Identifying aircraft from above

By Kai Roper-Blackman

Registration Number: 1602999

Supervisor(s): Adrian Clarke, Sebastian Halder

Second assessor: Luca Citi

Degree Course: BSc Computer Science (G400)



Contents

[Acknowledgements 4](#_Toc7133826)

[Abstract 5](#_Toc7133827)

[List of Symbols 6](#_Toc7133828)

[1. Project Aims and Objectives 7](#_Toc7133829)

[Primary Objectives 7](#_Toc7133830)

[Additional goals 7](#_Toc7133831)

[2. Methods 8](#_Toc7133832)

[2.1 Internal and External Libraries 8](#_Toc7133833)

[2.2 Computer Vision 9](#_Toc7133834)

[2.2.1 Development 9](#_Toc7133835)

[2.2.2 Applications 9](#_Toc7133836)

[2.2.3 Images 10](#_Toc7133837)

[2.3 Data 11](#_Toc7133838)

[2.3.1 Standalone data set 11](#_Toc7133839)

[2.3.2 Realistic data set 11](#_Toc7133840)

[2.3.3 Parsing 12](#_Toc7133841)

[2.4 Pre-Processing 13](#_Toc7133842)

[2.4.1 Histogram of oriented gradients 13](#_Toc7133843)

[2.4.2 Feature Vector 13](#_Toc7133844)

[2.5 Machine learning 15](#_Toc7133845)

[2.5.1 Supervised learning 15](#_Toc7133846)

[2.5.2 Support vector machines 15](#_Toc7133847)

[2.5.3 Cross-validation 17](#_Toc7133848)

[3. Technical Achievement 19](#_Toc7133849)

[3.1 Graphical user interface 19](#_Toc7133850)

[3.2 Standalone classification 20](#_Toc7133851)

[3.3 Large image search 21](#_Toc7133852)

[3.4 Saving of results 24](#_Toc7133853)

[3.5 Software design 25](#_Toc7133854)

[3.6 Project Planning 27](#_Toc7133855)

[3.7 Methodology 27](#_Toc7133856)

[3.8 Momentum 27](#_Toc7133857)

[3.9 Adapting to change and dealing with risks 27](#_Toc7133858)

[3.10 Performance 28](#_Toc7133859)

[3.11 What have I learnt? 28](#_Toc7133860)

[4. Legal, ethics and sustainability 29](#_Toc7133861)

[4.1 Sustainability 29](#_Toc7133862)

[4.2 Legal 29](#_Toc7133863)

[4.3 Ethical 29](#_Toc7133864)

[4.4 Intellectual property 29](#_Toc7133865)

[5. Conclusion 30](#_Toc7133866)

[6. References 32](#_Toc7133867)

# 

# Acknowledgements

A special thanks to my parents Annie Roper and Seon Blackman for supporting me throughout University.

Thank you to all the academic staff in CSEE, specifically Adrian Clarke, Sebastian Halder and Luca Citi for feedback and guidance throughout this project.

# Abstract

We all know what an aircraft looks like, but does a computer? A seemingly simple task that can be carried out by individuals at age two poses a complex problem to modern technology. Machine learning is a relatively new field with little research but already boasts claim to many applications such as driverless cars and facial recognition systems. The development of object recognition is the centre of many companies’ business models and objectives, making aircraft identification such an interesting topic to research.

Existing images of ground and aircraft are pre-processed using Histogram of Gradients to create feature descriptors. Feature descriptors describe the orientation of a gradient within an image subsection. Support Vector Machines are passed feature descriptors with labels for training. Once training is completed, the support vector machine accepts a test set and returns predictions. Large image search takes a large image and looks within a smaller area for aircraft. Search area parameters are provided by the user.

The results obtained from cross-validation show an accuracy of 100% when identifying standalone aircraft. However, when searching for aircraft in larger images, the accuracy drastically decreases to around 55% as some aircraft are overlooked. After optimization, the system used to identify aircraft can be applied to other identification problems with possible military and commercial uses.

# List of Symbols

|  |  |
| --- | --- |
| **Symbol** | **Meaning** |
| SVM | Support Vector Machine |
| PNG | Portable Network Graphics |
| px | Pixels |
| HOG | Histogram of Oriented Gradients |
| PIL | Python Image library |
| GUI | Graphical User Interface |
| ML | Machine Learning |
| AI | Artificial Intelligence |
| CV | Computer Vision/ Cross validation |
| IP | Intellectual Property |
| UUID | Universally Unique Identifier |
| CNN | Convolutional Neural Network |
| MVP | Minimum viable product |
| RGB | Red Green Blue |
| OCR | Optical Character Recognition |
| CCTV | Closed Circuit Television |

# 1. Project Aims and Objectives

The original aims of this project have changed since the initial report due to the change in machine learning methods. They have been tailored to suit the new approach. This was mainly down to a lack of research as discussed in chapter 3.9. The main objective is to allow a computer to differentiate between **aircraft** and **ground** images. This can be broken into several smaller objectives.

## Primary Objectives

1. To gather a set of aircraft and ground images to create a training and test set
2. To normalise the training and test set
3. Change the orientation of aircraft to ensure they face north
4. Resize the images so they can be pre-processed accurately
5. To pre-process the training set and label accordingly
6. To train an SVM with the training set
7. To test the SVM with the test set
8. Further optimise the SVM by tuning decision boundaries
9. Cross-validate the training set
10. Display results of classification to user

Assuming all primary objectives are completed successfully, additional objectives are to be attempted.

## Additional goals

1. To attempt to recognise aircraft in a large image using user defined search criteria and show aircraft locations to the user after the search is completed
2. To generate heat maps of large images to identify areas with a probability of containing aircraft
3. To attempt to recognise non-commercial aircraft such as private aircraft and helicopters

Once all goals have been completed, a software program with realistic uses should be established. The training set can be varied to identify additional vehicles or identify completely different objects.

This project is an opportunity for me to research and understand Machine learning. Machine learning is a topic often frowned upon by the media and therefore also by the public. It has been depicted in the past as a dangerous topic to develop.

# 2. Methods

## 2.1 Internal and External Libraries

During this project, I have used a variety of libraries to aid the completion of the aims and objectives of the project. Writing code to support all functionality provided by the product would be very difficult to achieve in the tight timescale given.

The language I have chosen for this project is Python as it has a large range of image processing libraries available. Therefore, it offers a wide range of features such as object orientation and detection. Internal libraries come as a standard with the python package whereas external libraries are developed by third parties. All third-party libraries used during this project are used in accordance with licensing laws and agreements laid out in their terms and conditions. This is further discussed in chapter 4.

#### OpenCV

The library used to read images from PNG format to NumPy arrays was OpenCV. OpenCV has tools for image manipulation and other image related functions; it was used to rotate images before training.[1]

#### Scikit-Learn

Scikit-Learn is a well-documented library that offers a wide range of tools for machine learning, data mining and data analysis. Scikit-learn has developed a Support Vector Machine that can be easily utilised for aircraft identification. The SVM forms the basis of my product and classifies objects as it sees fit.[2]

#### Scikit-Image

Scikit-Image is a collection of image processing algorithms for the Python programming language. This library contains the HOG algorithm used in image pre-processing to allow the SVM to distinguish between objects in images. [3]

#### FPDF

FPDF is a library that allows interaction between Python and pdf files. This allows the results of the classification to be saved and viewed by the user. This includes raw data that can be written to a table containing information about probability and image data that shows aircraft locations.[4]

#### NumPy

NumPy is the fundamental package for scientific computing in Python [1]. It comes with a variety of features and functions such as mathematical, logical and shape manipulation. This makes it the perfect library to store images as arrays. Other libraries used during this project utilise NumPy; often returning NumPy arrays after specific functions. During the training stages, images are rotated using NumPy’s rotate function.[5]

#### Matplotlib

Matplotlib is a 2D plotting library that produces publication quality figures. During this project, I used its features to display the results of classification to the user and allow them to interact with them using the libraries GUI. This allows them to zoom in and move images and plots around as they desire. The user can also choose to save the output if they wish for later viewing.[6]

#### Tkinter

Tkinter is the Python standard for GUI development. The library allows the development of complex windowed GUI’s. It allows the placement of buttons, labels, text boxes, images, radio buttons and drop-down menus. Tkinter also has GUI’s for functions such as file selection. [7]

#### PIL

PIL is python’s image library that adds support for opening and saving images in multiple formats. The library contains functions to easily manipulate images and draw shapes/ text over them. The shape functions have been utilised in this project.[8]

## Computer Vision

Computer vision is the study of how computers can interpret and obtain information from complex multi-dimensional arrays of data also known as images. It seeks to automate tasks the human visual system undertakes every day. The entirety of this project is based around computer vision seeking to demonstrate its power and complexity.

### 2.2.1 Development

The development of computer vision started in the late 1960s by universities pioneering in artificial intelligence. The goal of development was to create a system that mimicked features of the human visual system, designed to act as a stepping stone towards the creation of robots with intelligent behaviour. During the 1970s, many studies were conducted to form the basis of many computer vision algorithms such as edge extraction, labelling of lines, non-polyhedral and polyhedral modelling where many of which still exist today.

Today, the development of such technologies is primarily focused upon data set improvement and the optimisation of existing functions and learning models.

