Machine Learning and Path Optimization-Based Autonomous Drones for Early Forest Fire Detection

Rushiv Arora
Bachelor of Science, Computer Science
Bachelor of Science, Computer Engineering
Commonwealth Honors College, UMass Amherst

Faculty Chair: Dr. Pishro Nik, College of Engineering
May 10, 2021

Machine Learning and Path Optimization-Based Autonomous Drones for Early Forest Fire Detection

By Rushiv Arora

I. Introduction

(a) Inspiration

Australia and California are witness to the straining environmental, financial, and life-uprooting effects of their raging wildfires (See Table 1 for more details). A premature detection of a small fire in the forest would have prevented its spread to the large-scale wildfire that it ultimately spread to, and this project aims to create a working prototype of an automated UAS that will, without supervision, fly over a forest and detect such fires and smoke in its early stages.

Cost of fighting forest fires in the US topped itself in 2017, with an estimated cost of 2 billion dollars [1]. Combating the Australian bushfires in 2019-2020 are estimated to have costed 690 million US Dollars. Apart from causing financial distress to the government and community, wildfires released a lot of carbon dioxide into the atmosphere thereby contributing to global warming. The prototype created in this project will assist in surveying and preventing wildfires.

Wildfire	Year	Number of Fatalities	Land Area (acres)	Suppression Costs (USD)
Australia	2019-20	33	27.2 mi.	100 bi.
California	2019	5	259,823	163 mi.
California	2018	103	1.9 mi.	> 3.5 bi.
California	2017	211	1.3 mi.	> 18 bi.
California	2016	8	669,534	> 480.3 mi.
California	2015	9	893,362	> 4.8 bi.
California	2014	2	625,540	> 204 mi.
California	2013	1	601,635	> 218 mi.
California /Nevada	2012	None reported	869,599	> 338 mi.
Texas	2011	10	4 mi	513.9 mi.

Source: Wikipedia and BBC
Note: All cost estimates are as of January of the following year

Table 1. Summary of major wildfires in the past decade

For over 50 years the practice of detecting wildfires in the United States was conducted by land surveyors who would have designated posts [2]. These surveyors would manually inspect their allotted land by sight and foot, thereby making this method slow, inefficient, and prone to human error. Fortunately, this method started to disappear in the late 1990s/early 2000s.

In 2014, Menlo Park in California implemented a drone surveillance program and since then over 180 fire departments in the United States have implemented drone surveillance programs to better rescue and combat wildfires [3]. These drones are equipped with infrared sensors which assist in search and rescue operations. The smaller size of drones is advantageous to navigate narrow/smoke covered paths and areas that are potentially difficult for firefighters and rescue operators to search. Despite all the advantages of drones in fighting wildfires, a limited number of projects extend their functionality to prevention of wildfires.

(b) This Project

This honors project is a student generated project idea that will take elements of Professor Pishro-Nik's research on Unmanned Aircraft Systems (UAS) [4] paths and combine it with prior machine learning, deep learning, vision, algorithms and software engineering experience of the student in order to create a working model of an automated drone that will scan a given region and detect early stages of disturbances of fire and then notify the respective authorities in order to curb the disturbance and avoid any further spreading of it.

(c) Scientific Endeavor and Significances

There are two main scientific aspects that will be tested by this project: Deep Learning and Vision for Fire detection in practice, and the effectiveness of Path Planning and Optimized Algorithms for Aerial Autonomy in a Forest based terrain.

The Deep Learning and Vision system will involve using a custom dataset of variable size and training/evaluating multiple models on that dataset along with applying feature extraction tricks for optimizing gain.

The Path Planning section of the project will involve programming the drone to follow an optimized path planning algorithm. The drone will be studied in grave detail to maximize throughput and area scanned by leveraging its safety and autonomous flight features.

II. Related Work

(a) Fuego

As mentioned earlier, while work has been done on combatting forest fires, not a lot of work has been done on preventing them. The main competing work that has been done is FUEGO: Fire Urgency Estimator in Geosynchronous Orbit [5], a project by Astrophysicist Carl Pennypacker and his team at the University of California Berkeley.

This project is different in multiple ways from the work being done at FUEGO. FUEGO is in close collaboration with the Fire Research Group and uses machine learning algorithms along with satellites and drones for early detection of forest fires. Preliminary tests for FUEGO involved use of Raspberry Pis attached to aircrafts to detect smoke and forest fires which was a preliminary implementation of the FUEGO unit, and an eventual transition was made into Cubesats: a theoretical satellite equipped with megapixel infrared cameras to monitor California.

A lot of FUEGO's work is still theoretical and still being developed. Further, most of FUEGO's work relies on satellites and analysis of California's forests using the infrared sensors on those satellites. The student's project will emphasize on delivering a prototype of drones that can potentially be used by local authorities to survey an area of land. Currently, FUEGO is working only with authorities in the bay area around California, where they are testing their software and satellites. The student's prototype could potentially be used by the Amherst/Massachusetts authorities, thereby going beyond California and bringing the same technology of preventing forest fires to the East coast.

Finally, FUEGO is focusing heavily on smoke detection and heat signatures using infrared sensors on satellites. The student's project will focus on image recognition for fire detection, using drones instead of watchtowers and satellites.

(b) University of Ruse - Bulgaria

Another previous work done in the field was published by D. Kinaneva, G. Hristov, et al. In their paper "Early Forest Fire Detection Using Drones and Forest Fire Detection" [11] they aim to provide a surveillance system for monitoring high risk areas for smoke.

Their proposed solution uses two drones: a fixed-wing drone that monitors fires at higher altitudes (350 – 550 meters) and which uses a optical and thermal camera to detect fires, and a rotary-wing drone that will then verify if the positives reported by the fixed wing drone are positive or negative. The fixed wing drone tends to have a higher false positivity rate due to its flight height and weather conditions. The work also involves the use of Artificial Intelligence, particularly neural networks and subfield of Machine Learning due to their ability to learn from data.

This project is different from the work at the University of Ruse in the following ways:

- No detail is given about the machine learning model, including the layer description, performance time, accuracy (training, validation, testing). The information provided regarding the model is that it is a convolutional neural network and a downloaded ssd_mobilenet_v1 model with a single class of smoke. While the use of thermal cameras is mentioned, the only class being detected is smoke and the use of thermal cameras is not elaborated on.

  In comparison, the student's work aims to provide a more detailed analysis by using state of the art models and provide more details including the layer-by-layer description of the model, the training procedure, the accuracy measures, runtime, and the scope for future. This work also extends the number of classes in the detection to include fire and smoke, giving more scope for detecting fires in their early stages. The number of the training and validation images are increased as well (by nearly 8-fold – 2600 images vs 300 images) with ideal and real images to give scope for improvement and generalization. (See Methods – Deep Learning and Results)

- The drone's used in the 2019 paper by the University of Ruse uses the ALTi transition VTOL aircraft (fixed wing) and the DJI Matrice 210 RTK with dual gimbal (rotary wing)

which have prices $ 35,000 and $ 10,000 USD respectively.

In comparison, this project will reduce the dependency on two drones and bring the price significantly lower by using cheaper more readily available drones which require lesser maintenance and area (See Methods – Drone). Further, in this project the possibility of getting false positives or false negatives by the high-flying fixed wing drone is removed by using a single rotary-based drone. Finally, this project will also make the system drone independent to a certain extent, allowing the use of any drone by DJI (the drone brand of selection).

Therefore, this project aims to not only build a fully autonomous drone that detects fire and smoke in the early stages, but it also aims to reduce the price of doing so as compared to the previous attempts made. All details including the dataset, model training and performance details, and future scope of improvement are highlighted for this project.

III. Literature Review

This literature review is divided into three sections to cover the important aspects to each individual part of the project:

(a) Autonomous Drones

In 2014 by L. Apvrille, T. Tanzi, and J. Dugelay of the telecommunications department at ParisTech [2] published a paper that highlighted their work on autonomous drones that would be able to independently scan its environment that is both, indoor and outdoor, and navigate itself around it. This is the earliest paper that can be found that focusses on autonomous drones.

The 2014 paper kept disaster management in mind as well and identified three scenarios of autonomous drone flying: dynamically scan and cover the area (fly in large fields and narrow paths), identify people and distinguish them based on age and physical condition, and be ethically compliant to be deployed internationally. To achieve this goal the authors identified two milestone technical contributions: *Environment Identification* and *People Identification*. This literature review will only focus on the former.

Environment Identification in UAVs is different than in cars as UAVs cannot be equipped with sensors due to power and weight limitations, and it cannot be done using GPS due to the static nature of GPS. Therefore, the 2014 ParisTech paper assumed that the drones only 720p cameras. To identify the dynamic environment, the authors made the drone reconstruct a 3-dimensional model of its environment by working with monocular images of it over a small modification in its flight altitude. This can be achieved in two approaches: making a *dense 3D scan* of the environment using an estimation of pixel distance in images or using sparse3D that relies on spatial locations of 100 points in the image.

A 2018 IEEE paper on real-time computing by Y. Tatsumi, H. Kawanaka and T. Koita of Doshisha University in Kyoto [6] had a different approach to this problem. They developed a path- planning algorithm for estimating the position of the UAV using probe requests. This

approach is based on the assumption that survivors in the disasters would have a wireless communication device that the drone could send the probe requests to. By measuring the distance and time of the requests, it was successfully demonstrated that this approach could be used to detect and reach survivors in a disaster.

A third and different approach was applied in 2017 by B. Alsalam, K. Morton and their group at the Queensland University of Technology in Brisbane [7]. This was one of the handful autonomous drone projects that was not built for natural disasters. Instead, they used an ultrasonic sensor and a micro-Arduino to measure the health and activity of plants and soil in precision agriculture. This paper falls under another category of autonomy called *controlled autonomy* where UAVs are used for remote sensing. In this kind of autonomy, the drones will fly over a designated area and adjust its altitude dynamically depending on the feedback it gets from the sensors. If the sensors report concerning conditions (like weeds growing) then the drone will lower its altitude and click higher resolution images.

(b) Convolutional Neural Networks

Convolutional neural networks (or any variants of them) are currently the benchmark for processing and visualizing images and they are, therefore, the ML model of choice for this project. The usefulness of CNNs in image visualization/recognition is bolstered by the two papers discussed in this literature review.

The first paper is a 2017 paper by A.M. Al Saffar, H. Tao, and M.A. Talab of the University of Malaysia Pahang [8] in which they talk about the robustness of CNNs for image classification (labelling the images). In this paper they have compiled various papers on CNNs and compare their performance and compare how introducing a CNN improved (lowered) the error in the classification.

In CNNs there are three types of layers: input, hidden and output; where the hidden layers correspond to the various features (anything that is significant/important) in the images. These hidden layers can be used to process the image(s) and extract important features in each hidden layer. Finally, all the extracted features can be used to classify the images as each class of images will have its own specific set of features relevant to it.

The authors claim that with the rise of big data and with the improvement in image resolution the CNNs require an increased number of hidden layers. This can be expressed by expressing the large image as a matrix, and each feature as a smaller matrix called the kernel matrix (activation matrix/convolutional matrix). Dot projects can be obtained by placing the kernel matrix over each possible position in the larger image matrix, and by doing this the feature can be localized to a specific section of the image to be classified. This step can be repeated for all features and all images. Ultimately the set of features pertaining to the image can be used to classify the image into one of the many possible classes.

