BTP-II

AIRCRAFT DETECTION IN SATELLITE IMAGES

TESTING & DEPLOYMENT

Geordi Jose
DEPT. OF CIVIL ENGINEERING
IIT BOMBAY

ABSTRACT

With the previous BTP-I carried out on detecting whether an image contains an aircraft or not and simultaneously another methodology applied to find out where in a given image aircraft is present, with this BTP I aimed at creating a merged and faster solution which can detect and locate (using bounding boxes) aircrafts in digital satellite images. The problem was divided into two problems, with my partner working on preparing a deep learning object detection model, and with me working on testing, validating and deploying the said model. Hyperparameters of the model like epochs, iterations per epoch were improved by testing on dataset and feeding back the results. The mAP (mean Average Precision) score is used to evaluate the model. After testing, a deployable version of our solution was built using ANVIL, an open source tool for developing full stack python web applications. A simple front-end UI was built, which interacts with a python server script on the backend where any user can simply upload an image and run the model on it. The results obtained are promising on the given set of testing images, but generalizes very poorly outside of it. This is due to a very limited artificially generated training dataset and therefore there is much scope in for this project if a larger more versatile dataset can be used.

INTRODUCTION

Object detection in images has been area of research for over many decades and among these, aircraft detection still remains a challenging task because of the complex background, illumination change and variation of aircraft kind. The detection still needs to be carried out for military and security purpose. Various different approach has been carried out for aircraft detection starting from using filter with support vector or coarse-to-fine edge detection, using directional gradient and so on. But these methods use low-level features and thus has high false alarm. With the introduction of neural networks works shifted towards using supervised training approach and deep learning methods. Running deep networks on sliding windows for object proposal proved to be accurate than previous approaches but was very slow in execution. To increase the speed, detection based on spatial pyramid pooling method was adopted but the issue of slow candidate region proposal network still remained. Later the concept of using convolutional neural network along with region proposal network was introduced which proved to be very accurate and also reducing the runtime of the execution. The key requirements of using a convolutional structure is that as it involves so many weight parameters so the training data set needs to be very accurate and present in large number. To compensate for large training data set, data augmentation becomes a crucial aspect in this method. Also, the hidden structure of CNN doesn't follow a fixed pattern so finding the appropriate CNN hidden layers varies from the targeted purpose and the kind of dataset that we have. In this project Faster-RCNN technique with region proposal network was incorporated to cater to detection and creating bounding box along the object.

Evaluating the accuracy of the model is also a challenging task as it is not quite as simple as a classification problem. AP (Average precision) is a widely used metric in evaluating the performance of object detectors like Faster R-CNN, SSD, etc. Average Precision (AP) is calculated by finding the area under the precision-recall curve. In our case we have an Intersection over Union (IoU) >0.5 criterion to define a true positive (TP) case with respect to accuracy. We define the True Positive (TP), False Positive (FP), and False Negative (FN) in this manner. Based on this, we can now calculate the precision and recall of our detection for a given class across the test set and use this for calculating average precision.

An aircraft detection solution is expected to have several applications in defence/military as well as civil aviation sectors. Therefore it is important to deploy this work accessible to any ordinary user. Anvil was used to create a web app front end for this work, which interacts with users to accept input images and display output images.

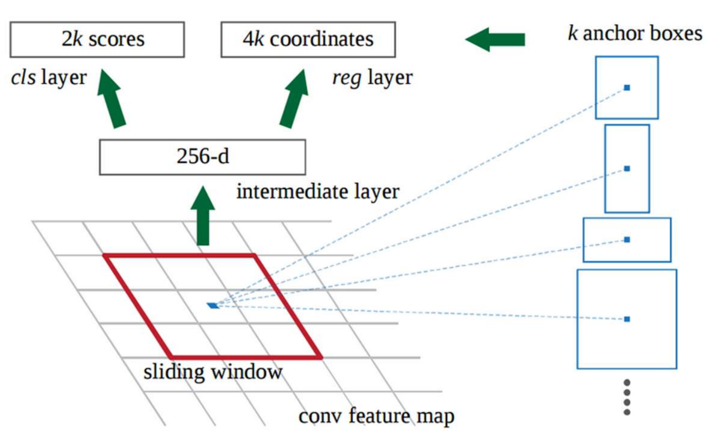
BACKGROUND

BRIEF EXPLANATION OF FASTER- RCNN MODEL EMPLOYED

Faster R-CNN is intended to replace the slow selective search algorithm of Fast RCNN with a fast-neural network. The main concept being the introduction of the region proposal network (RPN). Here's how the RPN works:

•In the final layer of an initial ConvNet (a base network which is VGG-16 in our case), a 3x3 kernel moves across the generated feature map and further maps it to a lower dimension (e.g. 256-d)
•For each such sliding-window locations, multiple possible regions are generated based on a 'k' fixed-ratio called anchor boxes (which become the default bounding boxes)
•Each such region proposal includes of a) an "objectness" score for that region and b) four coordinates (width, height, change in width and change in height) representing the bounding box for the region

To reiterate, we simply look at each location in our output feature map from the base network as an anchor in the original image and propose k (k=3 in our case) different boxes centered around it: i.e. a tall box, a wide box, a square box, etc. by defining an area and ratio of width to height.
Subsequently for each of these boxes, we also calculate whether or not we think it contains an object, and also what the coordinates for that box are (which will be regressed to better match the object location). So, one sliding window location would look something like this:



In this image, the 2k scores indicate the softmax probability of each of the k bounding boxes being an “object.” It could be any object, as it is important to note that although the Region Proposal Network outputs a set of bounding box coordinates, in this stage it does not try to classify which object class the potential object belongs to. The sole objective here is to simply propose candidate object regions for subsequent analysis. Among such candidate regions if any anchor bounding box has an “objectness” score (the softmax probability) greater than a certain threshold value, then that box and its coordinates are further fed forward as a region proposal to the subsequent ConvNet layers.