### 2.2.2 Applications

There are several applications of computer vision, some more obvious than others. Image enhancement is the most available and used repeatedly by the general population. Image enhancement is used to edit photos and videos. It could be a simple task such as placing text over an image or applying a filter.

OCR is a computer vision technique used in number plate recognition by public services. The number plate is cropped from the image and resized before each individual character is classified. Eventually, all characters are processed, and the number plate can be passed through numerous databases to help the police.

Medical applications of computer vision are currently emerging and becoming more commercially used. At present, computer vision is used in organ measurement, tumour detection, blood samples and blood flow analysis. Computer vision is also currently used to enhance x-ray and ultrasound results by removing noise from the image.

Other applications include facial recognition, CCTV and biometrics.[9][10]

### 2.2.3 Images

To a human, an image is a snapshot of light at a given moment. However, storing an image using a computer is much more complex. An image in computer science is a regularly-sampled rectangular grid of pixels. This grid is stored in what is commonly referred to as an array. Grayscale images are stored in 2D arrays whereas colour images are stored in 3D arrays. An RGB image essentially has 3 layers (arrays). Each array in the 3D structure contains the intensity values for a given channel. When combined, a colour image is formed.

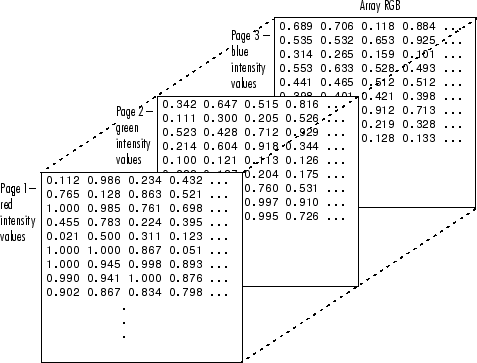


Figure 1: 3D array representing an image. Each layer represents an array containing RGB channel intensities[11]

Figure 1 shows a visual representation of a 3D array with colour intensities stored as decimals. A decimal value of 1.0 would mean a computer would turn the given pixel on at full power. 0.0 would mean the given pixel would be turned off. This allows a computer to show a range of different colours at different brightnesses. To show an image at a lower brightness, the image array would simply be multiplied by a decimal to reduce all intensity values. To reduce the brightness of an image by 50%, the entirety of the array would be multiplied by 0.5. Conversely, to increase the brightness the opposite would be done. If the array was to be multiplied by 1.5, the brightness of the image would be increased by 50%. Values that peak above 1.0 or below 0.0 are normalised to ensure they can still be displayed.

## 2.3 Data

This chapter provides a description of the data required to train an ML model, how it was gathered and how suitable images were selected.

For a computer to recognise an object, first you must teach it what the object looks like. This can be accomplished using different learning techniques and methods. The technique I chose to implement was supervised learning. To carry out supervised learning, a training set must be created. The training set must show an accurate representation of the object, or the model will fail to understand what it is identifying.

### 2.3.1 Standalone data set

The standalone data set currently consists of 100 images of aircraft and 100 images of the ground. 20 images are retained for training: 10 aircraft and 10 ground images. Images were obtained from Google earth by taking screenshots of airports and proceeding to crop areas that contain aircraft and ground. Cropped images are then saved as individual images. These images of aircraft have little empty space around them and minimal surrounding structures so that the SVM can focus primarily on the aircraft. Examples of training images are shown in Figure 2.

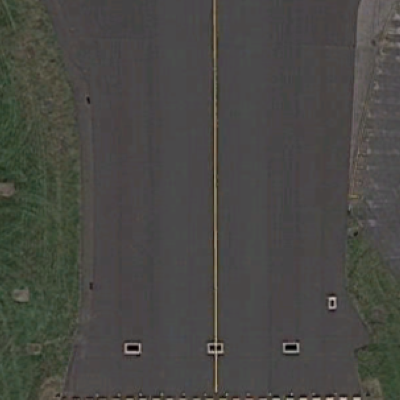


Figure 2: Image of ground (left) image of aircraft (right) used in the standalone dataset

All images of aircraft were rotated so that the aircraft faced north. They were also of the same dimensions (400x400 px) ensuring training was fair and balanced and are of the PNG format for simplicity.

On Google earth, the majority of aircraft are parked at the terminal. However, I did not want the SVM to think that aircraft must appear at the terminal to be classified as aircraft. To stop this from occurring, the location of aircraft in the airport surroundings are as balanced as possible. The data set shows aircraft in places such as taxiways and runways. The ground images are from a variety of airport locations. This was to vary the data set as much as possible while keeping it relatively simple.

### 2.3.2 Realistic data set

After the development of the large image search classification method, it quickly became apparent that aircraft in real life can exist in many different sizes and rotations relative to image search criteria. At present, the user defines the search area size and movement settings meaning the SVM may not always have a perfect view of an aircraft. The aircraft may appear off centre meaning the SVM will struggle to classify it. To overcome this problem, I needed to generate a realistic training set, so the SVM has a more realistic understanding of aircraft in a real-life environment. To do this, I gathered images of some aircraft at random rotations and of varying sizes, therefore, filling different amounts of the image. This gave the SVM a more realistic understanding of how an aircraft appears in real life.

To again simplify the realistic data set, I ensured each photo contained no more than one aircraft. If an image contained more than one aircraft, it would interpret an aircraft as an object of two or composed of two aircraft combined. Figure 3 shows an example of this occurrence.



Figure 3: Image not included in the final data set showing two aircraft

Images of aircraft with complicated surroundings such as complex jetways, terminals and vehicles in many cases were also disregarded. This is because it shifts the centre of attention of the SVM. The SVM will start to think an aircraft is an aircraft in addition to the surrounding vehicles, buildings and jetways in proximity. This severely limited the images that can be used to train the SVM. If Figure 3 was included in the final data set, the SVM would be led to believe an aircraft is half an aircraft and another wing from a nearby aircraft. This would hinder classification and skew results.

### 2.3.3 Parsing

An image in computer science is regarded as an array of values. Each field contains a value which is displayed onscreen as a channels colour intensity. To train a machine learning model, the training set is provided as an array of image data. The image data created by reading images from the file system using the OpenCV library. The OpenCV function that reads images returns a NumPy array of raw image data. The image data is then added to a large array containing other training data. The process is repeated for every image in the training set until the training data is contained in a single array. Concurrent to this process, a separate label array is filled with 1’s and 0’s. Aircraft are defined by 1’s, whereas ground is defined by 0’s. This array tells the SVM which image belongs to which classifier.

To add a component of further realism, the element of bias that occurs by the default training set is removed. This occurs prior to training using the method that generates the training set for the large image search. Images are rotated randomly by intervals of 90 degrees and then pre-processed (See chapter 2.4). They are then added to the large training array.

## 2.4 Pre-Processing

This chapter describes the methods used to process images before data is used for training and classification. Additionally, it defines how specific algorithms function and how the results yielded are of help to the overall system.

Pre-processing takes an image and extracts key information about it. There are several pre-processing methods that are available and commonly used in machine learning. Pre-processing is used to reduce the load on the machine learning model by supplying it with relevant information. When images are not pre-processed, the machine learning model is often slow and inaccurate. This is because of the excess data provided to it. Image pre-processing creates a feature vector that describes an image. The feature vector is then provided to the machine learning model for training and evaluation.[12][13]

### 2.4.1 Histogram of oriented gradients

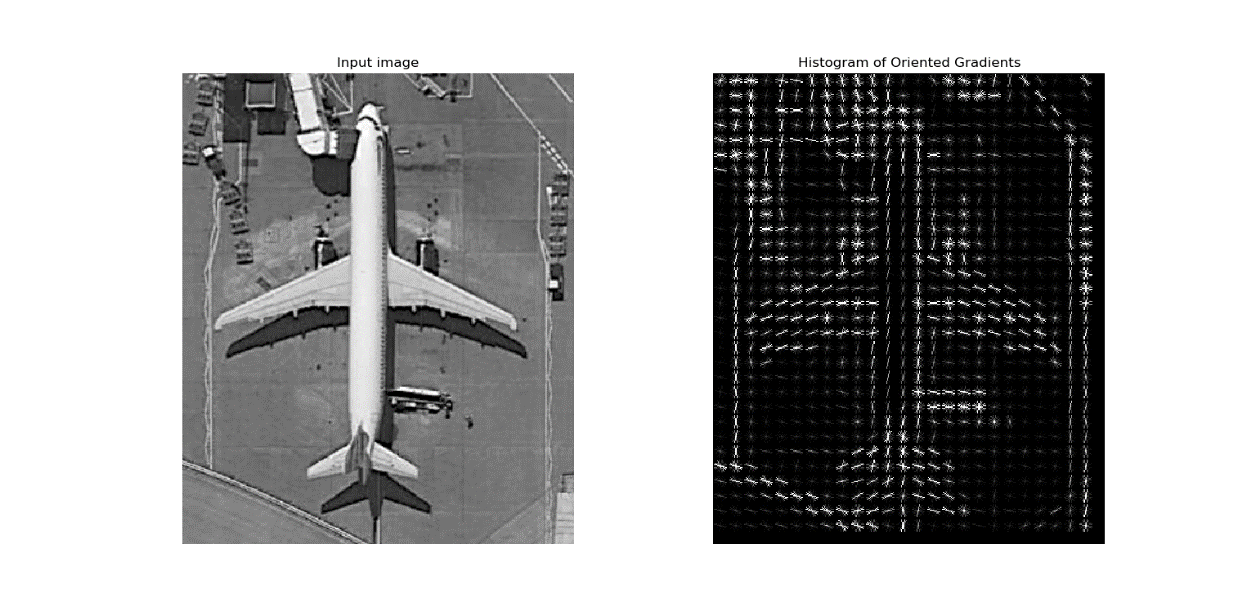
Histogram of oriented gradients (HOG) is the current pre-processing method used in the parsing stage of the aircraft detection program. HOG counts the occurrences in gradient orientation in localized cells. The cells are normalised in blocks using contrast normalisation. The size of a cell or block can vary. Blocks can consist of any amount of cells. In this case, the blocks cells are normalised to ensure the results have a low variance.[14][15]

Contrast normalisation is a simple image enhancement technique that changes the range of image intensity values. Minimum and maximum values are supplied to a normalisation function which is applied to the image. This process is carried out to remove bias caused by illumination and shadowing.

