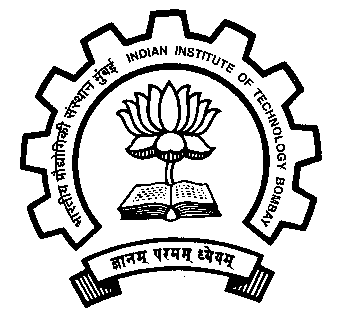


Object detection(aircraft) & Bounding box problem

B. Tech Project - 2



By- Ujjwal & DavidJ

Supervisor- Prof. J. Indu

Department of Civil Engineering

IIT Bombay

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.

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

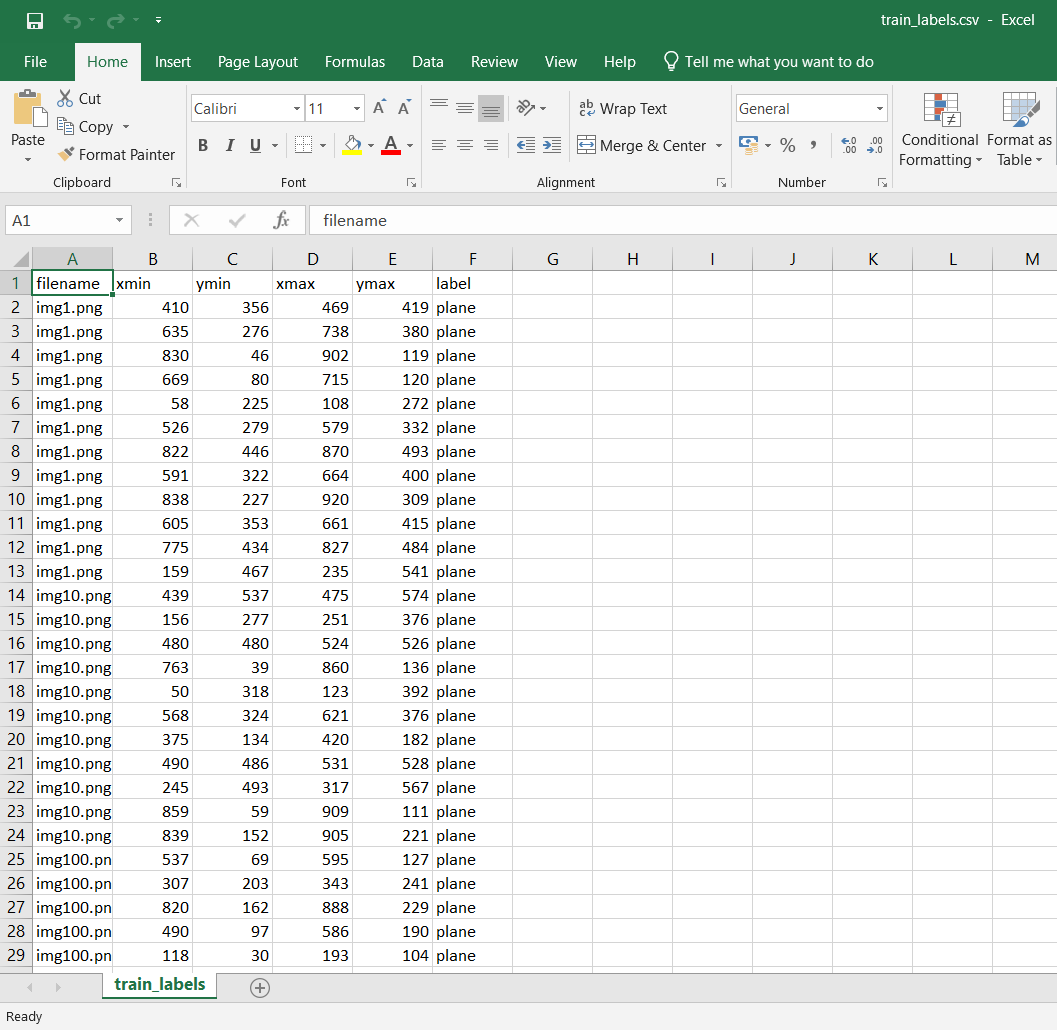
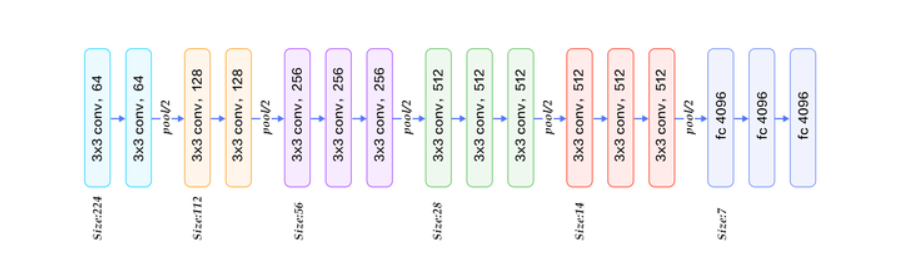


