Analysis of Low-Cost Agricultural Drone with Machine Learning Object Detection Effectiveness

a project presented to

The Faculty of the Department of Aerospace Engineering

San Jose State University

in partial fulfillment of the requirements for the degree

Master of Science in Aerospace Engineering

by

Zhijie Chen

approved by

Dr. Periklis Papadopoulos

Faculty Advisor

ABSTRACT

Analysis of Low-Cost Agricultural Drone with Machine Learning Object Detection Effectiveness

by Zhijie Chen

With advancements of drone and machine learning technologies, drones have evolved to have more applications in different industries. One of the lesser explored industries for use of drones is agriculture. Due to the high investment cost of UAV systems from larger companies, many localized farmers may not hold value for an advanced drone system from large drone manufacturers to be incorporated into the normal workflow. This report aims to analyze the effectiveness of a lower cost drone model and open-source machine learning based object detection to determine if there is viability in more financially friendly agricultural drones.

Acknowledgements

Firstly, I would like to thank Dr. Periklis Papadopoulos for his guidance in working on this project as an advisor and as an invaluable professor during my time studying at San Jose State University. In addition, I would like to thank my fellow classmate and friend, Alberto Rodriguez, for support and experimental insight through out this project.

Table of Contents

List of Figures

1. Introduction

1.1 Motivation

While UAV companies, such as DJI, have made strides in creating agricultural drones, UAVs have not had a major impact on the agricultural industry. Local farmers have still not been introduced to the technology into their workflows. One of the major factors in this is cost as many of the commercial options are expensive. This project aims to create a backbone of a low-cost drone that can be trained and adopted to target crops that interested farmers can use and act as a gateway to increase the use of UAVs in agriculture.

1.2 Literature Review

The use of drones and unmanned aerial vehicles (UAVs) has been explored in various industries as technological advancements have allowed for UAVs to broaden the viability of use. Some of the applications of drones include, but are not limited to, search/rescue, surveillance, law enforcement, photography/film, construction, delivery, firefighting, real estate, and agriculture. This literature review explores the documented history of the application and use of UAVs in the agricultural industry. Additionally, it will illustrate the required equipment to generate a better understanding of the realistic parameters of analysis for this project.

When analyzing the usefulness of drones within agriculture, the potential use cases of drones must be first identified. In an article by S. Ahirwar for Anand Agricultural University, the use of agricultural drones can be identified into two main categories, analysis and task performance [1]. The analysis applications include providing data on crops that can be used to improve crops' overall health and condition. This example presented in the article uses 3D mapping for soil analysis which can provide information about irrigation and nitrogen-level management in addition to crop planning. Thermal sensors can also be used to identify potential dry areas on the field. Task performance applications include planting and crop spraying. Planting crops with a drone tends to involve ejecting seeds from the drone; this use case can reduce the total labor needed in cases of large fields. Crop-spraying uses drones to release liquid-based chemicals that can allow for the task to be performed up to five times faster than from the ground [1].

Focusing on the analysis function of drones, imaging can be a useful tool as a basis for analysis. According to an article by Yoshio Inoue, drones commonly use three different forms of imaging: multi-spectral, thermal, and visible video [2]. The spectral analysis aspect of the imaging includes the inspection of the reflectance of the captured imagining based on wavebands presented; the values can be compared with vegetarian indices to determine the overall condition

of the crop/field. In thermal analysis, thermal remote sensing can display data related to possible diseases based on the leaf temperature of the crop. In addition to video imaging for visual inspections, predictive algorithms, and conversion models can be used to generate realistic analyses of the state of crops. Shown below is a system flow chart of the specific imaging drone from Inoue's article.

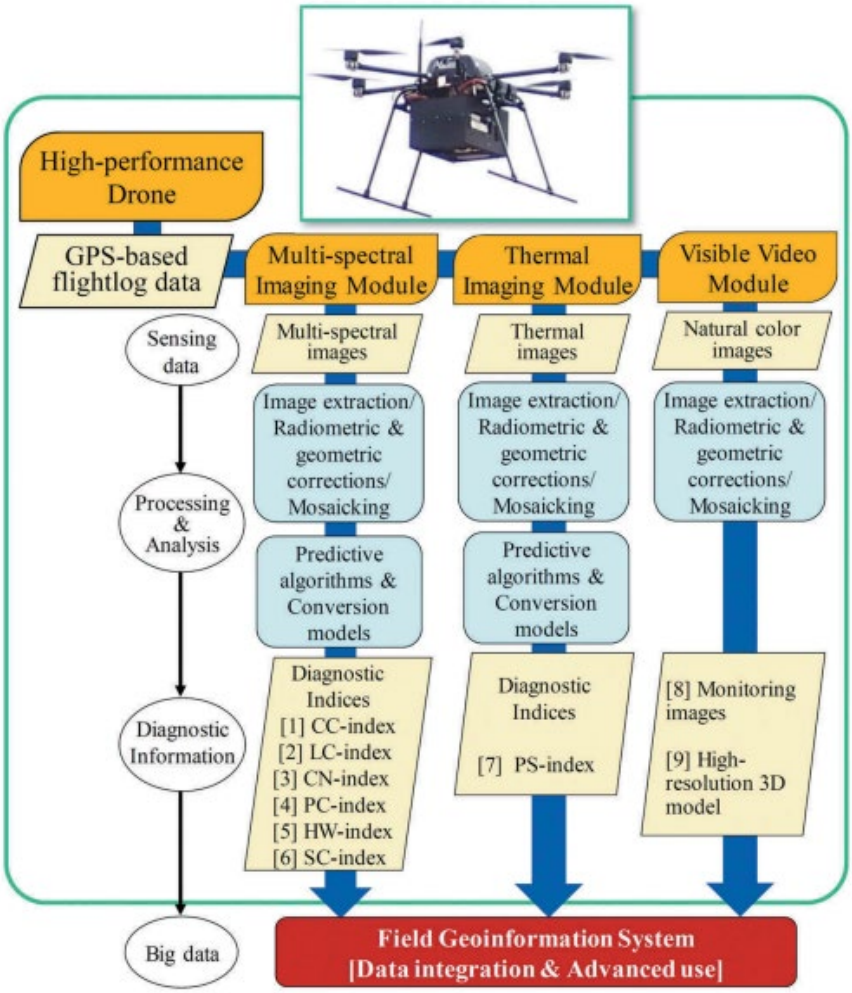


Figure 1.1: Example system operation flow chart of agricultural imagining drone [2]

While imagining provides a basis for the analysis of agricultural drones, not all parameters of crop health can be easily identified through imaging. Another form of data that can be gathered by a drone is the composition of compounds in the air, such as ammonia. A research article led by Theerapat Pobkrut explored the viability of using electronic noses, chemical sensors that can identify patterns of airborne odors, for UAVs. The main compounds focused on in the experiment were ammonia and toluene, but this model can be adapted to detect other airborne compounds. The experiment was performed in two different environments. The first experiment was performed in a closed-room environment with distinct scenarios of control and the presence of the respective compounds. The second environment was composed of an open-air simulation with the same scenarios as the closed-room environment. While the date for the closed-room experiment was more defined in its detection than the open-air environment, the sensors and drone configuration were able to discern the presence of ammonia and toluene in

both environments [3]. The sensors used in this experiment return data in the form of resistance changes from the sensors. The sensor response from six different sensors for both ammonia and toluene presented in Pobkrut is shown below.

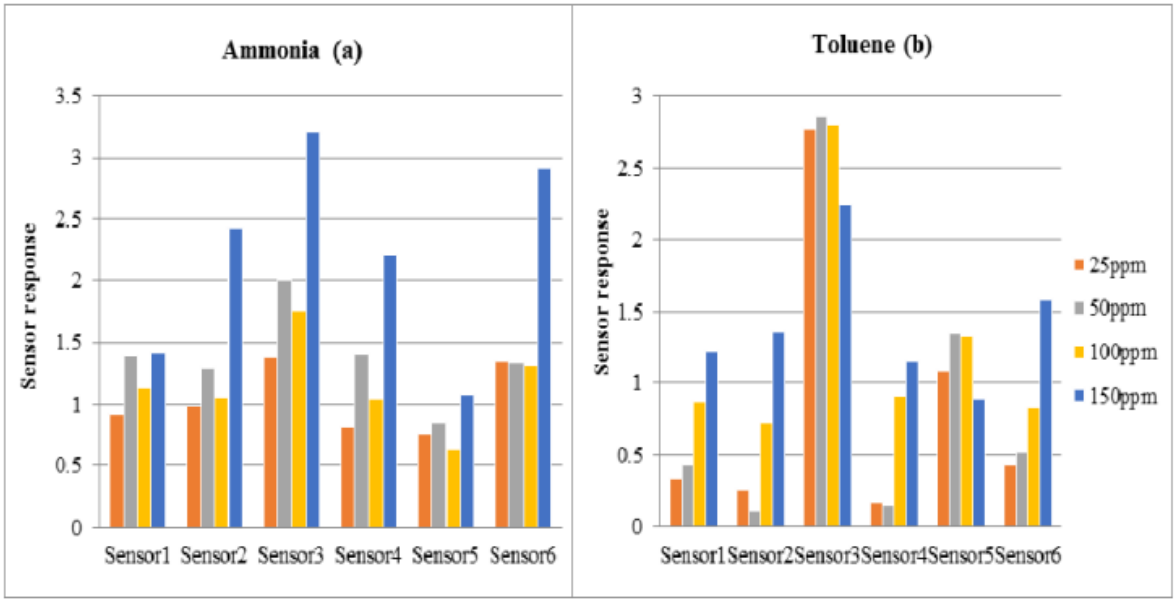


Figure 1.2: Comparison of sensor performance of six e-nose configurations SWCNT-COOH with PVC, SWCNT-COOH with PSE, SWCNT-COOH with PVP, SWCNT-COOH with PVC, SWCNT-COOH with PVA, and SWCNT-NH2 respectively [2]

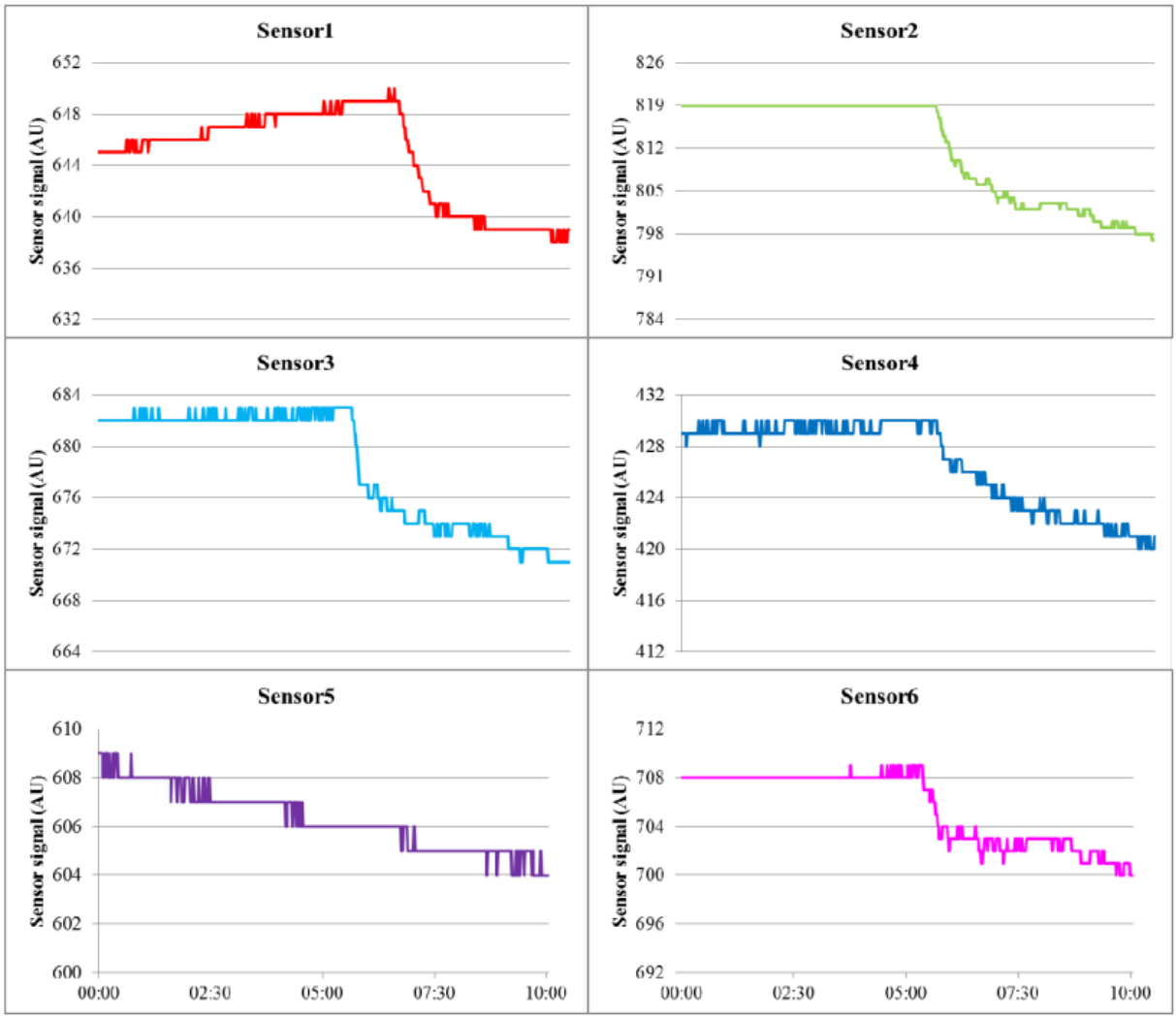


Figure 1.3:E-nose results from six sensors of five minutes of clean air followed by five minutes exposed to chemical [2]

According to a review of UAV designs led by Abdul Aabid, factors that affect the overall performance of a UAV are influenced by mechanical systems (structural, computational, fabrication), communications systems (IoT, AI, navigation), materials (metallic, non-metallic, lightweight), and electrical systems (electronic motors, control, transducers) [4]. The design elements noted above affect weight, endurance/flight time, payload capabilities, and range performance parameters. In addition to pure performance metrics, the two main types (fixed-wing and vertical takeoff) have inherent benefits depending on the desired mission. Rotating-wing (vertical takeoff) UAVs have the main benefits of hover capabilities, rapid movement, small take-off area requirement, and flight versatility; in contrast, fixed-wing UAVs have the main benefits of flight speed, flight endurance/range, and wind resistance [4].

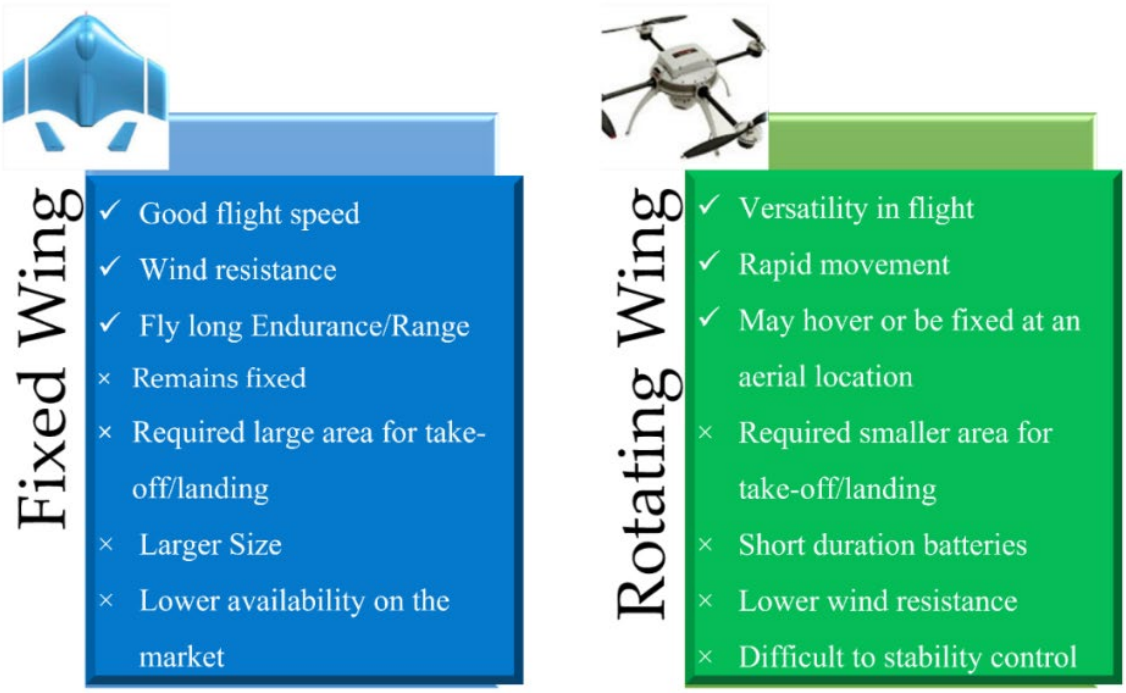
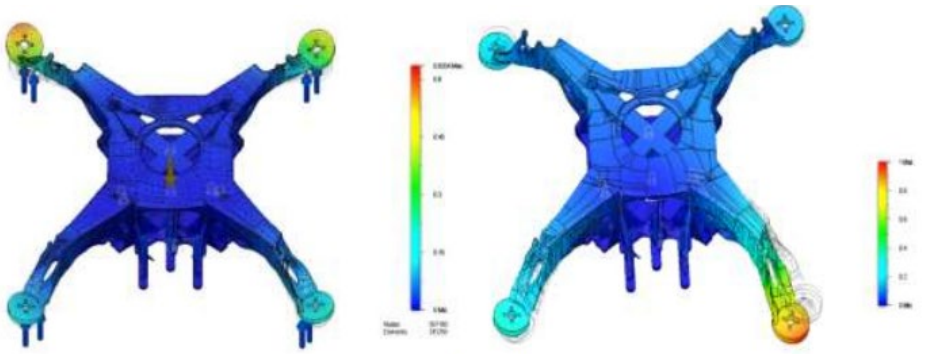
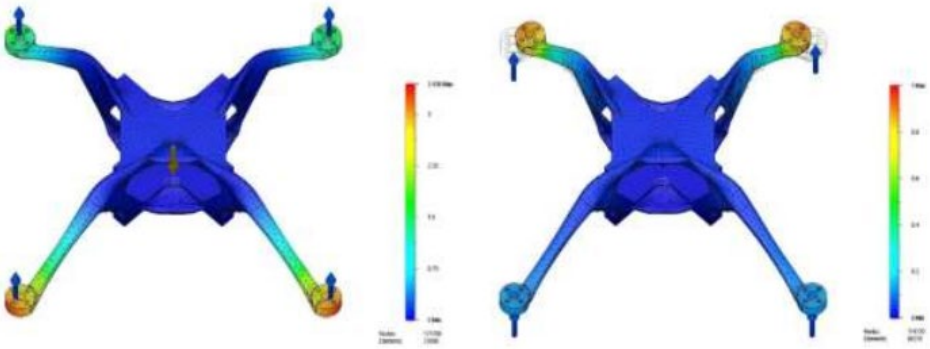


Figure 1.4: Pros and cons of fixed-wing and rotating-wing UAVs [4]

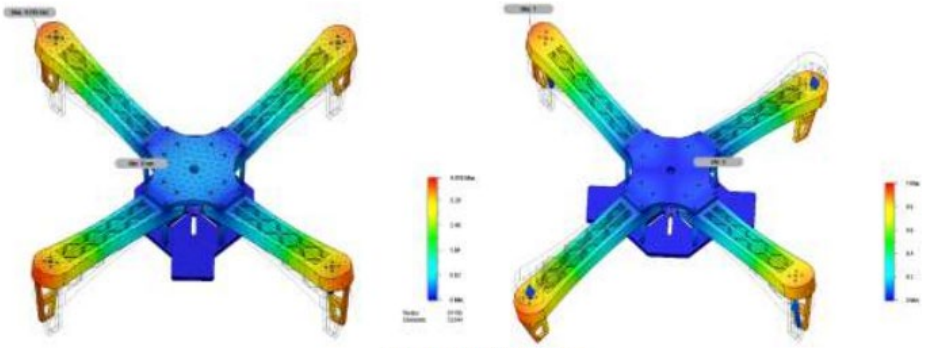
The basis of the design of a UAV can be found in the main structure that houses all its components. While the design of the structure is mainly based on the mass requirement of equipment and flight forces acting upon the body, structural design elements may be analyzed to establish where critical points within the body are most important. In a generative design analysis by Jerrin Bright, two specific generated frames were compared with the baseline DJI design; Frame one features a thinner armed frame with structural support panels added spanning from the root to about the halfway point of the arm, and Frame two features the frame without the root supports. Both generated frames have an inherent downward curve for the arms. The base DJI design had wider arms with no angularity. The DJI design displayed max stress/strain located at the root of the arms due to the moment generated by the motors. Frame two displays the change of location of the maximum stress/strain from the root to where the downward angle starts; with the max stress being greater than the DJI design [5]. Frame one with the reinforced arm displayed a surprising consistency in stress/strain distribution with no noticeable location of max stress [5]. The DJI design showed the greatest amount of deflection due to the stresses of the motor. The main takeaways from this article include the use of angularity to shift the location of max stress and the ability to add more support to a frame to create a consistent stress distribution at the cost of higher mass.



