# TOAD DETECTION WITH DRONES AND MACHINE LEARNING

Ching Lee
CID:01351497
Supervisor: Dr.John Hassard
Assessor:
Word Count:3400

## Declaration of Work

The workload is evenly distributed between me and my partner. I have never done a related project in a summer placement or with my supervisor prior to this project.

## Table of Contents

***Abstract***

In this project, an existing Faster-RCNN model is trained with aerial images taken with drones. We have developed a model which identify toad from different background with 88.8% precision and 80% recall.

***Introduction***:

Dates to 1935, cane toads are introduced to Australia in hope to control the population of cane beetle. The current population of cane toads is estimated to be more than 200 million dues to the fact that they are highly adaptable and have no natural predators. Cane Toads are sweeping across Australia at 50 km per year and they are invading these areas and destroying the biodiversity there. These cane toads are mainly located in the inaccessible part of northern Australia and it will be very costly to locate these toads with human force. Therefore, we aim to develop a simplified version of a machine learning algorithm which can be deployed with drones to locate these toads and help control their population. In this project, we focus on developing a machine learning algorithm with optical images.

***Theory***

***Comparison between previous RCNN-model***

RCNN stands for region based convolutional neural network. We will compare Faster-RCNN with other alternatives to justify why Faster-RCNN is best suited for this project.

*Fast-RCNN and RCNN*

To save computing time, a selective search algorithm is used to generate 2000 region proposal instead of feeding the whole picture into the CNN. The selective search [1] model is written in the following way

1. Generate initial sub-section
2. Use greedy algorithm to combine similar regions
3. Produce region proposals using the combined regions

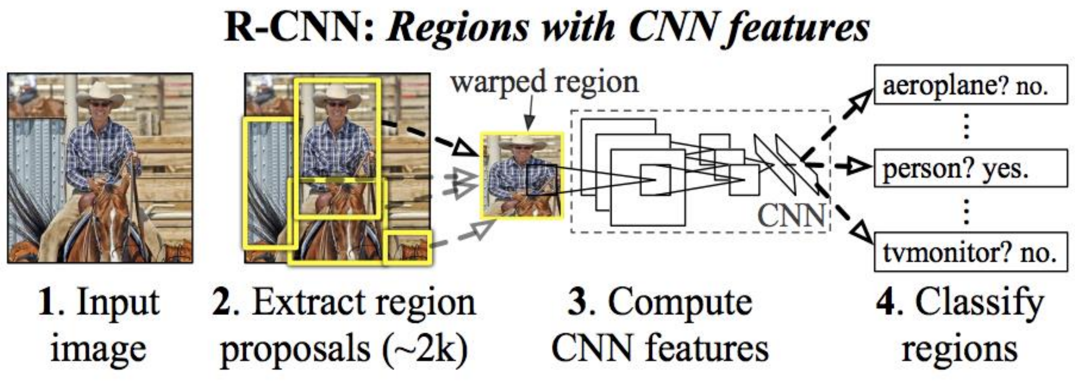


*Figure 1: Schematic of RCNN (Source: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e)*

The CNN extracts feature from the images and feed it into a support vector machine (SVP) which classifies the presence of objects in the proposed regions. Despite a selective search algorithm is used to reduce running time, analyzing 2000 regions is still too time consuming to be implemented real time and it requires long training time. Apart from that, selective search algorithm is fixed, and this will lead to bad region proposals which affects the accuracy of the model.

Fast-RCNN [1] is an improved version of RCNN and the model feeds the entire image into the CNN which produce a feature map and region proposals are chosen from this map and selective search is used to combine similar regions. A RoI pooling layer is used to reshape the combined images into fixed shape and pass it onto a fully connected layer. This produces a RoI feature vector and a SoftMax layer [2] is used to classify the proposed region. Fast-RCNN is faster than RCNN as it only feeds one image to the CNN instead of 2000 region proposal into the CNN. Despite being significantly faster than RCNN, selective search algorithm is slowing down the fast-RCNN and it is still not quick enough for real time detection.