The authors mention pooling operations for the classified features, which is the principle that if a section of the image has a feature present, then it is likely that the neighboring sections will also

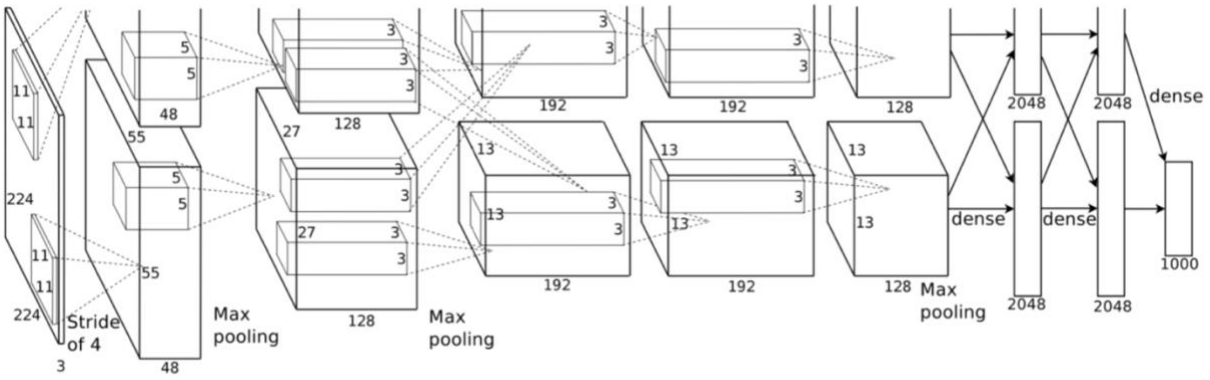

Figure 1: The AlexNet model. As mentioned by them: "An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264– 4096–4096–1000."

have the same feature. Therefore, before feeding the data from the feature extractor into the classifier, the features can be pooled to reduce the amount of data sent to the classifier.

The second paper is a 2012 paper by Alex Krizhevsky and his doctoral advisor, Dr. Geoffrey Hinton, of the University of Toronto [9] in which they introduce the world to AlexNet: a kind of CNN architecture developed by Alex that has now become one of the most famous neural network architectures for image recognition.

AlexNet was a CNN developed for ImageNet LSVRC-2012 (Large Scale Visual Recognition Challenge) in which 1.2 million 224 x 224-pixel images needed to be classified into over 1000 different classes, thereby being a computational and statistical challenge on big data and high-resolution images. AlexNet achieved the top-1% and top-5% scores in the competition which was 37.5% and 17% better than the other benchmark image classifiers. To achieve these low error rates AlexNet used eight layers – five convolutional and three fully connected. Two GPUs were used to split the work and get a faster training time for the network.

The first paper mentions that the introduction of AlexNet made the ILSVRC-2012 an "important turning point" in image classification. The paper also mentions that the winners of ILSVRC-2013 and 2014 used CNNs as well. The winning team in 2013, Clarifai, used a deconvolution of AlexNet and then visualized the outputs of each layer which brought the error rate down to 11.7%. The work done by Clarifai "deepened our understanding on why the CNN can achieve good results in image classification." The 2014 winner was a Google team that used GoogLeNet combined with the work of the previous year's winners, and they brought down the AlexNet error rate from a 17% in 2012 to a drastic 6.7% in 2014.

IV. Methods

The methods of approach will be divided into 3 sections: *Deep Learning*, *Drone*, and *Path Planning*. A final section titled *Deployment* covers the integration of the 3 section into the final product.

(a) Deep Learning

The Deep Learning section involves creating a dataset and training a model that generalizes well to real world images and can accurately detect fires from heights greater than the height of the tallest tree in the area of surveillance.

The Deep Learning Model of selection is a Residual Neural Network (ResNet) due to its ability to avoid the errors caused by the vanishing gradient problem [12] – the problem of the gradients become vanishingly small in deep learning methods that use gradient-based training and backpropagation for their training. In addition to avoiding the vanishing gradient problem, residual networks were also selected for their ability to extract features from the input data irrespective of their position [13, 14], as opposed to a traditional feed-forward neural network which would learn features relative to their location in the input data. Further, for the purposes of maximizing accuracy, the input images are scaled to be of a larger size (256 x 512 x 3/length x width x channels) so that all features in the training, validation, and testing images are well captured. This is in distinction to the MNIST and CIFAR-10 datasets which are 32 x 32 pixels which gives them limited scope for expression thereby limiting the number and quality of features that a Machine Learning model could capture on the data [24, 25].

Model	Reason for not selecting
Multi-Layer Perceptron	Prone to vanishing gradient problem
CNN	Better than MLP but still prone to vanishing gradient problem
VGG16	Not as fast and accurate as ResNet

Table 2: Alternate architectures considered

The ResNet of design has 6 convolution and 3 residual blocks, or in more detail 12 convolutional layers, 12 activation layers, and 6 pooling layers. Each convolutional block consists of a sequential arrangement of (in order) a convolutional layer, a batch normalization layer, rectified linearity activation (ReLU) layer, and a maximum pooling layer. A residual block, on the other hand, consists of two sequential convolutional blocks (minus pooling) that are added to the input of the first convolutional blocks. This architecture is better described in figure 2 and table 3. Considering the limited dataset (2646 images) and large model size, the use of ResNets was determined to be appropriate to maximize generalization and minimize errors.

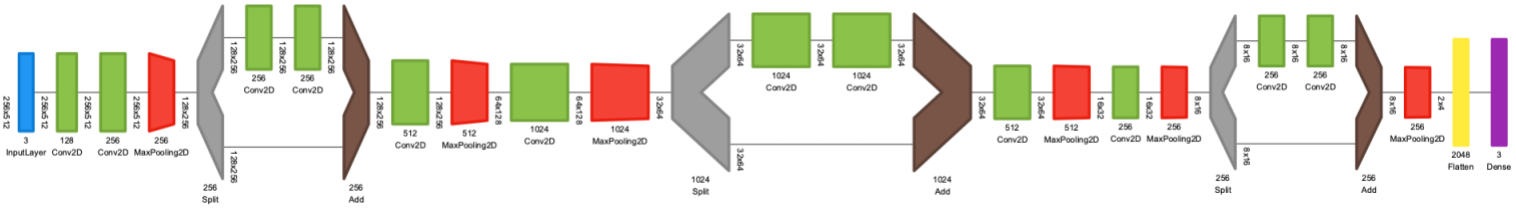

Figure 2: The architecture of the ResNet being trained. There are 6 convolutional blocks and 3 residual blocks, with the convolutional block consisting of a convolutional layer, batch normalization layer, and ReLU activation layer. Each Residual Block consists of two convolutional blocks added to the output the previous convolutional block. All normalization and activation layers have been removed from the diagram since they do not affect network architecture. Visualisation generated using Net2Vis [10]

Legend:

Layer Number	Layer	Dimensions
1	Input	[3, 256, 512]
2	Convolution	[128, 256, 512]
3	Convolution	[256, 256,512]
4	MaxPool-2	[256, 128, 256]
5	Convolution	[256, 128, 256]
6	Convolution	[256, 128, 256]
7	Residual: 6 + 4	[256, 128. 256]
8	Convolution	[512, 128, 256]
9	MaaxPool-2	[512, 64. 128]
10	Convolution	[1024, 64, 128]
11	MaxPool-2	[1024, 32, 64]
12	Convolution	[1024, 32, 64]
13	Convolution	[1024, 32, 64]
14	Residual: 13 + 11	[1024, 32, 64]
15	Convolution	[512, 32, 64]
16	MaaxPool-2	[512, 16. 32]
17	Convolution	[256, 16, 32]
18	MaxPool-2	[256, 8, 16]
19	Convolution	[256, 8, 16]
20	Convolution	[256, 8, 16]
21	Residual: 20 + 18	[256, 8, 16]
22	MaxPool-4	[256, 2, 4]
23	Flatten	[2048]
24	Linear	[3]

Table 3: A layer-by-layer description of the neural network architecture used.

The dataset consists of 2646 carefully selected images that can be divided into categories: Smoke, Fire, and Spare. The Spare dataset represents the absence of fire or smoke, which in the case of a forest would be a collection of trees, branches, soil, water bodies (such as ponds and

lakes), snow, and log cabins amongst many others. Out of these 2646 images, 987 are of Fire, 781 are of smoke, and 818 are Spare. In order to maximize the performance of the machine learning model, the images are resized to a size of 256 x 512 and then pre-processed through a chain of feature extraction techniques that consist of:

- Data Normalization: The image tensors are normalized by subtracting the mean and dividing by the standard deviation, both of which are calculated separately. [15]

  See Figures 3 – 5 for example images. It can be seen that the primary features in the images are highlighted, while the backgrounds are be ignored.

- Data Augmentation: Since the size of the dataset is relatively small to the much widely available datasets (such as the 60,000 images in MNIST and CIFAR-10), the apparent size of the dataset needs to be increased using data augmentation techniques.

  The images are 1) *padded on the right*, 2) *randomly cropped*, and then 3*) flipped with a fixed probability*. Since this is done with every epoch, the model will see "new" images with each epoch of training and hence the size of the dataset is "increased" by randomizing the data slightly. This added randomness in the data further prepares the model for deployment to the drone by exposing it to the more random nature of the images that might be captured when in use.

Figure 3: An image of fire from the dataset. On top - the original image, on bottom - the image after Data Normalization

In addition to pre-processing the data, the model is optimized using the following techniques that will prevent it from overfitting:

- Batch Normalization: The data is initially normalized by subtracting the mean and diving by the standard deviation (Data Normalization). Batch Normalization takes the same principle and apply it to each layer in the neural network in order to further extract the

features in the outputs of each layer, before feeding it into the next layer [17].

- Learning rate Scheduling: Instead of using a fixed learning rate, a learning rate scheduler is used that will adjust the learning rate after every batch of the training iteration.

- Weight Decay: Weight Decay is another regularization technique that prevents the weights from becoming too large by decaying them a little. Weight Decay is used in addition to regularization with a loss function [17].

- Gradient Clipping: In order to avoid the gradient from becoming too large in each iteration, it is clipped (cut-off) once it reached a certain value [18].

Figure 4: An image of smoke from the dataset. On top is the original image, on the bottom is the image after being flipped along the horizontal axis (part of Data Augmentation) and processed through Normalization

The model is trained for 8 epochs using an Adam Optimizer, a learning rate scheduler, with the Cross Entropy Loss Function. A train-validation split of 2410 and 236 is used with a maximum learning rate of 0.0005, a weight decay of 1e-4, and a gradient clipping of 0.1 (See Results – Pre-Deployment)

Figure 5: A batch of 8 images from the final dataset. The classes (row-wise) are: Smoke, Fire, Fire, Spare, Fire, Smoke, Smoke, Spare

(b) Drone

The drone section of the project consists of selecting and using a drone that maximizes efficiency, throughput, and effectiveness of the model while being easily controllable, programmable and safe.

The DJI Mavic 2 Zoom [19] was selected for the purposes of this project. Some of the important features (relevant to this project) are [20, 21]:

- APAS 3.0 for Multidirectional Obstacle Avoidance
- 4K Video and raw/HDR photos
- 35-minute battery life
- 10 km transmission distance and max speed of 20 m/s
- Programming SDK and functionality with Windows, Android and iOS.
- Autopilot Modes that allow for autonomously and dynamically scheduling and determining the flight route.
- Waypoint selection for 99 waypoints with curved and straight-line routes.
- Autonomously focusing and following a point of interest
- Return to home functionality to return to initial start point
- Safety Features for loss of connection.

All of the above features are used for creating the autonomous drone and pipeline. A complete list of all the important and relevant features of the drone and their use in the project can be found in table 4 below. These features were used to justify selection of the DJI Mavic 2 Zoom for the project.

