EVALUATION OF DRONE DETECTION TECHNOLOGIES

Sashil Anand[1], Ma Zhiyu[2], Jerry S/O Tamilchelvamani[3]
[1]Raffles Institution, 1 Raffles Institution Ln, Singapore 575954
[2]Methodist Girls School, 11 Blackmore Dr, Singapore 599986
[3]Defence Science and Technology Agency, 1 Depot Road, Singapore 109679

Abstract
While the popularity and widespread availability of the drone benefits photography and pilot enthusiasts, the errant use of drones poses risks of espionage, safety, and even criminal activities. It is thus pertinent for authorities to be alerted to cases of unauthorised flying and have an effective means of detecting drones. This paper will evaluate the effectiveness of a convolutional neural network (CNN) model for the detection of drones in varied conditions and how resilient the technique is against false positives due to birds and planes. The CNN model that we will be focusing on in this paper is the pytorch-based YOLOv5 model.

Introduction

Drones pose risks of espionage when flown over restricted areas such as military bases and government buildings, safety risks and disruption to air traffic when flown near airports and last but not least criminal activities when flown near borders and in vicinity of prisons. The airspace in these sensitive areas requires close monitoring and immediate action to provide protection against errant drones. Thus an effective means of drone detection is a valued capability for authorities.

Today the most effective means to detect drones are by far radar systems as they have the longest detection range with reasonable accuracies. But they too can be prone to mistakes and mis-detect birds and other similar cross-sections objects as drones instead. In addition, to take immediate action, operators need visual information of the drone, to decide upon the most appropriate course of action. Radars cannot provide visual information and thus a camera system would instead look in the direction designated by the radar and a CNN-based detector will scan and provide the visual information to the operators.

There are many CNN-based detectors, however the YOLO (You Only Look Once) algorithm, stands out due to its high inference throughput. The model only needs one-pass to detect the presence of drones and it is very suited for real-time applications. However, the model does face certain challenges in accurately detecting drones due to distractor objects such as birds and planes. The distance of the object to the camera system and the background clutter are key factors that determine the resultant accuracy in a YOLO model.

For this project, we will use the fifth edition of the YOLO model, YOLOv5 from the internet herein referred to as the base model and a transfer learnt variant of the YOLOv5 model by DSTA herein referred to as the DSTA model. DSTA did transfer learning by feeding thousands of images of DJI mavic and phantom models onto the base model. In this paper, we will evaluate the performance of the base model and the DSTA model on the false positives due to birds and planes across varying distances and environments. We will also attempt to train our own variation of the YOLOv5 model for detection of drones.

<u>Part I: Evaluation of Drone Detection Models</u>

Methodology

Our methodology was as follows. We first curated an image dataset by sourcing for drones, birds and planes images from the internet, artificially generating them using a simulator and physically going down to specific locations in Singapore to capture the required footage. We then annotated each image to specify the location of the drone in the image where it contained a drone and nil where it did not. The details of our annotation process is described in Part 3. This was followed by categorising the image dataset into 9 bins sorting them according to the distance the subjects, the drones, birds and planes were from the camera and type of the background i.e. skies, trees or urban clutter that the image constituted of. We then ran the base and DSTA model against all 9 bins to arrive at scores to denote the effectiveness of the models. The details of our scoring process is described in Part 2. Lastly we analysed and provided our evaluation on how well the models dealt against birds and planes.

We went to significant locations such as MBS, Sentosa and Gardens by the Bay to gather images of birds and drones. In order to arrive at a varied dataset, we captured images at sunrise as well as sunset and positioned ourselves such that the subjects were captured at varied distances and backgrounds. A varied dataset would prevent any instances of underfitting and overfitting. Underfitting refers to the model not having enough information to effectively discern objects while overfitting refers to biases in the model.

The images gathered were of high resolution 1080p and the inference was to be done on a RTX 5000 GPU laptop. We thus used the "x6" configuration of the base and DSTA model for the evaluation.

<u>Figure 1.1</u>: Mavic model

<u>Figure 1.2</u>: Phantom model

This section describes how the dataset was categorised and sorted into 9 bins. We had 2 variables as mentioned, i) the distance the subjects were from the camera and ii) type of the background the image constituted of. There were three values for each variable and permuting, resulting in 9 bins. The three values for the distance variable were a) <10m, b) 10-40m and c) >40m. The three values for the background variable were a) sky, b) trees and c) urban. We ensured that each bin contained approximately 60% of only drones, 20% of birds and drones, 10% of only birds. Figure 1.1 and 1.2 shows the type of drones that were in the bins.

