

# Gold Crest Award

An Investigation into the Applications of AI in Air Defence Systems

By George L, Tim K, Joe C, Bai Gao L, Piotr B

AGSB

Supervised by Dr Steven Squire

## ABSTRACT

This project involves the use and application of artificial intelligence (AI) object detection algorithms and neural networks in the field of modern defence systems. We are investigating the use of the You Only Look Once (YOLO) model, as well as Region-based Convolutional Neural Networks. We will evaluate the efficacy of each approach in the context of our project – detecting a drone using limited resources (EdgeTPU/Raspberry Pi 4). We will also create a program to control a small firing mechanism, which will be custom-made by us to fit this project. The aim of this is to prove that not only can AI be used to identify targets, but also effectively be implemented with minimal resources to reduce spending on defences.

## CONTENTS

Abstract.....	2
Contents.....	2
Project Aim .....	4
Wider Purpose .....	5
Project and Time management.....	6
Currently Employed Defence Systems .....	8
The Ethics of AI .....	9
AI in Defence .....	10
Types of artificial Intelligence.....	11
How does AI work? .....	11
The Cost Function.....	12
Cross-entropy Loss.....	12
Activation Function.....	13
Back-Propagation .....	14
Training.....	15
Issues with Training .....	17
CNN .....	18
RCNN .....	19
Fast-RCNN.....	21
Faster-RCNN.....	22
YOLO .....	23
Implementation of Target Recognition .....	24
Creating the Dataset .....	25

Design specification.....	27
Model.....	28
Firing Mechanism .....	28
Loading System.....	28
Projectile Type .....	28
Rotation.....	29
Firing Mechanism .....	31
Change of Plans.....	33
Creating the model.....	34
Training the AI .....	34
Metrics.....	34
Edge TPU.....	36
Model Size .....	37
Input image size.....	38
Programming and implementation .....	39
Discussion .....	52
Implications and Limitations.....	53
References.....	54

## PROJECT AIM

The present study investigates the application of AI in defence systems, most notably surface-to-air defence. An artefact will be produced alongside this study examining the feasibility of using computer vision models in anti-drone defence systems.

Maintaining a robust defence infrastructure, including personnel, equipment and facilities is essential to ensuring the safety of a country's citizens. The UK's defence budget between 2021-2022 was £45.9 billion. This is expected to rise to £51.7 billion in 2024/25.[1] As new armaments are developed, there becomes a necessity for new forms of defence. The first recorded serious development of Surface-to-air missile (SAM) took place in World War II, when Friederich Halder proposed a "flat rocket" concept.[2] Since then, surface-to-air defences have become more sophisticated. The implementation of AI in surface-to-air defence is a relatively new development and will be the focus of this study.

Regarding modelling our defence system there were many approaches we could have taken. Considering our limited resources, our options diverged to two ideas. In all designs we will prototype an omni-directional firing mechanism; in the first design this will be used in tandem with an electromagnetic coil gun (see Model); the second design will model real-time tracking using a laser. Of course, modelling using a laser is not a truly accurate modelling of anti-ballistic missiles, however there is evidence to suggest that this may become technology in the near future.

Our main objective lies in exploring the application of AI in modern anti-aircraft systems. For us, this means breaking down and researching how AI is programmed. Our focus will be on utilizing computer vision in our own model to simulate the target recognition aspect of modern defence systems. An important part of our research will be deciding which object detection model is most fitted to our project. Popular examples of object detection models are RCNN and YOLO (see page 9). Furthermore, we will use computers to control the movement of our model.

Aside from the investigation, it is worth noting that our group is composed mainly of computer scientists. We hope to learn essential skills in AI and engineering from this investigation that will apply to what we are doing in the future.

[1] UK Parliament (2023) 'UK defence expenditure'

[2] Wikipedia 'Surface-to-air missile'

## WIDER PURPOSE

The aerospace industry has been around for a long time, and from the very beginning, it was clear that if they were to function properly, they needed a dedicated place to take off and land from, and thus the airport was born. Its sole purpose was to make sure the plane and its passengers had a place to land and depart from, and that is largely the same role it fulfils now.

Sure, there are many shops and other attractions inside airports, but its main purpose remains to be a hub dedicated to aircraft. But for it to fulfil this purpose, the airspace around it must be clear of any flying objects. This is why, in the UK, any airspace above 500 to 1000 feet of a property is no longer owned by the individual who owns the land, so that aircraft can perform their intended function, i.e. flying.

However, there is also an emerging risk to the airports themselves, which are drones, and with this project, we want to show the risks of having a drone obstructing aircraft in an airport, and how these risks can be combatted, to make life more pleasant for travellers and airport staff.



The only practical defence that airports really have at this time is waiting for the drones to go away of their own accord, or for it to run out of battery and fall. It is very impractical to 'patrol' the airspace of the airport, particularly as drones are getting faster and harder to detect, particularly with the human eye. Even though drone operators who fly within one kilometre of an airfield can face jail time [3], this would not prevent the actual impact of the drones. This is due to the threat of drones colliding with aircraft during take off or landing, which has the potential to damage the aircraft, causing even more issues. This is why it is vital to pre-emptively research and develop countermeasures to drones, as bringing any sort of mass armament to an airport just to protect against drones would be impractical and unfeasible financially for most airports and would not be guaranteed to work.

This leads us to the conclusion that an automated system is needed, as more advanced drones are also becoming more available to the public, which massively increases the risk of disruption at airports, where the airspace is vital to its operation. A way to combat this is necessary, which is why this project is important. With an automated system, the risk of obstruction will be reduced, allowing for a better airport experience, and less aviation incidents.

[3] The Guardian 'The Guardian view on drones: effective regulation needed'

## PROJECT AND TIME MANAGEMENT

For our project, we are required to build a functioning model working with a team of 5 people over 8 months. For the process to run smoothly, finish on time and to a high standard, we will use Gantt charts and a project management framework to control the workflow in our group.

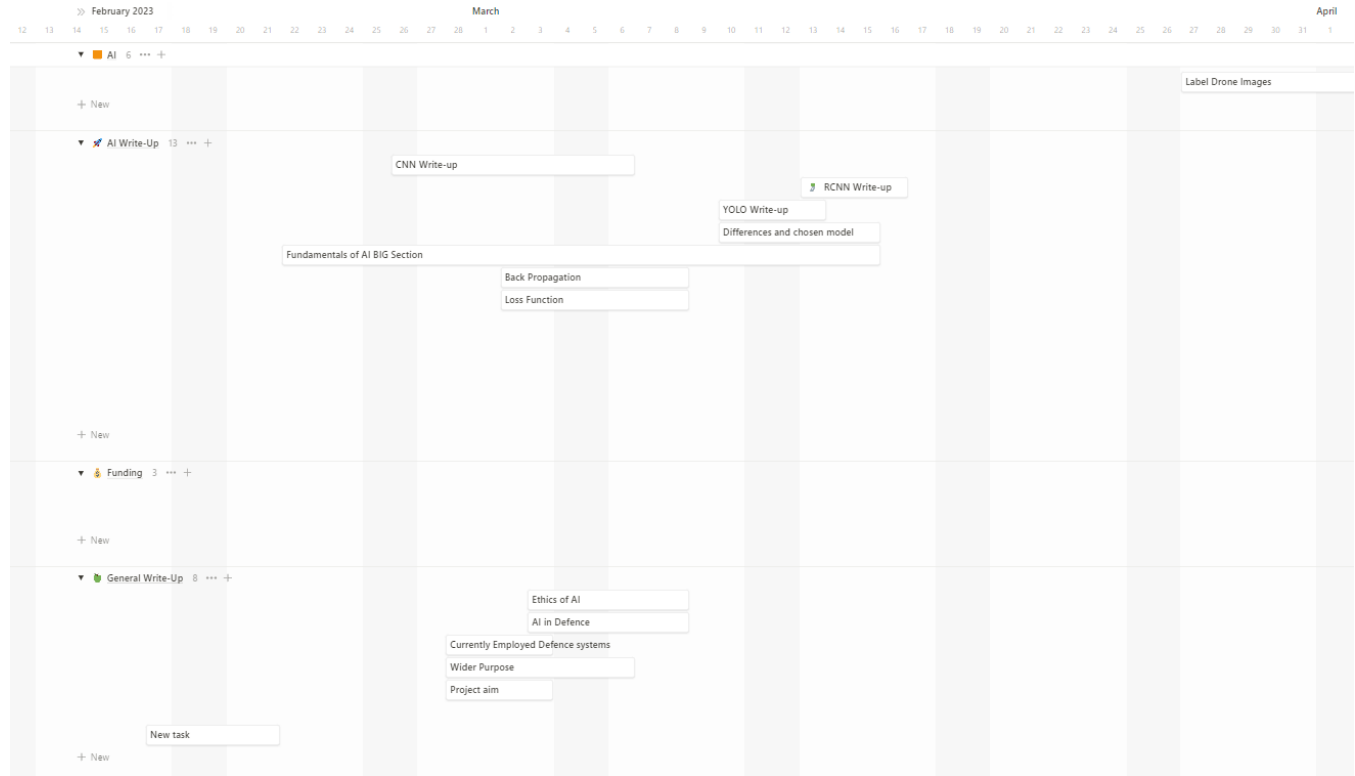


Figure 1 Gantt Chart February

After brief research, it seems an agile methodology is best suited to our project as they emphasise collaboration and frequent feedback, which is easiest with a smaller group of collaborators. Furthermore, we need flexibility and adaptability; a consistent method to ensure this is to allocate a large proportion of time to prototyping our model as this stage is most prone to failure. As agile methodology requires constant collaboration and engagement, we will allocate a minimum of 3 hours per person every week. Agile planning is structured as follows:

- Product vision statement, project aims are established.
- Product roadmap: an outline of goals and priorities
- Product backlog: list of tasks yet to be achieved.
- Release Plan – outlines short term goals.
- Sprint Backlog – micro tasks relating to a specific goal.

[4][5]

Scrum: Assign team members specific roles so that you may complete projects in certain time frames; This is a form of delegation and is generally recommended in almost all team project settings. You may also need a scrum master, who coordinates the development team and keeps members focused. The stages of the agile

scrum approach involve a sprint, which is a period in which the team works towards a short-term goal. Sprint planning, daily scrum, which is a short meeting to discuss progress, sprint review which is a chance to present to stakeholders and a sprint retrospective.[6]

Perhaps the most important tool for planning this study's programming section is the Kanban Board; all tasks, deadlines and collaborator allocation are noted on Notion.[7]

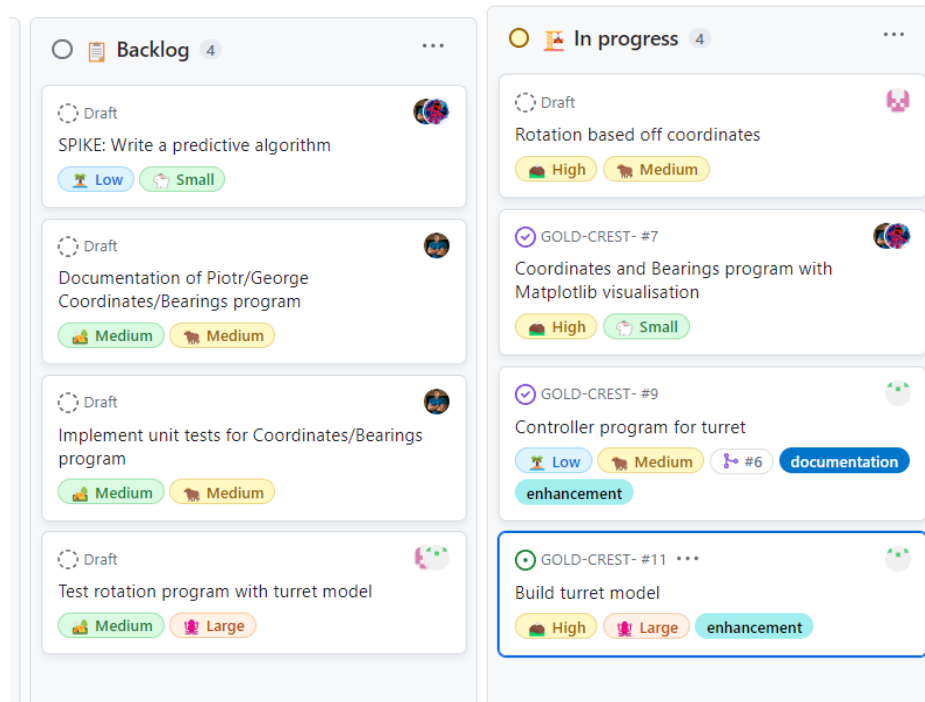


Figure 2 Backlog

We have left suitable gaps in the project timeline to account for internal school exams and year 12 work experience.

[4] Adobe (2022) *Beginner's Guide to Agile Project Management*

[5] Laoyan, S., asana (2022) *What is Agile methodology?*

[7] Notion. Project management application Available at: <https://www.notion.so/>

## CURRENTLY EMPLOYED DEFENCE SYSTEMS

There are four forms of systems that are used against aerial targets:

- 1) Gun systems
- 2) Missile systems
- 3) EMP systems
- 4) Lasers

A gun system is one that uses a kinetic projectile to neutralise an incoming threat, generally a large calibre cannon that can provide enough energy to propel the projectile with enough velocity and destructive force to damage an extremely mobile and/or fast-moving target. Since bullets cannot be guided mid-flight, a modern method of counteraction is to fire many rounds in a short time span to enable the highest chance of hitting the target.

Gun systems use a variety of propulsion mechanisms. Some examples are:

- Air pressure propulsion
- Electromagnet propulsion
- Explosive propulsion

Air pressure propulsion uses high pressure air or other gases in order to utilize the rapid expansion of these gases upon release from a container to accelerate the projectile to extremely high speeds.

Electromagnet propulsion uses either solenoid electromagnets or connected magnetic and conductive metal contacts in order to create extremely strong magnetic fields for a short amount of time, which can accelerate the projectile to the necessary high speeds. The two main different types are coilguns and railguns respectively.

Explosive propulsion uses explosives in order to generate a large pressure force on the projectile due to the heat released, which then accelerates to a high speed as needed.

A missile system generally uses some sort of guided rocket to ram into the target and detonate an explosive charge to neutralise a threat. A missile is commonly guided with the “fire and forget” approach, using infrared or radar [9]. This means that once a missile is fired, it can guide itself to its target, which it tracks using either radio/radar or infrared systems. This, however, can be counteracted with either signal jamming or detection evasion in the case of radar, or flare countermeasures for infrared.

EMP systems are used against smaller aircraft, often those which are remotely piloted. This works by sending a large pulse of electromagnetic waves to the aircraft, which disrupts its communications systems which in the case of it being a remote piloting aircraft, it disrupts the piloting system, neutralising the aircraft.

