) and m(C) where we have excluded the cases where B∪C=Ω, which correspond to the subsets B and C that do not support the hypothesis a. These subsets have a combined belief mass of K.:

$$m_a(A) = \sum_{B \cap C = A, BUC \neq \emptyset} m(B) m(C) \,/(1 - K) for \; A \neq \emptyset$$

Next, we can define a new basic belief assignment m'a(B) that is obtained by considering only the basic belief assignments m(B) that support the hypothesis a:

$$m'_a(B) = \frac{m_a(B)}{1-K} (for\ B \neq \emptyset)$$

This assignment represents the belief mass that is assigned to the subset B for the hypothesis a, after excluding the cases where the masses are assigned to the subsets that do not support the hypothesis a.

We can now use the definition of m'a(B) to rewrite the equation for m(A) in terms of m'a(B) as follows:

$$m(A) = \sum_{B \cap C=A, BUC\neq\emptyset} m'_a(B)(1-K)m(C)\ (1-K)\ /(1-K)\text{^}2$$

K from the definition:

$$K = \sum_{B\cap C=\emptyset} m_b(B)m_c(C),$$

Substitute value of K in m(A):

$$m(A) = \sum_{B \cap C=A, BUC\neq\emptyset} m'_a(B)m(C) / \left(1 - \sum_{B \cap C=\emptyset} m(B)m(C)\right)$$

We can use the fact that $m_a(\Omega) = m'_a(\Omega) + K$:

$$m_a(\Omega) = m'_a(\Omega) + \sum_{B \cap C=\emptyset} m(B)m(C)$$

This expression gives the belief mass assigned to the hypothesis for the entire frame of discernment Ω. We can now substitute the expression for K and the definition of ma(Ω) into the equation for m(A) to obtain:

$$m_a(A) = \sum_{B \cap C=A} m_b(B)\, m_c(C)\ for\ A \neq \emptyset \qquad \text{(A-3)}$$

K above is the normalization constant or conflict content. It plays a vital role in combining information between 2 sensors. K can be used as a time threshold for the information between 2 sensors.

The above Equation can be further refined by introducing more parameters and constants like precision factor or reliability factor and discounting factors, as Smets Philippe mentioned in data fusion in the transferable belief model [78].

This factor will address the limitations of radars and cameras; for example, our output from the camera sensor would not be reliable in occluded and worst weather conditions. Also, radar is limited with DJI Phantom 3 and 4 models and is not trained on hovering states and loaded conditions; we can utilize all these different scenarios and parameters to complement our model.

Using Yager's rule of combination mentioned in eq A-3, we can fuse data from cameras and radar sensors. We are also using a few weighting schemes to complement these sensors in various sensors; more details on the algorithm will be provided in the following sections.

Appendix B: Key Definitions

TCP (Transmission Control Protocol)

TCP (Transmission Control Protocol) is a reliable, connection-oriented protocol within the Internet Protocol Suite that ensures error-checked, ordered data transmission between devices over IP networks.

CNN (Convolutional Neural Network):

A type of deep learning model specifically designed for processing grid-like data, such as images. It employs convolutional layers to automatically learn features from input data and is commonly used in image analysis tasks.

R-CNN (Region Convolutional Neural Network):

A family of object detection models that combine region proposal techniques with CNNs to identify and localize objects in images. It involves generating region proposals followed by fine-tuning with a CNN for object recognition.

True Positive (TP):

The number of correct positive predictions made by a model during testing or evaluation.

False Positive (FP):

The number of incorrect positive predictions made by a model during testing or evaluation.

Localization:

The process of determining the precise position of an object within an image or a 3D space.

Sensor Fusion:

The integration and combination of data from multiple sensors to improve accuracy, robustness, and reliability in object detection and tracking.

Radar (Radio Detection and Ranging):

A technology that uses radio waves to detect the presence, distance, and speed of objects. It is commonly used in object detection, especially in aviation and automotive applications.

UAVs (Unmanned Aerial Vehicles):

Aircraft operated without a human pilot onboard, commonly known as drones. UAVs are used for various purposes, including surveillance and remote sensing.

Precision (P):

The ratio of true positive predictions to the total number of predicted positives (true positives plus false positives). Units: Ratio (0 to 1).

Recall (R):

The ratio of true positive predictions to the total number of actual positives (true positives plus false negatives). Units: Ratio (0 to 1).

Loss:

A mathematical measure that quantifies the difference between predicted and actual values during model training. It guides the model to minimize errors.

Cross-Entropy:

A loss function often used in classification tasks to measure the dissimilarity between predicted probabilities and actual class labels.

Occlusion:

A situation where an object is partially or completely hidden from view by another object in the scene.

Feature Extraction:

The process of transforming raw input data into a reduced set of meaningful features, often used to highlight important information for further analysis.

Feature Map:

Output maps produced by convolutional layers in a neural network, containing abstract features extracted from the input data.