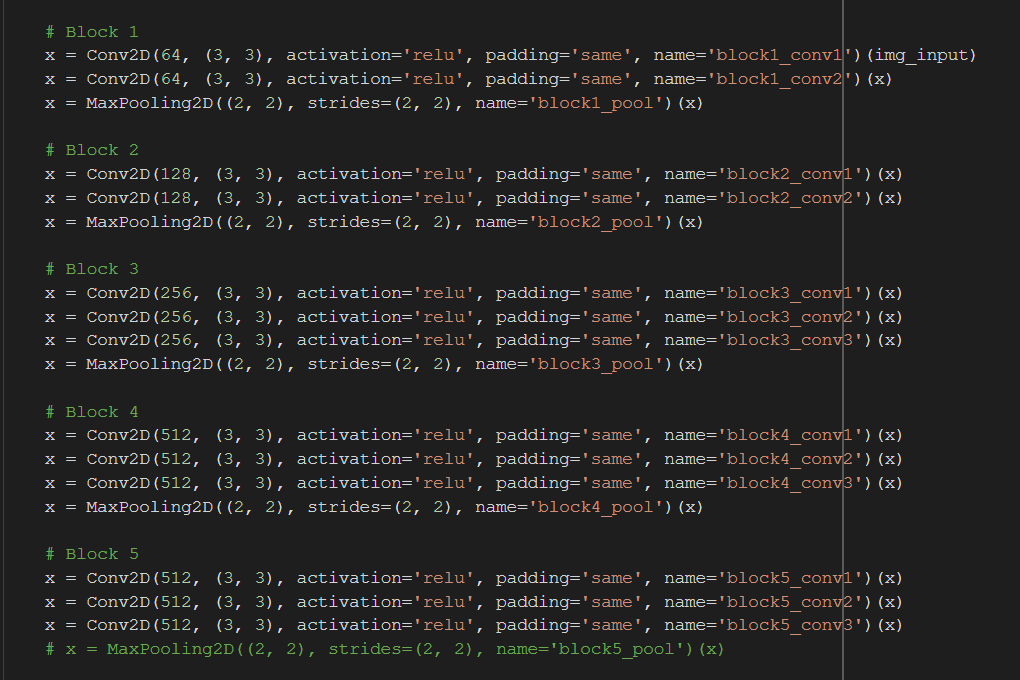
Figure 2- training data's csv file storing coordinate of aircrafts

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



1. 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.
2. 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).
3. Intersection of union is calculated
4. Calculated the region proposal network for all anchors of all images and returns which bounding boxes are valid or not valid.
5. 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.
6. Then the loss function is defined for regression of bounding boxes, region proposal network classification.
7. Non-maximum suppression is carried out to accept only those bounding boxes whose values are greater than the threshold value
8. 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.
9. 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.