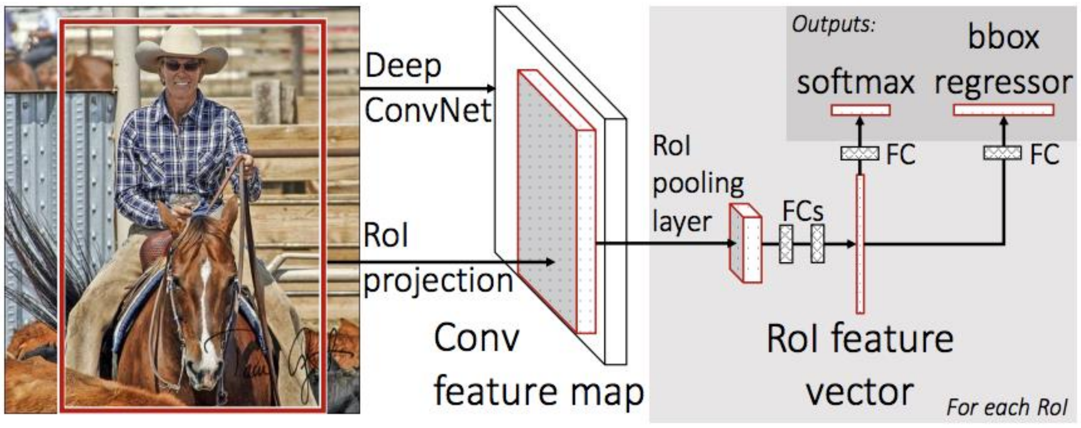


*Figure 2: Schematic of Fast-RCNN (Source: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e)*

<u>*Faster-RCNN*</u>

Like Fast-RCNN [1], the image is fed to a convolution layer to produce a feature map. However, a region proposed network (RPN) [1] make proposed regions and this RPN learns region proposal so its proposal improve over time and this makes it a lot more accurate and efficient than the selective search algorithm. The proposed regions are then being processed by ROI pooling and predictions are made.

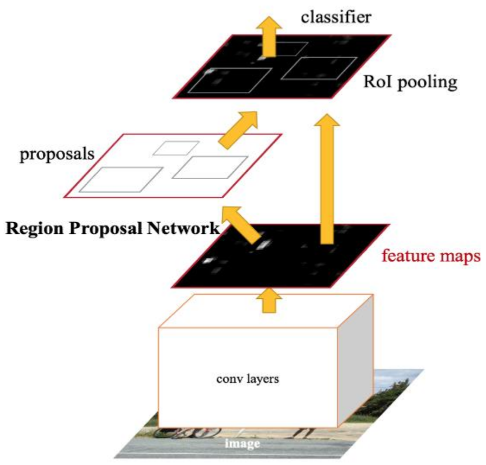


*Figure 3: Schematic of Faster-RCNN (Source: [https://medium.com/@hyponymous/paper-summary-faster-r-cnn-f8ea90ef1ff4](https://medium.com/@hyponymous/paper-summary-faster-r-cnn-f8ea90ef1ff4))*

This improves the efficiency significantly and the model is suitable for real time detection. Accuracy is also improved as the algorithm learn from region proposal and it makes better proposal during the learning progress.

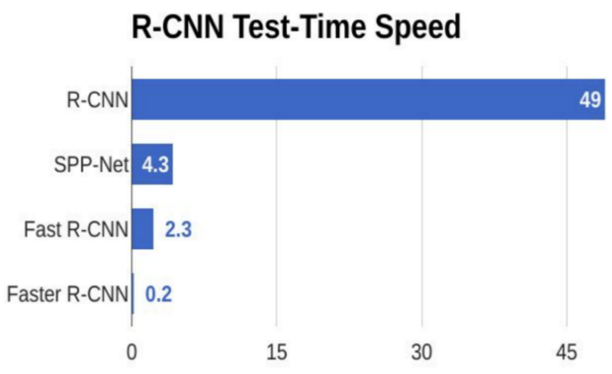


*Figure 4: Comparison between RCNN Speed (Source : https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e)*

This justifies our choice of using faster-RCNN as we intend to detect toads in real time and faster RCNN is the only available model for instant detection and it produces more accurate predictions.

***Convolutional Neural Network (CNN)***

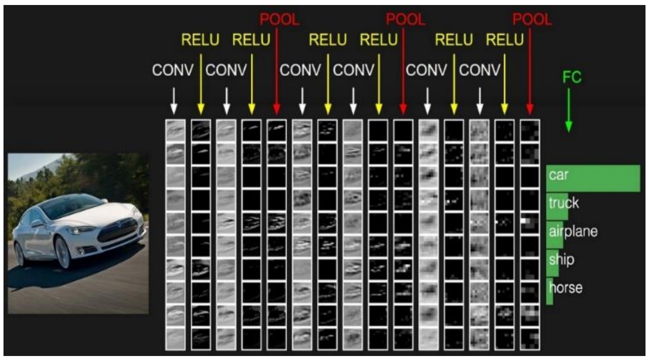


*Figure 5: An example of CNN (Source: https://cs231n.github.io/convolutional-networks/#fc)*

A common CNN contains three main components which include convolutional layer [3], pooling layer [3] and fully connected (FC) layer [3]. Feeding images into a CNN produces a final class score.