IoU (Intersection over Union):

A metric used to evaluate the accuracy of object localization. It measures the overlap between the predicted bounding box and the ground truth box.

Activation Function:

A non-linear function applied to the output of a neuron in a neural network. It introduces non-linearity and is crucial for learning complex patterns.

Deep Learning:

A subset of machine learning that employs deep neural networks to automatically learn and represent patterns and features in data.

Interacting multiple models (IMM):

Interacting multiple models (IMM) is a statistical estimation framework used in sensor fusion and control systems, where multiple models or filters are employed to capture different behaviors of a dynamic system, and their outputs are combined or switched based on the current state of the system to improve overall estimation accuracy and robustness.

Image Augmentation:

A technique used to artificially expand training data by applying transformations like rotation, flipping, and cropping to images.

Backpropagation:

An algorithm used to calculate the gradient of the loss function with respect to the weights of a neural network, enabling gradient descent optimization.

Kalman Filter:

An algorithm used for estimating the state of a dynamic system from a series of noisy measurements, widely used in tracking applications.

Fusion Center:

The central processing unit that combines data from various sensors in a sensor fusion system.

Anchor Boxes:

Predefined bounding boxes of different aspect ratios and sizes used in object detection algorithms.

Non-Maximum Suppression (NMS):

An algorithm used to eliminate redundant bounding box predictions and keep only the most relevant ones.

Inference:

The process of using a trained model to make predictions on new, unseen data.

Regression:

A type of machine learning task that predicts continuous values (e.g., object coordinates) rather than discrete categories.

Binary Classification:

A classification task with two possible outcomes (e.g., object or background).

Multiclass Classification:

A classification task where the goal is to assign an input to one of several possible classes (e.g., different types of objects).

Data Augmentation:

Techniques used to artificially increase the diversity of training data by applying transformations like rotation, scaling, and cropping.

Bounding Box:

A rectangle that encloses an object in an image, typically defined by its top-left and bottom-right coordinates.

Activation Map:

The output of a layer in a neural network that represents the presence or absence of a certain feature in an input.

YOLO (You Only Look Once):

A real-time object detection system that predicts bounding boxes and class probabilities directly from an image.

SVM (Support Vector Machine):

A machine learning algorithm used for classification and regression tasks, often used in combination with feature extraction techniques.

LiDAR (Light Detection and Ranging):

A remote sensing technology that uses lasers to measure distances, commonly used for 3D mapping and object detection.

Histogram of Oriented Gradients (HOG):

A feature descriptor used for object detection that captures the distribution of gradients in an image.

False Negative (FN):

The number of actual positive instances that were incorrectly predicted as negative by a model.

Target Tracking:

The process of continuously estimating and predicting the trajectory and state of moving objects.

Mean Average Precision (mAP):

An evaluation metric that calculates the average precision across multiple object classes, providing a comprehensive assessment of object detection performance.

Semantic Segmentation:

A computer vision task that involves assigning a class label to each pixel in an image, segmenting it by object category.

Instance Segmentation:

A task that combines object detection and semantic segmentation, assigning a unique label to each instance of an object in an image.

Data Fusion:

The process of integrating data from multiple sources to create a unified representation.

Sensor Redundancy:

The use of multiple sensors of the same type to enhance reliability and robustness in sensor data.

LiDAR-Camera Fusion:

The integration of LiDAR and camera data for improved object detection and scene understanding.

Saliency Map:

A map that highlights the most important regions in an image, often used to guide attention or focus in object detection.

Transfer Learning:

A technique where a pre-trained model is fine-tuned on a new dataset, leveraging knowledge learned from a related task.

GPU (Graphics Processing Unit):

A hardware component designed for parallel processing, commonly used to accelerate deep learning training and inference.

IoT (Internet of Things):

The network of interconnected devices that can collect and exchange data over the internet, often used for real-time monitoring and data collection.

Homography:

A transformation that relates the perspective of one camera view to another, used for rectifying images from different viewpoints.

Active Vision:

A vision system that actively selects viewpoints to optimize the information collected from the environment.

Generative Adversarial Network (GAN):

A type of neural network architecture consisting of a generator and a discriminator, used for generating synthetic data.

Object Occlusion:

The situation where part of an object is hidden or obscured by another object in the scene.

Residual Network (ResNet):

A deep neural network architecture that employs residual connections, allowing for training of very deep networks.

Data Annotation:

The process of labeling data with ground truth information (e.g., bounding box coordinates or object classes) for training machine learning models.

Semantic Understanding:

The ability of a model to comprehend the meaning of objects and their relationships in an image.

Model Ensemble:

A technique where multiple models are trained, and their predictions are combined to improve overall performance.

Batch Normalization:

A technique used to stabilize and accelerate the training of deep neural networks by normalizing the inputs of each layer.

Image Registration:

The process of aligning two or more images taken from different viewpoints or times to create a unified representation.

Semantic Mapping:

The creation of a map that not only represents the physical layout but also includes information about objects and their classes.

Fused Image:

An image generated by combining data from multiple sensors, enhancing the overall quality of information.

