Department of Computer Science

BSc (Hons) Computer Science (Artificial Intelligence)

Academic Year 2021 - 2022

Airborne Object Sense & Avoid Using Machine Learning

Pratikgiri Goswami

1945204

A report submitted in partial fulfilment of the requirements for the degree of Bachelor of Science

Brunel University London
Department of Computer Science
Uxbridge
Middlesex
UB8 3PH
United Kingdom
T: +44 1895 203397
F: +44 (0) 1895 251686

Abstract

Over the past few years, Unmanned Aerial Vehicles (UAVs), like drones, have risen in popularity. Drones have become vital to the functions of many businesses, with the market expected to grow to $64 billion by 2025. Companies like Amazon take advantage of this opportunity by swiftly incorporating autonomous drone flights to deliver packages. While the route an autonomous drone takes can be carefully planned ahead of its mission, there remains a chance that the drone will encounter unforeseen airborne objects. Therefore, it has become vital that drones develop sense and avoidance capabilities to help evade these obstacles. A safety-critical system like this one can help reduce crashes and improve public safety. In this project, I will design and develop a web application that intends to showcase the model's accuracy and performance. The solution aims to develop convolutional neural networks that can analyse images and detect airborne objects. The detected object will then be classified and tracked to predict a manoeuvre that creates a safer route. Existing solutions have been able to perform object detection using expensive sensors: ultrasonic, lasers, infrared, increasing the overall drone's weight and hurting its battery life. Solving the problem by re-using the drone's onboard cameras is an attractive solution. A model was designed, developed and trained on the Airborne Object Tracking (AOT) dataset. The project has been rigorously statistically evaluated on the model's accuracy and performance. Lastly, the model got integrated into a web application and tested on its functionality. In conclusion, it performed sense & avoidance on most airborne objects classes; however, the model's inference time can be improved to offer real-time response.

Acknowledgements

I want to thank my project supervisor, Dr Mahir Arzoky, for his support throughout the development of this project, encouraging me when faced with difficult bugs and offering invaluable advice.

I certify that the work presented in the dissertation is my own unless referenced.

Signature	Mr Pratikgiri Goswami
Date	31/03/2022

Total Words: 12171

Table of contents

List of Tables

List of Figures

1 Introduction

Across the globe, demand for cheap, reliable, and swift transportation is in desperate need. Commercial, agricultural, and military sectors (Fly Ability, 2020) have extensively used unmanned aerial vehicles (UAV), i.e., drones (Bray Solutions, 2017). As demand for drones skyrockets, sales surpassed $1.25 billion as of 2020 and are estimated to grow to $63.6 billion by 2025. The drone market is estimated to sell 2.4 million units in 2023 –increasing by 66.8%. Agricultural sector consumption is estimated to rise by over 69% from 2010 to 2050 (Intelligence Insider, 2021).

It has become crucial for drones to detect airborne objects and avoid them. The project aims to build an airborne object detection and tracking application, enabling any UAV equipped with a camera to sense and avoid an airborne object. The application takes input images and performs image analysis to detect any airborne objects present in the frame. The object detected would then be classified and tracked. Finally, the model returns a necessary manoeuvre to create a safer route. This sort of safety-critical system would help reduce crashes and improve public safety. The project required research on the existing models, algorithms, and evaluation techniques for the model within a virtual environment (i.e., Docker Containers).

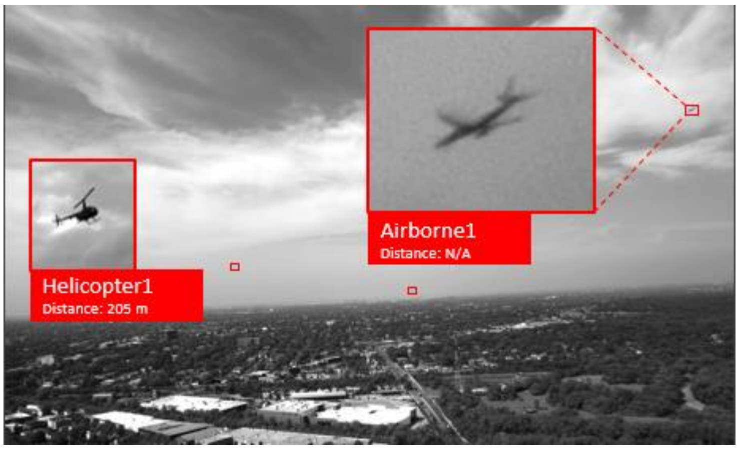


Figure 1.1 - Example of frame analysis - expected results (AIcrowd, 2021)

1.1 Aims and Objectives

The project aims to create a machine learning model that can infer images to detect airborne objects, which would then be classified and tracked to predict a manoeuvre that avoids the obstacles. The project's goals would be achieved by completing the following objectives:

1. Conduct research to find literature on existing drone sense & avoidance systems and machine learning algorithms to build the basis for this project.
2. Research potential methodologies to use and create a plan based on the methodology that, when followed, obtains project goals.

3. Source publicly available datasets on airborne objects and gain approval by Brunel research ethics online (BREO) where necessary.
4. Design, train and implement a machine learning inference model that fulfils the project's aim.
5. Design and create an interface to allow users to interact and test its interface on functionality.
6. Evaluate the model's performance against evaluation metrics (such as detection accuracy, rate of false positives and overall classification accuracy).
7. Evaluate the project on what went well, what could be improved and potential future works.

1.2 Project Approach

This project has been developed through an iterative process, providing a more flexible approach. The project followed the rapid application development (RAD) methodology, where each functionality was sequentially prototyped and implemented in an iterative cycle to improve the final application repetitively. Firstly, a dataset of airborne objects was acquired, and a pipeline was put into operation, providing the images upon which the machine learning model would be trained. The project implemented you only look once (YOLO-v5) object detection network, and the final model would be evaluated against a range of metrics. The final model would-be integrated into a web application for demonstration and testing purposes. The table below illustrates the ordered stages:

Task No.	Task Description
1.	Conduct research to find literature on existing sense & avoidance systems, computer vision algorithms, and pre-trained networks.
2.	Research to find a suitable methodology that fits this project.
3.	Source airborne object dataset and gain ethics approval from BREO where necessary.
4.	Create a project plan and actively monitor the project progress.
5.	Design, train and implement the model
6	Design and create an interface that allows users to interact.
7.	Evaluate the final model against a range of metrics such as the rate of false positives, object classification accuracy and inference time.
8.	Integrate both the backend and frontend and test its functionality.
9.	Create a report to check what went well, what could be improved, future works and check if all aims have been met.

Table 1.1 - Project Approach Stages

1.3 Dissertation Outline

The report is structured as follows:

Chapter 2

It contains a literature review used to develop this project, outlining existing sense and avoidance technologies and an outline of artificial intelligence techniques.

Chapter 3

Describes the rapid application development (RAD) methodology used to develop the project, methodology comparison, and chosen tools.

Chapter 4

Details the development of front and back end designs: prototypes, flowcharts and pseudocode. Additionally, it explores data scrapping and preparation pipelines.

Chapter 5

Showcases the process of implementing the model, frontend and backend.

Chapter 6

It contains a detailed overview of functionality testing on the web application.

Chapter 7

It contains an evaluation of the model's performance.

Chapter 8

Details a final project conclusion, a justification of overall effectiveness and potential future improvements.

2 Literature Review

2.1 Unmanned Aerial Vehicles

There are various unmanned aerial vehicles (UAV) categories, from rotorcrafts to hybrid UAVs, where no human pilot is aboard to operate the vehicle. Drones are one of the most widely used UAVs, and with drone sales rocket spiking, it has become a vital tool to society. The drone service market is estimated to grow to $63.6 billion by 2025, and the latest analytics from Business Insiders predicts that consumer drone shipments will hit 29 million by 2021. The famous Goldman Sachs forecasts – supported by growth in the commercial and government sectors, suggest that the drone market is worth $100 billion (Intelligence Insider, 2021).

Pros	Cons
Drones are cheap.	Lack the ability to sense and avoid obstacles.
Easy to deploy (can be used 24/7) and widely used for transportation.	Risk to the public in the event of a drone collision.
They are used in the agriculture sector for monitoring and collecting data.	Drones are very dangerous near airports/planes and pose a danger to aircraft.
Helpful in a natural disaster and help reduce the risk of a dangerous solution.	Potentially lethal if flown by untrained personnel.

Table 2.1 - Pros & Cons of Drones (Andreas, 2017)

Drones are rapidly changing the agricultural sectors, where the Agras line of drones from DJI can assist farmers in precisely delivering a variety of products such as fertiliser, seeds, pesticides, and desiccants. Modularisation of the drones has also allowed various fittings to be swapped and installed. The P4 Multispectral drone comes equipped with a bundle of sensors that could capture the infrared radiation from the ground paired with the amount of heat to generate Normalised Difference Vegetation Index (NDVI) maps (see Figure 2.1). NDVI maps can indicate malnourishment and droughts. These fundamental tools are essential to improving crop yield (DJI Enterprise, 2021). Amazon has introduced a faster logistic system that utilises drones to deliver "packages to customers in 30 minutes" (Amazon, 2013).

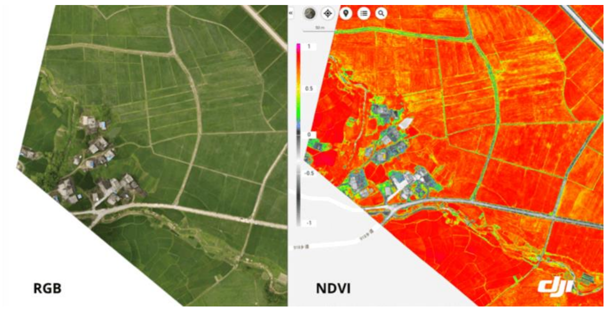


Figure 2.1 - NDVI Map Showing Malnourishment & Drought (DJI Enterprise, 2021)

Figure 2.2 - Amazon Prime Air Delivery Drone (Amazon, 2013)

2.2 Sense & Avoidance

Drones are revolutionary and rapidly integrated into our day-to-day tasks. Companies such as Amazon Prime Air incorporating their fleet of drones to deliver packages quicker through automated flight plans raises a risk to the public. Object avoidance is a critical safety feature that scans the drone's surrounding environment, detects any obstacles along the planned route in real-time, and reacts accordingly to steer clear of them immediately. Drones must develop a sense and avoid capabilities to evade non-static obstacles they might encounter; otherwise, a mid-air collision with other airborne objects such as aircraft or helicopters could be lethal. An avoidance feature is necessary, especially for drones that fly at high altitudes.

The current drone systems employ a range of expensive sensors such as ultrasonic, lasers, stereo vision, infrared and LIDAR. These sensors emit sound or light waves at high frequencies, reflected and detected by the drone's onboard sensors. The time it takes for the waves to reflect is measured, and when combined, it produces a map of the surrounding environment (Ciobanu, 2021). However, the addition of these sensors increases the drone’s weight, leading to an adverse effect on payload size and battery life (Drone Tech Planet, 2019). Solving the problem using a camera is an attractive solution as it re-uses the existing onboard cameras.

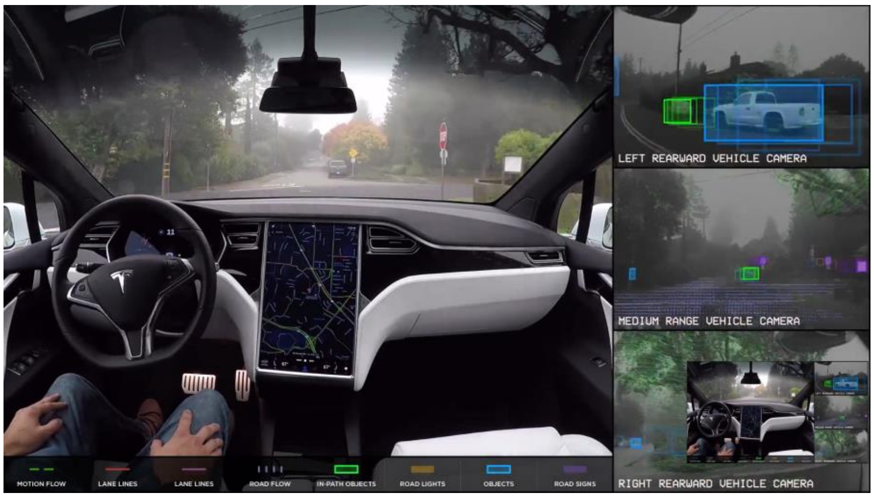

LEFT REARWARD VEHICLE CAMERA
MEDIUM RANGE VEHICLE CAMERA
RIGHT REARWARD VEHICLE CAMERA
MOTION FLOW
LANE LINES
LANE LINES
ROAD FLOW
IN-PATH OBJECTS
ROAD LIGHTS
OBJECTS
ROAD SIGNS

Figure 2.3 - Tesla Autopilot Collison Avoidance System (Tesla, 2016)

Similar sense and avoidance systems have been incorporated into new Tesla cars that use high-resolution cameras with ultrasonic sensors (Tesla, 2016). The car actively and passively monitors its surroundings to predict future collisions and immediately manoeuvres to evade them. Tesla’s vehicles safety reports highlight that “one crash for every 4.31 million miles” was recorded compared to the National Highway Traffic Safety Administration's (NHTSA) most recent data - Quarter 4 of 2021, illustrating “an automobile crash every 484,000 miles”(Tesla, 2018). This technology conveys how crucial and beneficial a safety system like this is to help improve road safety and lower the chances of a vehicle accident.

Further benefits of an object avoidance system are that it helps improve public safety, ensures confident flights, reduces business costs - fewer collisions and finally, allows parallel autonomous flights to occur, exponentially increasing efficiency.

2.3 Artificial Intelligence Background

2.3.1 Deep Neural Networks

Neural networks are a form of machine learning algorithms that are roughly structured like the neurons found in a human brain. Neural networks can solve problems without explicitly being programmed; for example, recently, neural networks have come a long way in computer vision and natural language processing. These networks are extremely good at discovering patterns and extrapolating them, as they can model a range of complex non-linear relationships (Mahanta, 2017). For example, inferring the classification of an object and its bounding box and identifying the relationship given an image with a range of red, green, and blue values requires forming a complex non-linear relationship. This method has many flaws and a variety of mitigations strategies to avoid them (see below).

Flaws	Mitigation Strategies
Hard to interpret most of the times	Use statistical tests and set out metrics to evaluate the model's accuracy and performance.
Require a large amount of data	The project has close to 14TB of data; thus, the training set should be equally proportionate to each category.
Has long training time	Reduce the image resolution from the original image (2448x2048) to 1600px.
Requires expensive hardware to train models.	Use free online platforms such as Google Collab and Kaggle Workspace to train and use small training batches.

Table 2.2 - Flaws and Mitigation Strategies (Bhatia, 2018)

2.3.2 Neural Network Implementation

Each neuron in the neural network takes one or more input signals, either from the raw data set or the neuron positioned at a previous layer. The inputs are multiplied by the neuron's weight and sum these values together (McCullum, 2020). The neuron's weight can be randomly assigned at creation and later improved upon; on the other hand; weights can be imported from a pre-trained model, known as transfer learning; various factors such as architecture and neural networks size need to match with an existing network. The summed values would pass through a synapse; that may have a bias value representing the strength of the connection between two neurons, an activation function such as sigmoid function, rectifier functions (ReLU) or hyperbolic tangent function (McCullum, 2020). These activation functions create more sophisticated neural networks as the inputs and outputs can transition from linear outputs to a more complex non-linear relationship.

As neurons are organised in layers, training is more manageable and straightforward, where the layers share a learning rate and an activation function. The layer's position in a neural network represents its purpose; a top layer is specialised to recognise more complex objects than a lower layer that may only recognise simple lines and changes in gradients. The neurons are trained using a cost function, which varies from network to network. The project uses multiple loss functions to update the neuron's weight values and bias. The value to update is calculated by calculating the error between expected and actual results.

2.3.3 Convolutional Neural Networks (CNN)

In a convolutional neural network, one or many neurons can be replaced with various operators. Firstly, a convolutional layer would reduce the amount of data passing through the network without losing too much detail. This operator allows for faster and smaller networks with low inference time enabling real-time image processing. The convolution layer aims to extract features by performing a dot product between two matrices; the convolutional mask and the image provided, which produces a feature map (see Figure 2.4).

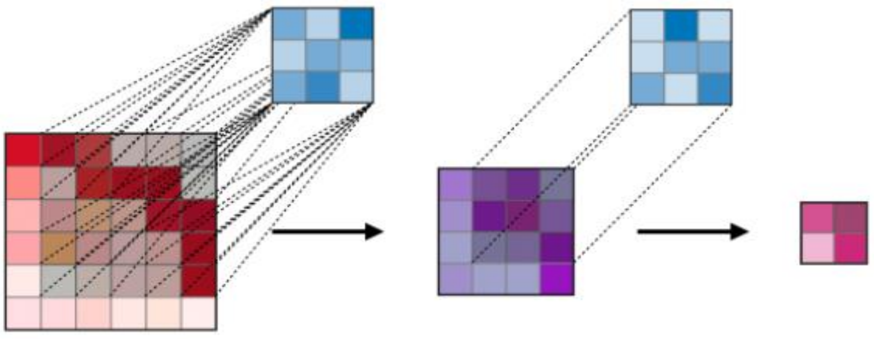


Figure 2.4 - Convolution Mask Applied To An Image (Amidi and Amidi, 2018)

A pooling layer is a downscaling operation (see Figure 2.5), typically applied after a convolutional layer. A max-pooling operator will take the maximum value of the current view to preserve the detected features. Whereas an average pooling would average the values to downscale the feature map.

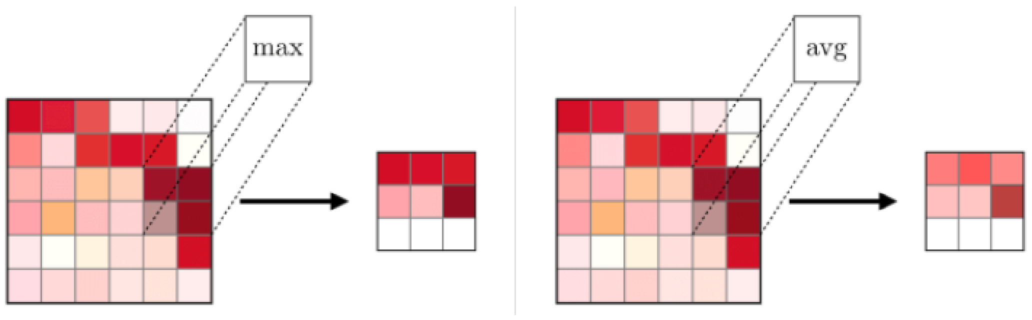


Figure 2.5 - Max & Average Pooling Demo (Amidi and Amidi, 2018)

2.3.4 You Only Look Once (YOLO) Algorithm

You only look once (YOLO) is an advanced real-time object detection algorithm, one of the most effective algorithms, as it "only looks once" at the image, meaning it only requires one forwards propagation pass through the network. A non-max suppression is applied, and then it outputs the recognised objects together with its bounding box and a confidence score. Traditional object detection algorithms such as Fast-CNN and Faster-CNN are composed of a two-stage architecture; firstly, detecting possible object regions and then classifying those regions into an object class (Bandyopadhyay, 2022). YOLO is a one stage end-to-end neural network – reducing project complexity, in which both region proposals and classification steps are combined.