After contrast normalisation is completed, the next step is to create a histogram for every cell present. Once the histogram is created, the orientation can be calculated. The direction of the gradient is added to a feature vector which is returned and used to train the SVM.

HOG was implemented in this program as it is widely used with SVM’s for object detection. Figure 4 shows a visualisation of the HOG algorithm on an aircraft from the training set.

Figure : Histogram of Oriented Gradients algorithm applied to an image containing an aircraft.



The sizes of the cells are used in the HOG algorithm in the aircraft recognition program are 16x16px. The blocks consist of only one cell as this is the default value. If the SVM’s predictions were not accurate enough, the block size could be increased to take make full use of contrast normalisation.

### 2.4.2 Feature Vector

A feature vector is a vector containing information describing objects important details and characteristics. It can describe the features and characteristics of a small area of an image or can describe an entire image. A feature can take many forms such as average density, centroid (centre of the image where all medians intercept), pixel colour count, gradient magnitude and orientation.

The feature vector produced by HOG in the aircraft recognition system contains a number of blocks per row, number of blocks per column, number of cells per row, number of cells per column and number of orientations and, the orientation of every cell processed. Take an image with dimensions of 400x400px:

|  |  |  |
| --- | --- | --- |
| **Image size: 400x 400px (RGB)** | **Calculation** | **Length** |
| **Image length flattened** | (400x400)x3 | 480000 |
| **Feature Vector** | (400x400)/16 | 10000 |
| **Percentage difference** | (480000-10000)/480000\*100 | -97% (470000) |

By applying the HOG algorithm before training, we can not only reduce the amount of information passed to the SVM by roughly 97% but also increase the accuracy of the data passed to it. This plays a large part in computational complexity as the training and classification of a single image is 47 times faster. Figure 4 shows the difference between an aircraft and ground feature vector.

A screenshot of a computer

Description automatically generated

Figure 5: Visual representation of a ground feature vector (Red) and an aircraft feature vector (Blue)

Figure 5 shows how the feature vectors obtained from HOG pre-processing vary dependant on the input image. The red points show that a ground image has quite a flat and consistent distribution of features as there are not many changes to gradient orientation. The blue points highlight the number of changes to gradient orientation there are in an image of an aircraft. From this information, it can be predicted that high amounts of change to gradient orientation in an image points to the image being classified as an aircraft.

## 2.5 Machine learning

This chapter explains the machine learning techniques used during this project to classify images of aircraft and ground.

Machine learning is an application of Artificial intelligence that allows the computer to automatically learn and improve using statistical models, algorithms and access to data without explicit programming. Machine learning has multiple processes with the first being training. Training data is provided to the ML model so it can observe the data and look for patterns. The more data fed to a model, the greater the understanding thus increase in accuracy. However, to get accurate and reliable results, the ML model must be passed reliable data; otherwise, its ability to predict correctly will be greatly flawed.

After the training stage is completed, the ML model can be passed test data. The model can evaluate the test data and return a prediction. In our case, the results determine whether the image contains an aircraft or is purely ground. After predictions are made, if correct, it can be added to the training set to improve accuracy.

Optimisation is the final stage and can often make or break an ML models accuracy. Decision boundaries and functions created by the model can be altered by the input variables such as C, gamma and kernel. These essentially define the way the model can make decisions. While ML can analyse vast quantities of data and quickly yet accurately make predictions, it requires a large amount of time and resources to train it correctly.

### 2.5.1 Supervised learning

Supervised learning is the process of providing a machine learning model with labelled training data. The ML model produces an inferred function to make predictions about the test data.[16]

A set of N training examples in the form **{(x1,** y1**),…, (xN,** yN**)}**, **xi**representing the feature vector provided by HOG pre-processing, yi being the label to accompany the data that informs the ML model of its class. The ML model will produce a function in the form **g : X -> Y** where **X** is the input space and **Y** is the output space. The function **g** is referred to as the hypothesis space. Many ML algorithms are probabilistic meaning **g** takes the form of a conditional probability model where **g(x) = P(**y**|x)**. The model returns its prediction based on the highest probability.

To contrast, unsupervised learning allows a computer to draw its inferences from training data. This allows the ML model to group classes as it sees fit. This can yield unexpected results; however, after extensive training, it becomes very accurate.

### 2.5.2 Support vector machines

A Support vector machine is a supervised learning algorithm that analyses data. The common uses of SVM’s are classification and regression analysis. SVM’s are commonly used in real-world problems such as protein classification, image classification, text classification and classification of handwritten characters.

An SVM is provided training data in the form **{(x1,** y1**),…, (xN,** yN**)}**, **xi**representing the feature vector provided by HOG pre-processing, yi being the data’s label. In our case, **xi** represents images of ground and yiis either a 1 or 0 dependant on the human classification of the image. The SVM plots a graph and plots feature vectors as individual points after kernel processing. After all feature vectors in the training set are plotted, a hyperplane is plotted between classes. This graph can exist in 3-Dimensions the hyperplane can exist as any shape. This can be seen as something similar to a line of best fit. An SVM can have as many hyperplanes as it wishes to group classes. The more hyperplanes present, the more complex classification becomes decision boundaries are challenged. Figure 5 shows an example of an SVM after training is completed.



Figure 6: An example of a support vector machine. The blue dots represent one class, and the red dots represent a second class. The yellow dotted line shows the decision boundary. The solid black lines show the effect of the C parameter[17]

To make predictions, the SVM is provided with an unseen test set of feature vectors. This could be from a large image or smaller image. The SVM plots the feature vector and dependant on its position relative to the decision boundary returns a probability and prediction.

#### 2.5.2.1 Kernel

The kernel is a set of mathematical functions that control the way an SVM classifies data. There are several commonly used kernels such as linear, non-linear, polynomial and Radial Basis Function. The kernel often defines the shape of the hyperplane

This is the mathematical equation of the RBF kernel. RBF is commonly used as the default algorithm for ML models as it needs no prior knowledge about the dataset. The SVM used in the project uses the RBF kernel. Figure 6 shows the RBF kernel equation.

Laplace RBF kernel equation

Figure 7: RBF kernel equation

Feature vector **xi** and label yiare passed to the RBF kernel function. After the computation is completed, the kernel returns coordinates to be plotted by the SVM. This equation is computed for every feature descriptor provided to the SVM during training and testing stage.

#### 2.5.2.2 Hyper parameters and decision boundaries

The SVM can take parameters that help define decision boundaries. For many applications, the default values provided by the algorithm can be used. However, dependant on the type of data and shape, these may need to be tuned.

**C** is a penalty parameter of the error term. This controls the distance of the margins from the hyperplane that ignores features that may have some form of error during training. Margins are shown either side of the yellow dotted line in Figure 5

**gamma** is the coefficient used in the RBF kernel along with many others. A high gamma value leads to high bias and low variance whereas a low gamma variance leads to a small bias and high variance.

A hyperplanes margin is regarded as soft when it ignores features close to the hyperplane. This often leads to a better fit and overall increase in accuracy in many cases. A margin is hard when it ignores no features and tries to divide classes perfectly down the middle.

#### 2.5.2.3 Tuning

To optimally tune hyperparameters, a grid search algorithm is used. Grid search takes a set of seemingly random values and passes them to the SVM. In our case, C and gamma values are generated by decrementing from 10 and 1 respectively. The SVM is trained using its own data set and then tested. The grid search algorithm cross-checks the prediction from the SVM with the human classifications obtained and data collection. Based on the highest scoring combination of C and gamma with the given test set, the algorithm chooses the optimal input parameters. We now know the values of C and gamma are optimised specifically for the purpose of classifying aircraft and ground images.

#### 2.5.2.4 Under and Overfitting

Under and overfitting occurs when the hyperplane divides classes too closely or too loosely. Figure 7 shows the different types of fits.



Figure 8: Shows the different types of fits an SVM’s hyperplane can follow [18]

When an ML model is overfitted, it makes it difficult for future predictions to be accurate as they must fit very specific criteria therefore often fail to reach the correct result. Overfitting occurs when a hyperplanes margin is hard.

Underfitting occurs when the ML model fails to gain an understanding of the underlying structure. This can be a result of selecting the wrong kernel with the shape of the data provided at the training.