(a) Frame 1



(b) Frame 2



(c) DJI F450 frame

Figure 1.5: Generative UAV structure design displacement FEA comparison [5]

In a quadcopter UAV, the four motors are the main contributing factors in the movement of the drone. The manipulation of the speed and direction of the motors directly affects the vehicle's attitude, pitch, and speed [6]. The reference motor orientation is shown below.

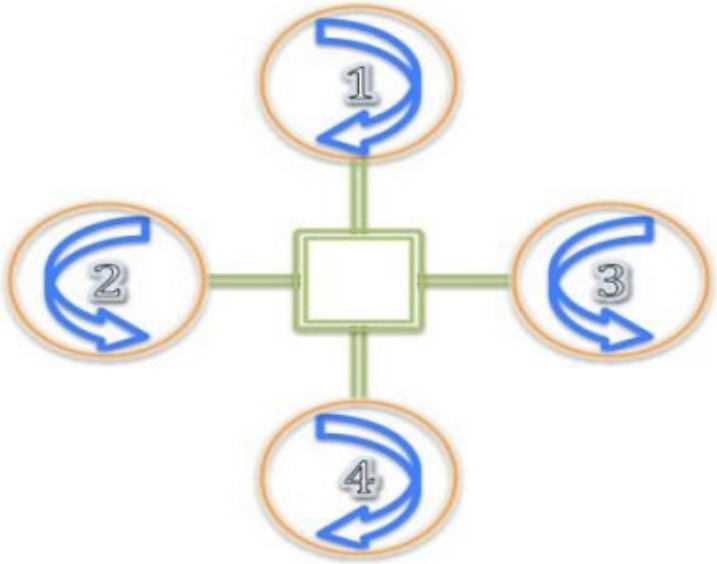


Figure 1.6: Basic motor orientation of a quadcopter UAV [6]

With this reference configuration, the motor's power input (mainly voltage) can be modulated to allow for movement in cases of hovering, forward, backward, left, right, pitch, and roll. In quadcopter designs the speed of movement is based on the pitch of the aircraft. Motors one and four are coupled in the same direction, and motors two and three are coupled. The difference in thrust production between the two couples results in the rotation of the aircraft [6].

For control of movement, UAVs rely on flight controllers to maintain and alter power and signals received by the motors. In general, many of the flight controllers are based on 32-bit ARM MCUs using programming languages of C, C++, Python, or Java. However, it is shown that there is a lack of industrial standards in open-source flight controllers, which makes applications of each controller more specific to its architecture [7].

(a) PIXHAWK (b) Pixfalcon (c) Pixhawk 2

(d) Pixhawk mini (e) PixRacer (f) Pixhawk 3 Pro

Figure 1.7: Pixhawk flight controller series [7]

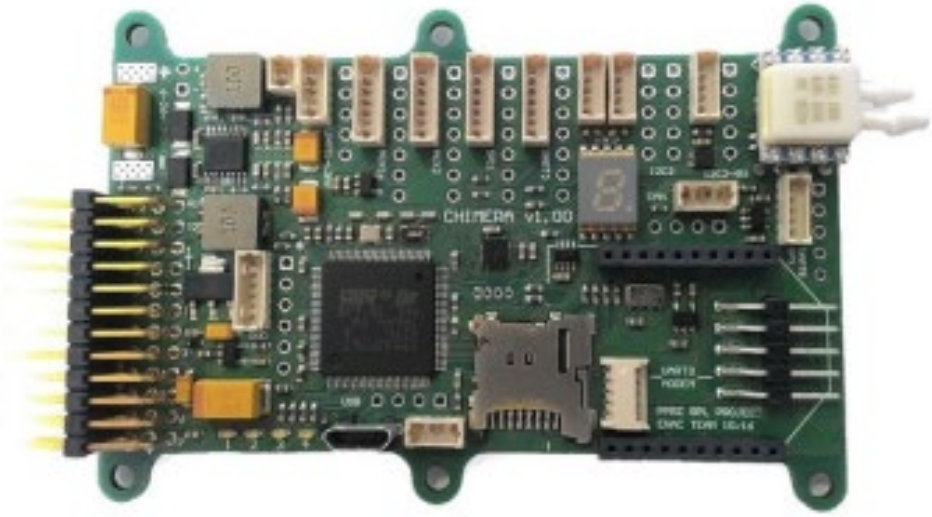


Figure 1.8: Paparazzi Chimera flight controller [7]

1.3 Project Proposal

The objective of this project is to produce a drone that can support the use of a high-definition camera. A program is produced that will focus on machine learning object detection that will be used with the camera footage of the drone as the source.

1.4 Methodology

This project begins with the physical design of the drone; this body is comprised of mostly commercially available parts as this project is intended to be easily replicated. Modifications will be made to the structure to allow for the equipment of required additional components such as a GoPro sized camera. Motors and flight controllers will be selected based on the desired flight speed and movement requirements. A machine learning model is created based on the testing scenario for object detection. Using online resources, the capabilities of the object detection will be analyzed based on detection effectiveness.

2 Subsystems

2.1 Defining Subsystems

The project of building a drone can be broken down into eight main sub-systems. These sub-systems include drone structure, flight control, user-to-drone communication, propulsion, drone power, auxiliary camera, object detection software, and user input.

2.1.1 Drone Structure

The main role of the drone structure is to house all the necessary electronics for the drone to operate and sustain the stresses of flight without signs of damage. This subsystem is important due to its direct integration with the following subsystems: flight control, user-to-drone communication, propulsion, drone power, and auxiliary camera.

The drone structure subsystem must be able to support the equipment needed for the subsystems listed above to operate. This includes proper and secure mounting of important electronics in safe positions to prevent damage in flight. For some, such as flight control and propulsion, standardized FPV mounting formats (bolting locations and motor positioning) will be used; however, some systems (user-to-drone communication, drone power, and auxiliary

camera) may not have standardized mounting. The drone structure must accommodate both standardized and non-standardized forms.

2.1.2 Flight Control

The flight control subsystem is responsible for processing and transmitting information to operate motors in flight. The flight controls have direct integration with the user-to-drone communication and propulsion systems. This subsystem must be able to receive information from the user-to-drone system and translate it into pulse width modulation (PWM) signals that are required for motor operation and control.

2.1.3 User-To-Drone Communication

The user-to-drone communication subsystem is responsible for enabling wireless connection between the user and the drone. The electronics found within this system will receive the user input from the ground station and transmit the information to the flight controller. The main performance metrics associated are wireless range and stability of the connection.

2.1.4 Propulsion

The propulsion system mainly consists of the motor and propellers. The components allow for the drone to sustain flight by generating lift. Propulsion will require vital information from flight controls and will need to follow specifications based on the overall mass and size of the drone.

2.1.5 Drone Power

Drone power will be required to supply sufficient power for all electrical components of the drone to operate. Systems that require power include propulsion, flight control, and user-to-drone communication. The auxiliary camera and user input will be operating on their dedicated power. The battery chosen will be based on target flight times concerning the operational power needs of all electronic components onboard the drone according to manufacturer specifications.

2.1.6 User Input

The user input includes the control pad necessary for the pilot to relay flight commands and the screen or goggles needed for the pilot. This subsystem will be the ground station and is isolated from the drone power and drone structure systems. However, it will need to directly operate in conjunction with user-to-drone communication.

2.1.7 Auxiliary Camera and Object Detection Software Subsystems

The auxiliary camera and object detection software are the most isolated from the rest of the subsystem due to these not being flight-effecting. The auxiliary camera will capture the main

video feed used for analysis and the object detection will use machine learning to analyze the video feed. The auxiliary camera will be needed as the main flight camera will focus on low latency, while the auxiliary camera will focus on video fidelity. The drone structure will need to accommodate the mounting for the auxiliary camera.

2.2 Subsystem Connections

The connections between the subsystems can be seen below in the N2 diagrams.

Drone Structure			Mass and size of drone				
Flight control electronic mounting determines drone frame specifcations	Flight Control		PWM information for motor operation and control	Defines controls power requirements			
User communication electronics must be mountable on frame	Flight control based on signal transmitted from user to drone communication	User to Drone Communication		Communication power requirements of electronics			Feed first person flight video feed to user
Frame must have proper motor mounting and clearance for propellers			Propulsion	Defines power required for motors			
Frame must support mounting for needed battery				Drone Power			
Frame must support mounting for additional cameras					Auxilery Camera	Video feed used for object detection analysis	
						Object Detection Software	
		User imput must be able to be recieved					User Input

Figure 2.1: Random N2 diagram

User Input	User imput must be able to be recieved						
Feed first person flight video feed to user	User to Drone Communication	Flight control based on signal transmitted from user to drone communication		Communication power requirements of electronics	User communication electronics must be mountable on frame		
		Flight Control	PWM information for motor operation and control	Defines controls power requirements	Flight control electronic mounting determines drone frame specifcations		
			Propulsion	Defines power required for motors	Frame must have proper motor mounting and clearance for propellers		
				Drone Power	Frame must support mounting for needed battery		
			Mass and size of drone		Drone Structure		
					Frame must support mounting for additional cameras	Auxilery Camera	Video feed used for object detection analysis
							Object Detection Software

Figure 2.2: Ordered N2 diagram

3. Flight Controls and Communication

3.1 Flight Control/Communication System Design Overview

The controls and communications systems of a drone allow for the control of the motors that are responsible for motion and connectivity of the drone to the user control. A simple drone control and communication system will be based on modern small first-person view (FPV) drones. The core motor controls components in a FPV drone are the flight controller and the electronic speed controller (ESC). These two components are generally commercially produced to work with each other. The flight controller is a microprocessor that uses the intended input and translates the input into commands for motor control; the ESC uses the output commands from the flight controller and applies power to generate phase electronic pulses that allows for the motors of the drone to spin [8]. These two components act as the core of the drone and other components are wired to the flight controller and ESC to operate.

For FPV drones to operate, cameras and camera signal transmission will be required for the user to be able to have visual of how the drone is moving. Flight cameras are small, low latency modules that are equipped to the front of the drone to allow for visual and video transmitters (VTx) transmit the flight camera feed to drone flight goggles or ground station; these components are usually compliant to different wireless protocols depending on the manufacturer [8]. Reliability is very important for these two components as loss in visual signal can lead to accidents when flying FPV drones. It is important to note that flight cameras will only be used to fly and will not be recorded for machine learning object detection. A higher fidelity, small form factor camera will be equipped for recording purposes.

The last major component of this system is the control receiver. The receiver wirelessly connects with the physical flight controller that the pilot will use to operate the drone. In the use case of a FPV drone, the receiver will use the ExpressLRS (ELRS) protocol. ELRS is an open-source radio control protocol operating in the 2.4GHz frequency with the main benefits of long-range performance, low latency, small size, and low cost [9].

3.2 Flight Controller and Electronic Speed Controller

With cost in mind, the commercially available flight controller and ESC that will be used for this drone are the SpeedyBee F405 v4 flight controller and SpeedyBee BLS 55A ESC. This controller stack was designed specifically to be used with quadcopter drones with support for all the other components mentioned in the design overview above and the BLS 55A having support for four motors.

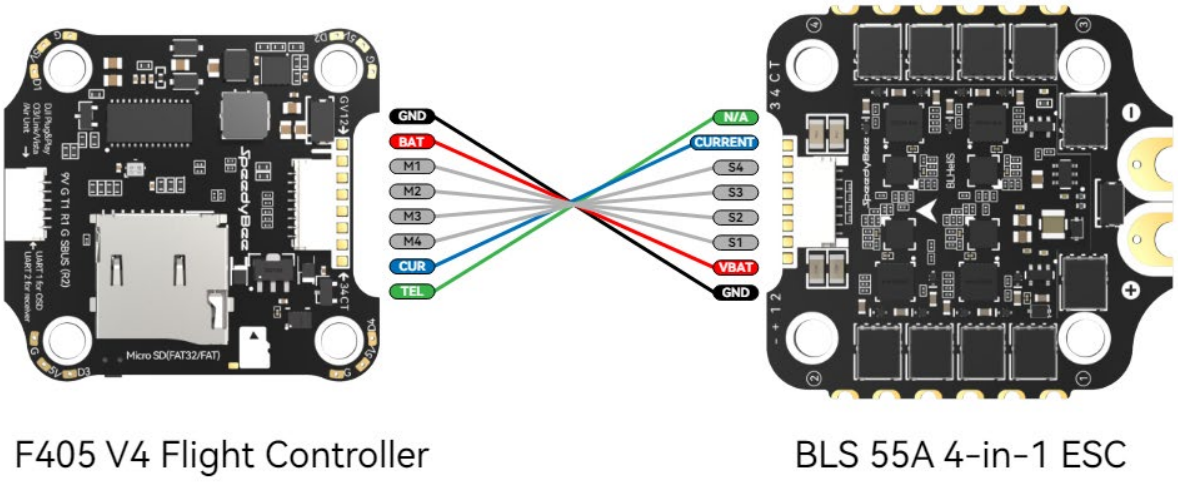


Figure 3.1: SpeedyBee F405v4 and BLS 55A [10]

The wiring for the connection between the F405 and BLS includes individual data for each motor and battery power pass through [10]. The connection can be made through individual wire soldering or using the included 8-pin cable. Power is fed through the BLS from a battery to the F405 as seen in the diagram below.

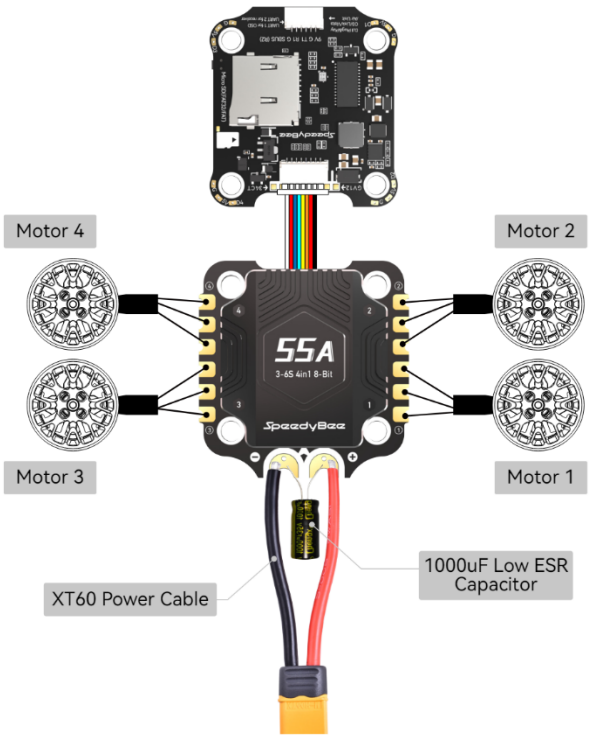


Figure 3.2: SpeedyBee BLS 55A wiring [10]

Connections to the other components such as the camera, video transmitter, and receiver will be wired through the F405v4 as seen in the figure below.