The algorithm trains on the whole image and directly optimises the weights through its inbuilt loss functions. It is swift, beneficial in this project to ensure real-time performance. Furthermore, it "learns a generalisable representation of the object", thus delivering similar accuracy in a real scenario (ODSC - Open Data Science, 2018). One of the limitations of YOLO is that it "struggles to detect objects that appear in groups" (Bandyopadhyay, 2022), as each cell is limited to detecting only a single object.

The YOLO algorithm works by dividing the given image into N grids of cells, where each cell has an equal dimension of region S x S (see Figure 2.6). Each grid cell is responsible for detecting and localising an object that falls in its region. The cells predict the bounding box, class label and confidence score of the object present in that cell. The bounding box prediction contains an (x, y) indicating the centre of the box relative to the cell and a (w, h) representing the width and height values. Furthermore, a class label and its confidence score are predicted for each grid cell; a non-maximal suppression is applied: "we define confidence as Pr(Object) *

IOU(pred, truth)" (Redmon *et al.*, 2015). This further implies that the probability is conditioned only to contain one object per cell. If no object exists, a confidence score of zero returns; otherwise, " we want the confidence score to equal the [IOU] between the predicted box and the ground truth" (Redmon et al., 2015). The loss function comprises four separate loss functions (see Figures 2.7), where each function calculates the error between the predicted and ground truth and then sums these values together.

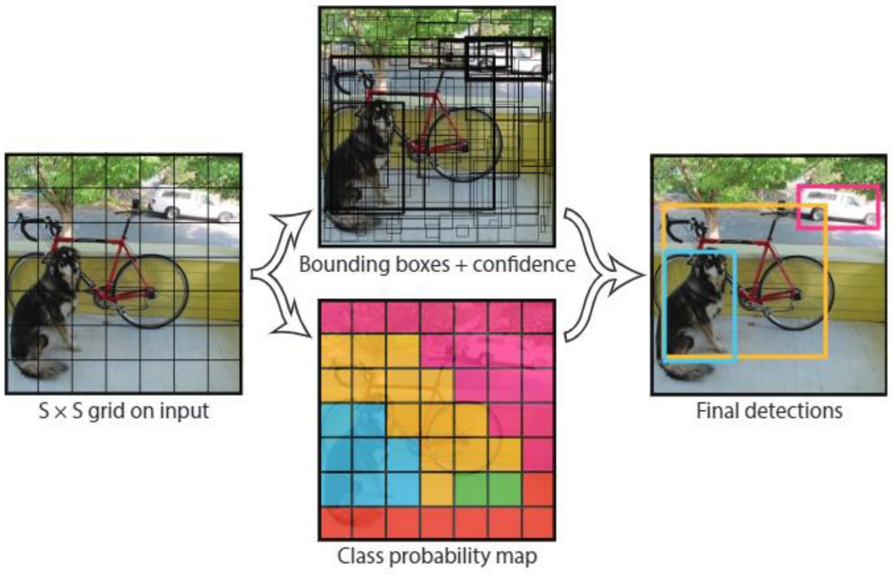


Figure 2.6 – The YOLO Model Explained (Redmon *et al.*, 2015)

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

Loss Function For Predicted Bounding Box

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

Loss Function For Predicted Box Width & Height

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

Loss Function For Predicted Confidence Score

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

Loss Function For Predicted Classification

Figure 2.7 - YOLO Loss Functions (Redmon *et al.*, 2015)

2.3.5 Residual Neural Network (RNN)

Residual neural networks, referred to as "ResNet", are used in the YOLO algorithms. The residual network integrates a "shortcut connection" (He et al., 2016), providing a "skip connection" for information to flow from one layer to another and skip over several layers. The main reason for skipping many layers is to dodge "vanishing/exploding gradients" in deep neural networks. The addition of such skip pathways will not harm the model's accuracy or performance, as it just skips over parts of those not useful layers (raghunandepu, 2019). Without a skip connection (see Figure 2.8), the input 'x' would be multiplied by weights ('w') of the layers in addition to some bias ('b'); giving F(wx + b), whereas a skip connection only adds an 'x' identity; F(x)+x.

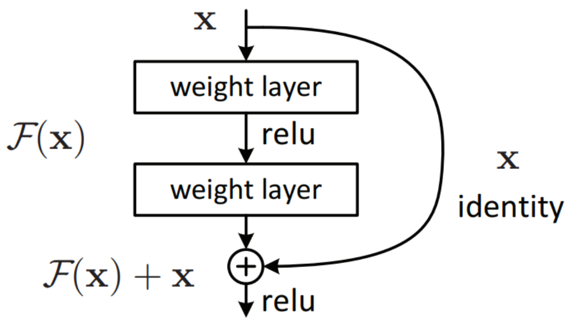

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.8 - Residual Learning: a building block (He *et al.*, 2016)

2.4 Related Work

The section focuses on the design and architectural choices made for this project. The project's initial aim was to train a few models and use those to form a pipeline that detects, classifies, and tracks the object in a single forward propagation. For detecting and classifying objects, the paper "Rich feature hierarchies for accurate object detection and semantic segmentation" (Girshick *et al.*, 2013) demonstrated an implementation of R-CNN. Object detection, firstly, the module would generate "region proposals" then a "large convolutional neural network that extracts a fixed-length feature vector from each region" (Girshick *et al.*, 2013). This method would provide the most flexibility and accuracy; however, the implementation would be highly complex, time-consuming, and require Python proficiency.

A more straightforward, attractive approach is showcased in the paper "You Only Look Once: Unified, Real-Time Object Detection" (Redmon *et al.*, 2015), implementing a one-stage object detection network. The network performs both the detection and classification in a single network compared to three separate modules approach and has similar performance and accuracy to Fast R-CNN. The YOLO algorithms also use residual blocks and "shortcut connection" described in "Deep Residual Learning for Image Recognition" (He *et al.*, 2016). The project benefits from utilising YOLO, as large chunks of code can be reused from an existing implementation by Ultralytics – reducing overall project time and complexity.

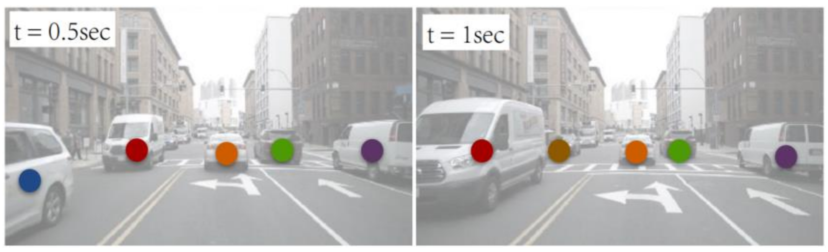


Figure 2.9 - Example Of Tracking Objects As Points (Zhou, Koltun and Krähenbühl, 2020)

The YOLO object detector can be paired up with "a point-based tracking framework for joining detection and tracking", demonstrated in the paper "Tracking Objects as Points" (Zhou, Koltun and Krähenbühl, 2020). The project can also implement a detector that takes in two consecutive frames and a heap map representing objects as points and “an offset vector from the current object [centre] to its [centre] in the previous frame” (Zhou, Koltun and Krähenbühl, 2020).

3 Methodology

3.1 Rapid Application Development (RAD)

The rapid application development (RAD) methodology minimises the planning stage and maximises the development stage. RAD provides a fast-paced environment with short turnaround periods to emphasise the development process through its iterative process. The repetitive prototyping and development cycle helps resolves bugs/issues and delivers a superior end product.

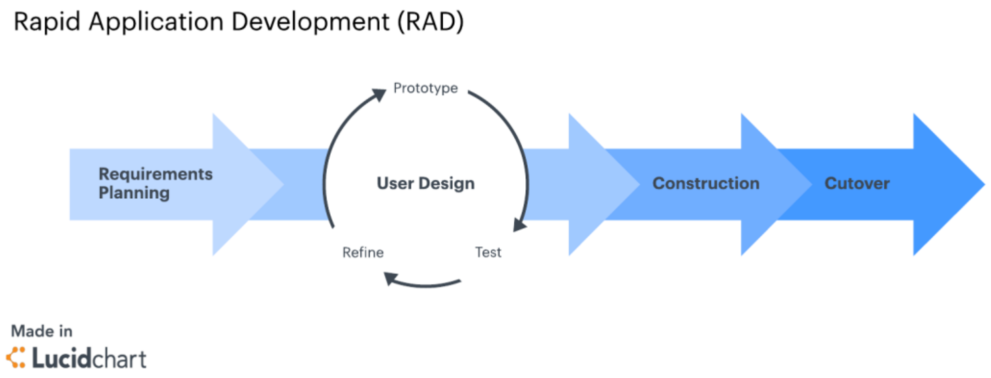


Figure 3.1 - Rapid Application Development (RAD) Methodology

3.1.1 Planning & Preparation

The planning stages involved extensive research on problem definition, background and project approach (detailed in sections 1 & 2). The dissertation was drafted early, which provided an outline framework for establishing the project brief, project approach and objective goals. For version control and to actively keep commit logs, GitHub was integrated since the start of the project. Furthermore, an automated Kanban board (see Appendix C) found in GitHub was utilised for project management. Where project aims, bugs and issues were tracked and monitored.

An airborne object tracking (AOT) dataset is selected with a community license hosted on Amazon Web Services (AWS) buckets. The dataset contains 3.3 million annotated images of numerous airborne objects: planes, helicopters, birds, flocks and hot air balloons; for further details, see Section 5. The project’s dataset was too big to be exported locally (close to 14 terabytes) and trained upon; thus, the dataset scope was limited to ensure on-time completion.

Images were obtained via a data pipeline integrated using Python, which fetches the amazon bucket and automatically exports images into a local directory. The files are already decolourised (in black & white form); therefore only need to be downscaled to 1600px resolution. An automated script generates labels by converting and normalising bounding box coordinates between 0 and 1; for more details, see Section 5.

3.1.2 User Design & Rapid Construction

In this phase, both the user design and software development were iterated repeatedly to produce a superior application. The YOLO algorithm was selected and implemented for the reasons given in Section 2.3; however, attempts were made to implement Fast R-CNN towards the start. The Ultralytics code repository is utilised, as it had the YOLO algorithm fully implemented and tested - with inbuilt scripts for importing data and training models.

In an iterative loop, the output layer was rigorously trained until high recall learning rates were achieved. Initially, training was done on a remote university pc; however, it was later migrated to Google Collab (see Section 5). The platform offered a free graphical processing unit (GPU); however, usage times were limited, and the flawed environment randomly terminated connection – resulting in loss of work. A tool called weights and biases had a range of valuable features: experiment tracking, dataset versioning and model management.

For evaluation, statistical methods such as confusion matric, line graphs were created to manual evaluate the model's accuracy and performance (see Section 6 for detailed results). The model was iteratively evaluated at the end of each training session, and a range of metrics: mAP, precision and recall, were compared between models.

3.1.3 Cutover

As an undergraduate university student, it is highly uncommon to have a real-world deployment. The model and web application were packaged in a docker container, making deployment quick and straightforward. The source code is available on GitHub for training and modification purposes.

3.2 Tools Used

When selecting the tools to use in this project, I selected Python and the Flask framework to implement the backend. An application programming interface (API) endpoint was implemented that integrated the completed machine learning model. This modularised approach allows the project's backend to be reused with a new independent frontend framework. The ReactJS framework integrates the front end as it provides a modularised approach with great flexibility for future modifications. The model was trained using Google Drive and Google Collab - which provides a free GPU; however, the inconsistent environment caused random hard crashes and disconnects. Finally, both front and back ends were containerised using Docker on completion.

Tool	Pros	Cons
Python	Excellent collection of in-built libraries. Easy to learn and develop. GPU implementation is readily available.	Problems with threading and has slower execution speeds. Underdeveloped database access layers.
Pytorch	Well documented, easy to learn and debug. Incorporates data parallelism Support for CPU and GPU. Commonly used by students and researchers.	Lack of monitoring and visualisation tools like weights & biases. Small community compared to other frameworks
ReactJS	Reusable components and virtual DOM. Excellent performance and speed. Strong community.	Lack of proper documentation. Slow development speed. No JSX support.
Flask	Scalable, flexible and lightweight. Well documented, and testing is easy.	Challenging to get familiar with it. Using many external modules – could lead to a security breach.
Google Colab	Free, easy to set up and has an excellent user interface. Free GPU	Has no IDE or linting support. Difficult to test. Limited usage & terminates the connection.
Weights & Biases	Excellent user interfaces and supports data visualisation. Free to use and provides model versioning. Collaborate in real-time.	Slow speeds. Lack of flexibility.
GitHub	Easy collaboration. Project management tools. Version control/backup for code. Excellent community	Pricing - some features are locked. Security.
Docker	Rapid development, security and has continuous integration support.	Graphical interfaces do not operate well, and the container

		ecosystem is split (only partially work).

Table 3.1 - Tools Comparison Pros & Cons

3.3 Methodologies Comparision

Waterfall methodology

The waterfall methodology is highly linear and sequential, using a waterfall approach where each process occurs in steps, and each step needs to be completed before starting the next one. It consists of 6 steps: requirement analysis, system design, development, testing, maintenance. The time constraints and working requirements are easy to manage and track(OS Systems, 2020). The cost of the project can be quickly and precisely calculated. However, the methodology is highly non-dynamic, and as no product exists until late in the project, changes or modifications would be very time-consuming. I decided not to use waterfall, as the project will constantly evolve with new changes and modifications. Furthermore, it does not have an iterative approach.

Agile Methodology

The agile methodology is centred around an iterative and incremental model. The developer begins with developing small modules, and after completion, the client evaluates the work and provides feedback. The agile approach can easily manage feedback in time and has a highly dynamic approach (OS Systems, 2020). However, the final product may not be what was intended due to its flexibility. Furthermore, this methodology was not used as the project does not have any clients, and agile best works with a small team.

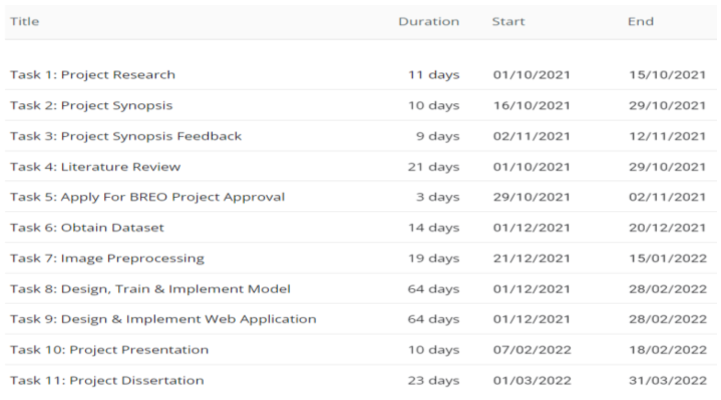
Spiral Methodology

The spiral methodology is a risk-driven development methodology combining a waterfall model and an iterative model. The iterative cycle has four stages: identification, design, risk evaluation and development, which helps with continuous development and helps with risk management. However, this methodology does not allow the ability to overlap stages compared to RAD; each stage needs to be completed before starting the next one.

3.4 Ethical Consideration

Ethics were considered, and an approval letter (see Appendix B) from Brunel Research Ethics Online (BREO) was obtained. The project does not require any human participants at any stage.

3.5 Project Management Tools



Title	Duration	Start	End
Task 1: Project Research	11 days	01/10/2021	15/10/2021
Task 2: Project Synopsis	10 days	16/10/2021	29/10/2021
Task 3: Project Synopsis Feedback	9 days	02/11/2021	12/11/2021
Task 4: Literature Review	21 days	01/10/2021	29/10/2021
Task 5: Apply For BREO Project Approval	3 days	29/10/2021	02/11/2021
Task 6: Obtain Dataset	14 days	01/12/2021	20/12/2021
Task 7: Image Preprocessing	19 days	21/12/2021	15/01/2022
Task 8: Design, Train & Implement Model	64 days	01/12/2021	28/02/2022
Task 9: Design & Implement Web Application	64 days	01/12/2021	28/02/2022
Task 10: Project Presentation	10 days	07/02/2022	18/02/2022
Task 11: Project Dissertation	23 days	01/03/2022	31/03/2022

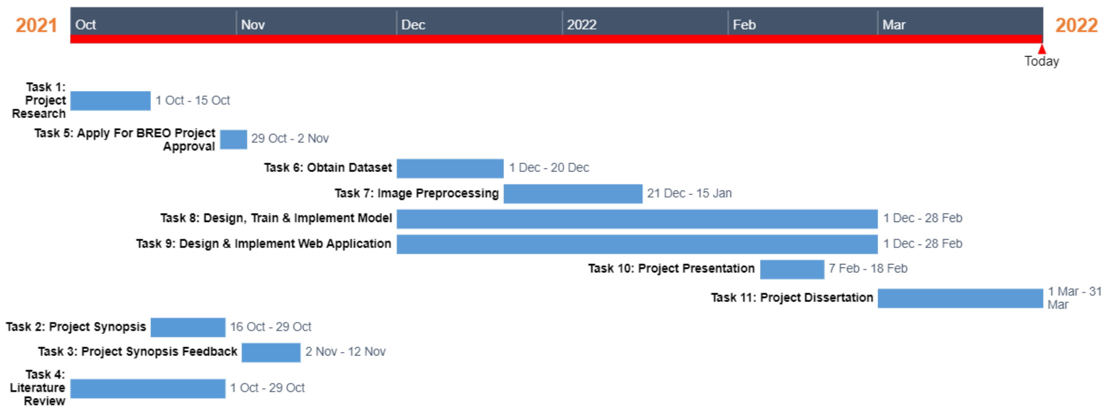


Figure 3.2 - Gantt Chart For Project Management

Figure 3.2 displays the Gantt chart, a timeline that represents the completion of each task. The chart was created as a plan to help with time management, monitoring projects progress, and setting milestones.

GitHub was incorporated for version control of the project's development (see Appendix C) and to have a backup copy of the code repository in case of any data loss. Weights and Biases tool was utilised to keep a version control history for the model's training process. The Kanban board feature with the Gantt chart (see Figure 3.2) was used in the development process to assist with project management.

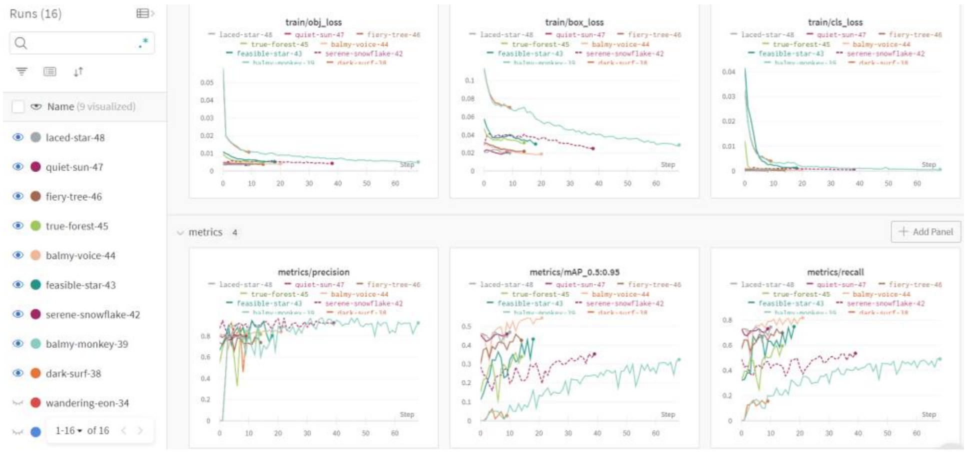


Figure 3.3 - Weights & Biases Project Management

4 Design

4.1 Frontend

4.1.1 Application Architecture Design

The project focuses on a single-tier architecture, where all layers: presentation, business and data, are in one place. Figure 4.1 demonstrates all API calls made by the presentation to the business layer. It focuses on reducing project complexity by decoupling the backend from the frontend, allowing future modifications. Furthermore, a decoupled architecture increases the project's reusability by allowing a different frontend framework to be swapped over. The application is containerised (shown in blue, see Figure 4.1) using Docker, permitting future deployments to be quicker and simpler. An external emailing service was incorporated as it provided convenience and, more importantly, reduced project complexity by not having to build and host a separate email server.