*Convolutional Layer*

The role of this layer is to reduce image size without losing features that affects prediction. Firstly, images will be separated in 3 color channels red, green and blue. Each color channel will be represented by a matrix. To perform convolution, a filter is chosen, and the convolved feature matrix can be calculated by sliding the filter across the image matrix.

The convolved feature matrix can be calculated using this formula and an example of this calculation is shown below.

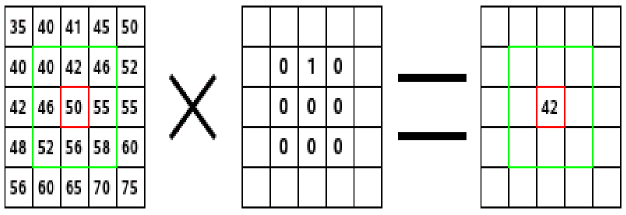


*Figure 6: An example of matrix convolution*

The result is that the input with depth 3, color channel, length and width 32, image dimension will be converted to the blue conv layer with neurons.

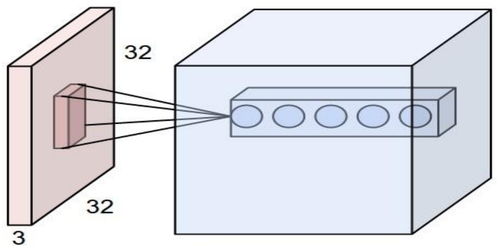


*Figure 7: Images of 3 colours are converted into a volume (Source: https://cs231n.github.io/convolutional-networks/#fc)*

The output volume is controlled by three hyperparameter and they are depth, stride and zero-padding [3]. Firstly, Depth corresponds to the number of kernels used and these kernels are all looking for different features in the images. Secondly, stride is the number of pixels we move the kernels across the input volume. If the stride is high, then the output volume will be low vice versa. Thirdly, padding the input volume boundaries with zeros enable to control the output volume conveniently.

*Pooling Layer*

It is very common to place pooling layers [3] between Convolutional layers. Its role is to reduce the computing power required to process the images and extract important features from the images. There are two main types of pooling which are average pooling and max pooling [3]. Max pooling suppresses noisy activations while average pooling only just reduces the dimension of the input. Thus, max pooling performs a lot better than average pooling.

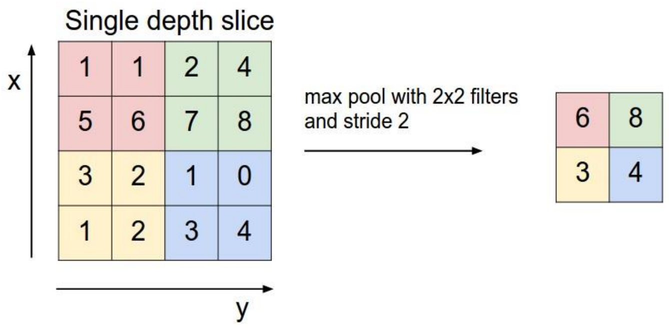


*Figure 8: An Example of Max Pooling (Source: https://cs231n.github.io/convolutional-networks/#fc)*

Convolutional layer and pooling layer form a layer of convolutional neural network (CNN) [3]. The number of layers of CNN to process an image will increase with the complexity of the image in order to capture low level features.

*ReLU layer*

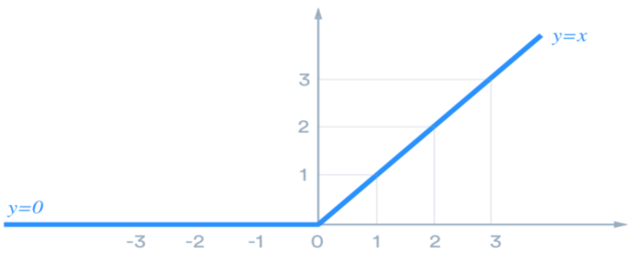


*Figure 9: ReLu Function ( Source: https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7)*

ReLU [4] stands for rectified linear unit and it is an activation function. Having a ReLU layer makes the network sparse. A sparse network usually has better accuracy and less noise. Placing a ReLU layer enable neurons to process meaningful aspects of the problem. For example, a model detecting cats and dogs and there are neurons that could detect other objects and these neurons will not be activated. This increases efficiency of the network and less computing time is required.

*Fully Connected Layer*

A Fully Connected layer [5] learns dominant features from non-linear combinations of the output of the CNNs.

