OBJECT DETECTION(AIRCRAFT) & BOUNDING BOX PROBLEM

B. Tech Project - 2

BY- UJJWAL KUMAR

160040089

SUPERVISOR- PROF. J. INDU

DEPARTMENT OF CIVIL ENGINEERING

IIT BOMBAY

ACKNOWLEDGEMENT:

I want to express my sincere gratitude and hearty thanks to my supervisor Prof. Indu J. for her continuous supervision. Her valuable suggestions at every stage of this research work have been beneficial. I would also like to thank her for being so supportive and motivating.

Special thanks to my wing mates for being a constant source of support and entertainment, every time I was stressed. I am deeply indebted to my family for their constant and unconditional support in this endeavor.

Finally a thanks to my project partner Geordi, without whom completion of this project would be a far fetched dream.

Ujjwal Kumar
(160040089)

Table of Contents

ABSTRACT: 5
INTRODUCTION: 6
BRIEF EXPLANATION OF FASTER- RCNN: 7
METHODOLOGY: 9
  DATASET: 9
  BUILDING THE MODEL: 11
RESULTS: 14
LIMITATIONS & IMPROVEMENTS: 18
FUTURE POSSIBILITIES: 19
BILBLIOGRAPHY: 20

DECLARATION:

I declare that this written submission contains my ideas in my own words and where other’s ideas or words have been included, I have adequately cited and referenced the original Sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will cause disciplinary action by the Institute and can evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken when needed.

Ujjwal Kumar
(160040089)

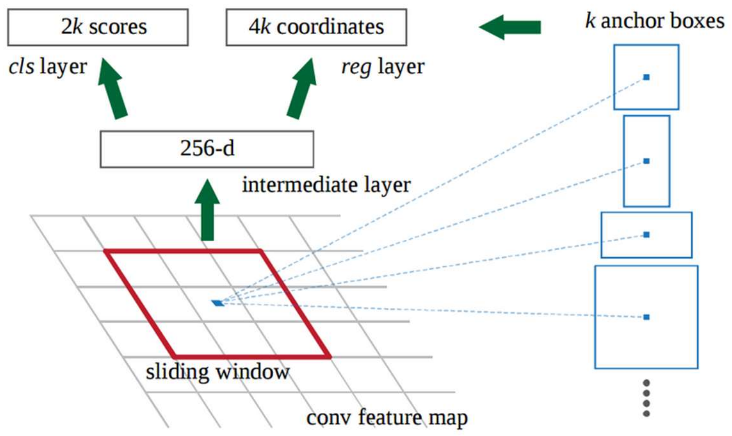
ABSTRACT:

With the previous BTP 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, this BTP aimed at creating a merged and faster solution i.e creating bounding box around the aircraft. For this VGG network was used and Faster RCNN technique was applied. Augmentation was applied on data to increase number of training samples, increasing the number from 400 to 4400 (10 different augmentation was performed). 1000 epoch of training was done using google colab (GPU) and weights were saved. A UI was created on Anvil server, connected with script running on colab and then for testing anyone could use the server to upload the image and get the output. The original images were also artificially generated, aircraft being placed with some random background to make it similar to satellite image, and the reason for this is because we required satellite-based data with bonding box coordinate already known to carry out supervised classification. Already existing VGG network was tweaked to make some layers of it as trainable and then training was carried out. The result obtained is very limited to given set of testing dataset and generalizes very poorly due to limited training dataset but limited can be easily eliminated if larger dataset is used for training.

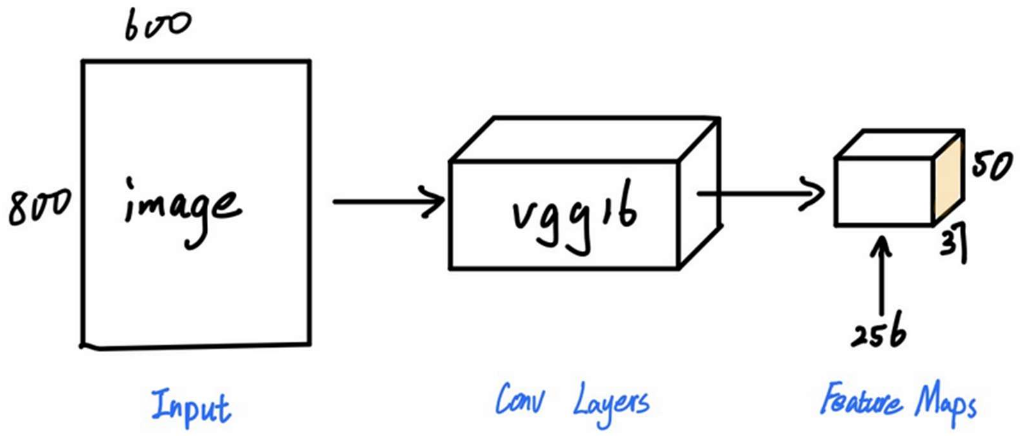
INTRODUCTION:

Detection of objects in an image has been area of research for over many decades and among these, detection of aircraft 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.