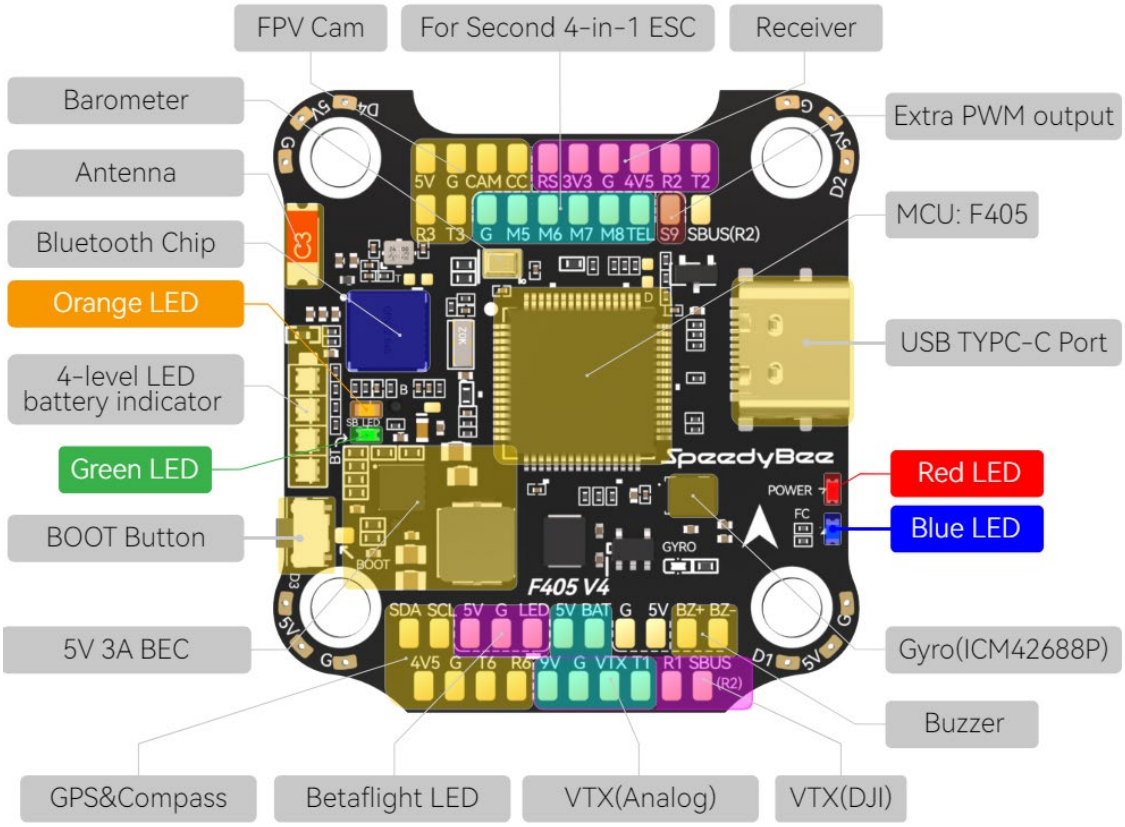


Figure 3.4: SpeedyBee F405v4 wiring guide [10]

3.3 Flight Camera and Video Transmitter

For flight visuals, a small camera module will be included on the drone. For this drone, the module used will be the Foxeer Micro Razer 1200TVL camera. This camera is capable of a video feed of 1200 TVL (equates to nearly 1080p video quality) at a 4:3 aspect ratio and 125 degrees field of view according to Foxeer's product specifications [11]. When analyzing compatibility with the SpeedyBee F405v4, the Foxeer camera can operate with voltage of 4.5V-25V; the F405v4 has a dedicated camera power trace rated at 5V. At five volts, the Foxeer camera only consumes 0.7 watts of power based on the recorded amperage of 140 mA [11]. Connection with the F405v4 requires three wires to be soldered to the PCB: CAM (referred to as VID on the Foxeer camera), 5V power, and ground. Reference figure 2.5 for the wiring connection points.

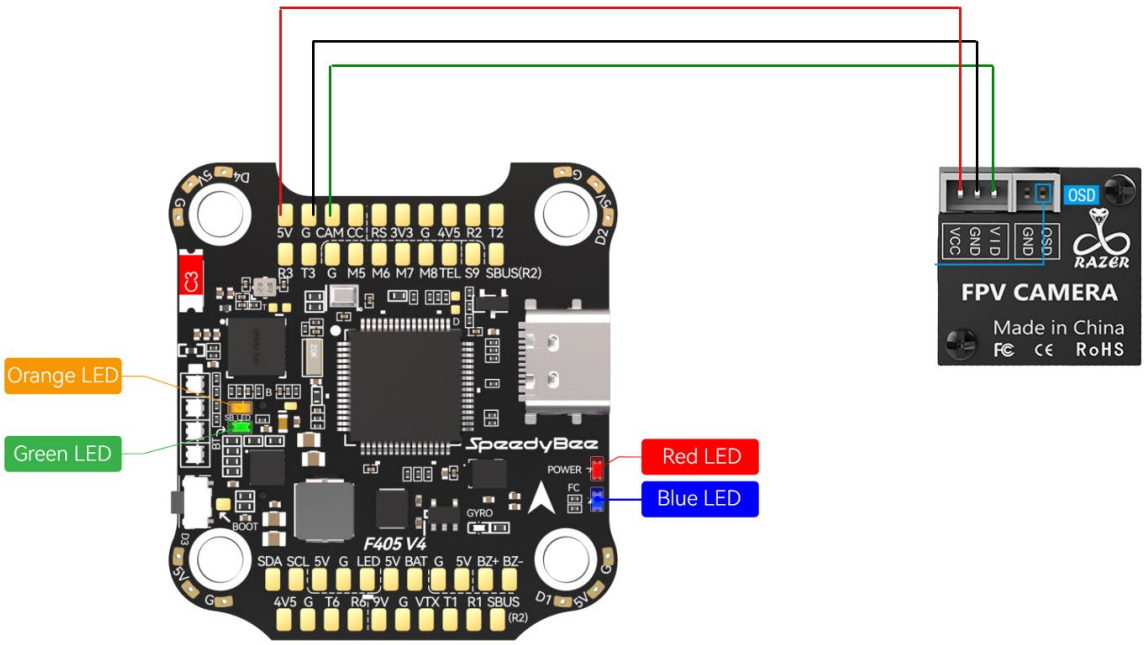


Figure 3.5: SpeedyBee F405v4 to Foxeer Micro Razer Camera wiring

To pair with the camera module, a video transmitter must be included to transmit the video signal from the drone to the ground station for flight visuals. The video transmitter selected in this drone is the Happymodel OVX303. The OVX303 is a module using the OpenVTX protocol that allows for the visual connection with any analog supported googles or ground station displays; the module operates in the 5.8GHz frequency with a required power input of 5Vs and transmit power of 25 mW, 125 mW, or 300 mW according to manufacturer specifications [12]. The OVX303 will require 5V power, ground, smart audio, and video connections with the flight controller. See the wiring guide below in figure 3.6.

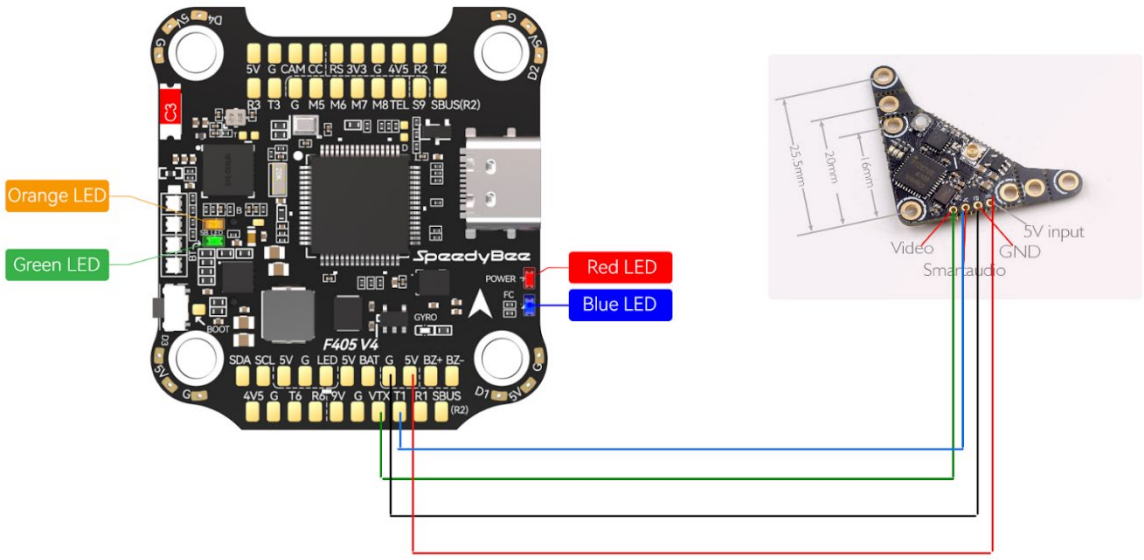


Figure 3.6: F405v4 to OVX303 wiring

3.4 Control Receiver

The control receiver allows the flight controller to communicate with the pilot by receiving input commands and transmitting feedback. The control receiver chosen is the Happymodel EP1 RX operating in the frequency range of 2400-2500 MHz, peak gain of 2.23dB, and has an operating voltage of five volts according to manufacturer specifications [13]. These specifications fit in line with the operational needs of the F405v4. For wiring, the ER1 will require four connections to be soldered: five-volt power, ground, transmit, and receive. The F405v4 has a dedicated five-volt power (labeled as 4V5) and ground for the power needs of the ER1. The transmit (T2) on the F405v4 will be connected to the receive (RX) on the ER1; the receive (R2) on the F405v4 will be connected to the transmit (TX) on the ER1. Refer to figure 3.7 for wiring between the F405v4 and ER1.

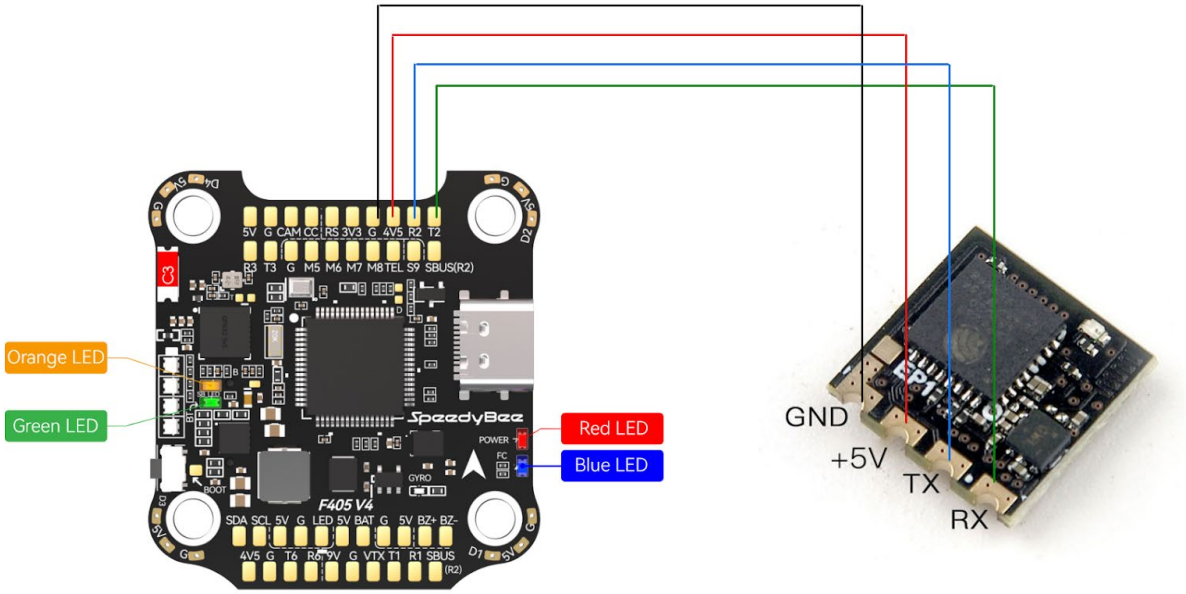


Figure 3.7: F405v4 to EP1 wiring

3.5 Flight Control/Communication System Mass and Cost

The masses and cost of the components for the flight controls and communications are compiled in table 3.1 below. These values do not include the required wires and antennas as they may vary based on drone frame.

Table 3.1: Mass and cost of components from the flight controls and communications

Part Name	Mass (g)	Cost (USD)
SpeedBee F405v4 + BLS 55A	23.5	69.99
Foxeer Micro Razer 1200TVL	4.5	22.9
Happymodel OVX303	1.5	29.99
Happymodel EP1 RX	0.42	16.99
Total	29.92	139.87

4. Propulsion and Power

4.1 Propulsion/Power System Overview

The propulsion and power systems of a FPV drone are directly connected due to the major components influencing each other's performance. The main component in the propulsion system is the motors. There are four motors in a quadcopter located at the end of each arm of the

drone. The motors are connected through the ESC through the dedicated soldering locations through wires. The main component in the power system is the battery. While there are a variety of the sizes for FPV batteries, the main indicator of what the requirements for the battery needed is the motors due to it being most of the power draw.

4.2 Propulsion

There are two numbers that are used within the naming of the motor that indicate the properties. The first four-digit number indicates the physical size of the motor. The number is structured in an AABB where the first two digits (AA) show the stator width and the last two digits (BB) represent the stator height [14]. For example, a 2207 motor will have a stator width of 22 millimeters and a stator height of seven millimeters. The stator is the stationary section of the motor which consists of a bearing, polled magnets extending out, and copper wiring coiled around the poles. Similar to how electric motors operate on a car, the electricity acting on the coils causes the magnets to change polarities, which interacts with outer magnets on the moving part to create rotation [15].

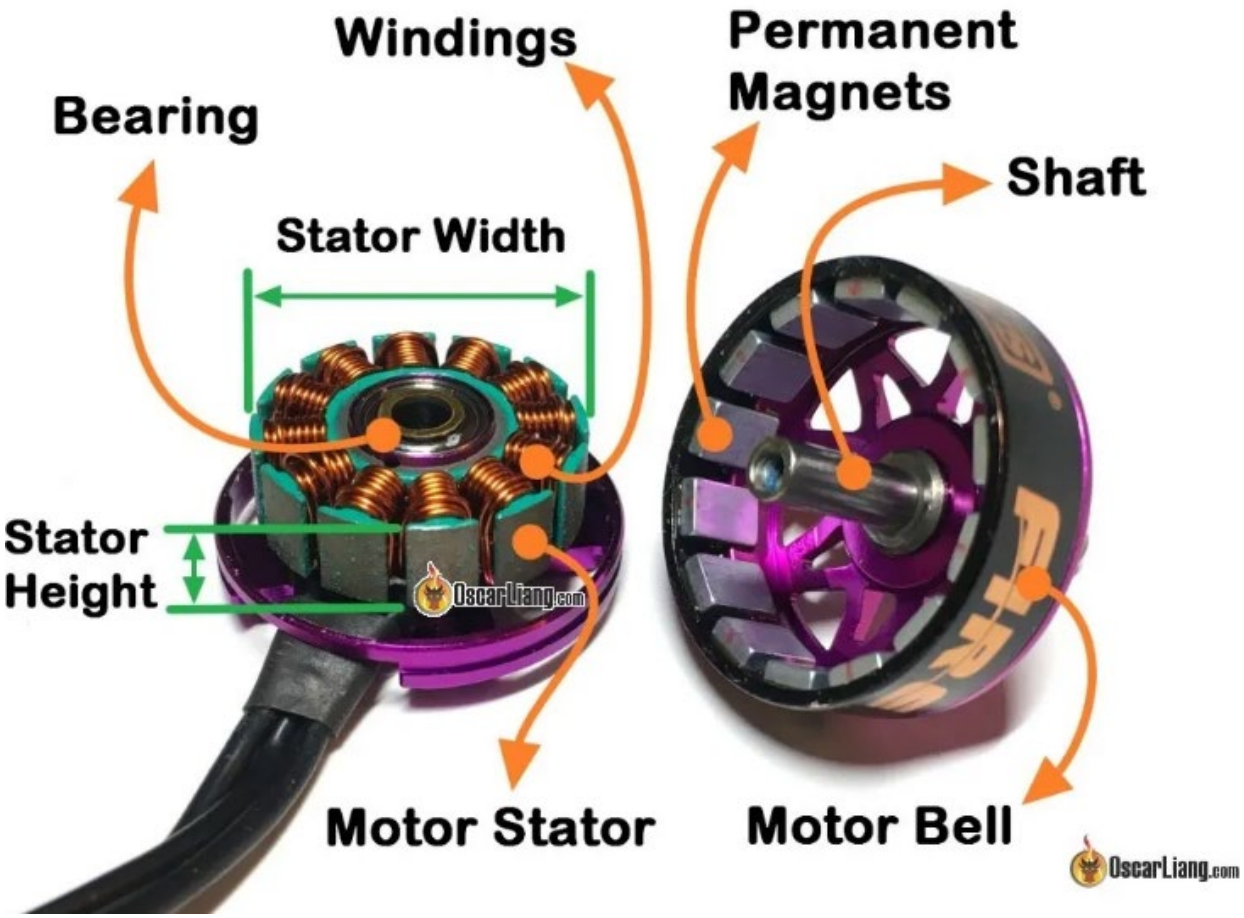


Figure 4.1: FPV drone motor components [14]

The second number in the naming convention of a FPV motor is known as the KV value. The KV represents the rated revolution of the motor when one volt of electricity is applied to the motor [14].

Being based on a five-inch drone design, the motor size that will be most appropriate for this drone will be 2207 or 2208. When looking at 2207 or 2208 motors, there is a large range of available KV ratings to explore from 1700 to 2200. Since the focus of this drone build is flight time as opposed to speed, it would be more beneficial to use a lower KV rating due to the lower power consumption. The motor chosen was the BrotherHobby VS 2207 1720 KV which has a max thrust of 1830g at 23.8V and 38.8A, this motor can handle up to 6S batteries [16].

Figure 4.2: BrotherHobby VS 2207 1720KV motor [16]

For connection between the motors between the motors and the ESC, there are three soldering pads for each motor on the ESC that corresponds to the three-wire coming from the motor. The wires will be soldered onto the pads with the middle motor wire going into the middle ESC solder pad. The position of the other two wires is less important due to the VS motor being bush-less (meaning able to spin in both directions), the directionality of the motors will be adjusted in software.

To pair with the motors, propellers will be needed to generate lifting forces for the drone to operate. There is a wide variety of propellers to choose from; for the use case of capturing footage, the propellers for this drone should be optimized for smoother flying instead of greater agility to prevent inconsistent footage for use in machine learning object detection. For this instance, the propellers used will be the Azure Power Johnny Freestyle 4.8 inch. These propellers feature a length of 4.8 inches, hub inner diameter of five millimeters, polycarbonate material, and overall weight of 3.9 grams each [17].

Figure 4.3: Azure Power Johnny Freestyle propellers [17]

4.3 Power

The drone will be battery power with the power entering the ESC as seen in section 3.2. The main design factor in deciding which battery to be used in the drone is flight time. The flight time of a drone can be determined by the equation shown below.

$$T = 60(CD/A) \tag{4.1}$$

T is representative of the total estimated flight time in minutes, C is the total battery capacity, D is the battery discharge (80% assumed), and A is the average current draw [18]. For the minimum flight time estimation, the current draw will be the sum of the max current of the motor; for the selected motors, that will be 155.2 amps. For a five-inch drone, the recommended battery size to minimize the effect mass would have on flight is between 1100 mAh to 1400 mAh. See the table below for the expected minimum flight time of each battery capacity within the designated range. The actual flight time may be greater depending on the flight speed and tuning of the motors. This estimation also does not include the power required to run other electronics onboard, but the power impact that other electronics will have been minimal compared to the motors.