Confusion Matrix:

A table used to describe the performance of a classification model by comparing predicted labels with true labels.

Inference Time:

The time it takes for a model to process an input and generate an output during real-time predictions.

Gating Mechanism:

A component in a neural network that controls the flow of information from multiple sources or branches.

Autoencoder:

A neural network architecture used for unsupervised learning, where the model learns to encode and then decode input data.

Dropout:

A regularization technique used during training, where random neurons are ignored to prevent overfitting.

Object Tracking:

The process of following the movement of an object across consecutive frames in a video or sequence of images.

Scale Invariant Feature Transform (SIFT):

A feature extraction algorithm used to detect and describe local features in images, often used in object recognition.

Descriptor Matching:

A process where feature descriptors of keypoints in different images are compared to find correspondences.

LiDAR Segmentation:

The process of categorizing points in a LiDAR point cloud into meaningful object classes.

Adaptive Thresholding:

A technique used to binarize images by determining a threshold value based on local pixel intensities.

Instance Mask:

A binary mask that specifies the precise boundary of an object instance in an image.

Vanishing Point:

The point at which parallel lines in an image appear to converge, often used in scene analysis and navigation.

Recurrent Neural Network (RNN):

A type of neural network that's well-suited for processing sequences of data, often used in tasks like natural language processing.

Fine-Tuning:

The process of adjusting the weights of a pre-trained model on a new dataset to adapt it to a specific task.

Batch Size:

The number of training examples used in a single iteration of gradient descent during model training.

Sigmoid Activation:

A common activation function that maps input values to a range between 0 and 1, often used in binary classification.

Rectified Linear Unit (ReLU):

An activation function that outputs the input if it is positive and zero otherwise, used to introduce non-linearity.

Overfitting:

A situation where a model learns to perform well on the training data but poorly on new, unseen data.

Underfitting:

A situation where a model is too simple to capture the underlying patterns in the data, resulting in poor performance.

Regularization:

Techniques used to prevent overfitting by adding penalties to the model's loss function for complex weights.

Mean Pooling:

A technique used in neural networks to aggregate features by taking the average value of a set of values.

Max Pooling:

A technique used in neural networks to aggregate features by selecting the maximum value from a set of values.

Global Average Pooling:

A pooling technique where the average of all values in a feature map is taken, resulting in a global feature representation.

Deep SORT (Deep Simple Online and Realtime Tracking):

An extension of the SORT (Simple Online and Realtime Tracking) algorithm that combines object detection with deep learning techniques for more accurate and robust multi-object tracking.

Multi-Object Tracking:

The process of following multiple objects as they move through a scene across different frames of a video or sequence of images.

Track Association:

The process of assigning detected objects to existing tracks based on similarity or proximity, a crucial step in multi-object tracking.

Kalman Filtering for Tracking:

The application of the Kalman filter algorithm to estimate the state of objects in motion, commonly used in multi-object tracking.

Extended Kalman Filter (EKF):

A variation of the Kalman filter that can handle non-linear models by linearizing them around the current state estimate.

Particle Filter:

A probabilistic filtering method that represents the belief of an object's state with a set of particles, used in non-linear and non-Gaussian tracking scenarios.

Motion Model:

A mathematical representation of an object's movement behavior over time, used in tracking algorithms to predict object positions.

Measurement Model:

A mathematical representation of how sensor measurements relate to an object's state, used to update object estimates in tracking.

State Vector:

A vector that represents an object's dynamic and kinematic properties, such as position, velocity, and acceleration.

Appearance Model:

A representation of an object’s visual characteristics often learned using deep learning techniques to improve tracking accuracy.

Sensor Fusion for Tracking:

The integration of data from multiple sensors (such as LiDAR and cameras) to enhance the accuracy and robustness of object tracking.

Fused Track:

A track obtained by combining information from multiple sensors, resulting in a more reliable and comprehensive representation of an object's trajectory.

Temporal Fusion for Tracking:

The fusion of tracking information across multiple time frames to improve tracking accuracy and continuity.

Sensor Calibration:

The process of determining the relative geometric and temporal alignment between different sensors, crucial for accurate sensor fusion.

Feature-Level Fusion:

The combination of features or characteristics extracted from different sensors to create a more informative and discriminative representation.

Decision-Level Fusion:

The combination of decisions or predictions from multiple sensors or models to make more informed and reliable decisions.

Track Integrity:

The quality of a track's representation, indicating how well it accurately reflects the object's true trajectory.

Multi-Hypothesis Tracking:

A tracking approach that maintains multiple hypotheses for the state of an object, useful for handling uncertainty and occlusions.

Multi-Sensor Data Association:

The process of associating measurements from multiple sensors with corresponding tracks, considering sensor-specific characteristics and errors.

Information Fusion:

The process of combining data and information from multiple sources to enhance decision-making or understanding of a situation.

Registration for Sensor Fusion:

The alignment of data from different sensors, such as LiDAR and cameras, to create a unified representation of the environment.