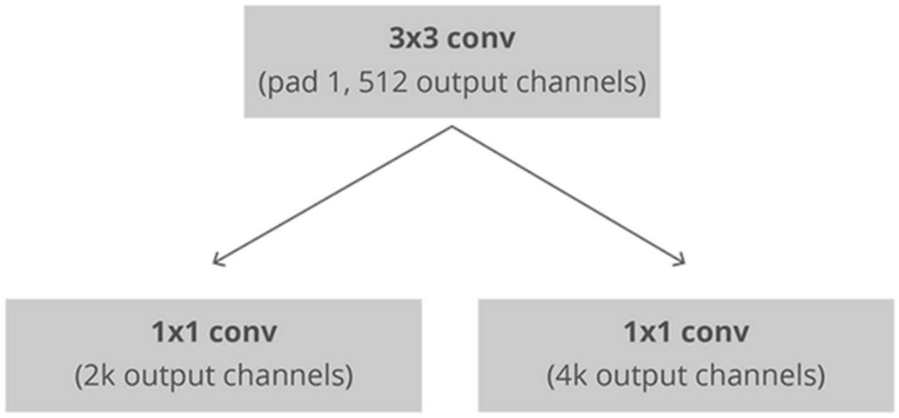
BRIEF EXPLANATION OF FASTER- RCNN:



In case of Faster RCNN there are 2 different process merged in one. Instead of using a sliding window approach to generate probable required region (computation cost becomes very high), it uses a region proposal network which makes entire computation very fast. For instance, if input data is of size 800*600*3 then after passing through convolution network, the feature map output would be 37*50*256



The entire 1st layer of the output is considered as an anchor. Then we define different anchor of varying sizes. For each anchor size and anchor point, we pass it to a 3*3 conv layer with padding 1 and 512 channels. The output of those passed upon a 1*1 conv gives output for classification layer to check whether box contains the required object or not and 2nd output does the box-regression to have maximum IoU with ground truth data



As we have only data for the object that is required for detection. So during training, those boxes which have IoU<0.5 are considered as “background” while the rest as “foreground”. This helps during prediction as the object and background maybe very similar as thus this step enhances the accuracy.

For every anchor point we define 9 different size and ratio boxes. Thus, we total generate 37*50*9 = 16650 boxes. Minibatch size of 256 is taken during training. To ensure an object is detected only once, we do non-maximum suppression wherein first remove boxes having probability less than threshold. Then for remaining boxes, pick box with highest probability and discard any remaining boxes with IoU>0.5 with the selected boxes.

Finally, we pool this, then flatten with some connected layers and then perform softmax classification.

METHODOLOGY:

DATASET:

The dataset had originally had 400 training images and 100 testing images along with bounding box created for aircrafts. Along with this, a csv file was given containing coordinate of objects (aircrafts) and name of file.

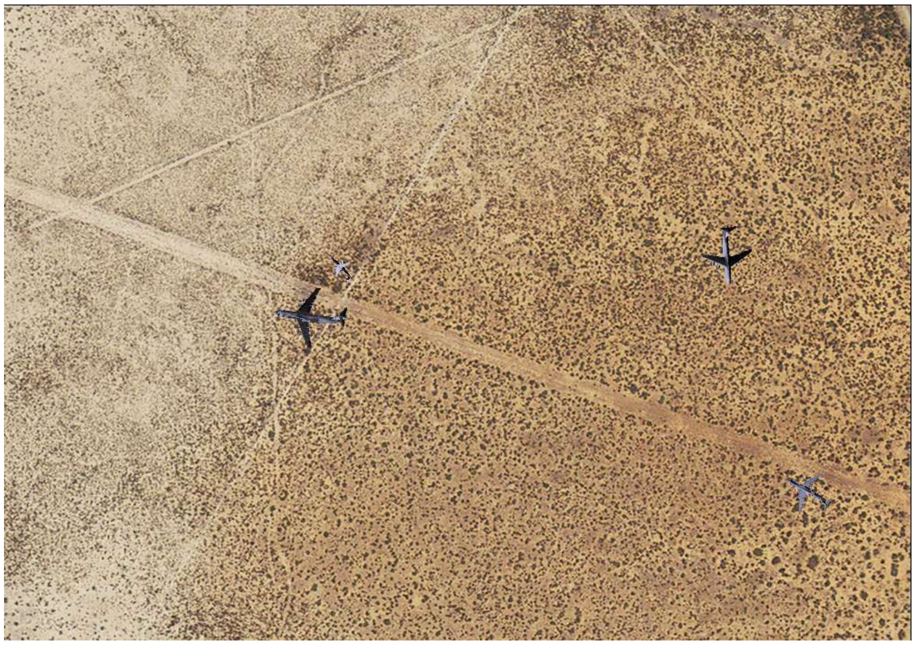
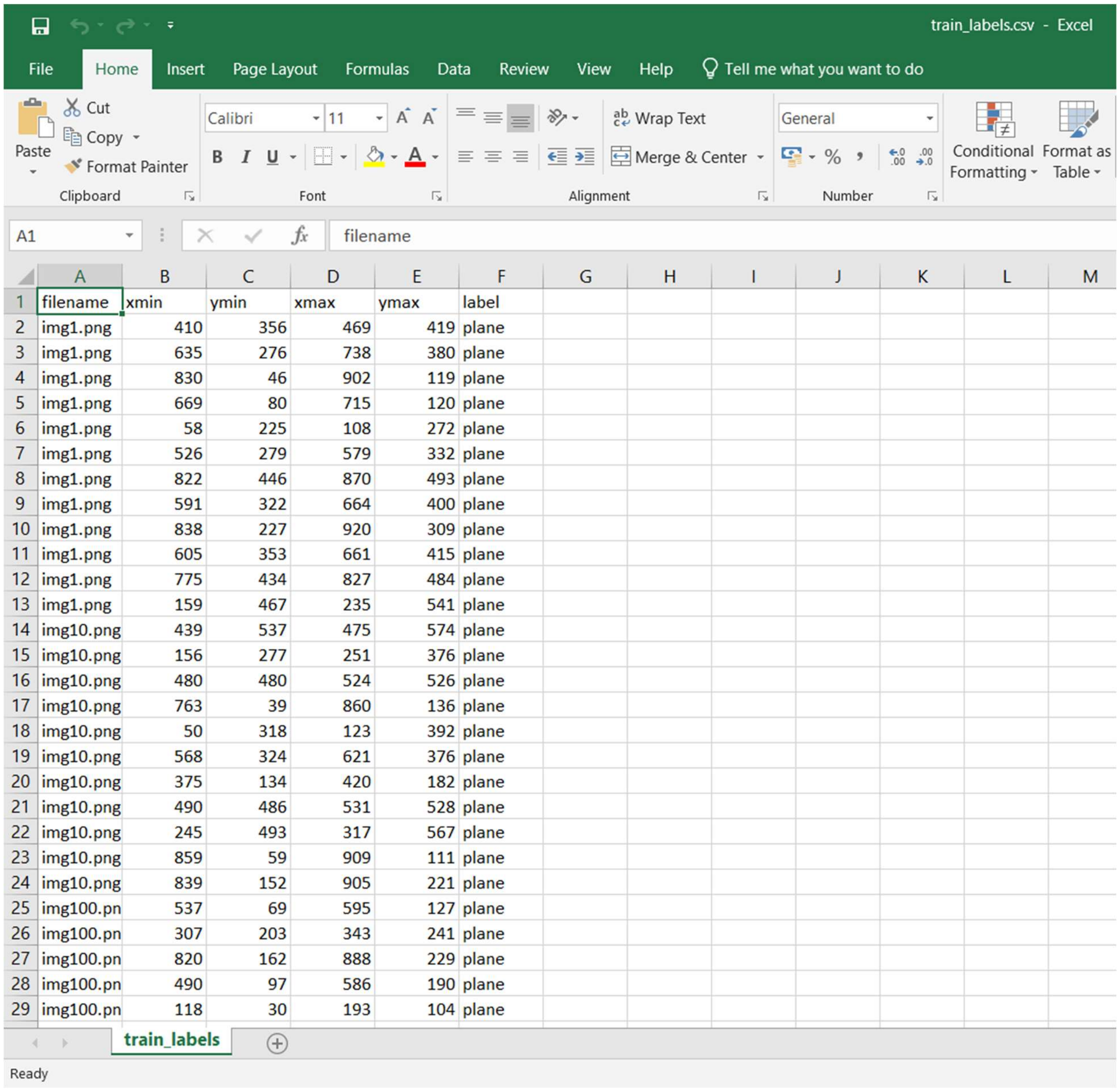