A laser system concentrates a high-power laser beam onto a target, which can disable it by destroying its sensors or by physically destroying it. This system is incredibly costly but is also highly effective as lasers are just highly energetic light, so therefore travel instantly and removes the need for any trajectory prediction software [8].



[8] - [Lasers Technology Targets Mini-UAVs | Defense content from Aviation Week \(archive.org\)](#)

[9] - [Fire-and-forget - Wikipedia](#)

## THE ETHICS OF AI

Many of the issues that come from AI all arise from the simple issue that AI is unreliable, and that responsibility cannot be easily placed upon just one person. Determining target friendliness is a major issue in the field of defence systems as it is of utmost importance that the mishaps do not occur. If such does happen, then a myriad of legal problems would appear. Anyone deploying the system could be prosecuted, but also the developers and manufacturers of the system as well, putting the livelihoods (and literal lives) at the mercy of something uncontrollable [10]. Ensuring maximum accuracy is also a potential point of concern, as missing a shot (e.g. in the case of a railgun, which usually use large firing loads) could be highly dangerous, depending on where the projectile is likely to land, should it miss.

Another concern is the possibility of malicious users hacking and the hijacking the system, as the software-controlled nature of the system is susceptible to a cyber-attack, especially if there is no mechanical, manual failsafe to disengage or control the system. This is an issue because this is a problem that only exists because of the software-based nature of these systems.

The most infamous issue with the ethics of the introduction of AI is the potential for human job losses that will occur because of its use. A whole sector of jobs regarding the use of manual defence systems could be at risk of being lost to AI, if its development is particularly successful since AI is extremely powerful, and as processing power improves, is likely to become even more dominant in all fields of society. [11]

On the other hand, there could be some benefits to having AI in defence systems. A well-designed AI is more reliable and accurate than any human could ever be, which renders some concerns such as danger of misfires much less potent. Perhaps leaving moral issues to an insentient program like an AI will relieve the burden from those who could have been in its place, as operating defence mechanisms is often considered to be an extremely stressful position.

Additionally, the issue of job losses may be insignificant in the long run as more importance is placed in the field of AI in technology, increasing the number of jobs in that sector instead. There is even a possibility that there is no need to completely eliminate humans from active duty, and that AI could be used to enhance the performance of people.

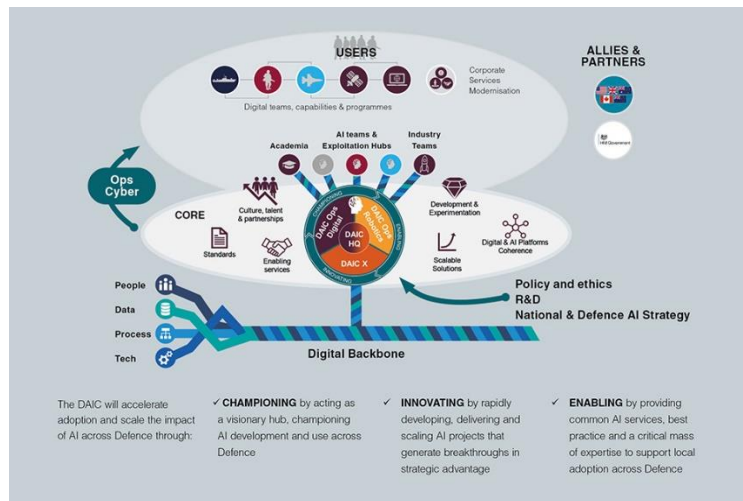
The black box effect is one that states useful information can be gained without the understanding of the inner workings of a mechanism. With context to AI, the concern is that because of the self-sufficient nature of AI, a human can never understand how the AI is operating, and therefore can never change specific parts of the AI. In the field of defence systems, this means that humans cannot adjust the output of the AI to increase precision or accuracy on the low-level scale, which could be dangerous.

[10] - [Artificial Intelligence and Civil Liability \(europa.eu\)](#)

[11] - [Ethical concerns mount as AI takes bigger decision-making role – Harvard Gazette](#)

## AI IN DEFENCE

The use of AI in mechanised defence systems is not a novel concept. As an example, the British Ministry of Defence published a paper on the 15<sup>th</sup> of June 2022 (Defence Artificial Intelligence Strategy) [12] about the integration of AI in the defence force to overhaul the entire process at all levels of command. The implementation of such AI technologies is going to be placed on the proposed “Defence AI Centre”, whose architecture looks like this:



Some specific technologies that will be employed by the MoD are Project SPOTTER/SQUINTER, image analysis AI's used to aid in surveillance optically (SPOTTER) and with radar (SQUINTER) via the use of various AI models such as YOLO, Mask R-CNN, and CenterNet; and Project SAPIENT, another image analysis AI used to help alleviate operators' cognitive burden by only choosing to show relevant data.

Artificial intelligence is also used in the training of military personnel in the form of simulations, [12], where AI is used to simulate certain situations such as military soldiers, civilians, or even vehicles. These simulations can help prepare personnel so that they are aptly equipped and knowledgeable for the real world by bypassing the need for theory via the direct accumulation of first-hand knowledge without physical repercussions or can be used to model certain strategic events to visualise the efficacy and predict their outcomes.

AI can also be used in transportation. AI technology is already being used in transportation in the form of self-driving cars, for example Tesla cars. For defence purposes, AI could be used in the transportation of crucial supplies through rough terrain by using the AI to path find and balance the machine(if it must), such as the concept carriers demonstrated by Boston Dynamics, the BIGDOG from 2004 and the LS3 from 2010 which were used by the US military to carry gear in harsh conditions [13].

However more relevant to our project are the AI assisted “autonomous weapons systems” (AWS) [14] that are used as “automatic defence systems”, used to defend areas of interest or areas of importance. Such systems are used to defend a plethora of different sectors, such as a base from intruders or even vehicles from potential projectiles. These are used when high reaction speeds are required in a given situation, such as a “close-in weapons system”(CIWS) for naval ships[15], or an automated sentry turret.

- [12] - [Defence Artificial Intelligence Strategy - GOV.UK \(www.gov.uk\)](https://www.gov.uk/government/policies/artificial-intelligence)
- [13] - [Legged Squad Support System - Wikipedia](https://en.wikipedia.org/wiki/Legged_Squad_Support_System)
- [14] - [Lethal autonomous weapon - Wikipedia](https://en.wikipedia.org/wiki/Lethal_autonomous_weapon)
- [15] - [Goalkeeper Close-In Weapons System \(CIWS\) – Missile Defense Advocacy Alliance](https://missiledefensealliance.org/goalkeeper)

## TYPES OF ARTIFICIAL INTELLIGENCE

Artificial intelligence is a term coined by John McCarthy in 1956, meaning “the science and engineering of making intelligent machines”. Since then, the definition has been changed, as the prevalence of AI in the media has also changed. Unlike the general definition given by McCarthy, we think of artificial intelligence purely as computers that attempt to mimic intelligence. This difference in definitions arises from 3 key factors: the fear instilled in us from popular fiction pieces about machines enslaving humanity, and therefore attempting to forget that such a thing is not impossible, and from the fact that more advanced types of AI do not yet exist.

There exist 4 main types of artificial intelligence: reactive machines, limited memory, theory of mind and self-aware. The types are listed in order of sentience, where reactive machines are not sentient at all. For the purposes of this project, our scope of the definition of AI is limited to reactive machines and limited memory, where the machines use past data (past experiences) to make more informed decisions in the future.

## HOW DOES AI WORK?

Artificial intelligence attempts to approximate stimuli-response relationships through mathematical functions. This can be achieved through several different methods. The simplest is hard coding the stimuli-response relationships into a function, which is suitable for simpler problems, such as a computer making a move in noughts and crosses. In our project we will use this type of artificial intelligence to predict the flight path of the target drone and fire at where we will expect the drone to be assuming it continues its current course.

A more advanced type of artificial intelligence are neural networks. This is where a group of stimuli-response relationships are modelled by a network of points (nodes) and connections (neurons), where each connection has a weight associated: the impact that the node from the previous layer has on the next. A layer is a group of neurons that are linked to the previous and next layers and share common algorithms. The first layer of a network is the input layer, and the final layer is the output layer. All networks have at least one “hidden” intermediate layer, where the activation for each neuron is based on a weighted sum of activations in the previous layer plus bias. The whole network represents a mathematical function, with  $n$  dimensions, and the goal of training the model is to get the function as close as possible to the wanted values, or in other terms, find the minimum value of the function. While it is nearly impossible to ever find the true minimum value of the complex function, we attempt to get as near to the minimum as possible. This is done through back-propagation and the cost function.

## THE COST FUNCTION

The cost function, also known as the loss or objective function, is used to determine how well a model's predictions match the ground truth (actual values). A machine learning algorithm works by trying to minimise the value of the cost function, i.e. the model's predictions are closer to the ground truth and hence more likely to be ideal output of the model.

There are several cost functions depending on the target output of the model, as to best compare the predictions and ground truth.

For regression problems, the mean squared error (MSE) cost function is preferred. As the name suggests, the function works by finding the mean of the square of the difference between the predicted and ground truth values.

For classification problems, cross-entropy loss, also known as log loss, is used. This quantifies the dissimilarity between the ground truth class labels and the predicted class probabilities.

The YOLOv5 model uses variations of both functions together, as object detection is both a classification and regression problem: binary classification (in our case anyway) for whether a drone is present, and regression for bounding box coordinates.

Mack, C., *Machine learning fundamentals (I): Cost functions and gradient descent*

Shah, Saily. (2021) *Cost Function is no rocket science!*

## CROSS-ENTROPY LOSS

Entropy is a key concept in information theory, first proposed by Claude Shannon in 1948. It is a measure of the amount of uncertainty or randomness associated with a signal. A distribution which has "less surprise" (where likely events dominate) will have a low entropy whereas a distribution where events have equal probability will have a larger entropy. Cross entropy builds on the idea of entropy. In machine learning, cross entropy loss measures the dissimilarity between the predicted probability and the true probability of the classes. The cross entropy between two probability distributions  $p$  and  $q$  can be interpreted as the average number of bits needed to encode data coming from a source with distribution  $p$  when we use model  $q$ . Cross entropy loss can be calculated as follows.

$$H(P^* | P) = - \sum_i \underbrace{P^*(i)}_{\text{TRUE CLASS DISTIRBUTION}} \log \underbrace{P(i)}_{\text{PREDICTED CLASS DISTIRBUTION}}$$

Usually, the loss function is applied at the output layer of the network. The gradients of the loss function with respects to weights are then computed using backpropagation. (See page 8)

## ACTIVATION FUNCTION

The activation function decides whether a neuron should be active or not. The purpose of the activation function is to introduce non-linearity into the output of the neuron, allowing the algorithm to learn more complex patterns in the data. The activation function is applied in every layer of the neural network, a commonly used activation function is the sigmoid function. The sigmoid function is an example of a non-linear neural network activation function. Sigmoid function takes any real value as input and outputs values in the range 0 to 1.

Mathematically, it can be represented as:

*Sigmoid / Logistic*

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid is commonly used for models where we are predicting probability, as probability values always lie within the range 0 to 1 [16], which is a characteristic of the Sigmoid function. Another activation function is ReLU, which is generally preferred over Sigmoid as it is more computationally efficient. The range of output values for ReLU is from 0 to positive infinity, which can be useful for capturing greater variation in the input data. Mathematically, ReLU can be represented as:

## ReLU

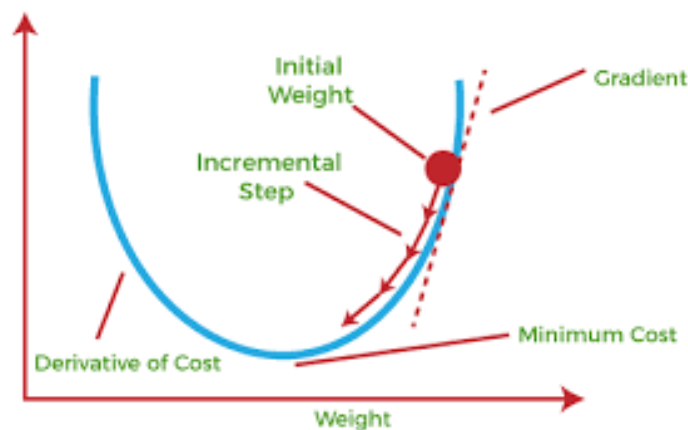
$$f(x) = \max(0, x)$$

Situations where ReLU may not be as advantageous as Sigmoid are in cases where outputs must be constrained between 0 and 1. Furthermore any negative values inputted will result in zero. This is known as the dying ReLU problem, where neurons can become “dead” and not contribute to the learning process of the network. You can avoid the dying ReLU problem by using either Batch Normalisation or the Weight Initialisation method.

[16] <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>

## BACK-PROPAGATION

As previously mentioned, the key part about the function of neural networks is that it uses the cost function and back propagation to learn. This is at the heart of AI, and what makes it able to do so many revolutionary things, that would have seemed completely impossible without years of gruelling research. To do this, we need to be able to objectively evaluate what the AI has done, so that it can change its own parameters it to make it more accurate. For this, we use what is known as back-propagation. This is an algorithm that allows use to evaluate the gradient of the cost function of input data and compare it to the desired data, and then makes small changes to the weights of the neurons so that the cost function is minimised. We use the cost function to do this, as it maps the values associated with the neural network to a real number, which allows us to calculate the difference between the output from our neural network and the desired output. This is done by computing what is known as the gradient or slope, which models how much a parameter needs to change to minimize the cost function. These parameters are the weight of the links between the neurons in the network, and by finding the local minimum of the gradient, we are optimising the network to be as efficient as possible.



It is very hard to achieve the perfect value of the cost function, i.e. that it is completely minimised, as it is not guaranteed to find the global minimum of a function, only the local one, but is something that can be worked with and for most purposes, a local minimum is sufficient anyway. To implement this, we must adjust the weights of the connections in the network. The 'weights' are simply how strong the links are between each individual neuron in the network, allowing the neural network to change its neural connections fluidly.

The weights are adjusted according to the sum of all the desired changes per layer for each neuron. This occurs because each output node has the same layer of neurons linking to it, but the weights it has with different neurons are already stronger for some for one node, and different for another node. By finding the desired changes for every neuron, i.e. strengthening the neurons with the strongest link and weakening the ones with the weakest links, we can get a vector for each output node of the changes it wants to make.