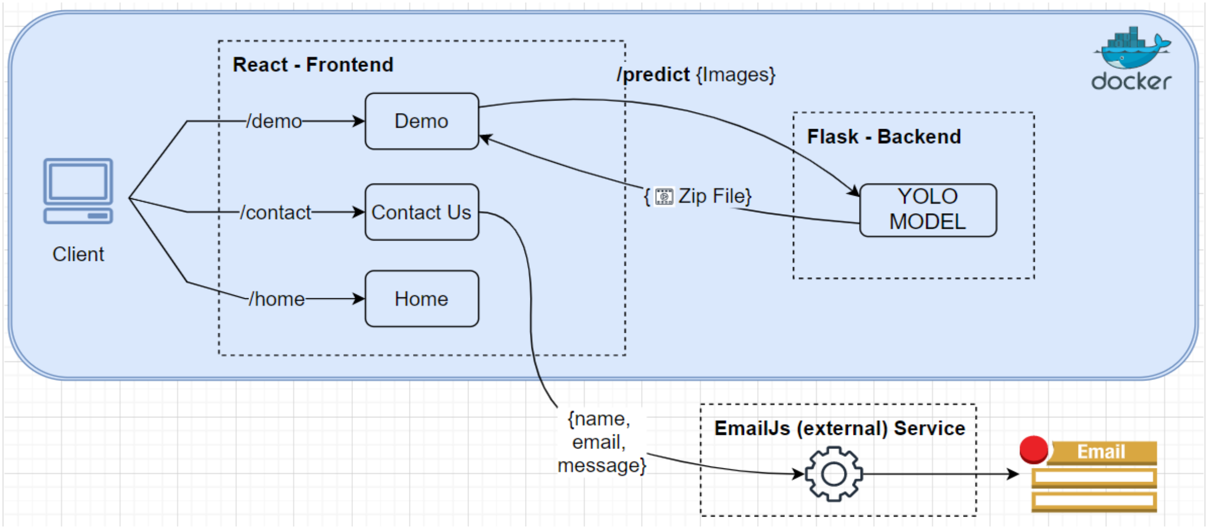


Figure 4.1 - Application's Architecture

4.1.2 Prototypes

A prototype is a draft version of a product that allows you to explore and illustrate your intention behind the overall project design before investing time into developing it. As prototypes are early in product development, many iterations are done to build up a superior final product. Low-fidelity prototypes are usually paper-based drafts, typically quickly hand-drawn to lay down a foundation and make changing effortless. As demonstrated in Figure 4.2, the creation of these low-fidelity prototypes provided a structure to further improve upon.

High-fidelity prototypes (see Appendix D) are traditionally computer-based, usually allowing users to have some degree of interaction. The prototypes created (see Appendix D) are inspired to design a simple, minimalistic modern user interface with minimal clutter and distractions. The design has excellent visibility and consistency between pages, as each page focuses either on delivering information or executing a service. Appropriate icons and colours were integrated to enhance the user's recall. Clear markings, feedback prompts and instructions are specifically added to improve documentation and accessibility.

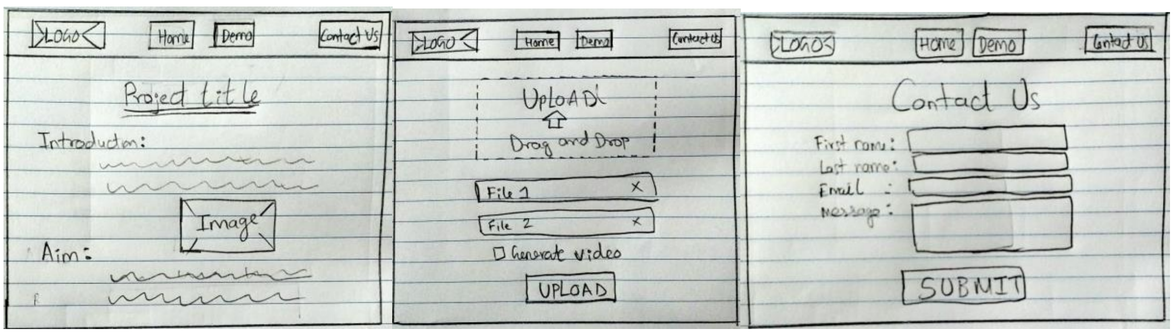


Figure 4.2 - Low Fidelity Prototypes (Frontend)

4.2 Backend & Dataset

4.2.1 Data Scrapper & Import Pipeline

Figure 4.3 displays the data structure entity included in the ground truth file. Each flight contains a collection of entities representing the frame sequence, where each entity corresponds to an image. The image name follows a convention of <timestamp><flight_id>.png, and each image is 2448px wide by 2048px high; encoded in 8-bit grayscale.

```
{
  "time": 1573043646380340792,  # Timestamp associated with the image
  "blob": {
    "frame": 3,  # Frame number associated with the image
    "range_distance_m": 1366,  # (optional) Distance to planned airborne objects [m]
  },
  "id": "Airplane1",  # (optional) Identifier for the label (unique for the sequence)
  "bb": [1355.2, 1133.4, 6.0, 6.0],  # (optional), Bounding box [top, left, width, height]
  "labels": {
    "is_above_horizon": -1, # the object is Below(-1)/Not clear(0)/Above(1) the horizon
  },
  "flight_id": "673f29c3e4b4428fa26bc55d812d45d9",
  "img_name": "1566556046185850341673f29c3e4b4428fa26bc55d812d45d9.png",
}
```

Figure 4.3 - Data Definition Example (Structure Block)

As previously discussed in Section 3.1.1, the dataset size is too large (14 TB) and thus, cannot be directly imported locally; therefore, a data import pipeline was integrated. Firstly, a script iterates the ground truth JSON file and downloads a flight at a time, image by image (see Figure 4.5). The flowchart in Figure 4.4 illustrates how files are retrieved from start to finish, where the images are retrieved in small batches, and their resolution scaled down to 1600px. This resolution was selected as many frames contained airborne objects as wide as few pixels. Additionally, a lower resolution would produce lower quality feature maps, which would hurt the model's performance. The pre-processed images are split in an 80:20 ratio (training : testing), and lastly, all the files get deleted.

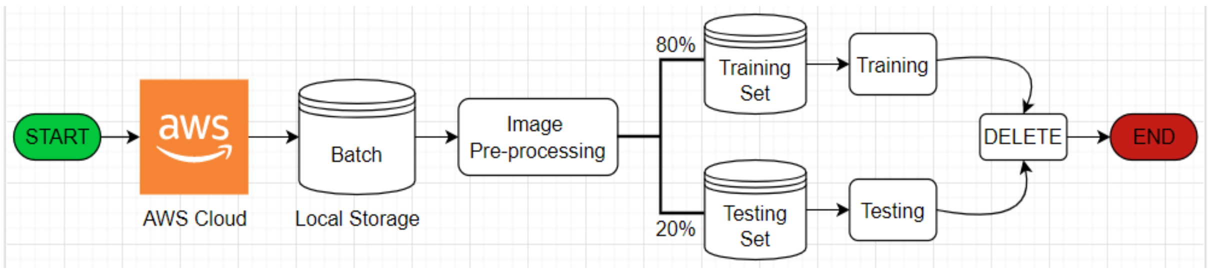


Figure 4.4 - Dataset Scrapping & Import Pipeline Flowchart

```
1    Define an export area and a dataset object
2    Download the ground truth JSON file if not present
3    Convert ground truth data into custom flight objects and add them to dataset
4    Download all flights present in the dataset object into the defined export area
5    For each image, read the object's class and bounding box coordinates and normalise it
     between 0 and 1 relative to the image size
6    Write the labels and the normalised values into a text file
7    Downscale and decolourise each image
8    Split the prepared batch into two separate folders: training (80%) and testing (20%)
9    Store all the labels and images for training
```

Figure 4.5 - Data Scrapping & Import Pipeline Pseudocode

4.2.2 Data Storage & Training Checkpoints

Initially, the training environment was integrated on a remote university computer, with a terabyte of storage space – available 24/7. The large disk size would enable more significant batch sizes to be trained simultaneously. Due to hardware issues (Nvidia CUDA not incompatible with GPU), the training was migrated to Google Collab (as discussed in Section 3.1.2), limiting the storage space to 100GB – drastically reducing batch sizes.

Collabs free GPU was capped at 4-5 hrs/day (mileage varied each day) with random captcha prompts every 30 minutes; to prevent bots from abusing it. The environment would terminate the connection if it were left idle for 10+ minutes or exceeded the daily usage limit. Fortunately, the YOLO repository by Ultralytics included an inbuilt checkpoint feature that would export the model's weight after each epoch. This design approach is exceptionally beneficial as it provides a safety net if a failure occurs; training would retain. Furthermore, storing the entirety of the dataset in one location helps minimise data duplication while maximising disk efficiency.

4.2.3 Data Augmentation Design

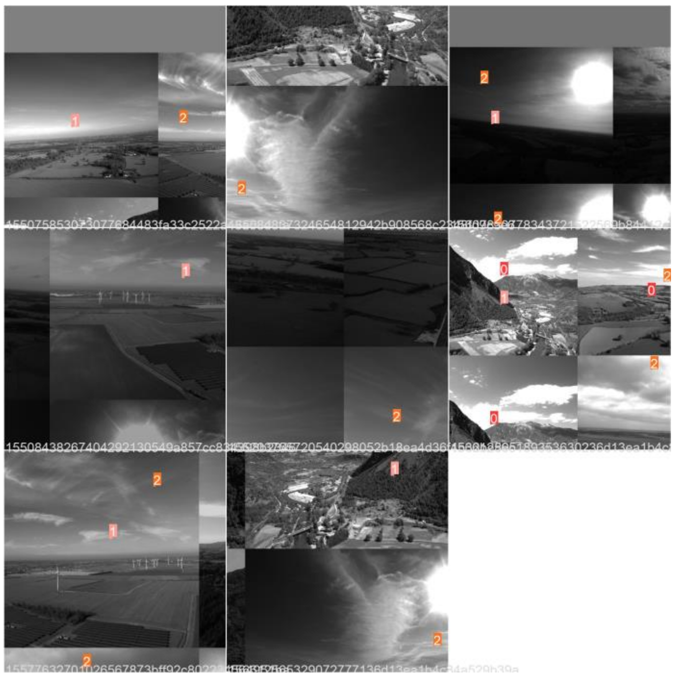


Figure 4.6 - Mosaic Data Augmentation

Figure 4.6 demonstrates a fluid mosaic for training data; this technique has been implemented in the YOLO algorithm and focuses on combining four training images to form a complex. The approach helps the model learn how to detect small objects at a smaller scale than usual. The data augmentation technique is particularly beneficial when training on natural world objects. Additionally, it forces the model to learn objects at different locations in a frame by providing unusual scenarios.

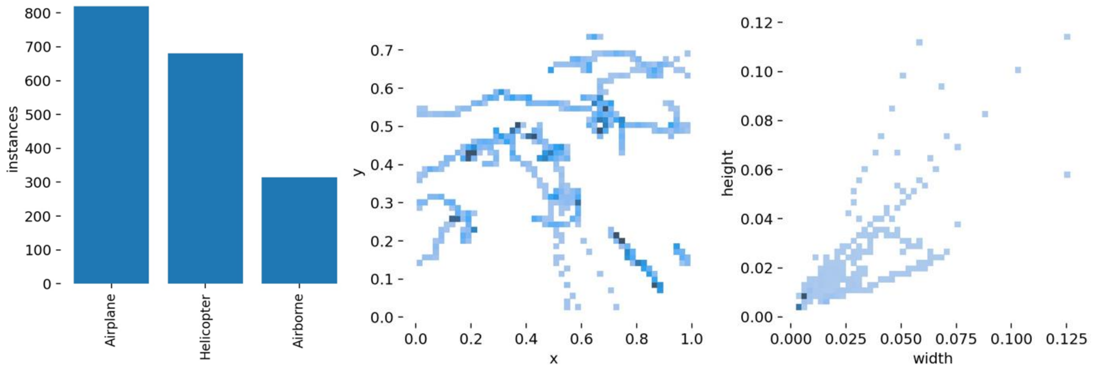


Figure 4.7 – Training Set Distribution

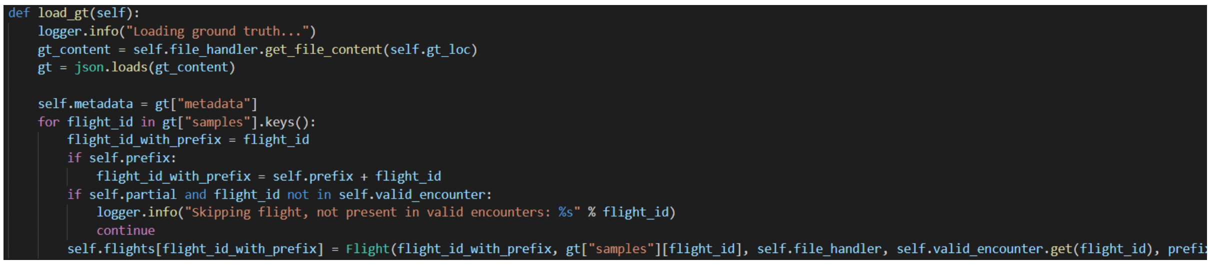
Figure 4.7 explores the design approach for preparing a typical training batch. This approach ensures no bias is introduced and further illustrates that the AOT dataset has been randomly distributed. Therefore, no stratified sampling would be necessary when preparing a training batch. Furthermore, the figure conveys that most objects would have small bounding box dimensions and that their location is random; random x and y coordinate distribution.

5 Implementation

5.1 Model Implementation

5.1.1 Importing & Defining Data

Before training can start, the data needs to be in a correct structure that contains several steps. Figure 5.1 shows the method which converts the JSON ground truth file into a custom dataset object (provided in the starter kit) (AIcrowd, 2021). If the ground truth file is not present in the specified directory, the method retrieves it and then reads it into an object. A for loop traverses through each flight in the object; prefix (if provided) is combined with the flight id. Lastly, if the partial value is false and flight is not present in the valid encounters array, it is skipped and a log message prints. Otherwise, a new flight object is declared and appended to the dataset object. This approach is drastically better than using a Panda Dataframe, as it provides flexibility and forms complex nested objects.

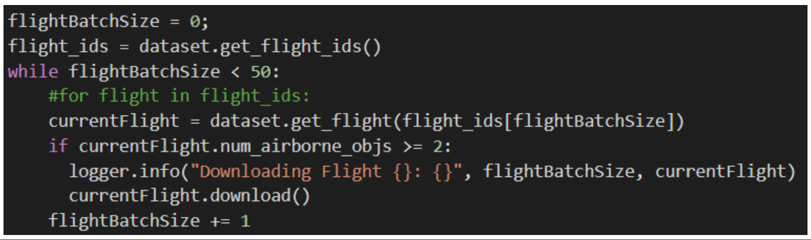


```
def load_gt(self):
    logger.info("Loading ground truth...")
    gt_content = self.file_handler.get_file_content(self.gt_loc)
    gt = json.loads(gt_content)

    self.metadata = gt["metadata"]
    for flight_id in gt["samples"].keys():
        flight_id_with_prefix = flight_id
        if self.prefix:
            flight_id_with_prefix = self.prefix + flight_id
        if self.partial and flight_id not in self.valid_encounter:
            logger.info("Skipping flight, not present in valid encounters: %s" % flight_id)
            continue
        self.flights[flight_id_with_prefix] = Flight(flight_id_with_prefix, gt["samples"][flight_id], self.file_handler, self.valid_encounter.get(flight_id), prefi
```

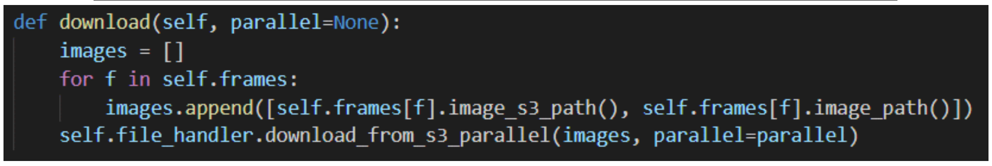
Figure 5.1 - Importing & Defining Data Structure

5.1.2 Data Scrapping

Once the dataset assembles, a while loop iteratively until the batch size matches counter retrieves the corresponding flight object from the dataset object. For each flight, its flight metadata is retrieved and an if condition validates if the flight consists of at least two or more airborne objects. If not, the flight is skipped, as this approach would help refine our training set and provide better accuracy due to a wide variation in the set. The download method (see Figure 5.2) is executed, iterating through all the frames present in that flight entity. For each frame, a list is assembled, consisting of an S3 bucket path and path to the export folder and appended to the images list. Lastly, a parallel pooling process is initialised that downloads images listed in the images array using an instance of a boto3 client. Boto3 is a low-level class that directly connects to Amazon Web Services (AWS) bucket server and downloads multiple files at a time. This approach uses the inbuilt AWS API, delivering excellent speed and reliability through its inbuilt multiprocessing feature. As previously discussed in Section 4.2.2, small training batches were assembled due to limited storage space (50GB) on Google Drive.



```
flightBatchSize = 0;
flight_ids = dataset.get_flight_ids()
while flightBatchSize < 50:
    #for flight in flight_ids:
    currentFlight = dataset.get_flight(flight_ids[flightBatchSize])
    if currentFlight.num_airborne_objs >= 2:
      logger.info("Downloading Flight {}: {}", flightBatchSize, currentFlight)
      currentFlight.download()
    flightBatchSize += 1
```

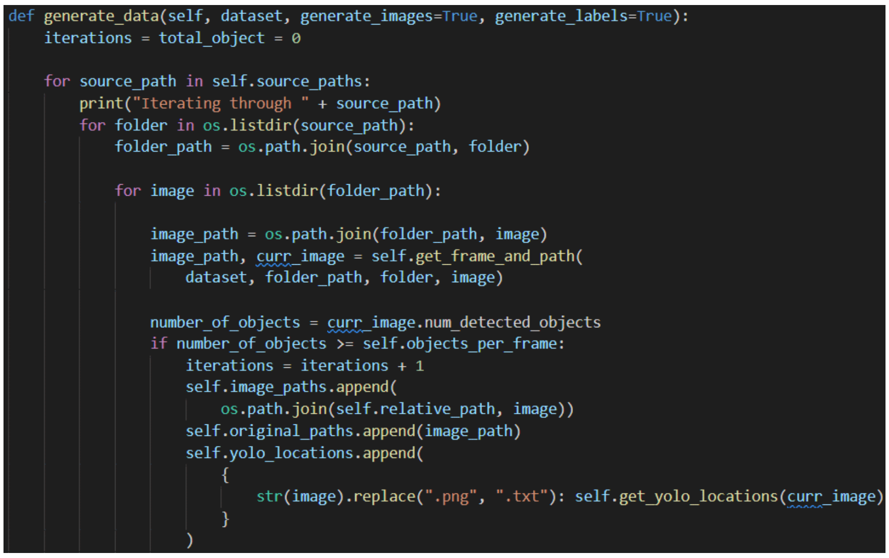


```
def download(self, parallel=None):
    images = []
    for f in self.frames:
        images.append([self.frames[f].image_s3_path(), self.frames[f].image_path()])
    self.file_handler.download_from_s3_parallel(images, parallel=parallel)
```