Table 4.1: Minimum flight time estimation

Battery Capacity (mAh)	Minimum Flight Time (minutes)
1100	3.388960205
1150	3.543003851
1200	3.697047497
1250	3.851091142
1300	4.005134788
1350	4.159178434
1400	4.31322208

Since the goal is to optimize flight time, the largest battery possible without significant impact on the mass of the drone will be best, 1400 mAh. The battery that would match the requirements would be the Tattu R-Line 1400mah LiPo. The battery features a six cell, 22.2V design capable of 1400 mAh capacity at a below market mass of 222 grams for the capacity [19]. The Tattu battery also has a discharge plug of XT60, which is the same connector used by the ESC. Since the battery is easily swappable, an extra battery can be held on standby to extend the total flight time if needed.

Figure 4.4: Tattu R-Line 1400 mAh battery with XT60 plug [19]

4.4 Propulsion/Power Mass and Cost

The masses and cost of the components for the propulsion and power systems are compiled in table 3.2 below.

Table 4.2: Propulsion/power mass and cost table

Part	Mass (g)	Cost (USD)
BrotherHobby VS 2207 1750 KV x 4 (motor)	130	19.99
Azure Power Johnny Freestyle 4838 (prop)	16	3.99
Tattu R-Line 1400 mAh (battery)	222	39.99
Total	368	63.97

5. Structure

5.1 Drone Structure

The drone structure consists of the frame that will be onboard electronics needed from the other subsystems. The construction of a drone frame consists of four main parts, the main plate, top plate, arms, and bottom brace. The main plate includes the mounting for most electronics with the top plate installed on top of the main plate with spacers and bolts. The top plate also allows for the mounting of the battery. The arms are bolted below the main plate which allows the motors to be mounted. The bottom brace is installed below the arms to add structural rigidity to the complete construction. In a frame where the arms of the drone are separate instead of a single part, the bottom brace becomes more important to prevent the arms from loosening. Drone frame parts are generally fabricated from cut carbon fiber or similar fiberglass panels, varying in thickness from two millimeters to four millimeters.

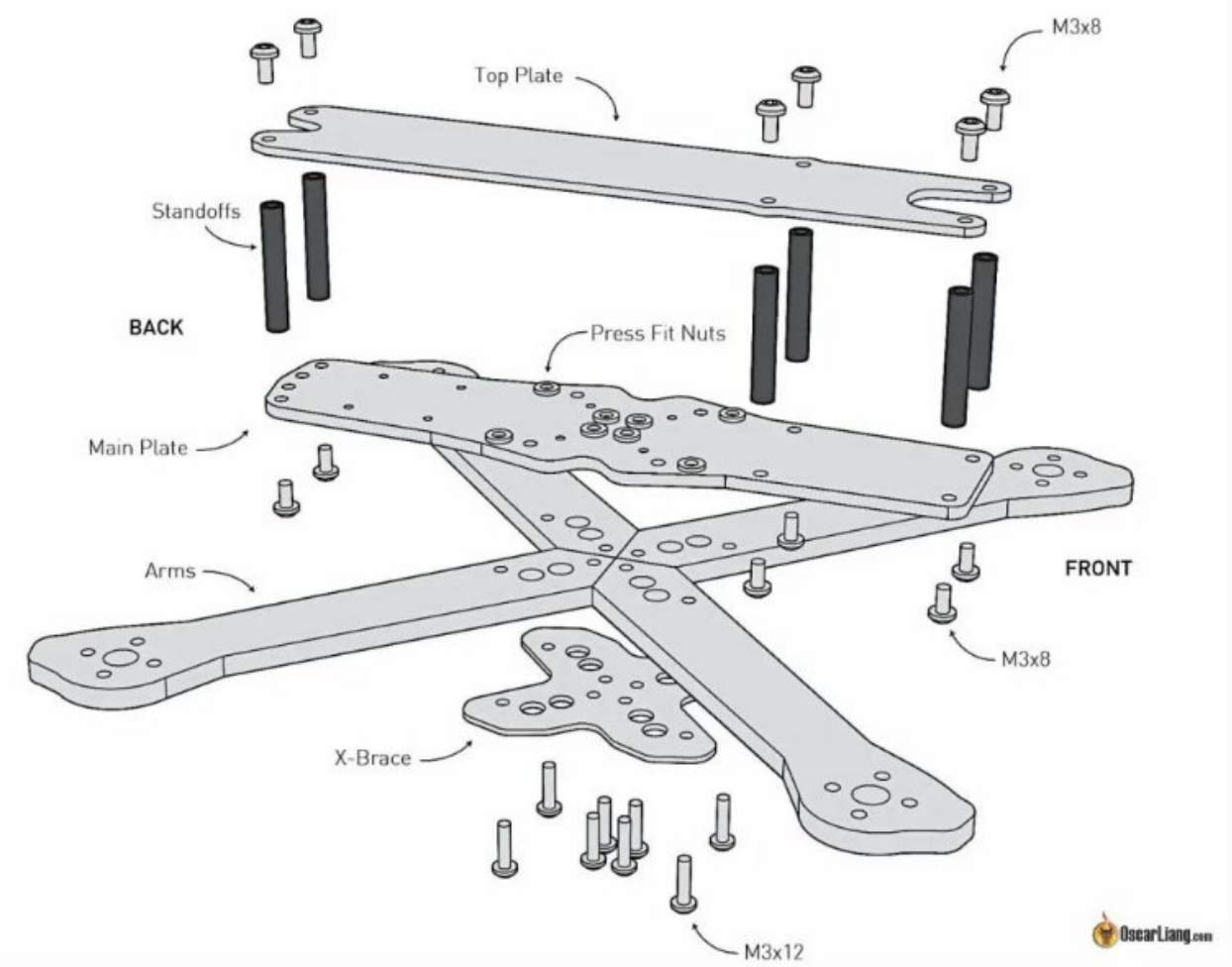


Figure 5.1: Construction of an FPV drone frame [20]

One aspect to consider when choosing a frame is the configuration. Drone frames can be broken down into two distinct categories, symmetrical and asymmetrical. Symmetrical frames position the motors an equal distance away from the center point of the drone, while asymmetrical frames may position the front or back motors slightly closer than the other two motors. Examples of symmetrical frames include X-frames, H-frames, and box frames. The letter in the X-frame and H-frame naming convention refers to the shape that the arms of the drone create, and box frames connect the tips of the frames creating a square. Symmetrical frames tend to have better inherent flight dynamics resulting in the motor tuning being simpler; in addition, box frames add more structural protection in case of collisions. An example of an asymmetrical frame is the deadcat. In the case of the deadcat configuration, the front motors are swept back compared to the rear motors. The advantage of the deadcat is the ability to keep the propellers from entering the frame of any forward-facing cameras.

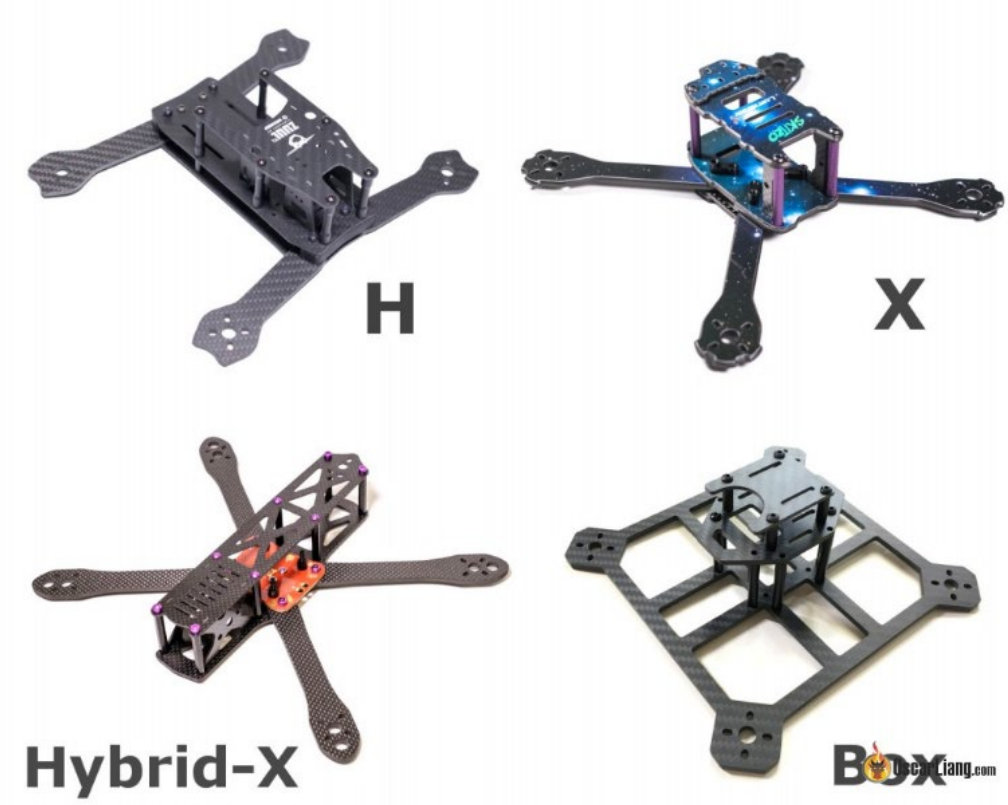


Figure 5.2: Symmetrical drone frames [20]

Figure 5.3: Deadcat drone frame [20]

5.2 Drone Frame Selection

While it would be ideal to custom design a frame for this project, the additional cost and production time of custom laser cutting parts would not be beneficial for creating a low-cost and easily accessible for the masses to experiment with object detection drones. Commercially available options were explored to find a frame that would suit the use case of capturing footage for use with object detection. As mentioned in section 4.1, the deadcat configuration will be best suited. The frame must also have 30 mm x 30 mm mounting for the flight controller and mounting for analog-style micro cameras.

The drone frame selected for this project was the RotorRiot CL2-Air. The CL2 is a five-inch deadcat frame with support for 30 mm x 30 mm stack mounting [21]. Other specifications include a wheelbase of 220 mm, 4 mm arm thickness, 3 mm main plate thickness, 2 mm top plate thickness, analog micro camera mounting support, 16 mm diameter motor mount, max stack height of 25 mm, and estimated mass of 166 g [21]. These specifications correlate with the dimension and mounting specification of the components from the other subsystems. Mounting for the auxiliary camera can be produced by 3D printing based on the dimension of the front micro camera mounting bolts. This frame also support mounting for additional cameras.

Figure 5.4: RotorRiot CL2-Air drone frame [21]

6. Machine Learning

6.1 Machine Learning Object Detection

Machine learning object detection is the basis of the analysis of effectiveness for this project. Machine learning can be applied to object detection software to automate the identification process using a preset database of contextual information. This means that human creation of datasets and training of machine learning will be required to fine-tune the effectiveness of object detection. YOLOv5 is the open-source object detection model that can be implemented in this project. YOLOv5 is an example of a regression analysis-based convolutional neural network (CNN) based on CUDA graphics acceleration in its algorithm [22]. YOLO object detection models have been renowned for the performance of target and location recognition and scalable performance based on computational hardware.

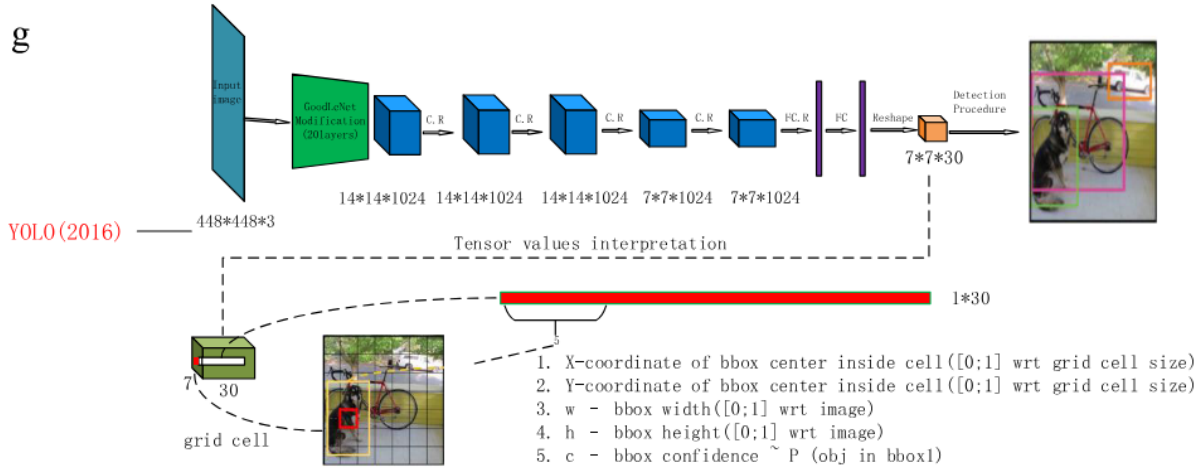


Figure 6.1: YOLO machine learning object detection model [22]

Since YOLOv5 still uses a fixed dataset constructed by human identification of objects, the program will still need a large library of reference data. More context given to the machine learning algorithm will result in greater accuracy in object detection.

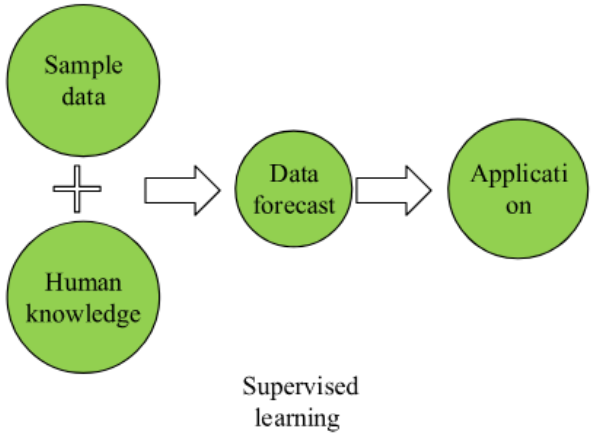


Figure 6.2: Machine learning training model [22]

6.2 Incorporation of Machine Learning Object Detection

Implementing machine learning into this project will require the use of an external computer that can allow for the processing of drone footage due to the higher computational needs of object detection. YOLOv5 can be easily used with any computer using Python as a control console, Nvidia's CUDA toolkit to access computer GPU computing, and PyTorch as machine learning model management [23]. It is important to consider that YOLOv5's performance is highly dependent on the processing power of the user's GPU. There are modifications of model size to accommodate lesser hardware, but generally, there will be a trade-off between power and performance based on the model size.

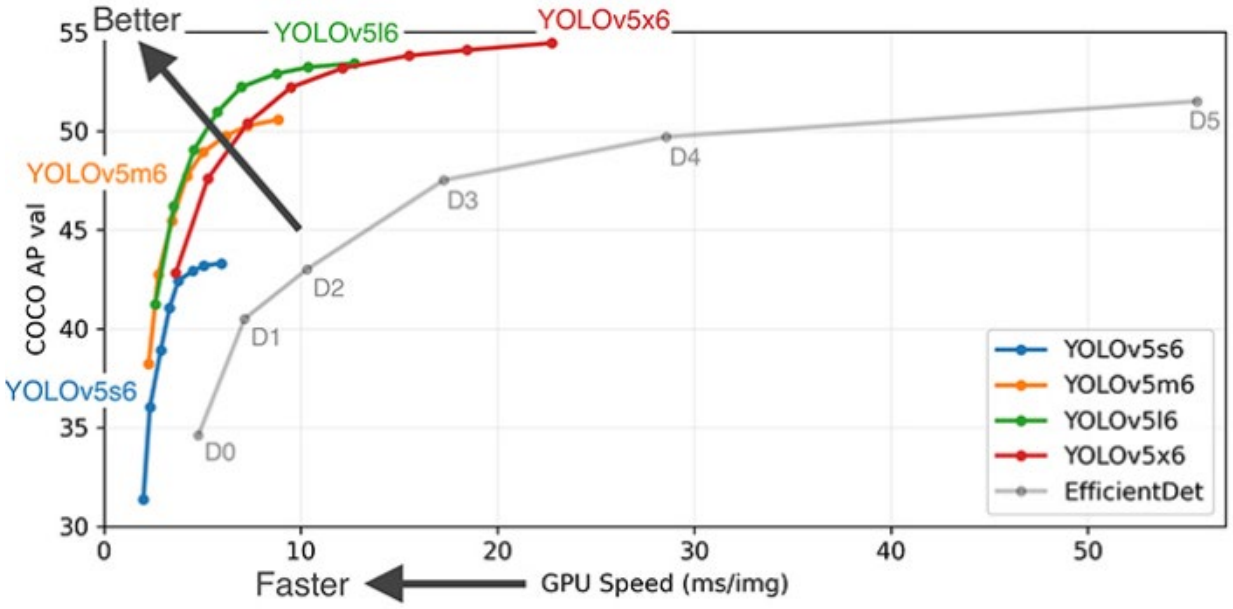


Figure 6.3: Comparison of different model sizes of YOLOv5 [23]

7. Base YOLOv8 Testing

The version of YOLO being used in this project is YOLOv8 adapted by Ultralytics. YOLOv8 is a more updated version of YOLO compared to the previous version mentioned above which uses command line implementation with Python, allowing for the program to be more easily implemented and used. The program is still dependent on CUDA protocols and has five default model with varying speeds and detection parameters: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x [24]. The specifications between the model are included below.

Table 7.1: YOLOv8 models [24]

Model	size (pixels)	mAPval 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.2	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

The following tests in this chapter will compare the overall performance of the YOLOv8 models in terms of detection and classification results.

7.1 Base YOLOv8 Test Case 1: High Object Environment

In the first test case of the capabilities of YOLOv8, a picture of a crowded highway was used. The photo below contained many vehicles of varying types, and the algorithm would be required to detect each vehicle and classify the type of vehicle (car, truck, or bus). In addition, some of the vehicles overlap due to the camera's perspective, which may prove to be a challenge for YOLOv8 to detect multiple vehicles.