### 2.5.3 Cross-validation

k-Fold cross-validation is a resampling technique used to evaluate ML models accuracy and performance on a limited data set. It is often used to gain an insight into expected results with an unseen data set. The technique has a single parameter referred to as ‘k’. Parameter k defines the number of groups the total data set will be split into. If k = 10, the process will be called 10-fold cross-validation. Cross-validation is a popular method because it is simple to understand and usually results in a less biased and optimal estimate of the model as opposed to a simple train/test split. It also shows how accurate the training data is. If the cross-validations core of the training data is low, then the model may struggle to evaluate unseen test data.

The steps for cross-validation are as follows:

1. Randomize the order of the dataset
2. Split the dataset into k groups
3. For each group create

3a. Retain group for test

3b. Train model using remaining groups and evaluate the accuracy with the test set

3c. Retain the evaluation score and discard the model

1. Summarize the skill of model using k evaluation scores

In most cases, the scores are averaged to create the most realistic score for the training set. To evaluate the performance of the aircraft recognition program, I cross-validated the standalone and realistic data set across various image sizes.

#### 2.5.3.1 Standalone dataset

|  |  |  |  |
| --- | --- | --- | --- |
| **10-fold CV of standalone dataset** | **Lowest Fold** | **Highest Fold** | **Average** |
| **400 x 400 px** | 0.94 | 1 | 0.99 |

Results of 10-fold cross-validation of standalone dataset show that the accuracy of the ML model and data set is incredibly high at 99%. The variance between the lowest and highest fold is 6% meaning the model is incredibly accurate. 8 folds achieved 100% during this test.

#### 2.5.3.1 Realistic dataset

|  |  |  |  |
| --- | --- | --- | --- |
| **10-fold CV of realistic dataset** | **Lowest Fold** | **Highest Fold** | **Average** |
| **800 x 800 px** | 0.37 | 0.99 | 0.74 |
| **400 x 400 px** | 0.66 | 0.96 | 0.75 |
| **200 x 200 px** | 0.66 | 0.67 | 0.67 |
| **100 x 100 px** | 0.60 | 0.61 | 0.60 |

Results of 10-fold cross-validation of realistic dataset show that when image size is increased, the average cross-validation score increases; however, the variance between highest and lowest fold also increases. As variance was originally higher, images had to be deleted that didn’t conform to dataset rules.

# 3. Technical Achievement

This chapter explains the functionality of the finished product developed throughout the project.

## 3.1 Graphical user interface

The graphical user interface is the starting point of the program. The user is able to select the classification method and options for the image search classification process. Figure 9 shows the windowed GUI that the user is shown the following execution.

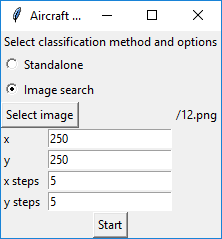


Figure 9: GUI is shown to user after program execution allowing the user to select classification options

The user must first select the classification method. Standalone classification requires no other input from the user. The user can now press Start to begin the training and classification process.

If the user wishes to search for aircraft in a larger image, the image to search must be selected. Clicking the select image button allows the user to navigate to the location of the image they wish to classify and select it. Figure 10 shows the window used to select images from the files system



Figure 10: File selection dialog used to select the image to be searched using large image search

After image to search is selected, the user must specify;

* x- The x dimension of the search area in px
* y- The y dimension of the search area in px
* x steps- The number of moves along the x-axis
* y steps- The number of moves along the y-axis

From this data, the program selects suitable x and y step values to ensure image boundaries are met.

After the user has selected search criteria, the Start button can be pressed to start the image search process.

## 3.2 Standalone classification

Standalone classification is the recognition of aircraft images where images from a data set are selected at random for training and classification. There are 200 images in the standalone dataset. 100 images of ground and 100 images of aircraft. 5 images of aircraft and 5 images of ground are selected for testing. The remaining 180 images are used to train the ML model.

The images are put through the usual process of parsing, HOG pre-processing before being passed to the SVM. After training is complete, test images are passed to the SVM. The SVM returns a set of predictions for the images tested. To show the user the results, the test images are displayed in a grid with accompanying predictions and probabilities. The probability shows how sure the ML model is that the image belongs to a class.

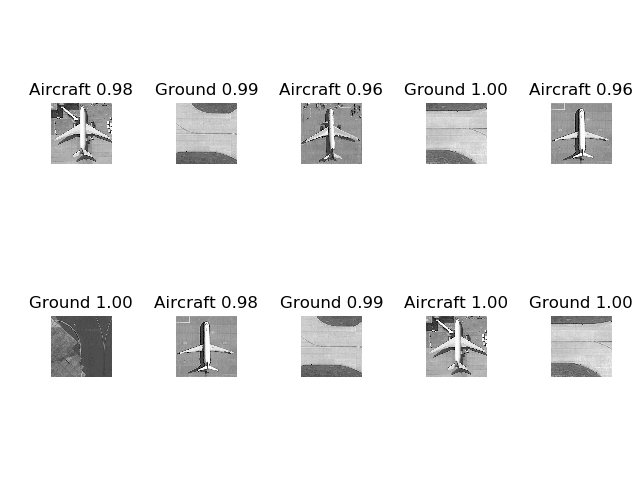


Figure 11: Results from standalone classification showing 10 correctly classified images

The results in Figure 11 show that the program correctly identified all ground and aircraft. The accuracy of standalone classification in all test cases is 100%.

## 3.3 Large image search

Large image search requires a more complex data set. The data set has more samples, and the data in the image is more representative of aircraft in a real-world setting. Aircraft in a real-life setting appear at random locations with different backgrounds, are of different sizes and appear in a multitude of orientations. Training the SVM for large image search Is slightly more complicated than training for standalone classification. First, all training images are resized and to ensure no element of bias is incurred. The training images already show aircraft at a variety of random orientations. On top of that, images are rotated in random increments of 90 degrees .

The basic principle behind large image search is that the user defines search criteria to the program. The program iterates the image searching in small portions for aircraft. The execution behind it is slightly more complicated.

The image is first divided into several smaller images. The image sizes are defined by the user's input. If the user has given search settings that mean the images overlap, parts of the large image will be processed more than once. If the user has supplied search criteria that mean the search boxes do not overlap, some areas will be skipped. This meant the image would not be classified correctly. Originally the user defined the number of pixels in each step; however, this meant that boundary data was often ignored. This meant the bottom or right row would not be included in the classification. To overcome his problem, the user defines how many steps they wish to carry out. The higher the step count, the more the overlap of the search area.

The x and y steps are calculated as such

x\_step = image\_width – x\_search\_size / x\_steps

y\_step = image\_height – y\_search\_size / y\_steps

This ensures that no area of the image is missed.

The smaller images are parsed, HOG pre-processed and passed to the SVM for evaluation. The SVM returns a prediction array with associated probabilities. To display results to the user, there are two forms of output. Heat map and the original image with boxes overlaid to show the location of aircraft.

Heat maps highlight areas of the image that show a high probability of aircraft. They are generated by first creating a black image that is identical in size to the original search image. The probability of an area is plotted on the image by calculating the colour intensity values based on the probability. Once all areas are plotted, adjacent search areas are averaged to blend colour together.

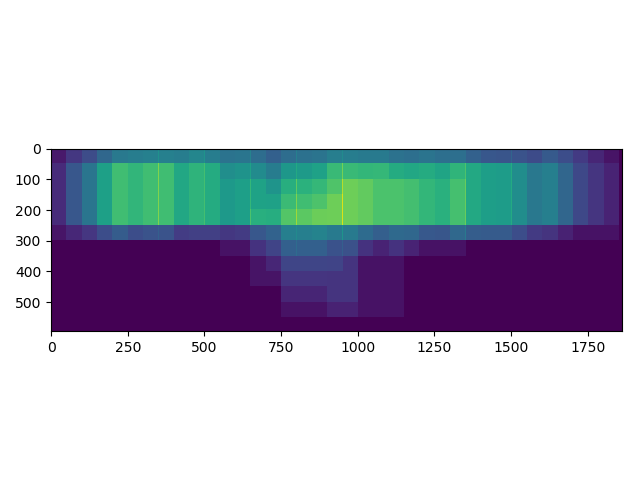


Figure 12: Heat map of aircraft probabilities showing areas of interest to the SVM based on classification probabilities

Figure 12: Heat map of aircraft shows the results of an image search as a heatmap. The bright areas indicate the SVM has predicted there is a high probability of aircraft at that specific location.

To show the user the location of discovered aircraft, the original image is taken and copied. The predictions obtained from the SVM are used to determine the location of aircraft. Boxes are placed on the image to show the locations clearly to the user.



Figure 13: Image search results. Green boxes highlight area that the SVM believes to contain an aircraft

Figure 13: Image search results shows where the SVM has found aircraft. The boxes overlap as the program was provided with overlapping search criteria. Accuracy of large image search around 60%. This is because it is difficult to provide search criteria that align the search area perfectly with aircraft. This means several aircraft are overlooked. It is difficult to reduce the number of boxes in a tight location because it is difficult to tell which areas of interest are relating to the same aircraft.