Figure 1- Example of a training image



	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

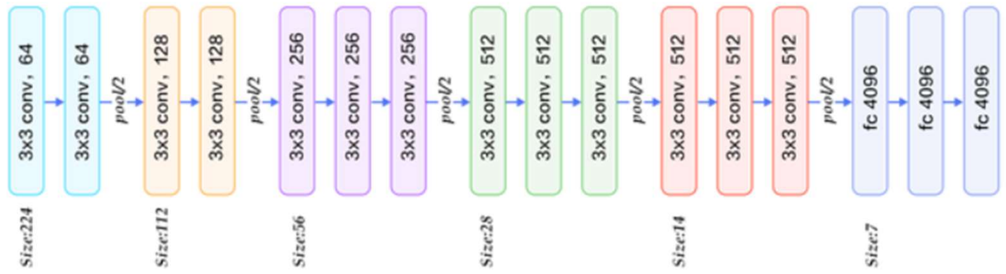
Figure 2- training data's csv file storing coordinate of aircrafts

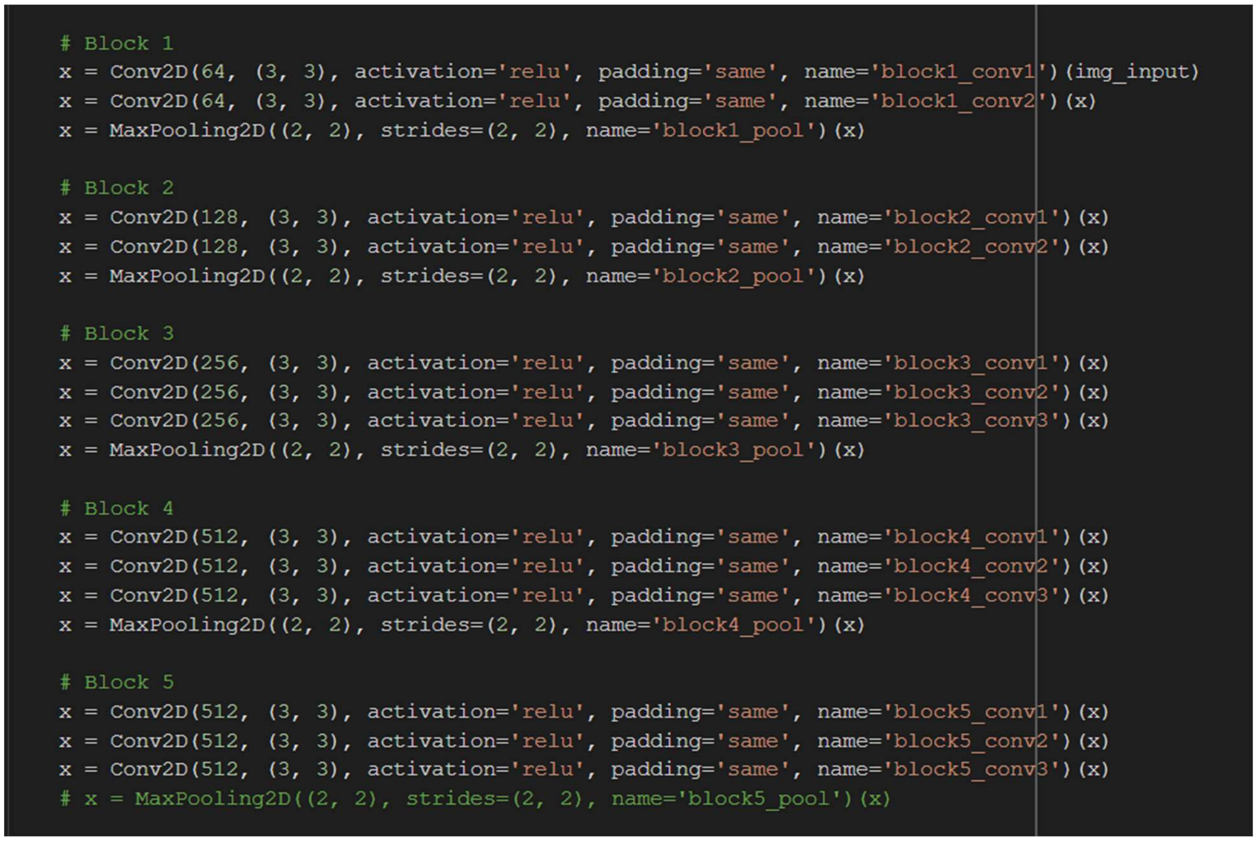
To this dataset 10 different augmentation was done which was Flipping, Scaling, Rotating, Shearing, Translating and combination of these. With this the number of training dataset was increased to 4400.

