2018

**Talha Usman**

**Saad Bin Khalid**

**BETE 52-D**

**7/2/2018**

**Target Detection on Satellite Imagery**



**Target Detection on Satellite Imagery**

**Introduction:**

The repository contains scripts that enable the automatic detection of aircraft in [Planet](https://www.planet.com/) imagery using machine learning techniques. Included are files which define a machine learning model, train it using the Planesnet dataset in json format, and apply it across an entire image scene to highlight aircraft detections.

**Methodology:**

[PlanesNet](https://www.kaggle.com/rhammell/planesnet) is a labeled training dataset consiting of image chips extracted from Planet satellite imagery. It contains thousands of 20x20 pixel RGB image chips labeled with either a "plane" or "no-plane" classification. Machine learning models can be trained against this data to classify any given input chip into either one of these classes.

With an accurately trained model, this classification process can be extended to a full Planet image scene by using a sliding window technique. A 20x20 pixel window is moved across each pixel position in the image, extracted, and classified by the model. Neighboring window poistions that are classified as "plane" are then clustered into a single detection. These detections are highlighted with a bounding box in a copy of the original Planet scene.

**Libraries Used:**

1. **Numpy**
2. **Scipy**
3. **Tensorflow**
4. **Tflearn**
5. **Pillow**

**Training and Model:**

Train the model created by tflearn using the tensorflow framework. The model.py file is attached. This model supports the 20x20x3 input dimensions of the PlanesNet image data.

# Train the model

mkdir models

python train.py "planesnet.json" "models/model.tfl"

# Run on demo image with defined output path

python detector.py "models/model.tfl" "image/scene\_1.png" "image/scene\_1\_detections.png"

**Detector**

A trained model can be applied across entire images using the sliding window detector function detector.py, which takes the model file path, input image path, and optional output image path as arguments. The output image will cluster positive detections and draw a bounding box around their center point.

# Run on demo image with defined output path

python detector.py "models/model.tfl" "images/scene\_1.png" "images/scene\_1\_detections.png"

**Results:** Tested on imagery collected from planet.com only.

Original Image:



Image after Detection