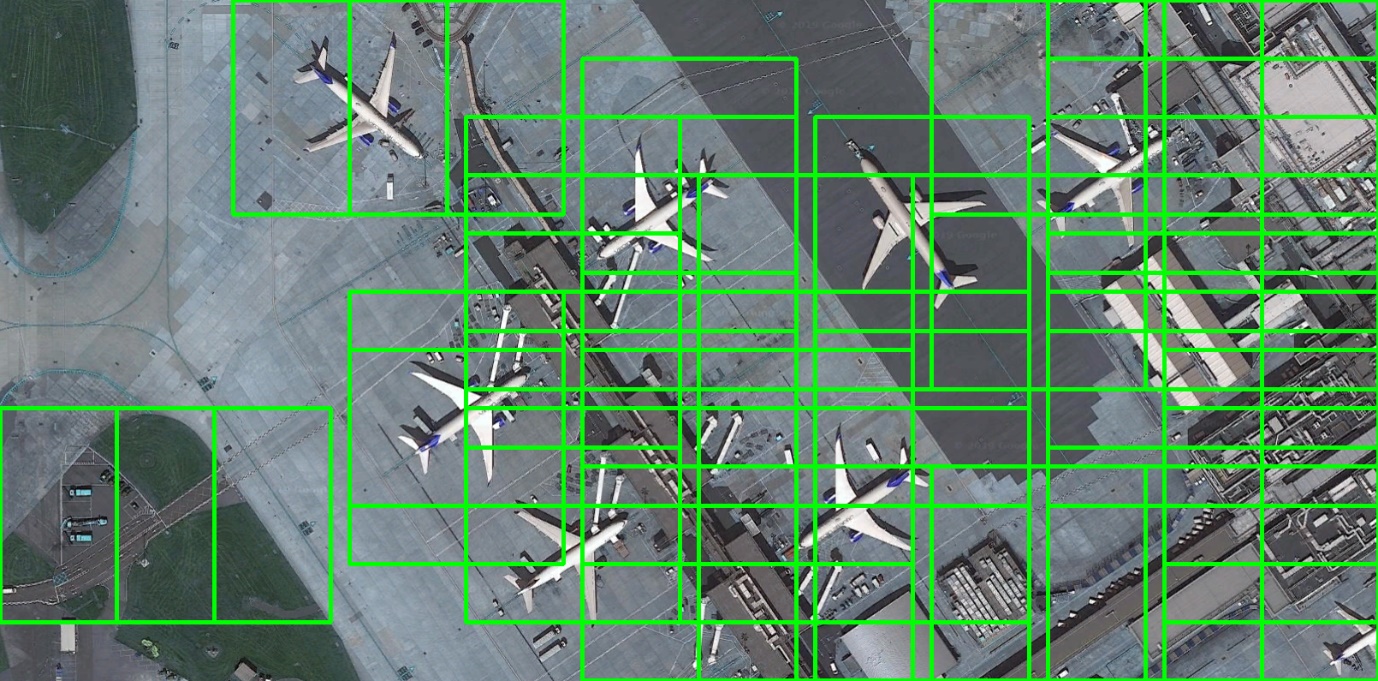


Figure 14: False positives from large image search

Figure 14 shows the results of another large image search. The system has identified all aircraft at varying rotations and sizes in the image. However, it has also classified a large amount of ground as aircraft. The areas classified as aircraft which are actually ground contain complex structures that confuse the SVM. The terminal has many different changes in colour gradient which is similar to images of aircraft which have major colour gradient changes.

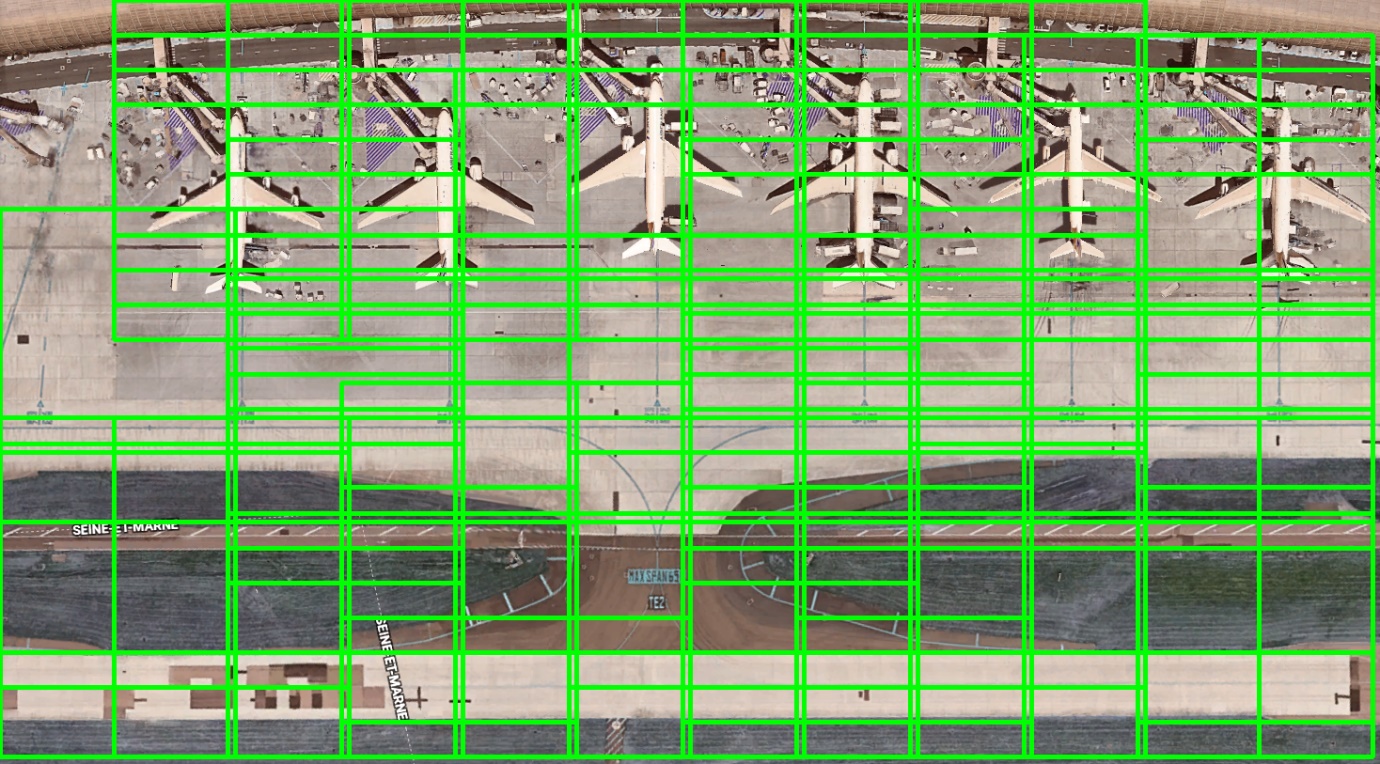


Figure 15: Large image search with an extremely high amount of false positives

Figure 15 shows a large overlap of search area criteria provided by the user. This leads to an excessive amount of rectangles in a small space.

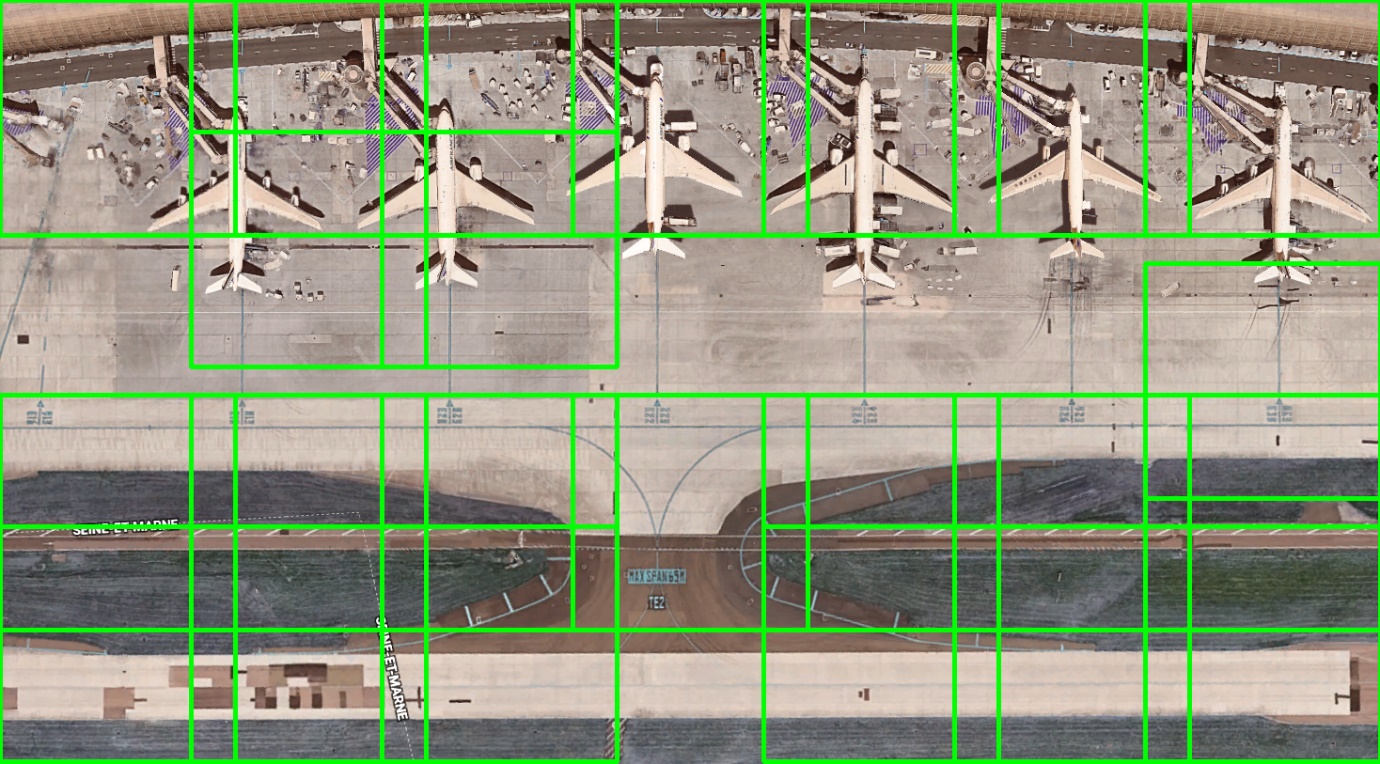


Figure 16: Large image search results on the same image as used in Figure 15 but with different search criteria

Figure 16 shows how the search criteria can alter results so profoundly. Decreasing the number of steps between search areas reduces the number of rectangles laid over the image. There are still a considerable amount of false positives as the area towards the bottom of the image has a high amount of colour gradient changes. The ideal image to present to the large image search would be an image with a plain background

## 3.4 Saving of results

After the classification of a large image is completed, the results are automatically saved to the user's file system. The heat map and search results are assigned a UUID and saved to a folder. The newly generated images will have the same UUID for simplicity.

