CMPT 412 Project 3 Report

Ritika Goyal (301401516)

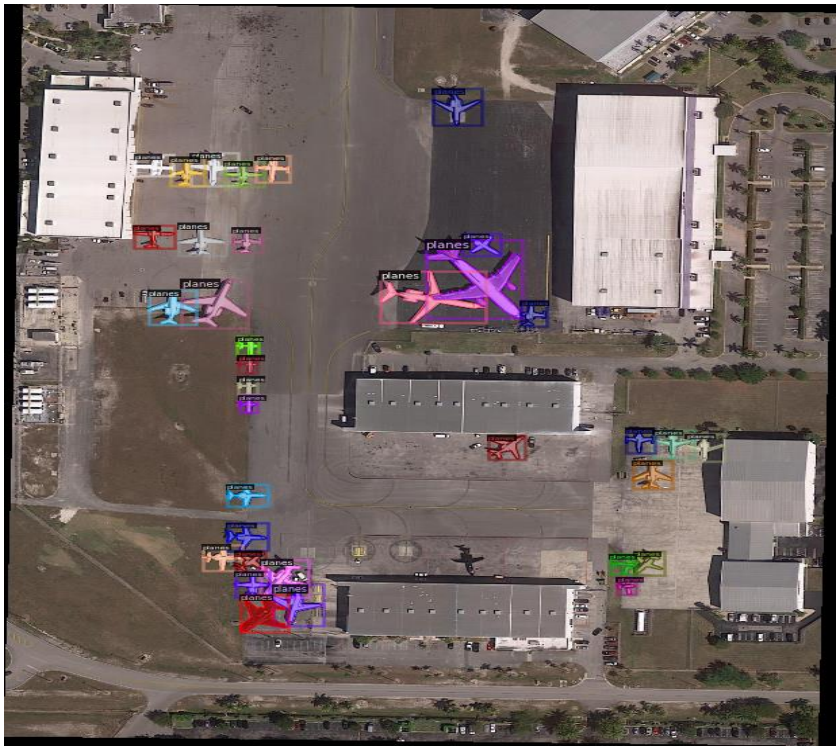
Project 3: Object Detection, Semantic Segmentation, and Instance Segmentation

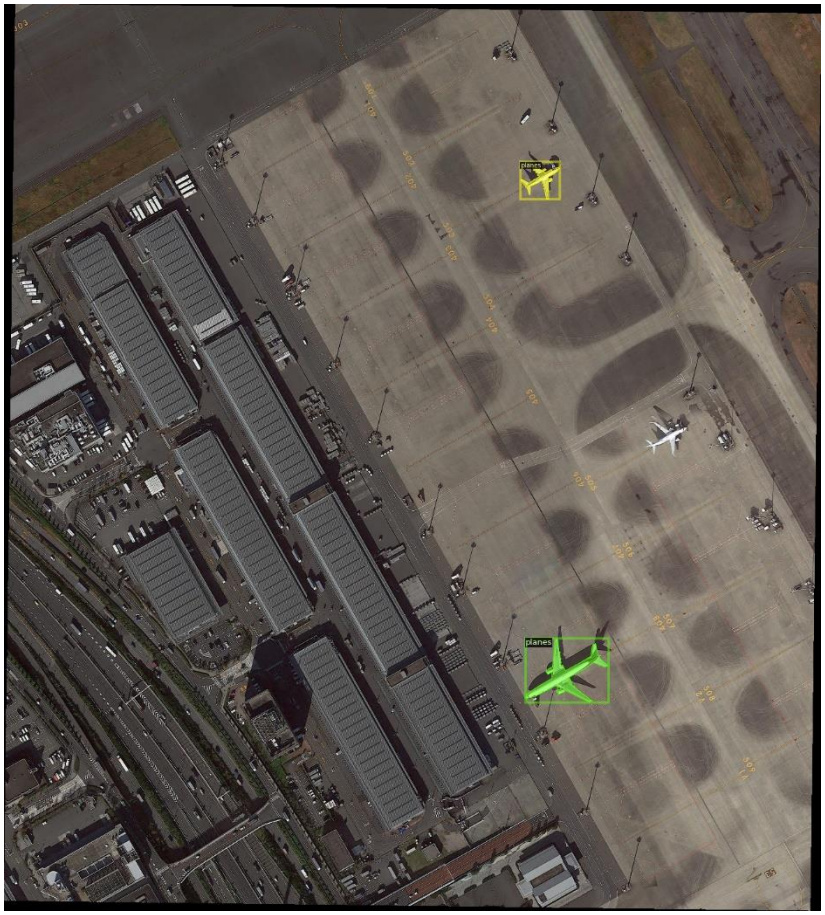
I am using two late days for assignment3