In this manner for a single image once we have the candidate region proposals from the RPN, they are fed forward into a Fast R-CNN architecture which basically includes a pooling layer, some fully-connected (dense) layers, and in the end a softmax classification layer for object classification and a bounding box regressor to fit the boxes better in the image.So in simpler terms Faster R-CNN can be looked at as a combination of an RPN with a Fast R-CNN.

BRIEF EXPLANATION OF MEAN AVERAGE PRECISION (mAP) SCORE

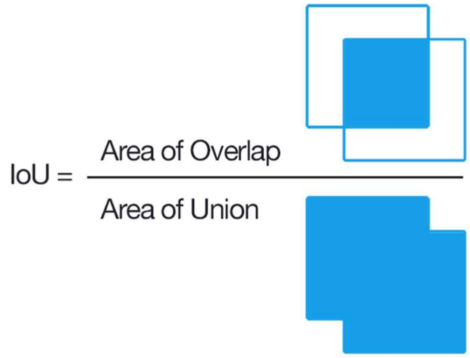
Precision and recall are two very popular metrics employed to evaluate the performance of any classification model. To understand how mAP is calculated, we would need to first calculate precision and recall values. Precision of a given class in classification, or the positive predicted values, is defined as the ratio of true positive (TP) to the total number of predicted positives.

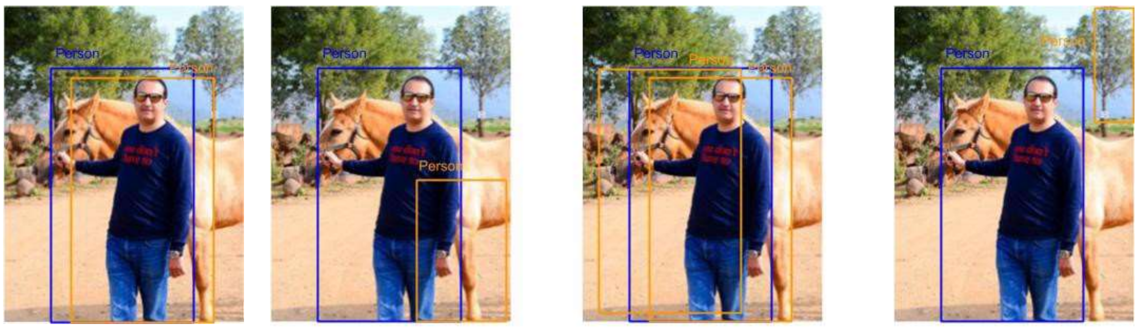
$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

The recall, or the true positive rate (also often referred to as sensitivity), of a given class, is defined as the ratio of True Positives (TP) to the total of ground truth positives.

$$\text{Recall (TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Obviously, there lies a trade-off between its precision and recall performance. In order to have a high precision in classification, we need to decrease the number of False Positives (FP) in our predictions, but by doing so, recall value for the same classification would decrease. Similarly, if the number of False Negatives (FN) is decreased, it would improve the recall value while reducing the precision value. Most often for information retrieval and object detection cases, we ideally want a higher i.e our predicted positives to be True Positives (TP). In the context of using a neural network for classification, this accuracy recall trade-off can be adjusted by tweaking the model's final layer softmax activation thresholds. But for implementing this concept into an object detection model, we need to define what a True positive or False Negative case is. For this we use the concept of Intersection over Union (IoU). The IoU is calculated as the ratio of the area of intersection and area of union between the model's predicted bounding box and the ground truth bounding box.





IoU of predicted Bounding Box (yellow) and Ground truth Bounding Box (blue) > 0.5 with the correct classification

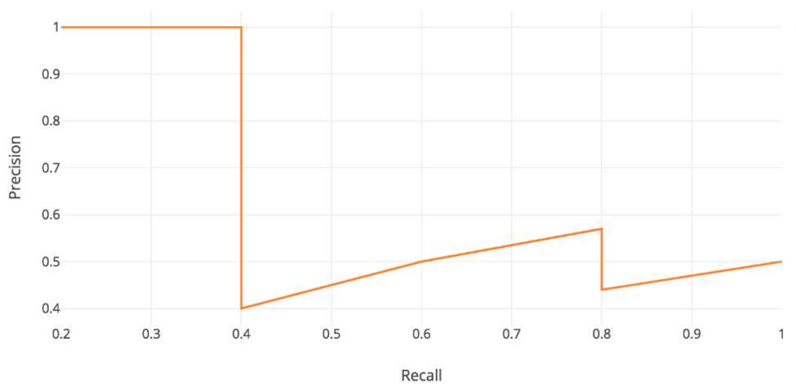
IoU < 0.5

Duplicate BB are considered as FP

No IoU

Usually, we define a prediction to be a True Positive (TP) if the IoU is > 0.5. However, there are two possible scenarios where a Bounding Box would be considered as a False Positive: an IoU < 0.5 or a Duplicated Bounding Box. For our single class detection problem, False Negative is when there is no detection at all. So with this formal definition of TP, FP and FN, we are able to calculate the precision and recall values for our detection results for any given class across the test set. To reiterate, The general definition for the Average Precision (AP) is finding the area under the precision-recall curve.