Some of the bins did not have enough images of the "birds and drones" and "only birds". This was because capturing birds flying in real life was opportunistic. As such, we did not have enough data for some bins. To resolve this, we synthetically imposed silhouettes of birds into urban and tree backgrounds. We took pictures of forested areas and buildings around Singapore and artificially added birds in. This was done on the software Canva. We also removed images that were too similar to each other in all the bins. This is to achieve as much variation as possible in terms of camera angle, background etc, as well as to prevent overfitting. After this step, we achieved an average of about 30 images in each bin, with the bin with the lowest number of images had 20 images, while the bin with the highest number of images had 50 images.

Another challenge was estimating the distance the subjects were from the camera. To help us sort the images into the 3 distance categories, we estimated the pixel size of the object in each of the images as we took these photos ourselves. This was also one of the challenges we faced as we could not determine whether the drones in some of the images were far away enough to be 40m and above away from the camera. To resolve this challenge, we made reference to a ratio method.

1) $size\ of\ drone\ in\ photo = \frac{width\ of\ drone}{distance\ of\ drone\ from\ camera} \times magnification\ of\ photo$
2) $pixel\ size\ of\ drone\ =\ pixel\ spacing\ of\ phone\ /\ size\ of\ drone\ in\ photo$

Part 2: Calculating statistical data

Scoring

Precision refers to the number of times the model was correct while recall refers to the number of times the model missed a drone. Here are the Precision and Recall's mathematical definitions:

$$Precision = \frac{TP}{TP + FP} \qquad TP = \text{True positive}$$

$$TN = \text{True negative}$$

$$Recall = \frac{TP}{TP + FN} \qquad FP = \text{False positive}$$

$$FN = \text{False negative}$$

$$\text{mAP score} = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Figure 2.1: Mathematical Definitions

We made use of the Precision and Recall to compute an mAP score. The mAP stands for mean average precision, and it ranges between 0 to 1. A score closer to 1 indicates a more accurate model.

Hypothesis

We had the following hypothesis:

1) As the distance of the drone from the camera increases, the mAP score will decrease exponentially.
2) Drones located in a clear sky background will also be the most accurately detected compared to that of tree and urban backgrounds.
3) The DSTA model will have a significantly higher mAP score for all 9 bins compared to that of the base model as the DSTA model was trained to detect mavic/phantom drone models.

Results & Data

		Sky background			Urban background			Tree background		
		<10m	10-40m	>40m	<10m	10-40m	>40m	<10m	10-40m	>40m
Drone mAP score	Base model	0.872	0.780	0.713	0.673	0.473	0.393	0.745	0.547	0.485
	DSTA model	0.945	0.792	0.717	0.710	0.489	0.401	0.783	0.587	0.493