Geometric Transformation:

A transformation that relates the spatial coordinates of one sensor to those of another, necessary for accurate sensor fusion.

Temporal Synchronization:

Aligning data from different sensors in time to ensure that measurements correspond to the same moment.

Sensor Redundancy for Robustness:

The use of multiple sensors of the same type to enhance tracking robustness, even in cases where some sensors may fail or provide inaccurate data.

Heterogeneous Sensor Fusion:

The integration of data from sensors of different types (e.g., vision, LiDAR, radar) to leverage their complementary strengths for tracking.

Sensor Registration Error:

The discrepancy between the aligned data from different sensors due to inaccuracies in calibration or alignment.

Geometric Consistency:

Ensuring that the geometric relationships between objects in the fused data are consistent across different sensor views.

Cross-Sensor Calibration:

Calibrating sensors that provide different modalities (e.g., depth and visual data) to ensure their measurements are aligned.

Sequential Sensor Fusion:

The fusion of data from multiple sensors over time to create a coherent and continuous representation of object movement.

Multi-Modal Tracking:

Tracking objects using data from different types of sensors, such as cameras and LiDAR, to capture different aspects of the object's behavior.

Feature Engineering:

The process of selecting, transforming, or creating relevant features from raw data to improve the performance of machine learning models.

Supervised Learning:

A type of machine learning where the model is trained on labeled data, learning the relationship between inputs and corresponding outputs.

Unsupervised Learning:

A type of machine learning where the model is trained on unlabeled data to identify patterns, structures, or clusters in the data.

Semi-Supervised Learning:

A learning approach that combines both labeled and unlabeled data to train a model, often useful when obtaining fully labeled data is challenging.

Cross-Validation:

A technique for estimating the performance of a machine learning model by partitioning the data into subsets for training and testing.

Bias-Variance Trade-Off:

The balance between a model's ability to fit the training data (low bias) and its generalization to new data (low variance).

Hyperparameter Tuning:

The process of selecting optimal hyperparameters for a machine learning model to achieve the best performance.

Feature Selection:

The process of choosing a subset of relevant features from the original set of features to improve model efficiency and generalization.

Regularization:

Techniques (e.g., L1 and L2 regularization) used to prevent overfitting by adding penalties to the model's loss function for complex weights.

Decision Tree:

A hierarchical tree-like structure used for classification and regression tasks, where each internal node represents a decision rule based on a feature, leading to leaf nodes representing class labels or predictions.

Random Forest:

An ensemble learning method that constructs multiple decision trees during training and combines their predictions for improved accuracy.

Gradient Boosting:

An ensemble learning technique that builds a model in a stage-wise manner, with each new model correcting the errors of the previous one.

Principal Component Analysis (PCA):

A dimensionality reduction technique used to transform high-dimensional data into a lower-dimensional representation while preserving its variance.

Naive Bayes:

A probabilistic classifier based on Bayes' theorem, assuming that features are independent given the class.

Computer vision:

Computer vision is a field of artificial intelligence that enables computers to interpret and understand visual information from the world, including images and videos, to make informed decisions or perform tasks.

SoftMax:

SoftMax is a mathematical function used in machine learning to convert numerical values into probabilities, often applied for multi-class classification tasks. It normalizes the input values to produce a probability distribution.

Ensemble Learning:

The technique of combining multiple models (ensemble) to improve overall predictive performance and robustness.

Bias in Machine Learning:

Systematic errors or assumptions that a model makes due to the data it is trained on, which can result in incorrect predictions.

Variance in Machine Learning:

The sensitivity of a model's predictions to small fluctuations in the training data, leading to overfitting.

Area Under the ROC Curve (AUC-ROC):

An evaluation metric that measures the performance of a classification model across different thresholds.

Receiver Operating Characteristic (ROC) Curve:

A graphical representation of a classification model's performance, showing the trade-off between true positive rate and false positive rate.

Mean Squared Error (MSE):

A common loss function used in regression tasks that measures the average squared difference between predicted and actual values.

Confusion Matrix:

A table used to describe the performance of a classification model by comparing predicted labels with true labels.

Down sampling:

Down sampling refers to a process that reduces the spatial dimensions (e.g., width and height) of feature maps within a neural network layer, often using operations like max-pooling or strided convolutions. This helps extract important features while reducing computational complexity.

Convolution:

Convolution, in deep learning, is a fundamental operation for feature extraction, applying a filter to input data to detect patterns like edges in images. It is a key component of convolutional neural networks (CNNs).

Strided convolutions:

Strided convolutions in deep learning refer to convolutional operations where the filter (kernel) moves across input data with a specified step size (stride), resulting in a reduced output size. They are used to control the spatial dimensions and computational complexity in convolutional layers while extracting features.

Bias Correction:

Techniques used to mitigate biases in machine learning models and ensure fair predictions across different groups.

Learning Rate:

A hyperparameter that determines the step size taken during gradient descent optimization in training a model.