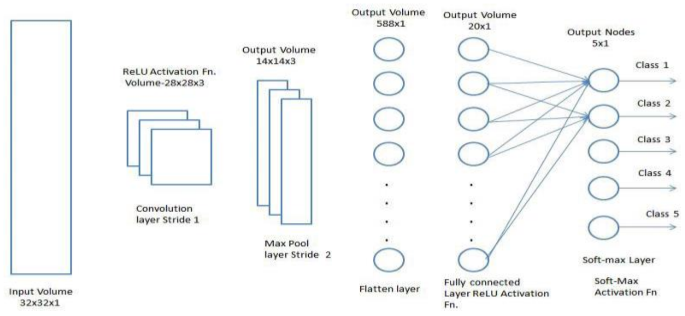


*Figure 10: Fully Connected Layer at the back of the CNN (Source : https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53)*

The output of CNN is flattened and feed into the fully connected later where a SoftMax [2] classification technique is used to distinguish between dominant and low-level features in images. SoftMax function returns probability from input number. Firstly, the function raise exponential to the power of each input number and divide it by the sum of the exponential as shown below. The function increase the weighting of dominant features and it produces a confidence level for classification.

***Loss Function***

Loss function [6] is a method to evaluate the machine learning algorithm performance on modelling a given dataset. If the loss function shows a large value, it means the predictions made by the algorithm is deviating too much from the actual results. There are two main types of loss function and there are regression loss and classification loss. The one we used in the project is the Multi class SVM loss.

$$SVM\ Loss = \sum_{j \neq y_i} \max\,(0, s_j - s_{y_i} + 1)$$

In simple terms, the score of correct class should be greater than the sum of the score of those incorrect classes by some safety margin which is normally set as one. Consider the following example.

| | Image #1 | Image #2 | Image #3 |
|---|---|---|---|
| **Dog** | -0.39 | -4.61 | 1.03 |
| **Cat** | 1.49 | 3.28 | -2.37 |
| **Horse** | 4.21 | 1.46 | -2.27 |

*Figure 11: Example of SVM Loss Calculation (Source: https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23)*

The SVM loss for these three images is 8.48, 0, 5.2 respectively. The SVM loss is high when a wrong decision is made by the algorithm while the SVM loss is zero when a correct decision is being made.

***Precision & Recall***

Precision [7] and recall [7] are important metrics to help us evaluate how accurate the model predictions are. Precision measure the percentage of true positive to all positive. Recall measures the percentage of true positives to the sum of true positives and false negatives.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

***Intersection Over Union (IOU)***

IOU [7] measures the overlap between the prediction boundary and the ground truth boundary. In some models, classification will be made if IOU exceed a threshold value where we select this value to be 0.6

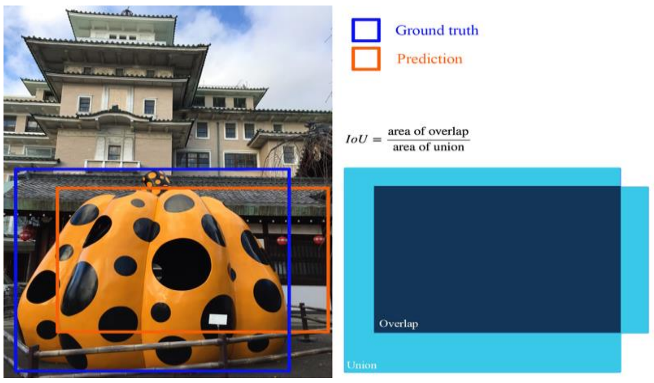


*Figure 12: An Example of IOU Calculation ( Source: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173)*

***Method***

***Fluffy toy simulation***

Cane toads are toxic and very lethal which make it impossible for us to release it and collect aerial data. To simulate cane toad to our best abilities, we found a fluffy toy which is shaped like a cane toad, and we developed a machine learning algorithm trying to find this fluffy toy in different background.

***Data Collection***

To train a working object classifier, we need a lot of images for the algorithm to learn from. As we cannot find any aerial images of cane toads online. We decide to collect our own data using a camera on the drone DJI Phantom 3. This drone is very steady and its stability in the air enable us to collect clear and quality images. To minimize the number of images required to produce a working classifier, we decide to collect images with grass in the background as shown in Figure 13 so the algorithm could learn more features about the targets with less data than in a complex background.

During our trial runs, we found that classifier might not be working when the drone is moving. To address this problem, we try to collect data at different angle and at different height so that the algorithm could learn about features of the target at different orientation. In the first classifier we made, we only train it with images where there is only one object. Our first classifier performs poorly when we test it with a video with human and other objects in the background as it keeps identifying incorrect objects as the target. To improve the algorithm, we collect data with multiple objects in the background as shown in Figure 14 so that the algorithm could learn about the differences between the target and other objects such that it could make better predictions.