Table 2.1

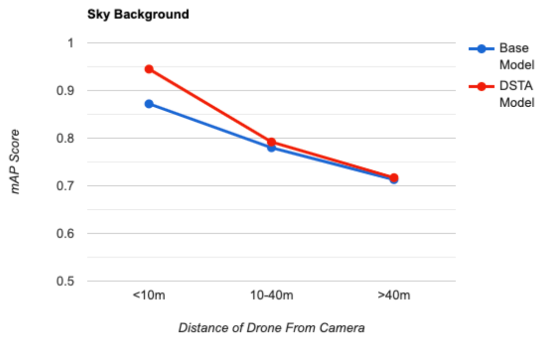

Figure 2.2

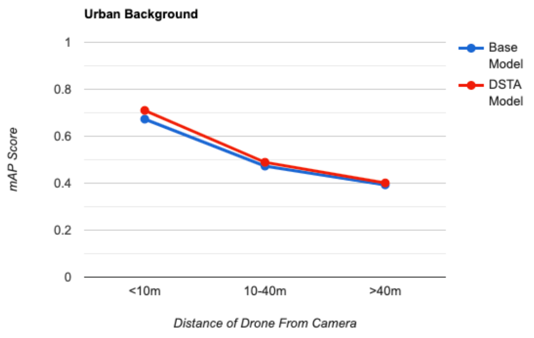

Figure 2.3

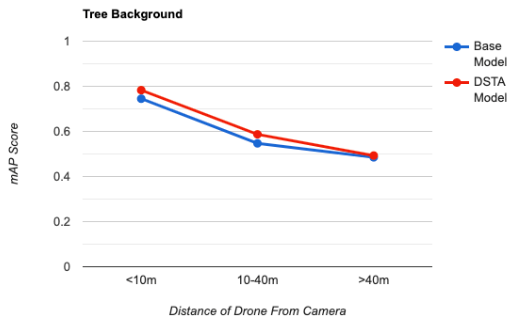

Figure 2.4

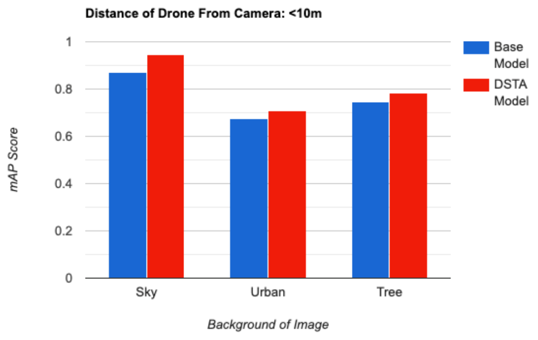

Figure 2.5

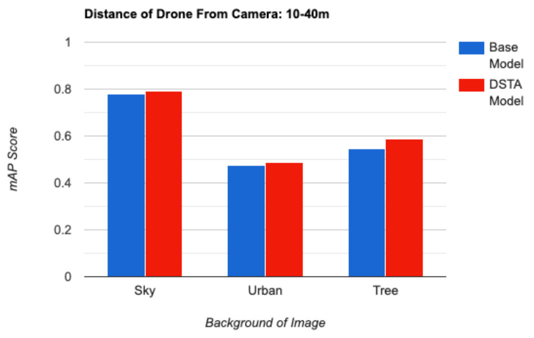

Figure 2.6

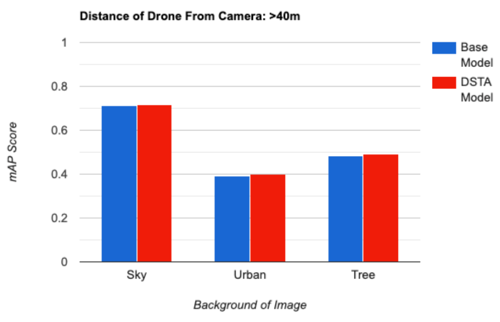

Figure 2.7

Discussion

Hypothesis 1

Firstly, we realised that as the distance of the drone from the camera was increased, the mAP score of the drone decreased on a rather linear scale for both models (see figure 2.2, 2.3, 2.4 above). For example, for the sky background (see figure 2.2) , the base model recorded a 0.159 mAP score difference, which was an 18.2% drop, whereas the DSTA model had a bigger mAP score drop of 24.1%.

This matched our hypothesis, which we suspect was because the further the drone was in the picture, the smaller the pixel size of the drone. Thus, it was harder for both models to distinguish a drone from a black dot in the image, which led to more false negatives being generated. Moreover, since the drone was further away from the camera, the drone was also mistaken more easily for distractor objects, such as birds, which also led to more false positives being generated.

We also realised that as the distance of the drone from the camera was increased, there was a greater mAP score drop for tree and urban backgrounds compared to sky backgrounds. The DSTA model recorded a mAP score drop of 43.5% and 37.0% in urban and tree backgrounds respectively, which was significantly higher than the mAP score drop of 24.1% in the sky background. The base model recorded a mAP score drop of 41.6% and 34.9% in urban and tree backgrounds respectively, which was also higher than that of 18.2% drop in the sky background. We think the reason for this trend was because when drones are captured further away from the camera, it was even harder for the model to distinguish features of the drones, which may blend in with the trees or buildings in the background. Moreover, we realised that the DSTA model recorded a higher mAP score drop in all 3 backgrounds compared to the base model. This could possibly be due to overfitting in the <10m proximity range, which rendered the DSTA model unable to detect drones from further ranges as efficiently.

Hypothesis 2

Secondly, the mAP scores for sky backgrounds was comparably higher than the mAP score for urban and tree backgrounds for both models (see figure 2.5, 2.6, 2.7 above). When the drone was placed against a tree or urban background, the presence of trees and buildings created a silhouette, causing the drone, which was also black, to blend in with the trees and buildings respectively. This caused the drone to not be easily detected as the features of the drone, such as the blades of the drone, were unanimous with the background colour. Thus, even though the drone could be near the camera (such as 0-10m), the mAP score was still lower than that of a sky background.