Imbalanced Data:

A situation where one class in a classification problem has significantly fewer instances than the others, requiring special handling to prevent bias.

Data Preprocessing:

The process of cleaning, transforming, and preparing raw data to make it suitable for machine learning algorithms.

Kernel Trick:

A technique used in algorithms like SVM to implicitly map input data into a higher-dimensional space for better separation.

Feature Scaling:

The normalization or standardization of features to ensure that all features contribute equally to model training.

ROC-AUC Score:

A single metric that represents the area under the ROC curve, summarizing a classifier's ability to discriminate between classes.

F1 Score:

A metric that balances both precision and recall, useful for evaluating model performance on imbalanced datasets.

Bias-Variance Decomposition:

A technique used to analyze the expected error of a model by decomposing it into bias and variance components.

Batch Gradient Descent:

An optimization technique used to update model parameters by computing gradients based on the entire training dataset.

Stochastic Gradient Descent (SGD):

An optimization technique that updates model parameters using gradients computed on a small batch of training data.

Mini-Batch Gradient Descent:

A compromise between batch gradient descent and SGD, where updates are based on small batches of training data.

synchronization:

The process of aligning and coordinating data or processes from different sources or components to ensure accurate and consistent timing.

Timestamp:

A time reference assigned to data or events, used for synchronization and chronological ordering of information from different sensors or systems.

Temporal Alignment:

The adjustment of timestamps to ensure that data from multiple sensors or components correspond to the same time reference.

Interpolation:

The estimation of values at intermediate points based on known data points, used to align data sampled at different rates.

Latency:

The time delay between the occurrence of an event and its detection or processing, affecting the synchronization of real-time systems.

Data Fusion Synchronization:

The alignment of data streams from different sensors or sources to create a unified representation with consistent timing.

Intersection over Union:

IOU (Intersection over Union), also known as Jaccard Index, is a metric used in computer vision and object detection to measure the accuracy of the overlap between two bounding boxes or regions of interest. The unit of IOU is dimensionless because it represents a ratio of areas, and the result is typically expressed as a value between 0 (no overlap) and 1 (perfect overlap).

Real-Time Processing:

The execution of tasks and computations with timing constraints that require responses within specific time limits.

Data Timestamping:

Assigning a timestamp to each data sample or event to facilitate synchronization and chronological ordering.

Time Synchronization Protocol:

A standardized set of rules and procedures used to ensure consistent time across distributed systems or devices.

Network Time Protocol (NTP):

A protocol used to synchronize clocks of devices over a network, maintaining accurate time across distributed systems.

Precision Time Protocol (PTP):

A protocol designed to achieve sub-microsecond synchronization accuracy in local area networks, critical for applications requiring high precision.

Deadlock:

A situation in which multiple processes or components are waiting for resources that are held by others, resulting in a standstill.

Frame Alignment:

The synchronization of data frames or packets from different sources to align with a common time reference.

Real-Time Clock (RTC):

A clock module designed to keep track of time even in the absence of power, used for accurate timestamping in embedded systems.

Asynchronous Data Acquisition:

The collection of data samples without strict timing coordination, suitable for systems where precise timing is not critical.

Universal Transverse Mercator:

A UTM (Universal Transverse Mercator) zone is one of the 6-degree longitudinal segments into which the world is divided for mapping and coordinate reference purposes. Each zone has a unique central meridian and is used to provide accurate, flat-surface representations of the Earth's surface for specific regions.

Modified National Institute of Standards and Technology (MNIST) Dataset:

MNIST is a widely used dataset in machine learning and computer vision for handwritten digit recognition. MNIST contains a large collection of 28x28 pixel grayscale images of handwritten digits (0 to 9) along with their corresponding labels, making it a common benchmark for developing and testing image classification algorithms.

Subdivisions: Divisions of a batch used for memory efficiency during training.

Width:

The horizontal dimension of an image or layer.

Height:

The vertical dimension of an image or layer.

Channels:

The individual color or feature channels in an image.

Momentum:

A parameter in optimization algorithms that influences the update direction.

Decay:

A reduction factor applied to learning rates to control training stability.

Angle:

An orientation parameter often used in image transformations.

Saturation:

A parameter affecting color intensity in image processing.

Exposure:

A parameter controlling image brightness.

Hue:

A parameter controlling image color tint.

Learning Rate:

A rate at which a model adjusts its parameters during training.

Burn In:

An initial phase of training with a lower learning rate.

Max Batches:

The maximum number of batches used in training.

Policy:

A strategy for adjusting hyperparameters during training.

Steps:

Points in training where hyperparameters are adjusted.

Scales:

Scaling factors used for hyperparameter adjustments.

Keras-Retinanet:

Keras-Retinanet is an open-source deep learning framework that implements the RetinaNet architecture for object detection tasks, known for its efficiency and accuracy, particularly in detecting objects at multiple scales within images. It provides a user-friendly interface for training and deploying object detection models.

References

[1] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, 2001, vol. 1.