(Kaggle submission under name RitikaGoyal)

Part1: Object Detection

In this section, first I loaded the data from train.json and split it into training and validation set with 168 items in training and 30 items in validation. For the test set, the annotations were kept empty. I cached the data which saved time for loading it as it was just loaded once. Then I visualised the three random images from training_set which are shown below:




planes
planes

List of configs and modifications used:

These are the configs that I used for training:

```
[13] '''
     # Set the configs for the detection part in here.
     # TODO: approx 15 lines
     '''
     cfg = get_cfg()
     cfg.OUTPUT_DIR = "{}/output/".format(BASE_DIR)

     cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml"))
     cfg.DATASETS.TRAIN = ("data_detection_train")
     cfg.DATASETS.TEST = ()
     cfg.DATALOADER.NUM_WORKERS = 2
     # cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/mask_rcnn_R_50_FPN_3x.yaml")
     cfg.SOLVER.IMS_PER_BATCH = 2
     cfg.SOLVER.BASE_LR = 0.00025  # pick a good LR
     cfg.SOLVER.MAX_ITER = 1000 #500
     cfg.SOLVER.STEPS = []        # do not decay learning rate
     cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512   # faster, and good enough for this toy dataset (default: 51
     cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1  # only has one class (ballon). (see https://detectron2.readthedocs.io/t
     # NOTE: this config means the number of classes, but a few popular unofficial tutorials incorrect uses num_c

     os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
```

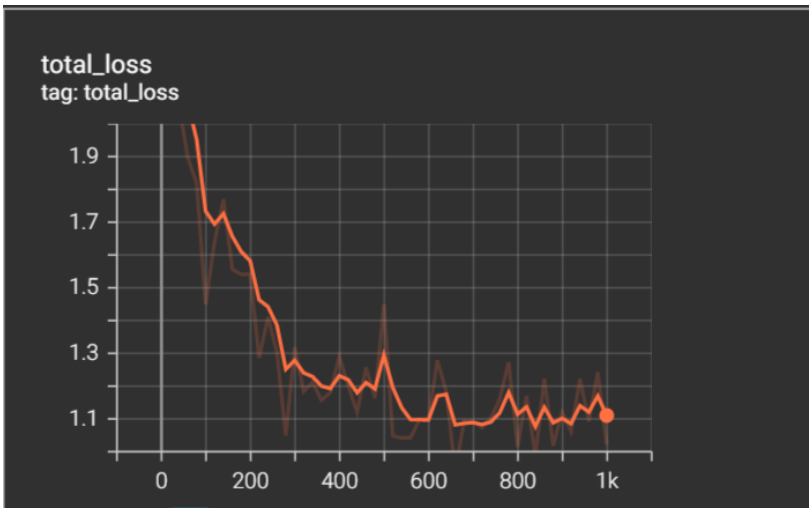
Explanation:

Initially, I tested different learning rates to train the model. I tried 0.001, 0.0001 and 0.0005 which had similar issues that they started great for first few iterations and then converged really fast, and loss fluctuated. Choosing 0.00025 as learning rate improved the accuracy as this learning rate is smaller than others that I tried and large learning rate does not converge to local minima.