This is expressed as a vector, due to the network being represented as an "n" dimensional space, meaning that this vector has "n" elements in it. For each layer, all of these vectors are summed up, which leads to one vector, called the gradient vector. This is the average redistribution of all of the weights, and this is applied to the layer. We then move back a layer, and repeat the process, but this time with the previously adjusted neurons as the new 'output' layer. This results in the name of back propagation, as the changes start from the desired output, and this effect is then propagated backwards throughout the network, adjusting weights and biases as it goes. [17]

[17] - [What is backpropagation really doing? - YouTube](#)

## TRAINING

To get the wanted stimuli-response relationships for our model, we must give the model data from which to learn and adjust to. There are 3 fundamental ways of doing so: supervised learning, unsupervised learning and reinforcement learning. Supervised learning is where an untrained model is given inputs and the corresponding target responses. If the model predicts the output correctly, the weights leading to that result are boosted, while wrong outputs lead to weights being reduced (see back-propagation below). Of this type of training, there exist two main types of models: regression and classification. Classification is where a discrete variable is the output, while regression models have continuous outputs.



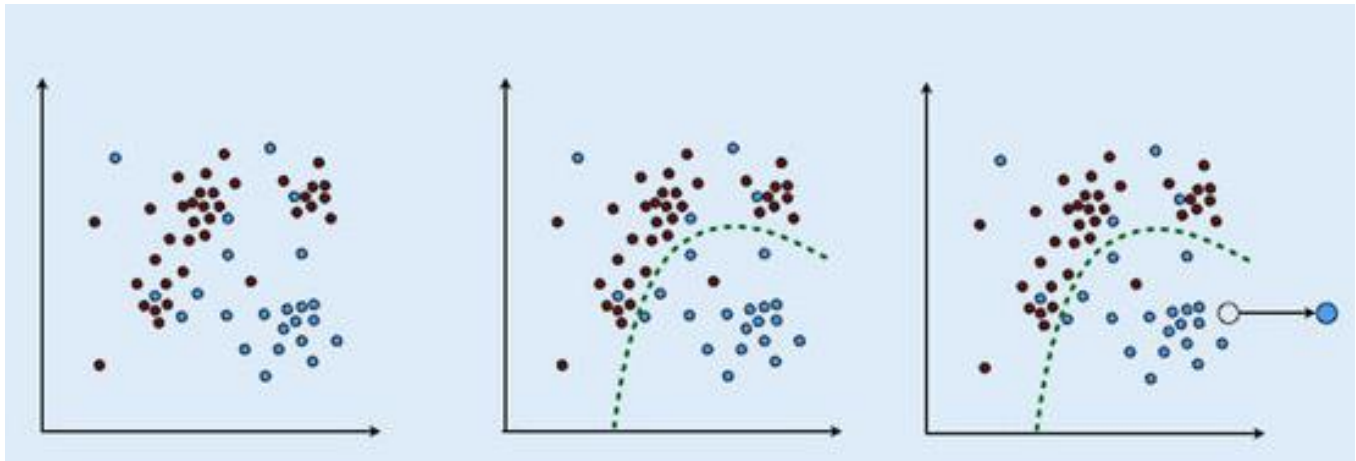


Figure 3 [https://www.researchgate.net/figure/Supervised-and-unsupervised-machine-learning\\_fig2\\_325867536](https://www.researchgate.net/figure/Supervised-and-unsupervised-machine-learning_fig2_325867536)

The figure above shows how a supervised training model uses the given labels (blue and black) to estimate a relationship and predict the state of future inputs, such as assigning the white datapoint to blue, seeing as it is closest to where the majority of blue datapoints lie. The figure shows the training data, the model that has been developed through training, and finally the model being applied to a new input.

Unsupervised learning is where only inputs are given, with no targets to compare to. This is used where we as humans do not know the correlation or relationship between data, so we build a model from a prediction of correlations and relationships. In the past, reliable models have been developed to predict a patient's prognosis of certain cancers, based on data such as tumour size, sex, age, etc. This is called clustering: the model clusters the variables into groups.

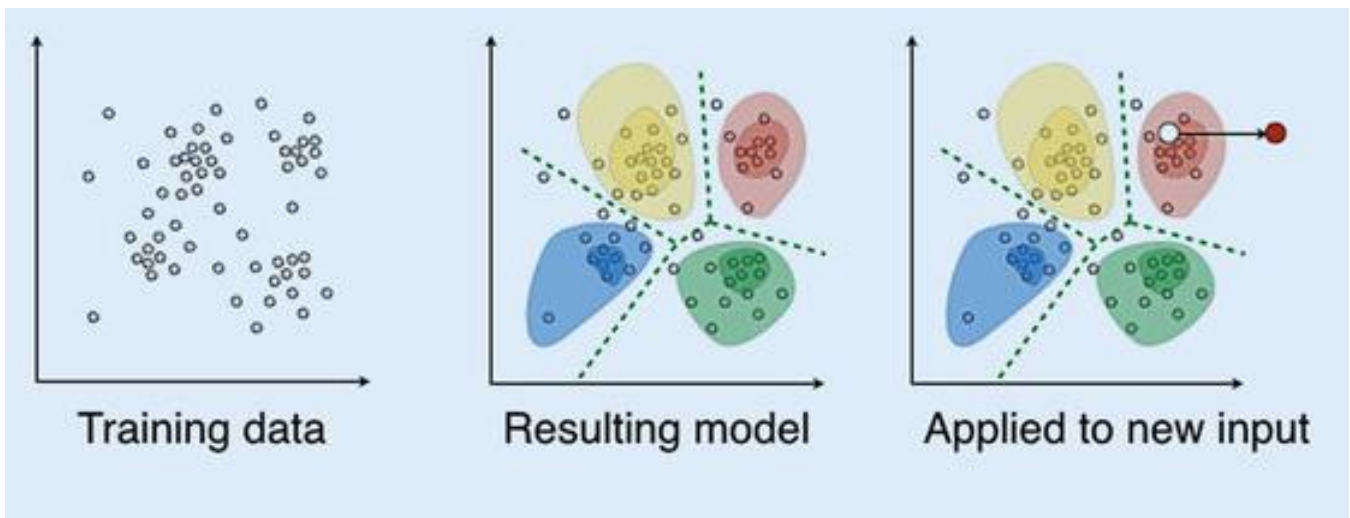


Figure 4 [https://www.researchgate.net/figure/Supervised-and-unsupervised-machine-learning\\_fig2\\_325867536](https://www.researchgate.net/figure/Supervised-and-unsupervised-machine-learning_fig2_325867536)

The figure above shows a visualisation of unsupervised learning. The unlabelled training data has been clustered and categorised into a model, and input data is classified based on the cluster in which it lands, with higher certainties closer to the centre of the clusters.



The final fundamental type of AI training is reinforcement learning, where no data is given, other than a goal to achieve. The goal is typically quantified and gradual, as the model needs to be able to track progress. If progress is achieved, the model is rewarded, making the model more likely to repeat the same successful behaviours in the past, while regress results in lower rewards, meaning the model is less likely to repeat unsuccessful behaviours. An element of randomness is introduced, to encourage the model to seek alternate, possibly superior, choices. This type of AI training is akin to Darwin's Theory of Natural Selection.

For the purposes of our project, we will be using a supervised learning model for target recognition, acquisition, and tracking, as we will have well-defined inputs and outputs.

## ISSUES WITH TRAINING

When training a network's weights, several issues can occur when trying to reach the optimum point. One of the more frequently encountered problems is oscillation. This is where the cost oscillates up and down due to the learning rate (rate of gradient descent) being set to a high value for that model. This causes the network to overshoot the minimum, and take longer to reach the optimum point, as it has to make overall small progress through the extreme swings past the optimum. This is akin to water swirling around the side of a funnel before eventually making it through, as opposed to slowly going straight down.

An extreme version of oscillation is instability. This is where the learning rate is even higher, meaning that not only does it overshoot the optimum, but the model becomes less accurate after successive epochs of training. This snowballs and can lead to an infinite value for cost, which ends up making the model totally unusable.

All these issues can be easily remedied by retraining the network, with a lower learning rate. However, if this is set to a value which is too low, gradient descent per epoch can be too small, resulting in wasted time and computation. Therefore, it is necessary to find a good starting learning rate and monitor the loss and accuracy graphs.

Another common problem plaguing the training stage, is for the network to be stuck in local optima. This is where the graph has an optimum point in the local scope, but if you look further across the graph of the cost function, there is a more optimum point. This is normally fine, unless the learning rate is too small for the magnitude of this local optimum, at which point, training gets stuck in this local optimum. This is like being in a valley, and not being able to get to a deeper value, as there are mountains between. The set of weights that lead to this local optimum is known as a Basin of Attraction. While you cannot spot the issue occurring on its own, you can spot and avoid the issue by retraining the network with different starting weights and retrain several times and pick the model with the best accuracy, as that is the most optimum, so least likely to be a local optimum.

Another problem that can occur while training networks, is plateauing. This is where the loss seems to reach a constant value. This can occur either when in a local minimum (as explained above), or when nodes are "dead" or "saturated". The former means that a node is towards the minimum value of activation for the activation function used by that node, while the latter means that the node is towards the maximum value.

This means that the cell will be affected by back-propagation far less, meaning that the weights and biases of that node will take a long time to change. This can be overcome, through the use of the ReLU activation function, which has no maximum, hence cannot become saturated. However, it is still susceptible to dying, so biases should all be positive at the start of training.

## CNN

A convolutional neural network (CNN) is a deep learning neural network used for image detection. CNN works by assigning weights and biases to various objects in the image, to differentiate them from one another. In CNNs you have convolutional layers, which extract features from the image, a pooling layer which reduces the size and resolution of the feature map, and fully connected layers which classify the image into known objects.

When creating the convolution layer you specify the kernel, which is a square matrix of variable size which contains values used to identify features within itself. The kernel, when convolved with the input image, acts as a moving processing unit which performs a calculation at its current position, usually a matrix multiplication. It will then return the value calculated at each position into a new feature map. In the case of images with multiple colour channels, the kernel has the same depth as the input image. The number of pixels by which the kernel shifts each time it moves across the input feature map is known as the stride length - a higher stride length results in a lower resolution output as the kernel will perform calculations over a lower number of positions. The benefit of using a higher stride length is that it will use less computational power which can be more cost efficient.

Operations in the convolution layer usually result in a smaller feature map [12] than the input, due to the size of the kernel. You can use padding, where the size of the feature map is increased without adding new data, for example its matrix dimensions are made larger, and the new space is filled with zero values to ensure the kernel will also perform calculations on the edges of the feature map. Padding is also applied so that the output image will have the same spatial dimensions as the input image, for example when we augment a 5x5 image into a 6x6 image and apply a 3x3 kernel, the output image turns out to be dimensions 5x5, meaning that the size of the feature map which contains information is not changed in size.

The aim of the Pooling layer of a CNN is to reduce the dimensions of the input feature map, to reduce computational strain and to avoid overfitting. Overfitting occurs when the model has been trained for too long or with an overly complex model; a problem that causes the model to exactly fit the training data but is not able to accurately predict with new data, making the model useless. Furthermore, it is useful for extracting dominant features in an image, as when the feature map is scaled down to a lower resolution, the dominant shapes remain visible, whereas less relevant small details are discarded. There are two types of pooling, max pooling, which provides the highest data value from the portion of the image covered by the kernel, and average pooling which returns the average of all data values in the kernel.

The final layer of CNN is the FC layer (Fully Connected layer). The output Feature Map from the convolutional and pooling layers represents key features of the data and is passed into this layer in the form of a flattened vector matrix. This means that the rows of the matrix are laid out lengthwise, i.e. a 5x5 Feature Map is

flattened to a 25 unit long string of values. This is then fed directly into a standard neuron-based system, which can be taught to recognise the now-flattened dominant features. The FC layer is a cheap way of learning non-linear combinations of input data in the form of Feature Maps. In the FC layer, a linear transformation is first applied to the flattened input vector through a weights and biases matrix, and then through a non-linear activation function.

The column in the weights matrix which is active will have different numbers and would be optimised as the neural network is trained. By taking the dot product of the input vector and the weights matrix and applying the non-linear transformation function, we can get the output vector.

## RCNN

R-CNN stands for region-based CNN. This works by first predicting the regions in which points of interest may lie. Each region is then processed by a CNN (described above) to find which category, if any, the object falls under. The region prediction is an unsupervised AI, finding clusters of data and predicting them to be all one object.

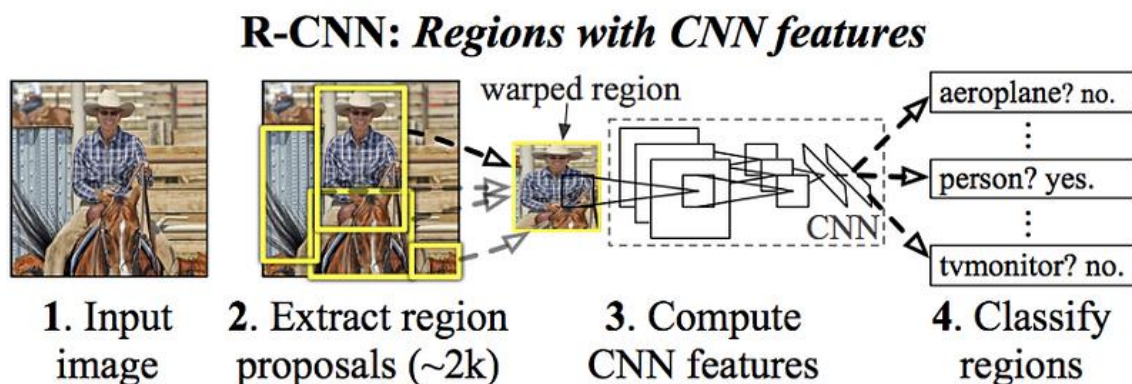


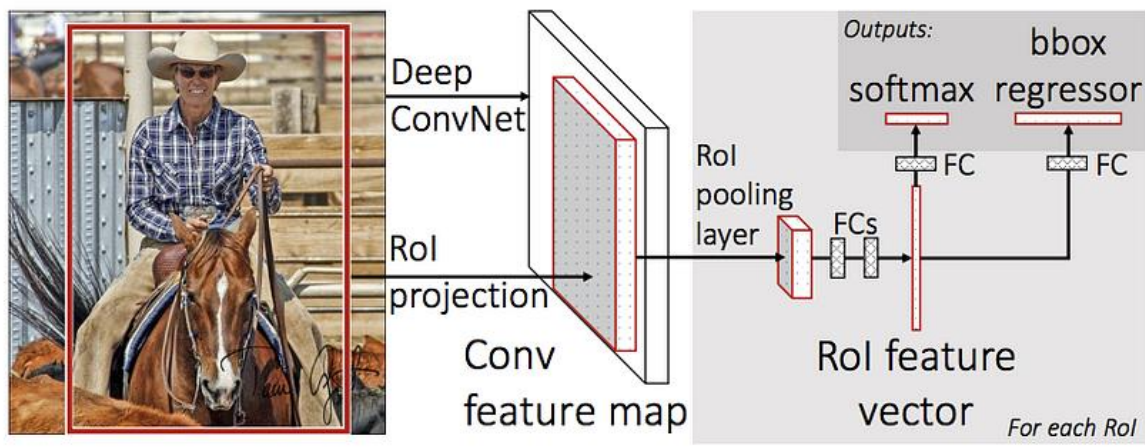
Figure 5 <https://arxiv.org/pdf/1311.2524.pdf>

A selective search is used, finding 2000 possible regions of interest, called region proposals. The selective search works by first finding all possible clusters of pixels. Each of these are recursively combined with close similar regions, to become one larger region. This is repeated until only around 2000 region proposals exist.