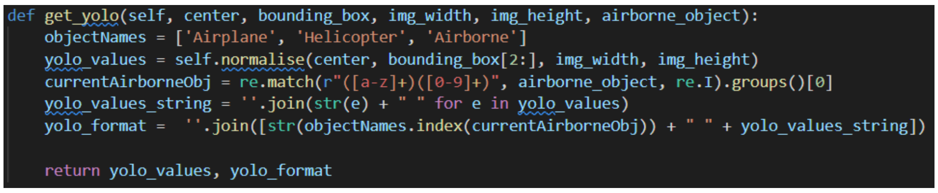
Figure 5.2 - Downloading Files From AWS S3 Bucket

5.1.3 Generating Custom Labels For YOLO

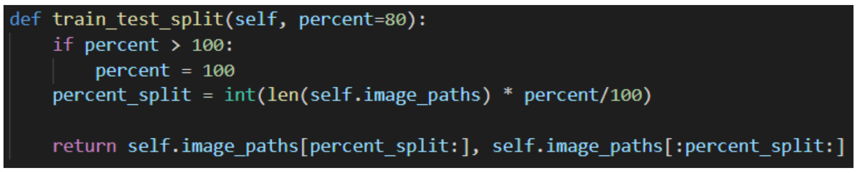
The training phase not only requires images but a label corresponding to the object's bounding box coordinates and the class of the object. When the generate data method is executed, which begins by iterating through the supplied directory. Each image file's corresponding details are retrieved from the dataset object to perform a final validation; if the number of objects exceeds the set threshold. If all conditions satisfy, get_yolo() should execute with the object's bounding box passed as parameters (see Figure 5.3). An objectName list is initialised, containing the three most seen object classes; to limit project scope due to hardware and training constraints and achieve better accuracy, discussed in Section 4.2.2.



```
def generate_data(self, dataset, generate_images=True, generate_labels=True):
    iterations = total_object = 0

    for source_path in self.source_paths:
        print("Iterating through " + source_path)
        for folder in os.listdir(source_path):
            folder_path = os.path.join(source_path, folder)

            for image in os.listdir(folder_path):

                image_path = os.path.join(folder_path, image)
                image_path, curr_image = self.get_frame_and_path(
                    dataset, folder_path, folder, image)

                number_of_objects = curr_image.num_detected_objects
                if number_of_objects >= self.objects_per_frame:
                    iterations = iterations + 1
                    self.image_paths.append(
                        os.path.join(self.relative_path, image))
                    self.original_paths.append(image_path)
                    self.yolo_locations.append(
                        {
                            str(image).replace(".png", ".txt"): self.get_yolo_locations(curr_image)
                        }
                    )
```



```
def get_yolo(self, center, bounding_box, img_width, img_height, airborne_object):
    objectNames = ['Airplane', 'Helicopter', 'Airborne']
    yolo_values = self.normalise(center, bounding_box[2:], img_width, img_height)
    currentAirborneObj = re.match(r"([a-z]+)([0-9]+)", airborne_object, re.I).groups()[0]
    yolo_values_string = ''.join(str(e) + " " for e in yolo_values)
    yolo_format =  ''.join([str(objectNames.index(currentAirborneObj)) + " " + yolo_values_string])

    return yolo_values, yolo_format
```



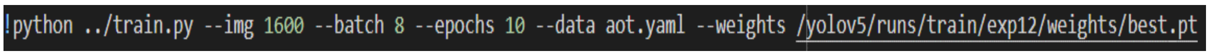
```
def train_test_split(self, percent=80):
    if percent > 100:
        percent = 100
    percent_split = int(len(self.image_paths) * percent/100)

    return self.image_paths[percent_split:], self.image_paths[:percent_split:]
```

Figure 5.3 - Steps For Generating Training Labels (Simplified)

Later, bounding box values would be normalised between 0 to 1; by scaling the provided values down by the width and height of the frame. The airborne object value is passed through a regex match function to remove any numbers or symbols and only return the class. Finally, the function returns normalised bounding box values and the object class index to which the label belongs (see Figure 5.3). The generated labels have the same title as the image, with a ".txt" extension. The batch gets split into an 80% training set and a 20% testing set; each corresponding image path gets written to its text file.

5.1.4 Training YOLO Model

The model's training was facilitated with the Ultralytics code repository, containing an implementation of you only look once (YOLO) v5 algorithm and with necessary scripts to train a custom model. The project uses a pre-trained model, yolov5s, for its initial weights, as it would provide better accuracy than having random weights assigned to it. The yolov5s model is small, with an accuracy of 37.2 mAP on the COCO dataset and an inference time of 6.4 ms on an Nvidia V100. An "aot.yaml" file is declared, specifying the directories of the training set, testing set and a list of airborne objects. Figure 5.4 conveys the commands need to run to start training.



```
!python ../train.py --img 1600 --batch 8 --epochs 10 --data aot.yaml --weights /yolov5/runs/train/exp12/weights/best.pt
```

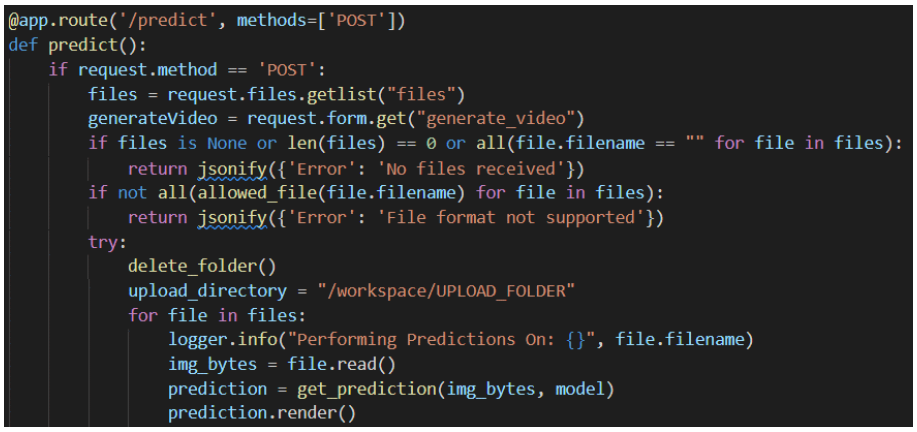
Figure 5.4 - Google Collab Training Command

The command firstly takes in "train.py", the python file containing training code. Followed by the image size, set to 1600 pixels, as the airborne object within the images were only a few pixels broad, meaning detail would be lost when downscaled too harshly. Additionally, using lower resolution images, train.py would raise errors regarding best precision-recall values, or the current anchors are not a good fit for the dataset. The batch represents the number of images that can be loaded at once, depending on the GPU or CPU memory. The epoch represents the number of passes on the entire dataset, which varied depending on if the previous run crashed or not. The weights parameter takes the weight files to be trained on; the last best weight is provided for the next training run. Other variables such as momentum, learning rate and weight decays were unchanged; to avoid the model from over or underfitting.

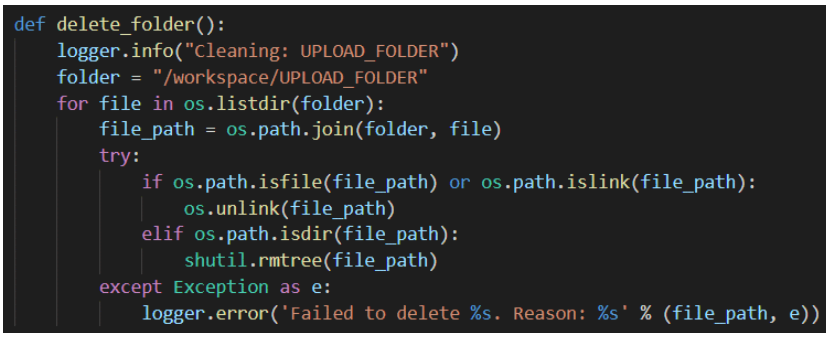
5.2 Backend (Flask)

5.2.1 Prediction API Implementation

The backend uses the Flask framework, a small, lightweight micro-framework with little to no dependencies and provides valuable libraries to build a working web application compared to other alternatives like Django. The final machine learning model gets implemented into an API endpoint, shown in Figure 5.5; the app routes the POST query on the '/predict' URL to the predict methods. Firstly, a list of files for the keys: "files" and "generate_video" gets collected and assigned their variable. A JSON object is returned with an error message if no files get provided. Similarly, an error message would get returned if the files provided do not match the support extensions. Each file gets read as bytes, passed into get_prediction as a parameter with the model, and the predictions get rendered.



```python
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        files = request.files.getlist("files")
        generateVideo = request.form.get("generate_video")
        if files is None or len(files) == 0 or all(file.filename == "" for file in files):
            return jsonify({'Error': 'No files received'})
        if not all(allowed_file(file.filename) for file in files):
            return jsonify({'Error': 'File format not supported'})
        try:
            delete_folder()
            upload_directory = "/workspace/UPLOAD_FOLDER"
            for file in files:
                logger.info("Performing Predictions On: {}", file.filename)
                img_bytes = file.read()
                prediction = get_prediction(img_bytes, model)
                prediction.render()
```

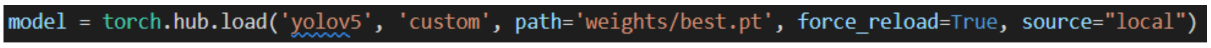


```python
def delete_folder():
    logger.info("Cleaning: UPLOAD_FOLDER")
    folder = "/workspace/UPLOAD_FOLDER"
    for file in os.listdir(folder):
        file_path = os.path.join(folder, file)
        try:
            if os.path.isfile(file_path) or os.path.islink(file_path):
                os.unlink(file_path)
            elif os.path.isdir(file_path):
                shutil.rmtree(file_path)
        except Exception as e:
            logger.error('Failed to delete %s. Reason: %s' % (file_path, e))
```

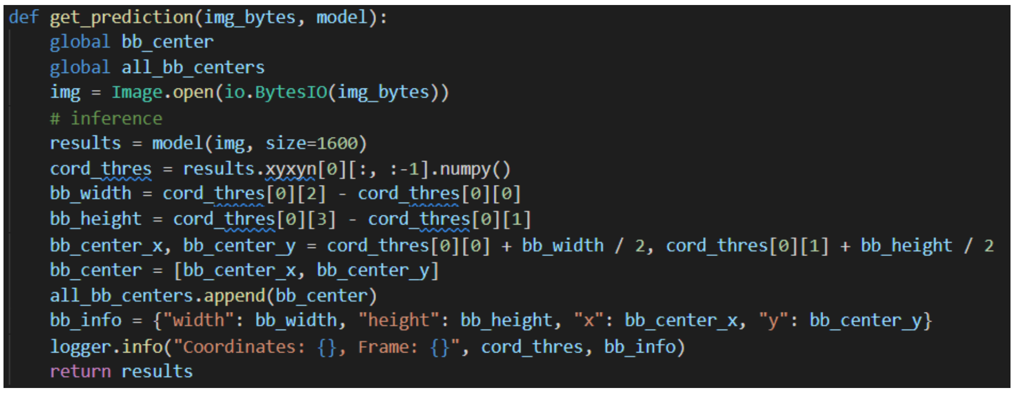
Figure 5.5 - Model API Integration (FLASK) Part 1/3

The delete_folder method is executed (see Figure 5.6), aiming to delete all files in the upload directory folder. Firstly, a simple message gets consoled, followed by a for loop iterating over the files in the provided folder. A file path gets assembled using the join operator; to combine the folder directory with the file's directory. An if condition check where the provided file path is either a file or a symbolic link; if so, the unlink method would remove the file. If the condition were not satisfied, another condition would evaluate if the path is a directory and remove it by calling the rmtree method.

Figure 5.6 showcases the integration of the final model, where torch - a machine learning library, is utilised to load the machine learning model. The source parameter gets set to "local" in the load method, representing how the repository supposes to get interpreted. The path to best weights and a model name would get supplied, which outputs a callable model.



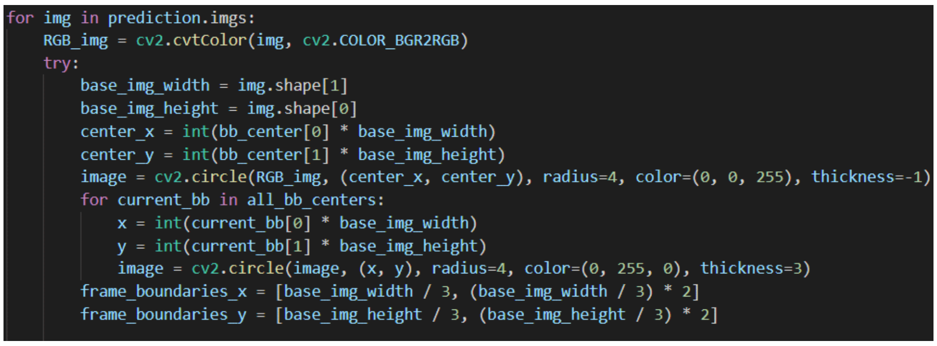
```
model = torch.hub.load('yolov5', 'custom', path='weights/best.pt', force_reload=True, source="local")
```



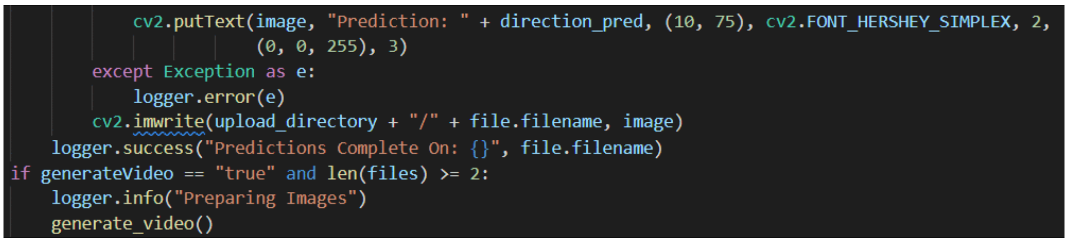
```
def get_prediction(img_bytes, model):
    global bb_center
    global all_bb_centers
    img = Image.open(io.BytesIO(img_bytes))
    # inference
    results = model(img, size=1600)
    cord_thres = results.xyxyn[0][:, :-1].numpy()
    bb_width = cord_thres[0][2] - cord_thres[0][0]
    bb_height = cord_thres[0][3] - cord_thres[0][1]
    bb_center_x, bb_center_y = cord_thres[0][0] + bb_width / 2, cord_thres[0][1] + bb_height / 2
    bb_center = [bb_center_x, bb_center_y]
    all_bb_centers.append(bb_center)
    bb_info = {"width": bb_width, "height": bb_height, "x": bb_center_x, "y": bb_center_y}
    logger.info("Coordinates: {}, Frame: {}", cord_thres, bb_info)
    return results
```

Figure 5.6 - Model API Integration (FLASK) Part 2/3

The get_prediction method (shown in Figure 5.6) reads, converts the image into an image object on which operations can get performed - part of the python pillow library. The image then gets passed into the model image size set to 1600 pixels, as trained at this resolution. Bounding box coordinates get extracted from the results and further operated to calculate the centre of the bounding box and stored in a list; a visual trail can be plotted on the image. Lastly, the results object gets returned to the predict method.



```
for img in prediction.imgs:
    RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    try:
        base_img_width = img.shape[1]
        base_img_height = img.shape[0]
        center_x = int(bb_center[0] * base_img_width)
        center_y = int(bb_center[1] * base_img_height)
        image = cv2.circle(RGB_img, (center_x, center_y), radius=4, color=(0, 0, 255), thickness=-1)
        for current_bb in all_bb_centers:
            x = int(current_bb[0] * base_img_width)
            y = int(current_bb[1] * base_img_height)
            image = cv2.circle(image, (x, y), radius=4, color=(0, 255, 0), thickness=3)
        frame_boundaries_x = [base_img_width / 3, (base_img_width / 3) * 2]
        frame_boundaries_y = [base_img_height / 3, (base_img_height / 3) * 2]
```



```
            cv2.putText(image, "Prediction: " + direction_pred, (10, 75), cv2.FONT_HERSHEY_SIMPLEX, 2,
                        (0, 0, 255), 3)
        except Exception as e:
            logger.error(e)
        cv2.imwrite(upload_directory + "/" + file.filename, image)
    logger.success("Predictions Complete On: {}", file.filename)
if generateVideo == "true" and len(files) >= 2:
    logger.info("Preparing Images")
    generate_video()
```

Figure 5.7- Model API Integration (FLASK) Part 3/3

The results get rendered, and a box is placed on the detected object with a label and confidence score. Figure 5.7 demonstrates parts of code that track the detected object's position relative to the frame and predict a manoeuvre to avoid it. Firstly, the coordinates for the centre of the bounding box get calculated; normalised values get converted by multiplying with image dimensions. A flight trail is outlined (see Figure 5.8) for each image by plotting every centre point of all previous bounding boxes, represented by the all_bb_centers array.

A simple avoidance is integrated by dividing the whole frame into nine grid cells. Each object's location is calculated relative to the image frame, and the object gets assigned to only a single cell. Through a series of conditional statements, the cells with the objects recognised and algorithms suggest the furthest directly opposite cells. Finally, shown in Figure 5.7, the prediction gets labelled in the top-left corner (see Figure 5.8), then all the folder contents are zipped and returned to the frontend.