The results of classification appear in their own GUI window. The user is given options to move plots, zoom in and out, and further save the view. They can select choose a name for the results and define the format before saving. The library used for this process is Matplotlib. The library used to automatically save the results is PIL.

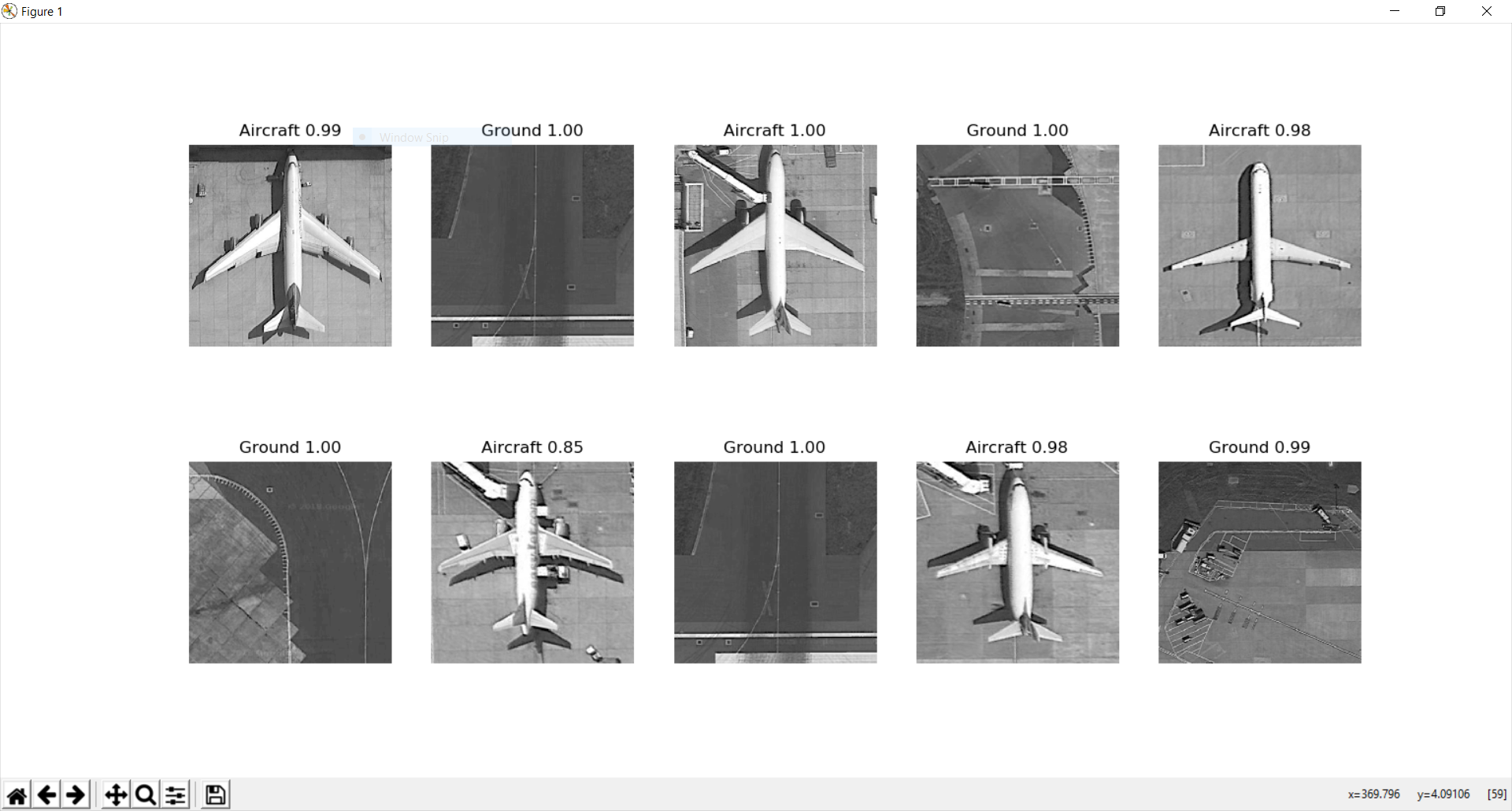


Figure :Shows the GUI that appears post classification of large image search. Options to manipulate and save can be selected at the bottom of the window

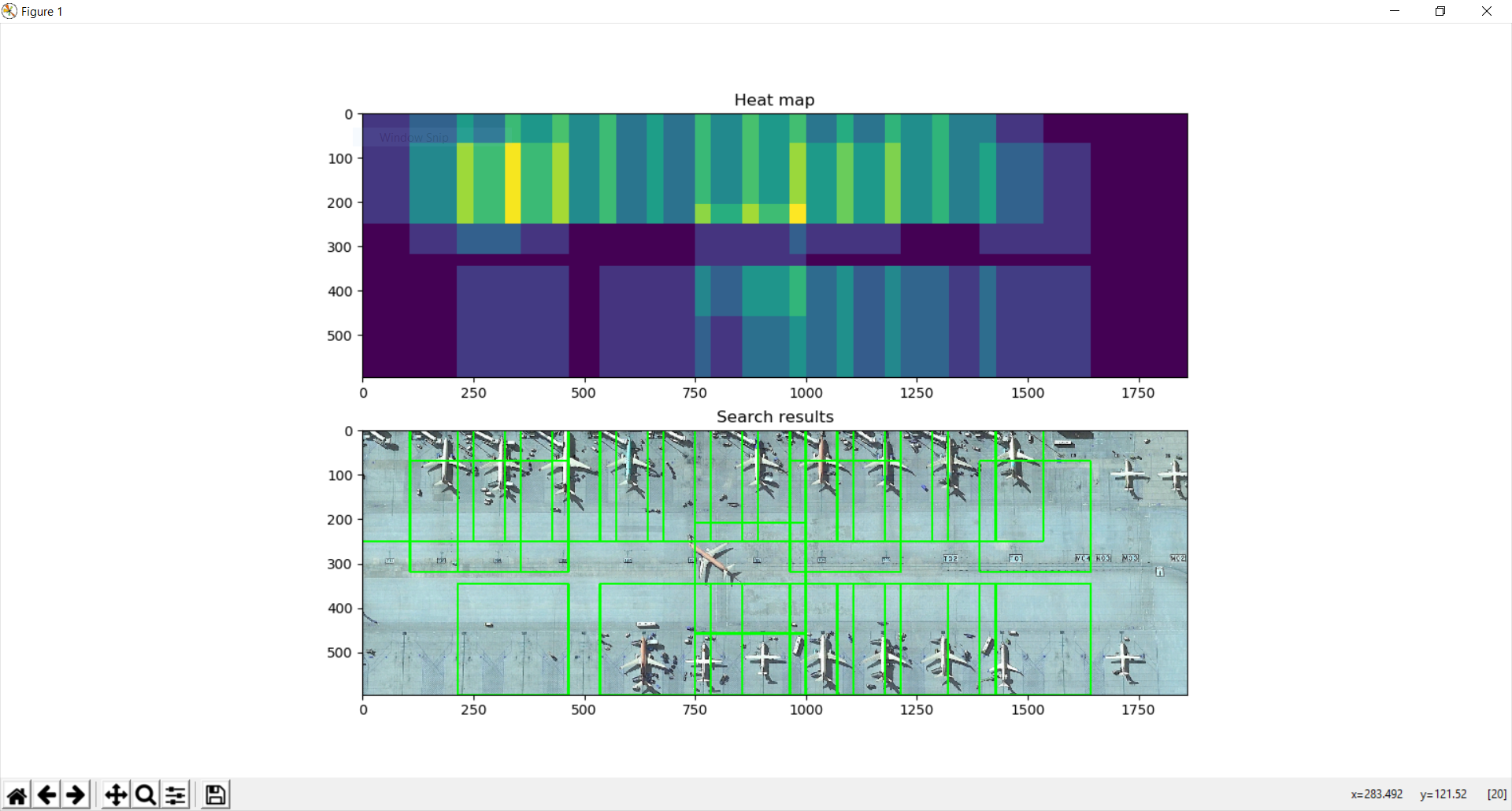


Figure 18: Shows the GUI that appears post classification of large image search. Options to manipulate and save can be selected at the bottom of the window