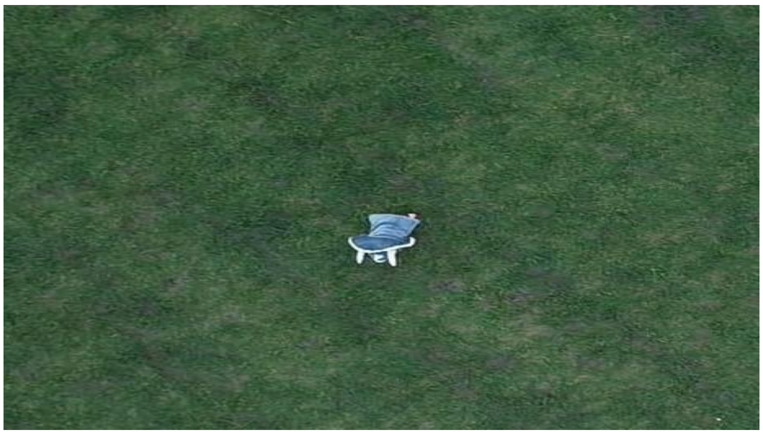


*Figure 13: An Aerial image with just the target*

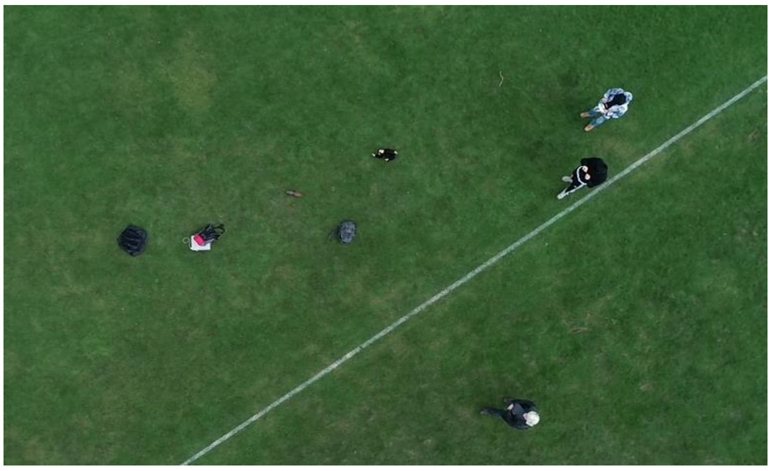


*Figure 14: An aerial image with multiple objects in the background*

***Labelling Images***

The images we collected are in .jpg file and we use a software called labellmg [8] where we could draw boxes on images and assign those boxes to different class. We used rectangular boxes to highlight regions of interest as shown in Figure 15. These labelled images will be turned into .xml file and we used some open source python script. These labelled boxes will be transformed into coordinates in a .csv file where it contains all the boxes coordinate of all images as shown in Figure 16. We feed this .xml file into training such that IOU can be calculated to track the algorithm performance.

The algorithm will assign predictive boxes to objects in the images and IOU will be calculated for these objects and the algorithm is looking to keep learning from the data to maximize IOU. To improve the classifier performance in messy background, we have added some extra labels such as human, backpack, monkey, giraffe, bear to make the algorithm to learn about the differences between target and background objects such that it will not predict any background object as a toad.

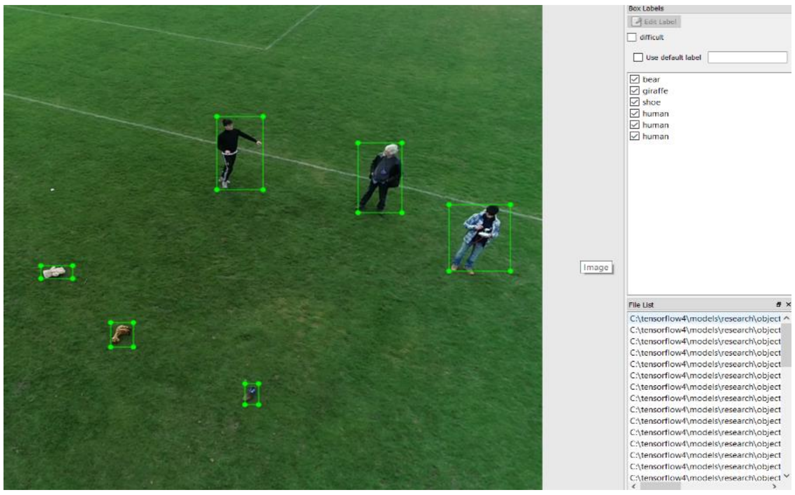
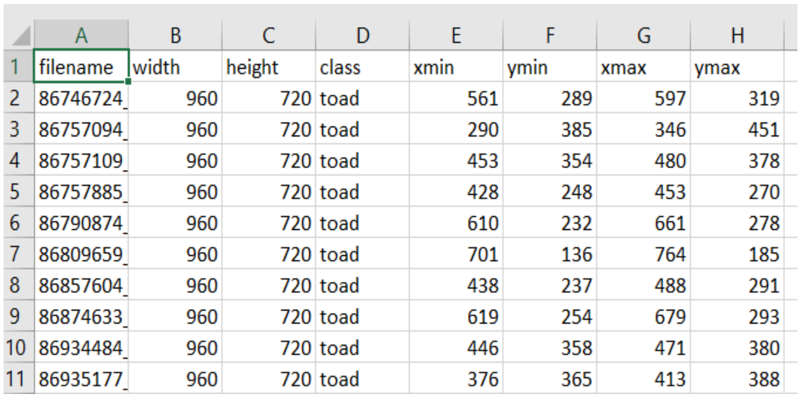