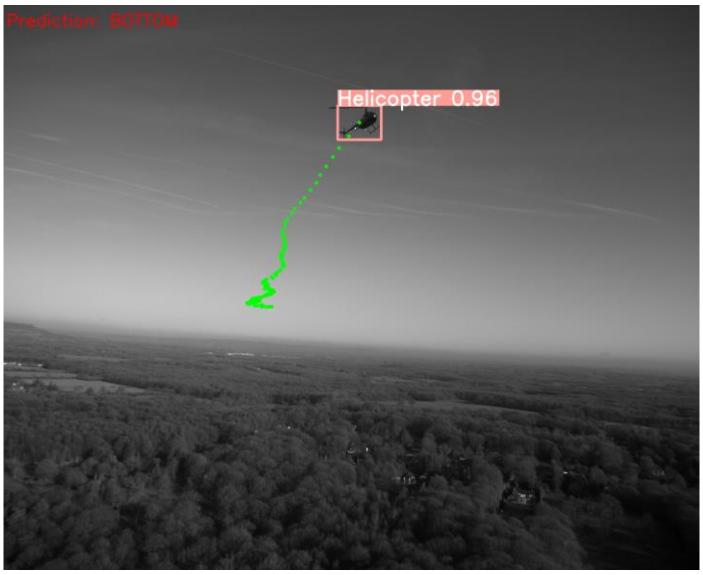


Figure 5.8 - Example of An Actual Frame (Result)

5.3 Frontend (React)

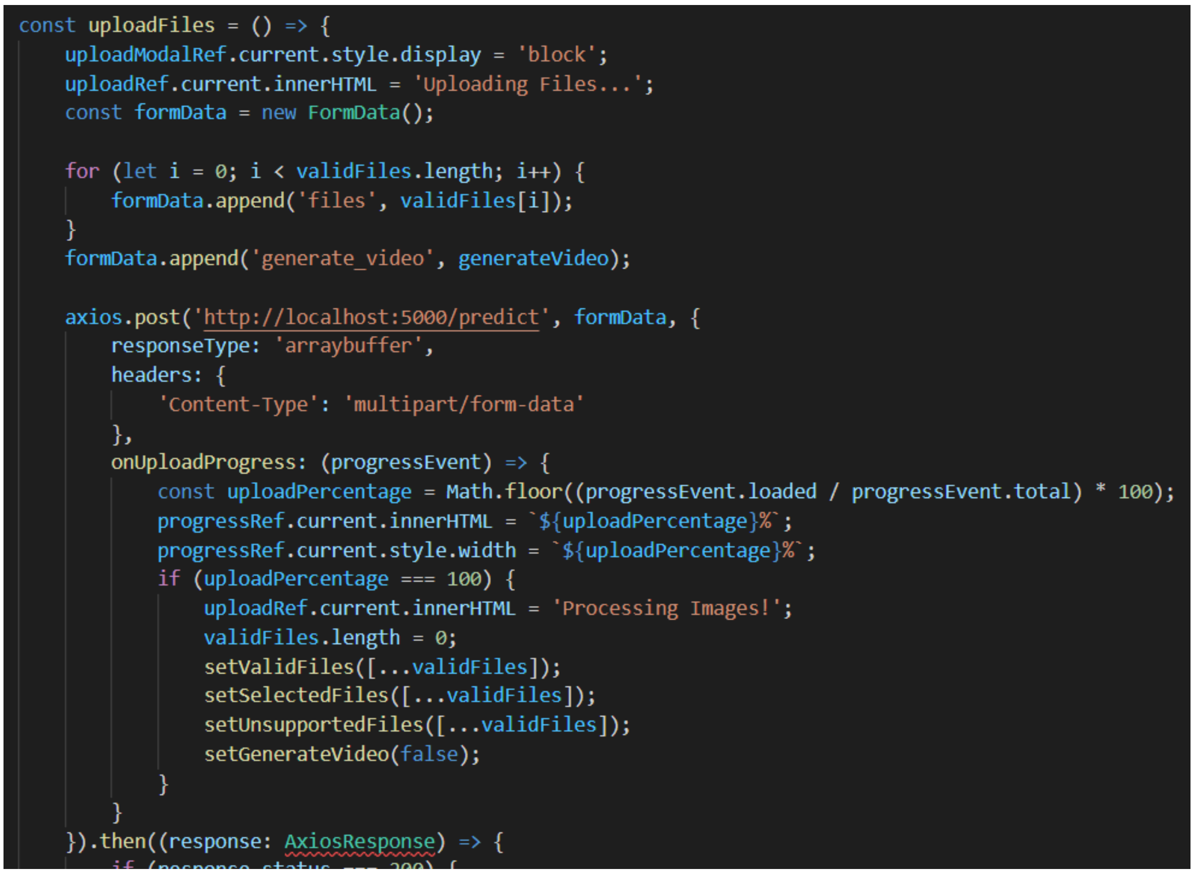
The ReactJS framework was selected to implement the frontend user interface. The React framework can reuse components, has excellent performance and has strong community support (see Table 3.1). React is one of the more accessible languages to pick up compared to other frameworks like Angular and NextJS. For the project's final frontend framework, see Figures 5.10-5.12, which articulates the implementation of a simple, minimalistic, modern user interface (UI).

Figure 5.9 illustrates the drag & drop component upload functionality implemented using Axios, a promised-based React library that uses Javascript's asynchronous and await operators to serve and create HTTP requests. Using Axios over the traditional fetch provided more backwards compatibility, the onProgress handler lets a progress bar to integrated, and Axios can make simultaneous requests at once.

The method shown in Figure 5.9 updates display attributes to make the upload progress bar visible. A new instance of FormData gets declared, and a for loop appends formData with all valid files. An asynchronous POST request gets made to the backend with the relevant formData, and the progress bars get updated with the percentage of progress completed (see Figure 5.11).

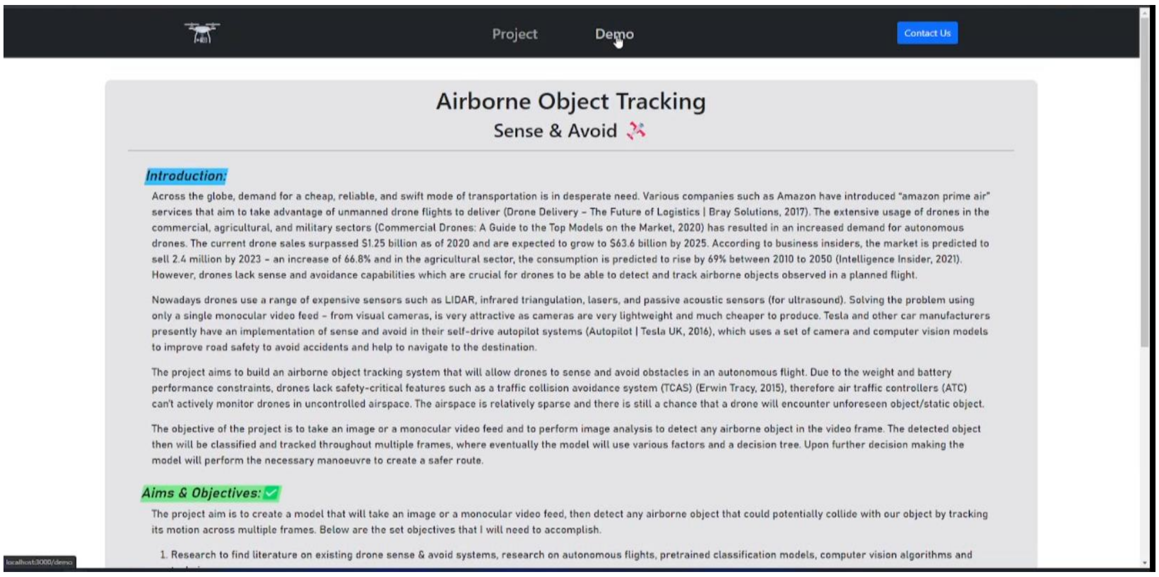
The UI gets cleared and refreshed without reloading the page once all the files are uploaded. Lastly, the zipped file gets automatically downloaded to the user's local machine (contents similar to Figure 5.8).

The UI is kept simple and consistent throughout the application (see Figures 5.10-5.12), with each page focusing on executing a service or portraying information to the user. For example, the front page (see Figure 5.10) only displays project information and the contact us page (see Figure 5.12) only sends form data to an external service. This implementation approach drastically improves users' visibility and prevents errors as much as possible. An upload modal (Figure 5.11) and clear errors provide users with informative feedback. The UI is mobile-friendly, further increasing flexibility and user control.



```
const uploadFiles = () => {
    uploadModalRef.current.style.display = 'block';
    uploadRef.current.innerHTML = 'Uploading Files...';
    const formData = new FormData();

    for (let i = 0; i < validFiles.length; i++) {
        formData.append('files', validFiles[i]);
    }
    formData.append('generate_video', generateVideo);

    axios.post('http://localhost:5000/predict', formData, {
        responseType: 'arraybuffer',
        headers: {
            'Content-Type': 'multipart/form-data'
        },
        onUploadProgress: (progressEvent) => {
            const uploadPercentage = Math.floor((progressEvent.loaded / progressEvent.total) * 100);
            progressRef.current.innerHTML = `${uploadPercentage}%`;
            progressRef.current.style.width = `${uploadPercentage}%`;
            if (uploadPercentage === 100) {
                uploadRef.current.innerHTML = 'Processing Images!';
                validFiles.length = 0;
                setValidFiles([...validFiles]);
                setSelectedFiles([...validFiles]);
                setUnsupportedFiles([...validFiles]);
                setGenerateVideo(false);
            }
        }
    }).then((response: AxiosResponse) => {
        if (response.status === 200) {
```

Figure 5.9 - Uploading Files Using Axios (Frontend)



Airborne Object Tracking

Sense & Avoid

Introduction:

Across the globe, demand for a cheap, reliable, and swift mode of transportation is in desperate need. Various companies such as Amazon have introduced "amazon prime air" services that aim to take advantage of unmanned drone flights to deliver (Drone Delivery – The Future of Logistics | Bray Solutions, 2017). The extensive usage of drones in the commercial, agricultural, and military sectors (Commercial Drones: A Guide to the Top Models on the Market, 2020) has resulted in an increased demand for autonomous drones. The current drone sales surpassed $1.25 billion as of 2020 and are expected to grow to $63.6 billion by 2025. According to business insiders, the market is predicted to sell 2.4 million by 2023 – an increase of 66.8% and in the agricultural sector, the consumption is predicted to rise by 69% between 2010 to 2050 (Intelligence Insider, 2021). However, drones lack sense and avoidance capabilities which are crucial for drones to be able to detect and track airborne objects observed in a planned flight.

Nowadays drones use a range of expensive sensors such as LIDAR, infrared triangulation, lasers, and passive acoustic sensors (for ultrasound). Solving the problem using only a single monocular video feed – from visual cameras, is very attractive as cameras are very lightweight and much cheaper to produce. Tesla and other car manufacturers presently have an implementation of sense and avoid in their self-drive autopilot systems (Autopilot | Tesla UK, 2016), which uses a set of camera and computer vision models to improve road safety to avoid accidents and help to navigate to the destination.

The project aims to build an airborne object tracking system that will allow drones to sense and avoid obstacles in an autonomous flight. Due to the weight and battery performance constraints, drones lack safety-critical features such as a traffic collision avoidance system (TCAS) (Erwin Tracy, 2015), therefore air traffic controllers (ATC) can't actively monitor drones in uncontrolled airspace. The airspace is relatively sparse and there is still a chance that a drone will encounter unforeseen object/static object.

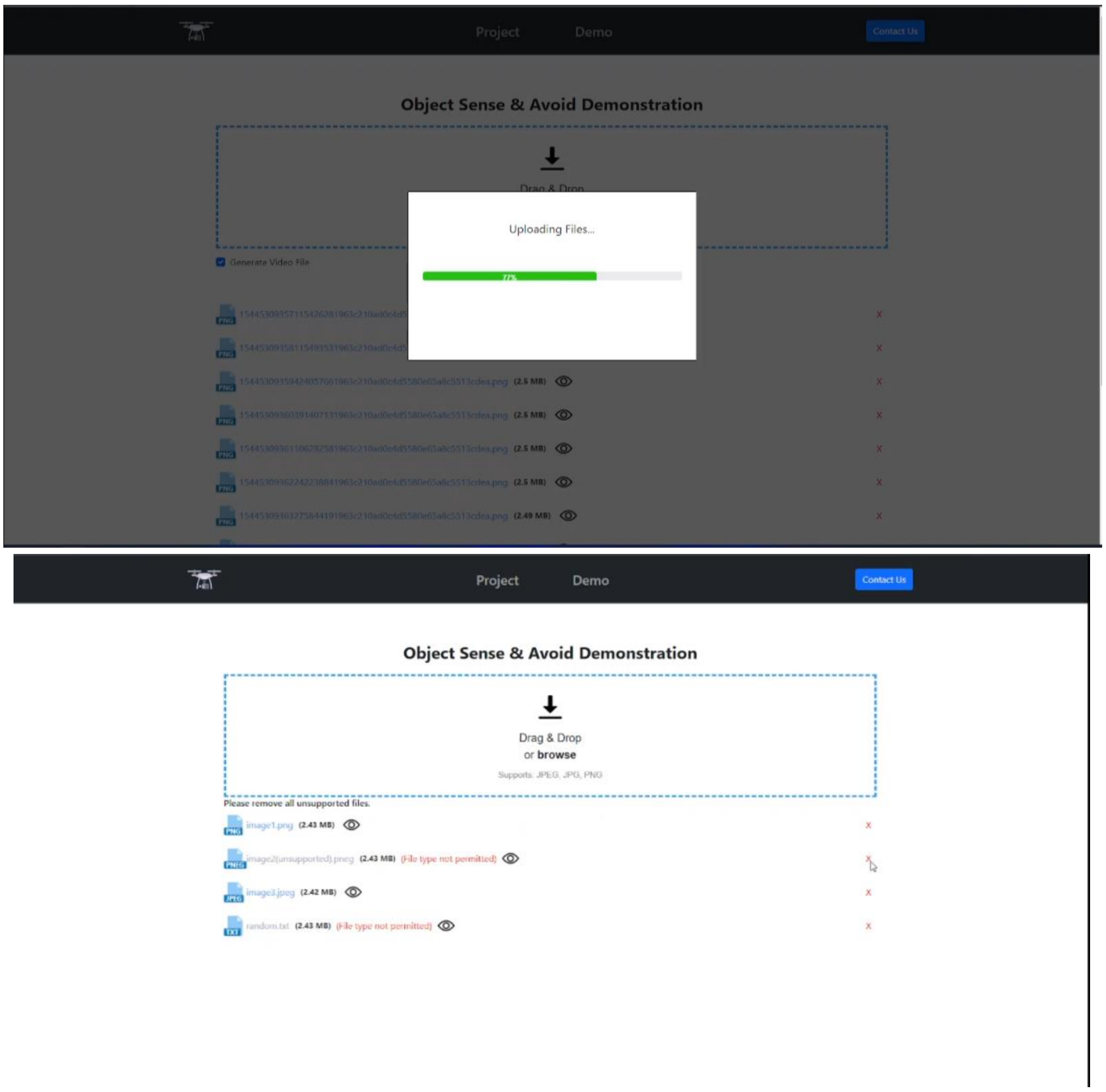
The objective of the project is to take an image or a monocular video feed and to perform image analysis to detect any airborne object in the video frame. The detected object then will be classified and tracked throughout multiple frames, where eventually the model will use various factors and a decision tree. Upon further decision making the model will perform the necessary manoeuvre to create a safer route.

Aims & Objectives: ✓

The project aim is to create a model that will take an image or a monocular video feed, then detect any airborne object that could potentially collide with our object by tracking its motion across multiple frames. Below are the set objectives that I will need to accomplish.

1. Research to find literature on existing drone sense & avoid systems, research on autonomous flights, pretrained classification models, computer vision algorithms and

Figure 5.10 - Home Page (Final Product)



Object Sense & Avoid Demonstration

✓ Generate Video File

Uploading Files...

77%

1544530935711542628 1963c210ad0c4d5

15445309358115493531963c210ad0c4d5

15445309359424057661963c210ad0c4d5580e65a8c5513cdea.png (2.5 MB)

15445309360391407131963c210ad0c4d5580e65a8c5513cdea.png (2.5 MB)

15445309361106292581963c210ad0c4d5580e65a8c5513cdea.png (2.5 MB)

15445309362242238841963c210ad0c4d5580e65a8c5513cdea.png (2.5 MB)

15445309363275844191963c210ad0c4d5580e65a8c5513cdea.png (2.49 MB)

Object Sense & Avoid Demonstration

Please remove all unsupported files.

image1.png (2.43 MB)

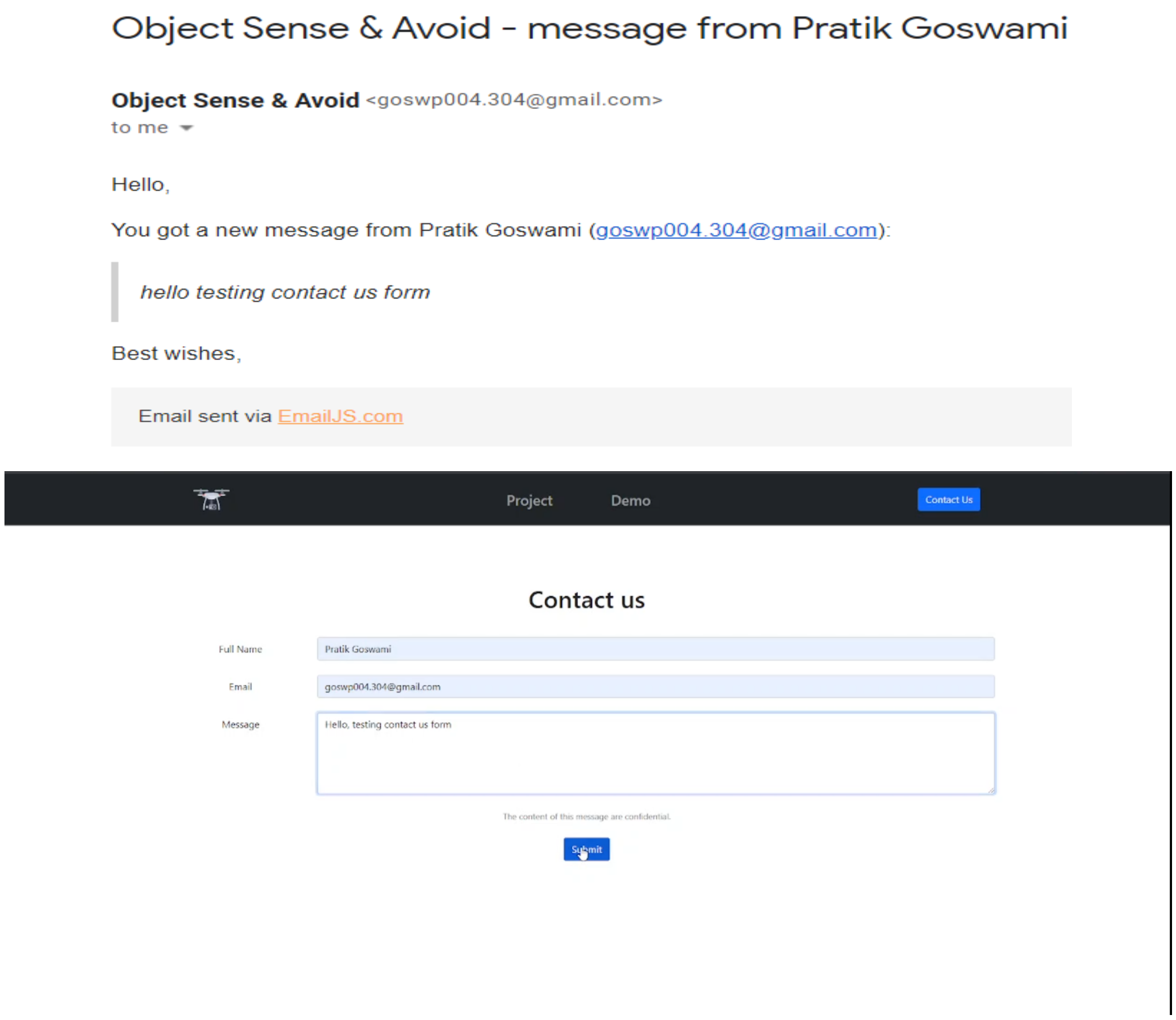
image2(unsupported).pneg (2.43 MB) (File type not permitted)

image3.jpeg (2.42 MB)

random.txt (2.43 MB) (File type not permitted)

Figure 5.11 – Demo Page With Incompatible & Working Files (Final Product)

An external emailing service was integrated, called EmailJS, as this approach avoids the need for developing and hosting a separate email server. Furthermore, it simplifies and reduces project complexity by not having an additional email server to maintain and transitioning to a microservice architecture, where multiple instances could get deployed. The form data is from the contact us page and sent to an API that relays an email to the set receipt's address.