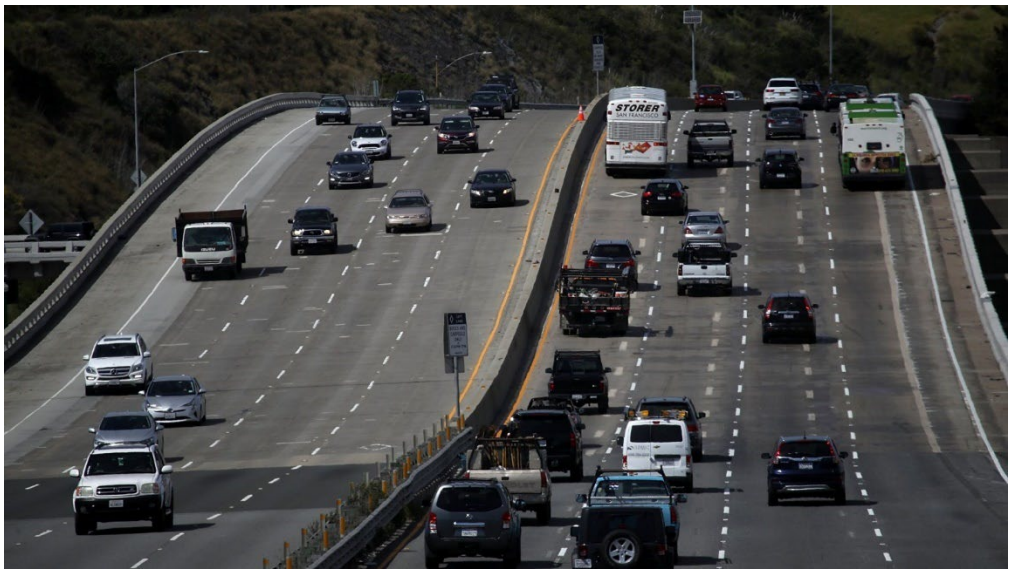


Figure 7.1: Base test 1 reference photo [25]

Using the YOLOv8n model, the program was able to detect many of the vehicles on the high with a detection rate of 76%. However, there were instances of vehicles being detected twice as seen by the overlapping detection boxes. There were also instances of misclassifications as a truck was identified as a bus and some cars were identified as trucks. In this case, some SUVs can be confused as trucks from the front due to similar structures without the visible back of the vehicle to provide context of the differences.

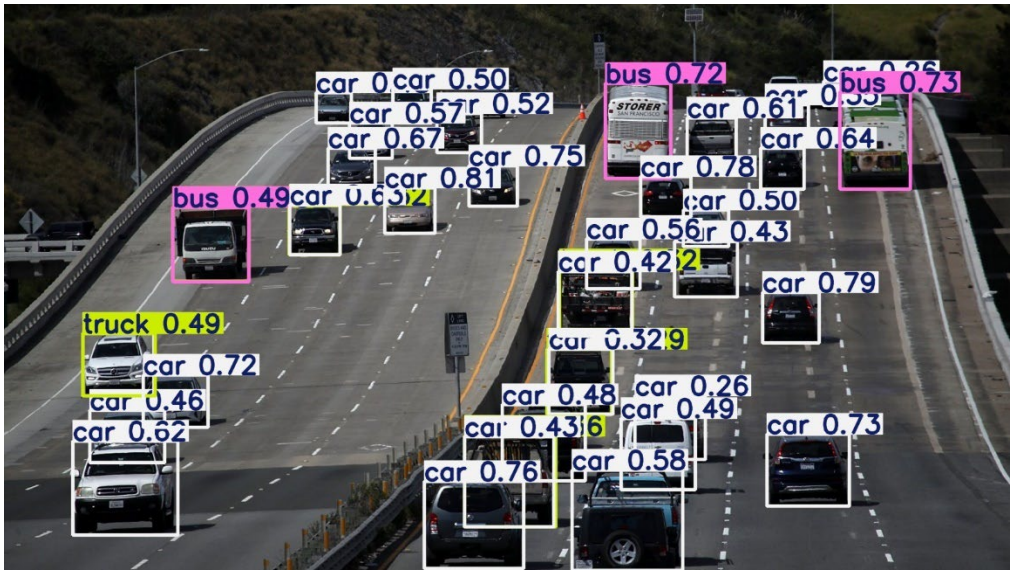


Figure 7.2: YOLOv8n test case 1

Using the YOLOv8s model, the results were similar to the YOLOv8n model despite a slightly lower detection rate (71%) and classification rate (64%) due to lesser total detections and classifications. In comparison to the previous model, there were no instances of double detection and fewer instances of misclassification for buses. The misclassification of some cars as trucks persisted.

Figure 7.3: YOLOv8s test case 1

Using the YOLOv8m model, the results were more stagnant as opposed to improvement. The detection rate was the same as the YOLOv8n model with a slightly higher classification rate, but there was an instance of misclassification of a bus and double detection. Detection of trucks improved overall compared to the previous models.

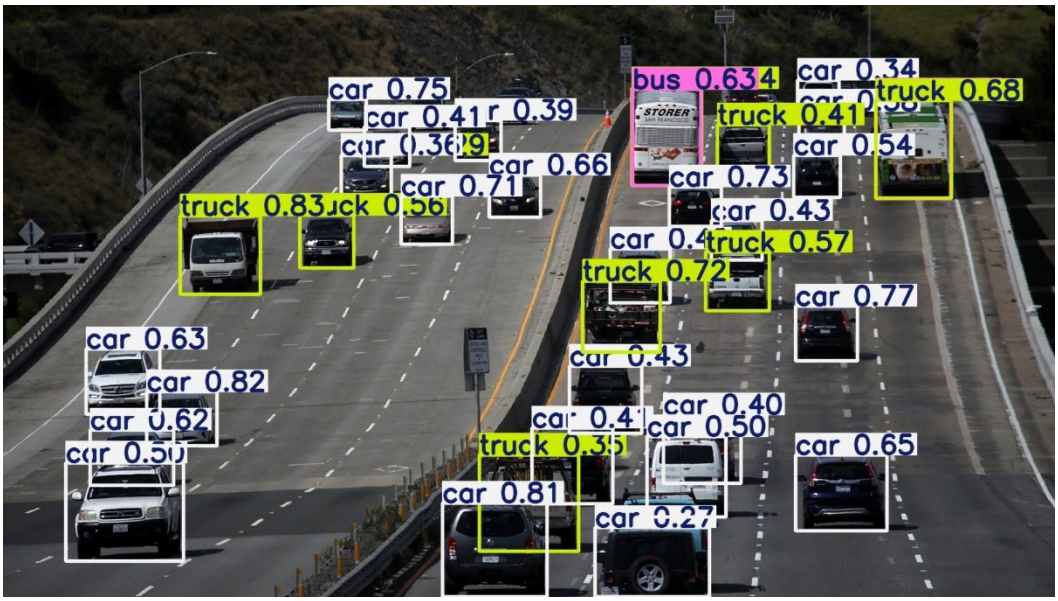


Figure 7.4: YOLOv8m test case 1

The results of the YOLOv8l model yielded visually similar results to YOLOv8m with a similar detection rate and classification rate to YOLOv8s model. This model also resulted in the highest average classification certainty.

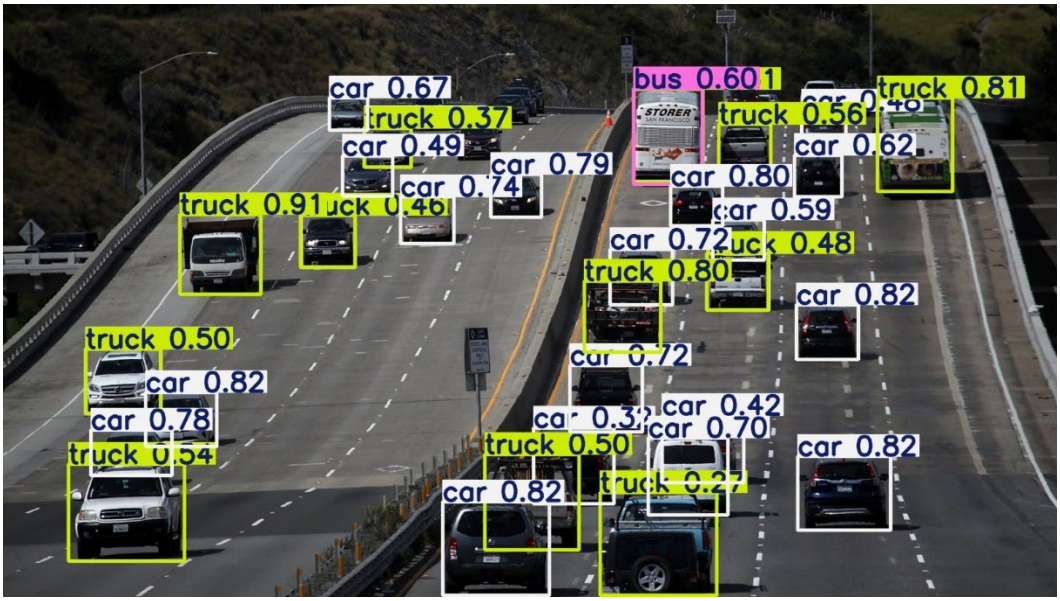


Figure 7.5: YOLOv8l test case 1

Using the YOLOv8x model, the results were statistically the best of all the models. This model yielded the highest detection rate, classification rate, and average classification certainty. Visually, this model provided the best distinction between cars and truck, but still had a problem with identifying buses.

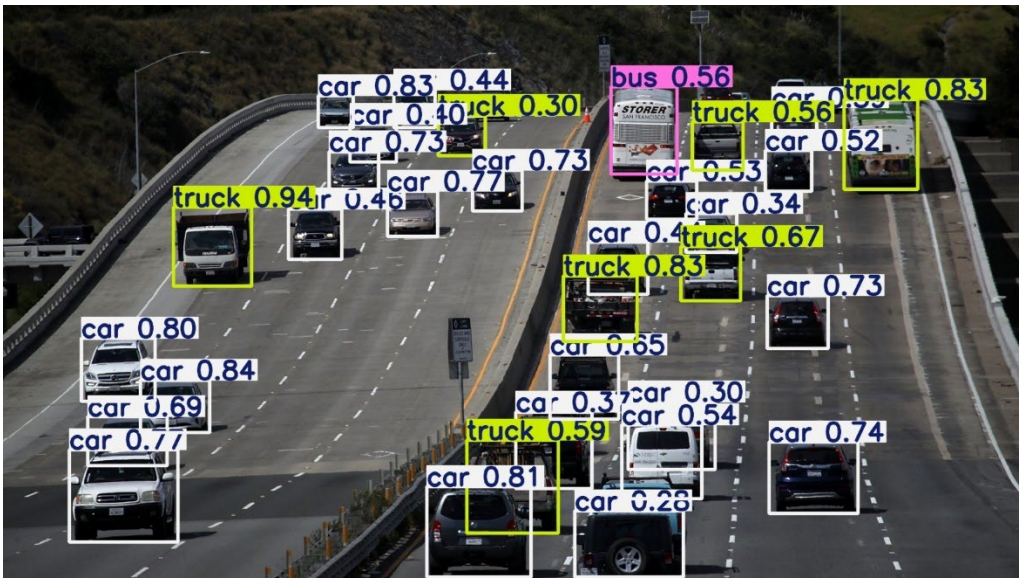


Figure 7.6: YOLOv8x test case 1

Included below are the numerical statistics for comparison of all the models shown above. These statistics reference the total amount of vehicles visible in the original photo. Statistics should be used in conjunction with visual observations of each model to have a complete understanding of the overall performance.

Table 7.2: Test case 1 numerical results

Model	Number of Detections	False Detections	Detection Ratio	Correct Classifications	Classification Rate	Average Certainty
YOLOv8n	36	4	0.7619047619	29	0.6904761905	0.57
YOLOv8s	30	0	0.7142857143	27	0.6428571429	0.53
YOLOv8m	35	3	0.7619047619	30	0.7142857143	0.56
YOLOv8l	31	1	0.7142857143	26	0.619047619	0.63
YOLOv8x	33	0	0.7857142857	32	0.7619047619	0.61

7.2 Base YOLOv8 Test Case 2: Low Light Conditions

In the second testing case, the original photo is the same as the first testing case with alteration made to result in a more difficult detention environment. Adobe Lightroom was used to

digitally adjust the photo to have a much darker exposure and contrast to simulate an inferior camera configuration or improper setup.

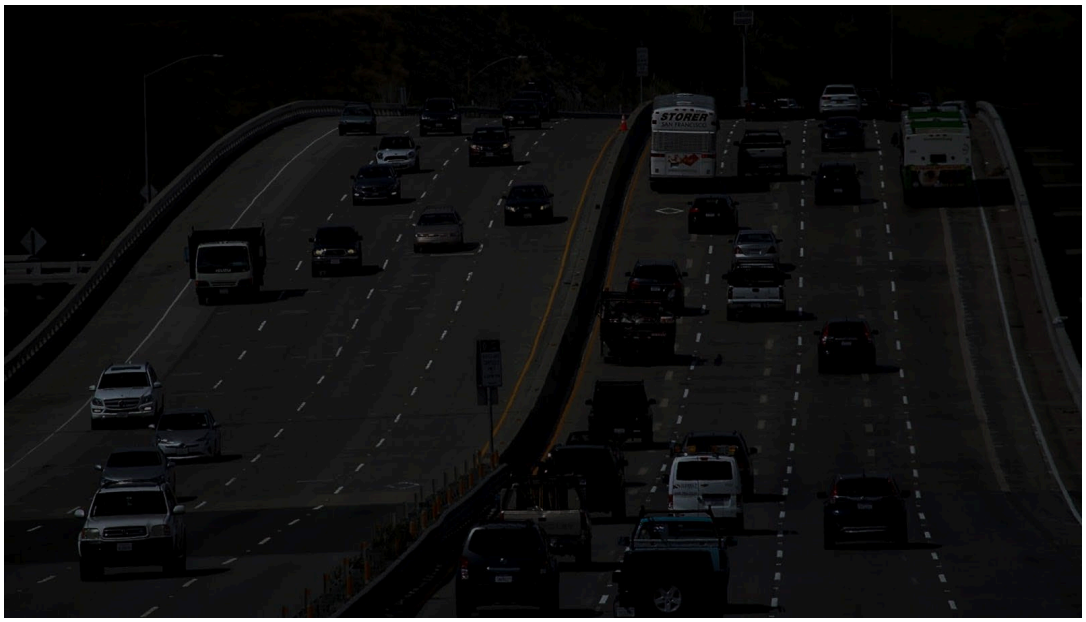


Figure 7.7: Base test 2 reference photo

The results of the different models with the digitally augmented photo can be seen below.