$$\mathrm{AP} = \int_0^1 p(r)dr$$



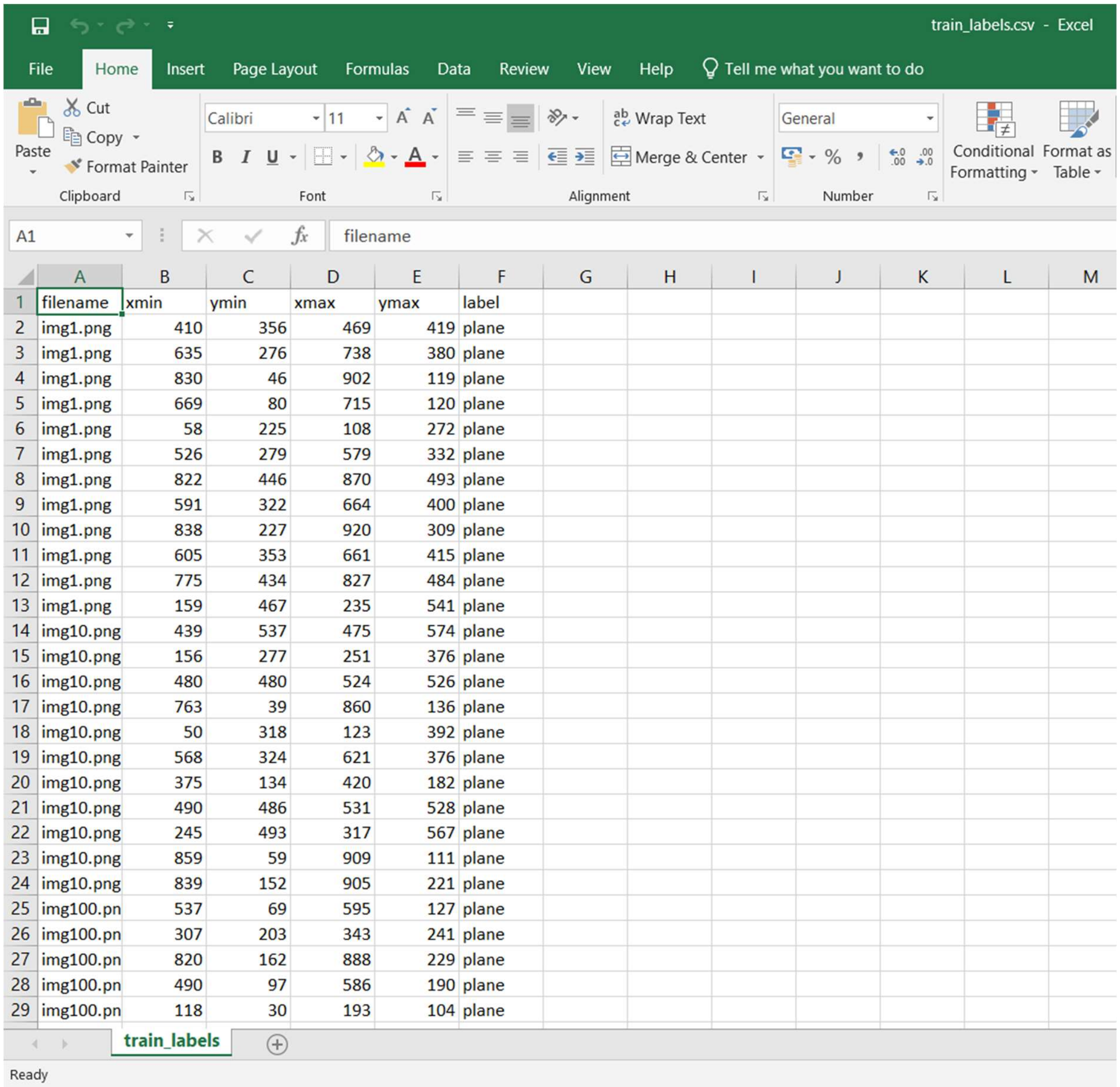
Precision and recall values are bound by 0 and 1. Therefore, AP also falls within 0 and 1. The mAP (mean Average Precision) for object detection is then calculated as the average of the AP calculated for all the classes, which in our specific case only across the airplane class.

METHODOLOGY

DATASET

Dataset for testing and training included a set of 500 computer generated imagery (CGI) of satellite images with aircrafts placed randomly across the image. 400 images were used for augmentation and training followed by testing on 100 test images.

