# **Capstone Project - Wildlife**

# Detecting Poachers

Difficulty: ☆☆

**The goal** of this challenge is to detect poachers in the frames from a video feed. The videos are made with a drone that flies over South African wildlife reserves. This drone is used to locate and track down poachers in order to protect wildlife.

**Dataset description**

We worked with two subfolders within the given dataset source. One with annotations in COCO format, and the other with images divided over train, val & test sets. Each of these sets contains frames from different video streams filmed and recorded by the drone. The train set contains 13111 frames, the val set contains 3278 frames, and the test set contains 1900 frames. The size of all images is 640x470 pixels.



**Pre-processing steps taken**

* Importing the required libraries.
* Exploring the COCO annotations through the following steps:

1. Exploring the 3 COCO datasets given namely;
2. Instances\_train.json
3. Instances\_test.json
4. Intances\_val.json;
5. Mounting the annotations folder from the drive and treating it without making any changes to their format with the use of the COCO Python API.
6. Utilizing the COCO API to read and extract annotations conveniently through the following:
7. Retrieving the Super\_categories present in the dataset.
8. Retrieving the number of categories and their respective ids
9. Retrieving the Category names.
10. Retrieving the Category names with their ids .
11. Retrieving the number of images for each category in a set.
12. Retrieving the images by image ID.
13. Loading images and getting their annotations.
14. Loading random images with segmented objects inside them.

**Observation**

Of the 3 categories of images present in the dataset, only the Human Category contains images making it pointless to explore any kind of filtration.

**Model specifics**

We worked with the Detectron-2 model, a successor of Detectron and maskrcnn-benchmark from the Facebook AI Research's next generation library that provides state-of-the-art detection and segmentation algorithms.

The model interaction was as follows:

1. Installing all the necessary dependencies to enable smooth running of the model.
2. Configuring and visualizing the dataset to get familiar with the data.
3. Training the model on the train image set while tuning different parameters such as the maximum number of iterations to avoid overfitting with the training sprint tracked using the tensorboard.
4. Evaluating the model using the val image set.

**Results (evaluation metrics for train and test set)**

For training the dataset we implemented Transfer Learning using PyTorch Decetron2 and used the Faster RCNN X101-FPN pretrained model.

The model is trained for 1500 iterations.

**Training Metrics**

Evaluation results for bbox:

| **AP** | **AP50** | **AP75** | **APs** | **APm** | **API** |
| --- | --- | --- | --- | --- | --- |
| 21.685 | 59.247 | 8.245 | 21.685 | nan | nan |

**Test Metrics**

Evaluation results for bbox:

| **AP** | **AP50** | **AP75** | **APs** | **APm** | **API** |
| --- | --- | --- | --- | --- | --- |
| 29.666 | 65.727 | 22.183 | 29.813 | 13.410 | nan |

With these results it has been observed that the model has learned something and is performing well on new data.

The following images depict that model is able to detect ‘human’ objects quite accurately.

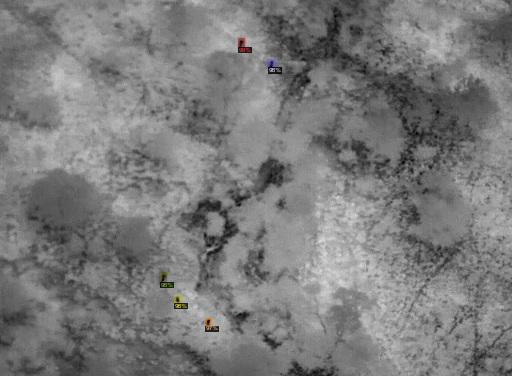




The model can identify ‘human’ objects accurately in the image, even when there are other parts of images that have patches similar to ‘human’ objects.







The below image shows that the model accurately finds human objects between other objects.





In many observations on the test results, we found that the model sometimes draws extra bounding boxes due to which we received smaller accuracy in the training and testing metrics.

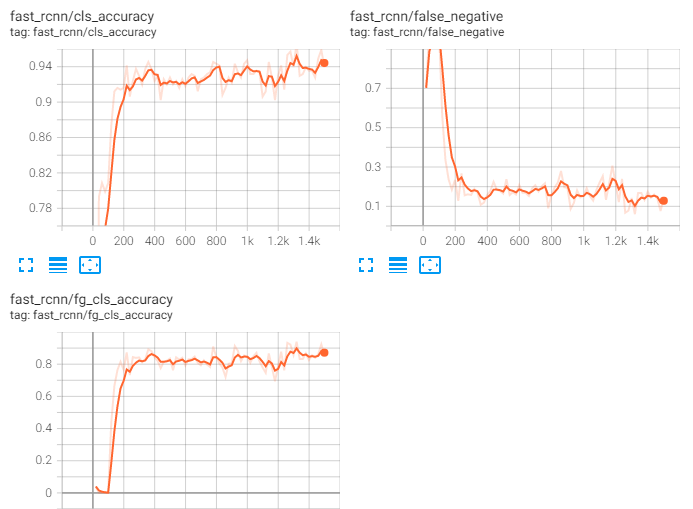


Following is the classification accuracy during training.

Classification Accuracy: 0.9442

False Negative: 0.1284

Classification Accuracy for Foreground Proposals: 0.8716

****

It shows that the model is able to detect the objects in the frame with good accuracy.

**Potential improvements**

1. Transformation and exploration of data in any other possible formats other than the COCO format given such as Cvat by OpenCv and Label-studio by Heartx.
2. Using better detection models to train on the COCO formatted datasets and other metrics to evaluate the model architecture.
3. Tuning other related model parameters to optimize model performance such as training the model for more epochs.

**Overview of work division in the team**

1. Data Exploration and Data Preprocessing - Data was explored by both Catherine Auma and Yastika with Catherine taking charge to further complete preprocessing successfully.
2. Model and Architecture - Model was prepared by Timothy Malche with support input from Catherine on fixing a few errors.
3. Report Writing - The draft was done by Yastika with support input from Evan and Timothy.
4. Final Presentation - The draft was done by Evan with support input from Yastika and a few adjustments by Timothy.