Feature Name	Feature Value	Use
Programming	SDK for Windows, Android, and iOS	Programming the drone and creating custom functionality
Modes	Position, Tripod, Sport	Mode is selected dependent on battery and use case
Image/Video Resolution	12 Megapixel/4K 3840×2160	Clear image boost accuracy
Video bit rate	100 Mbps	Real-time video for instant emergency action and response
Zoom	35 mm Format Equivalent: 24-48 mm	Depending on height, zoom needs to be adjusted for accuracy. Automatically done by the code
FOV	83° (24 mm) to 48° (48 mm)	Path optimization is based on FOV calculation
Aspect Ratio	4:3 and 16:9. 4:3 is used	Calculations from diagonal FOV to horizontal and vertical FOV
Gimbal Range	Tilt: -135–45° Pan: -100–100°	Gimbal is tilted for different images
Battery	3850 mAh, 35–37-minute life	Essential for covering a large area
Maximum Run Time	2 hours and 15 minutes	Increased duration for programming and testing (idle, resting)
Intelligent Flight Battery	Battery automatically discharges to avoid swelling; Calculates lifetime; Detects temperature, overcharging, overcurrent,	Automatic calculation of remaining flight time and distance helps drone decide safety measures and RTH; Longer Battery lifetime.
Multidirectional Obstacle Avoidance	APAS 3.0 (Advanced Pilot Assistance Systems)	Avoid obstacle, especially when drone is not visible
Smart Return-to-home (RTH)	Returns to home point with obstacle avoidance	Automatically triggered when drone connection is lost, or battery is low.
Range	8 km	Forests are large – increased distance for autonomous flight
Maximum Flight Speed	20 m/s	Used when drone needs to be flown back in an emergency/low battery/large distance
Waypoint Missions	Can travel to 99 waypoints	Drone can be programmed to autonomously fly along waypoints
Intelligent Flight Missions	7	Intelligent Flight Modes are used to build autonomy
Wind Resistance	29 – 38 kmph (level 5 on Beaufort scale)	Drone should resist the heavy winds
Operating Temperature	-10°C to 40°C	Drone must be useable in cold New England Weather
Landing Protection	Drone lands only if ground is gentle/safe	Safety of the drone and its environment
Loss of connection safety	RTH, hover and land, land	Safety of the drone and its environment
DJI Simulator	Realistic simulator that connects to drone and simulates flight	Testing of the drone before in-person flight to work out bugs and ensure safety
Emergency Pause	Pause button that instantly stop all drone actions and hovers	Safety of the drone and its environment; Dealing with emergencies
Memory	8 GB internal memory, extendable by MicroSD card	The drone will gather a lot of image/video data in autonomous flight

Table 4: All the features in the DJI Mavic 2 Zoom that are used in this project [20. 21]. These features also highlight the reason for purchase.

Some notes on the above table of features are:

- Even though there is 99 waypoint limit on the waypoint mission, intelligent programming tricks are applied to override this limitation.

- The drone is equipped with a Downward Vision System and Infrared Sensing System. DJI also provides a thermal SDK [26] for its more advanced drones (Zenmuse, Mavic 2 Enterprise) which can be used to analyze infrared images clicked by the downward infrared system, and while this can be used in conjunction with the vision system/machine learning algorithm it wasn't used for the purposes of this project. This would, however, be a method of adding confidence to the results (See Discussion – Improving Accuracy)

- There are a total of 7 Intelligent Flight Modes: Hyperlapse, Quickshots, ActiveTrack (follow a moving object), Point-of-Interest (hover/circle around a POI), Waypoints, TapFly, and Cinematic. Currently only Waypoint missions are used for the autonomous flight, but there is scope for expanding to more intelligent flight modes in combination (See Discussion – Multiple Flight Modes in Combination)

- In order to ensure that the flight is safe for both, the drone and the surrounding environment, the following features are present to avoid disasters: Intelligent Smart-Battery, Smart RTH (Return-to-Home), Low-power RTH, loss of connection smart RTH/hover/land, DJI Simulator, emergency pause,

The DJI SDK is extremely detailed and provides programmability to each individual part of the drone. The breadth, functionality, and detailed documentation of the SDK was of the highest priority when selecting the drone for the project.

Drone	Important Features	Reason for not selecting
DJI Mavic 2 Pro	Programmable with DJI SDK, Identical flight performance as DJI Mavic 2 Zoom	More expensive than the Mavic 2 Zoom; No optical Zoom
Autel Evo II	360° Obstacle Avoidance, 40-minute flight time, 39 mph wind resistance,	Lesser developed SDK; Lesser support for SDK
Skydio 2	AI enabled autonomous drone with 9 deep networks, NVIDIA Tegra X2 Processor, Omnidirectional obstacle avoidance	Lower battery life of 23 minutes;
DJI Mavic Air 2	Programmable with DJI SDK, 34-minute maximum flight time	No support for waypoint missions;

Table 5: Alternate drone models considered and reasons for non-purchase

Some alternative drone options considered for the project are highlighted in Table 5 above.

(c) *Path Planning*

Path Planning and Optimization is important to maximize the output of surveying the forest by using prior knowledge of the distribution along with new knowledge that might be gained during flight. It is important the path planning algorithm covers all probable areas in a timely fashion while abstracting the details and exposing the minimum functionality to the user. The path planning algorithm in this project will take advantage of the drone's GPS navigation system along with the Waypoint Mission capabilities of the drone. Through the means of intelligent programming practices, the 99-waypoint limit of the drone will be overridden (refer Methods – Integration)

The path planning algorithm is designed to receive 4 points from the user using a map interface, such that the 4 points form a rectangular shape on the map (distorted shapes allowed). Using the maximum flight height received from the user paired with the current focal length of the camera (24 mm – 48 mm), with the default value being 10 meters and 24mm respectively, the drone will calculate the maximum horizontal and vertical distance that it will capture with the camera facing directly downwards. The calculations performed to calculate the captured frame/field of view are further explained in figure 6. As specified in Section IV Methods – Drone, the DJI Mavic 2 Zoom has a field of view of 83° (24 mm) to 48° (48 mm) which is measured diagonally. Using an aspect ratio of 4:3 and the following formulae:

$$D_a = \sqrt{H_a^2 + V_a^2}$$

$$D_a = \tan D_f/2$$

$$D_f = \arctan 2D_a$$

$$H_f = 2\arctan\left(\frac{H_a}{D_a} * \tan\frac{D_f}{2}\right)$$

where H_a:V_a is the horiztonal:vertical aspect ratio, D_a is the length of diagonal, and D_f, H_f, V_f is the diagonal, horizontal, and vertical field of view in degrees respectively. Note that D_a is in the same units as aspect ratio [22].

Once we have all FOVs, we can move to our units and get our D, H, and V values which are lengths of diagonal, horizontal, and vertical by using height h:

$$D = 2 * h * \tan D_f/2$$

$$H = 2 * h * \tan H_f/2$$

$$V = 2 * h * \tan V_f/2$$

the diagonal FOV is converted to a horizontal and vertical FOVs which are used for calculating the total rectangular area viewed by drone when facing downwards (as seen in figure 6).

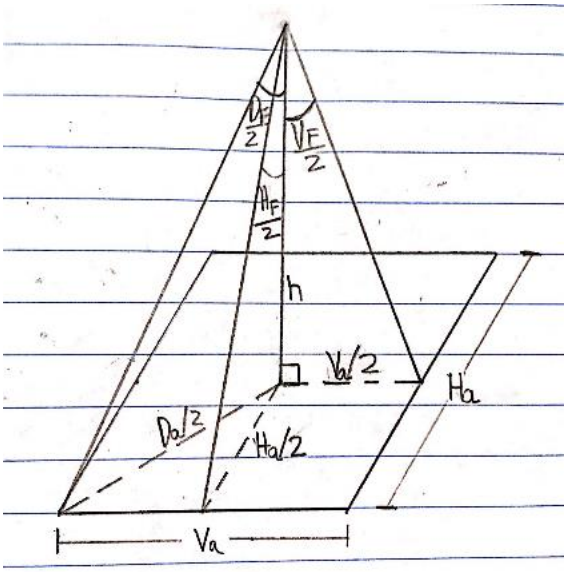

Figure 6: The calculation of the area (Horizontal and Vertical Lengths) seen by the drone by using the diagonal FOV and the height

Once the drone's field of view for a captured image is calculated, the boundary points given by the user can be broken down into a summation of multiple fields of view with the drone laying in the center of the field of view (intersection of diagonals for the rectangular field of view). Therefore, the boundary provided by the user can be broken into multiple frames and the different calculated waypoints are the geometric centers of each of the frames. This is visually explained in figure 7 and figure 8.

All distances between longitude and latitude are calculated using the Haversine Formula [23], which gives the shortest distance between two points on a sphere. The Vincenty formula was considered but not selected due to complex calculations involved [23]. The Haversine Formula is defined as:

$$a = sin^2(\Delta\omega/2) + cos(\omega_1).cos(\omega_2).sin^2(\Delta\lambda/2)$$

$$\text{c}= 2.Arctan2\left(\sqrt{a},\sqrt{(1-a)}\right)$$

$$\text{Distance} = \text{R.c}$$

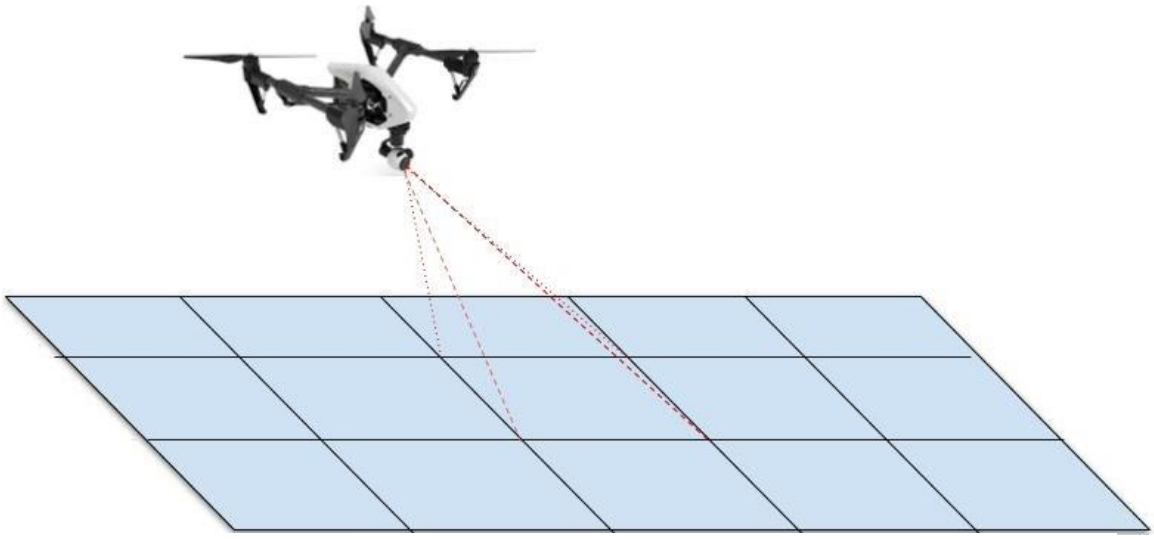

Figure 7: Multiple FOVs of the drone placed adjacently together with the drone located at the center of all of them. All adjacently placed FOVs cover the entire region spanned by the drone

Where R is the radius of the earth, with a value of 6378137 meters. Since the Earth is not spherical but rather ellipsoid in shape, some errors are expected in the calculation of the distances between two points. To minimize these errors, the closest estimation to the radius of the Earth (had the Earth been a sphere) in Massachusetts is selected. Depending on the latitude, the value of the radius can be changed to get more accurate results. The radius of the earth is independent of longitude and only dependent on latitude. The calculation of the geocentric radius given a geodetic latitude is presented below:

$$R(\varphi) = \sqrt{\frac{(a^2 \cos \varphi)^2 + (b^2 \sin \varphi)^2}{(a \cos \varphi)^2 + (b \sin \varphi)^2}}$$

where $R(\varphi)$ is the radius at the given geodetic latitude φ, a = 6378137 meters is the equatorial radius, and b = 6378100 meters is the polar radius. Source: [27 - 29]

In order to further account for loss of accuracy in calculating the distance, an accuracy rate of 90% is assumed for the Haversine Formula. Adjustments were made to the calculations to account for this accuracy assumption, and the vertical and horizontal distances between the waypoints was adjusted to be 90% of the calculated distances. This accuracy rate was arbitrarily selected, and then tested in the field on the calculated distance (without assumption) and travelled distance (with assumption) for 100 distance measures to get an average accuracy of 92.3 % with a standard deviation of +/- 4 %. Finally, in order to ensure that the entire area is covered (and avoid any mistakes in the haversine formula) the diagonal FOV is subtracted by 3 degrees before performing any distance/area calculations.