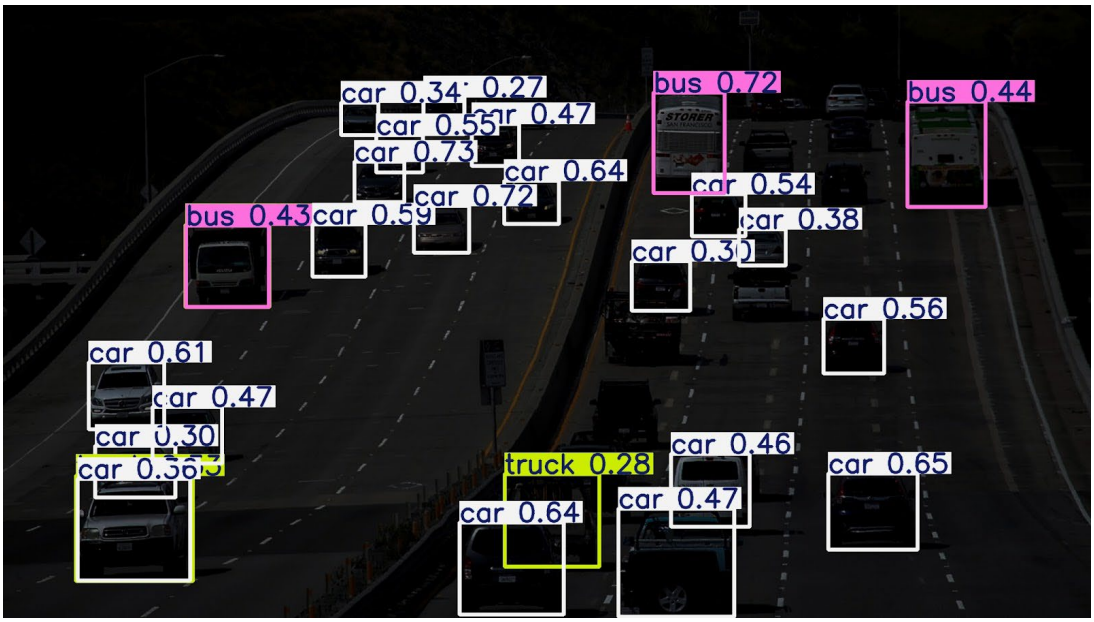


Figure 7.8: YOLOv8n test case 2

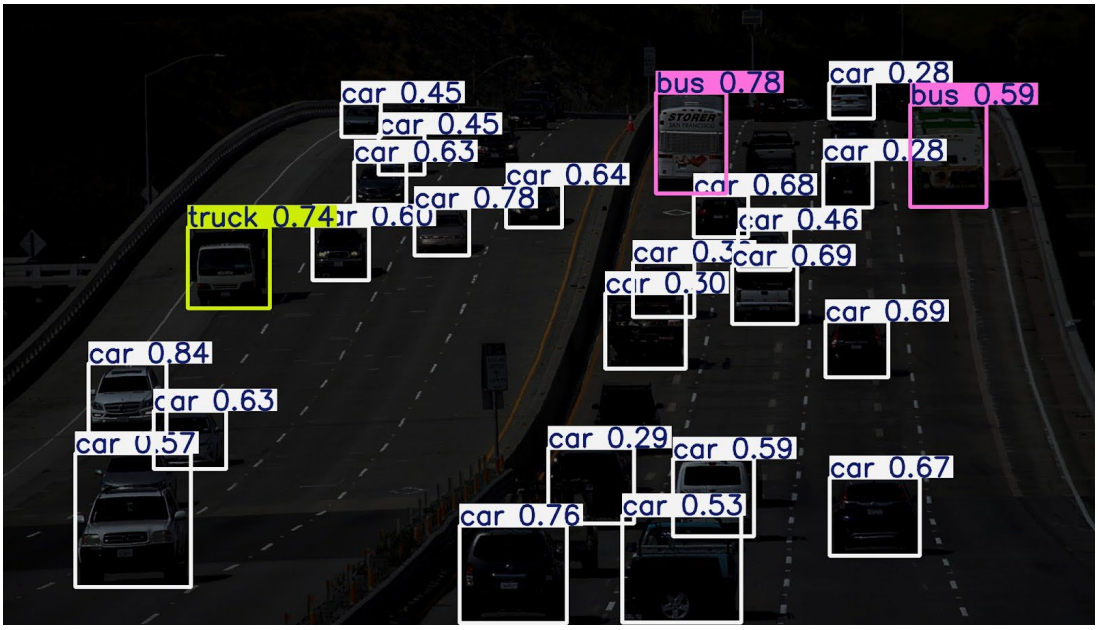


Figure 7.9: YOLOv8s test case 2

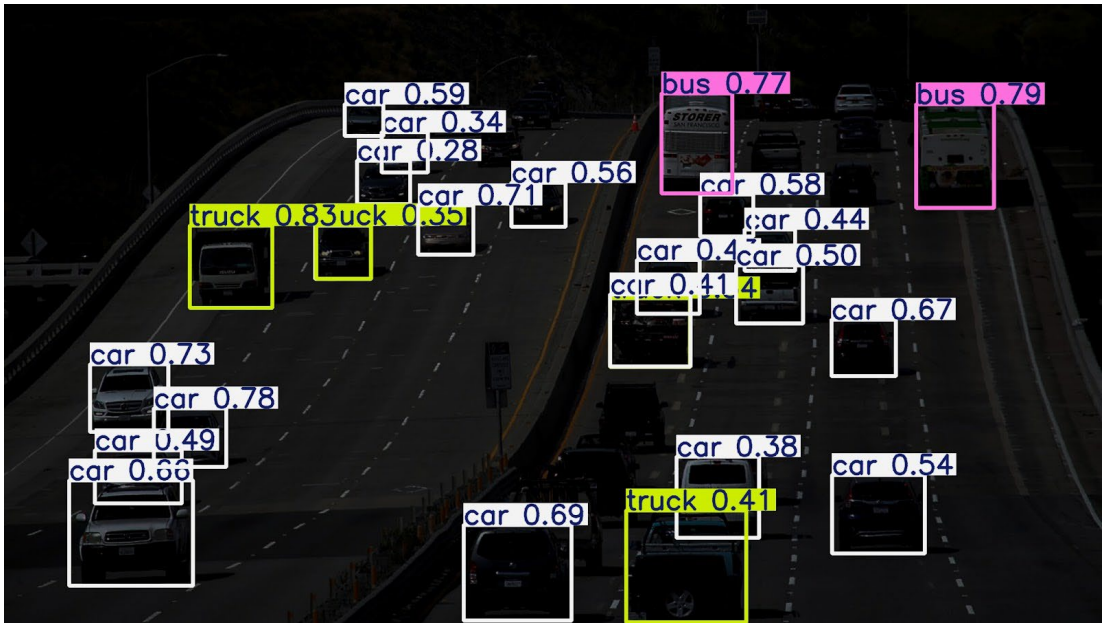


Figure 7.10: YOLOv8m test case 2

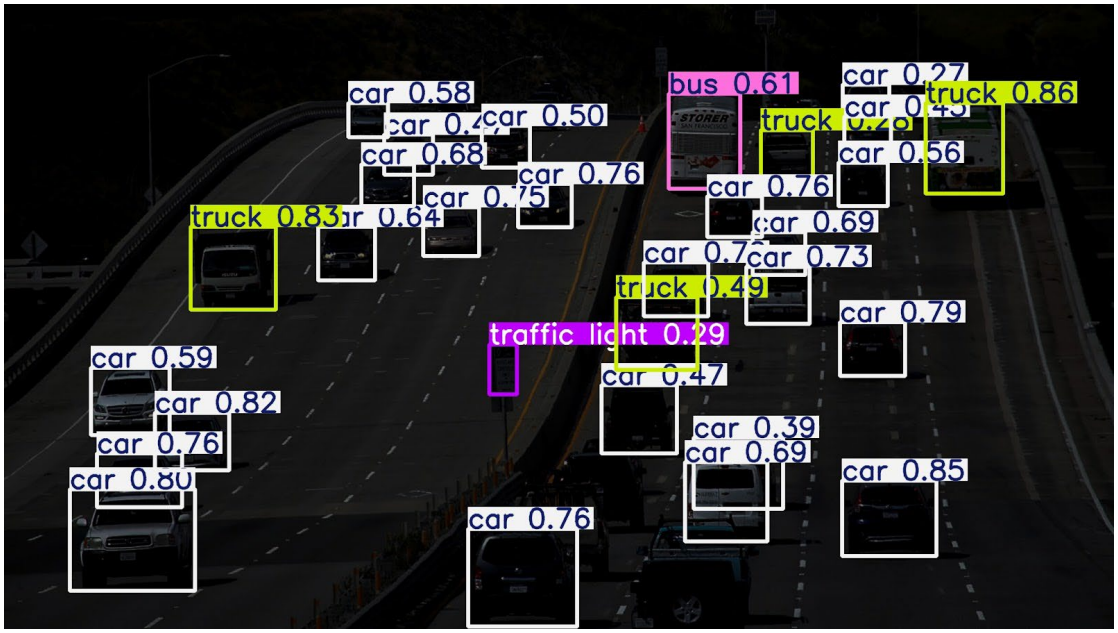


Figure 7.11: YOLOv8l test case 2

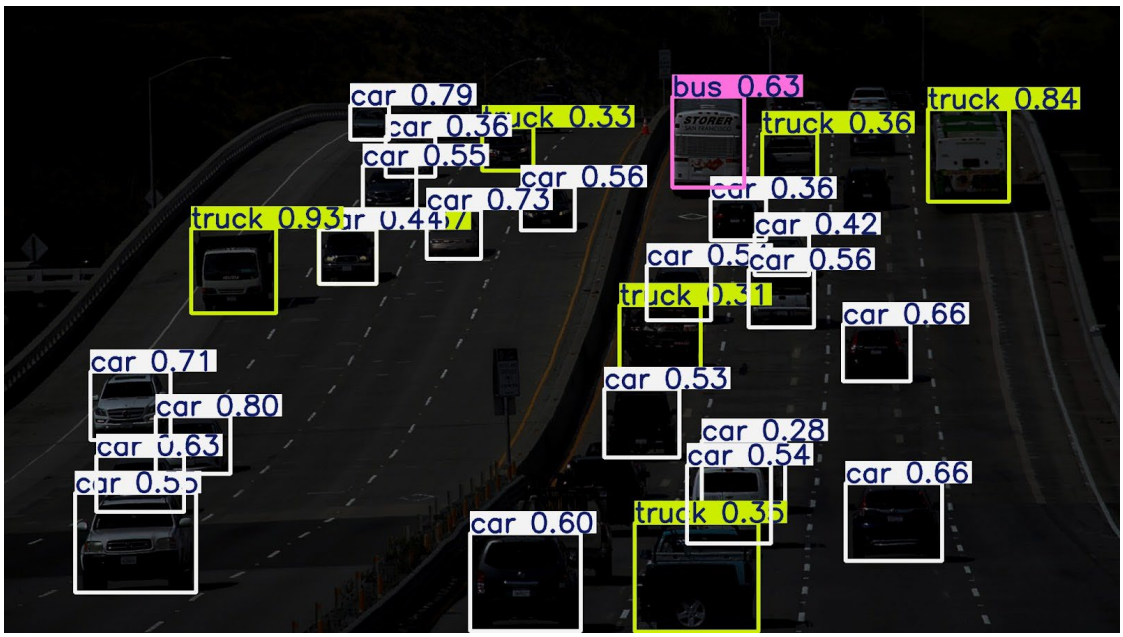


Figure 7.12: YOLOv8x test case 2

The results of the darkened photo and the original were similar. However, the darkened photo yielded noticeably lower total detections and classification. Issues with cars overlapping due to the camera angle proved to be a greater problem with worse lighting conditions. The numerical statistics for the results can be seen below.

Table 7.3: Test case 2 numerical results

Model	Number of Detections	False Detections	Detection Ratio	Correct Classifications	Classification Rate	Average Certainty
YOLOv8n	25	1	0.5714285714	23	0.5476190476	0.49
YOLOv8s	25	0	0.5952380952	23	0.5476190476	0.57
YOLOv8m	24	1	0.5476190476	23	0.5476190476	0.56
YOLOv8l	28	0	0.6666666667	27	0.6428571429	0.64
YOLOv8x	28	1	0.6428571429	26	0.619047619	0.56

7.3 Base YOLOv8 Test Case 3: In and Out of Focus

The third test of the base YOLOv8 models shows the effectiveness of each model's ability to perform with varying focus on the object. The photo chosen for this test was a picture of apples shot with a large aperture allowing for the apple in the foreground to be in focus and the apples in the background to be out of focus. There was one overlapping apple in the foreground and one in the background; however, only the overlapping apple in the foreground will be expected to be detectable due to the blur in the background.

Figure 7.13: Test case 3 reference photo [26]

The YOLOv8n model was only able to detect four apples out of the realistically possible eleven apples with a classification certainty of 0.51.

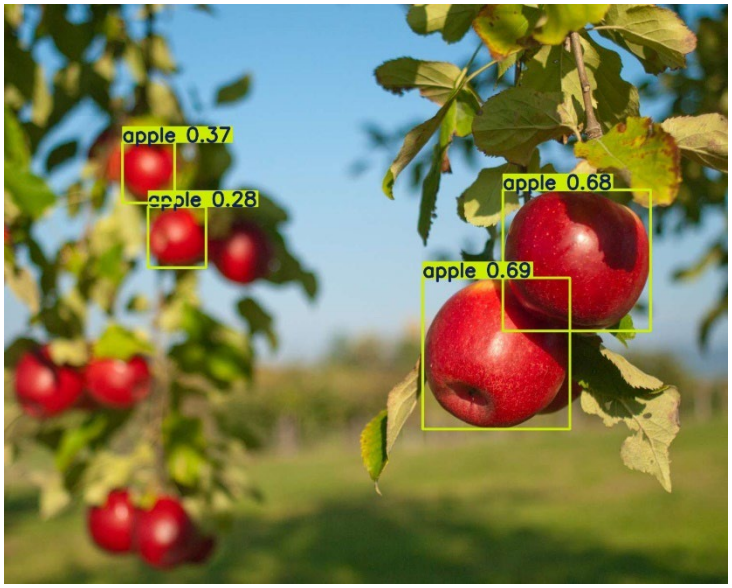


Figure 7.14: YOLOv8n test case 3

With the YOLOv8s model, five of the eleven apples were detected. However, there was a wrong detection of a leaf as an apple. The average classification certainty was 0.55.

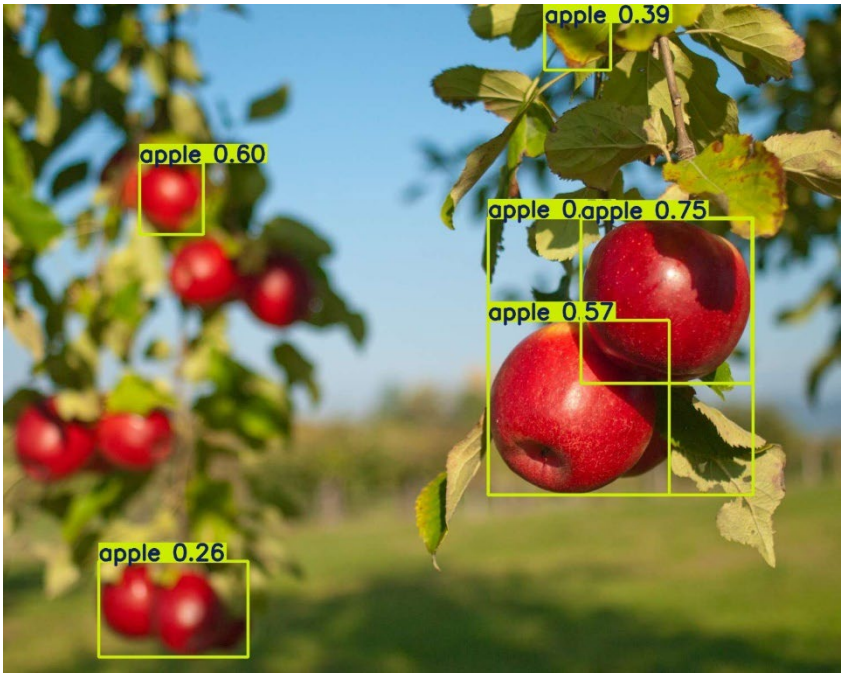


Figure 7.15: YOLOv8s test case 3

Using the YOLOv8m model, there was a notable improvement in total detection with eight of the eleven apples detected and an average classification certainty of 0.60.

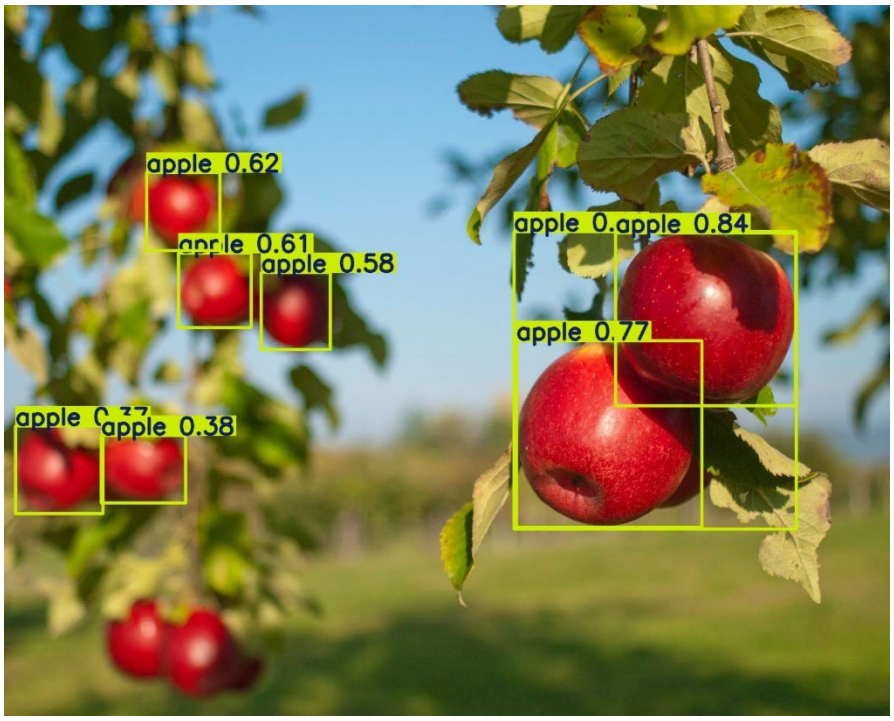


Figure 7.16: YOLOv8m test case 3

The YOLOv8l model yielded the best performance with a detection rate and classification rate of 91% and average classification certainty of 0.64.

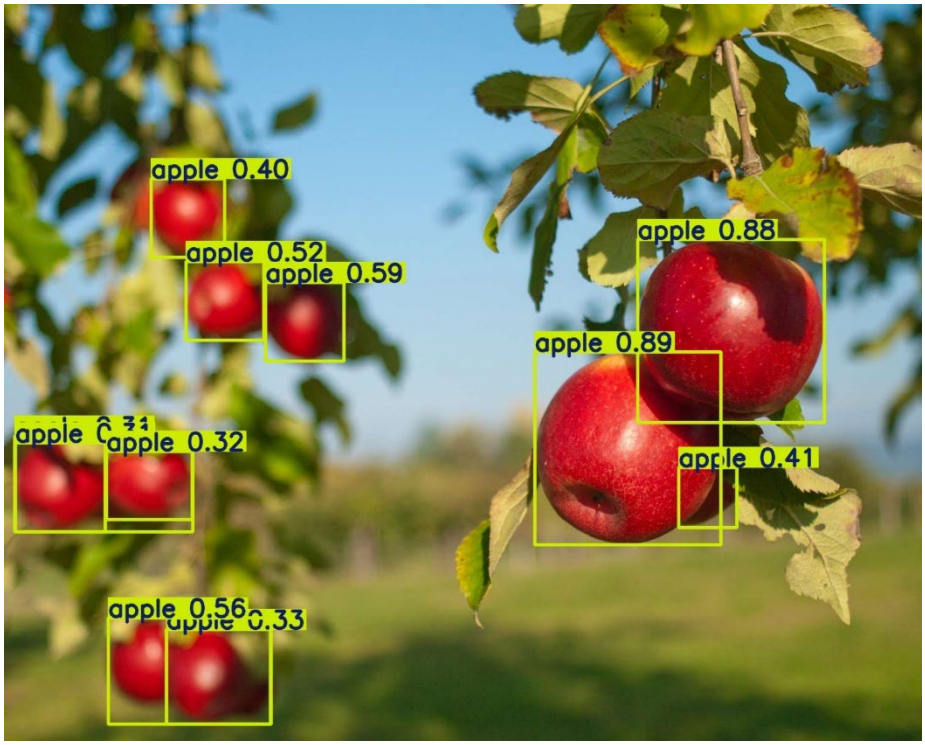


Figure 7.17: YOLOv8l test case 3

Using model YOlOv8x, the number of detections seemed to have regressed but yielded the highest classification certainty of them all at 0.90. There seems to be priority with high certainty within this model analyzing this photo.

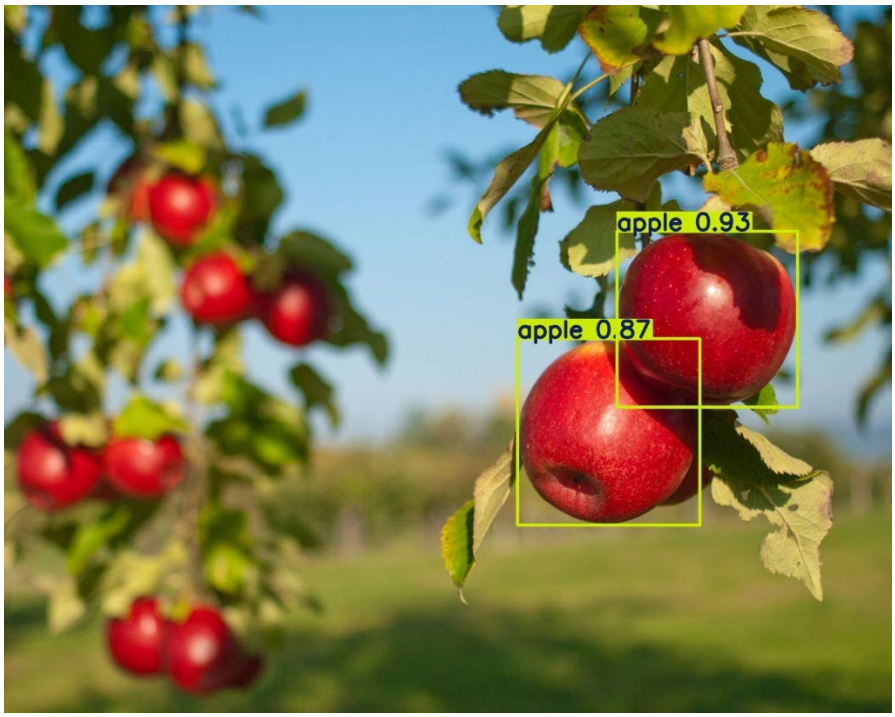


Figure 7.18: YOLOv8x test case 3

The statistical data for this test can be seen below.