Object Sense & Avoid - message from Pratik Goswami

Object Sense & Avoid <goswp004.304@gmail.com>
to me

Hello,

You got a new message from Pratik Goswami (goswp004.304@gmail.com):

> *hello testing contact us form*

Best wishes,

Email sent via EmailJS.com

Contact us

Full Name	Pratik Goswami
Email	goswp004.304@gmail.com
Message	Hello, testing contact us form

The content of this message are confidential.

Figure 5.12 - Contact Us Page (Final Product) With An Example Email

6 Testing

6.1 Unit Testing

Unit testing is a software development process where individual functionality and components are tested systematically. The purpose of performing unit tests is to validate if each component performs as expected. One of the valuable benefits is that it helps identify bugs at an early stage of the development and avoids costly later changes. Additionally, it assists in identifying any flaws and highlights any required improvements. The project uses black-box testing to test a system with no prior knowledge of its internal working and only focuses on the inputs and outputs. Each test case will have a test id, description, input passed and expected output. The testing is conducted manually, and the output is validated to pass or fail the test. If an error is returned, the test case will be classified as a bug. The project had no human participants testing the user interface; therefore, most of the testing done focuses on the project's functionality.

Table 6.1 – Black Box Testing Results

Test ID	Description	Input Data	Expected Output	Actual Output	PASS OR FAIL
1	Selecting unsupported file type.	".txt" & ".doc" files are selected.	Error message "Remove unsupported Files" and upload button is hidden.	An error message is shown, and no upload button present	PASS
2	Able to remove unsupported files	Select ".txt" & ".doc" files	Clicking on the remove icon should remove the files from the panel.	The file gets successfully removed.	PASS
3	Can view the images from the panel	Select a few ".png" and ". jpg" files	Clicking on the eye icon shows the image.	The image gets rendered and displayed.	PASS
4	Able to select or drag and drop files into the browser.	Drag & drop and select image files onto the browser window	The files should get rendered and updated into the UI	Files successfully get listed and are rendered into UI.	PASS
5	Can generate video files.	Select image files and select generate video option, and upload them.	Should return a zipped file with a video file.	A zip file gets returned with the correct video file.	PASS
6	Can upload the provided files and returns	Select images of supported file types and	An upload modal should show up "uploading" message with a progress bar. A	An upload model shows up with a functioning progress bar. A	PASS

	processed images.	press the upload button.	zipped file should get returned with all processed images.	zip file is returned with all processed images.	
7	Contact us page form sends an email.	Enter a name, email address, a sample message, and submit it.	An email should get sent out with the name in the subject and the message in the body.	An email was received with the correct information.	PASS
8	It is mobile-friendly, the app can scale to different resolutions.	Change the browser window to simulate a phone, tablet and desktop.	The screen should automatically scale appropriately for all devices.	The web app scales between devices perfectly.	PASS

7 Evaluation

In this section, I will be evaluating the model's accuracy and performance using a range of metrics such as mean average precision, recall, accuracy, precision. I will see if the model has any symptoms representing over or underfitting. Due to time constraints, I was only able to train a single model

7.1 Models Performance

Precision is defined as the fraction of true positives among all of the examples which were predicted to be in a specific class. A true positive is a prediction based on an observation that belongs to its actual class. A false positive is a prediction that does not belong to the correct class. A confusion matrix is plotted, a performance measuring tool for machine learning classification models.

The confusion matrix portrays a solid diagonal line which conveys that most classes are being correctly labelled and classified into their appropriate category. The model accurately predicts “aeroplanes” as it correctly predicted 76% of those objects. Similarly, the “helicopter” class has an extremely high value, with an accuracy of 98% of the helicopter images being classified correctly. However, 2% of the helicopters and 24% of aeroplane objects got mislabelled. A possible reason for this could be that the model did not receive a variety of scenarios between training to help distinguish a helicopter from a plane. Additionally, helicopters have distinct shapes than planes or birds, with simple cylindrical bodies.

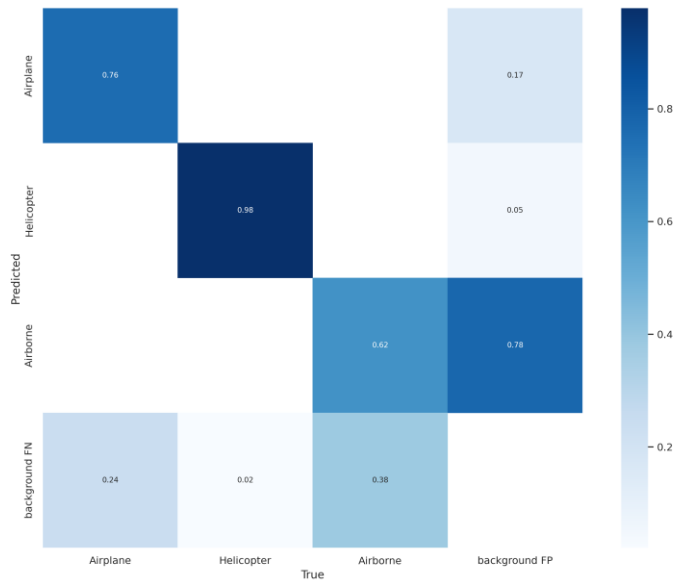


Figure 7.1 - Confusion Matrix Of Final Model

One of the major flaws of this model is that it has a true positive value of only 62% and a high false-negative value of 38%. A false negative value represents a prediction that does not belong to that class when it actually does. On the other hand, a false positive is a prediction that belongs to a class, but in reality, it does not. This conveys that the model struggles to classify airborne objects correctly and has a hard time, further supported by a high false-positive value of 78% compared to false-positive values for other classes.

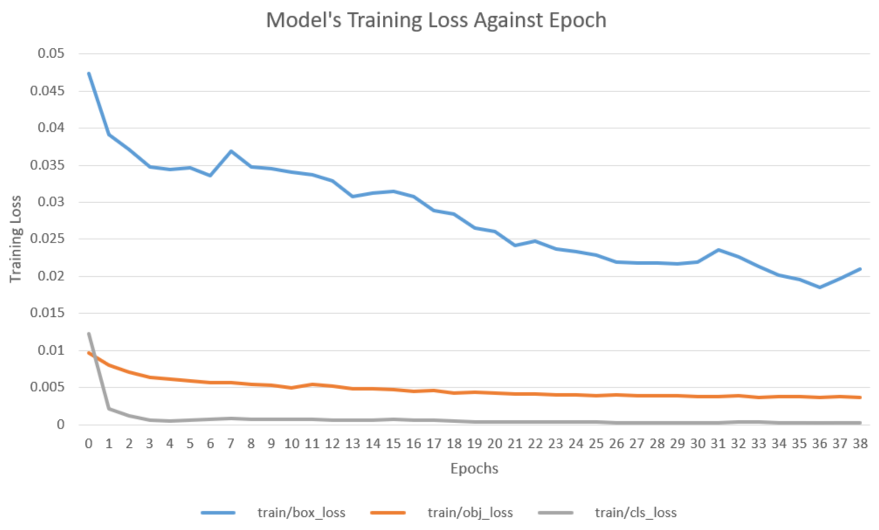


Figure 7.2 - Model's Training loss Against Epochs

Figure 7.2 illustrates a graph for all the training losses against epochs for a single training run. The graph shows significantly small training loss values for both object and class. The model initially started at 0.01 and gradually declined close to zero by the end of this training cycle. The training loss for the bounding box starts at 0.05 and gradually decreases, which implies that the model is still learning as it is trying to minimise the loss. This suggests that a few more epoch cycles could be executed to lower the training loss further; however, the model will eventually begin to overfit the dataset if continued to train further.

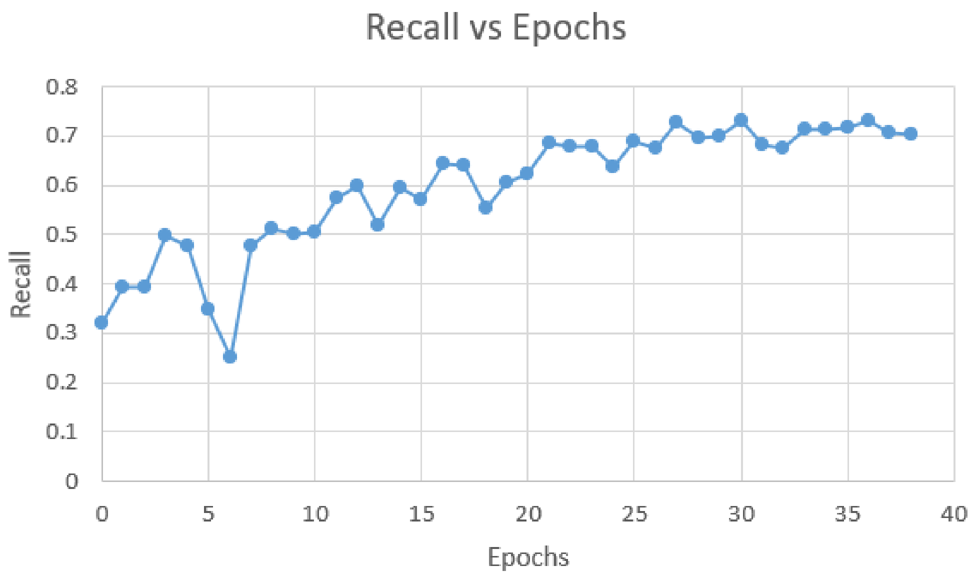


Figure 7.3 – Recall vs Epochs Graph

The recall metric measures the model's ability to detect all the ground-truth objects. An epoch indicates the total number of passes of the whole dataset that an algorithm has made. The line graph in Figure 7.3 conveys a positive correlation where the total number of epochs increases, the recall value also increases. The model initially begins with a recall value of 0.3, which gradually increases with total epochs completed in an oscillating behaviour. An interpolation suggests that increasing epochs will directly increase models recall. A high recall value conveys that the model can detect most ground truth values with almost 80% certainty.

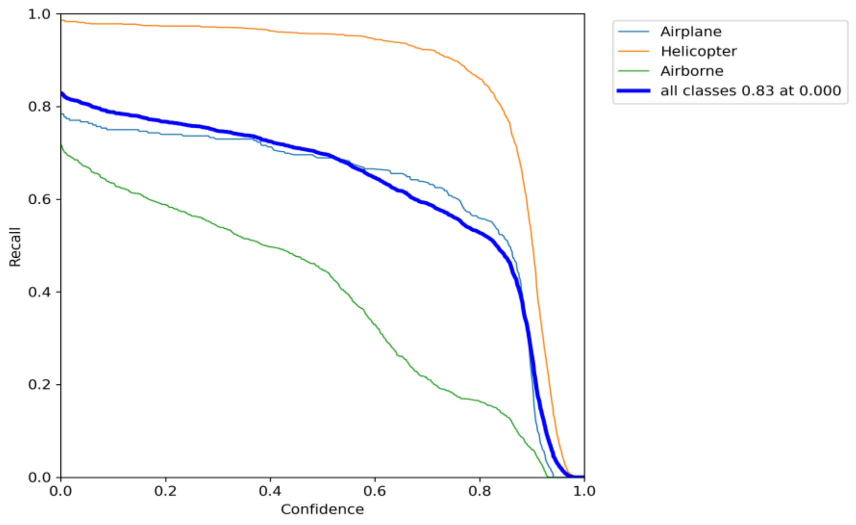


Figure 7.4 - Recall vs Confidence

A confidence score, a number between 0 and 1, represents the likelihood that the model's predictions are correct. A high confidence score conveys that the model can get large proportions of its predictions correct from all ground-truth objects. Figure 7.4 portrays a negative inverse relationship between recall and confidence, where an increase in recalls has a directly inverse effect on confidence. The model aims to obtain a maximum value for each class that gives the optimal value for both variables, ideally as close to the top right corner. The line representative of the helicopter object class is better fitted than the airborne object class, implying more helicopters are identified with more confidence than other classes. Additionally, the confidence score would depend on the threshold, where increasing the threshold too high can result in objects being missed out, leading to more false negatives, lower recall and a high precision value.

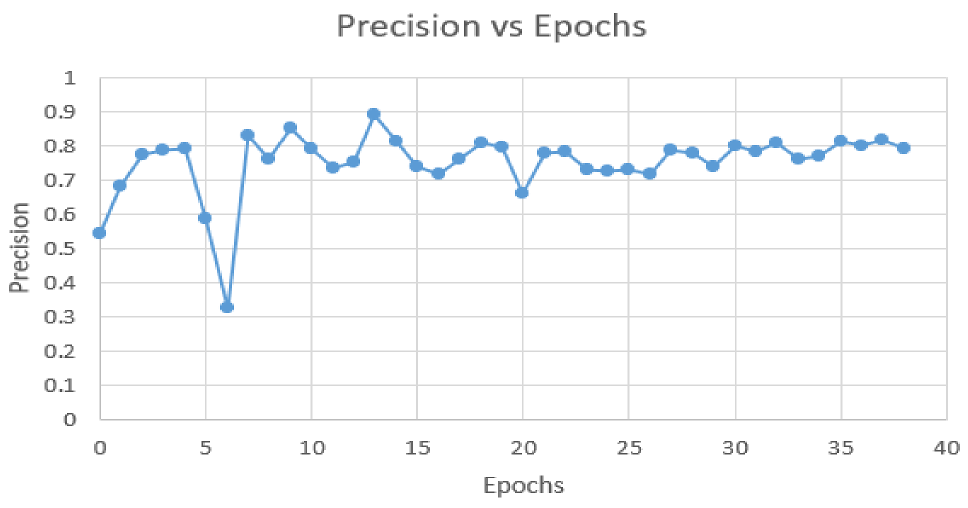


Figure 7.5 - Precision vs Epochs

The precision metric represents the ratio between all true positive predictions over all the predictions made by the model. Furthermore, precision illustrates the degree of exactness in finding all the relevant objects in a frame (Agrawal, 2021). Figure 7.5 illustrates a positive correction between precision and epochs, as the initial model starts with 0.6 precision and slowly increases to 0.8. The model seems to perform well, with interpolations predicting a precision close to 1 when further trained; however, this could lead to overfitting and the model overlearning particular scenarios.

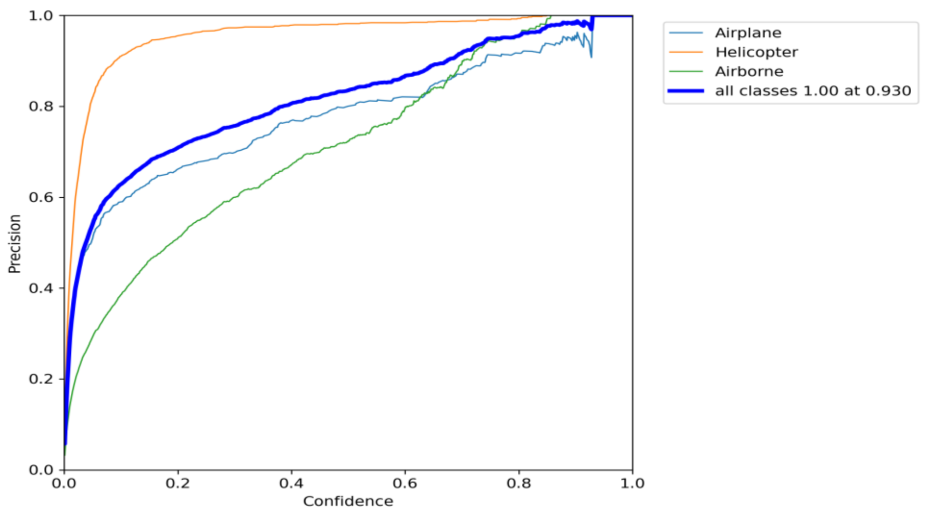


Figure 7.6 - Precision vs Confidence

The graph in Figure 7.6 conveys that the model is overfitting, as it has a perfect precision of 1 (not feasible) at 93% confidence. The precision metric alone is not accurate enough to conclude this; however, it illustrates that model has slowly started memorising the training scenario rather than learning from them. Another possibility could be that this batch did not have enough variation between flights. The model shows that the helicopter class is best fitted compared to others. A better metric for supporting the model's performance could be accuracy, which measures how often the object classifier can correctly predict. The helicopter class has an accuracy of 93.5%, followed by the aeroplane class with 70.9%, and lastly, the airborne class has a drastically lower accuracy of only 43.6%. The confusion matrix (see Figure 7.1) further supports that variation in a training scenario is required with a more significant proportion of airborne class instances.

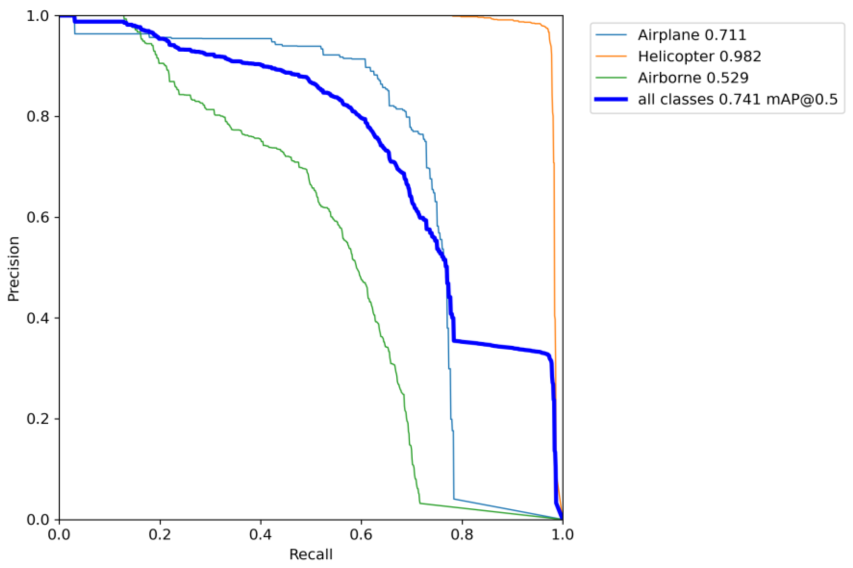


Figure 7.7 - Precision vs Recall Curve

The graph in Figure 7.7 illustrates a negative relationship between precision and recall. A precision-recall (PR) curve highlights the trade-off between precision and recall values at different confidence thresholds. A high area under the curve would symbolise the particular class having high precision and recall values, i.e. the helicopter class with 0.982, implying that the model would make fewer false-positive predictions. Additionally, a good model would have consistent precision and recall values even when the confidence thresholds are changed.