Sample Test images



	A	B	C	D	E	F
1	filename	xmin	ymin	xmax	ymax	label
2	img1.png	410	356	469	419	plane
3	img1.png	635	276	738	380	plane
4	img1.png	830	46	902	119	plane
5	img1.png	669	80	715	120	plane
6	img1.png	58	225	108	272	plane
7	img1.png	526	279	579	332	plane
8	img1.png	822	446	870	493	plane
9	img1.png	591	322	664	400	plane
10	img1.png	838	227	920	309	plane
11	img1.png	605	353	661	415	plane
12	img1.png	775	434	827	484	plane
13	img1.png	159	467	235	541	plane
14	img10.png	439	537	475	574	plane
15	img10.png	156	277	251	376	plane
16	img10.png	480	480	524	526	plane
17	img10.png	763	39	860	136	plane
18	img10.png	50	318	123	392	plane
19	img10.png	568	324	621	376	plane
20	img10.png	375	134	420	182	plane
21	img10.png	490	486	531	528	plane
22	img10.png	245	493	317	567	plane
23	img10.png	859	59	909	111	plane
24	img10.png	839	152	905	221	plane
25	img100.pn	537	69	595	127	plane
26	img100.pn	307	203	343	241	plane
27	img100.pn	820	162	888	229	plane
28	img100.pn	490	97	586	190	plane
29	img100.pn	118	30	193	104	plane

Bounding box coordinates for each image

TESTING AND mAP SCORE CALCULATION

Testing was performed in Google Colab GPU runtimes, for efficient results, the same environment where the model was trained. Each image is passed through the RPN network followed by the trained CNN. Hyperparameters for the model like the number of epochs, iterations per epoch, threshold values were fed back into the model after testing based on mAP scores obtained. An accuracy threshold of 0.9 is set for classification i.e the model will locate and display an aircraft only if the final associated probability after prediction of the region being an aircraft is at least 0.9 (or 90 percent surety).

```python
def get_map(pred, gt, f):
    T = {}
    P = {}
    fx, fy = f

    for bbox in gt:
        bbox['bbox_matched'] = False

    pred_probs = np.array([s['prob'] for s in pred])
    box_idx_sorted_by_prob = np.argsort(pred_probs)[::-1]

    for box_idx in box_idx_sorted_by_prob:
        pred_box = pred[box_idx]
        pred_class = pred_box['class']
        pred_x1 = pred_box['x1']
        pred_x2 = pred_box['x2']
        pred_y1 = pred_box['y1']
        pred_y2 = pred_box['y2']
        pred_prob = pred_box['prob']
        if pred_class not in P:
            P[pred_class] = []
            T[pred_class] = []
        P[pred_class].append(pred_prob)
        found_match = False

        for gt_box in gt:
            gt_class = gt_box['class']
            gt_x1 = gt_box['x1']/fx
            gt_x2 = gt_box['x2']/fx
            gt_y1 = gt_box['y1']/fy
            gt_y2 = gt_box['y2']/fy
            gt_seen = gt_box['bbox_matched']
            if gt_class != pred_class:
                continue
            if gt_seen:
                continue
            iou_map = iou((pred_x1, pred_y1, pred_x2, pred_y2), (gt_x1, gt_y1, gt_x2, gt_y2))
            if iou_map >= 0.5:
                found_match = True
                gt_box['bbox_matched'] = True
                break
            else:
                continue

        T[pred_class].append(int(found_match))

    for gt_box in gt:
        if not gt_box['bbox_matched']:# and not gt_box['difficult']:
            if gt_box['class'] not in P:
                P[gt_box['class']] = []
                T[gt_box['class']] = []

            T[gt_box['class']].append(1)
            P[gt_box['class']].append(0)

    #import pdb
    #pdb.set_trace()
    return T, P
```

Code for mAP score computation

```
Elapsed time = 0.551131010055542
plane AP: 0.5401217304735948
mAP = 0.5401217304735948
94/100
Elapsed time = 0.5796689987182617
plane AP: 0.5397114055931123
mAP = 0.5397114055931123
95/100
Elapsed time = 0.5873677730560303
plane AP: 0.5324633435070996
mAP = 0.5324633435070996
96/100
Elapsed time = 0.567589521408081
plane AP: 0.5299526700697973
mAP = 0.5299526700697973
97/100
Elapsed time = 0.5438728332519531
plane AP: 0.5290003837731405
mAP = 0.5290003837731405
98/100
Elapsed time = 0.5564498901367188
plane AP: 0.527798500775934
mAP = 0.527798500775934
99/100
Elapsed time = 0.5385639667510986
plane AP: 0.5269807580818849
mAP = 0.5269807580818849

mean average precision: 0.5386784240274048
```