The final path optimization and use can be visualized in figure 8 that displays the selection of the boundary, the parameter selection of the flight, and the final trajectory generated based on all the calculations listed above.

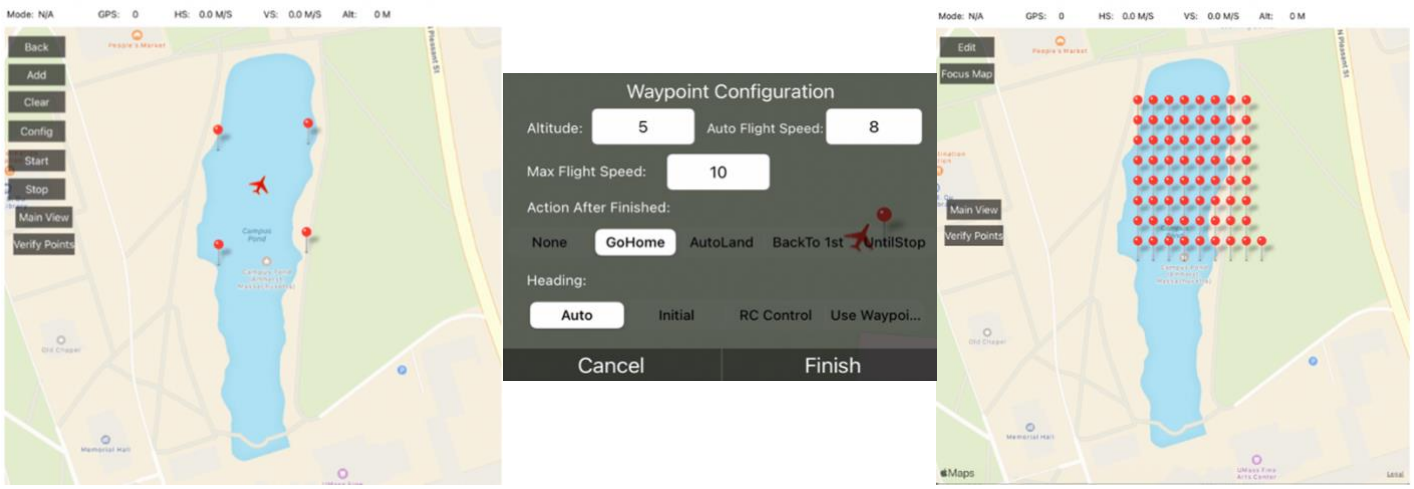

Figure 8: The autonomous path generation system in action. At the left is the aircraft location and the four boundary points, in the middle is the flight parameter window, and at the right is the generated flight path

The code for the path generation algorithm can be summarized as:

Algorithm 1: Autonomous Path Generation Algorithm

Input: Boundary $B = \{b_1, b_2, b_3, b_4\}$ a set of 4 locations
Result: Path $P = \{p_1, p_2, \ldots, p_n\}$ a set of locations for autonomous flight

$R \leftarrow$ Radius Calculated at drone latitude;
$P \leftarrow \emptyset$;
$P \leftarrow P \cup \{b_1\}$;
$H_f \leftarrow$ Calculated Horizontal FOV; $V_f \leftarrow$ Calculated Vertical FOV;
$H \leftarrow 2 * altitude * \tan(\frac{H_f}{2})$; $V \leftarrow 2 * altitude * \tan(\frac{V_f}{2})$;
$H \leftarrow 0.9H$; $V \leftarrow 0.9V$;
$D_v \leftarrow$ Vertical Distance in boundary points/V;
$D_h \leftarrow$ Horizontal Distance in boundary points/H;
$D_hCopy \leftarrow D_h$;
for $i \leftarrow 1$ ***to*** $Ceil(D_h)$ **do**
  $D_vCopy \leftarrow D_v$;
  for $j \leftarrow 1$ ***to*** $Ceil(D_v)$ **do**
    if $D_vCopy > 1$ **then** $value \leftarrow 1$;
    else $value \leftarrow D_vCopy$;
    $location \leftarrow$ Move North by value*V;
    $P \leftarrow location$ $D_vCopy \leftarrow D_vCopy - 1$
  end
  if $D_hCopy > 1$ **then** $value \leftarrow 1$;
  else $value \leftarrow D_hCopy$;
  $location \leftarrow$ Move East by value*H;
  $P \leftarrow location$ $D_hCopy \leftarrow D_hCopy - 1$
end
return P;

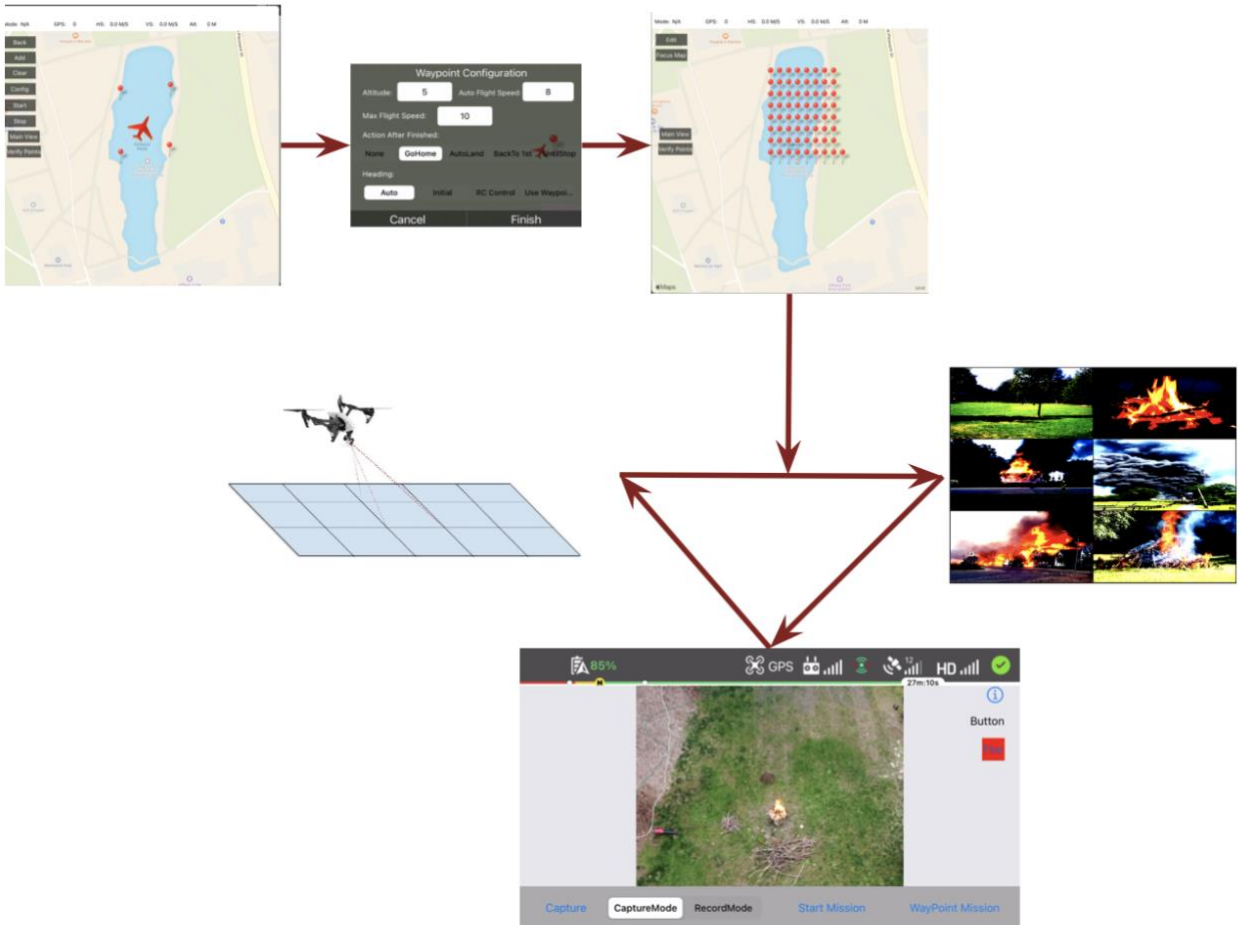

Figure 9: The integrated system put together. The system performs as: 1) Selecting 4 boundary points, 2) Allocate height, zoom, speed, completion action, 3) Verify the path generated, 4) An alternating and repeating cycle in the Timeline mission of going to a waypoint, preprocessing image, and running machine learning on it

(d) Integration

The integration of the system involved combining all the different components to create an end-to-end pipeline of autonomous flight and fire detection.

The DJI SDK Timeline Mission was used to create this end-to-end system. Timeline missions allow the user to schedule a chain of Missions (DJIMission) and Actions (DJIMissionAction) to be executed one after the other in the order they were scheduled in. To program the autonomous flight a combination of Waypoint Missions was used, while to program the Machine Learning a custom Action was created by inheriting the DJIMissionAction class. At first, the user is asked to select four points to form a rectangular area (distorted shapes allowed) that becomes the boundary of the area they would like to survey. The user is also given the option to adjust the flight height, speed, and zoom, which in combination with the boundary is used to generate the path (list of coordinates) for the autonomous flight. Each coordinate in the path is then selected to be its own Waypoint Mission, and therefore a path with *n* coordinates will be used to create a Timeline with *n* Waypoint Missions. Since every Waypoint Mission must consist of at least 2 waypoints, one for the start and one for the finish, the drone is made to ascend by 0.5 meters and descend by 0.5 meters at each point in the point to give a total of 3 waypoints for each Waypoint Mission in the Timeline.

Each Waypoint Mission in the Timeline is programmed with 4 actions that are executed in order once the drone reaches the waypoint: 1) Rotate gimbal pitch to 90 degrees to make the camera face down, 2) Shoot a Photo using the Camera, and 3) Rotate the gimbal pitch to 0 degree to make the camera face in drone heading. These captured images are accessed by the custom Mission Action that was created to run the Machine Learning algorithm on each image. Each custom action is scheduled right after each of the Waypoint Missions, giving rise to *n* waypoint missions and *n* custom actions that run alternatively in the Timeline.

The custom machine learning action inherits the DJIMissionAction class which is the base class for all Mission Control Timeline actions. The inheritance requires the creation and programming of the following inherited functions: run, willRun, pauseRun, stopRun (for this project only the run and willRun functions were fully implemented while the others were inherited and used to log to the console). The custom machine learning action executes as follows: 1) Fetch the camera, 2) Download the image using Media Download (as opposed to Photo and Video Playback), 3) Set the camera back to shoot mode, 4) Resize the image to 256 x 512, 5) Run the ML recognition on the image, and 6) Update the global variable that saves the results. All operations in this chain scheduled to be executed sequentially thereby ensuring that none of them overlap. The entire machine learning algorithm is defined in more detail in Algorithm 2.

Algorithm 2: Custom Machine Learning Action

```
Input: Signal from Timeline to start execution. Call to the run()
       function
Result: Execution of the action with call to Mission Control
class ← result of ML Recognition. Global variable;
image ← Null;
if Fetch Camera then
    if Set Media Download Mode then
        if Download Media List then
            image ← required image from list;
            if Run ML Recognition then
                result ← 0—1—2  Fire—Smoke—Spare;
                return Tell Mission Control that Execution was
                 successful;
            end
            else return Machine Learning Error to Mission Control ;
        end
        else return Media Download Error to Mission Control ;
    end
    else return Set Media Mode Error to Mission Control ;
end
else return Camera Fetch Error to Mission Control ;
```

The entire end-to-end pipeline for autonomous flight is visually described in Figure 9 while the algorithm for generating the Timeline is described in Algorithm 3.