## 3.5 Software design

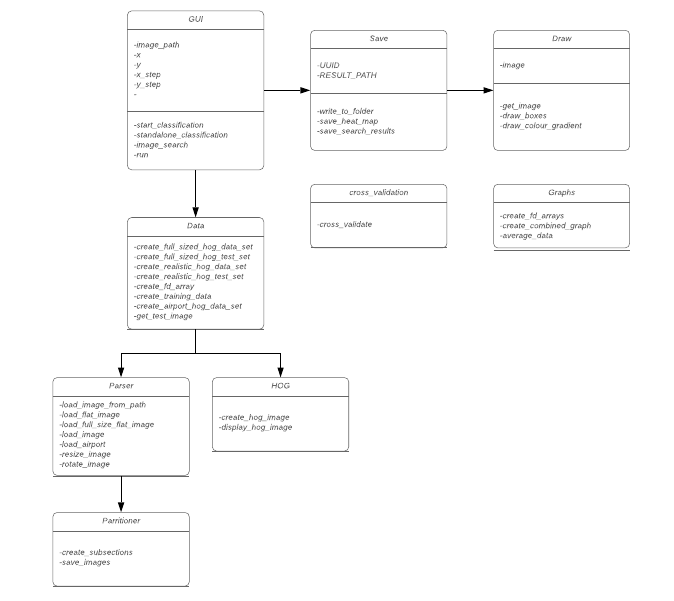
This sub-chapter explains in technical terms of the software structure. Below shows a class diagram of the aircraft identification system. Arrows show the class has an instance of the class pointed to upon execution

Figure : A class diagram showing the structure of the software solution. Classes not joined to an entity are not used upon normal execution of the system

The classes shown in Figure 19 are dependent on each other to meet all project aims and objectives. Upon execution of the program, all necessary imports from internal and external libraries and all imports of other classes are also completed and instantiated.

Upon execution of the program, the GUI’s constructor creates a Tkinter instance and adds labels, text input and buttons. These are used by the user to select classification options.

Upon the execution of classification, a data set is generated automatically by the data class. The data class formats data for the SVM to understand through the help of HOG, Parser and Partitioner classes. The Parser retrieves images from the user's file system. For standalone classification, the images are stored in a folder named “Images”. Each image is passed to the HOG class to create a HOG feature descriptor. The feature descriptor is added to a larger array used for training. This process is repeated for every image in the data set. 10 images are excluded from the training set and added to the test array however

After classification is complete, the user is shown the results from image classification. For large image search, training images are stored respectively in folders called “Ground” and “Aircraft”. All images are parsed using the same methods as the standalone data set with the addition of a random image rotation function that randomly rotates the training images. To create the test set for large image search, a large image is loaded from the user's file system. The images absolute path is passed from the GUI class to the Data class where the image proceeds to be loaded. The Partitioner class then divides the image into several smaller segments defined by the user's search options. Each smaller image is passed to the HOG class where it is pre-processed and added to the testing array.

After the classification of the large image partitions, the image searched, and prediction array returned by the SVM are passed to the Draw class. The Draw class creates a heat map using the draw\_colour\_gradient() function. The heat map created will be the same size as the image used for the search. The values generated to show areas of high probability are calculated as follows

heat\_map\_intensity = prediction\_probability \* 255

The heatmap averages the gradient transitions to ensure the heatmap shows a smooth colour gradient.

The Draw class also overlays rectangles based upon the predictions made by the SVM using the draw\_boxes() function. It takes the original image, makes a copy and returns the image with rectangles highlighting areas containing aircraft. The heatmap and results overlay are then displayed to the user and automatically saved to the users file system with a newly generated UUID.

The cross\_validation class is used to cross validate the data set to gain an understanding of its accuracy. This class is executed separately during the development stage of the data set. The graphs class creates a graph that shows a visual example of feature vectors obtained from the training set.

## 3.6 Project Planning

During this project, I used CSEE Jira to manage issues arisen and tasks to be completed. The platform allows the project to be divided into tasks, sub-tasks, stories and epics. Main sections of the project are divided into stories and epics. For example, I created an epic called “GUI”. All tasks that relate to GUI programming are linked to the GUI epic. The same process was repeated for tasks that fell under technical documentation, final report, image segmentation, image capture and SVM. During every meeting with my supervisor, we agreed on work to be completed. Feedback of the meeting was left on the platform so that I could break it into smaller tasks and move to solve these issues.

## 3.7 Methodology

During this project, I used the Kanban methodology to organise tasks, issues and stories. The Kanban methodology aims to limit work in progress and stop the number of tasks becoming half completed. By limiting the work in progress, the number of tasks in progress by an individual is decreased allowing the development to be of a higher quality. Kanban uses a set of key values that define the way individuals should conduct themselves during a project

* Transparency- The sharing of information between individuals increases the rate of task completion.
* Balance- Different aspects and viewpoints must be balanced to increase workability by all individuals
* Collaboration- To improve the way individuals work together.
* Customer focus- An emphasis on customer focus through regular updates.
* Flow- Ensuring work is continuous
* Leadership- Inspiring others via inspiration, example and reflection.
* Understanding- The importance of individual and organisational understanding.
* Agreement- Moving towards goals agreed upon collectively.
* Respect- Value, understanding and consideration of others.

The key values enable the Kanban methodology to be closely followed by all individuals.[19]

## 3.8 Momentum

To maintain momentum during the project, I stuck to a strict schedule to ensure the weekly 15 hours of work was met. This ensured the project received regular updates and changes to meet the final finished product. While I was working on tasks completed, I moved them to the in-progress column of the table. I would log work on the task and talk about what was done in the allotted time. After a task was completed, I would test the task in progress works and interacts correctly with the system. If this was the case, I would move the task to the done column of CSEE Jira.

On occasion, bugs would plague the progress of small tasks such as formatting the training set data structure so that it would be accepted by the SVM. This took a large amount of time out the project as this specific task was blocking the progress of the rest of the project.

Approaching the Christmas break, the momentum of the project slowed a little as deadlines for other modules approached however picked up nicely towards the end of the Christmas vacation. Since then, momentum has been in line with the requirements of the module.

## 3.9 Adapting to change and dealing with risks

Adapting to change and risk during a project is always difficult. This is because it is hard to foresee problems that may arise. For example, a project may be based around a certain approach. If the approach fails, the project could collapse leading to a waste of resources. Project collapse happens very often and is often why new projects can struggle to gain momentum during the early stages.

At several points during the project, I had to change the approach to a problem I was having. This is difficult to implement but often required in large projects. Time is often lost as the previous approach may be completely shelved and not used again hence a waste of man-hours. During the early stages of the project, the original goal was to use a Convolutional neural network to classify aircraft and ground images. However, I quickly discovered that the data set required for accurate training of a CNN would be much larger than the data required for accurate training of an SVM. The switch was a difficult decision to make but the consensus was the time required to create the data set for a CNN would hinder the progress of the project. The risk associated with the change was high as I would need to carry out new research on the project.

As this project used external libraries, there is always a danger that a new version of the library could become available. When a new version of a library becomes available, there is always a risk that the updated version may be incompatible with the current software through the deprecation and addition of methods and classes. This risk can be minimalised through monitoring of releases and specifying which version of the given library to be used in the requirements.bat file. This ensures the software can continue to function as anticipated.

## 3.10 Performance

Overall, I feel I performed well during this project. I managed my time effectively, balanced my learning and deadlines from other modules and kept the project alive. The only part that has let me down so far is the quality of research. The first few weeks of development were wasted as the approach I had research was deemed inappropriate to the given solution and timeframe. I started the project using the wrong ML model. This was a quick and easy fix, however, could have been easily avoided by an in-depth research of CNN’s. The extra research required by the change was additional work that I didn’t want to worry about and could have been undertaken at the research stage.

My personal achievement during this project was the development of a highly accurate standalone classification process and the development of large image search. Although the large image search shows lower accuracy scores of around 50-60%, to develop a program that detects aircraft in the given surroundings is a difficult task to undertake.

## 3.11 What have I learnt?

During this project, I have learnt the importance of project management and gained an understanding of why large projects need to be managed effectively. The use of CSEE Jira really enabled me to track issues and look at my overall progress. It allowed me to identify issues within the way I work through the use of its analytic tools. I discovered I was spending too much time on solving small issues that really had no real impact on the overall project or that weren’t part of the project objectives. For example, the development of a GUI that looks nice was a smaller priority than the overall objective of identifying aircraft. However, I spent more time than I should have to attempt to perfect it when there were underlying issues with the classification. This issue links closely to time management as I now understand how important it is to prioritise tasks in order of importance to the MVP and overall project objectives.

I have discovered how important the quality of research for a new project can be. If the research is of a poor standard, the project may fail as it may begin using the wrong methods or working towards unachievable objectives. Poor research can lead to project collapse as the product may not even be required in the current market. The importance of communication with my supervisor(s) is a skill that I have improved greatly. Getting my objectives, progress and issues encountered during the project across were difficult for me too. This highlights the importance of documenting technical achievement, issues and solutions.

# 4. Legal, ethics and sustainability

## 4.1 Sustainability

Sustainability of software defines how software may be available in the future and the impact it will have and how it can be updated and adapted to meet newer needs. The software I have produced can be adapted to classify different data and theoretically an unlimited number of classes. This means that differentiating the data set allows the software to classify completely different material. This allows the needs of the software to adapt to change. The software can also create new decision boundaries to classify new data. The program is runnable on the majority of operating systems.