Figure 15: Labelling Images with LabelImg



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | filename | width | height | class | xmin | ymin | xmax | ymax |
| 2 | 86746724_ | 960 | 720 | toad | 561 | 289 | 597 | 319 |
| 3 | 86757094_ | 960 | 720 | toad | 290 | 385 | 346 | 451 |
| 4 | 86757109_ | 960 | 720 | toad | 453 | 354 | 480 | 378 |
| 5 | 86757885_ | 960 | 720 | toad | 428 | 248 | 453 | 270 |
| 6 | 86790874_ | 960 | 720 | toad | 610 | 232 | 661 | 278 |
| 7 | 86809659_ | 960 | 720 | toad | 701 | 136 | 764 | 185 |
| 8 | 86857604_ | 960 | 720 | toad | 438 | 237 | 488 | 291 |
| 9 | 86874633_ | 960 | 720 | toad | 619 | 254 | 679 | 293 |
| 10 | 86934484_ | 960 | 720 | toad | 446 | 358 | 471 | 380 |
| 11 | 86935177_ | 960 | 720 | toad | 376 | 365 | 413 | 388 |

Figure 16: Coordinates of Labelled Boxes in a .xml file

***Model Training***

We train the algorithm using TensorFlow-GPU [8] packages. To produce a better a classifier, we use a high-performance laptop with a high-quality graphics card to shorten the required time for training and improve the quality of training. Considering we are training the classifier with around 200 images, we set the training rate to be 0.004 so that the classifier performance is not undermined, and the training is not lengthy.

To train a classifier we need to assign images to test or train pile, so that it will learn from the train pile primarily and test it against the test pile to evaluate its performance. This process keeps happening until we terminate the training process. A recommended 80/20 split between train and test pile is used and images in both piles are chosen randomly so that a pile will not contain similar images in a higher proportion than the other pile. This reduces bias such that the classifier is trained to make better prediction at certain angle and height.

Training is terminated when the total loss of the machine learning algorithm reaches below 0.1 steadily as shown in Figure 17 where the loss function is used to evaluate how well the classifier is learning from the images.

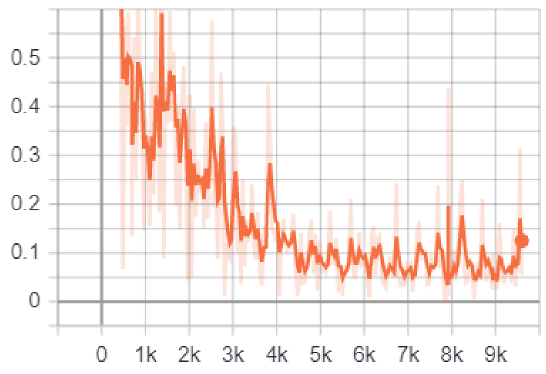


*Figure 17:Total Loss Function*

As we expected, training time increases with the number of images while the learning rate is constant. The training time for the final version of classifier we produced is around 7 hours.

### *Model Evaluation*

To evaluate the model, we have taken two one-minute long videos with the drone. One of the videos contain only the target in a different background than those training data. It is used to evaluate how the model perform with different background. Another video is taken with grass and other objects in the background so that we could evaluate how accurately the model perform with multiple objects in the background. We used an open source python script [8] which enables the classifier to work while we play the video and we could see how well the classifier performed. To evaluate the model quantitively, we used a confusion matrix [9] where we could calculate metrics like precision and recall etc.