After this annotation.txt file was created containing the location of each training data, its name and min-max value of x, y to get the coordinate of aircrafts in those images.

BUILDING THE MODEL:

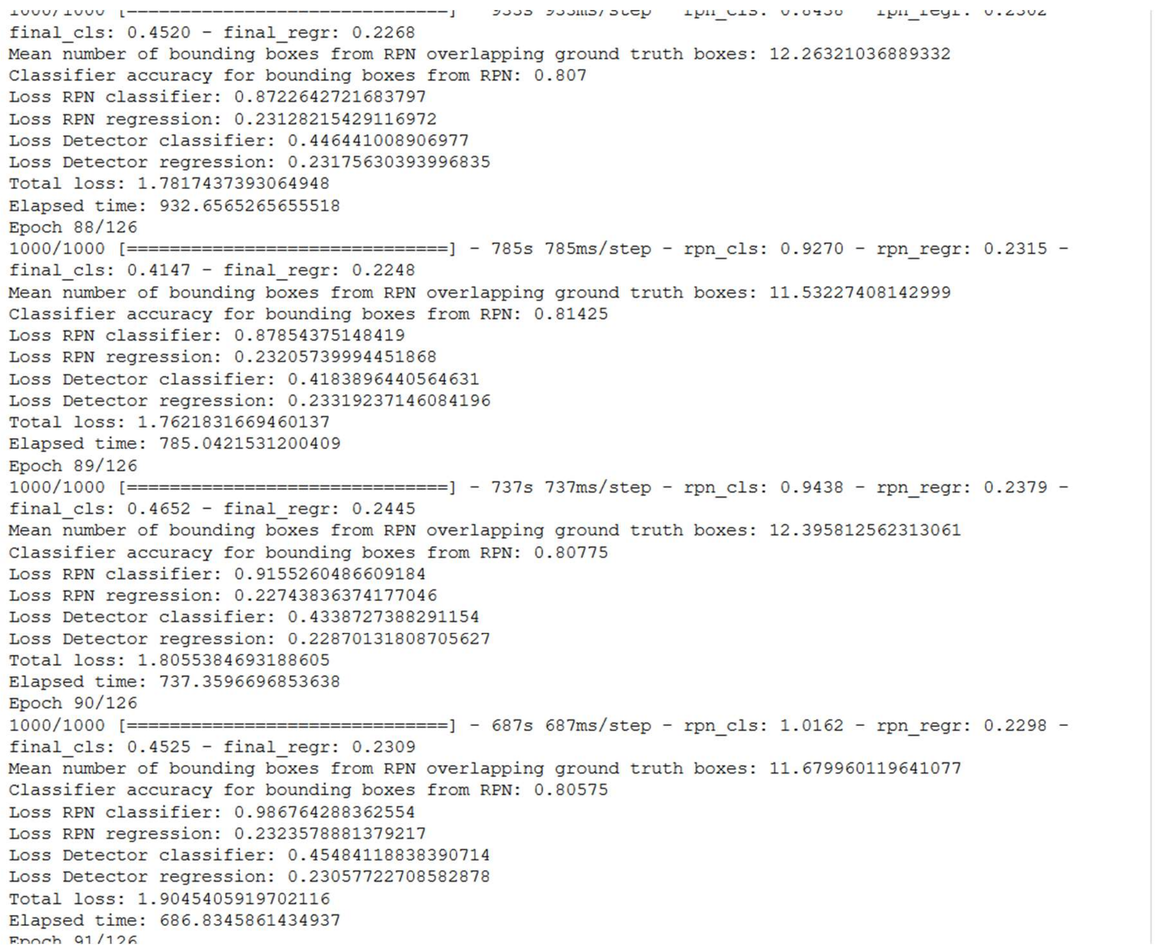
1. Parsing the data from annotation file: We first parse the data from annotation file returning list of file path, width, height, list (bounding boxes), a dictionary data type containing class name and count
2. Defining ROI Pooling Convolutional Layer: With the given pool size (pool size = 7 results in a 7x7 region) and number of regions of interest to be used. It returns a 3D tensor with shape (1, num_rois, channels, pool_size, pool_size)
3. VGG Network: The VGG architecture is rebuild





```
# Block 1
x = Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv1')(img_input)
x = Conv2D(64, (3, 3), activation='relu', padding='same', name='block1_conv2')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block1_pool')(x)

# Block 2
x = Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv1')(x)
x = Conv2D(128, (3, 3), activation='relu', padding='same', name='block2_conv2')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block2_pool')(x)

# Block 3
x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv1')(x)
x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv2')(x)
x = Conv2D(256, (3, 3), activation='relu', padding='same', name='block3_conv3')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block3_pool')(x)

# Block 4
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv1')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv2')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block4_conv3')(x)
x = MaxPooling2D((2, 2), strides=(2, 2), name='block4_pool')(x)

# Block 5
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv1')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv2')(x)
x = Conv2D(512, (3, 3), activation='relu', padding='same', name='block5_conv3')(x)
# x = MaxPooling2D((2, 2), strides=(2, 2), name='block5_pool')(x)
```