mAP scores for each image followed by average

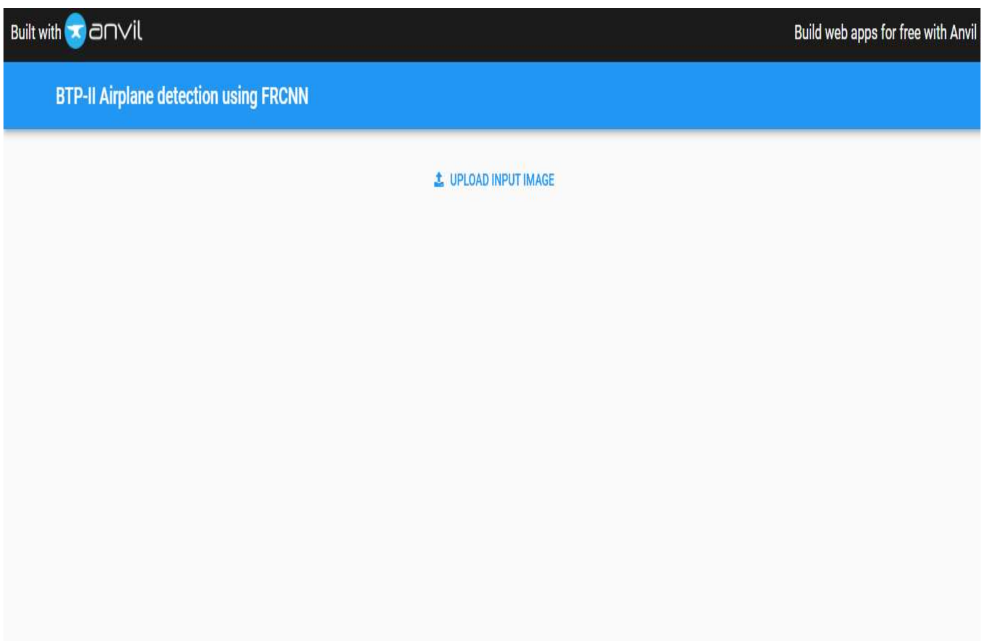
WEB APP FOR DEPLOYING THE MODEL

The front end web app was created using a free open source tool ANVIL, which enables developing full webpages using python alone. This web app enables user to upload an input image to the model and view results without any technical hindrance. If necessary, users can also specify the accuracy threshold they want to be applied for object detection.

```python
from ._anvil_designer import Form1Template
from anvil import *
import anvil.server

class Form1(Form1Template):

  def __init__(self, **properties):
    # Set Form properties and Data Bindings.
    self.init_components(**properties)
    self.label_1.text= "BTP-II Airplane detection using FRCNN"
    self.file_loader_1.text="Upload Input Image "
    self.image_1.display_mode='original_size'
    self.image_2.display_mode='original_size'

    # Any code you write here will run when the form opens.

  def file_loader_1_change(self, file, **event_args):
    """This method is called when a new file is loaded into this FileLoader"
    self.image_1.source=self.file_loader_1.file
    my_media=self.file_loader_1.file
    processed_image=anvil.server.call('process_image', my_media)
    self.image_2.source=processed_image
```

ANVILcode

The anvil code's file loader change function is called when the user inputs an image. Subsequently the image is passed on to a server side script by invoking the 'process-image' function defined there.



Anvil Design

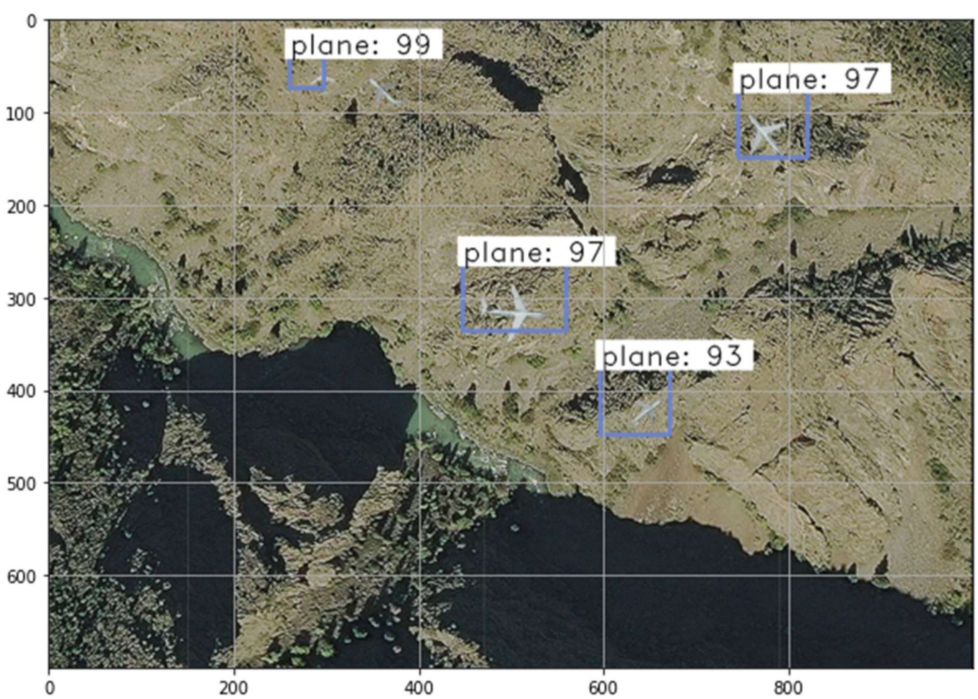
```
[ ] Connecting to wss://anvil.works/uplink
    Anvil websocket open
    Authenticated OK

@anvil.server.callable
def process_image(my_media):

  import numpy as np
  import cv2

  # read in image from media object
  arr = np.fromstring(my_media.get_bytes(), np.uint8)
  im = cv2.imdecode(arr, cv2.IMREAD_COLOR)
  cv2.imwrite('input.png', im)
  ####
  ifilepath='input.png'
  findplane(ifilepath)
  ofilepath=('result.png')
  im=cv2.imread(ofilepath)
  ####

  # convert image to string and return downloadable media
  im_str = cv2.imencode('.png', im)[1].tostring()

  return anvil.BlobMedia("image/png", im_str, name='myimage.png')
```