Hypothesis 3

Thirdly, the DSTA model had an overall better performance compared to the base model. Across all 9 categories, the DSTA model had a higher mAP score than that of the base model. This was because the DSTA model was better in detecting mavic/phantom drone models as compared to the base model. As such, it was easier for the DSTA model to detect distinct features of the mavic/phantom drones even in rough backgrounds where distractor objects such as trees and buildings were present. Here are some side-to-side comparisons:

However, we observed that the mAP score for the DSTA model was comparably better than the base model for the <10m range for all 3 ranges. For the 10-40m and >40m categories, the DSTA model only did slightly better than that of the base model. We suspect this was because the DSTA model was only trained for the <10m ranges and for detecting drones in images at close range. Moreover, since the DSTA model was mainly trained to distinguish between mavic and phantom drones, it was thus unable to detect drones at further distances from the camera as effectively.

Presence of false negatives

A problem we faced was the mAP scores for certain bins were below 0.50, which means the model has a drone detection rate of less than 50%. We realised that this trend occurred the most in the urban background bins – 10-40m and >40m, as well as the >40m tree background bins for both models. When we looked at the images returned after training, we realised that there were numerous false negatives in both models (see figure 2.8 & 2.9 below). The bins of the 2 lowest mAP scores both belonged to the 40m and above urban background. This was possibly due to the underfitting in this particular bin as we struggled to capture sufficient images in real life of drones >40m away from the camera. As such, we had to synthetically impose all the images in this bin, which formed the smallest bin of only 20 images.

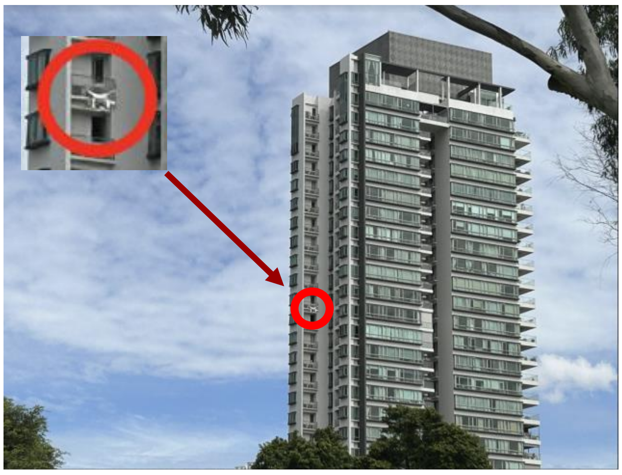
Figure 2.8: DSTA Model
No Drone Detected

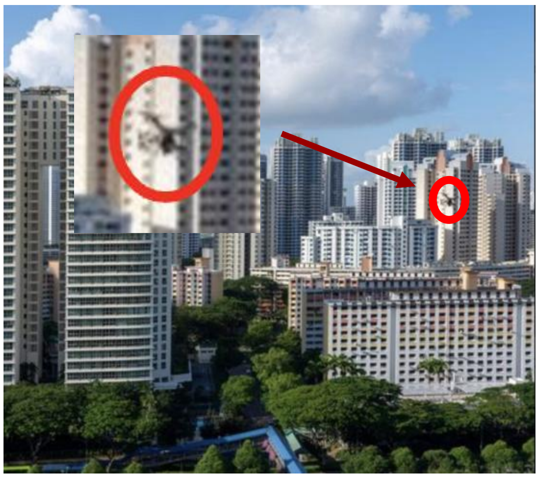
Figure 2.9: Base Model
No Drone Detected

Presence of false positives

A challenge we faced was also the distractor objects present in our dataset. As our dataset is 20% birds and drones and 10% birds, many false positives were generated, especially when the birds and drones were placed near each other, which may have confused both models respectively, causing false classification of birds as drones. When looking at the images after being run through the models, we realised that an average of 30% of detected drones were actually birds. The bin with the highest number of false positives was the 10-40m urban background bin, with an average of 40% of detected drones actually being birds. We suspect this was due to the fact that the drone was well camouflaged in the buildings, whereas the birds were flying in the sky, causing them to stand out compared to the drones.