Table 7.5: Test case 3 numerical results

Model	Number of Detections	False Detections	Detection Ratio	Correct Classifications	Classification Rate	Average Certainty
YOLOv8n	4	0	0.3636363636	4	0.3636363636	0.51
YOLOv8s	6	1	0.4545454545	5	0.4545454545	0.55
YOLOv8m	8	0	0.7272727273	8	0.7272727273	0.6
YOLOv8l	11	1	0.9090909091	10	0.9090909091	0.52
YOLOv8x	2	0	0.1818181818	2	0.1818181818	0.9

7.4 Base YOLOv8 Test 4: Video

For the testing case of video, the focus was the comparison of the processing time. Since YOLOv8 processes video frame by frame, there are more chances for an object to be detected and identified. However, frame-by-frame detection results in a much longer processing time due to the equivalence of processing multiple pictures for a span of a video. In this testing case, a video of cars driving by a camera on a highway was used that was uploaded in 30 frames per second with a total run time of about 21 seconds. The video results in a total of 610 total frames. The results will vary based on the hardware used; in this test case, the graphics processing unit used was a Nvidia RTX 3080. Lower-grade hardware will result in slower processing time and

higher-grade hardware will result in faster processing time. Other factors that impact overall time will be hardware optimization for CUDA protocol and hardware allocation if background processes are being run in addition to YOLOv8.

Table 7.6: Test case 4 run time results

Model	Run Time (s)
YOLOv8n	23.17
YOLOv8s	23.52
YOLOv8m	24.32
YOLOv8l	25.38
YOLOv8x	28.33

When analyzing the overall impact on time due to the fidelity of each model, there is a 5.16 seconds delta between the highest and lowest fidelity models. This delta can be expected to be greater as the video used is longer or uses a higher framerate (both resulting in higher total frames).

8. Custom Dataset Integration

To allow YOLOv8 to detect specific information for agricultural needs, a custom dataset must be created for the internal Ultralytrics training program to train with. The training process includes finding a formation of a library of reference photographs, labeling, formatting of dataset files, and choosing a reference model of YOLOv8 to train with.

8.1 Dataset Construction and Labeling

For dataset construction, many pictures of the target detected item must be collected in preparation for labeling. The larger number of reference pictures allows for a more robust and accurate detection model. In this case, two data sets were created: one for leaf disease detection and another for fire detection. These two cases were chosen to test YOLOv8's ability to detect

biological issues (disease) and landscape issues (fire). Upwards of 200 photographs were gathered as a reference for both datasets.

For labeling, the program Labelme was used which can be acquired through pip install in Windows terminals (in this case Anaconda Prompt). This program allows the user to polygon around the object desired in the detection and label them with the correct identifier (label). The GUI of Labelme with a processed photograph can be seen below.

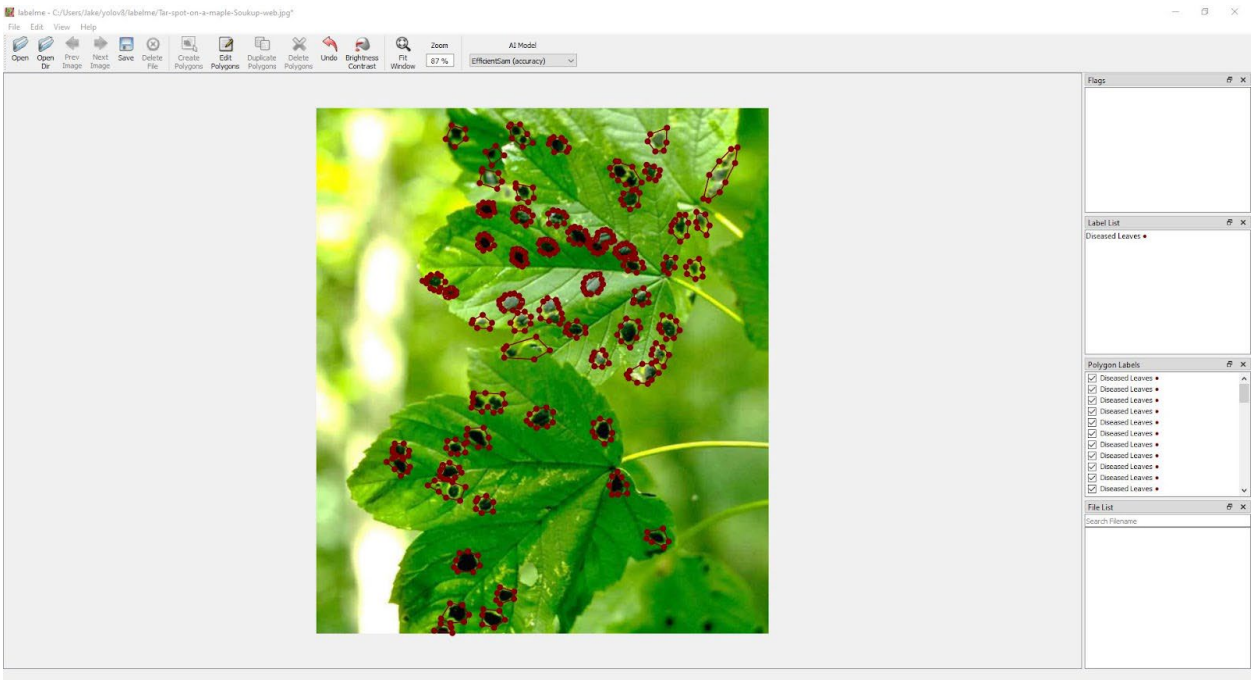


Figure 8.1: Labelme interface

After the labels are placed, Labelme will create a .json file containing information on each item labeled item and its relative position coordinated and identifier. This process is repeated for each photo in the dataset.

For formatting, YOLOv8 requires images and labels to be in separate subfolders. By default, Labelme saves the .json label files to the same subfolder. This separation can be done manually or by using Labelme2Yolo, a separate program that can be pip-installed that allows for automated conversion of a Labelme folder. Using Labelme2Yolo also automatically makes a .yaml file that contains the directory path to the files and label names needed for the training, this file will be modified later to have updated directory pathways. Pull some images (five to ten) and

labels from the current “training: and recreate the subfolders for a “validate” folder as both will be needed for the training process.

8.2 Training Custom Dataset YOLOv8 Model

The training protocol built into the Ultralytics Python library requires some additional parameters in addition to the YOLOv8 formatted training folder and .yaml file. These parameters include task, mode, epochs, data, model, image size, and batch. The task in this instance is “segment” and the mode is “train”; this indicates that this is an attempt to train a new model. Epochs represent the number of cycles/instances of training; higher epochs will result in higher training time. For this instance, the epochs value used was 100 for both models. Data represents the .yaml file used for this training linked to the dataset created using the process demonstrated in section 8.2; this will be the complete name of the .yaml file, including the file type designator. Model is the base YOLOv8 model that the new model will be based upon; the different base models are shown in Chapter 7 above. The model used for both trained models for this session was YOLOv8m. Image size is linked to the base YOLOv8 model; see Table 7.1 under the size section. Lastly, batch relates to the number of tasks being performed; this correlates to the hardware being used to perform the train, specifically GPU and tied to GPU VRAM. Since this training session is being performed on a Nvidia RTX 3080, the batch value was set to -1 for automatic 60% GPU VRAM utilization. The definitions for the input arguments can be found in official Ultralytics documentation [27].

Once training is successful, the “weights” folder of the training run folder will constrain two .pt files that represent the deemed best model and the last model from the epochs cycles. These two files are the complete trained YOLOv8 model created with the custom data set. In addition, analysis data is also presented to verify the results for the training session. The resultant data of the crop disease and fire model are included below.

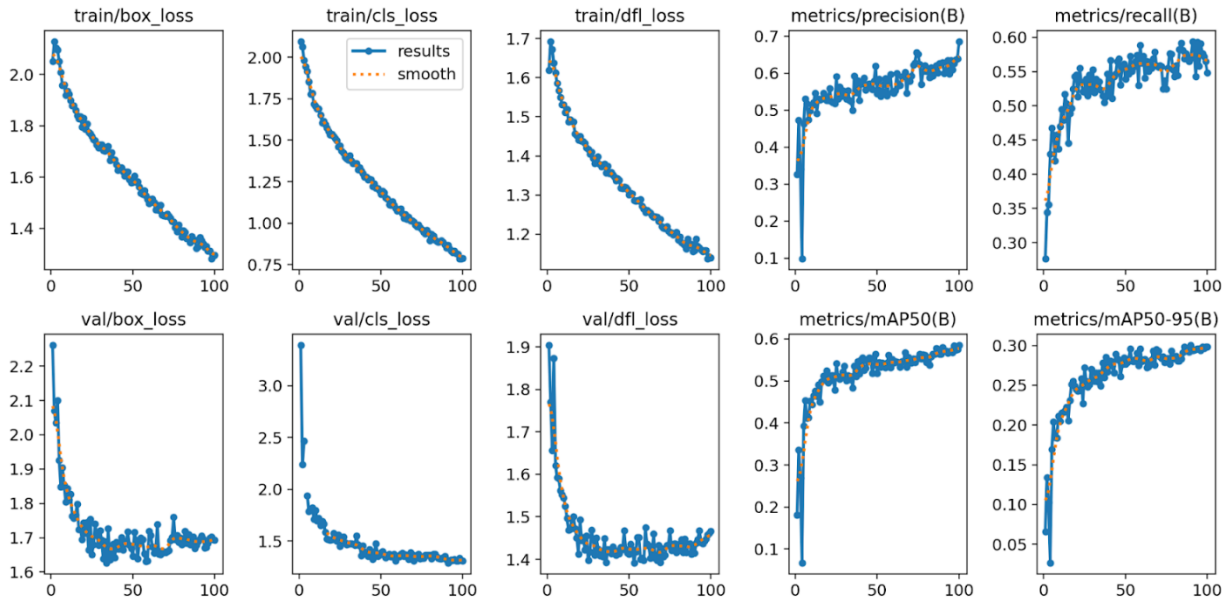


Figure 8.2: Crop leaf disease detection model post-training performance metrics

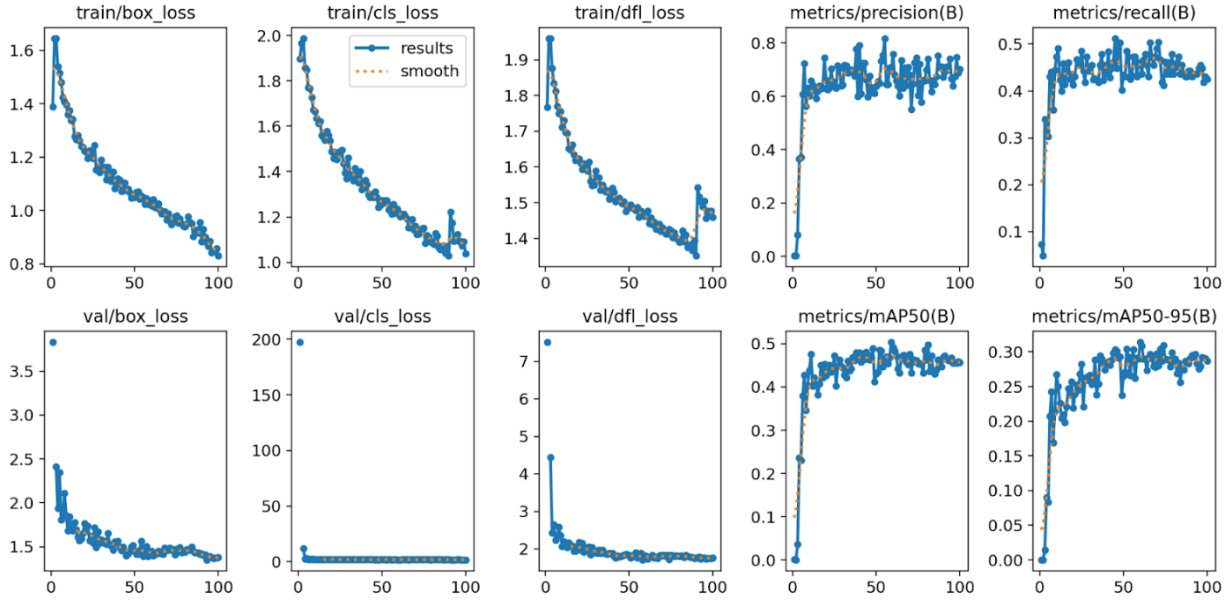


Figure 8.3: Fire detection model post-training performance metrics

When evaluating the performance metrics of both models, the main graphs to focus on are “box_loss”, “cls_loss”, and “dfl_loss”. Each value on the graph represents the performance of each training cycle in the epochs. Box loss (box_loss) evaluates the accuracy of the bounding

boxes based on regression loss and error; a lower value is better. Classification loss (clc_loss) evaluates the performance of identifying a detected object based on the error; a lower value is better. Deformable conversion layer loss (dfl_loss) evaluates the models' effectiveness in counter deformation of the detected object due to factors such as variance in photographed angle or obstructions; a lower value is better. Definitions for these performance metrics can be found in official Ultralytics documentation [28].

When evaluating the resultant metrics of the trained models for crop leaf disease detection and fire detection, all three types of loss showed a downward relationship as epoch increases in training cycles; this displays clear improvement over time with continued training.

8.3 Trained Model on Real Photographs

Using the trained leaf disease model, resultant detection runs with some example photographs are included below:

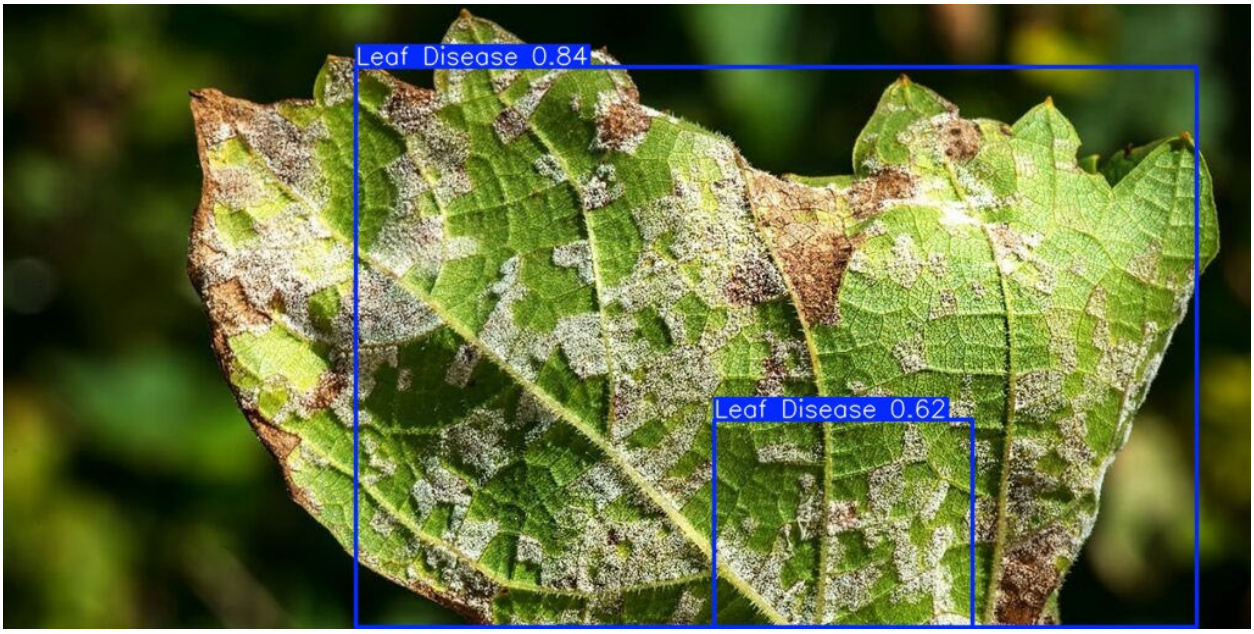


Figure 8.4: Plant disease detection 1

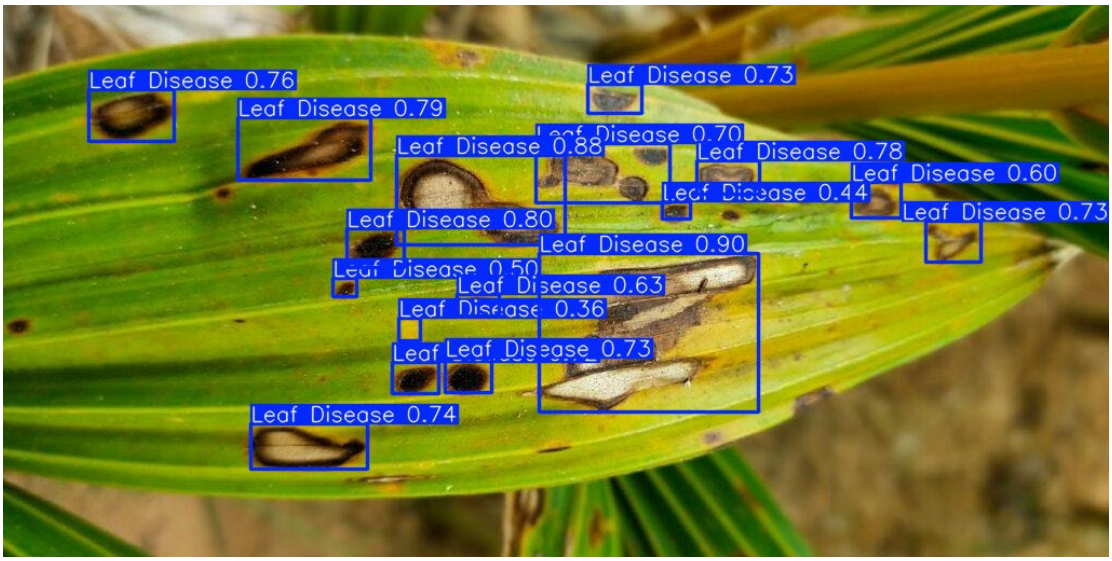


Figure 8.5: Plant disease detection 2

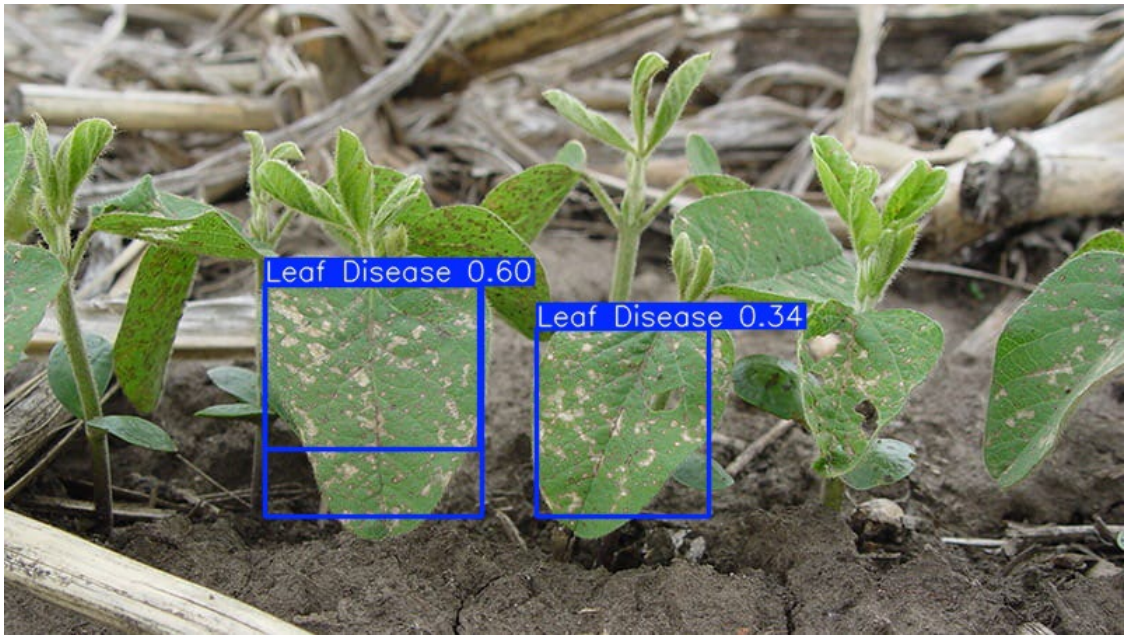


Figure 8.6: Plant disease detection 3

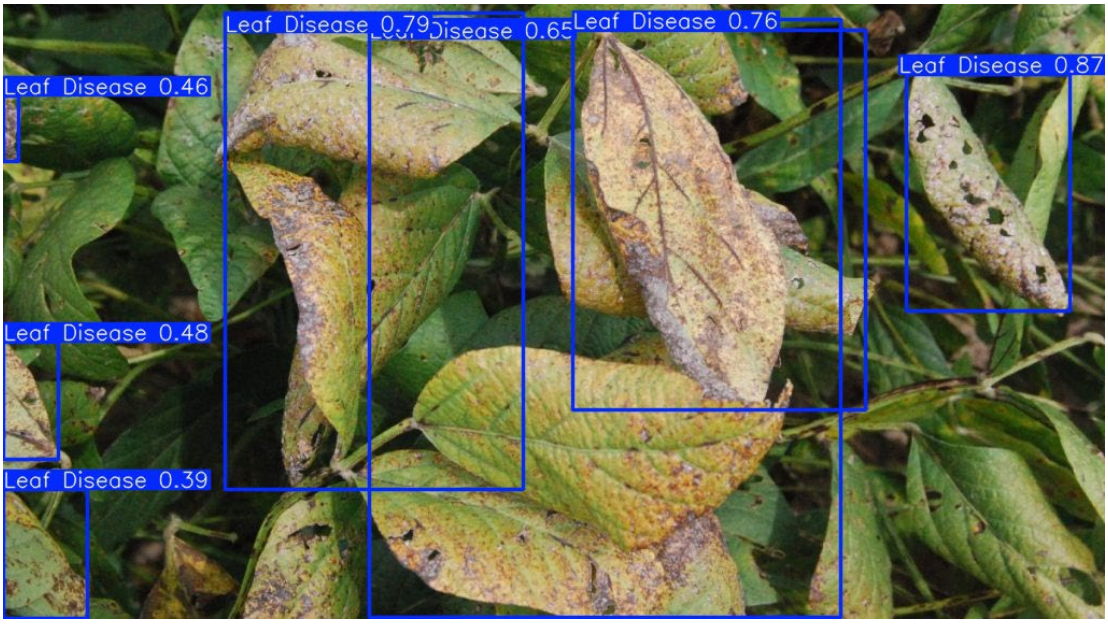


Figure 8.7: Plant disease detection 4

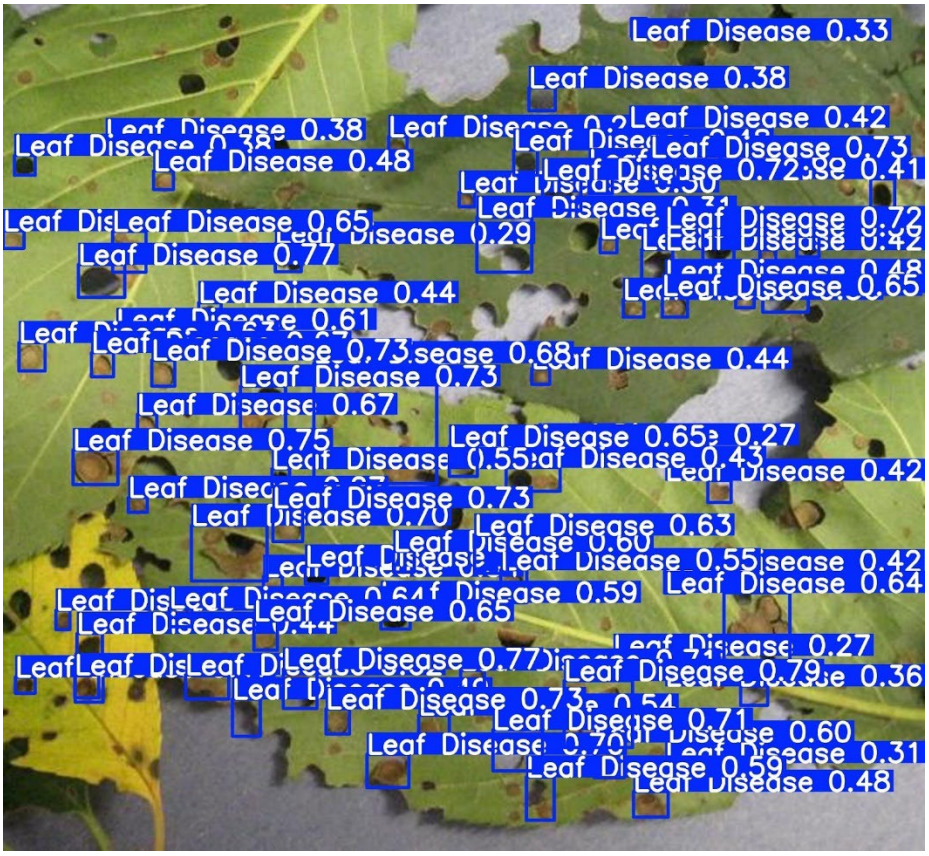


Figure 8.8: Plant disease detection 5

Overall, the results of the disease detection seemed reasonable. For the most part, the detection model was able to detect and identify the potential evidence of plant disease except in the third photograph where two of the leaves were detected. In image five, there were still noticeable instances of missed detection, but the image has an extremely high density of holes in the leaf which will result in some missed detections.

Using the fire detection model, the detection run results are displayed below:

Figure 8.9: Fire detection model result 1

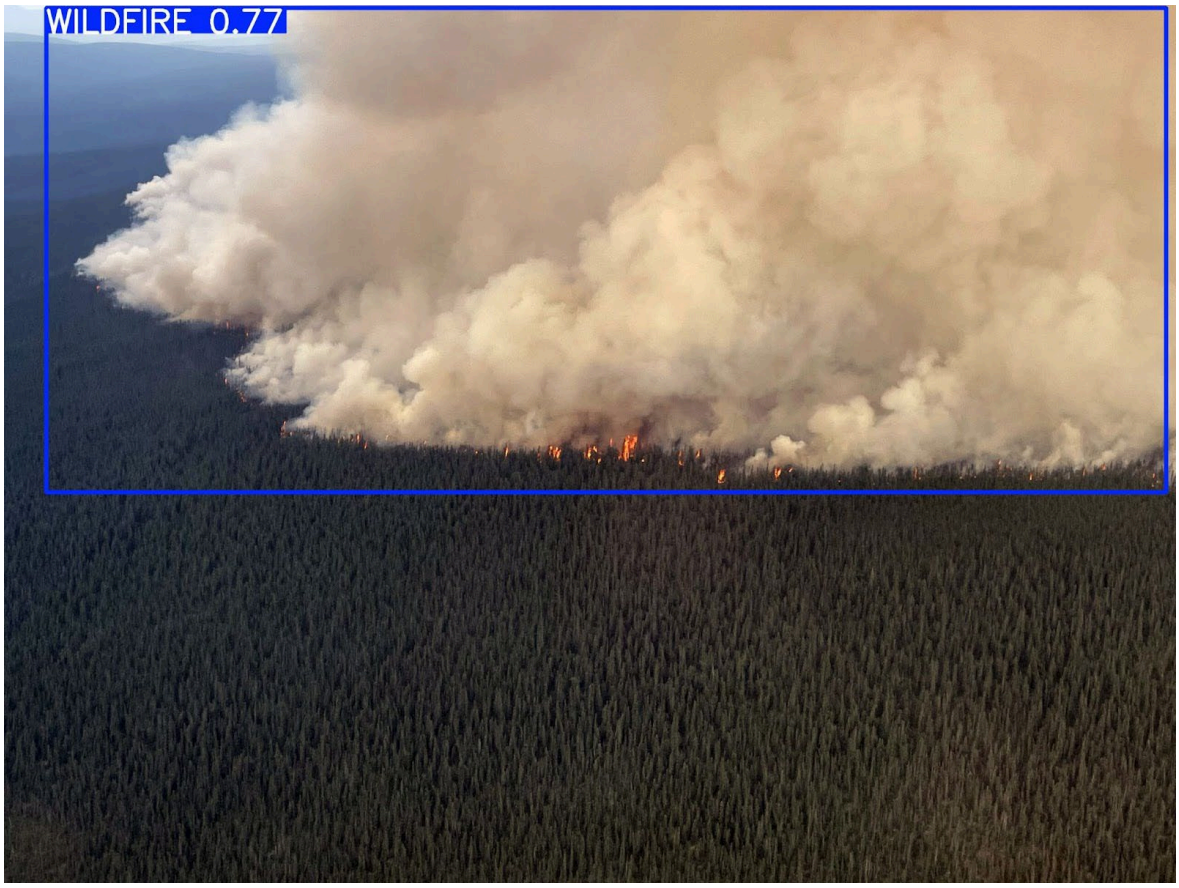


Figure 8.10: Fire detection model result 2

Figure 8.11: Fire detection model result 3

The results of the fire detection model rely much on the existence of smoke and groups that in with the identification of the fire. There could be some improvements. in a higher aerial view, these results would be acceptable.

9. Conclusion and Suggested Future Work

This project aimed to analyze the potential in low-cost drone design and machine learning object detection in agricultural use. A low-cost drone design was produced using commercially available parts that would be able to capture stable video footage and photography for machine learning analysis. The use of open-source object detection and custom dataset creation was successful in being able to identify common cases of vegetation diseases and fire detection. However, there are some pitfalls in the current environment that can prove to be challenges to the intended user. Firstly, there is a large commitment in working with developing custom datasets as each reference photo must be labeled correctly for data to be used in dataset compilation and machine learning. For the preliminary plant disease dataset created for this project, there were a total of 248 photographs analyzed and took close to a month to complete. In addition, for more specialized use cases for specific plants, there may not be enough available reference photographs available online to meet the same requirements as the custom dataset in this project. While datasets can still be successful with smaller reference data, chances of effective object detection models increase as the amount of reference data increases. There are viable alternatives to creating the users' own custom dataset as libraries of custom made labeled sets exist on the internet, but quality of mentioned datasets will vary.

References

[1] Ahirwar, S., Swarnkar, R., Bhukya, S., & Namwade, G., "Application of drones in agriculture", *International Journal of Current Microbiology and Applied Sciences*, Vol. 8, No. 1, 2019, pp. 2500-2505.

[2] Inoue, Y., "Satellite-and drone-based remote sensing of crops and soils for smart farming–a review", *Soil Science and Plant Nutrition*, Vol. 66, No. 6, 2020, pp. 798-810.

[3] Pobkrut, T., Eamsa-Ard, T., & Kerdcharoen, T., "Sensor drone for aerial odor mapping for agriculture and security services", In *2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, June, 2016, pp. 1-5. IEEE.

[4] Aabid, A., Parveez, B., Parveen, N., Khan, S. A., Zayan, J. M., & Shabbir, O., "Reviews on design and development of unmanned aerial vehicle (drone) for different applications", *J. Mech. Eng. Res. Dev*, Vol. 45, No. 2, 2022, pp. 53-69.

[5] Bright, J., Suryaprakash, R., Akash, S., & Giridharan, A. "Optimization of quadcopter frame using generative design and comparison with DJI F450 drone frame", In *IOP Conference Series: Materials Science and Engineering*, Vol. 1012, No. 1, 2021, p. 12-19. IOP Publishing.

[6] Khan, M. Quadcopter flight dynamics. *International journal of scientific & technology research*, Vol. 3, No. 8, 2014, pp. 130-135.

[7] Ebeid, E., Skriver, M., Terkildsen, K. H., Jensen, K., & Schultz, U. P. "A survey of open-source UAV flight controllers and flight simulators", *Microprocessors and Microsystems*, Vol. 61, 2018, pp. 11-20.

[8] Martin-Taylor, Zakarish. "What is a FPV Drone? (Parts List)". April 2019. Retrieved 06 March 2024 from https://risingsunfpv.com.au/blogs/helpful-guides/what-is-a-fpv-drone-parts-list

[9] ATX Airbourne. "What is ELRS?". Retrieved 07 March 2024 from https://atxairborne.com/elrs/what-is-elrs/

[10] "SpeedyBee F405 V4 BLS 55A 30x30 Stack User Manual V1.0". SpeedyBee. Retrieved 09 March 2024 from https://store-fhxxhuiq8q.mybigcommerce.com/product_images/img_SpeedyBee_F405_V4_Stack/SpeedyBee_F405_V4_Stack_Manual_EN.pdf?ref=TRONCATFPV

[11] "1200TVL Foxeer Micro Razer FPV Camera PAL NTSC Switchable 1.8mm lens 4ms Latency". Foxeer. Retrieved 10 March 2024 from https://www.foxeer.com/1200tvl-foxeer-micro-razer-fpv-camera-pal-ntsc-switchable-1-8mm-lens-4ms-latency-g-265

[12] “Happymodel OVX300 OVX350 5.8G 40ch 300mw VTX OpenVTX”. Happymodel. Retrieved 11 March 2024 from https://www.happymodel.cn/index.php/2021/07/27/happymodel-ovx300-ovx303-5-8g-40ch-300mw-vtx-openvtx/

[13] “Happymodel 2.4g ExpressLRS ELRS nano series receiver module PP RX/ EP1 RX/ EP2 RX”. Happymodel. Retrieved 16 March 2024 from https://www.happymodel.cn/index.php/2021/04/10/happymodel-2-4g-expresslrs-elrs-nano-series-receiver-module-pp-rx-ep1-rx-ep2-rx/

[14] Liang, Oscar. “How to Choose FPV Drone Motors”. May 2023. Retrieved 29 March 2024 from https://oscarliang.com/motors/#Motors-for-5-inch

[15] Tušar, T., Korošec, P., Papa, G., Filipič, B., & Šilc, J. “A comparative study of stochastic optimization methods in electric motor design”. Applied Intelligence, Vol. 27, 2007. pp. 101-111.

[16] “BrotherHobby VS 2207 1720KV/2400KV/2700KV Motor (CW)”. BrotherHobby. Retrieved 31 March 2024 from https://www.brotherhobbystore.com/products/vs-2207-1720kv-2400kv-2700kv-motor-cw

[17] “Azure Power Johnny Freestyle 4.8x3.8x3 POPO Compatible Tri-Blade 5" Prop”. RaceDayQuads. Retrieved 03 April 2024 from https://www.racedayquads.com/products/azure-johnny-freestyle-5-prop?variant=21048044585073&aff=58

[18] Grepow. “FPV Drone Flight Time: How to Calculate?”. July 2020. Retrieved 06 April 2024 from https://www.grepow.com/blog/how-to-calculate-fpv-drone-flight-time.html

[19] “Tattu R-Line Version 5.0 1400mAh 6S 150C 22.2V Lipo Battery Pack With XT60 Plug”. GenTattu. Retrieved 10 April 2024 from https://genstattu.com/tattu-r-line-version-5-0-1400mah-6s-150c-22-2v-lipo-battery-pack-with-xt60-plug/

[20] Liang, Oscar. “A Comprehensive Guide to FPV Drone Frames”. April 2023. Retrieved 11 April 2024 from https://oscarliang.com/fpv-drone-frames/

[21] “CL2-AIR 5” Frame”. RotorRiot. Retrieved 15 April 2024 from https://rotorriot.com/collections/frames-new/products/cl2-air-5-frame

[22] Bai, Q., Li, S., Yang, J., Song, Q., Li, Z., & Zhang, X. “Object detection recognition and robot grasping based on machine learning: A survey”. *IEEE access*, Vol. 8, 2020. 181855-181879.

[23] Davies, Dave. “YOLOv5 Object Detection on Windows”. December 2022. Retrieved 20 April 2024 from https://wandb.ai/onlineinference/YOLO/reports/YOLOv5-Object-Detection-on-Windows-Step-By-Step-Tutorial---VmlldzoxMDQwNzk4

[24] Ultralytics. “Ultralytics YOLOv8”. GitHub. Retrieved 3 September 2024 https://github.com/ultralytics/ultralytics

[25] Fink, J. “These are the most dangerous highways in the U.S.” Newsweek. August 2018. Retrieved 5 September from https://www.newsweek.com/these-are-most-dangerous-highways-us-1053225

[26] Herr, L. “From Tree to table: How new apple varieties are born”. Allrecipes. September 2021. Retrieved 5 September 2024 from https://www.allrecipes.com/article/how-new-apple-varieties-are-created/

[27] Ultralytics. “Train - Ultralytics YOLO Docs”. Retrieved 1 October 2024 from https://docs.ultralytics.com/modes/train/#apple-m1-and-m2-mps-training

[28] Ultralytics. “YOLO Performance Metrics - Ultralytics YOLO Docs”. Retrieved 1 October 2024 from https://docs.ultralytics.com/guides/yolo-performance-metrics/#interpreting-the-output