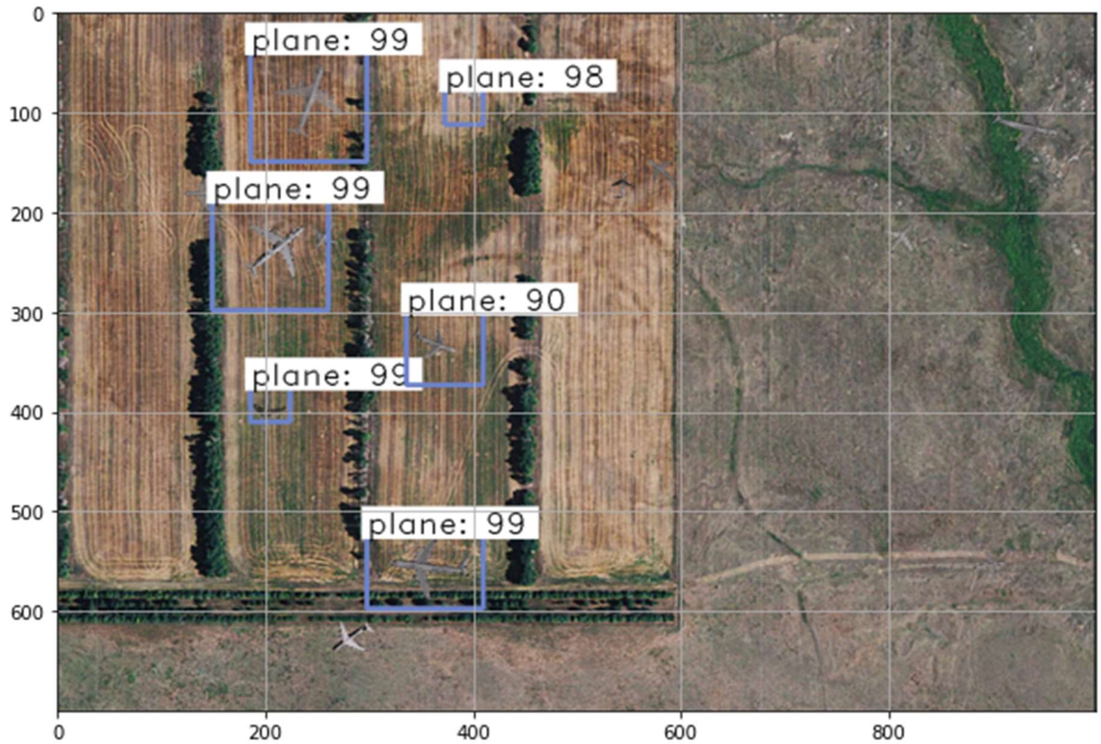
Server Side Script on Colab

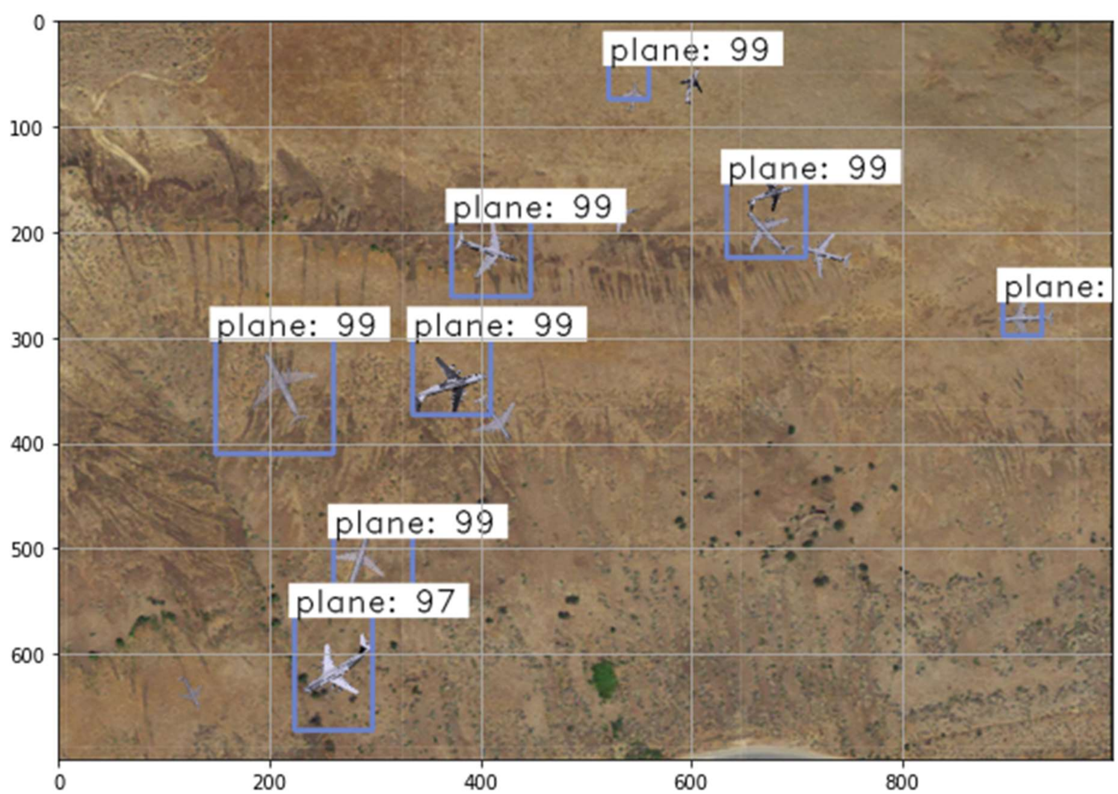
The server side script on Google Colab. The process image function receives the media object from anvil, and after converting this to an image file, the findplane function is called which predicts airplanes in the image using the trained and tested FRCNN image. The output image of this function is passed back to anvil after converting it to downloadable media.