4. Creating Region proposal network layer: Passing through the feature map from base layer to a 3x3 512 channels convolutional layer keeping the padding same to preserve the feature map's size. The output of this is passed to two (1,1) convolutional layer to replace the fully connected layer. The base layer in this case is VGG. It returns classification for whether its an object and bounding boxes regression.
5. Classifier layer: The input given is list of regions of interest, with ordering (x, y, w, h) and returns the classifier layer output (softmax activation function) and regression layer output (linear activation function).
6. Intersection of union is calculated
7. Calculated the region proposal network for all anchors of all images and returns which bounding boxes are valid or not valid.
8. Then we generate the ground truth anchors: Yields the ground-truth anchors as Y (labels). Yield function is used to avoid overloading the memory as the size maybe quite large.
9. Then the loss function is defined for regression of bounding boxes, region proposal network classification.
10. Non-maximum suppression is carried out to accept only those bounding boxes whose values are greater than the threshold value
11. Finally, the region proposal network layer is converted to region of interest boxes and the coordinate for these boxes (on the feature map) is stored.
12. And finally, training was done of 1000 epochs on google colabs GPU runtime server, google drive was mounted which kept updating weights whenever the losses went down.

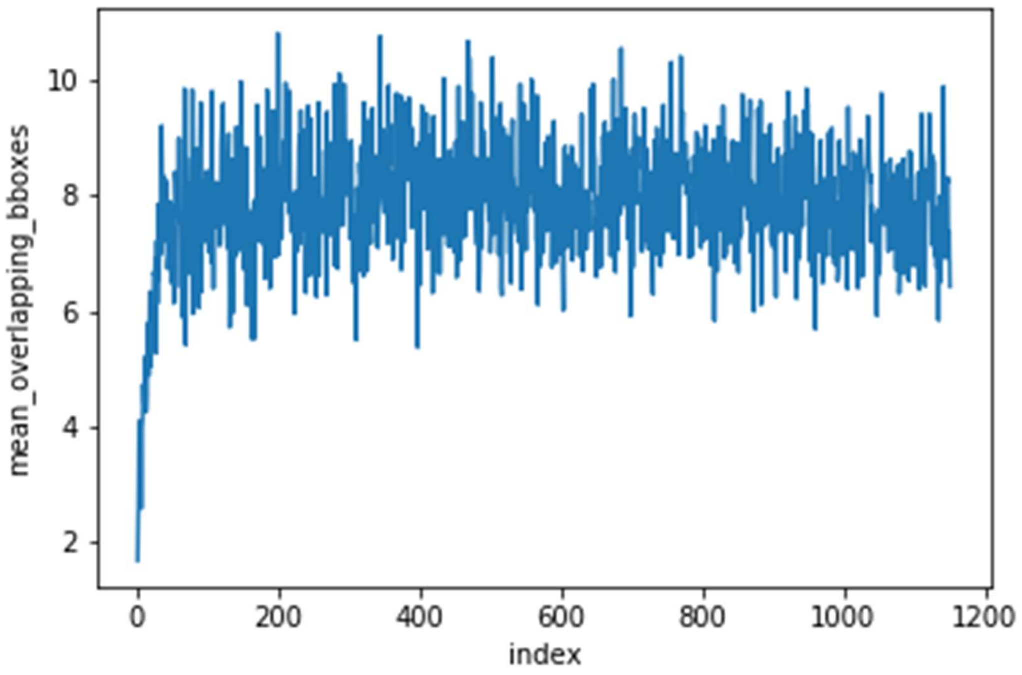


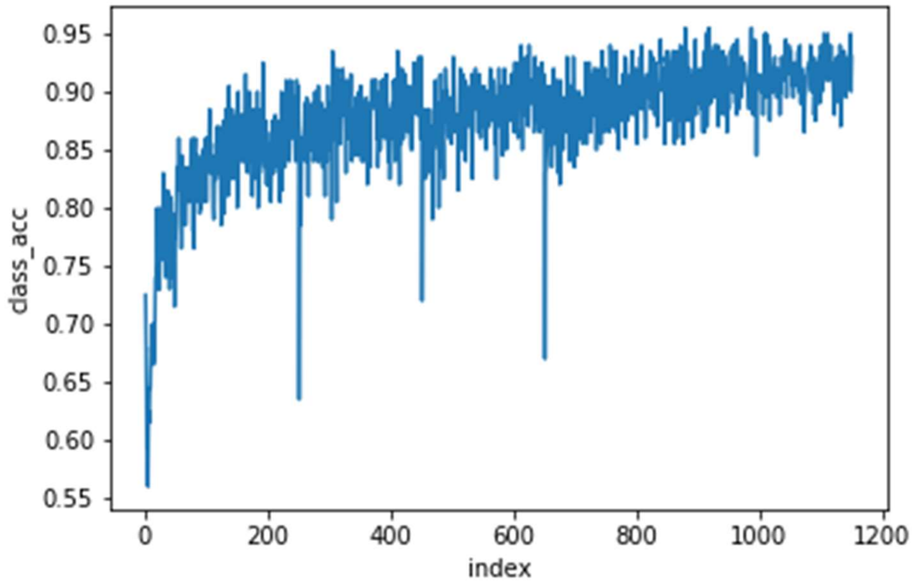
```
1000/1000 [==============================] - 933s 933ms/step - rpn_cls: 0.8436 - rpn_regr: 0.2302
final_cls: 0.4520 - final_regr: 0.2268
Mean number of bounding boxes from RPN overlapping ground truth boxes: 12.26321036889332
Classifier accuracy for bounding boxes from RPN: 0.807
Loss RPN classifier: 0.8722642721683797
Loss RPN regression: 0.23128215429116972
Loss Detector classifier: 0.446441008906977
Loss Detector regression: 0.23175630393996835
Total loss: 1.7817437393064948