Algorithm 3: Generating the Timeline Mission

Input: Path $P = \{p_1, p_2, \ldots p_n\}$ the path to be followed. Each point is a Latitude, Longitude pair;
Altitude h of the mission

Result: Timeline Mission $T = \{m_1, m_2, \ldots, m_2 n\}$ a Timeline Mission that will be executed by the Mission Control

$T \leftarrow \emptyset$;
$Count \leftarrow 0$;
`// Add a waypoint if Count is even, ML Action if it is odd`
for *point* $\in$ *Path* P **do**
    `// Add a Waypoint Mission`
    Waypoint Mission w $\leftarrow$ new WaypointMission;
    w.add(point);
    w.point1.height $\leftarrow h$;
    w.point1.gimbal.pitch $\leftarrow -90$;
    w.point1.camera.shootPhoto;
    w.point1.gimbal.pitch $\leftarrow 0$;
    w.add(point);
    w.point2.height $\leftarrow h + 0.5$;
    w.add(point);
    w.point3.height $\leftarrow h$;
    $T \leftarrow T \cup w$;
    `// Add a Custom Machine Learning Action`
    MLAction ML $\leftarrow$ new MLAction;
    $T \leftarrow T \cup$ ML;
end
return T;

The entire integrated interface exposed to the user is described in figure 10. The top image describes the main interface which has the following views and options: live feed, switch between camera and record mode, click an image (capture), start an autonomous mission (start mission), pick the points for the autonomous mission (waypoint mission), get drone flight information (the information button), and get a prediction on the previously clicked image (Predication). The second image in figure 10 is the map view that opens up when you press the waypoint mission, and this view has options that allow a user to edit, add, select, and configure the parameters of the autonomous flight while also viewing the path generated based on their parameters. The final image in figure 10 is the information view that displays the state of various working systems in the drone to ensure that the drone is in a flying condition before takeoff. Apart from providing a preflight checklist, this view also provides an interface to change certain

parameters of flight such as editing the maximum flight distance, modifying the camera settings, formatting the memory, and calibrating the different systems.

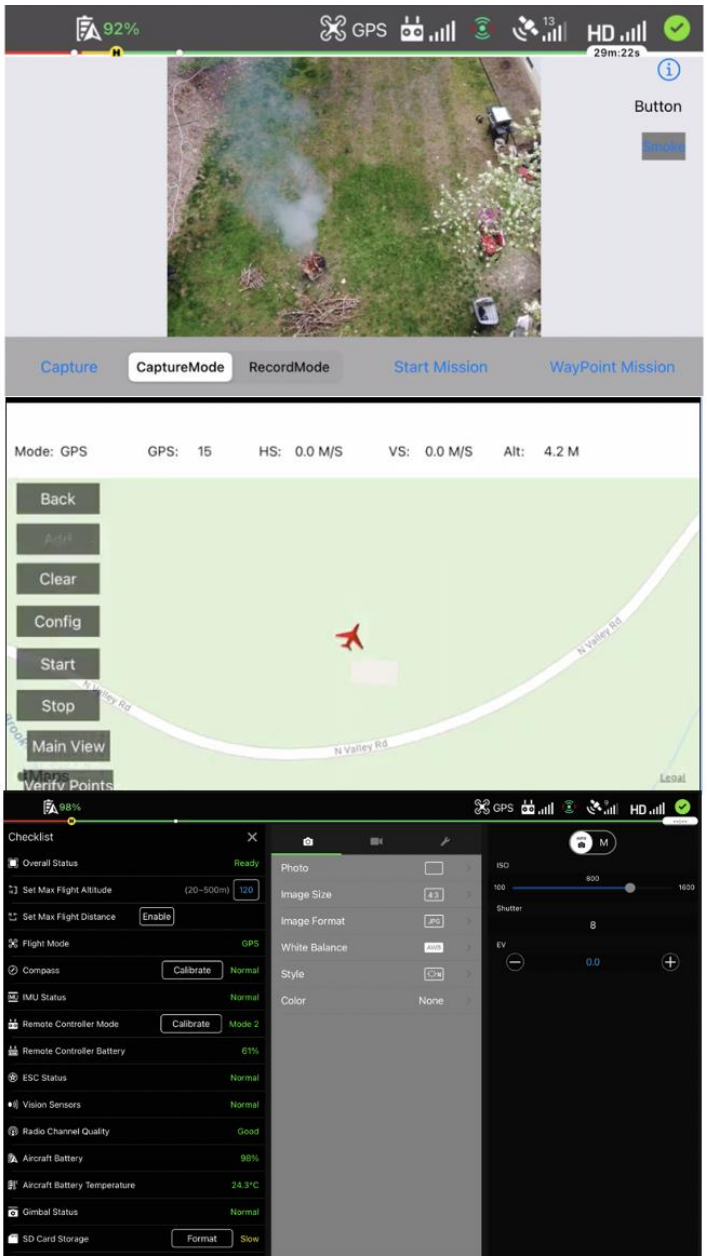

Figure 10: The views of the integrated system. On top is the primary view with the various controls to switch between defining the mission and testing it, in the middle is the first page to the path optimization and selection view (see figure 8), and on the bottom is the information and settings page for adjusting the camera settings and conducting a preflight checklist

V. Evaluation

The completed autonomous drone was evaluated on real fires and smoke. Communication with the local (Amherst and Pelham) fire authorities were established to discuss the feasibility of testing, and it was been determined that the testing will occur in three phases:

(a) Campfire Phase

This is the early form of testing because it will be on a campfire which is relatively small in size.

The drone will fly at a lower altitude and slower speed to simulate the flying over a large forest (in which case it would fly higher and faster). The Path planning algorithm will be run and all aspects to the path planning will be tested, including the dynamic flight adjustment, safety features, and focusing on areas of interest.

This evaluation mode is classified as definite since we know it can be tested. All communications for safely starting and evaluating a campfire have been made. The drone will be evaluated in safe and social distanced manner at two backyard campfire spots (with registered campfires).

(b) Controlled Forest Fire Phase

Once the campfire testing has been conducted and minor details worked out, the drone would be tested on a real, simulated, and controlled forest fire. Communications were made with the Amherst and Pelham Fire authorities who provided access to conduct a simulated burn in the backyard of a firefighter associate during open burning season.

(c) Smoke Phase

Alongside the testing for fire, tests for smoke were also conducted. For the purposes of generating smoke fresher materials were ignited in the fire including grass, freshly cut plants (not specially cut for the project, but fallen/recently dead), and fruit and vegetable scrapings amongst many others.

Please note: Due to special considerations for the environment and the climate crises we face, special care was taken to ensure that no trees or plants were specially cut/deforested for this project but that all materials burned were readily available materials that naturally fell/dropped from the trees.

VI. Preliminary Tests and Results: Observations and Modifications

Before the final tests were performed and the final results were obtained, 5 preliminary tests were conducted on the system to check end-to-end functionality and detect any issues that might potentially arise when rigorous testing is conducted.

These 5 preliminary tests are briefly discussed below:

(a) Test 1

This test was performed in the simulator provided by DJI in their DJI Assistant 2 for Mavic Software. The drone, even though connected to the simulator, will move and behave exactly as if it were flying. The only difference lies in the propellors, which move virtually in the simulator

rather than physically.

The end-to-end system was testing in the simulator, and it worked as expected. Once the boundary of the flight was selected, the autonomous path was generated, and the drone flew to each point as expected. At each point the gimbal rotated to make the camera face downwards, an image was clicked, and the gimbal returned to its original position facing upwards. Since a timeline mission was used the drone ascended and descended at each point by 0.5 meters (refer Methods – Integration). After the waypoint mission was completed, the machine learning action was run, and the result was printed on the screen (the result was fixed in order to make sure that the correct value is being returned and printed). All Waypoint Missions and Machine Learning Actions occur sequentially and alternatively.

(b) Test 2

The second test was performed in a large field with no fire to test if the autonomous flight works correctly and if it progresses to the real world from the simulator as well. As seen in the simulator, the process ran as expected and the drone was able to autonomously select points to visit and evaluate the Machine Learning algorithm on.

The second test led to the conclusion that the user must be allowed to more dynamically select the necessary parameters of flight for weather conditions such as wind, and as such modifications were made to the code to allow the user to have more direct control over parameters such as flight speed, flight height, and end of flight action.

(c) Test 3

The third preliminary test was the first test with fire, and it exposed an important improvement to the autonomous system: for the purposes of testing, it is important to have manual control over the drone's flight and hence direct access to the Machine Learning algorithm. This is because it is infeasible to test the drone on hundreds on images when using the autonomous algorithm since confidence bounds are required for testing and running the autonomous algorithm each time while generating a fire would be difficult. This is also useful in crowded areas or areas with a strong presence of trees and houses where it might be important to have manual control over the drone for the purposes of testing.

As such, it was determined that the user (in this case the author – Rushiv Arora) must be given manual access to the drone and the Machine Learning algorithm for the sake of testing.

(d) Test 4

Building on the third test, a feature allowing manual inspection and access to the Machine Learning algorithm was exposed to the user. With this added feature, the drone was manually flown over fire and the camera was adjusted to face downwards. The images were clicked, and the drone was able to detect fires up to 9 meters (29.53 feet) of height.

Due to the strong presence of mulch on the ground with dead and gray grass from winter and drought conditions, the drone had a little difficulty identifying the grass as part of the Spare class and identified it as Smoke. This gave rise to the hypothesis that the Spare dataset was too green and ideal, and therefore it had to be trained a little more on real world Spare data such as branches, snow, dead grass and leafless trees. To further support this hypothesis, the drone was elevated to 30 meters of height, and it correctly classified the mountains and green trees as Spare for all 50 times that it was tested on. It was therefore concluded that the Spare dataset needed to be optimized further to reflect the real world more.

*Figure 11: The test images for the Spare class in Preliminary Test 4. On top is an image of mulch and leafless trees that was **initially** misclassified. On the bottom is one of the 50 images used to test the hypothesis of the ideal nature of the Spare dataset. See Preliminary Test 4 and Results – Pre-Deployment for more detail*

(e) Test 5

Armed with all the results and knowledge from preliminary tests, the Machine Learning model was further trained on the additional images of fire, smoke, and spare clicked with the drone along with more accurate images of what it might experience in the real world (See Results – Pre-Deployment for more details on how the data was selected to avoid bias and overfitting while still improving accuracy). Confidence intervals were generated to ensure consistency in training and that the model was not trained on a lucky seed. The drone was tested on fire, smoke, dead grass, and water again and it returned accurate results (refer Results – Deployment).

With these optimizations and modifications applied to the drone, the end-to-end system of autonomous early fire detection was ready to be tested to generate final results.

VII. Results

(a) Pre-Deployment

Before the deployment of the drone and the end-to-end system for testing with campfires and controlled fires, individual parts of the system were evaluated, and performance was measured. The three aspects primarily evaluated are: *Machine Learning, Drone Movement, and Detection Time*.

Machine Learning: The Machine Learning model performed well on training and validation data with their respective sizes being 2410 and 236. The parameters used for training are:

- Maximum Learning Rate: 0.0005
- Loss Function: Cross Entropy Loss
- Number of Epochs: 8
- Gradient Clipping: 0.1
- Weight Decay: 1e-4
- Optimizer: Adam
- Train Size: 2410
- Validation Size: 236

The model was trained for 30 seeds and the validation accuracy, validation loss, and training loss were recorded for each seed. The average validation accuracy at the end of the 30 seeds was 93.8% while the average training and validation losses were 0.5857 and 0.6061 respectively. The plots of the average metric vs epoch, with the standard deviation for confidence intervals, can be found in Figure 12. The plot for the learning rate vs batch (determined by scheduler) can also be found in Figure 12.