R-CNN is however quite inefficient, as each image needs around 2000 predictions through the CNN, which is very computationally demanding. This means that a R-CNN is completely impractical for our project, as we require near-real-time object detection, and on a less computationally powerful machine- a Raspberry Pi. Finally, R-CNN has a flaw in the selective search algorithm. As it is fixed, that means that no learning can happen, and the algorithm may keep selecting bad region proposals, which would hamper the accuracy of the system.

Due to these weaknesses, Fast R-CNN was developed. Fast R-CNN fixes both the issues of R-CNN: the computational load and the fixed algorithm for determining candidate regions.

## FAST-RCNN



The input image is first passed into a Deep ConvNet, which is a type of CNN that can have hundreds of layers, allowing it to learn more complex features in an input image. However, this requires increased computational costs. This outputs a convolutional feature map.

Next is ROI pooling, which determines the class label of each object. Pooling takes a convolutional feature map, and ROIs as input. Pooling is used to down sample features, and to ignore minor distortions in input e.g., rotations in an image. For ROI pooling, max pooling is usually used on a region of interest. ROI pooling will divide each region of interest into a fixed number of cells and applies max pooling to obtain a feature vector. In ROI pooling, the stride is quantized meaning it will round to the stride to the nearest integer. This can lead to a loss of data. The output of this leads into the fully connected layers.

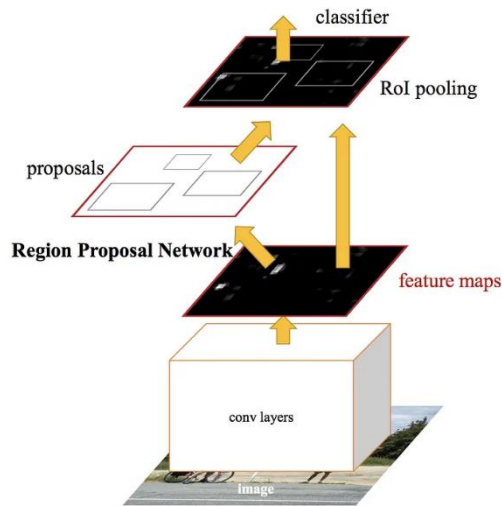
These feed into a SoftMax, which gets the class of the object, and feed into a “bbox regressor” - a function to predict the offset of the bounding box of the object.

The creators of fast-RCNN tested their models and found that fast-RCNN was 146x as fast as their original RCNN. This was all despite taking just over a tenth of the time to train. This is already much more suitable for real-time computer vision.

However, the creators of RCNN and fast-RCNN still felt improvements could be made with the speed and accuracy of region-based convolutional neural networks.

## FASTER-RCNN

In faster RCNN, a new layer is added (RPN, region-proposal network). The main idea behind Fast-RCNN is to create bounding boxes over ROI for the entire image just once. In comparison, RCNN would perform region extraction for each ROI separately. Faster-RCNN builds upon fast-RCNN, through the use of the RPN, which is trainable.



Like in fast-RCNN, the input image is passed into the deep ConvNet, creating a feature map, which is passed into the RPN as an input.

The generation of these regions of interest is achieved through the RPN (region proposal network). RPN works by sliding a fixed size window over the image in order to create anchor boxes with varying aspect ratio and scales depending on the training data. This is similar to kernels in CNN, however generation of anchor blocks is generally done from predefined knowledge of the objects detected, whereas kernels are learned during training. Anchor boxes are also much larger than kernels. It then returns the probability that each box contains an object. This is done through the classifier. The RPN then identifies whether the object in the anchor box is a part of the foreground, or the background and adjusts offsets accordingly. An RPN will learn a set of regression offsets in order to adjust the coordinates of the anchor box to fit the object inside of the box.

Fast RCNN uses multi-task loss. This is because Fast RCNN requires a classification task for predicting object labels, as well as a localisation task for predicting coordinates of bounding boxes. Multi-task loss combines these into a single loss function which can help improve accuracy.

## YOLO

In comparison to RCNN, YOLO is fundamentally different, mostly due to one key aspect: it only uses one forward pass over the image, unlike RCNN, which uses two. This leads to the name You Only Look Once, which is what the acronym YOLO comprises of. The main advantage of YOLO is that it is much faster than RCNN and its variants. By only using one forward pass, it has a much higher FPS than RCNN, as YOLO can process over 80 frames per second of images and video, making it much more suitable for real-time applications. This results in it being used very widely across the field of computer vision, as real-time object detection is very relevant in today's world. Of course, such speed cannot be reached with perfect accuracy, and this results in a lower MAP (mean average precision) compared to RCNN, but this is more than acceptable with its speeds.

YOLO works by firstly dividing up the image into  $N \times N$  equal boxes (squares), which each have the responsibility to localise and predict the class of an object in itself, and give the confidence (i.e. the probability of success) of its prediction. This prediction can be achieved due to this being done in the final two fully connected layers of the whole network, with the previous layers being fully made up of convolutional layers, which are pre-trained using ImageNet. ImageNet is a massive database of over 14 million pictures, and is used by visual object recognition algorithms, such as YOLO, to be much more accurate in its predictions. This is due to all of the images uploaded being annotated by people, which allows networks like YOLO to identify the objects in a given image. [18]

Next, the boxes which do not contain an object are discarded by the network and are not taken into consideration anymore. This now means that bounding boxes are calculated for all the objects. These bounding boxes can overlap the previous boxes we used, and their purpose is to give more precision over where a certain object is. Each box also has a confidence score associated with it, which reflects the confidence over the box containing an object, and the accuracy, dimensions wise, of the box.

Next, we calculate the intersection over union, or IOU, between the calculated bounding boxes and the grid boxes. To do this, we take the area covered by the intersection of the boxes over the total area of the boxes, giving us a decimal less than or equal to one. This is a threshold which we can change, which can allow us to have finer control over the identification of objects. Once all the IOU values are calculated for every bounding box and every grid box, YOLO discards the grid boxes with an IOU below the threshold, and finally picks a grid box in which it thinks the object is most likely to be. Of course, some boxes will all have IOU values above the threshold for a given bounding box, so NMS, or Non-Max Suppression, is used to identify the ideal grid box to include, and we end up with one bounding box per object. [19]

[18] – [YOLO Object Detection Explained - Datacamp](#)

[19] – [YOLO: Algorithm for Object Detection Explained – V7Labs](#)



## IMPLEMENTATION OF TARGET RECOGNITION

“Tiny YOLO” also exists as a possible alternative, as it being a compressed, lightweight version of YOLO makes it run at a significantly higher fps even in comparison to the higher speed nature of YOLO. The cost however is that the accuracy is lower (as fewer processing layers are present in tiny YOLO). Depending on the version of tiny YOLO used, it can be 2x-10x faster than its non-tiny counterpart, making it more suitable for real-time applications.

The main constraint and ultimate decider on which model to use is the device that will be running the model – the Raspberry Pi 4. Even with the Coral Edge Tensor Processing Unit (TPU), a Raspberry Pi is still a weak computer in comparison to a full-sized desktop PC due to lower power, less memory, and no discrete GPU. This therefore makes it a requirement that whatever model we use must be able to be compiled and run on the Edge TPU.

This therefore means that YOLO, or Tiny YOLO should be implemented. Any form of RCNN will be too slow given its two-pass nature. Due to the small cache size of the Edge TPU, any model we use must be small, also meaning only YOLO and Tiny YOLO are suitable.



There are multiple libraries we can use, notably TensorFlow lite and OpenCV. OpenCV helps with the deployment of already trained neural networks, which is better for performance. TensorFlow however is a framework in which neural networks are built upon, which therefore means it is better for when specific datasets are required for the training of an AI, which is perfect for our use case of detecting drones. Since each are different in their own right, both are likely to be used in our final program as the functionalities of both are vast and varying.



## CREATING THE DATASET

Using images scraped from Google and Bing Images, and a recorded drone flight provided by Tim Kolesnichenko, we were able to produce a dataset of 3000 labelled images in Labelbox. We also used so-called “background images” from a Kaggle dataset, adding images of other airborne objects. These will be implemented into our YOLO model in the next stage of the project. Labelling of images involved drawing bounding boxes manually around images representing drones. One of the immediate things we noticed was that inaccurately labelled images affect the accuracy of the YOLO model. This meant that each image had to go through a review process before being accepted into the dataset. Images had to go through an initial labelling, followed by review and if rejected, would go through a rework stage. Over the course of 12 hours, we collectively created a dataset that we hope will provide us with an accurate identification of a drone. There is a large gap in start and completion time due to Year 12 mock exams.

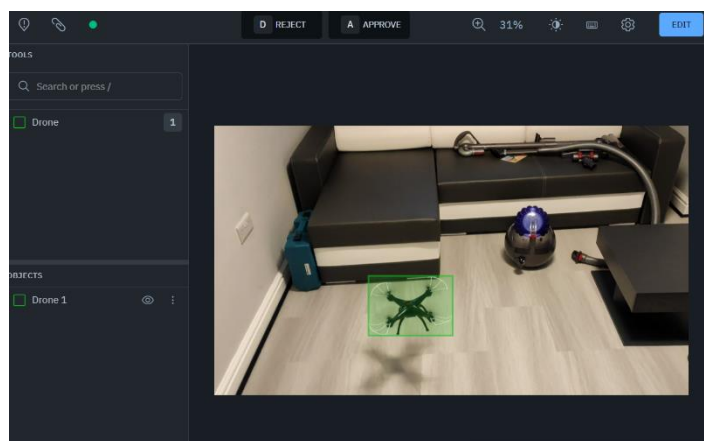


Figure 6 Review Process of Labelling

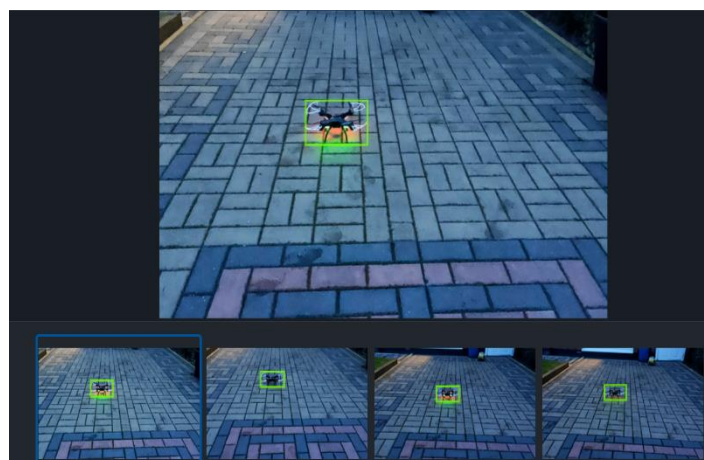


Figure 7 Labelled images taken from drone test flight.



Figure 8 Performance metrics taken from Labelbox. Available at: <https://labelbox.com/>

## DESIGN SPECIFICATION

Within this project, we aim to achieve the following:

Artificial Intelligence:

- To be able to detect and identify aircraft
  - It must be able to tell apart, for example, a drone from a bird
- To be able to differentiate between hostile and friendly aircraft
  - It will be trained to recognize a marker indicating whether an aircraft is friendly

Physical Model:

- To be able to fire a small projectile
  - Ideally something which will not damage the testing target
- To be able to aim within an 180° arc in 2 planes of rotation
  - This is so that it can hit anything it can see
- To be able to fire at least 60 RPM
  - This is to improve consistency in hitting the target
- To be automatically loading
  - This is to save time during testing and usage
- To be able to accurately control firing force and speed
  - This is so that the computations are easier

Computations:

- To be able to predict the trajectory of the projectile
  - This will use vector maths and mechanics
- To take into account local windspeed and air resistance
  - This will involve some basic aerodynamics principles
- To compute the position of the target using 3D stereo vision
  - This involves mostly trigonometry as well as a small amount of input from the AI

## MODEL

### FIRING MECHANISM

As we found in our research into forms of defence systems, we observed that our main options in terms of a firing mechanism would be either a pressurized air firing mechanism or an electromagnetic system. We decided to ignore the prospect of an explosive firing mechanism as it would not cause the projectile to launch at either a safe or controllable velocity, as well as an existing potential for the explosives to damage the model. We also considered using an elastic propulsion system such as rubber band slingshots, but we also decided to not use this mechanism as it could be unreliable and potentially slow.

After deliberation, we decided that of the two better mechanisms, the electromagnetic system would be better for the model. This is because it can be controlled more accurately using a computer system when compared to the air pressure system, which would be controlled by a valve. But since controlling a valve is subject to discrepancies such as frictional forces, as well as the turning motor not being strong, it is likely more problematic than using a solenoid magnet, which is only dependent on the amount of current passing through the solenoid which can be easily controlled by the computer with no intermediate factors.

### LOADING SYSTEM

We initially had a couple of ideas for a loading system which would automatically refill projectiles into the firing mechanism, such as either spring-loaded or gravity-fed systems. Spring-loaded systems have the advantage of being reliable as well as functional in any orientation as the spring force can counteract gravity, however this has the disadvantage of often limiting storage of projectiles to a relatively small size so would require reloading fairly often. Gravity-fed systems are much simpler to create as they only rely on the downward force of gravity in order to load the system and because of this simplicity, it often means that there is less room in the container spent on a spring-loader which means that it can generally hold a larger number of projectiles. We will likely end up using a gravity-fed system but plan to test both in the development process.

### PROJECTILE TYPE

Since we had settled on an electromagnetic propulsion system, it was necessary to use some form of magnetic projectile. We considered either permanent or ferromagnetic magnets, and decided on using small ferromagnetic ball bearings, as they are cheap, mostly harmless, and easy to measure. We will aim to find resources such as these of a good quality as discrepancies could affect accuracy of the machine.