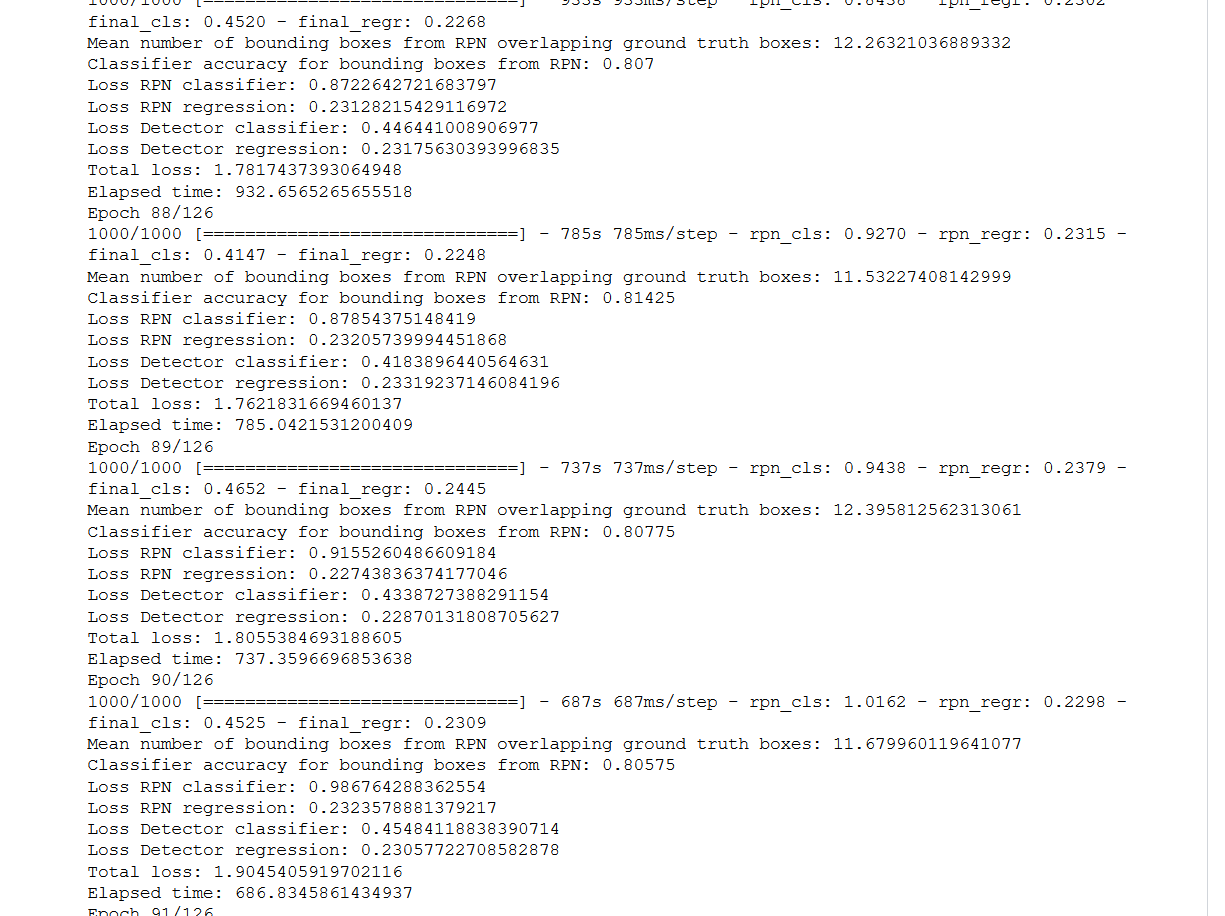
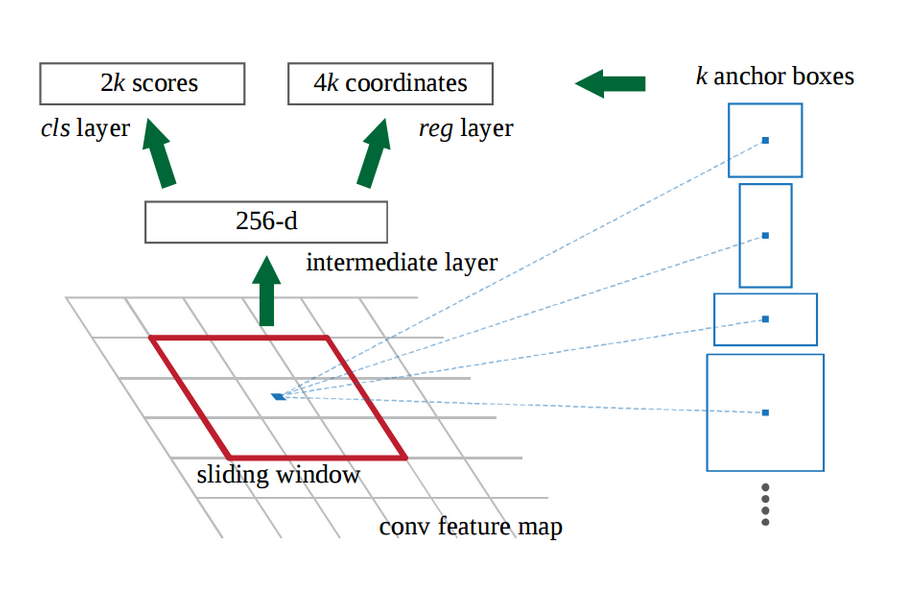
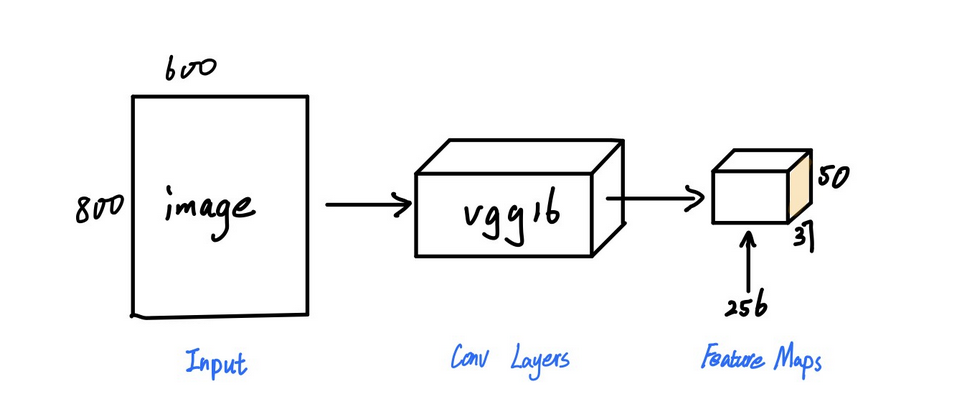