Part 3: Labelling of Drone Detection Model

In our attempt to improve the DSTA model, we trained our labelled dataset using the weight of the DSTA YOLOv5x6 model as well as the weight of the transfer learnt model. We also included the distractor objects such as birds, kites and planes in most of these images so that the model is trained to not detect such objects

Methodology
After understanding the performance of the model on each type of bin, the final step was to improve the model. To do this, we uploaded all our datasets we collected from Part 1 of the project onto roboflow which is a software that helps to annotate images for computer vision models. As we understood that a larger dataset would provide a better trained model, our image dataset was a total of 1400 images. Our dataset included 500 images from all 9 bins from Part 1 of this project, as well as images of drones, birds and planes that were found online, which made the rest of the 900 images in the dataset.

After uploading the datasets, we then proceeded to manually apply bounding boxes to images, with 70% assigned for training the YOLO model, 20% assigned for validation and 10% assigned for testing the model. The 9 bins which make up about 150 images were part of the test set. The test set were images which have never been seen before by the model which was where the accuracy of the model was tested. To make the labelling of the images efficient and accurate, we had 3 different labels, namely drones, birds and planes. This was so that when we applied the bounding boxes, the model was able to recognise the different features in each label.

Some challenges that arose when we were labelling the data was we realised that there was not enough variation of images for some labels. For example, most images on the internet that showed planes flying in the sky were taken from high up above the clouds where the plane was flying. As such, we could not find many images online where planes flying in the sky were taken from a bottom-up angle and taking real time pictures would be opportunistic and time consuming. To resolve this, we tried to synthetically impose planes flying on clear sky images to get various angles of the plane. Moreover, another challenge was the sheer amount of data we had to manually label. Since our image dataset was 1400 images, it was rather tedious for us to manually apply bounding boxes for every single label in all the images.

When generating different versions using our image datasets, we made use of image augmentation (see figure 1.3 & 1.4 below), which increased the diversity of the images included in the database. Image augmentation included changing different settings of the images, such as blur, brightness, exposure, grayscale – changing the image to be black & white, as well as noise. The purpose of this was to mimic the disruptions a drone in real life would face. Adjusting such settings of the images in the dataset assisted with overcoming photometric distortion while random flipping, scaling, cropping, and rotating were used to overcome geometric distortions. This step was implemented to improve the robustness of the object detection model. This resulted in an increase in the variability of images so that an unknown environment will not create any issues for the detector model, such as false positives, false negatives etc.

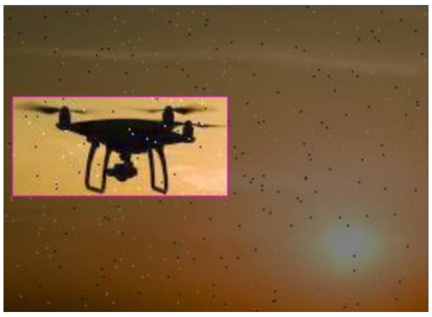
Figure 3.1: Augmented image with noise and blur

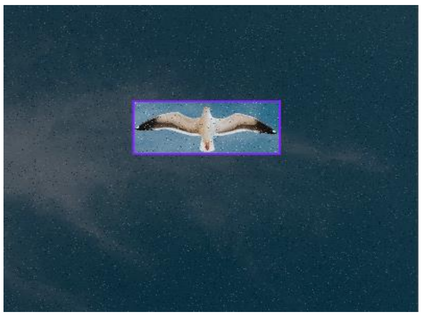
Figure 3.2: Augmented image with noise and exposure

Part 4: Custom Training of Data

We then moved on to do some of our own transfer learning on the DSTA YOLOv5x6 model. In order to train our dataset, we did some programming on Google Colab and made use of a Python script. First of all, in order to be able to train our dataset, we had to import certain softwares such as PyTorch, OS, Cuda as well as install dependencies.

```
#clone YOLOv5 and
!git clone https://github.com/ultralytics/yolov5  # clone repo
%cd yolov5
%pip install -qr requirements.txt # install dependencies
%pip install -q roboflow

import torch
import os
from IPython.display import Image, clear_output  # to display images

print(f"Setup      complete.      Using      torch      {torch.__version__}
({torch.cuda.get_device_properties(0).name    if    torch.cuda.is_available()
else 'CPU'})")
```

Next, we had to import our Roboflow project over to Google Colab. In order to do this, we exported the Roboflow project in a zip file, as seen from the code below.

```
from roboflow import Roboflow
rf = Roboflow(api_key="kzkXoHRzrjMvnIMsXtWf")
project = rf.workspace("yolo-bv4k3").project("drone-detection-rmyxm")
dataset = project.version(34).download("yolov5")
```