## ROTATION

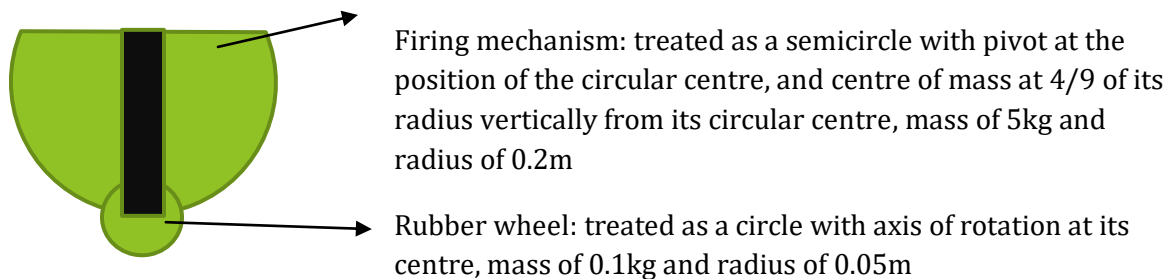
When considering a mechanism for rotation, it must satisfy a few criteria in this particular case:

- It must be able turn with a relatively low overall torque
- It must not impede the function of the firing mechanism itself
- It must be simple
- It must be able to be operated using motors

Following these criteria, a few ideas spring to mind; these include gears, rubber or other high-friction wheels, and pulleys.

Pulleys were quickly disregarded as an idea, due to fact that they operate using strings which means that without some form of track or clips, it would likely be limited to linear motion, whereas we were looking for something more suited for circular/radial motion. Not only this, but the clips which could potentially bind a string to a circular shape are likely merely an extra design element which could be avoided if we were to use a simpler system.

The other two systems were of equal interest to us, since they both perform the desired function simply and efficiently, however the use of rubber wheels which relied on friction to produce torque is at first sight slightly problematic, as there is always a risk of slipping, where the friction is not high enough to cause a potentially quite heavy firing mechanism to rotate against the downward gravitational force. However, I decided to verify this mathematically using rough approximations for values.



Maximum torque is when angle rotated =  $90^\circ$

$$M = F_s = mg \times \frac{4r}{9} = 5 \times 9.81 \times \frac{4(0.2)}{9} = 4.36 \text{ Nm}$$

Which causes a force at the circumference of the firing system of  $F = M/r = 4.36/0.2 = 21.8\text{N}$

Assuming the normal contact force = weight of firing system,

$$F_{\text{friction max}} = F_{\text{normal}} \times \text{friction coefficient} = mg \times 1.15 = 5 \times 9.81 \times 1.15 = 56.4\text{N}$$

(Value for friction coefficient obtained from [https://www.engineersedge.com/coefficients\\_of\\_friction.htm](https://www.engineersedge.com/coefficients_of_friction.htm))

Since the rotational force caused by the weight of the firing system is smaller than the maximum frictional force, the rubber contacts should be sufficient to prevent slipping between the wheels.

Since it has now been determined that the rubber wheel is likely a reliable potential choice of mechanism, it must now be decided which system will be used in our model without the argument of reliability.

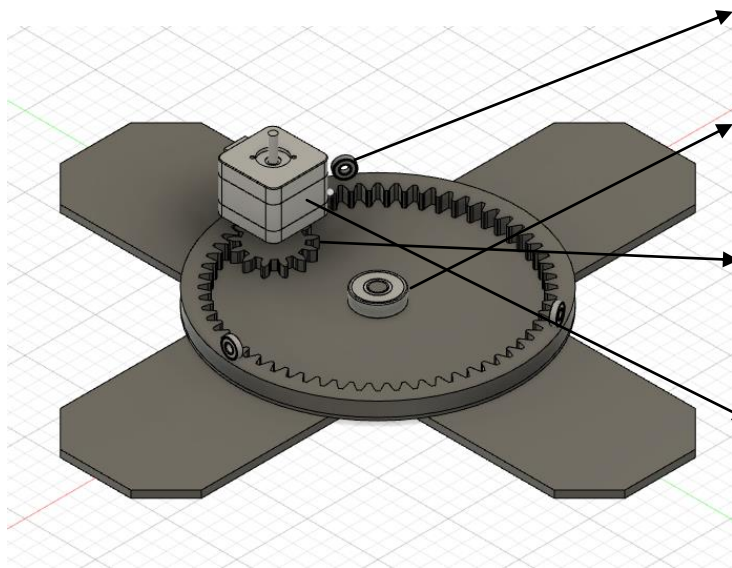
We ended up choosing the gear system, because since the rubber wheels rely on the contact force, it is dependent on the weight of the firing system being transmitted through the rubber wheels at a constant angle. This would require a lot more engineering, as well as the fact that the system may be required to be put under tension in order to maintain contact as the wheel rotates. The gear system should avoid all these issues, as the height of the teeth of the gears mean that the gears can shift slightly and still maintain proper contact, so do not need to be put under tension.

## HORIZONTAL ROTATION

We began designing a mechanism which would rotate the model on a horizontal plane (The yaw axis), and decided on some basic workarounds that needed to be met:

- The system had to have a fairly wide base, so that it could maintain the balance of the system as it rotates higher up the model.
- It also had to be able to rotate around  $360^\circ$  while making sure there are no places in which there are wires that get pulled or tangled.
- It had to be able to rotate smoothly so that the motor that turns it does not have to be too powerful to turn against a large frictional force.

Our first CAD model looked like this:



Edge Ball Bearings: Allow for free, smooth rotation around the track.

Centre Bearing: Further allows for rotation and offers structural support at the centre of the turntable.

Gears: Gear ratio of 59:13 to reduce the work which the stepper motor must do

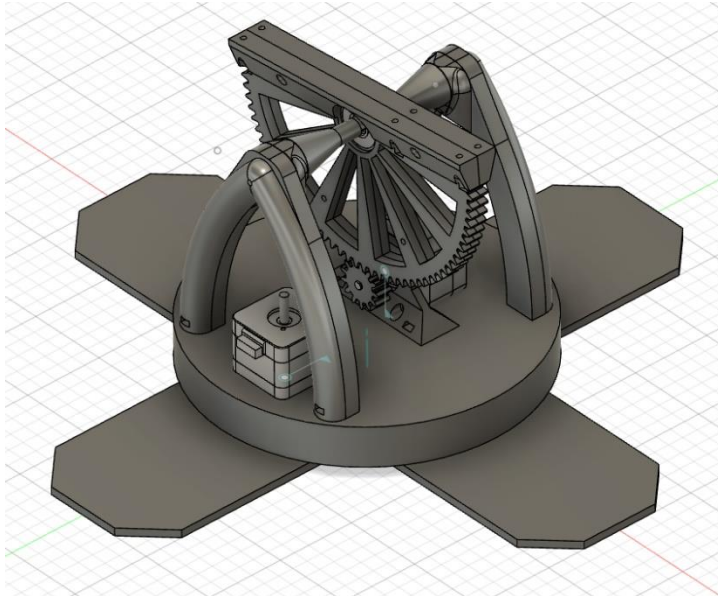
Stepper Motor: Oriented pointing downwards, will be mounted on a turntable which contains the electronics and inner workings of the system, and placed at the edge of the turntable to reduce torque.

Wide Base Plates: Increase stability and decrease wobbling of the model.

Overall, this iteration of the model meets the requirements very well and will likely be used in the final design.

## VERTICAL ROTATION

With vertical rotation, we used the previously investigated system, however developed the idea of the barrel further. Since it would be difficult to ensure that a gravity-fed pellet loading system would always be oriented in a way that makes it functional with the previous design as the barrel was vertically placed, we changed the system's design so that the barrel lay horizontally across the semicircular gear.



The open top visible is the place into which pellets would be loaded, and they would roll downwards into the chamber where they can be fired using the mechanism.

This design is controlled by a stepper motor at the bottom of the gear, and the large semicircular gear is supported by two arches and ball bearings which allow free rotation with low friction.

Both stepper motors here are shown, one is the aforementioned one which controls the vertical rotation plane, and one is the downward facing one which controls the horizontal plane of rotation.

This model overall appears to meet the requirements

## FIRING MECHANISM

For the firing mechanism, as we had already decided on an electromagnetic firing system, we had to perform some calculations in order to find out what specification of components we would need for our circuit. Some components we had already decided on as we had them to hand, and as we are on a tight budget we were trying to save as much as possible.

We had on hand: 8V battery, 100F capacitor, assorted transistors, copper wire of diameter 1mm

We planned to buy steel ball bearings of diameter 5mm, and density  $7.85 \text{ g/cm}^3$  or  $7850 \text{ kg/m}^3$

To lay out the problem initially:

We want to calculate the components' specifications required to accelerate the pellets to the right speed.

The main equations that will be used are:



$$F = (B\mu_0\mu_r A)/2$$

where:

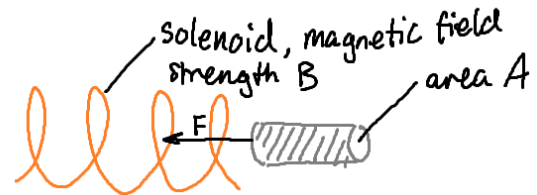
F is the force exerted on the ferromagnetic object (in Newtons),

B is the magnetic field strength produced by the solenoid (in Tesla),

$\mu_0$  is the permeability of free space, approximately equal to  $4\pi \times 10^{-7}$  Tesla meters per ampere (T·m/A),

$\mu_r$  is the relative permeability of the ferromagnetic object (dimensionless),

A is the cross-sectional area of the ferromagnetic object perpendicular to the direction of the magnetic field (in square meters).



$$B = \mu_0 n I$$

where:

B is the magnetic field strength (in Tesla),

$\mu_0$  is the permeability of free space, approximately equal to  $4\pi \times 10^{-7}$  Tesla meters per ampere (T·m/A),

n is the number of turns per unit length of the solenoid (in turns per meter),

I is the current flowing through the solenoid (in Amperes).



$$V(t) = V_0 * e^{-(t/RC)}$$

where:

V(t) is the voltage across the capacitor at time t,

$V_0$  is the initial voltage across the capacitor (at  $t = 0$ ),

e is the base of the natural logarithm (approximately 2.71828),

t is the time elapsed since the start of the discharge process,

R is the resistance in the discharge circuit (in ohms),

C is the capacitance of the capacitor (in farads).

These equations listed were all described upon request by ChatGPT at <https://chat.openai.com>.

Since the barrel is length 15cm, and the wire is width 1mm, we are able to fit a solenoid 150 coils long, and space allows for 5 layers of coils. Therefore, we are able to fit 750 coils into this firing mechanism.

The only remaining component variable is R, the resistance of the resistor in series with the capacitor. This is what the aim is now to find.

We plan to shoot the projectiles at 15 m/s out of the barrel, which as stated is 15cm or 0.15m long.



$$\int F dt = m \Delta v$$

$$\Rightarrow \int \frac{1}{2} B \mu_0 \mu_r A dt = m \Delta v \Rightarrow \frac{1}{2} \mu_0 \mu_r A \int B dt = m \Delta v$$

$$\Rightarrow \frac{1}{2} \mu_0 \mu_r A \int \mu_0 n I dt = m \Delta v$$

$$\Rightarrow \frac{1}{2} \mu_0^2 \mu_r A n \int I dt = m \Delta v$$

given that  $\mu_r = 5000$ ,  $A = \frac{\pi}{4} d^2 = 1.96 \times 10^{-5} \text{ m}^2$ ,  $n = 750$ ,

$$m = \rho V = 7850 \times \frac{1}{6} \pi d^3 = 5.14 \times 10^{-4} \text{ kg}$$

$$\Rightarrow 1.13 \times 10^{-7} \int I dt = \Delta v$$

$$\Rightarrow 1.13 \times 10^{-7} \int \frac{V}{R} dt = \Delta v$$

$$\Rightarrow 1.13 \times 10^{-7} \times \frac{1}{R} \times \int V dt = \Delta v$$

$$\Rightarrow 1.13 \times 10^{-7} \times \frac{1}{R} \int V_0 e^{-\frac{t}{RC}} dt = \Delta v$$

$$\Rightarrow 1.13 \times 10^{-7} \times \frac{V_0}{R} \int e^{-\frac{t}{RC}} dt = \Delta v$$

$$\Rightarrow 1.13 \times 10^{-7} \times \frac{V_0}{R} \times -\frac{RC}{t} e^{-\frac{t}{RC}} = \Delta v$$

NOTE THAT THESE CALCULATIONS WERE NEVER FULLY COMPLETED

## CHANGE OF PLANS

Just as we reached this stage in the development of the project, a deadline for the end of the summer term was imposed by our supervisor and so we had to hugely simplify our project. Since we had very little time to complete the project after this new deadline, we decided to switch to a laser system in order to test whether our AI detection and targeting system was accurate, since it would be far easier to implement in a short amount of time in comparison to a coil mechanism.

## CREATING THE MODEL

### TRAINING THE AI

As we began training the AI, we immediately ran into some roadblocks. The first was that training AI models takes a long time, so we would ideally do this on several computers in parallel. Newer Nvidia GPUs fortunately have Tensor cores, which are optimized for fast tensor calculations, such as those used in AI models. Only one of the team members had such a GPU, but AGSB fortunately had many new RTX 3060 GPUs.

We tried to train the AI models on the school computers, yet the process was unfruitful due to robust antivirus software massively slowing the caching process of the images. The YOLO training algorithm requires images to be cached, so we had to abandon the thought of training our models in school.

Hence, we resorted to training on the one Nvidia machine, which while slow, still got the job done.

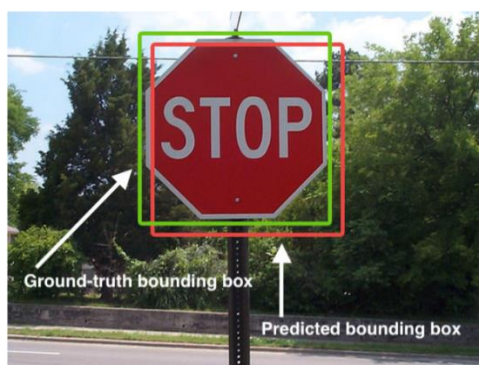
We trained with a range of parameters, from changing the input image sizes, to changing the size (and therefore the complexity) of the models. We also investigated the effect that number of training epochs has on the model.


### METRICS

Throughout this section we will be frequently mentioning several metrics that together help describe the performance of a computer vision model.

#### IOU

IoU is the abbreviation for intersection over union, or the way in which you measure the closeness of the predicted bounding box and the actual bounding box. (Shah, 2022). The typical IoU threshold to decide whether a detection is correct or not is 0.5.



$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


## PRECISION

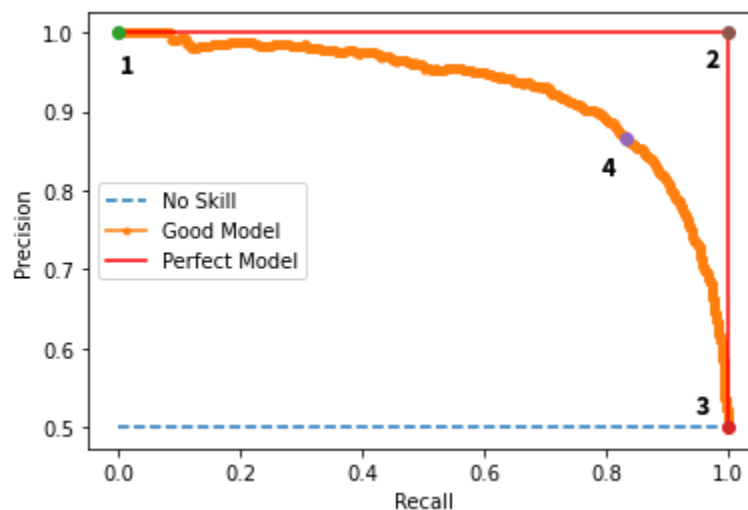
Precision is the likelihood of the model to be correct when it predicts an object. A value of precision is found by dividing the number true positives (cases where the model predicted the bounding box, and the IoU value with the true bounding box is greater than the selected threshold value) by the sum of all positive results. This is finding the percentage of all positive results that were correct.