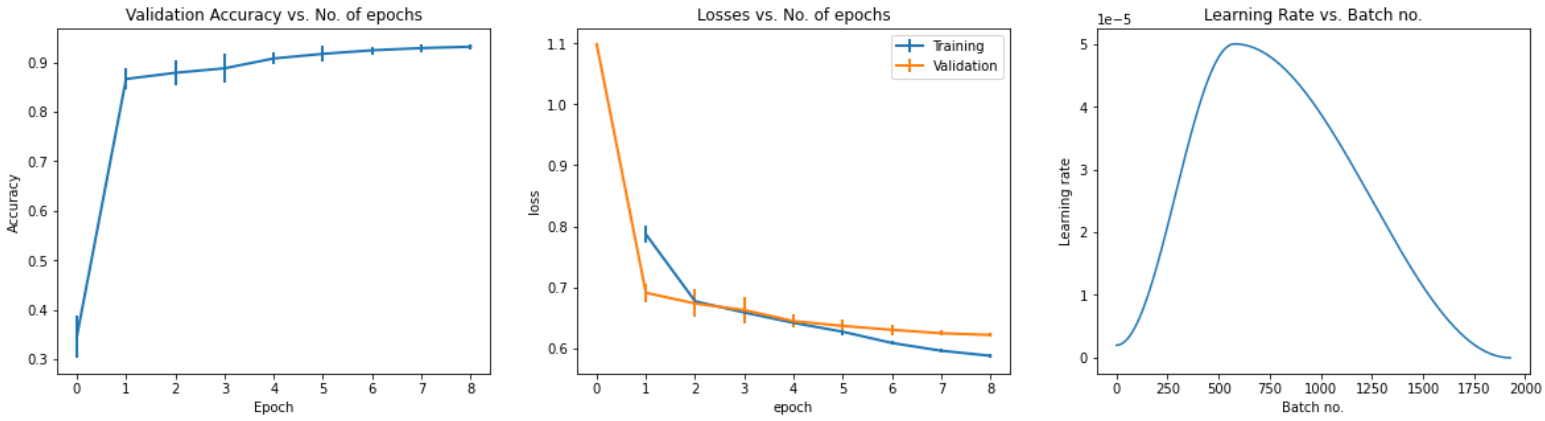

Figure 12: Training and Validation plots for the model over 30 seeds. The average values are plotted while the standard deviation is used for providing confidence intervals. The plots included are: 1) Validation accuracy vs epoch number, 2) Training and Validation loss vs epoch number, and 3) Learning rate vs batch number for training.

A lower learning rate was preferred due to the nature of the data, and it was found that a larger maximum learning rate would cause jumps around the global minima (causing the train/validation losses and accuracies to alternate between low and high). Due to concern of overfitting to the training data and not generalizing to real world data or falling in a local minima, the dataset was mixed with roughly 95 images per class (to give a total of 289 images) of images of fire, smoke, and spare collected by the drone on a controlled fire. These 289 images were in accordance with the results of the preliminary tests. To avoid overfitting by training and testing on the same images and to provide authenticity to the test results, the 289 images by the drone (and added to the dataset) were all collected at a one-time location that was not used for testing. Once the training was complete the model was validated again on the 289 images to give an accuracy of 94.83 %.

Drone Movement: As stated in Methods – Path Planning, the haversine formula is being used to calculate distances between two points. However, due to the Earth being an ellipsoid instead of a sphere, there is a loss of accuracy when using the Haversine formula. This loss of accuracy is avoided by 1) calculating the earth's spherical radius at the drone's latitude, 2) Assuming a 90% accuracy with the haversine formula, 3) lowering the drone's diagonal FOV by 3 degrees. In order to add more confidence to these claims and quantify the accuracy and confidence in the drone's movement, the drone was asked to move between two given points (decided by the algorithm) and the difference between the intended travel distance (distance to be moved) and observed travel distance was calculated. This was repeated for 100 times to get an average accuracy of 92.3 % with a standard deviation confidence of +/- 4 %.

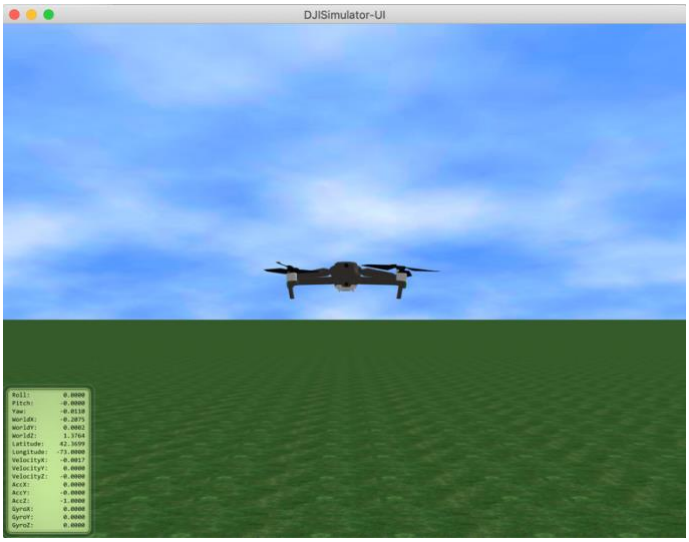

Figure 13: A screenshot of the DJI Simulator on a Macintosh. The bar on the bottom left prints all the flight related metrics received from the flight controller. The Windows simulator is more advanced and realistic, but the essential features for this project were accessible in the Macintosh simulator.

Measuring distances is arbitrary and prone to human error, and so the human measurement part of the project was reduced/eliminated as much as possible. This was done by using the DJI Simulator for testing which connects to the drone via a USB-C cable and flies the drone (in the simulator) as if it were flying while unconnected. The simulator provided access to real-time data such as roll, pitch, yaw, propeller rotation speed, distance travelled in (X, Y, Z) directions, speeds in the (X, Y, Z) directions, latitude, longitude and more. The distance travelled measurement is accurate and would represent the straight-line distance travelled by the drone if it

were unconnected, and by using these distance measurements the human error is avoided. To further avoid human error, the points between which the drone moves was fixed to be standard distances such as football fields, regulation swimming pools, and so on. An image of the drone in the simulator and the metric being printed can be found in figure 13.

Detection Time: An important part of the process is the detection time, or the time spent at each waypoint. This is time measured from when the drone arrives at a given point, to when it leaves it. It was found that since the Machine Learning processing is done on the mobile device, the detection time varies from one device to another depending on the processing power. However, the arrival at a waypoint, rotating the gimbal, clicking an image, and moving vertically (to provide 2 waypoints at each location, see Methods – Integration) is dependent on the drone and was independent of the mobile device.

The total time a drone spent at each waypoint was averaged to be 19.6 seconds using an iPhone 8 (as measured over 50 trials). The average time spent by drone in all mobile independent tasks was found to be roughly 12.2 seconds in those 50 trials, while the average time spent on the Machine Learning task was roughly 7.4 seconds (on an iPhone 8). The different individual and total detection times as measured on 4 different mobile devices, and the results are summarized in Table 6.

Device	Time on mobile-ind. task	ML Calculation	Average time
iPhone 8	12.2 s	7.4 s	19.6 s
iPhone SE 2020	12.2 s	4.1 s	16.3 s
iPhone X	12.2 s	5.9 s	18.1 s
iPad 4 Simulator on Mac XCode	12.2 s	20 s	32.2 s

Table 6: Average time taken by the system at each point. All averages have been taken over 50 trials.

It can be seen that the stronger the processing power on the mobile device, the faster the calculations are, and the lesser time spent on each point. This can be attributed to the fact that the iPhone 8 uses the A11 Bionic Chip which has 4.3 billion transistors while the iPhone SE has the A13 Bionic Chip which has 8.5 billion transistors. The iPhone XR on the other hand has the A12 chip with 6.9 billion transistors and takes roughly 5.9 seconds to execute showing a linear relationship between the number of transistors on the Apple Bionic chip and the time taken for the execution of the Machine Learning algorithm [30-32].

(b) Deployment

Once the drone was optimized and deployed it was tested on fire and smoke to check for accuracy. In total 360 images were clicked with there being 120 images for each of the classes (Fire, Smoke and Spare). All images were tested between heights of 10 meters to 20 meters, with a separate limitation testing also conducted after to test the maximum height of recognition. All tests were conducted on two separate firepits that the drone had never been trained/validated on before (See Results – Pre-Deployment). Depending on the availability and nature of the area surrounding the firepit, a disturbance level was added to each testing image which measured how much "disturbance" is being added from the surroundings. For instance, when testing for fire or smoke the drone camera was adjusted to display half or full of the surrounding tree in order to confuse the machine learning algorithm to determine if the image was Smoke or Spare, and this testing was conducted for varying amounts of environmental disturbance (See Figure 14).The

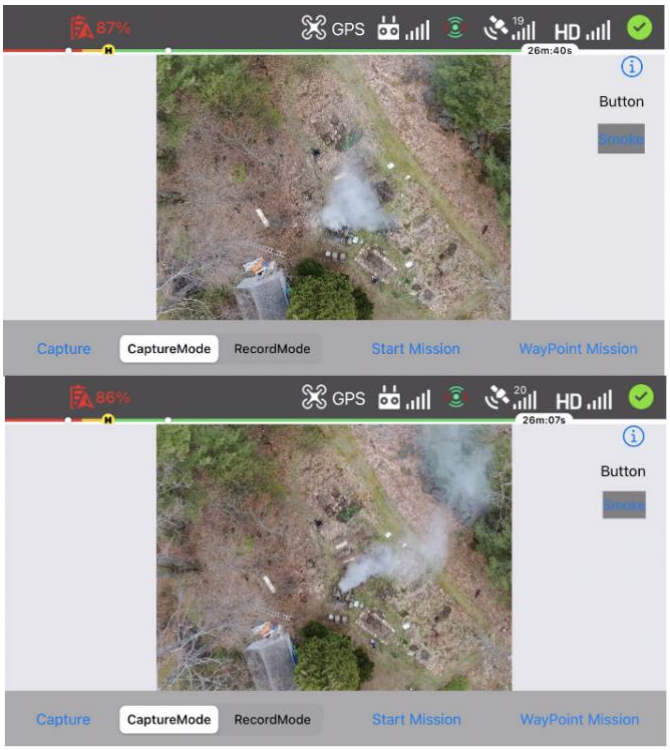

Figure 14: Two images of smoke being detected by the drone from a height of 25 meters (82 feet). The disturbance here is presented by the three green trees that represent the spare class, showing that smoke is predicted despite the trees

individual accuracies for the classes were 74.3%, 94.2 % and 98.3 % for the classes Fire, Smoke, and Spare respectively. The overall accuracy was 89%. The largest distance for detecting fire was 20.2 meters (66.3 feet) while the highest distance for detecting Smoke was 31 meters (101.7 feet). The individual discussion of results (and the slightly lower accuracy of the fire dataset) is described below and, in the Discussion – Machine Learning Accuracy:

Fire: Out of all the classes, Fire had the lowest accuracy (but highest precision). This was due to the nature of the fire that the tests were conducted on. For safety, and in accordance with Massachusetts state laws, the firepits used were of smaller size which made the detection of fires from larger heights of 15 meters and above difficult (both firepits produced fires of sizes 1 ft x 1.5 ft x 1.5 ft and 1.5 ft x 1.5 ft x 1.5 ft as measured). This lower accuracy was further bolstered by the fact that when viewing the fire vertically from above, it is difficult to see the flames as it is covered by sticks and logs (See Figure 15 for an example of a flame that is difficult to see). To boost the flame slightly at the moment of testing, a low-power leaf blower and combustible pine needles were used.