Figure 3- Screenshot of verbose during training

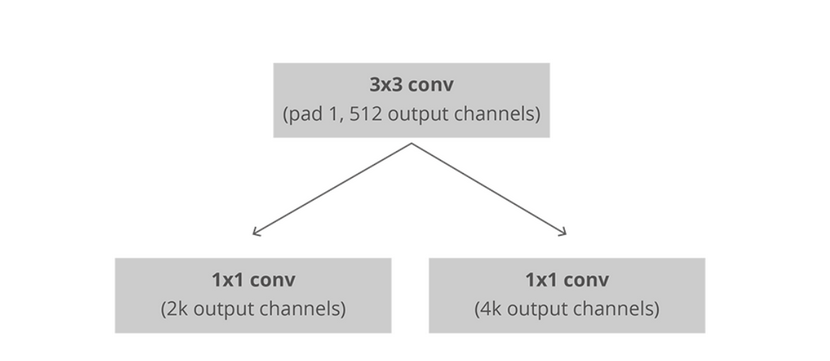
Brief explanation of Faster- RCNN:



Faster R-CNN (frcnn for short) makes further progress than Fast R-CNN. Search selective process is replaced by Region Proposal Network (RPN). As the name revealed, RPN is a network to propose regions. For instance, after getting the output feature map from a pre-trained model (VGG-16), if the input image has 600x800x3 dimensions, the output feature map would be 37x50x256 dimensions.



Each point in 37x50 is considered as an anchor. We need to define specific ratios and sizes for each anchor (1:1, 1:2, 2:1 for three ratios and 128², 256², 512² for three sizes in the original image). Next, RPN is connected to a Conv layer with 3x3 filters, 1 padding, 512 output channels. The output is connected to two 1x1 convolutional layer for classification and box-regression (Note that the classification here is to determine if the box is an object or not).



For training, we take all the anchors and put them into two different categories. Those that overlap a ground-truth object with an Intersection over Union (IoU) bigger than 0.5 are considered “foreground” and those that don’t overlap any ground truth object or have less than 0.1 IoU with ground-truth objects are considered “background”.

In this case, every anchor has 3x3 = 9 corresponding boxes in the original image, which means there are 37x50x9 = 16650 boxes in the original image. We just choose 256 of these 16650 boxes as a mini batch which contains 128 foregrounds (positive) and 128 backgrounds (negative). At the same time, non-maximum suppression is applied to make sure there is no overlapping for the proposed regions.