## RECALL

Recall is the likelihood of the model to not miss an object. Mathematically, this is found by dividing the total of true positive cases by the total number of true positives and false negatives. In other words, this means finding the number of detected objects by the actual number of objects in the images.

## PRECISION-RECALL CURVE

To form a precision-recall curve, the number of true positives (and therefore true negatives and false positives and negatives) are varied by changing the threshold value. For each threshold value, the values of precision and recall are recalculated and plotted onto a graph.



Source: [analyticsindiamag.com](http://analyticsindiamag.com)

Point 1 is where the IoU threshold is 1, meaning only absolutely perfect bounding boxes are said to be true positives. Point 3 is the inverse: where IoU threshold is 0, meaning any box is said to be a true positive. Point 4 lies somewhere in between.

The general shape of the curve is negative for typical models, which is logical. As recall increases, so the number of missed objects is reduced, there are bound to be more incorrect detections. Furthermore, when recall is low, the objects that do get detected are very likely to be correct. This is comparable to asking people to name elements: the first handful will most likely all be correct, but only a few out of the total number have been named. This translates to a low recall but high precision. As the number of named elements increases (recall increases), the odds of people not knowing any more and guessing element names increases, so the number of correct guesses decreases as they give wrong answers. This corresponds

to a drop in precision. Eventually, the group will name all the elements, but as they try to name the last few, they are increasingly more likely to guess incorrectly, and hence this corresponds to the sharp drop in precision as recall increases.

The perfect model always gets the predicted box correct, spot on, hence precision remains 1.0, or in other words perfect. Therefore, the closer to this perfect model curve your model is, the better your model is.

## AVERAGE PRECISION

Average precision is an industry standard for summarising precision-recall curves. Average precision is the area under the precision-recall curve. If you take the area under the perfect curve, you get 1.0, whereas the area of the worst curve (a constant precision of 0.0), you get 0.0. Therefore, average precision ranges between 1 and 0.

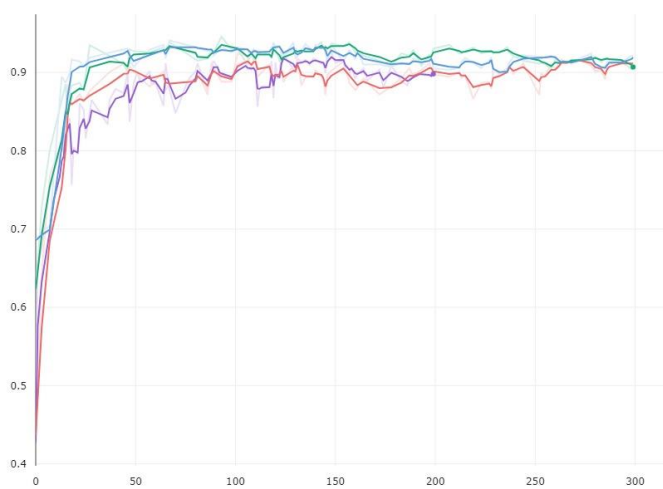
For multi-class object detection models, the mean average precision measure is used, finding the mean of all average precisions for each class. In our case, as we are only looking for drones, we have only one class, so our average precision for the drone class is the same as the mean average precision.

The most common metrics are mAP [0.5] and mAP [0.5,0.95]. The former means the mean average precision at the threshold value of 0.5, meanwhile the latter means the mean average precision between the threshold values of 0.5 and 0.95, with a step of 0.05.

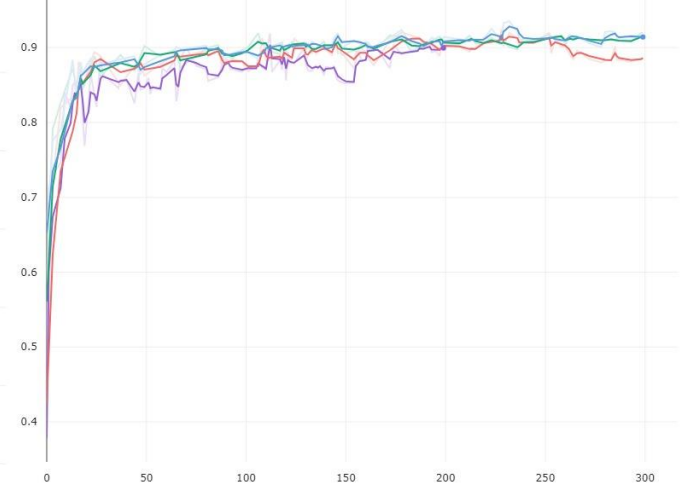
## EDGE TPU

To increase the computational power of our Raspberry Pi and achieve a higher framerate of detections, as that was by far the slowest step in the pipeline, we utilised the Coral Edge TPU. This hugely increased performance, while introducing a new limitation: a maximum model size (when converted to an Edge TPU-compatible format) of 8MB, in order to fit inside the RAM of the TPU. Effectively, this meant we were limited to using only small and nano YOLOv5 models. This meant we had to train new models, and the accuracy and precision of the models were lower.

## MODEL SIZE



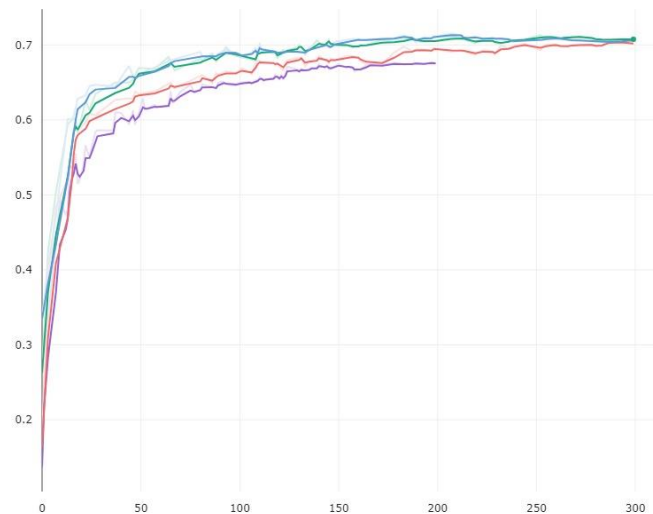
Recall against Epoch number



Precision against Epoch number

In the above charts, purple, orange, green and blue represent the YOLOv5 nano, small, medium and large models respectively. All hyperparameters, other than the epoch number for the nano model, were kept constant in order to fairly compare the models. The graphs have been smoothed by a factor of .5 to make comparison simpler.

As can be seen, the medium and large models finish with very similar values for both recall and precision, with the nano and small models being almost always a bit worse in those statistics. It is for this reason that being limited to small and nano models was a significant limitation.

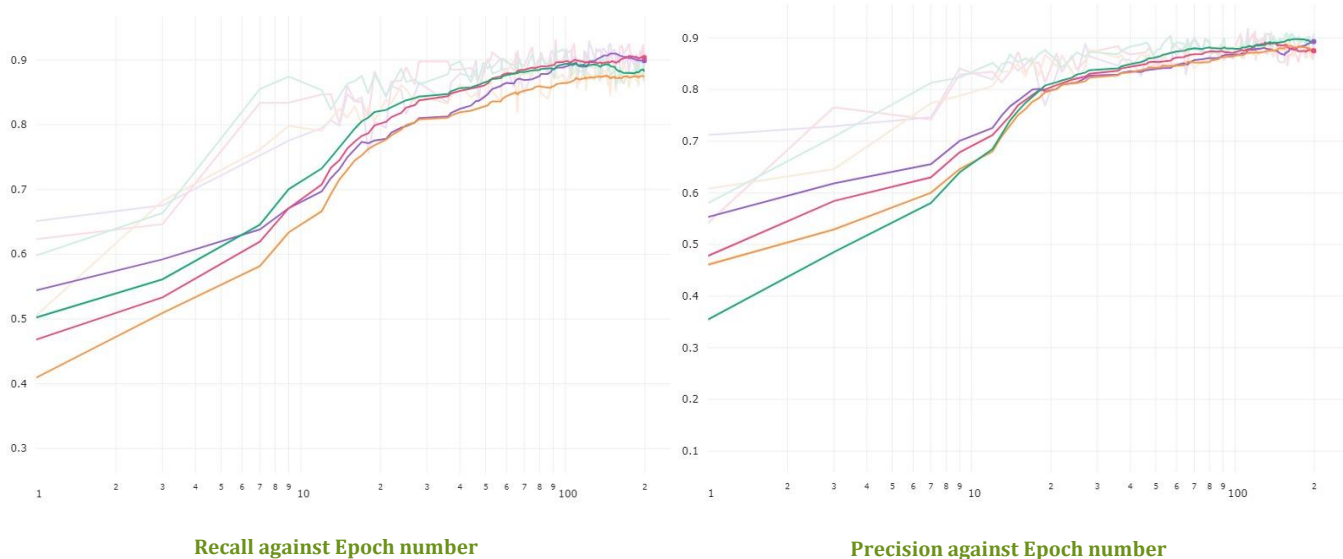


mAP[.5, .95] against Epoch number

As for mAP [.5, .95], the medium and large models are almost the same at each epoch number, with the small model being a level below, and the nano another step below that. Limited to only small and nano models, choosing the small would give us around a .020 better mAP[.5, .95].

## INPUT IMAGE SIZE

In the above charts, orange, green, pink and purple represent 224x224, 416x416, 512x512 and 640x640

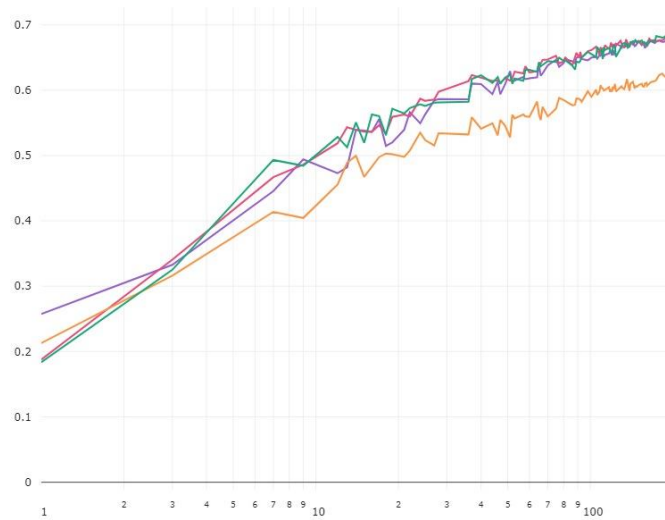


input image size models respectively. All other hyperparameters were kept constant in order to fairly compare the models. All models use a nano YOLOv5 model. The graphs have been smoothed by a factor of .9 to make comparison simpler, as there is a lot of jitter as the epoch number increases.

In the recall graph, 640x640 and 512x512 end up at the same recall value, although the 640x640 model does have a higher peak recall. The 416x416 model is .02 lower at the end, despite its peak being the same as that of the 512x512 model. The 224x224 model is by far the weakest in terms of recall, being constantly the lowest line on the graph.

In the precision graph, all four image size inputs perform very similarly. The 640x640 model is tied best with the 416x416 model, with the 512x512 model having a higher peak than the 224x224 model.

Overall, it is clear that the 640x640 model is superior in both precision and recall, meanwhile the 224x224 input model is inferior in both. The 512x512 and 416x416 models are in-between, with the 512x512 model slightly stronger.



mAP [0.5, 0.95] against Epoch number

For mAP [0.5, 0.95], the input image size has no impact at 200 epochs, other than for the 224x224 model, which performed significantly worse.

Overall, the 640x640 model seems to be the best to use for our application, with recall being the most important, as for an air defence system, we want to ideally be able to detect all target objects.

## FINAL CHOICE OF MODEL

After reviewing both the training metrics and the detection framerate on the Raspberry Pi, we came to the decision that using the nano 640x640 model was best. We trained the model with 200 epochs, as we found that to be the point at which all training plateaued.

Attempts were made to use a small 640x640 model, however that was too large for the RAM of the TPU hence did not compile. The 416x416 model did compile successfully, however, due to the larger model size, had a 2.5 fps framerate, as opposed to the 4.2 fps framerate of the nano 640x640 model. The higher framerate along with the better recall metric meant that we ended up sticking with the 640x640 nano model.