## ***Results***

### *Video Result*

After the training is completed, we test the model's performance on video using an open source python script [8]. The first model we made is trained on images containing the targets only and the second model is trained on images containing not only targets and objects in the background. When the first model is tested on video, we found that the model is performing very well when a video containing only the target is used. It identifies the target with a high confidence score when the drone is stationary or moving and it rarely misses out the target as shown in Figure 18. However, when we test it against the video with a messier background, it performs poorly by identifying human or any other objects in the background as the target with a high confidence score as shown in Figure 19. This leads to a high number of False positive and a low precision.

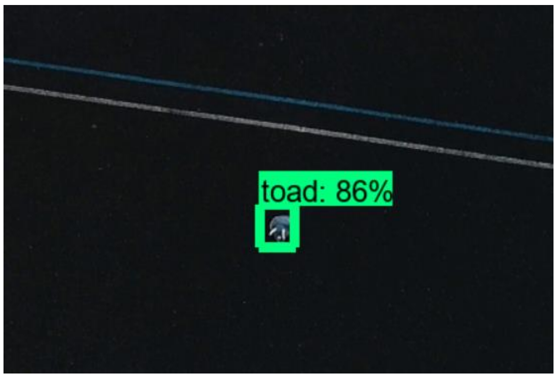


*Figure 18: Classifier accurately detects target*

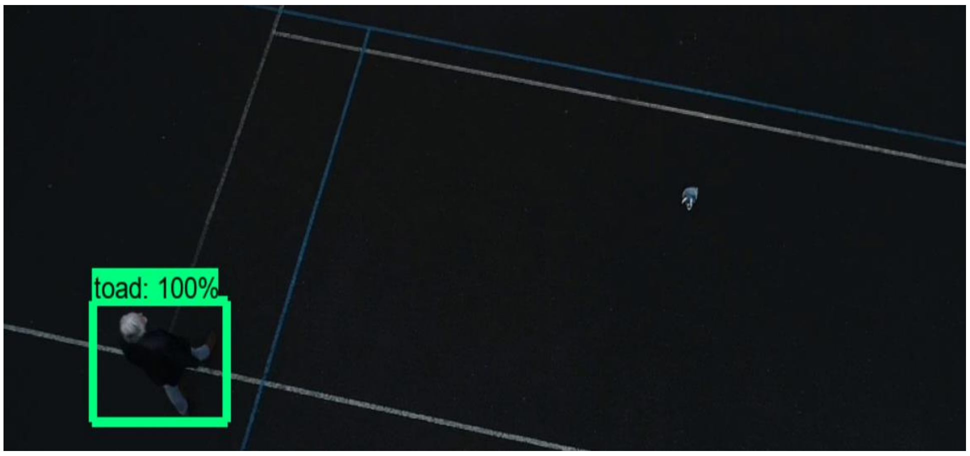


*Figure 19: Classifier detects human as target with 100% confidence score*

The second model trained with images with more background objects. It can identify different objects in the background such as backpack and human. When we test this model with the first video, it performs as well as the first model meaning that it is as good as the first model in identifying targets in a simple background. As expected, when it gets tested with the second video. It can identify the background objects correctly and being able to identify the target as shown in Figure 20. This model rarely misclassifies objects and miss out targets in the video. Therefore, this model has a high precision and a high recall.

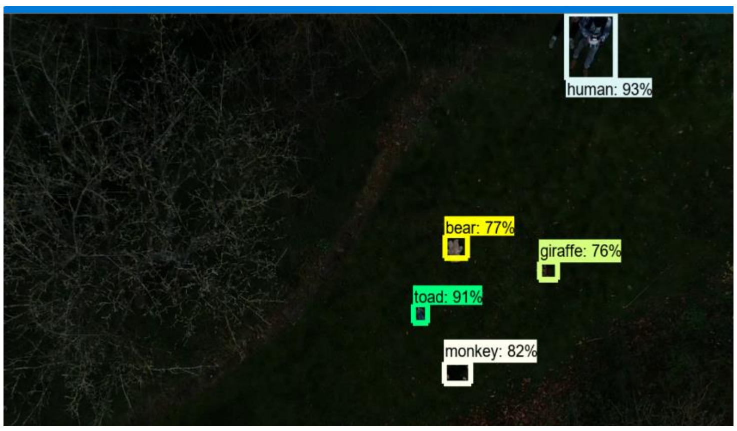


*Figure 2018: Classifier making correct predictions*

Comparing the performance of these two models, we conclude that the first model is not good enough to be implemented in real life. Although it performs very well in a plain background, its inaccurate predictions in messier background will lead to a lot of errors in real life application. While the second model performs greatly at low height, but it starts to misclassify or miss out objects when we fly at a greater height as shown in Figure 22. There is noticeable decrease in accuracy when the image is taken at a greater height.