RPN is finished after going through the above steps. Then we go to the second stage of frcnn. Similar to Fast R-CNN, ROI pooling is used for these proposed regions (ROIs). The output is 7x7x512. Then, we flatten this layer with some fully connected layers. The final step is a softmax function for classification and linear regression to fix the boxes’ location.

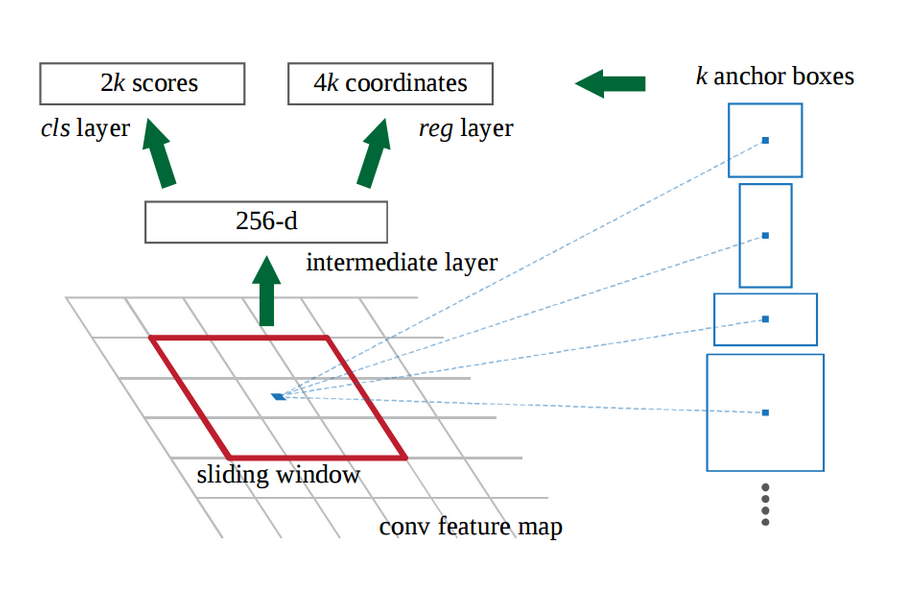
Brief explanation of Faster- RCNN (2):

The main insight of Faster R-CNN was to replace the slow selective search algorithm of Fast RCNN with a fast-neural net. Specifically, it introduced the region proposal network (RPN).

Here’s how the RPN worked:

* At the last layer of an initial CNN, a 3x3 sliding window moves across the feature map and maps it to a lower dimension (e.g. 256-d)
* For each sliding-window location, it generates multiple possible regions based on k fixed-ratio anchor boxes (default bounding boxes)
* Each region proposal consists of a) an “objectness” score for that region and b) 4 coordinates representing the bounding box of the region

In other words, we look at each location in our last feature map and consider k different boxes centered around it: a tall box, a wide box, a large box, etc. For each of those boxes, we output whether or not we think it contains an object, and what the coordinates for that box are. This is what it looks like at one sliding window location:



The 2k scores represent the softmax probability of each of the k bounding boxes being on “object.” Notice that although the RPN outputs bounding box coordinates, it does not try to classify any potential objects: its sole job is still proposing object regions. If an anchor box has an “objectness” score above a certain threshold, that box’s coordinates get passed forward as a region proposal.

Once we have our region proposals, we feed them straight into what is essentially a Fast R-CNN. We add a pooling layer, some fully-connected layers, and finally a softmax classification layer and bounding box regressor. In a sense, Faster R-CNN = RPN + Fast R-CNN.

BUILDING THE MODEL (2)

Prepare training data and training labels

The input data is from annotation.txt file which contains a bunch of images with their bounding boxes information. We need to use RPN method to create proposed bboxes.