[2] P. Viola and M. J. Jones, “Robust real-time face detection,” *Int J Comput Vis*, vol. 57, no. 2, pp. 137–154, 2004.

[3] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05)*, 2005, vol. 1, pp. 886–893.

[4] P. Felzenszwalb, D. McAllester, and D. Ramanan, “A discriminatively trained, multiscale, deformable part model,” in *2008 IEEE conference on computer vision and pattern recognition*, 2008, pp. 1–8.

[5] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester, “Cascade object detection with deformable part models,” in *2010 IEEE Computer society conference on computer vision and pattern recognition*, 2010, pp. 2241–2248.

[6] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *IEEE Trans Pattern Anal Mach Intell*, vol. 32, no. 9, pp. 1627–1645, 2010.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[8] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” *IEEE Trans Pattern Anal Mach Intell*, vol. 37, no. 9, pp. 1904–1916, 2015.

[9] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Adv Neural Inf Process Syst*, vol. 28, 2015.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[12] W. Liu *et al.*, "Ssd: Single shot multibox detector," in *European conference on computer vision*, 2016, pp. 21–37.

[13] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[14] J. Sanfridsson *et al.*, "Drone delivery of an automated external defibrillator–a mixed method simulation study of bystander experience," *Scand J Trauma Resusc Emerg Med*, vol. 27, no. 1, pp. 1–9, 2019.

[15] B. Taha and A. Shoufan, "Machine learning-based drone detection and classification: State-of-the-art in research," *IEEE access*, vol. 7, pp. 138669–138682, 2019.

[16] G. Ning, G. Chen, C. Tan, S. Luo, L. Bo, and H. Huang, "Data augmentation for object detection via differentiable neural rendering," *arXiv preprint arXiv:2103.02852*, 2021.

[17] P. Saxena, "Data Augmentation for Custom Object Detection," Aug. 30, 2020. https://medium.com/predict/data-augmentation-for-custom-object-detection-15674966e0c8 (accessed Oct. 19, 2022).

[18] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*, 2017, pp. 3645–3649.

[19] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. v Le, "Learning data augmentation strategies for object detection," in *European conference on computer vision*, 2020, pp. 566–583.

[20] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, “Interacting multiple model methods in target tracking: a survey,” *IEEE Trans Aerosp Electron Syst*, vol. 34, no. 1, pp. 103–123, 1998.

[21] Y. Zheng, Z. Chen, D. Lv, Z. Li, Z. Lan, and S. Zhao, “Air-to-air visual detection of micro-uavs: An experimental evaluation of deep learning,” *IEEE Robot Autom Lett*, vol. 6, no. 2, pp. 1020–1027, 2021.

[22] Y. Zheng, Z. Chen, D. Lv, Z. Li, Z. Lan, and S. Zhao, “Air-to-air visual detection of micro-uavs: An experimental evaluation of deep learning,” *IEEE Robot Autom Lett*, vol. 6, no. 2, pp. 1020–1027, 2021.

[23] V. Walter, M. Vrba, and M. Saska, “On training datasets for machine learning-based visual relative localization of micro-scale UAVs,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10674–10680.

[24] F. Svanström, C. Englund, and F. Alonso-Fernandez, “Real-time drone detection and tracking with visible, thermal and acoustic sensors,” in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 7265–7272.

[25] A. Coluccia *et al.*, “Drone vs. bird detection: Deep learning algorithms and results from a grand challenge,” *Sensors*, vol. 21, no. 8, p. 2824, 2021.

[26] M. Pawełczyk and M. Wojtyra, “Real world object detection dataset for quadcopter unmanned aerial vehicle detection,” *IEEE Access*, vol. 8, pp. 174394–174409, 2020.

[27] R. Padilla, S. L. Netto, and E. A. B. da Silva, “A survey on performance metrics for object-detection algorithms,” in *2020 International conference on systems, signals and image processing (IWSSIP)*, 2020, pp. 237–242.

[28] J. Redmon and A. Farhadi, “YOLOv3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[29] Codalab, “COCO Detection Challenge (Bounding Box).” https://competitions.codalab.org/competitions/20794 (accessed Oct. 20, 2022).

[30] K. Oksuz, B. C. Cam, E. Akbas, and S. Kalkan, “Localization recall precision (LRP): A new performance metric for object detection,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 504–519.

[31] J. Xing, H. Ai, L. Liu, and S. Lao, “Multiple player tracking in sports video: A dual-mode two-way bayesian inference approach with progressive observation modeling,” *IEEE Transactions on Image Processing*, vol. 20, no. 6, pp. 1652–1667, 2010.

[32] L. Khachatryan, “Centroid-Based_Object_Tracking.” https://github.com/lev1khachatryan/Centroid-Based_Object_Tracking (accessed Oct. 20, 2022).

[33] Labbe, R. “How a Kalman filter works, in pictures. Kalman and Bayesian Filters in Python blog”. Retrieved from https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/, 2014

[34] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric,” in *2017 IEEE international conference on image processing (ICIP)*, 2017, pp. 3645–3649.