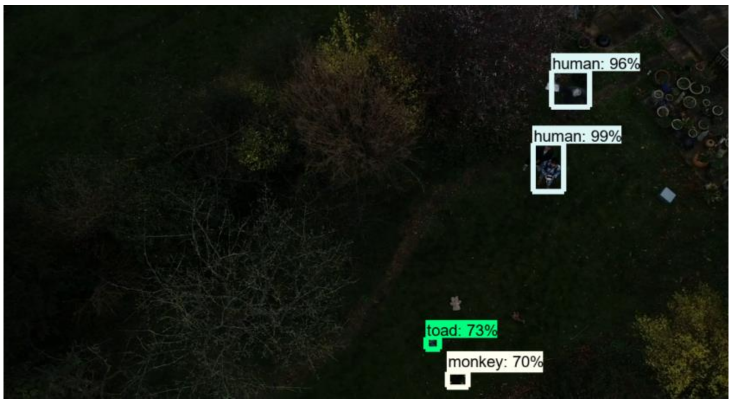


*Figure 19:Classifier misses out monkey and giraffe at 60 feet*

***Model Evaluation***

As the first model is deemed to be not good enough for real life implementation. Only the second model is being evaluated here. We use an open source code to construct the confusion matrix for the model. To calculate the confusion matrix, we need specify a threshold IOU above which the classifier will make a classification on the object and IOU is set to be 60% for this calculation. Confusion matrix is constructed using images in the test pile where all the classifications made by the model will be displayed in the matrix. A perfect model with 100% for both precision and recall will have a diagonal confusion matrix. The matrix for this model is shown below.

| truth/predicted | Toad | Human | Monkey | Backpack | Bear |
|---|---|---|---|---|---|
| Toad | 24 | 2 | 1 | 0 | 0 |
| Human | 2 | 27 | 1 | 0 | 0 |
| Monkey | 3 | 0 | 22 | 3 | 0 |
| Backpack | 1 | 0 | 3 | 31 | 0 |
| Bear | 0 | 0 | 0 | 0 | 27 |

*Confusion Matrix*

According to the matrix, the model confuses human, backpack and monkey several times. It is very likely that these wrong predictions are made when the image is taken at a great height as we see that the accuracy drops when height increases. As height increase, object features become less noticeable to the algorithm and the model finds it hard to distinguish objects from one and other. Another reason is that the color of classes like human, backpack and monkey are dark while bear is of bright color and we can see that the model has a 100% precision in identifying bear.

To evaluate the model, we use the matrix to calculate precision and recall for the model. We found the precision for toad to be 88.8% and the recall to be 80.0%. The precision is slightly lower than the precision which means that the model is more likely to produce false positive than false negative. Overall speaking, the model accurately classifies object into their corresponding classes with a decent precision and recall.

For further development, model’s performance at greater height needs to be improved. As objects features are getting less significant when height increases, a lot more images at these heights need to be used to train the model such that the model could tell tiny differences between different classes and make more accurate predictions.

Apart from using more aerial images to train the model, an infrared object classifier can work concurrently with this model so they could cross check each other. The infrared classifier could compensate for the loss in accuracy with height and it will improve the algorithm accuracy.

## ***Conclusion***

To conclude, we implement a pre-trained RCNN network and train it with around 200 aerial images. The model has a precision of 88.8% and a recall of 80.0%. Overall speaking, the model is performing adequately for height below 60 feet. To further improve this model, we hope to train it with more high-altitude aerial images and develop an infrared object classifier which cross check with this model to improve overall accuracies.

## ***Acknowledgement***

I would like to thank Nabeel Chaudhary for being a teammate throughout this project and Dr. John Hassard for always supporting us with equipment we need and advising us when we are stuck.

## ***Reference***

[1] Rohith Gandhi,R-CNN, Fast R-CNN, Faster R-CNN, YOLO — Object Detection Algorithms, https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[2] SoftMax Layer,

https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

[3] Stanford University, Convolutional Network,

https://cs231n.github.io/convolutional-networks/#fc

[4] Danqing Liu, A Practical Guide to ReLU,

https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

[5] Sumit Saha, A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[6] Ravindra Palmar, Common Loss functions in machine learning

https://towardsdatascience.com/common-loss-functions-in-machine-learning-46af0ffc4d23

[7] Jonathan Hui, mAP (mean Average Precision) for Object Detection

https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

[8] EdjeElectronics, How to Train an Object Detection Classifier for Multiple Objects Using TensorFlow (GPU) on Windows 10

https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10#1-install-anaconda-cuda-and-cudnn

[9] Santiago, Confusion Matrix in Object Detection with TensorFlow

https://towardsdatascience.com/confusion-matrix-in-object-detection-with-tensorflow-b9640a927285