## PROGRAMMING AND IMPLEMENTATION

CodeBlame108 lines (88 loc) · 3.58 KBRawDownloadEdit

```
1 import matplotlib.pyplot as plt
2 import matplotlib
3 import numpy as np
4 import torch
5 import cv2
6 import os
7
8 class drone:
9     def __init__(self, initialCoords: np.array) -> None:
10         self.x = np.array([(initialCoords['xmin'])[0] + initialCoords['xmax'])[0] / 2])
11         self.y = np.array([(initialCoords['ymin'])[0] + initialCoords['ymax'])[0] / 2])
12
13 def linesegments(coords):
14     for i in range(len(coords)):
15         try:
16             #plot the line segments
17             x_values = [coords[i][0], coords[i+1][0]] #
18             y_values = [coords[i][1], coords[i+1][1]]
19         except:
20             print("Error: Reached final point")
21
22         plt.plot(x_values, y_values, 'bo', linestyle="-")
23
24 def ang3Points(coords):
25     for i in range(len(coords)):
26         a = np.asarray(coords[i])
27         try:
28             b = np.asarray(coords[i+1])
29             c = np.asarray(coords[i+2])
30         except:
31             print("Out of range")
32
33         ba = a - b
34         bc = c - b
35
36         #dot product to work out the angle
37         cosine_angle = np.dot(ba, bc) / (np.linalg.norm(ba) * np.linalg.norm(bc))
38         angle = (np.arccos(cosine_angle)) * (180/np.pi)
39
40         print(f"Angle: {angle}")
41
42 def draw(imagepath, coords): #imagepath is currently just the image name : will need to change this a bit if saving in another folder
43     image = cv2.imread(imagepath)
44     try:
45         startpoint, endpoint = (round(coords['xmin'])[0], round(coords['ymin'])[0]), (round(coords['xmax'])[0], round(coords['ymax'])[0])
```



```

42 def draw(imagepath,coords): #imagepath is currently just the image name : will need to change this a bit if saving in another folder
43     path = '/home/george/Documents/Drone angle/images'
44     image = cv2.rectangle(image,startpoint,endpoint,(255,0,0),20)
45     cv2.imwrite(os.path.join(path, imagepath),image)
46 except:
47     print("Error")
48
49 def BBmodel(model):
50     # Inference
51     im = cv2.VideoCapture("VID_20230713_151704.mp4")
52
53     currentframe = 0
54     locationList = []
55     while True:
56         # reading from frame
57         ret,frame = im.read()
58         # print(f"ret output: {ret}, frame output: {frame}")
59
60         if ret:
61             results = model(frame)
62             # Results
63             results.print() # or .show(), .save(), .crop(), .pandas(), etc.
64             try:
65                 for i in range(len(results)):
66                     results.xyxy[i] # im predictions (tensor)
67                     initialCoords = results.pandas().xyxy[i] # im predictions (pandas)
68                     locationList.append(drone(initialCoords))
69             except:
70                 print("Error")
71
72             cv2.imwrite(f"image{currentframe}.png",frame) #creates the image
73             draw(f"image{currentframe}.png",initialCoords) #draws the box
74             currentframe += 1
75         else:
76             break
77
78     print(locationList)
79     return locationList
80
81 if __name__ == "__main__":
82     #model
83     model = torch.hub.load('ultralytics/yolov5', 'custom', '/home/george/Documents/Drone angle/best.pt')
84     matplotlib.use('TkAgg') #change backend after loading model https://github.com/ultralytics/yolov5/issues/2779
85
86     locationList = BBmodel(model)
87
88     coords =[]
89     # plt.rcParams["figure.figsize"] = [500, 500] #defines matplotlib figure size
90     # plt.rcParams["figure.autolayout"] = True
91     for index, i in enumerate(locationList):
92         #plot the points on the graph
93         plt.plot(i.x, i.y, 'r')
94         print(i.x)
95         #zip the x and y coordinates into a tuple
96         for xy in zip(i.x, i.y):
97             coords.append(xy)
98             plt.annotate(f'drone {index}', xy=xy)
99
100     linesegments(coords)
101     ang3Points(coords)
102     plt.show()

```

The purpose of this program is to provide a visualisation of the path of the drone using Matplotlib. We used torch to load the model in this program, however for the final tracking program we will use EdgeTPU to run inference as it is much higher performance for raspberry Pi 4. We take the coordinates initially as a pandas dataframe; searching through the data we can find the minX, minY, maxX, maxY for the bounding box of each frame. In the event that there is no detection, the program will print an error. From the data we can find the coordinates of the centre of the bounding box relative to the camera, which we will pass into a program that controls the rotation of the turret.

We have used a scatter graph to plot the coordinates of the centre of each bounding box. The camera of our turret is only capable of recording 5FPS, so to replicate the limitations of our prototype one point is

plotted every 6 frames. This was done using openCV, in order to read each frame of the video, as well as Matplotlib to visualise the data onto a graph. Finally, this program finds the angle between each three points using dot product. This may be useful when predicting the movements of the drone.

```
import pytest
import unittest
import pandas as pd
import numpy as np
from track import drone
from track import draw_to_files
import os

class TestDrone(unittest.TestCase):
    def test_can_construct_with_initialCoords(self):
        df1 = pd.DataFrame(np.array([[1500, 500, 2000, 900, 0.95, 0, drone]]), columns=['xmin', 'ymin', 'xmax', 'ymax', 'confidence', 'class', 'name'])

        droneTest = drone(df1)
        self.assertEqual(droneTest.x, np.array([1750]))
        self.assertEqual(droneTest.y, np.array([700]))

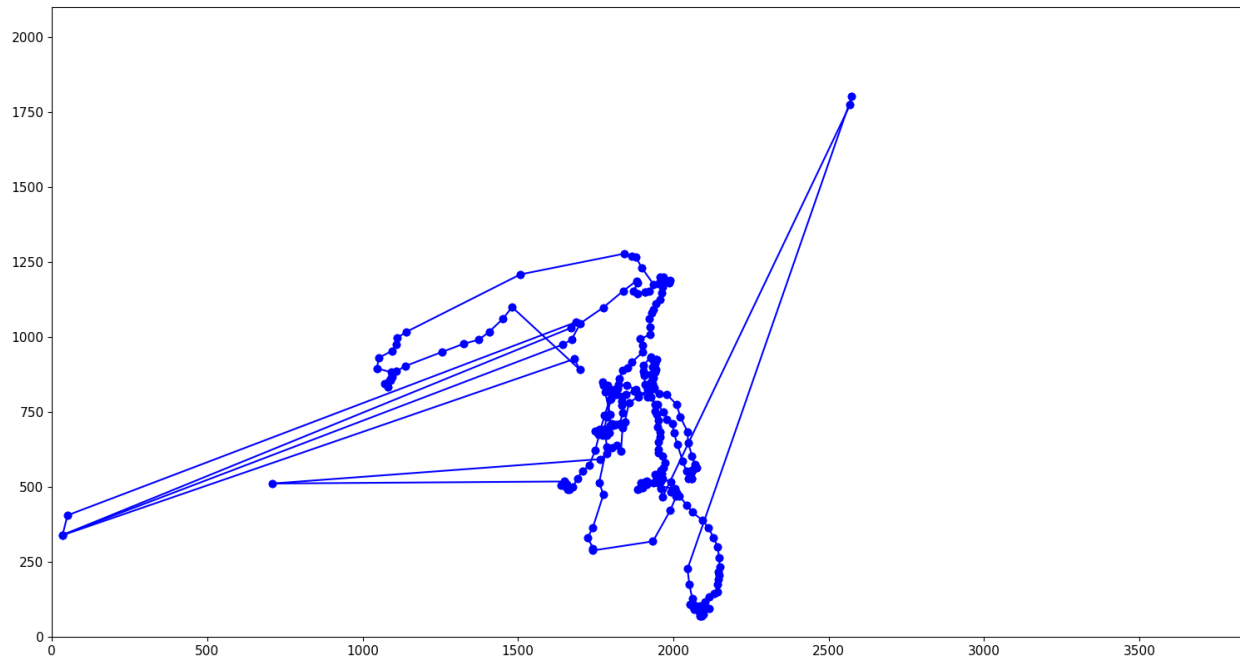
    def test_folder_contains_imageLocationtxt(self):
        df1 = pd.DataFrame(np.array([[1500, 500, 2000, 900, 0.95, 0, drone]]), columns=['xmin', 'ymin', 'xmax', 'ymax', 'confidence', 'class', 'name'])
        currentFrame = 1
        testDraw = draw_to_files(df1, currentFrame, 0)

        testDraw.set_path()

        assert os.path.isfile(f"{testDraw.path}/imageLocation.txt") == True
```

Many of our team went on work placements over the Year 12 summer holidays. Here are a few unit test cases used to test individual sections of the program. In the case that somebody pushes changes that break parts of the program they may not have initially seen, the unit tests will catch the error. In true Test Driven Development, you would write the unit test cases before developing the code.





Locations on coordinate plane.

Images of our prototype firing mechanism.



## Tracking/Prediction:

A critical consideration when choosing a tracking algorithm lies in balancing computational complexity and efficiency. We evaluated non-preemptive tracking and compared the results to predictive tracking, both using the same equipment to reduce bias.

## Manual Controls

First, we evaluated the use of manual controls. Controlling the prototype manually meant that we did not have to perform inference on any data, meaning we were not limited in computational power. Instead, we introduced human error, as using manual controls meant a human operator was required at all times.

The main benefits of manual controls are:

- 1) Continuous tracking that does not rely on camera and computational vision, which can result in a smoother, more consistent tracking.
- 2) Simpler to implement.
- 3) A human has “intuition” and high levels of tracking and prediction, so could be very effective when acclimated to the control schemes.
- 4) Humans can identify objects with greater accuracy, which helps to eliminate stray/incorrect identifications that might have otherwise occurred using the AI.

The main downsides of manual controls are:

- 1) Due to lack of optical alignment in the field besides the laser point, it could be hard to track the fast moving object in the air.
- 2) The human element in the system means that performance differs from operator to operator, which makes manual controls an inconsistent form of control.
- 3) Automation is a non-element as a human is required. This increases the amount of resources that are required to operate the device.

However, this can be used as a baseline comparison, or control, for the rest of the tracking methods.

The below code is very rudimentary and has a basic flaw in the fact that it cannot move the device diagonally, since movement is determined using an “if, elif” statement which means that the turret can only move either horizontally or vertically.

```

Code Blame 50 lines (49 loc) · 1.07 KB
1  import pygame
2  import threading
3  import RPi.GPIO as GPIO
4  from RpiMotorLib import RpiMotorLib
5  pygame.init()
6  direction = [22, 17]
7  step = [23, 18]
8  en = [24, 27]
9  motors = [RpiMotorLib.A4988Nema(direction[i], step[i], (21,21,21), "DRV8825") for i in range(2)]
10 GPIO.setup(en[0],GPIO.OUT)
11 GPIO.setup(en[1],GPIO.OUT)
12 GPIO.output(en[0],GPIO.LOW)
13 GPIO.output(en[1],GPIO.LOW)
14 js = pygame.joystick.Joystick(0)
15 js.init()
16 clock = pygame.time.Clock()
17 LEFT, RIGHT, UP, DOWN = False, False, False, False
18 def rotate(direction, axis):# True=Clockwise, False=Counter-Clockwise, axis=0 for y, 1 for x
19     threading.Thread(target=motors[axis].motor_go, args=(direction, "Full" , 1, .0005, False, 0.05), daemon=True).start()
20 while True:
21     clock.tick(60)
22     for event in pygame.event.get():
23         if event.type == pygame.JOYAXISMOTION:
24             if event.axis == 0:
25                 if event.value > 0.5:
26                     LEFT, RIGHT = False, True
27                 elif event.value < -0.5:
28                     LEFT, RIGHT = True, False
29             else:
30                 LEFT, RIGHT = False, False
31         elif event.axis == 1:
32             if event.value > 0.5:
33                 UP, DOWN = False, True
34             elif event.value < -0.5:
35                 UP, DOWN = True, False
36             else:
37                 UP, DOWN = False, False
38         elif event.type == pygame.QUIT or event.type == pygame.JOYBUTTONDOWN:
39             pygame.quit()
40             GPIO.cleanup()
41             exit()
42     if UP:
43         rotate(False, 0)
44     elif DOWN:
45         rotate(True, 0)
46     if LEFT:
47         rotate(False, 1)
48     elif RIGHT:
49         rotate(True, 1)
50

```

## Tracking

Tracking was much easier to implement than prediction as there is no algorithm required. The model would simply find coordinates of the centre of the drone and rotate accordingly. In this case, the prototype is constantly adjusting to move the centre of the camera, to the centre of the drone. Without the prediction the tracking is crude; the laser often falls behind the actual position of the drone, especially since the camera can only read 5 frames per second. Because this was much easier to code, we were able to release much earlier, which is ideal under time constraints.

## Prediction

To improve prediction we first decided to investigate and provide an estimate for the complexity of implementing a predictive algorithm. We evaluated how much new content we were required to learn, as well as feasibility. Some examples of algorithms we looked at are Kalman Filter and Optical Flow, however we had trouble using these in our model. We attempted to use a Kalman Filter Python package, documentation available at: [Kalman Filter](#). This proved to be a challenge to implement, and the results we produced were noticeably poorer than other prediction methods we used.

## Line Prediction

Initially, we thought to use the bearing, velocity and initial position of the drone to predict the next location. In this example, we take the 3 known data points preceding the point that we predict. We use the number of frames between each point to find the velocity of the drone. By following the trajectory of

the drone at the last known point, we can calculate the distance the drone will travel to predict the next location of the drone.

```
import matplotlib.pyplot as plt
import numpy as np
import pickle
WRITE = False
class drone:
    """
    Class to define drones.\n
    Properties:
    - Coordinates
    - Time of appearance\n
    Class functions:
    - Coords Getter
    - Coords Setter
    """
    def __init__(self, x: float, y: float, time: float) -> None:
        self._coords = np.array([x, y])
        self.time = time

    @property
    def coords(self) -> np.ndarray:
        return self._coords
    @coords.setter
    def coords(self, value) -> None:
        try:
            type(value) == tuple
            self._coords = np.array(value)
        except:
            raise TypeError("Incorrect type: Must be Tuple")

class Prediction:
    """
    Class for running the actual prediction. Requires files of x, y coordinates of drones and the time at which they appear.
    """
    def __init__(self) -> None:
        self.xlist = list(filter(lambda x: x>0, self.read_list("xlist")))
        self.ylist = list(filter(lambda x: x>0, self.read_list("ylist")))
        self.tlist = self.read_list("tlist")

    @staticmethod
    def read_list(name) -> list:
```

```

with open(f'{name}', 'rb') as fp:
    n_list = pickle.load(fp)
    return n_list

def getLocationList(self) -> list:
    ...
    Returns a list of drone objects based on the x,y coordinates and timestamps loaded on class initialisation.
    ...
    return [drone(self.xlist[i],self.ylist[i],self.tlist[i]) for i in range(len(self.xlist))]

def predict(self, drone1: object, drone2, drone3, reqtime: float) -> np.ndarray:
    ...
    Outputs the prediction in this order:
    - x coordinate
    - y coordinate
    - time of reaching that coordinate
    ...

    DIST = 50
    ttime = drone3.time-drone1.time

    distD1_D2 = drone2.coords - drone1.coords
    timeD1_D2 = drone2.time - drone1.time

    distD2_D3 = drone3.coords - drone2.coords
    timeD2_D3 = drone3.time - drone2.time

    #x          y          x
    if distD1_D2[0]**2 + distD1_D2[1]**2 <= DIST**2 and distD2_D3[0]**2 + distD2_D3[1]**2 <= DIST**2:

        vf = distD1_D2*timeD1_D2/ttime+distD2_D3*timeD2_D3/ttime

        final = list(vf*reqtime + drone3.coords)

        # final[0] returns x coordinate, final[1] returns y
        return final[0],final[1],drone3.time+reqtime
    else:
        return None,None,None