Elapsed time: 932.6565265655518
Epoch 88/126
1000/1000 [==============================] - 785s 785ms/step - rpn_cls: 0.9270 - rpn_regr: 0.2315 -
final_cls: 0.4147 - final_regr: 0.2248
Mean number of bounding boxes from RPN overlapping ground truth boxes: 11.53227408142999
Classifier accuracy for bounding boxes from RPN: 0.81425
Loss RPN classifier: 0.87854375148419
Loss RPN regression: 0.23205739994451868
Loss Detector classifier: 0.4183896440564631
Loss Detector regression: 0.23319237146084196
Total loss: 1.7621831669460137
Elapsed time: 785.0421531200409
Epoch 89/126
1000/1000 [==============================] - 737s 737ms/step - rpn_cls: 0.9438 - rpn_regr: 0.2379 -
final_cls: 0.4652 - final_regr: 0.2445
Mean number of bounding boxes from RPN overlapping ground truth boxes: 12.395812562313061
Classifier accuracy for bounding boxes from RPN: 0.80775
Loss RPN classifier: 0.9155260486609184
Loss RPN regression: 0.22743836374177046
Loss Detector classifier: 0.4338727388291154
Loss Detector regression: 0.22870131808705627
Total loss: 1.8055384693188605
Elapsed time: 737.3596696853638
Epoch 90/126
1000/1000 [==============================] - 687s 687ms/step - rpn_cls: 1.0162 - rpn_regr: 0.2298 -
final_cls: 0.4525 - final_regr: 0.2309
Mean number of bounding boxes from RPN overlapping ground truth boxes: 11.679960119641077
Classifier accuracy for bounding boxes from RPN: 0.80575
Loss RPN classifier: 0.986764288362554
Loss RPN regression: 0.2323578881379217
Loss Detector classifier: 0.45484118838390714
Loss Detector regression: 0.23057722708582878
Total loss: 1.9045405919702116
Elapsed time: 686.8345861434937
Epoch 91/126
```

Figure 3- Screenshot of verbose during training

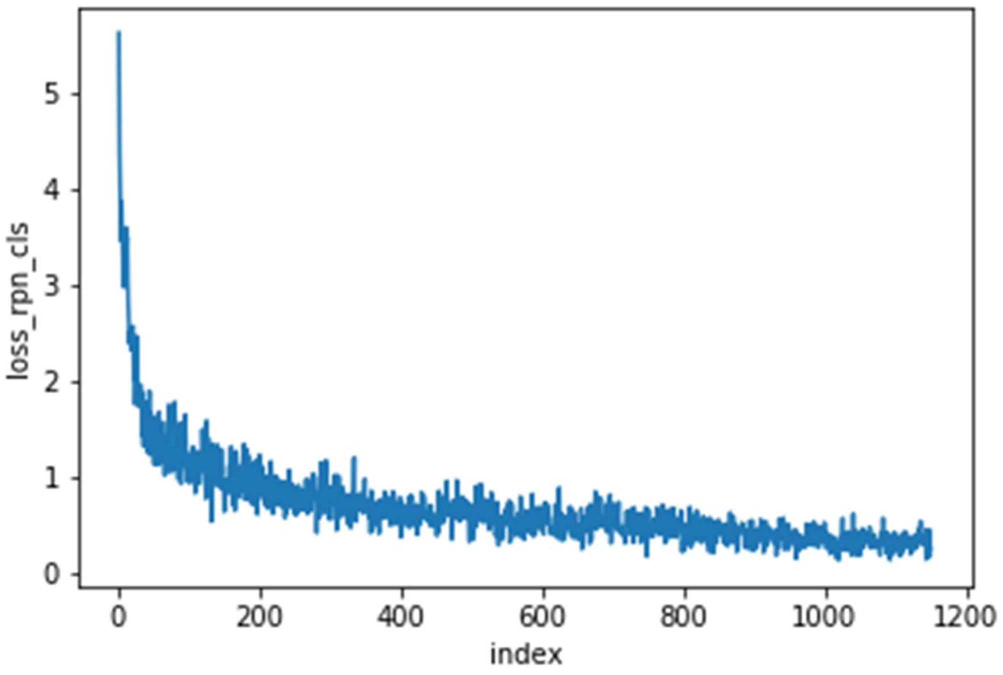
RESULTS:

For assessment, mean overlapping boxes and class accuracy shows an increasing trend. For quantifying the result, there are 2 separate process carried out classification and region proposal. The RPN gives out classification (to check whether box contains object or not) and regression (to check for accuracy of box in terms of IoU value with ground truth). And also, an account of total loss was plotted.