[35] D. S. Kaputa and B. P. Landy, “YOLBO: You Only Look Back Once–A Low Latency Object Tracker Based on YOLO and Optical Flow,” *IEEE Access*, vol. 9, pp. 82497–82507, 2021.

[36] F. Gökçe, G. Üçoluk, E. Şahin, and S. Kalkan, “Vision-based detection and distance estimation of micro unmanned aerial vehicles,” *Sensors*, vol. 15, no. 9, pp. 23805–23846, 2015.

[37] K. R. Sapkota *et al.*, “Vision-based unmanned aerial vehicle detection and tracking for sense and avoid systems,” in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 1556–1561.

[38] J. Li, D. H. Ye, T. Chung, M. Kolsch, J. Wachs, and C. Bouman, “Multi-target detection and tracking from a single camera in Unmanned Aerial Vehicles (UAVs),” in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2016, pp. 4992–4997.

[39] S. Minaeian, J. Liu, and Y.-J. Son, "Effective and efficient detection of moving targets from a UAV's camera," *IEEE transactions on intelligent transportation systems*, vol. 19, no. 2, pp. 497–506, 2018.

[40] J.-Y. Bouguet, "Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm," *Intel corporation*, vol. 5, no. 1–10, p. 4, 2001.

[41] R. Opromolla, G. Fasano, and D. Accardo, "A vision-based approach to UAV detection and tracking in cooperative applications," *Sensors*, vol. 18, no. 10, p. 3391, 2018.

[42] Kumar, Ashish. "Artificial Intelligence In Object Detection - Report.", 2020.

[43] N. Al-Qubaydhi *et al.*, "Detection of Unauthorized Unmanned Aerial Vehicles Using YOLOv5 and Transfer Learning," *Electronics (Basel)*, vol. 11, no. 17, p. 2669, 2022.

[44] H.-K. Jung and G.-S. Choi, "Improved YOLOv5: Efficient Object Detection Using Drone Images under Various Conditions," *Applied Sciences*, vol. 12, no. 14, p. 7255, 2022.

[45] W. Zhan *et al.*, "An improved Yolov5 real-time detection method for small objects captured by UAV," *Soft comput*, vol. 26, no. 1, pp. 361–373, 2022.

[46] B. Liu and H. Luo, "An Improved Yolov5 for Multi-Rotor UAV Detection," *Electronics (Basel)*, vol. 11, no. 15, p. 2330, 2022.

[47] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 734–750.

[48] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[49] I. Guvenc, F. Koohifar, S. Singh, M. L. Sichitiu, and D. Matolak, "Detection, tracking, and interdiction for amateur drones," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 75–81, 2018.

[50] M. Jahangir and C. J. Baker, "Robust Detection of Micro-UAS Drones with L-Band 3-D Holographic Radar," *2016 Sensor Signal Processing for Defence (SSPD)*, pp. 1–5, 2016.

[51] F. Fioranelli, M. Ritchie, H. Griffiths, and H. Borrion, "Classification of loaded/unloaded micro-drones using multistatic radar," *Electron Lett*, vol. 51, no. 22, pp. 1813–1815, 2015.

[52] N. Mohajerin, J. Histon, R. Dizaji, and S. L. Waslander, “Feature extraction and radar track classification for detecting UAVs in civillian airspace,” in *2014 IEEE Radar Conference*, 2014, pp. 674–679.

[53] P. Zhang, L. Yang, G. Chen, and G. Li, “Classification of drones based on micro-Doppler signatures with dual-band radar sensors,” in *2017 Progress in Electromagnetics Research Symposium-Fall (PIERS-FALL)*, 2017, pp. 638–643.

[54] B. K. Kim, H.-S. Kang, and S.-O. Park, “Drone classification using convolutional neural networks with merged Doppler images,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 1, pp. 38–42, 2016.

[55] D. A. Brooks, O. Schwander, F. Barbaresco, J.-Y. Schneider, and M. Cord, “Temporal deep learning for drone micro-Doppler classification,” in *2018 19th International Radar Symposium (IRS)*, 2018, pp. 1–10.

[56] B. Torvik, K. E. Olsen, and H. Griffiths, “Classification of birds and UAVs based on radar polarimetry,” *IEEE geoscience and remote sensing letters*, vol. 13, no. 9, pp. 1305–1309, 2016.

[57] A. Rozantsev, V. Lepetit, and P. Fua, “Detecting flying objects using a single moving camera,” *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 5, pp. 879–892, 2016.

[58] E. Unlu, E. Zenou, and N. Riviere, “Using shape descriptors for UAV detection,” *Electronic Imaging*, vol. 2018, no. 9, pp. 121–128, 2018.

[59] S. K. Boddhu, M. McCartney, O. Ceccopieri, and R. L. Williams, “A collaborative smartphone sensing platform for detecting and tracking hostile drones,” in *Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IV*, 2013, vol. 8742, pp. 293–303.