## 4.2 Legal

As computer vision and machine learning are such new topics, the legal issues surrounding the two subjects are unclear at best. This raises issues itself as the government is struggling to define laws at the speed that technology is being developed at. This means at times certain technology is ungoverned causing an array of issues. The most recent technological development that has been widely discussed is the introduction of driverless cars to society. The government scrambled to develop laws that govern the use of this technology and is still a topic of discussion to this date. People have asked who is responsible for the result of an ML error in a driverless car; the driver or the software?

There are specific laws that disallow the use of image capture equipment from being used in specific environments. However, there are no clear rules about the processing of that data if no one’s property is featured in the data.

## 4.3 Ethical

Many people are scared by the fact that technology is becoming ever more intelligent. Machine learning is a relatively new concept and very little insight is provided to the general population. This stunts the growth of ML development. There are no new ethical issues surrounding my project as the data I have used for the classification process is already available online. This means the ethical issues surrounding my project are inherited from Google (the data source) as the storage of satellite imagery is believed to be a breach of personal privacy. However, the imagery I have used are of public airports meaning no private property, i.e. Houses or private grounds/ airstrips are included in the data set. This means no privacy rules are being breached.

## 4.4 Intellectual property

Intellectual property is the system that aims to allow peoples creations to earn them recognition and financial benefit. It is supported by law through copyright, patents and trademarks. This allows a person’s creation to be protected from third parties. As I used third-party libraries to develop my project, it raises the issue of who owns the entirety of the program. This is where licensing comes into play. Each library I have incorporated in the final project has an associated license.

The Berkley Software Distribution (BSD) Licence is a permissive license that imposes minimal restrictions on the use and distribution of covered software. Scikit-learn, Scikit-image, Tkinter, OpenCV and Matplotlib are covered by this license.

PIL follows a license similar to an MIT license. An MIT license is also a permissive license that enforces a very limited restriction on reuse, therefore, has excellent licence compatibility. PIL gives permission to use, copy, modify, and distribute.

Both licences that belong to the software used to allow the near unrestricted use and distribution of their software providing the license agreement is copied and used in the release of the software. This means the software I have created is entirely owned by me in line with external library terms and conditions.

# 5. Conclusion

Object detection is a highly complex problem that has taken many years to develop to the stage it has reached presently. This project highlighted the computational complexity of processing large amounts of data and how systems must be optimised for specific purposes to perform accurately and effectively.

The development of the aircraft identification system involved many steps. First the development of a training set. The training set defines what the ML model will be classifying. This step was time-consuming as the repeated steps of cropping images from Google Earth was tedious. What to define as aircraft and ground was also a difficult task to undertake as results can only define how accurate the dataset is. There is very little feedback to point me in the correct direction. Rotating images to face the same way allowed the ML model to accurately focus on key points of the image. Keeping the images, the same size turned out to be compulsory to the accurate development of the HOG feature descriptor.

The second step is to parse and pre-process the images. This is a fairly straightforward process to implement; however, the formatting of data proved to be more of an issue than anticipated. The pre-processing stage was completed by the implementation of HOG. HOG returns a feature descriptor that is later used to train the SVM. The ML model was trained using the data set and provided with new unseen data for testing.

The third step is to pass the training feature descriptors to the SVM. The SVM is trained with the training images in the form of feature descriptors in order to gain an understanding of aircraft and ground objects. After training is completed, the SVM is presented with an unseen data set. The SVM returns an array of predictions and probabilities. The test data is pre-processed the same way as the training set before its evaluation. To improve SVM’s accuracy, the training set is used to optimise the SVM’s hyperparameters. Hyper-parameters allow the SVM to tune the decision boundary to fit the data correctly. In early versions of the project, the SVM was often over or under fitted. This led to a series of inaccurate results. It is important to tune an SVM in early stages of development and at regular intervals along the development process to ensure predictions remain accurate if changes to the data set occur. If more than one data set exists, it is important to tune an SVM for a specific data set therefore use different hyperparameters.

The final step is to parse results by showing areas of interest to the user. This involves taking the prediction array and placing it over the original image. This can be shown in two forms as either a heat map showing areas with high probabilities or an overlay of squares highlighting positions to the user. This step was fairly straightforward. Attempts to reduce the number of squares placed over the original image by looking for peaks in probability were unsuccessful as this meant h

The results of this project on the standalone dataset show an accuracy of 100% during classification and an average score of 99% during cross-validation. The classification of standalone images has a higher accuracy as the images of aircraft are centred and have the same orientation. Images of aircraft in the standalone dataset are cropped tightly to the aircraft removing parts of the image that skew an SVM’s understanding of aircraft. This makes it easy for the SVM to focus on aircraft alone.

The cross-validation scores of the realistic data set vary dependant on the size of the image. When the images are 800x800px, the average cross-validation score is 74% however when image size is reduced to 100x100px, and the cross-validation score drops to 60%. During the classification of larger images, the accuracy is between 50-60% dependants on the image provided. The images obtained from the large image search often contain aircraft at varying rotations. The aircraft can also appear in a large amount of surrounding image meaning if an aircraft is small enough, the amount of ground in the image will occupy a higher proportion of the image than aircraft. This can often convince the SVM that the image is primarily ground. This was an interesting observation as the realistic data set contains images of aircraft of varying sizes and rotations therefore not affect the SVM’s output. In a number of cases, the large image search returns images where parts of but not the entire aircraft are in clear view. For example, some images bisect an aircraft down the centre of the fuselage. To a human, we can deduct from half of the aircraft that an aircraft exists and a specific location. However, as only images of aircraft that contain the entirety of an aircraft were included in the training set, the SVM will proceed to predict that the image is primarily composed of ground. This shows that machine learning algorithms are accurate with a specific data set; however, given test data that does not tie closely to the training set, accuracy is significantly impaired. It further shows that ML models do not have the ability to reason with their training data as a human would.

I selected this project to research the effectiveness of machine learning and computer vision. The results gained from this project show machine learning is an effective means of classification and is rapidly developing; however, there is still room for improvement. The classification of aircraft in larger images seems random at times; however, this could be down to the realistic data set. Low cross-validation scores of the data set show that the ML model struggles to classify the training set itself.

The project shows the possibilities of computer vision and machine learning. Not only can it be used in private projects and for specific purposes but to help the general public. Computer vision and machine learning can assist in areas such as healthcare, security and banking, agriculture, automotive and industrial.

# 6. References

[1] "OpenCV", *Opencv.org*, 2019. [Online]. Available: https://opencv.org/

[2] "scikit-learn: machine learning in Python — scikit-learn 0.16.1 documentation", *Scikit-learn.org*, 2019. [Online]. Available: https://scikit-learn.org.

[3] "scikit-image: Image processing in Python — scikit-image", *Scikit-image.org*, 2019. [Online]. Available: https://scikit-image.org/.

[4] "FPDF", *Fpdf.org*, 2019. [Online]. Available: http://www.fpdf.org/.

[5] "NumPy — NumPy", *Numpy.org*, 2019. [Online]. Available: https://www.numpy.org/.

[6] "Matplotlib: Python plotting — Matplotlib 3.0.3 documentation", *Matplotlib.org*, 2019. [Online]. Available: https://matplotlib.org.

[7] "24.1. Tkinter — Python interface to Tcl/Tk — Python 2.7.16 documentation", *Docs.python.org*, 2019. [Online]. Available: https://docs.python.org/2/library/tkinter.html.

[8] "Pillow — Pillow (PIL Fork) 6.0.0 documentation", *Pillow.readthedocs.io*, 2019. [Online]. Available: <https://pillow.readthedocs.io/en/stable/>.

[9] Ballard, D. and Brown, C. (1982). *Computer vision*. Engelwood Cliffs: Prentice-Hall.

[10] D. Forsyth and J. Ponce, *Computer vision*. Upper Saddle River, N.J.: Prentice Hall, 2003.

[11] MICE Magazine. (2019). *Issue Three: Ghost Intimacies*. [online] Available at: <http://micemagazine.ca/issue-three>

[12] M. Sonka, R. Boyle and V. Hlavac, *Image processing*. Toronto [etc.]: Thomsson Learning, 2008.

[13] B. Schölkopf and A. Smola, *Learning with kernels*. Cambridge, Mass.: MIT Press, 2009.

[14] S. Mallick, "Histogram of Oriented Gradients | Learn OpenCV", *Learnopencv.com*, 2019. [Online]. Available: <https://www.learnopencv.com/histogram-of-oriented-gradients/>.

[15] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Available: 10.1109/cvpr.2005.177

[16] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms", *Proceedings of the 23rd international conference on Machine learning - ICML '06*, 2006. Available: 10.1145/1143844.1143865

[17] "Chapter 3: Support Vector machine with Math.", *Medium*, 2019. [Online]. Available: https://medium.com/deep-math-machine-learning-ai/chapter-3-support-vector-machine-with-math-47d6193c82be.

[18] "What is underfitting and overfitting in machine learning and how to deal with it.", *Medium*, 2019. [Online]. Available: https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76.

[19] Agile Alliance. (2019). *What is a Kanban Board?*. [online] Available at: https://www.agilealliance.org/glossary/kanban