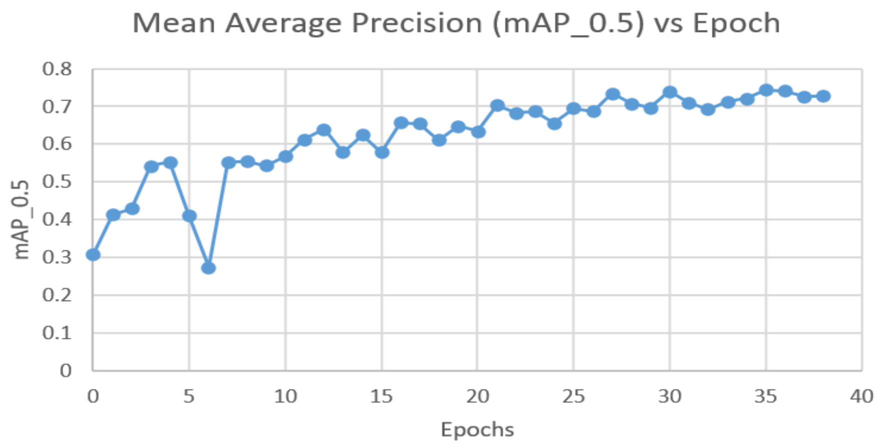


Figure 7.8 - Mean Average Precision (mAP_0.5) vs Epoch

The average precision (AP) represents the area under the precision-recall curve (AUP-RC) calculated at a specific intersection over union (IoU). A mean average precision (mAP) is the average of all AP values for all classes. Figure 7.8 displays a positive correlation between mAP and epoch, where initially model's mAP is 0.3 and ends with 0.75. The growing trend illustrates that each class's area under the curve increases. A larger AUP-RC conveys that ability of the classifier to distinguish between different classes is improving, implying the model is learning and not overfitting.

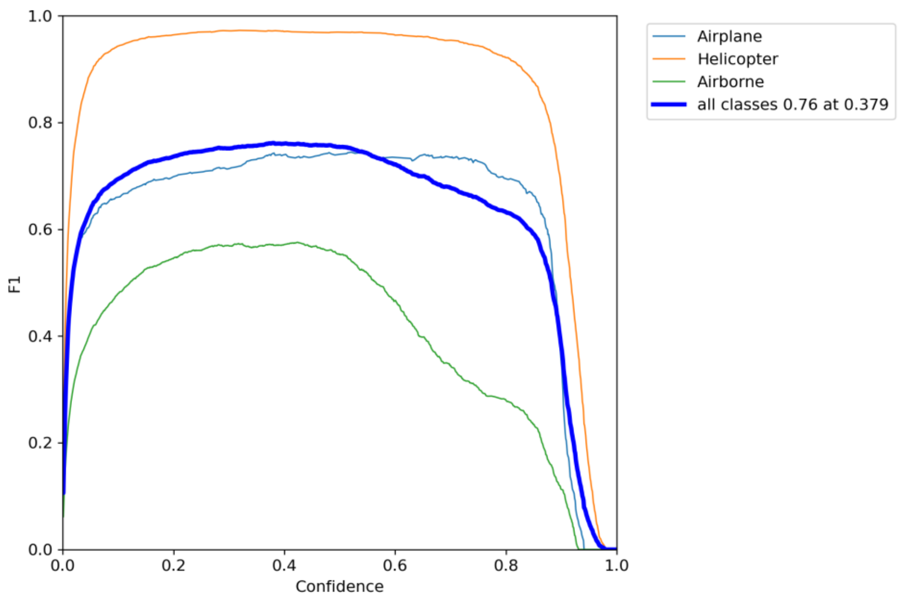
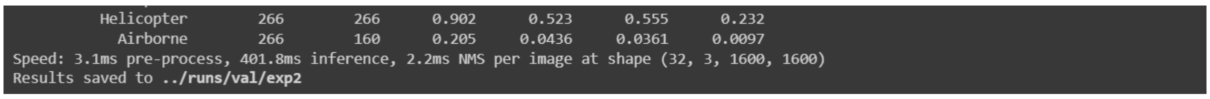


Figure 7.9 - F1 Score vs Confidence Curve

Figure 7.9 looks at the F1 score, a combined metric that measures precision and recall at once using a Harmonic Mean instead of an Arithmetic Mean by pushing extremes (Agrawal, 2021). The graph conveys that the model struggles more with airborne objects than aeroplanes or helicopters, as the curve is much lower, hence a lower F1 score. The average F1 score for all classes is 0.76, expressing that the model performs well but has a low confidence score of 0.38.



```
          Helicopter        266        266      0.902      0.523      0.555      0.232
            Airborne        266        160      0.205     0.0436     0.0361     0.0097
Speed: 3.1ms pre-process, 401.8ms inference, 2.2ms NMS per image at shape (32, 3, 1600, 1600)
Results saved to ../runs/val/exp2
```

Figure 7.10 - Models Inference & Speed Analysis

Lastly, Figure 7.10 shows the time taken for a single image to be analysed. The model takes 3.1ms for pre-processing and has 2.2ms to apply non-maximum suppression (NMS). The YOLO algorithm takes 401.8ms inference an image, meaning the model cannot operate in real-time. However, the tests were performed on a slow GPU (Nvidia Quadro520), and a real-time performance could be achieved on a more powerful GPU. Additionally, the model can be trained on an even lower resolution to limit the amount of data needed to be processed by implementing custom anchor boxes and configuring the hyper-parameter tuning files.

8 Conclusions

This section contains an overall project conclusion, critically evaluates whether or not the set aim and objectives have been accomplished. The project's main aim was to:

"Create a machine learning model that can infer images to detect airborne objects, which would then be classified and tracked to predict a manoeuvre that avoids the obstacles."

The project's main aim was successfully achieved as a model got designed, trained and thoroughly evaluated. The model can infer images (supports a range of file types) to detect airborne objects, classify and track their position relative to the frame. The model predicts manoeuvres that avoid potential obstacles, creating a safer route. The model performs well for most classes and delivers acceptable performance that takes 400 milliseconds per image (on a slow machine), cannot provide real-time response. The proposed approach could solve this problem and attempted to; however, it could not overcome time and hardware limitations.

8.1 Evaluation Against Objectives

8.1.1 Objective 1 (Achieved)

"Conduct research to find literature on existing drone sense & avoidance systems and machine learning algorithms to build the basis for this project."

Section 2 contains a detailed literature review exploring existing sense & avoidance systems, machine learning techniques and algorithms, and related works used to build this project's basis.

8.1.2 Objective 2 (Achieved)

"Research potential methodologies to use and create a plan based on the methodology that, when followed, obtains project goals."

Section 3 details research conducted comparing different methodologies and their pros & cons. The methodology was extensively followed and applied throughout the design, implementation and testing stages (see Sections 4, 5 & 6). A plan was created, and appropriate technologies (GitHub & Weights and Biases) were applied to monitor the project.

8.1.3 Objective 3 (Achieved)

"Source publicly available datasets on airborne objects and gain approval by Brunel research ethics online (BREO) where necessary."

A suitable dataset was acquired; however, finding it was challenging and required weeks of work; see Sections 4 & 5 for a detailed overview. Section 4 details the integrated data pipeline to handle the image pre-processing. Ethics approval was obtained; refer to Appendix B.

8.1.4 Objective 4 (Achieved)

"Design, train and implement a machine learning inference model that fulfils the project's aim."

Refer to Sections 4, 5, & 6 containing a detailed run-through of how the model was built. The model performs well for most classes; however, it struggles with small objects. A potential improvement is to continue training on the remaining dataset, as the model was trained on less than 1% of the dataset (consists of 4+ million images).

8.1.5 Objective 5 (Achieved)

"Design and create an interface to allow users to interact and test its interface on functionality."

Chapters 4 to 6 illustrate the design, implementation, and functionality testing of the user interface. Furthermore, it details how the RAD methodology was applied to develop the interface. Lastly, chapter 6 details the functionality tests performed.

8.1.6 Objective 6 (Achieved)

"Evaluate the model's performance against evaluation metrics (such as detection accuracy, rate of false positives and overall classification accuracy)."

Chapter 7 specifics the rigorous statistical and manual evaluation process to evaluate the model's performance.

8.1.7 Objective 7 (Achieved)

"Evaluate the project on what went well, what could be improved and potential future works."

The dissertation communicates the project's idea and approach, detailing the proposed solutions' implementation and extensive evaluation process assessing the model's performance. Section 9 concludes the final project evaluation, where all project objectives were achieved.

8.2 Limitations

In this project, there were several hardware and software limitations. Firstly, the project faced many setbacks, and difficulties as working with large datasets (upwards of 14 terabytes) required the development of pipelines to retrieve, handle and store the dataset. The training environment also had to be migrated several times from a remote university machine to Google collab, requiring modification of the pipeline.

Google Collab has a GPU usage cap that allows 4 to 5 hours of training time. Additionally, the flawed development environment would at random terminate the connection, resulting in training data loss. A captcha had to be solved every 10 minutes; otherwise, the session would terminate; meaning a person had to be present at all times while training occurred (not ideal). Lastly, due to storage limitations (50GB), training could only be performed in small batches.

At such an early stage, challenges created a domino effect on other parts of the development cycle and made time a precious commodity. For a future project of a similar scale, finding a suitable development environment would be one of the priorities. Additionally, the model could be trained at a lower resolution, requiring modifying the network's backbone structure and refining the hyper-tuning parameters. A model trained at a lower resolution also would drastically reduce the inference time and improve performance.

8.3 Future Work

The project has been designed and implemented as individual components; therefore, specific components can be swapped out for different ones and further improved. Some of the improvements that can be implemented are:

- Integrate the model into a drone and evaluate its performance in real-world scenarios. Furthermore, implementing a federated learning algorithm enables machine learning models to obtain experience from different scenarios without sharing training data.

- Train a separate frame alignment model to compensate for the drone's movement or use a gyro or a camera stabiliser.

- Implement a temporal shift module (TSM) to help improve video understanding (Lin MIT, Gan and Han MIT, 2018). A TSM could improve the model's performance by 20-50x and deliver real-time feedback. Additionally, multiple models can be ensembled to improve accuracy, as the current model is very susceptible to false positives (FP).

- A separate regression head could be integrated to predict the detected object's distance.

- Train two different models: a fast model; that inference on the whole image at full resolution and a heavy-duty model that predicts smaller cropped regions at a lower resolution.

- Use multiple GPUs to speed up the model training process and finish training on the remaining dataset until the desired performance is achieved.

In the future, other academics could develop the project to implement 3D spatial awareness using LIDAR and camera sensors to track the position of an object in space rather than in a still image frame.

In conclusion, the project was a success; I researched existing solutions, pre-trained models, machine learning algorithms, usable datasets, and methodology and conducted a thorough literature review to build the foundation for the project. I was able to justify and apply a suitable methodology throughout the project. I was able to obtain ethical approval where necessary. I successfully designed, implemented, and evaluated a machine learning model. Additionally, I was able to iteratively design, implement a user interface that was thoroughly tested on its functionality. Lastly, I developed a fully working web application that integrated a custom-built model that can infer images to detect airborne objects, perform object classification, track them and predict a manoeuvre that avoids the obstacles.

Reference

Agrawal, S. (2021) *Evaluation Metrics For Classification Model | Classification Model Metrics*. Available at: https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/ (Accessed: March 24, 2022).

AIcrowd (2021) *Airborne Object Tracking Challenge*. Available at: https://www.aicrowd.com/challenges/airborne-object-tracking-challenge (Accessed: March 10, 2022).

Amazon (2013) *Amazon Prime Air - future delivery system from Amazon*. Available at: https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011 (Accessed: March 20, 2022).

Amidi, A. and Amidi, S. (2018) *CS 230 - Convolutional Neural Networks Cheatsheet*. Available at: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks (Accessed: March 21, 2022).

Andreas (2017) *35 Important Pros & Cons Of Drones - E&C*. Available at: https://environmental-conscience.com/drones-pros-cons/ (Accessed: March 15, 2022).

Bandyopadhyay, H. (2022) *YOLO: Real-Time Object Detection Explained*. Available at: https://www.v7labs.com/blog/yolo-object-detection#what-is-yolo (Accessed: March 22, 2022).

Bhatia, R. (2018) *When not to use Neural Networks | Medium Data Driven Investor*. Available at: https://medium.datadriveninvestor.com/when-not-to-use-neural-networks-89fb50622429 (Accessed: March 21, 2022).

Bray Solutions (2017) *Drone Delivery – The Future of Logistics*. Available at: https://www.braysolutions.com/drone-delivery/ (Accessed: October 25, 2021).

Ciobanu, E. (2021) *Obstacle Avoidance in Drones Droneblog*. Available at: https://www.droneblog.com/dji-drone-obstacle-avoidance/ (Accessed: March 21, 2022).

DJI Enterprise (2021) *The Use of Drones in Agriculture Today*. Available at: https://enterprise-insights.dji.com/blog/drones-in-agriculture (Accessed: March 20, 2022).

Drone Tech Planet (2019) *Does Weight Affect a Drone's Battery Life? – Drone Tech Planet*. Available at: https://www.dronetechplanet.com/does-weight-affect-a-drones-battery-life/ (Accessed: March 19, 2022).

Fly Ability (2020) *Commercial Drones: A Guide to the Top Models on the Market*. Available at: https://www.flyability.com/commercial-drones (Accessed: October 25, 2021).

Girshick, R. *et al.* (2013) "Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5)." Available at: https://arxiv.org/pdf/1311.2524.pdf (Accessed: March 22, 2022).

He, K. *et al.* (2016) "Deep Residual Learning for Image Recognition." Available at: https://arxiv.org/pdf/1512.03385.pdf (Accessed: March 22, 2022).

Intelligence Insider (2021) *Intelligence Insider (2021) Drone Industry Analysis 2021: Market Trends & Growth Forecasts*. Available at: https://www.businessinsider.com/drone-industry-analysis-market-trends-growthforecasts?r=US&IR=T (Accessed: October 25, 2021).

Lin MIT, J., Gan, C. and Han MIT, S. (2018) "TSM: Temporal Shift Module for Efficient Video Understanding." Available at: https://github. (Accessed: March 25, 2022).

Mahanta, J. (2017) *Introduction to Neural Networks, Advantages and Applications | Towards Data Science*. Available at: https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207 (Accessed: March 20, 2022).

McCullum, N. (2020) *Deep Learning Neural Networks Explained in Plain English | Free Code Camp*. Available at: https://www.freecodecamp.org/news/deep-learning-neural-networks-explained-in-plain-english/ (Accessed: March 21, 2022).

ODSC - Open Data Science (2018) *Overview of the YOLO Object Detection Algorithm | by ODSC - Open Data Science | Medium*. Available at: https://odsc.medium.com/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0 (Accessed: March 21, 2022).

OS Systems (2020) *Software Development Methodologies: Comparison & Differences*. Available at: https://os-system.com/blog/top-software-development-methodologies-comparison-differences-pros-and-cons/ (Accessed: March 25, 2022).

raghunandepu (2019) *Understanding and implementation of Residual Networks(ResNets) | Medium*. Available at: https://medium.com/analytics-vidhya/understanding-and-implementation-of-residual-networks-resnets-b80f9a507b9c (Accessed: March 23, 2022).

Redmon, J. *et al.* (2015) “You Only Look Once: Unified, Real-Time Object Detection.” Available at: https://arxiv.org/abs/1506.02640 (Accessed: March 22, 2022).

Tesla (2016) *Autopilot & Collision System | Tesla*. Available at: https://www.tesla.com/autopilot (Accessed: March 17, 2022).

Tesla (2018) *Tesla Vehicle Safety Report | Tesla*. Available at: https://www.tesla.com/VehicleSafetyReport (Accessed: March 22, 2022).

Zhou, X., Koltun, V. and Krähenbühl, P. (2020) “Tracking Objects as Points.” Available at: https://arxiv.org/pdf/2004.01177.pdf (Accessed: March 22, 2022).

Appendix A Personal Reflection

A.1 Reflection on Project

The project was an overall success, as evident by successfully achieving all the objectives and aims (refer to Section 8). Conducting a thorough literature review was a good starting point as it provided a strong project basis. Applying the RAD methodology throughout the project provided an iterative approach that allowed a performant model to be implemented. I feel that the model does a great job of detecting, classifying and tracking airborne objects; however, it lacks in training hours needed to achieve better accuracy and precision. I believe that the project is an excellent addition, and with future improvements, I can see it being integrated into a drone.

The use of Google Collab was an excellent choice at that time; as the project progressed, the environment's limitations quickly became a bottleneck. I would highly prioritise finding a suitable development environment when working with big data in the future. Weights & Biases and GitHub – with its Kanban board feature, were comprehensively used to track, manage and for version control.

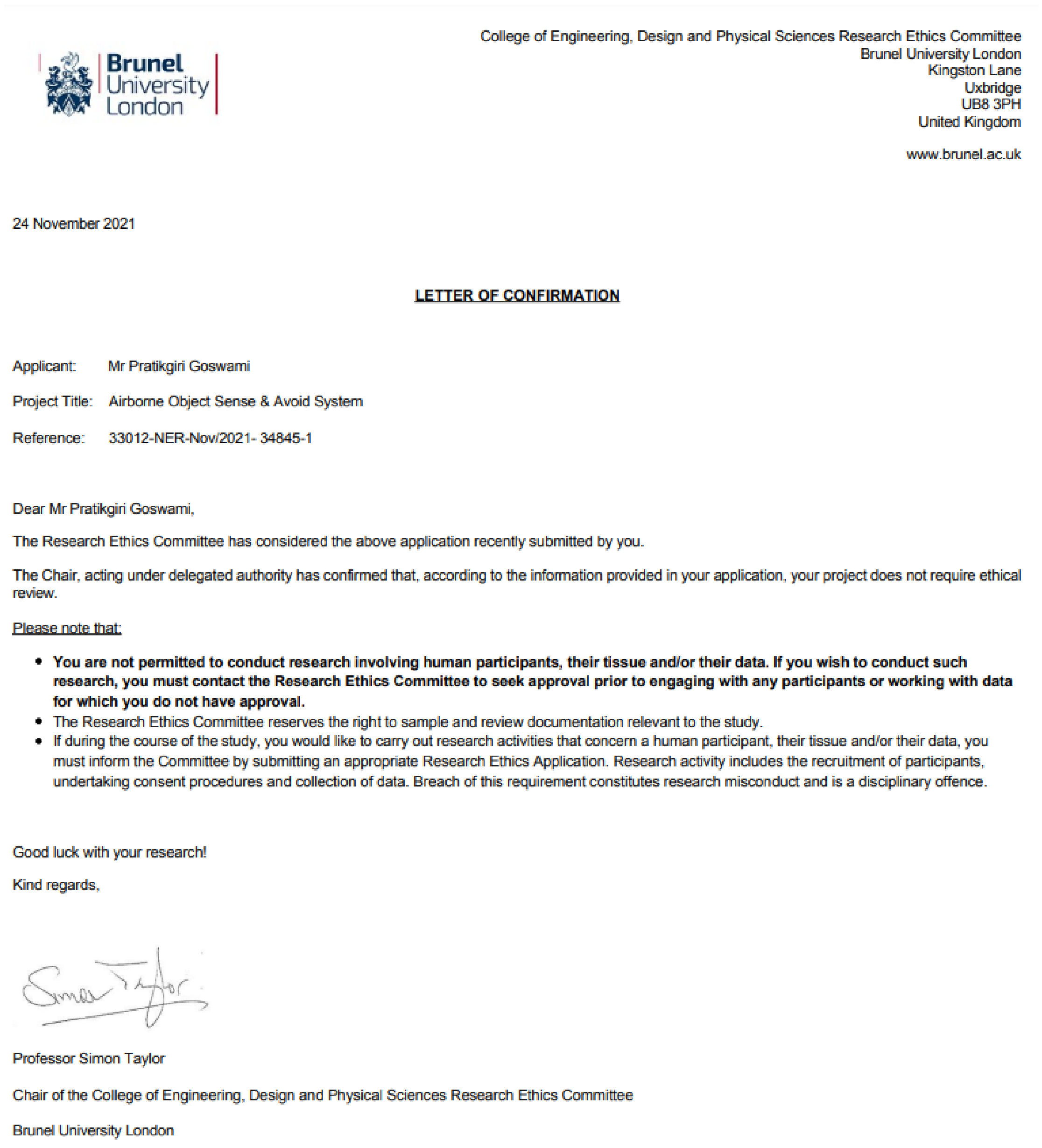
In conclusion, the project was an accomplishment, and it provides an excellent foundation for other academics to improve on further. I can further improve the model's accuracy and performance to achieve a real-time response with more time.