Then, we moved onto training the datasets obtained from roboflow by determining the image size, batch size and number of epochs. This was one of the more time consuming aspects of the project as a greater number of epochs(Eg. 200) would result in a greater amount of time(1.5 Hours) being spent. This was the code we used for training the dataset:

```
!python   train.py   --img   416   --batch   16   --epochs   200   --data
{dataset.location}/data.yaml --weights yolov5s.pt --cache
```

A challenge we faced while trying to train our dataset was that whenever we tried to increase the number of epochs to over 300, the training would not be executed as we would get an error that ‘Cuda out of memory’. We suspect this was because the number of epochs was too large and the computers we were using did not have sufficient RAM to be able to train a dataset this large over 300 times. In order to solve this problem, we tried to run our entire script in our computers Command app. When we tried this, it was even more time-consuming and took 20 minutes to train 1 epoch. As such, due to the limitation in resources, we decided to just keep the number of epochs to less than 300.

After training the data, we used tensorflow to visualise the progress of the training using graphs. In our best trial, the model averaged at an mAP score of 0.910. This was an improvement compared to the results we got from evaluating the base model and the DSTA model. The training of the model of over 200 epochs can be seen in the graph below:

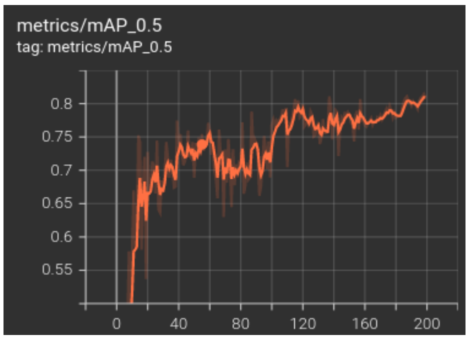

Figure 4.1: Tensorflow graph visualisation of training for 200 epochs

Finally, we viewed the model’s predictions on the test pictures which have never been seen by the model before. This would help us get a visual understanding of the reliability of the model and what improvements have to be made. We then compared these results with the results we obtained from the original yolov5 models, the transfer learnt model and reattempted the process till we made improvements.

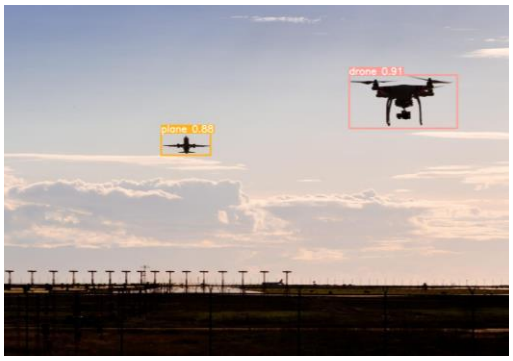

Figure 4.2

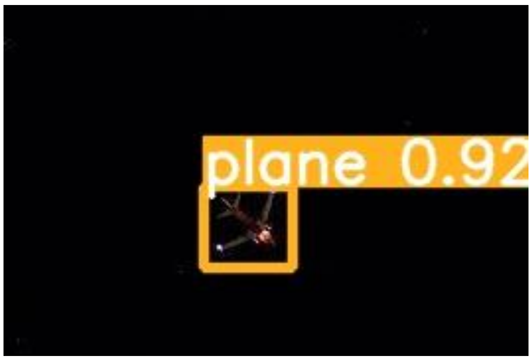

Figure 4.3

We realised that both the models, namely the DSTA model and the model that we have trained, performed similarly and managed to even detect drones that were at a range of 40-50 metres away from the camera. Some examples can be seen above.

Conclusion

Overall, there were several learning points in our research at DSTA. Firstly, we learnt about the importance of curating data for machine learning projects. The quality of the data is as important as the quantity of the data which was why we spent greater time focusing on the gathering of data. The data classified into the bins will have a severe impact on the results as well. We also learnt about the impacts of underfitting and overfitting in a real life scenario which required us to do multiple rounds of testing. Decreasing underfitting and overfitting would allow the model to perform more optimally and increase its reliability as well. As for future work, we could continue improving the model we had trained by gathering more relevant data and figuring out the effect specific augmentations have on the results.

Acknowledgements
We would like to thank Casper, Zhong Han, Sarah, Nicholas and Daryl for helping us in the project and our mentor Jerry for his guidance. They have been instrumental in our research and guided us throughout our internship period.

References

[1] Taha, B., & Shoufan, A. (2019). Machine learning-based drone detection and classification: State-of-the-art in research. *IEEE Access*, *7*, 138669–138682. https://doi.org/10.1109/access.2