[60] D. Lee, W. G. La, and H. Kim, “Drone detection and identification system using artificial intelligence,” in *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 2018, pp. 1131–1133.

[61] J. Peng, C. Zheng, P. Lv, T. Cui, Y. Cheng, and S. Lingyu, “Using images rendered by PBRT to train faster R-CNN for UAV detection,” 2018.

[62] M. Saqib, S. D. Khan, N. Sharma, and M. Blumenstein, “A study on detecting drones using deep convolutional neural networks,” in *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*, 2017, pp. 1–5.

[63] C. Aker and S. Kalkan, “Using deep networks for drone detection,” in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6.

[64] R. Yoshihashi, T. T. Trinh, R. Kawakami, S. You, M. Iida, and T. Naemura, “Differentiating objects by motion: Joint detection and tracking of small flying objects,” *arXiv preprint arXiv:1709.04666*, 2017.

[65] J. Park, D. H. Kim, Y. S. Shin, and S. Lee, “A comparison of convolutional object detectors for real-time drone tracking using a PTZ camera,” in *2017 17th International Conference on Control, Automation and Systems (ICCAS)*, 2017, pp. 696–699.

[66] C. Aker and S. Kalkan, “Using deep networks for drone detection,” in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6.

[67] M. Saqib, S. D. Khan, N. Sharma, and M. Blumenstein, “A study on detecting drones using deep convolutional neural networks,” in *2017 14th IEEE international conference on advanced video and signal based surveillance (AVSS)*, 2017, pp. 1–5.

[68] M. Wu, W. Xie, X. Shi, P. Shao, and Z. Shi, “Real-time drone detection using deep learning approach,” in *International Conference on Machine Learning and Intelligent Communications*, 2018, pp. 22–32.

[69] E. Unlu, E. Zenou, N. Riviere, and P.-E. Dupouy, “Deep learning-based strategies for the detection and tracking of drones using several cameras,” *IPSJ Transactions on Computer Vision and Applications*, vol. 11, no. 1, pp. 1–13, 2019.

[70] Q. Baig, “Multi sensor data fusion for detection and tracking of moving objects from a dynamic autonomous vehicle,” *HAL*, vol. 2012, 2012.

[71] Q. Baig, O. Aycard, T. D. Vu, and T. Fraichard, “Fusion between laser and stereo vision data for moving objects tracking in intersection like scenario,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 362–367.

[72] S. Chadwick, W. Maddern, and P. Newman, “Distant vehicle detection using radar and vision,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8311–8317.

[73] Z. Zhong, S. Liu, M. Mathew, and A. Dubey, “Camera radar fusion for increased reliability in ADAS applications,” *Electronic Imaging*, vol. 2018, no. 17, pp. 251–258, 2018.

[74] V. John and S. Mita, “RVNet: Deep sensor fusion of monocular camera and radar for image-based obstacle detection in challenging environments,” in *Pacific-Rim Symposium on Image and Video Technology*, 2019, pp. 351–364.

[75] F. Svanström, “Drone detection and classification using machine learning and sensor fusion.” *University: Halmstad University Sweden, Independent thesis Advanced level (degree of Master,* 2020 Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:hh:diva-42141.

[76] R. R. Yager, “On the relationship of methods of aggregating evidence in expert systems,” *Cybernetics and System*, vol. 16, no. 1, pp. 1–21, 1985.

[77] L. Liu and R. R. Yager, “Classic works of the Dempster-Shafer theory of belief functions: An introduction,” in *Classic works of the Dempster-Shafer theory of belief functions*, Springer, 2008, pp. 1–34.

[78] P. Smets, “Data fusion in the transferable belief model,” in *Proceedings of the third international conference on information fusion*, 2000, vol. 1, pp. PS21–PS33.

[79] A. P. Dempster, “A generalization of Bayesian inference,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 30, no. 2, pp. 205–232, 1968.

[80] “LabelImg”, Accessed: Oct. 27, 2022. [Online]. Available: https://github.com/heartexlabs/labelImg

[81] “IMGBIN”, Accessed: Oct. 27, 2022. [Online]. Available: https://imgbin.com/png/DZju8VmF/deep-learning-machine-learning-artificial-neural-network-brain-artificial-intelligence-png

[82] Ayoosh Kathuria, “What’s new in YOLO v3?,” Apr. 23, 2018. https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b#:~:text=YOLO%20is%20a%20fully%20convolutional,different%20places%20in%20the%20network. (accessed Oct. 27, 2022).

[83] Chavez-Garcia, R.O. (2014). Multiple sensor fusion for detection, classification and tracking of moving objects in driving environments. 14th International Conference on Control, Automation and Systems (ICCAS 2014), 1683-1688.

[84] R. Gandhi, “R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms.” https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e (accessed Nov. 11, 2022).

[85] A. Coluccia et al., “Drone-vs-Bird detection challenge at IEEE AVSS2017,” IEEE AVSS 2017.