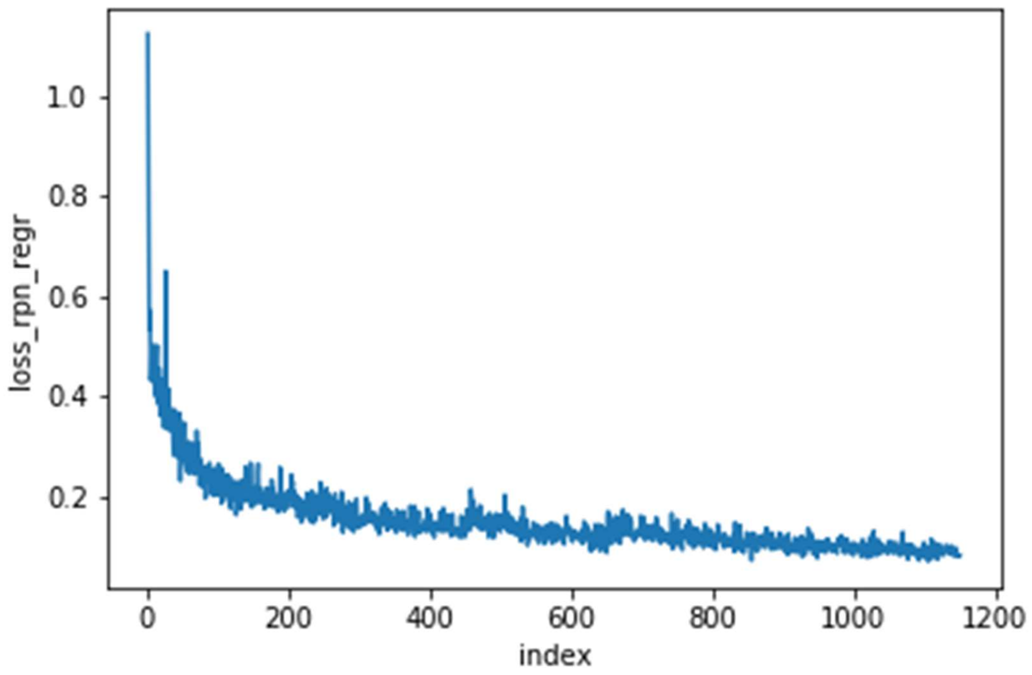


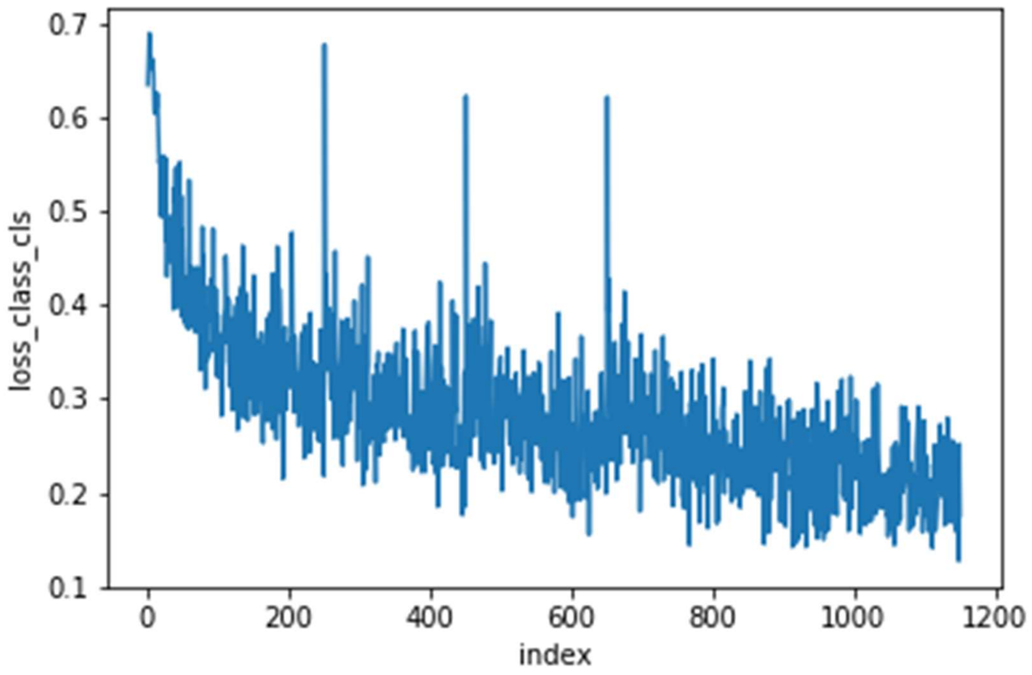


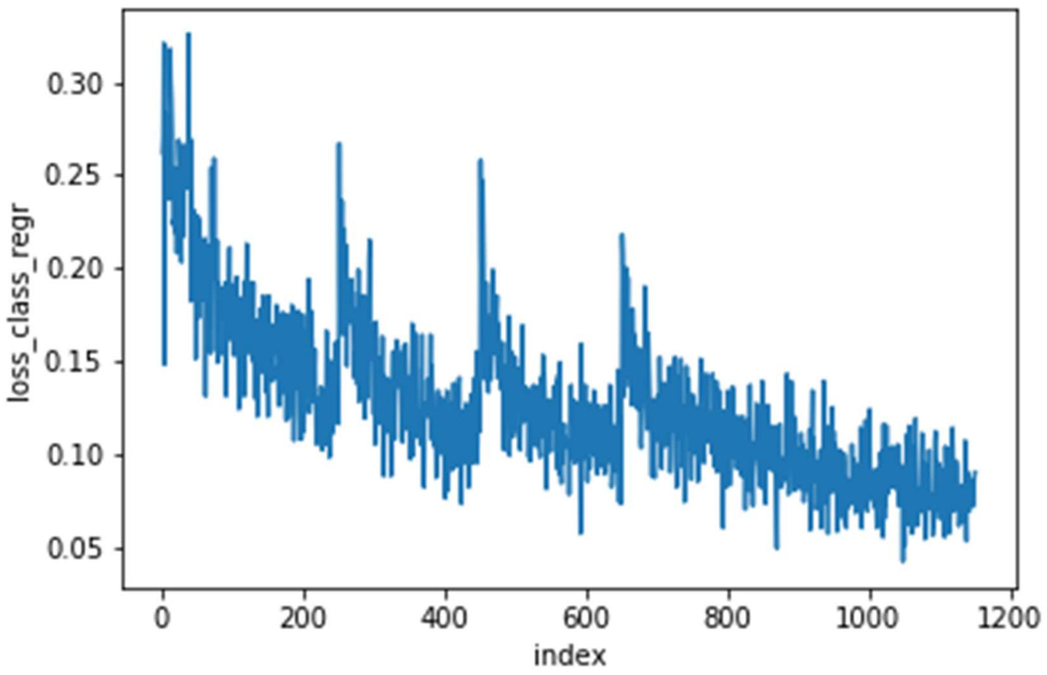
class_acc
index

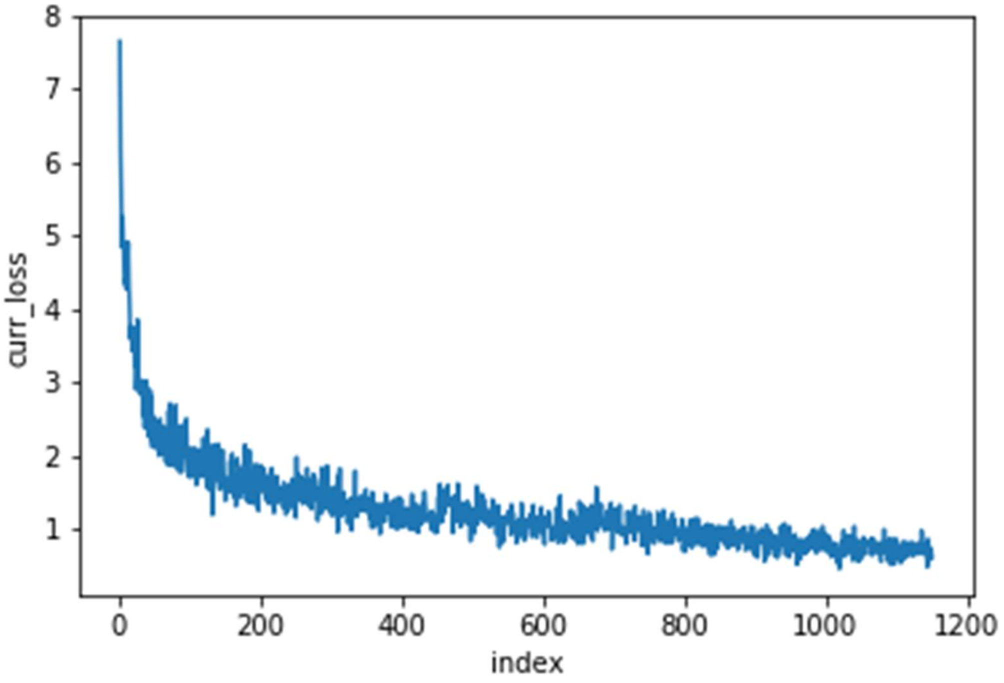


loss_rpn_cls
index


loss_rpn_regr
1.0
0.8
0.6
0.4
0.2
0
200
400
600
800
1000
1200
index


loss_class_cls
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
200
400
600
800
1000
1200
index


loss_class_regr
0.30
0.25
0.20
0.15
0.10
0.05
0
200
400
600
800
1000
1200
index


curr_loss
8
7
6
5
4
3
2
1
0
200
400
600
800
1000
1200
index

LIMITATIONS & IMPROVEMENTS:

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.

Also due to limited availability of computing resources 1000 epochs was done. Increasing the number of iterations would definitely increase accuracy of the model.

FUTURE POSSIBILITIES:

For the usage of this model, we created a UI 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.

BILBLIOGRAPHY:

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. Faster R-CNN (object detection) implemented by Keras for custom data from Google's Open Images Dataset V4 https://towardsdatascience.com/faster-r-cnn-object-detection-implemented-by-keras-for-custom-data-from-googles-open-images-125f62b9141a
3. 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
4. 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
5. 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.
6. 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.
7. Xu T.B., Cheng G.L., Yang J. Fast Aircraft Detection Using End-to-End Fully Convolutional Network
8. Aircraft Detection by Deep Convolutional Neural Networks January 2014IPSJ Transactions on Computer Vision and Applications 7:10-17 DOI: 10.2197/ipsjtcva.7.10
9. 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.