```



```

def getPredictions(self, locationList: list) -> list:
    ...
    Runs the prediction algorithm on the entire dataset given to the class on initialisation.
    ...

    predX, predY, predT = [], [], []
    for i in range(len(locationList)-3):
        x,y,time = self.predict(locationList[i],locationList[i+1],locationList[i+2],1)

        if x!= None:
            predX.append(x)
            predY.append(y)
            predT.append(time)
    return predX, predY, predT

def plotGraph(self, predX: list, predY, predT) -> None:
    ...
    Makes a graph plot of the predictions (coordinates)\n
    First half is actual points of data, second is prediction data.
    ...

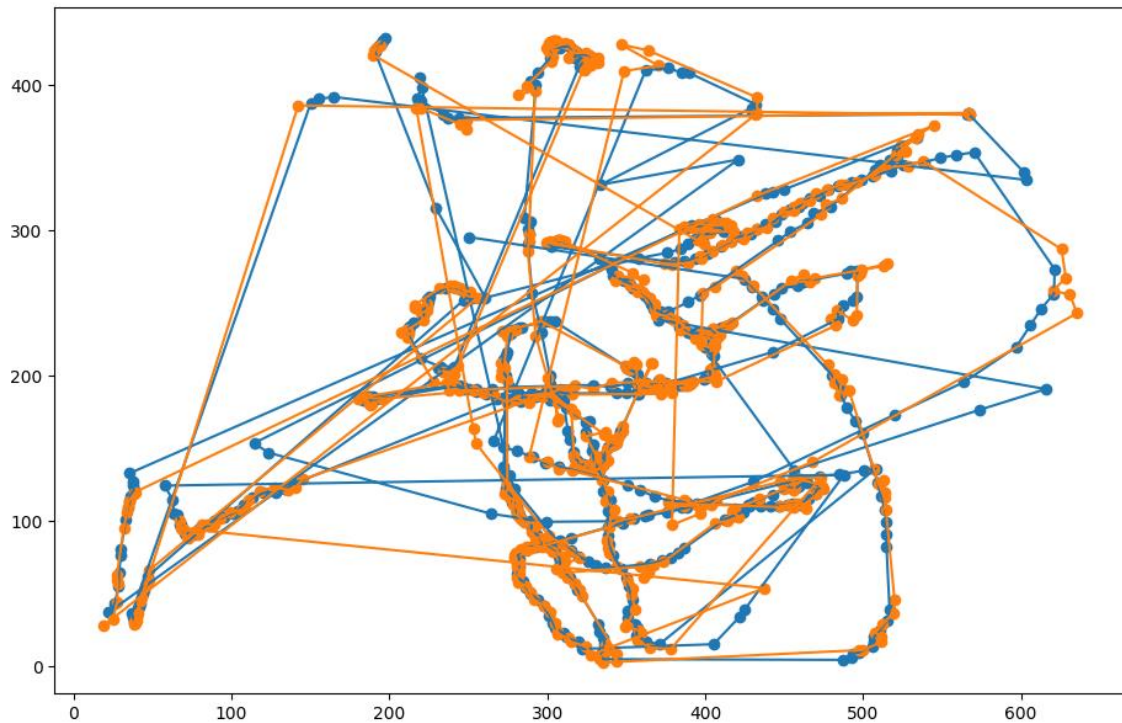
    plt.plot()
    plt.scatter(self.xlist, self.ylist)
    plt.plot(self.xlist, self.ylist)
    for x,y,t in zip(self.xlist,self.ylist,self.tlist):
        label = f"drone {t}"
        # plt.annotate(label,(x,y),textcoords="offset points",xytext=(0,10),ha='center')

    #prediction points
    plt.scatter(predX,predY)
    plt.plot(predX,predY)
    for x1,y1,t1 in zip(predX,predY,predT):
        label = f"prediction {t1}"
        # plt.annotate(label,(x1,y1),textcoords="offset points",xytext=(0,10),ha='center')

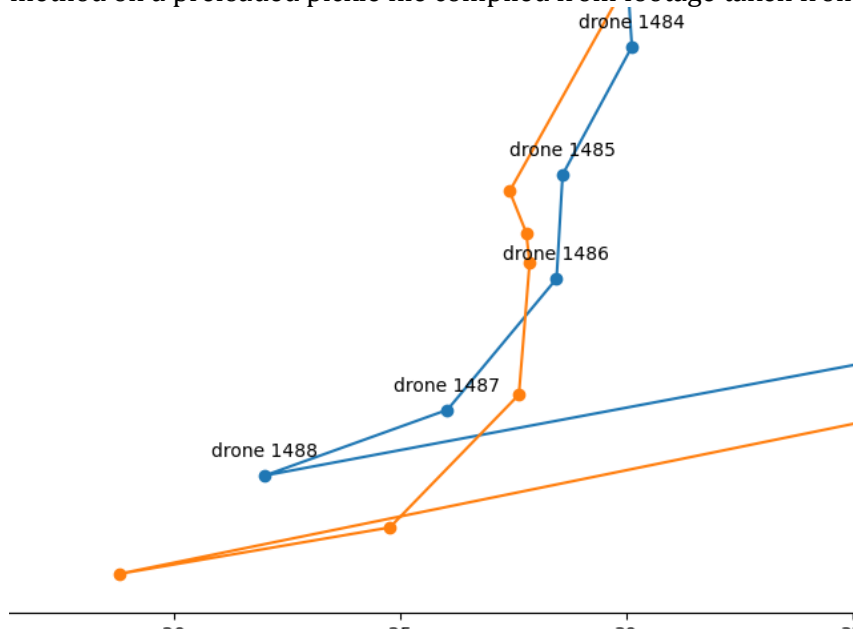
# def write_list(a_list,name):
#     with open(f'{name}', 'wb') as fp:
#         pickle.dump(a_list, fp)
#         print('Done writing list into a binary file')

if __name__ == "__main__":
    predictor = Prediction()
    locationList = predictor.getLocationList()
    # x = predictor.predict(locationList[150], locationList[151], locationList[152], 1)
    predX, predY, predT = predictor.getPredictions(locationList)
    predictor.plotGraph(predX, predY, predT)
    plt.show()

```



Testing this program, we found that the prediction works well for drones that do not move erratically such that the trajectory at the last known point does not deviate drastically. In the example above, the orange points indicate the prediction, the blue points are real coordinates. We tested this prediction method on a preloaded pickle file compiled from footage taken from the camera on our prototyApe.



Notice that the sudden change in trajectory is not efficiently tracked. Furthermore, if the drone is not detected for many consecutive frames, the laser will fail to move off the trajectory of the last known

point. Furthermore, for instances where the drone is not moving in a straight line (circular movement), we found that it does not effectively track the drone. Hence, we attempted to incorporate angles into our prediction.

```

if (abs(ang1-ang2) < 15) and (abs(ang2-ang3) < 15):
    print(f"ang1 {ang1}")
    ang = ((180-ang1) + (180-ang2) + (180-ang3)) / 3
    if drone1.coords[0] > drone2.coords[0] and drone2.coords[0] > drone3.coords[0]:
        ang = math.pi/180 * (360 - ang)
    else:
        ang = math.pi/180*(ang)

    cartesianx = (final[0]*math.cos(ang)-final[1]*math.sin(ang))
    cartesiany = (final[0]*math.sin(ang)+final[1]*math.cos(ang))
    print(f"cartesianx {cartesianx} cartesiany {cartesiany}")
    # print(final[0], final[1])

    final[0] = cartesianx
    final[1] = cartesiany
    # radius = math.sqrt(final[0]**2 + final[1]**2)
    # print(radius)
    # length = math.sqrt(radius**2 + radius**2 - 2*(radius)*(radius)*(math.cos(math.pi/180 * (180-ang1))))
    # print(length)

    # final[0] returns x coordinate, final[1] returns y
    return final[0],final[1],drone5.time+reqtime
else:
    return None,None,None

def calculateAngle(self, D1, D2, D3):
    ba = np.ndarray.flatten(D1-D2)
    bc = np.ndarray.flatten(D3-D2)

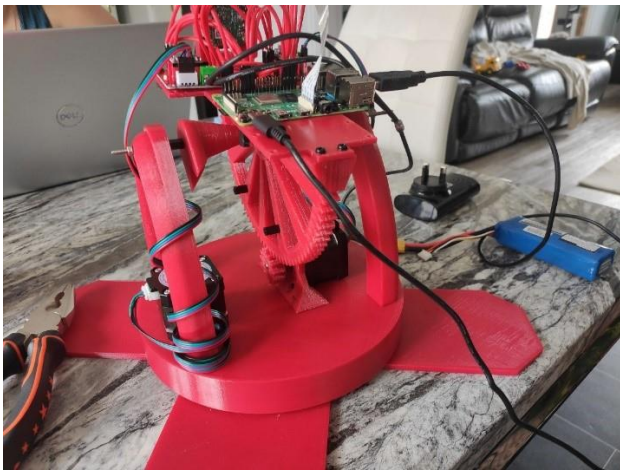
    #dot product to work out the angle
    cosine_angle = np.dot(ba,bc) / (np.linalg.norm(ba) * np.linalg.norm(bc))
    angle = float((np.arccos(cosine_angle)) * (180/np.pi))
    # print(f"Angle is {angle}")
    return angle

```

Here we have used dot product to work out the angle between 5 points preceding the point we are trying to predict. If the angle between each point is increasing or decreasing with each point leading to the prediction, we use cartesian rotation to factor the angle of trajectory into our final prediction. With this method, we managed to more confidently get predictions for curved motion. Concerning predicting without known coordinates, the only solution we had was to improve the accuracy of our model. This is only possible with more time, as we would have to provide the dataset with more labelled images.

## DISCUSSION

Our results shed light on the responsibility of AI in defence systems and the potential for further investigation into larger-scale implementation. In this study we created a model turret using a raspberry pi, EdgeTPU and a laser as a demonstration of feasibility in real-life application. The results suggest that a more accurate model is required for confident detections, as well as more powerful machinery. While we weren't necessarily able to realise our plans to the extent of our initial ideas and specifications, we have gained a strong insight into what uses this technology would have in the future of defence systems throughout this project. We overall feel that our shortcomings were due to our scaled-down model lacking the functionality of the full-scale product. If applied in a professional setting, the progress we have made could be extremely useful in this field. Nonetheless, the areas in which we feel we could have been more successful, as well as ways in which we feel were successful are listed in some detail below.



The design was modelled in Fusion 360, making sure that the mechanisms would work, and that there was a sufficient field of vision. Then, the design was 3D printed, with each part being printed individually and assembled. The motors were then installed, along with the Raspberry Pi, and wired up to properly interface with the program.

## IMPLICATIONS AND LIMITATIONS

There are a few things that we as a team have learnt. First and foremost, I think this project has given us a far better understanding of the creation process, and how prototyping both software and hardware can be a tough balance.

Furthermore, our expectations for a streamlined workflow were disrupted as regular errors and roadblocks were realised to be the norm, and not the exception. Some of the roadblocks we faced were:

- Lengthy internal school examination period that took place during our time doing the project.
- Numerous amounts of both logical errors and inefficiencies in our software that resulted due to a fragmented approach to our coding collectively.
- An unfortunate but necessary abandonment of our initial idea (electromagnetic coil system) after realising we had time constraints that needed to be met.
- Hardware failures due to the inconsistency with the medium we used for prototyping (3d printing) that led to wasted resources and time.
- A need to switch to EdgeTPU for our AI inference since PyTorch (the standard amongst hobbyists, and therefore the most developed and has great ease-of-use) was far too computationally heavy for a Raspberry Pi, which led to a significant increase in production time.

This means that we were unable to provide a realistic representation of what the military might have used as a prototype, despite our model being functionally capable. Perhaps we will see this technology thrive with more funding. We believe this technology could be further developed to create air defence systems which will keep citizens secure from hostile threat; considering the defence budget, it is entirely possible that this technology could be implemented for drones and missiles.

## REFERENCES

- UK Parliament. (2023) *UK defence expenditure*. Available at: <https://commonslibrary.parliament.uk/research-briefings/cbp-8175/#:~:text=The%20Spring%20Budget%202023%20allocated,over%20this%20five%2Dyear%20period.>
- Wikipedia. *Surface-to-air missile*. Available at: [https://en.wikipedia.org/wiki/Surface-to-air\\_missile](https://en.wikipedia.org/wiki/Surface-to-air_missile)
- The Guardian. (2018) *The Guardian view on drones: effective regulation needed*. Available at: <https://www.theguardian.com/commentisfree/2018/dec/20/the-guardian-view-on-drones-effective-regulation-needed>
- Adobe. (2022) *Beginner's guide to Agile Project Management*. Available at: <https://business.adobe.com/blog/basics/agile>
- Laoyan, S., asana. (2022) *What is Agile methodology?* Available at: <https://asana.com/resources/agile-methodology>
- Shah, S. (2021) *Cost Function is no rocket science!* Available at: <https://www.analyticsvidhya.com/blog/2021/02/cost-function-is-no-rocket-science/>
- Mack, C. (2017) *Machine Learning fundamentals (I) Cost functions and gradient descent*. Available at: <https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220>
- Koech, K, E. (2020) *Cross Entropy Loss function*. Available at: <https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e>
- Liusie, A. (2022) Youtube. Available at: <https://www.youtube.com/watch?v=PwgpI9mKars>
- Datacamp. (2022) *Yolo Object Detection Explained*. Available at: <https://www.datacamp.com/blog/yolo-object-detection-explained>
- V7Labs. (2023) *YOLO: Algorithm for Object Detection Explained*. Available at: <https://www.v7labs.com/blog/yolo-object-detection>
- Wu, W., Liu, H., Li, L., Long, Y., Wang, X., Wang, Z., Li, J. and Chang, Y. (2021). *Application of local fully Convolutional Neural Network combined with YOLO v5 algorithm in small target detection of remote sensing image*. Available at: <https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0259283>
- 3Blue1Brown. (2017) *What is backpropagation really doing?*. Available at: <https://www.youtube.com/watch?v=llg3gGewQ5U>
- Wikipedia. *Backpropagation*. Available at: <https://en.wikipedia.org/wiki/Backpropagation>
- Towards Data Science. (2017) *Activation Function in Neural Networks*. Available at: <https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6>
- Page 54 | Gold CREST Award

O'Shea, K., Nash, R. (2015) *An introduction to convolutional neural networks*. Available at:  
<https://arxiv.org/abs/1511.08458>

IBM *Convolutional neural networks*. Available at <https://www.ibm.com/topics/convolutional-neural-networks>

Towards Data Science. (2021) *Object Detection Explained: R-CNN*. Available at:  
<https://towardsdatascience.com/object-detection-explained-r-cnn-a6c813937a76>

To read for RCNN and Faster RCNN:  
[https://openaccess.thecvf.com/content\\_ECCV\\_2018/html/Bowen\\_Cheng\\_Revisiting\\_RCNN\\_On\\_ECCV\\_2018\\_paper.html](https://openaccess.thecvf.com/content_ECCV_2018/html/Bowen_Cheng_Revisiting_RCNN_On_ECCV_2018_paper.html)

Medium. *Region proposal network rpn backbone of faster r-cnn*. Available at:  
<https://medium.com/egen/region-proposal-network-rpn-backbone-of-faster-r-cnn-4a744a38d7f9>

Patel, M. (2020, updated 2023) *Towards AI. YOLO V5- Explained and Demystified*. Available at:  
<https://towardsai.net/p/computer-vision/yolo-v5%E2%80%8A-%E2%80%8Aexplained-and-demystified>