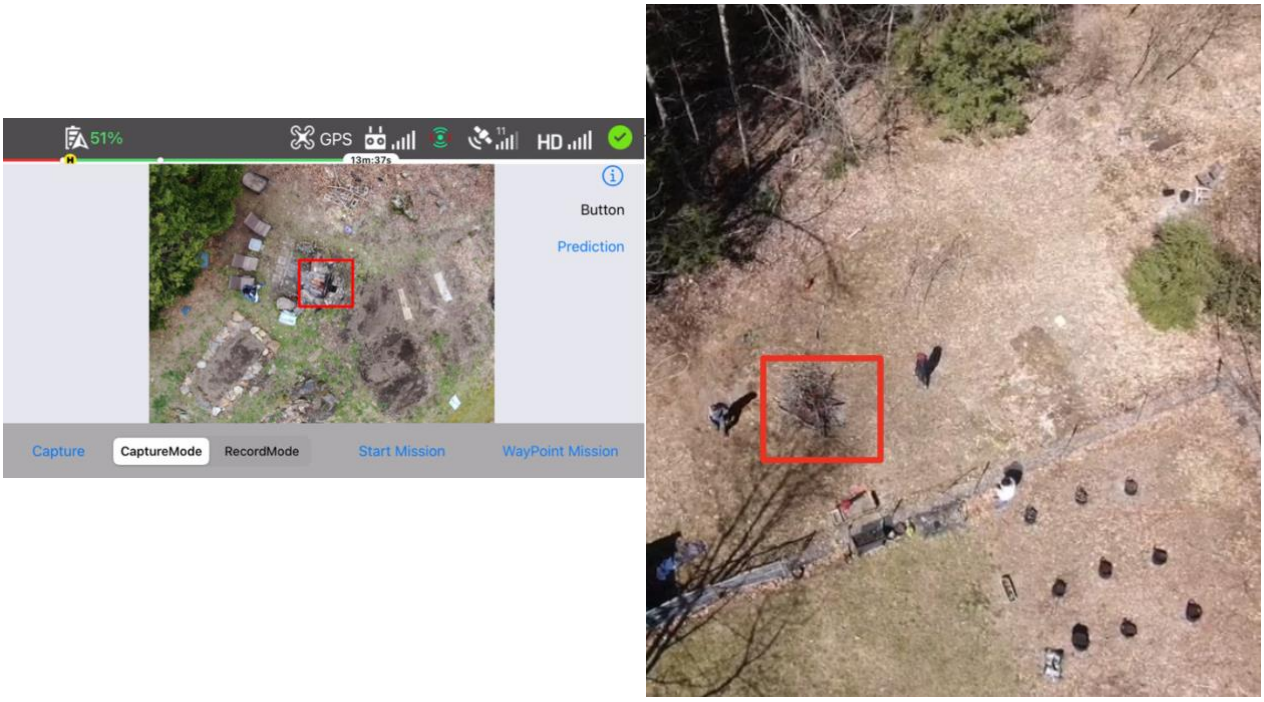

Figure 15: Two images of fire taken aerially. The flames, while visible when seen horizontally, are almost invisible when covered with sticks and viewed from above

To test this hypothesis of lower fire accuracy, more tests were conducted removing the assumptions made about the cause of the lower accuracy. 50 images were tested between heights of 6 – 9 meters and the received accuracy was 92 % (46 images correct). 50 more tests were conducted when viewing the flames from a horizontal angle at a distance of 10 – 15 meters, and an accuracy of 90 % was reported (45 images correct). For the sake of authenticity, these tests were not included in the final reported accuracies.

With a higher accuracy reported with more visible flames, the prediction is that on an early forest fire (which is significantly larger than a campfire) the results will be more accurate. Unfortunately, and not surprisingly, it is impossible to conduct an early-stage forest fire for the sake of testing. However, a single contained fire was conducted in open burning season (ending on May 1) and these images were added to the training dataset. Due to a shortage of time and expiration of open burning season, another contained fire was not performed on time.

Smoke: It is more feasible to test smoke rather than fire due to the possibility of generating larger amounts of smoke with lesser restrictions. Since more feasible tests were conducted on smoke rather than fire, the testing represented an early forest fire, and this led to the results of getting a 94.2 % accuracy with a maximum detection height of 35 meters.

The smoke for testing was generated by burning fresher materials such a fruit and vegetable remains, grass, and freshly fallen branches. A sample image and the detection can be seen in Figure 14 and in the Image Bank.

Spare: The spare dataset reported the highest accuracy with a value of 98.3 %, only misclassifying two images in the 120 images. This dataset classification was easier to test due to the presence of spare data everywhere. Some of the objects on which the test was conducted are: trees, grass, leafless trees, mulch, cars, houses, chairs and tables, and porches amongst others. A height limit test was not conducted for the Spare dataset.

Classification Metrics: The following metrics were obtained on the classification of the 360 test images:

	Precision	**Recall**	**F1-Score**	**Support**
Fire	0.99	0.74	0.85	120
Smoke	0.91	0.95	0.93	120
Spare	0.82	0.99	0.90	120
Accuracy			0.89	360
Macro Avg.	0.91	0.89	0.89	360
Weighted Avg	0.91	0.89	0.89	360

Table 7: Classification Metrics for the testing of the Machine Learning system on 360 images.

It can be seen that the Fire class has the highest precision due to the fact that it had extremely few false positives (2 to be precisive). The Smoke dataset had fewer false positives as well and hence had a precision of 0.91. The Spare class, on the other hand, has the lowest precision since many misclassifications in the Fire testing were classified as Spare when the flames were not visible, giving rise to a larger number of false positives. A highest precision for fire and smoke is advantageous as it is indicative of low number of false positives. The Fire class has a lower Recall, while the Smoke and Spare classes have a significantly higher recall which implies that the Fire class has a larger number of False Negatives while the Smoke and Spare classes do not.

The F1 scores are used to give a balance between precision and recall, and the F1 scores in this case is well balanced between the three classes with all values being 85% of above. While it is preferable to have a higher F1-score for the fire class, as discussed earlier the testing conditions did not permit to simulate an early forest fire.

Alternatively, however, if the experimental results were used where the Fire is viewing more flames from a lower height or horizontally, then an accuracy of 91% is achieved for the fire class with the metrics as seen in Table 8 below:

	Precision	Recall	F1-Score	Support
Fire	0.99	0.93	0.96	100
Smoke	0.97	0.95	0.96	120
Spare	0.92	0.99	0.96	120
Accuracy			0.96	340

Table 8: Classification Metrics for the testing of the Machine Learning system with the Alternative Fire tests when the flames were viewed between 6-9 meters or horizontally

It can be seen that the metrics here are improved and similar values would be expected if the Fire class could be tested on a larger controlled burned (while it was trained on a controlled burn, time shortages and end of open burning prevented testing it again on a controlled burn).

However, these metrics also leave for a scope in improvement (See *Discussion – Machine Learning Accuracy*, and *Improving Results*)

VIII. Discussion and Future Directions

(a) Machine Learning Accuracy

It is seen that the Machine Learning Algorithm performs well for the Spare and Smoke dataset with accuracies higher than 94 % when testing. However, the accuracy with the Fire class was slightly lower to be 74.3 %.

This can be attributed to the nature of the testing rather than the nature of the dataset and the model. On viewing directly from above, it is difficult to view a campfire due to it being covered by logs and sticks. This is in contrast to viewing the flames at an angle in which the flames are more visible. Further, all the testing for an early forest fire was conducted on a firepit of size 1 ft x 1.5 ft x 2 ft thereby yielding flames of size significantly smaller than that expected in an early forest fire.

This lower accuracy can be improved by training the model on carefully selected images of fire covered by sticks. However, better ways to improve these results are discussed in the following section: multiple images and infrared.

(b) Improving Results – Multiple Images and Infrared

While currently the Machine learning algorithm is evaluating a single image clicked by the drone, it is possible to add more confidence to the prediction by combining multiple results and different technologies together.

This can primarily be achieved in two ways: 1) Instead of evaluating the Machine Learning algorithm on a single image, it can be evaluated on 3 images clicked consecutively with the final probability of a class being the average of all the 3 images it was evaluated on. This would boost the confidence in the prediction by allowing the drone to capture the more dynamic nature of a smoke and fire and thereby avoid a false negative due to an ill-timed image;
2) Using the Infrared Sensors on the DJI Drones (if available) to boost the confidence in the prediction as well by capturing infrared images that capture heat signatures better.

Both of the above highlighted approaches provide methods for increasing confidence in the results obtained while also increasing the accuracy (and F1 score) of the different classes. The goal is to use multiple different readings in order to avoid an unlucky prediction where an image was misclassified due to uncontrollable parameters such as: wind speed and direction (affecting size of flames and smoke), size/shape of flame and smoke at the given moment, and weather conditions amongst many others.

(c) Non-Uniform Distribution of the Forest

Currently the path optimization has assumed a uniform distribution for the probability of fire in a forest, and as such the entire area marked by the boundary is surveyed by the drone in a sequential manner. This is efficient in situations where the entire area of land is the same at all points. However, it is not always the case that there is a uniform distribution for the likelihood of fire as there might be prior knowledge available for the probability of fire in the marked area.

A future direction in the project would be to implement a means of obtaining prior knowledge about the area to be surveyed and examining the entire region in decreasing order of likelihood of fire. This would make the detection of fire quicker thereby further decreasing the chances of an early forest fire spreading.

(d) Active Learning and Federated Learning

If this system were to be used on a wider scale, then it would be beneficial to include Active Learning [33] and Federated Learning [34] into the system. Both of them are Machine Learning strategies for continuous and large-scale systems to allow the system to always learn from new data rather than be a constant and fixed model that doesn't evolve over time.

Active Learning allows the learning algorithm to query a user (often called an oracle) in situations where it is usure about the classification and would like some more clarification from the oracle about what the classification result should be. Based on the feedback from the user, it will reevaluate its weights to better fit the data. Therefore, Active Learning is a form of semi-supervised learning (between supervised and unsupervised learning) in which the model is

trained on fewer data that accurately represents the dataset. Active learning also allows the model to constantly learn and improve when deployed in the real world.

Federated Learning is a form of collaborative machine learning developed by Google that "enables mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud." [35] In other words, multiple mobile devices (or by extension any device with a ML model) uses their local data on the device to improve the ML model and then the central cloud only receives the updated improved model. All the models stored on the cloud are independent of the users' data, and this protects the user's privacy since none of their data is pushed to the cloud but instead only the model (in this case developed by Google) is stored on the cloud. The working of federated learning is well described in the following excerpt: "It works like this: your device downloads the current model, improves it by learning from data on your phone, and then summarizes the changes as a small, focused update. Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model. All the training data remains on your device, and no individual updates are stored in the cloud." [35]

(e) Multiple Flight Modes in Combination: Waypoint + Point of Interest

While DJII Drone offer Waypoint Missions, they also offer 6 other missions (See Methods – Drone) two of which are Point-of-Interest Missions and Hyperlapse Mission. Once a fire is detected, the Timeline can be briefly interrupted, and the drone can be made to treat the fire as a point-of-interest for a few seconds before moving on to the next point in order to obtain better images of the fire from different rotating angles. This same can be achieved using the Hyperlapse mode to click multiple images over time before moving on to the next point.