A.2 Personal Reflection

I am delighted with how the project turned out. I believe this to be a key milestone in my software engineering career. Learning and developing new transferable Python, Flask, React, and data analytic skills was an enriching experience. The project provided valuable insight into how models are designed and implemented. It has strengthened my knowledge of machine learning and artificial intelligence in general. The project has inspired me to build other personal machine learning projects.

In conclusion, if I had to redo the entire project, I would personally start with allocating more time for the actual implementation and training of the model. Once again, I would prioritise obtaining a suitable environment for the project. Lastly, working on this project has provided me with an irreplaceable learning experience in machine learning and AI in general.

Appendix B Ethics Approval Confirmation Letter



College of Engineering, Design and Physical Sciences Research Ethics Committee
Brunel University London
Kingston Lane
Uxbridge
UB8 3PH
United Kingdom

www.brunel.ac.uk

24 November 2021

LETTER OF CONFIRMATION

Applicant:	Mr Pratikgiri Goswami
Project Title:	Airborne Object Sense & Avoid System
Reference:	33012-NER-Nov/2021- 34845-1

Dear Mr Pratikgiri Goswami,

The Research Ethics Committee has considered the above application recently submitted by you.

The Chair, acting under delegated authority has confirmed that, according to the information provided in your application, your project does not require ethical review.

Please note that:

- **You are not permitted to conduct research involving human participants, their tissue and/or their data. If you wish to conduct such research, you must contact the Research Ethics Committee to seek approval prior to engaging with any participants or working with data for which you do not have approval.**
- The Research Ethics Committee reserves the right to sample and review documentation relevant to the study.
- If during the course of the study, you would like to carry out research activities that concern a human participant, their tissue and/or their data, you must inform the Committee by submitting an appropriate Research Ethics Application. Research activity includes the recruitment of participants, undertaking consent procedures and collection of data. Breach of this requirement constitutes research misconduct and is a disciplinary offence.

Good luck with your research!

Kind regards,

[illegible]

Professor Simon Taylor

Chair of the College of Engineering, Design and Physical Sciences Research Ethics Committee

Brunel University London

Dear Mr Pratikgiri Goswami

Thank you for your application.

On the basis of the information you have provided, your project has been confirmed to not require research ethics approval.
You are free to begin your research.

Please ensure you have considered the following before commencing your research:

- Brunel University Research Integrity Code
- Brunel University Research Data Management Policy
- Brunel University Open Access Policy
- Provisions of the Data Protection Act 1998 and the University Data Protection Policy (further advice is available from the Information Access Officer by emailing data-protection@brunel.ac.uk)
- University Health and Safety practice and procedures.

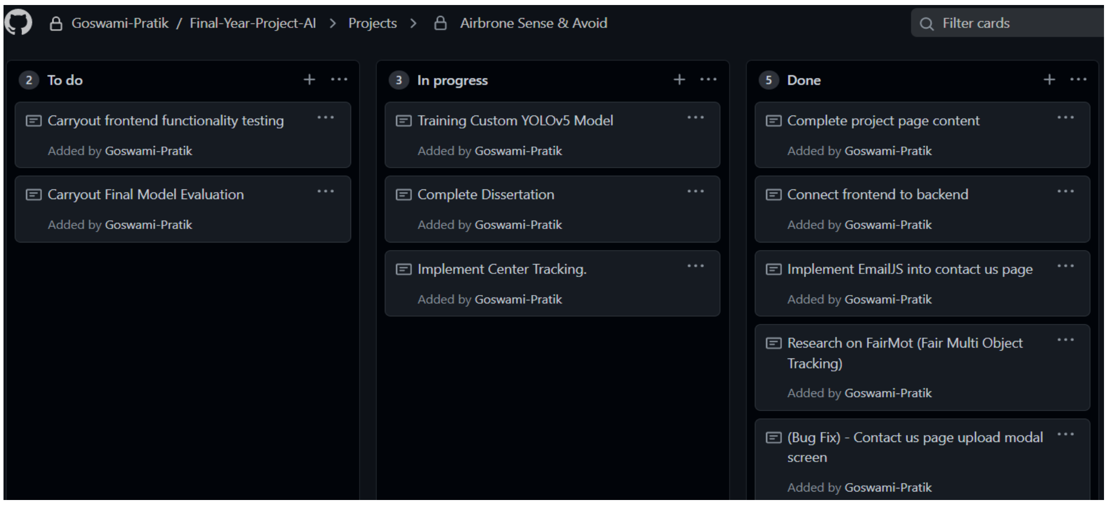
Kind regards,

Professor Simon Taylor

Chair, College of Engineering, Design and Physical Sciences Research Ethics Committee

Appendix C Project Management

C.1 GitHub - Automated Kanban Board



Goswami-Pratik / Final-Year-Project-AI > Projects > Airbrone Sense & Avoid

Filter cards

2 To do

- Carryout frontend functionality testing — Added by Goswami-Pratik
- Carryout Final Model Evaluation — Added by Goswami-Pratik

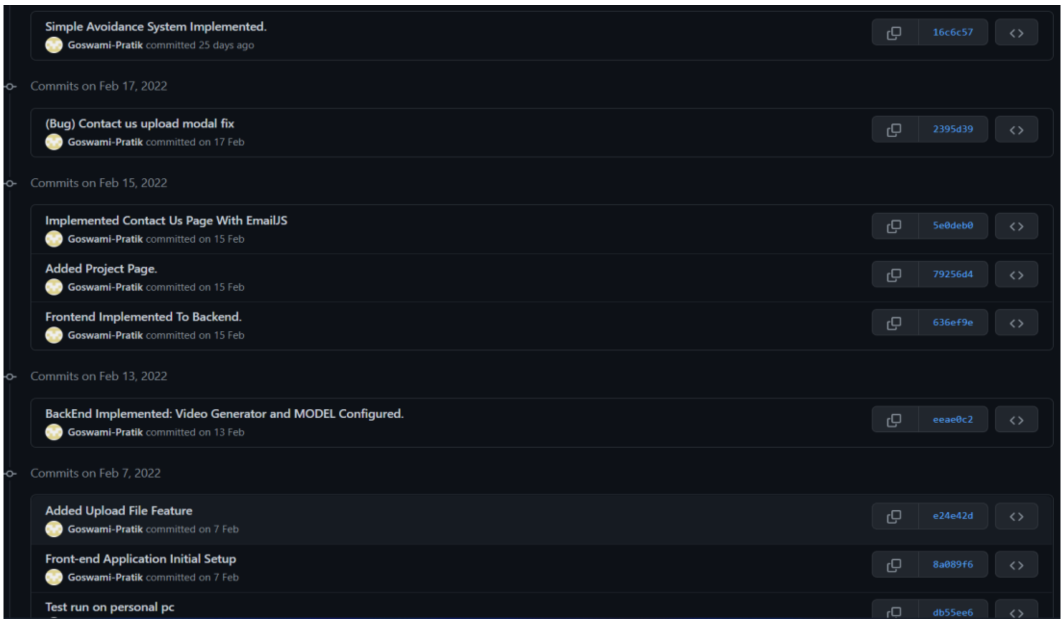
3 In progress

- Training Custom YOLOv5 Model — Added by Goswami-Pratik
- Complete Dissertation — Added by Goswami-Pratik
- Implement Center Tracking. — Added by Goswami-Pratik

5 Done

- Complete project page content — Added by Goswami-Pratik
- Connect frontend to backend — Added by Goswami-Pratik
- Implement EmailJS into contact us page — Added by Goswami-Pratik
- Research on FairMot (Fair Multi Object Tracking) — Added by Goswami-Pratik
- (Bug Fix) - Contact us page upload modal screen — Added by Goswami-Pratik

C.2 GitHub – Version Control (Commits)



Simple Avoidance System Implemented.
Goswami-Pratik committed 25 days ago — 16c6c57

Commits on Feb 17, 2022

(Bug) Contact us upload modal fix
Goswami-Pratik committed on 17 Feb — 2395d39

Commits on Feb 15, 2022

Implemented Contact Us Page With EmailJS
Goswami-Pratik committed on 15 Feb — 5e0deb0

Added Project Page.
Goswami-Pratik committed on 15 Feb — 79256d4

Frontend Implemented To Backend.
Goswami-Pratik committed on 15 Feb — 636ef9e

Commits on Feb 13, 2022

BackEnd Implemented: Video Generator and MODEL Configured.
Goswami-Pratik committed on 13 Feb — eeae0c2

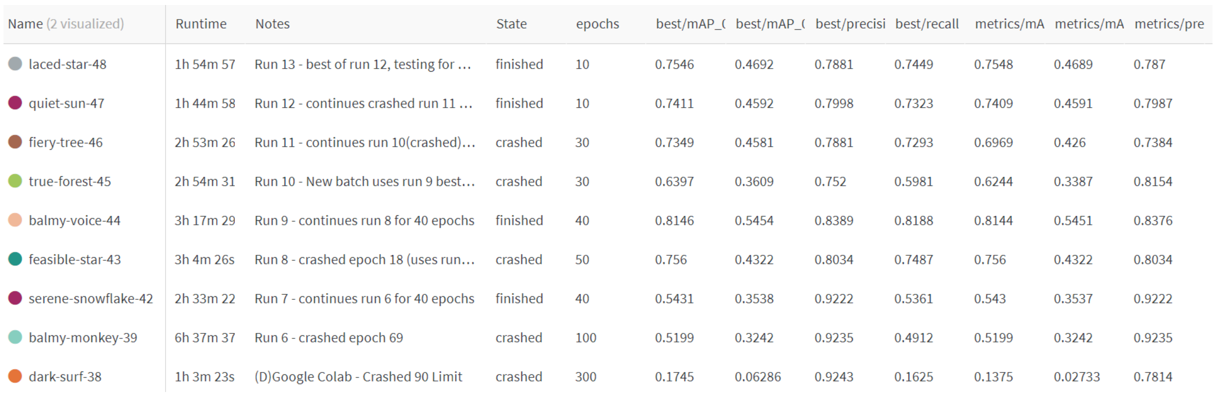
Commits on Feb 7, 2022

Added Upload File Feature
Goswami-Pratik committed on 7 Feb — e24e42d

Front-end Application Initial Setup
Goswami-Pratik committed on 7 Feb — 8a089f6

Test run on personal pc — db55ee6

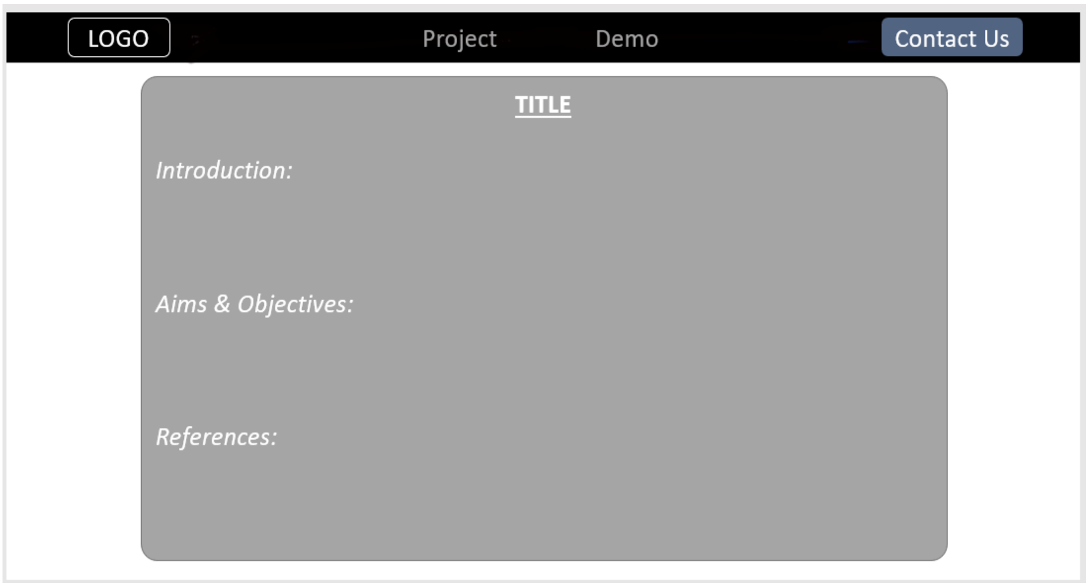
C.3 Weights & Biases - Model Version Control



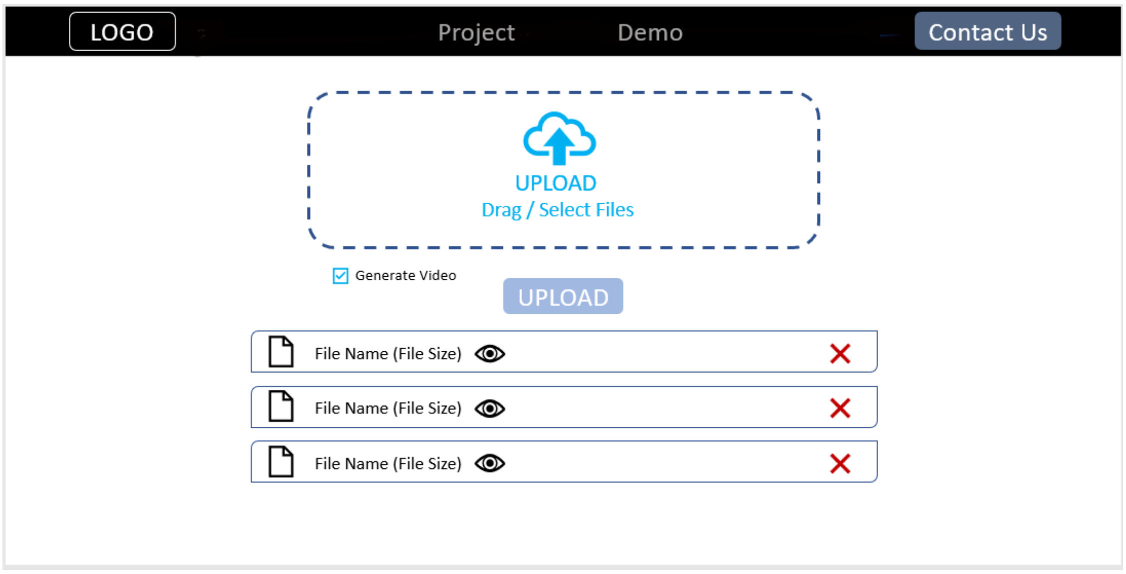
Name (2 visualized)	Runtime	Notes	State	epochs	best/mAP_(	best/mAP_(	best/precisi	best/recall	metrics/mA	metrics/mA	metrics/pre
● laced-star-48	1h 54m 57	Run 13 - best of run 12, testing for …	finished	10	0.7546	0.4692	0.7881	0.7449	0.7548	0.4689	0.787
● quiet-sun-47	1h 44m 58	Run 12 - continues crashed run 11 …	finished	10	0.7411	0.4592	0.7998	0.7323	0.7409	0.4591	0.7987
● fiery-tree-46	2h 53m 26	Run 11 - continues run 10(crashed)…	crashed	30	0.7349	0.4581	0.7881	0.7293	0.6969	0.426	0.7384
● true-forest-45	2h 54m 31	Run 10 - New batch uses run 9 best…	crashed	30	0.6397	0.3609	0.752	0.5981	0.6244	0.3387	0.8154
● balmy-voice-44	3h 17m 29	Run 9 - continues run 8 for 40 epochs	finished	40	0.8146	0.5454	0.8389	0.8188	0.8144	0.5451	0.8376
● feasible-star-43	3h 4m 26s	Run 8 - crashed epoch 18 (uses run…	crashed	50	0.756	0.4322	0.8034	0.7487	0.756	0.4322	0.8034
● serene-snowflake-42	2h 33m 22	Run 7 - continues run 6 for 40 epochs	finished	40	0.5431	0.3538	0.9222	0.5361	0.543	0.3537	0.9222
● balmy-monkey-39	6h 37m 37	Run 6 - crashed epoch 69	crashed	100	0.5199	0.3242	0.9235	0.4912	0.5199	0.3242	0.9235
● dark-surf-38	1h 3m 23s	(D)Google Colab - Crashed 90 Limit	crashed	300	0.1745	0.06286	0.9243	0.1625	0.1375	0.02733	0.7814

Appendix D High-Fidelity Prototypes

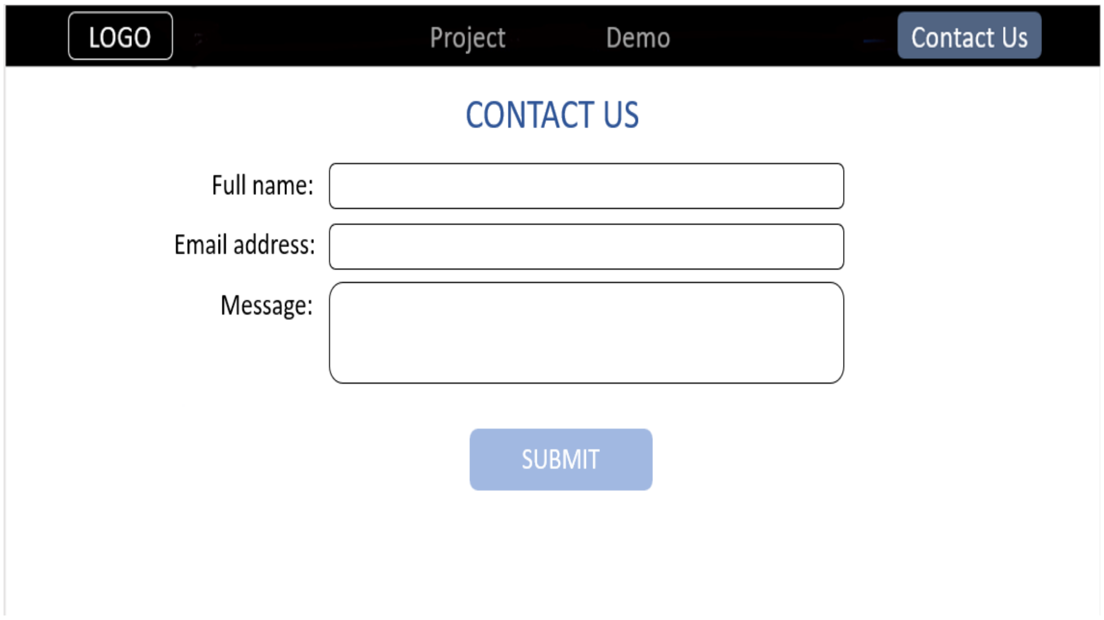
D.1 Home Page



D.2 Demo Page



D.3 Contact Us Page


LOGO
Project
Demo
Contact Us
CONTACT US
Full name:
Email address:
Message:
SUBMIT