* Arguments in this function  
  **all\_img\_data**: list(filepath, width, height, list(bboxes))  
  **C**: config  
  **img\_length\_calc\_function**: function to calculate final layer’s feature map (of base model) size according to input image size  
  **mode**: ‘train’ or ‘test’; ‘train’ mode need augmentation
* Returns value in this function  
  **x\_img**: image data after resized and scaling (smallest size = 300px)  
  **Y**: [y\_rpn\_cls, y\_rpn\_regr]  
  **img\_data\_aug**: augmented image data (original image with augmentation)  
  **debug\_img**: show image for debug  
  **num\_pos**: show number of positive anchors for debug

**Calculate rpn for each image (calc\_rpn)**

If feature map has shape 18x25=450 and anchor sizes=9, there are 450x9=4050 potential anchors. The initial status for each anchor is ‘negative’. Then, we set the anchor to positive if the IOU is >0.7. If the IOU is >0.3 and <0.7, it is ambiguous and not included in the objective. One issue is that the RPN has many more negative than positive regions, so we turn off some of the negative regions. We also limit the total number of positive regions and negative regions to 256. y\_is\_box\_valid represents if this anchor has an object. y\_rpn\_overlap represents if this anchor overlaps with the ground-truth bounding box.

For ‘positive’ anchor, y\_is\_box\_valid =1, y\_rpn\_overlap =1.   
For ‘neutral’ anchor, y\_is\_box\_valid =0, y\_rpn\_overlap =0.   
For ‘negative’ anchor, y\_is\_box\_valid =1, y\_rpn\_overlap =0.

* Arguments in this function  
  **C**: config  
  **img\_data**: augmented image data  
  **width**: original image width (e.g. 600)  
  **height**: original image height (e.g. 800)  
  **resized\_width**: resized image width according to C.im\_size (e.g. 300)  
  **resized\_height**: resized image height according to C.im\_size (e.g. 400)  
  **img\_length\_calc\_function**: function to calculate final layer’s feature map (of base model) size according to input image size
* Returns value in this function  
  **y\_rpn\_cls**: list(num\_bboxes, y\_is\_box\_valid + y\_rpn\_overlap)  
  **y\_is\_box\_valid**: 0 or 1 (0 means the box is invalid, 1 means the box is valid)  
  **y\_rpn\_overlap**: 0 or 1 (0 means the box is not an object, 1 means the box is an object)  
  **y\_rpn\_regr**: list(num\_bboxes, 4\*y\_rpn\_overlap + y\_rpn\_regr)  
  **y\_rpn\_regr**: x1,y1,x2,y2 bunding boxes coordinates

The shape of y\_rpn\_cls is (1, 18, 25, 18). 18x25 is feature map size. Each point in feature map has 9 anchors, and each anchor has 2 values for y\_is\_box\_valid and y\_rpn\_overlap respectively. So the fourth shape 18 is from 9x2.

The shape of y\_rpn\_regr is (1, 18, 25, 72). 18x25 is feature map size. Each point in feature map has 9 anchors and each anchor has 4 values for tx, ty, tw and th respectively. Note that these 4 value has their own y\_is\_box\_valid and y\_rpn\_overlap. So the fourth shape 72 is from 9x4x2.

## Calculate region of interest from RPN (rpn\_to\_roi)

* Arguments in this function (num\_anchors = 9)  
  **rpn\_layer**: output layer for rpn classification   
  shape (1, feature\_map.height, feature\_map.width, num\_anchors)  
  Might be (1, 18, 25, 9) if resized image is 400 width and 300  
  **regr\_layer**: output layer for rpn regression  
  shape (1, feature\_map.height, feature\_map.width, num\_anchors\*4)  
  Might be (1, 18, 25, 36) if resized image is 400 width and 300  
  **C**: config  
  **use\_regr**: Wether to use bboxes regression in rpn  
  **max\_boxes**: max bboxes number for non-max-suppression (NMS)  
  **overlap\_thresh**: If iou in NMS is larger than this threshold, drop the box
* Returns value in this function  
  **result**: boxes from non-max-suppression (shape=(300, 4))  
  **boxes**: coordinates for bboxes (on the feature map)

For 4050 anchors from above step, we need to extract max\_boxes (300 in the code) number of boxes as the region of interests and pass them to the classifier layer (second stage of frcnn). In the function, we first delete the boxes that overstep the original image. Then, we use non-max-suppression with 0.7 threshold value.

## RoIPooling layer and Classifier layer (RoiPoolingConv, classifier\_layer)

RoIPooling layer is the function to process the roi to a specific size output by max pooling. Every input roi is divided into some sub-cells, and we applied max pooling to each sub-cell. The number of sub-cells should be the dimension of the output shape.

Classifier layer is the final layer of the whole model and just behind the RoIPooling layer. It’s used to predict the class name for each input anchor and the regression of their bounding box.