IX. Image Bank

A lot of images were involved in this project, in both training and testing. Not all of these images were displayed in the course of the paper, and as such an image bank with some of the interested pictures is created below for reference:

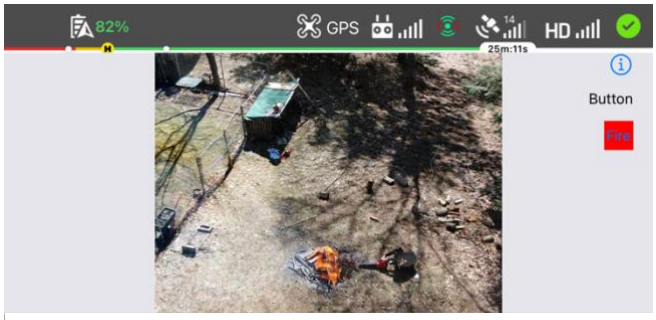

Image Bank 1: Fire being detected on the controlled fire at a distance of 18 meters (59.1 feet)

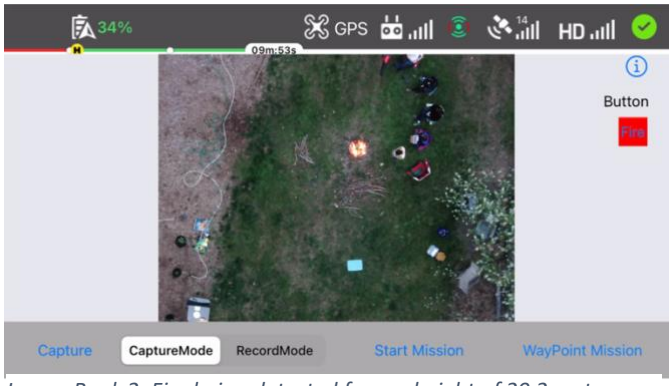

Image Bank 2: Fire being detected from a height of 20.2 meters (66.3 feet)

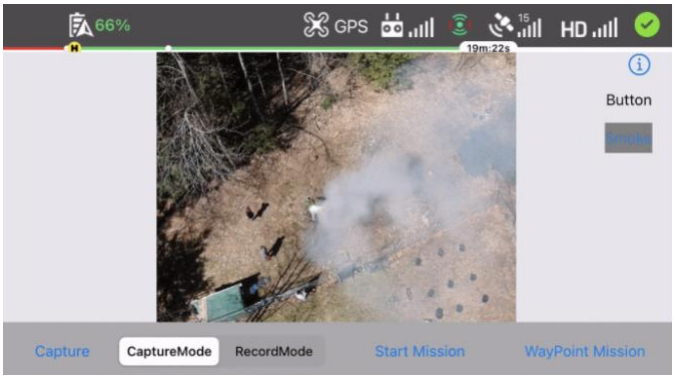

Image Bank 3: Smoke being detected from 20 meters (66 feet)

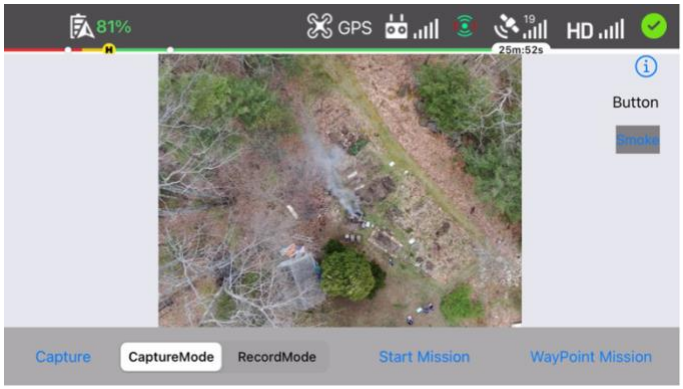

Image Bank 3: Smoke being detected from a height of 31 meters (101.7 feet)

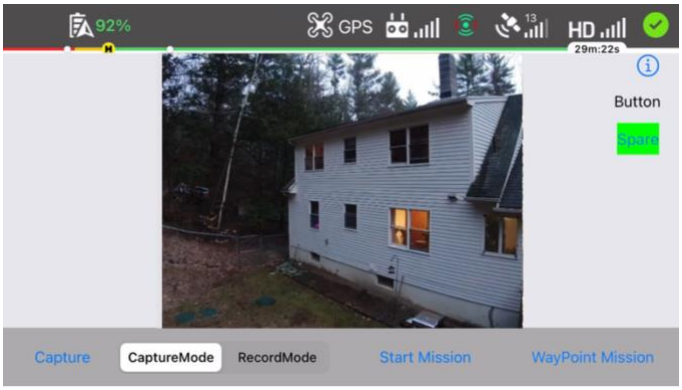

Image Bank 6: A house being detected as Spare

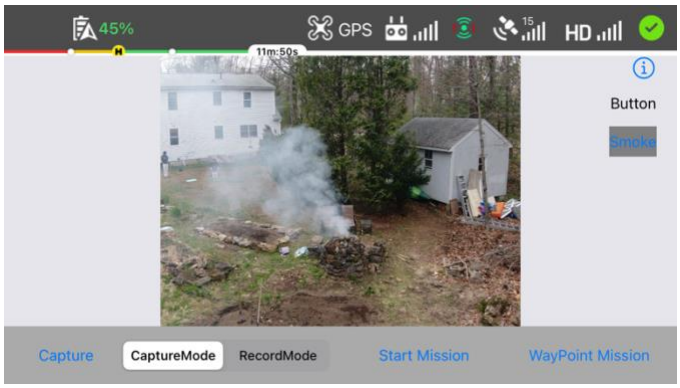

Image Bank 4: Smoke being detected from a distance of 15 meters (49.2 feet)

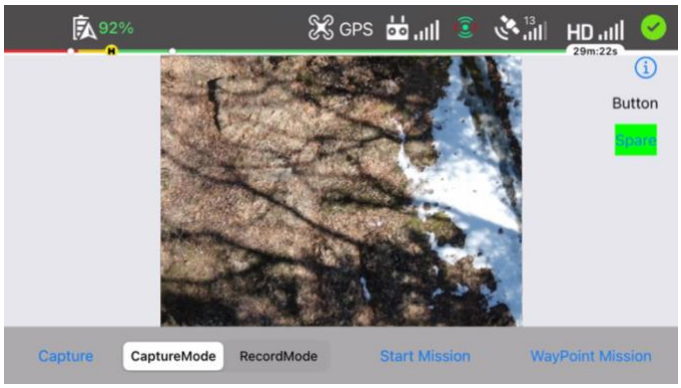

Image Bank 6: A house being detected as Spare

X. References

[1] Reuters, “Cost of fighting U.S.wildfires topped $2 billion in 2017”, September 14. 2017. Accessed on March. 17, 2020. [Online]. Available: https://www.reuters.com/article/us-usa-wildfires/cost-of-fighting-u-s-wildfires-topped-2- billion-in-2017-idUSKCN1BQ

[2] L. Apvrille, T. Tanzi, and J. Dugelay, “Autonomous Drones for Assisting Rescue Services within the context of Natural Disasters*,” 2014 XXXIth URSI General Assembly and Scientific Symposium (URSI GASS)*, Beijing, 2014.

[3] CNET, “California's fires face a new, high-tech foe: Drones”, August 27, 2018. Accessed on March. 17, 2020. [Online]. Available: https://www.cnet.com/news/californias-fires-face-a-new-high-tech-foe-drones/

[4] Saeede Enayati, Hamid Saeedi, Hossein Pishro-Nik, and H. Yanikomeroglu, “Moving Aerial Base Station Networks: Stochastic Geometry Analysis and Design Perspective.*” IEEE Transactions on Wireless Communications*, Vol. 18, No. 6, June 2019.

[5] C. Pennypacker, M. Jakubowski, M. Kelly, M. Lampton, C. Schmidt, S. Stephens, and R. Tripp, “FUEGO — Fire Urgency Estimator in Geosynchronous Orbit — A Proposed Early-Warning Fire Detection System,” Remote Sensing, vol. 5, no. 10, pp. 5173–5192, Oct. 2013

[6] Y. Tatsumi, H. Kawanaka and T. Koita, “A Path-planning Algorithm for UAV Position-estimation Systems at Disaster Sites,” *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, Hakodate, 2018.

[7] B. Alsalam, K. Morton, D. Campbell and F. Gonzalez, “Autonomous UAV with Vision Based On-board Decision Making for Remote Sensing and Precision Agriculture,” *2017 IEEE Aerospace Conference*, Big Sky, MT, 2017.

[8] A. A. M. Al-Saffar, H. Tao and M. A. Talab, "Review of deep convolution neural network in image classification," *2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET)*, Jakarta, 2017.

[9] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” *Conference on Neural Information Processing Systems*, 2012.

[10] Bauerle, A., Van Onzenoodt, C., & Ropinski, T. (2021). Net2Vis - A Visual Grammar for Automatically Generating Publication-Ready CNN Architecture Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 1–1.

[11] D. Kinaneva, G. Hristov, J. Raychev and P. Zahariev, "Early Forest Fire Detection Using Drones and Artificial Intelligence," 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2019, pp

[12] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[13] K. Fukushima, “Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position,” *Biological Cybernetics*, vol. 36, pp. 193--202, 1980.

[14] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

[15] Koo, KM., Cha, EY. Image recognition performance enhancements using image normalization. *Hum. Cent. Comput. Inf. Sci.* **7,** 33 (2017). https://doi.org/10.1186/s13673-017-0114-5

[16] Sergey Ioffe and Christian Szegedy (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR, abs/1502.03167*

[17] Anders Krogh and John A. Hertz. 1991. A simple weight decay can improve generalization. In Proceedings of the 4th International Conference on Neural Information Processing Systems (NIPS'91). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 950–957

[18] Razvan Pascanu and Tomás Mikolov and Yoshua Bengio (2012). Understanding the exploding gradient problem. *CoRR, abs/1211.5063.*

[19] SZ DJI Technology Co., Ltd., "Mavic 2 - DJI," *DJI Official*. [Online]. Available: https://www.dji.com/mavic-2. [Accessed: 10-May-2021].

[20] SZ DJI Technology Co., Ltd, "Mavic 2 - Product Information - DJI," *DJI Official*. [Online]. Available: https://www.dji.com/mavic-2/info. [Accessed: 10-May-2021].

[21] DJI, "Mavic 2 Pro/Zoom User Manual V 2.2." *DJI Official*, Jul-2020.

[22] Sensics, "Converting diagonal field of view and aspect ratio to horizontal and vertical field of view," *Medium*, 23-Aug-2013. .

[23] H. Mahmoud and N. Akkari, "Shortest Path Calculation: A Comparative Study for Location-Based Recommender System," 2016 *World Symposium on Computer Applications & Research (WSCAR)*, 2016, pp. 1-5, doi: 10.1109/WSCAR.2016.16.

[24] LeCun, Y., Cortes, C. and Burges, C.J.C. (1998) The MNIST Database of Handwritten Digits. New York, USA. http://yann.lecun.com/exdb/mnist/

[25] A. Krizhevsky, V. Nair, and G. Hinton, "The cifar-10 dataset," 2014

[26] "DJI Thermal SDK- Download Center - DJI," *DJI Official*. [Online]. Available: https://www.dji.com/downloads/softwares/dji-thermal-sdk. [Accessed: 10-May-2021]

[27] "Earth radius," *Wikipedia*, 06-May-2021. [Online]. Available: https://en.wikipedia.org/wiki/Earth_radius. [Accessed: 10-May-2021]

[28] MathWorks, "Geodetic to Geocentric Latitude," *Convert geodetic latitude to geocentric latitude - Simulink*. [Online]. Available: https://www.mathworks.com/help/aeroblks/geodetictogeocentriclatitude.html. [Accessed: 10-May-2021].

[29] Stevens, B. L., and F. L. Lewis. *Aircraft Control and Simulation*, Hoboken, NJ: John Wiley & Sons, 1992.

[30] "Apple A11," *Wikipedia*, 24-Apr-2021. [Online]. Available: https://en.wikipedia.org/wiki/Apple_A11. [Accessed: 10-May-2021].

[31] "Apple A12," *Wikipedia*, 24-Apr-2021. [Online]. Available: https://en.wikipedia.org/wiki/Apple_A12. [Accessed: 10-May-2021].

[32] "Apple A13," *Wikipedia*, 24-Apr-2021. [Online]. Available: https://en.wikipedia.org/wiki/Apple_A13. [Accessed: 10-May-2021].

[33] Settles, Burr (2010). "Active Learning Literature Survey" (PDF). *Computer Sciences Technical Report 1648*. University of Wisconsin–Madison. Retrieved 2014-11-18.

[34] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh, & Dave Bacon (2016). Federated Learning: Strategies for Improving Communication Efficiency. In *NIPS Workshop on Private Multi-Party Machine Learning*.

[35] B. McMahan and D. Ramage, "Federated Learning: Collaborative Machine Learning without Centralized Training Data," *Google AI Blog*, 06-Apr-2017.