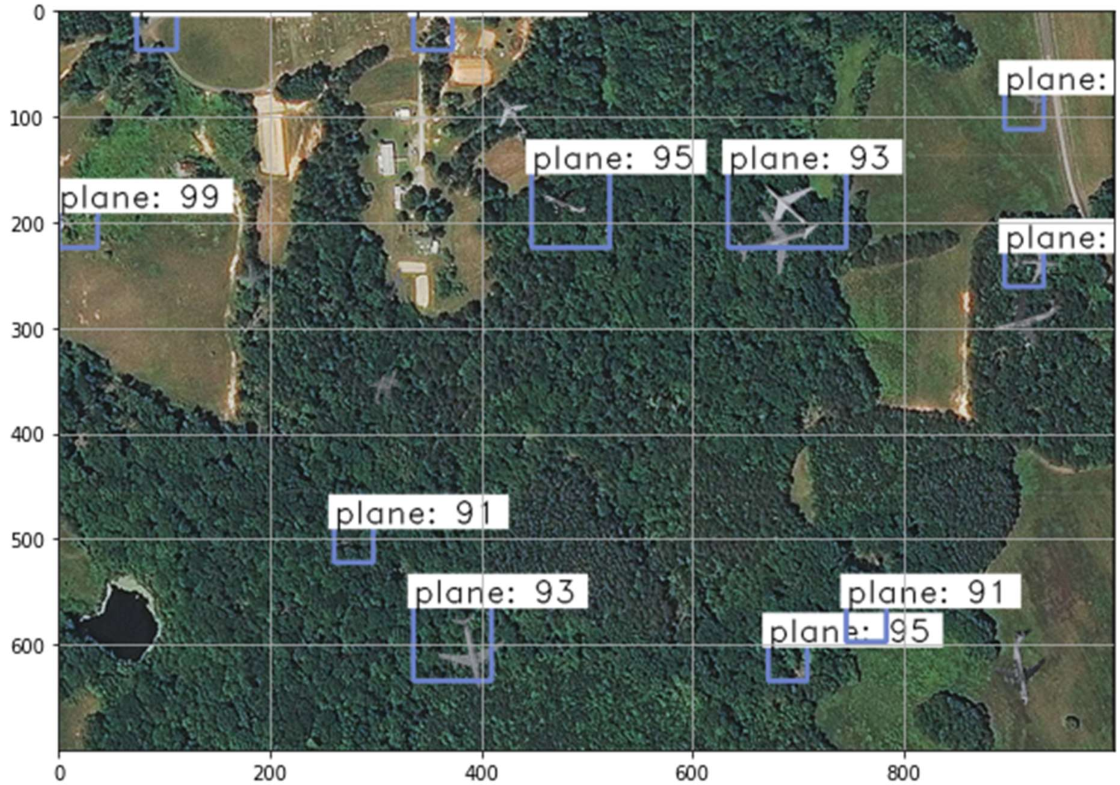
RESULTS

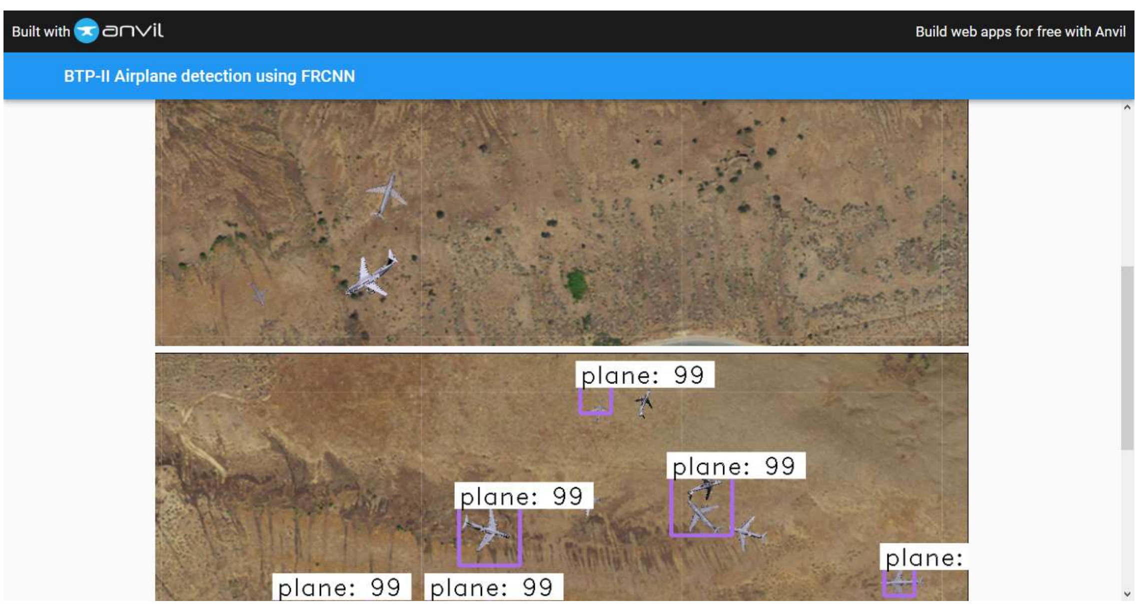
After final testing an mAP score of 0.54 was achieved. Some of the test result images:




plane: 99
plane: 98
plane: 99
plane: 90
plane: 99
plane: 99
0
100
200
300
400
500
600
0
200
400
600
800


plane: 99
plane: 99
plane: 99
plane:
plane: 99
plane: 99
plane: 99
plane: 97
0
100
200
300
400
500
600
0
200
400
600
800





1ANVIL web app for airplane detection

CONCLUSIONS

A solution has been developed to identify and locate aircrafts from digital images using a Faster R-CNN deep learning model in the backend. The model is developed using artificially generated satellite imagery. The model performs really well on images similar to the dataset it was trained on, and achieved an accuracy score (mAP) of 0.54 after testing on 100 images. In the limited domain the model has been trained on, it performs really well, and therefore proves promising for working on larger datasets. As of now performance outside the trained dataset is poor i.e the model does not generalise well. A web app front end foraccesing the model has also been developed.

LIMITATIONS AND SCOPE FOR FUTURE WORK

With the objective of locating any aircraft present in an image, this model has its own restrictions. Beyond the realms of training dataset, it generalizes very poorly and there are two major reason for this. First being the original dataset, itself is being computer generated and not comes from original surrounding. The aircraft has been put randomly on different backgrounds which leads to low representation of actual environment. Second major reason is the limited amount of dataset the model is trained on. There were only 400 original images on which augmentation was performed to increase it to 4400. Having a large set of original datasets covering all general environments would certainly boost the accuracy.

For the usage of this model, a UI has been hosted on anvil server where any use can upload any image and then would get the output with bounding box created around aircrafts present if any in that image. If we have a capable enough backend server we can even run this on a video input, splitting each video into multiple frames and then performing testing on each generated image.

As the entire framework has been developed such that it can detect any given object provided sufficient training has been done, so expanding the horizon of the dataset will lead to model predicting many different types of objects as and when trained upon. With bigger more versatile datasets, rather than looking at a single aircraft class, looking for multiple aircraft classes i.e identifying even the type of aircrafts is achievable.

REFERENCES

1.Faster R-CNN: Towards Real-Time Object Detectionwith Region Proposal Networks [Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun]   Microsoft Research{v-shren, kahe, rbg, jiansun}@microsoft.com

2.X. Li, S. Wang, B. Jiang and X. Chan, "Airplane detection using convolutional neural networks in a coarse-to-fine manner," 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, 2017, pp. 235-239. doi:10.1109/ITNEC.2017.8284943

3.Xu, Yuelei et al. “Rapid Airplane Detection in Remote Sensing Images Based on Multilayer Feature Fusion in Fully Convolutional Neural Networks.” Sensors (Basel, Switzerland) vol. 18,7 2335. 18 Jul. 2018, doi:10.3390/s18072335

4.Chen, Xueyun, Shiming Xiang, Cheng-Lin Liu and Chunhong Pan. “Aircraft Detection by Deep Convolutional Neural Networks.” IPSJ Trans. Computer Vision and Applications 7 (2014): 10-17.

5.Li, Yang; Fu, Kun; Sun, Hao; Sun, Xian. 2018. "An Aircraft Detection Framework Based on Reinforcement Learning and Convolutional Neural Networks in Remote Sensing Images." Remote Sens. 10, no. 2: 243.

6.Xu T.B., Cheng G.L., Yang J. Fast Aircraft Detection Using End-to-End Fully Convolutional Network; Proceedings of the 2016 IEEE International Conference on Digital Signal Processing (DSP); Beijing, China. 16–18 October 2016

7.Zhang W., Sun X., Fu K., Wang C., Wang H. Object detection in high-resolution remote sensing images using rotation invariant parts based model. IEEE Trans. Geosci. Remote Sens. 2014;11:74–78. doi: 10.1109/LGRS.2013.2246538.

8.Ren S., He K., Girshick R.B. Object Detection Networks on Convolutional Feature Maps. IEEE Trans. Pattern Anal. Mach. Intell. 2017;39:1476–1481. doi: 10.1109/TPAMI.2016.2601099.

9.Dataset obtained from Kaggle - https://www.kaggle.com/rhammell/planesnet/data

10. information on mAP score -https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52