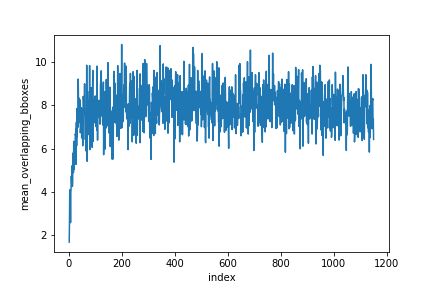
* Arguments in this function  
  base\_layers: vgg  
  input\_rois: `(1,num\_rois,4)` list of rois, with ordering (x,y,w,h)  
  num\_rois: number of rois to be processed in one time (4 in here)
* Returns value in this function  
  list(out\_class, out\_regr)  
  out\_class: classifier layer output  
  out\_regr: regression layer output

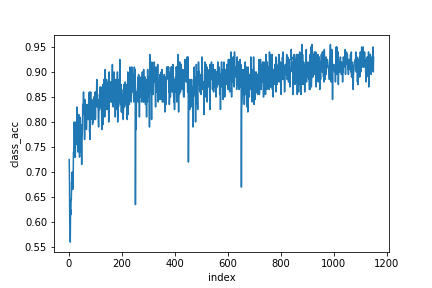
First, the pooling layer is flattened.   
Then, it’s followed with two fully connected layer and 0.5 dropout.   
Finally, there are two output layers.  
# out\_class: softmax activation function for classifying the class name of the object  
# out\_regr: linear activation function for bboxes coordinates regression

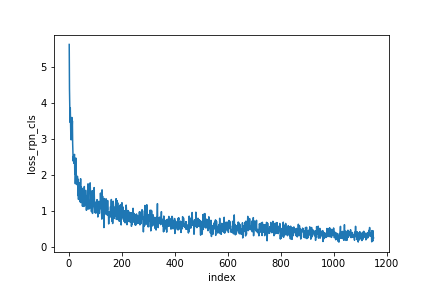
RESULTS:

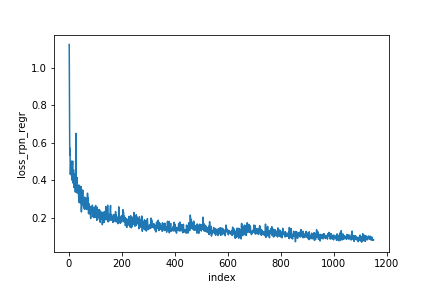
There are two loss functions we applied to both the RPN model and Classifier model. As we mentioned before, RPN model has two output. One is for classifying whether it’s an object and the other one is for bounding boxes’ coordinates regression.

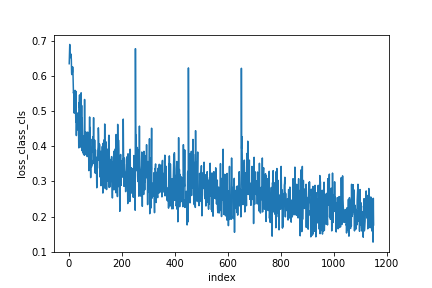
The similar learning process is shown in Classifier model. Compared with the two plots for bboxes regression, they show a similar tendency and even similar loss value. Compared with two plots for classifying, we can see that predicting objectness is easier than predicting the class name of a bbox.

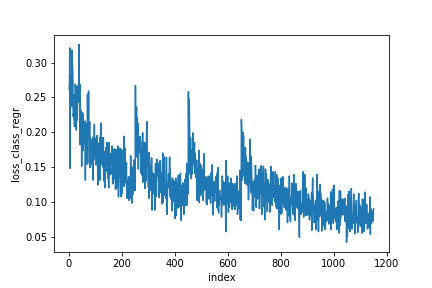


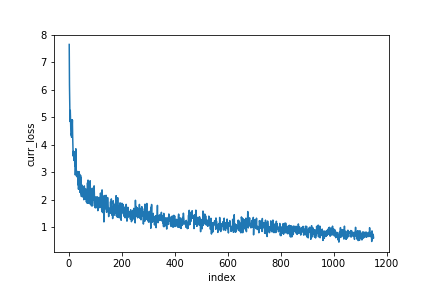












LIMITATIONS & IMPROVEMENTS:

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. 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