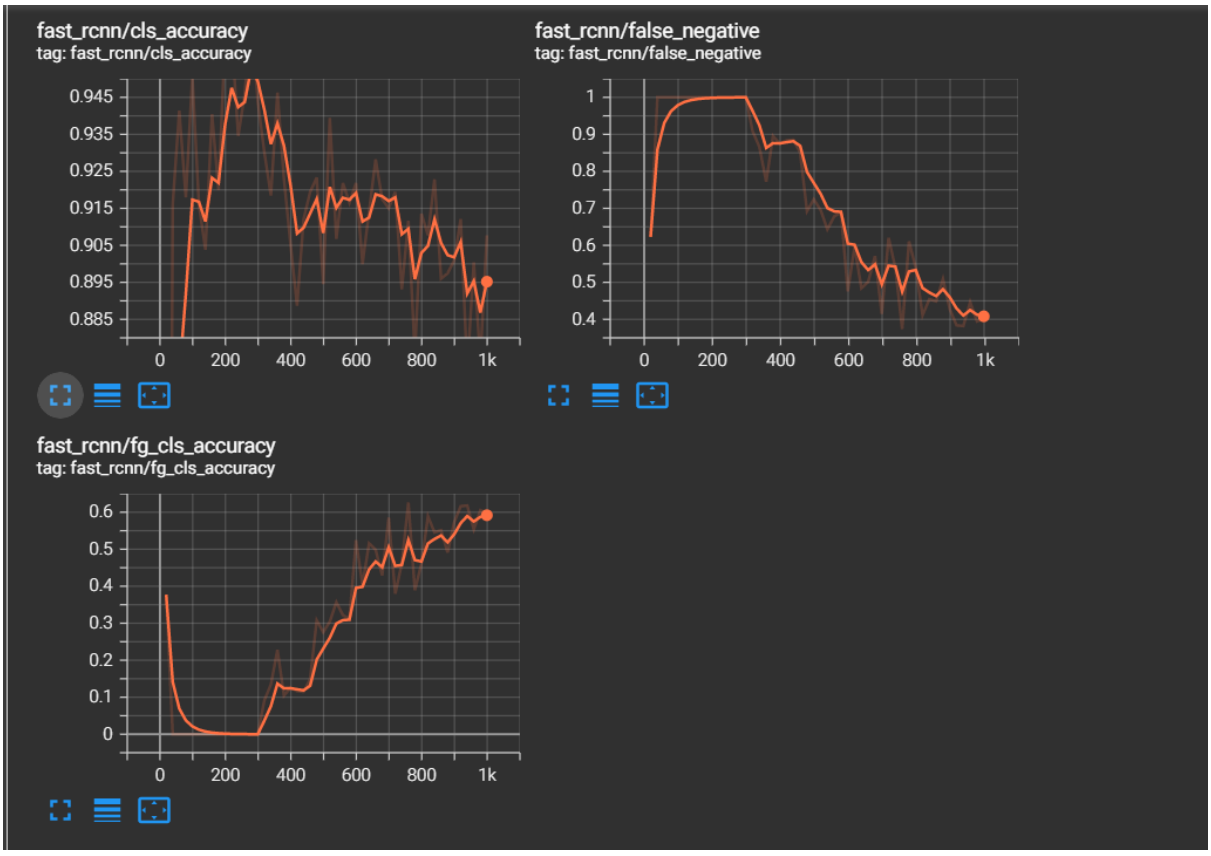
I used 1000 Max_ITER. The initial given max_iter was 500 but loss was still decreasing, So different values of max_iter were tried until loss was stable.

The model that I used was faster_rcnn_x_101_32x8d_FPN_3x.yaml because it had higher train mem that is 6.7 GB and it converged better as compared to faster_rcnn_R_101_FPN_3x.yaml and its box AP was also slightly higher. 2

Plot for total training loss:

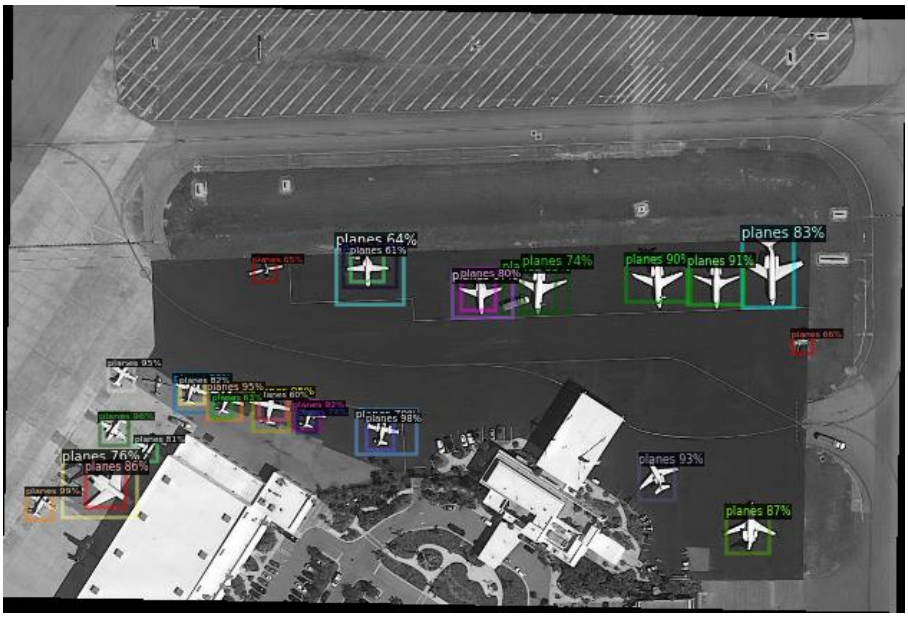


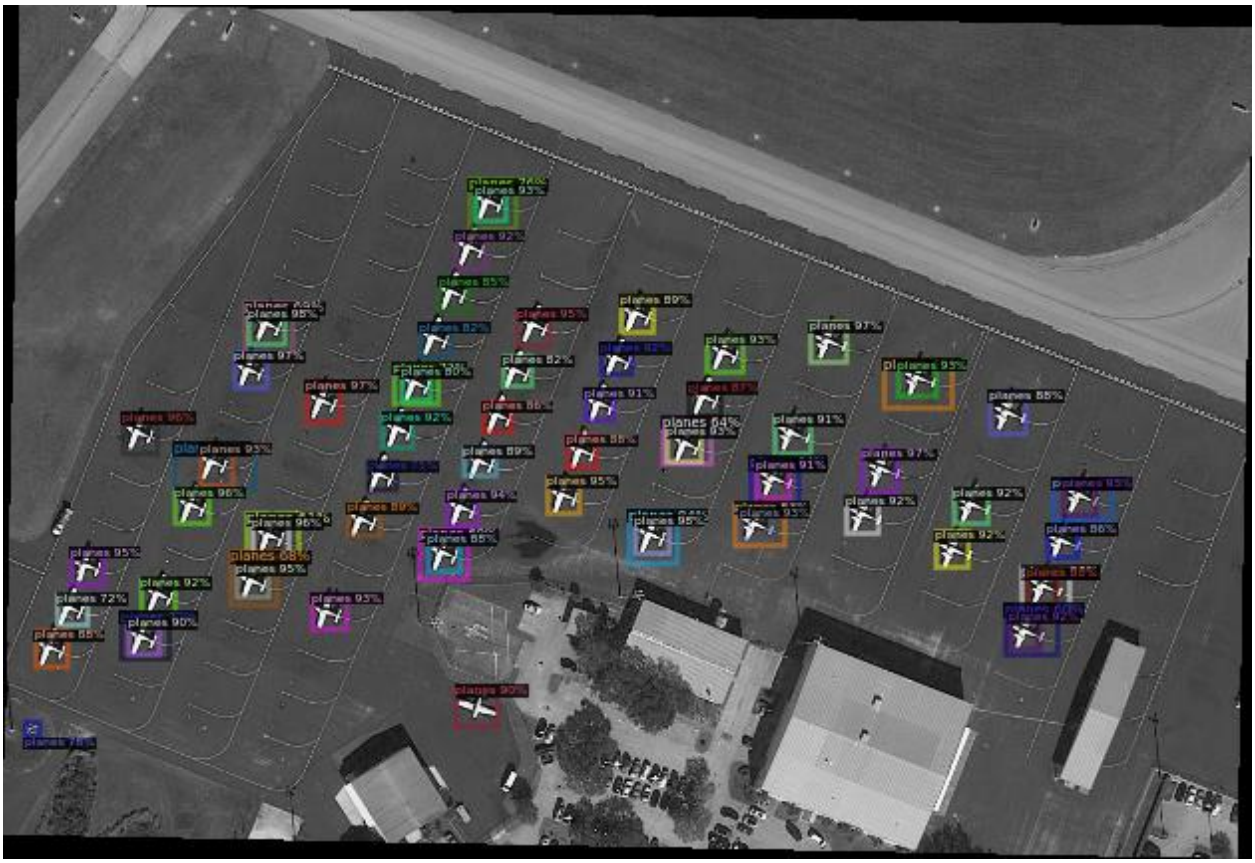
Plot for accuracy:



Visualised samples from test set and predicted results:

The visualised images using the above-mentioned configs and shown below:

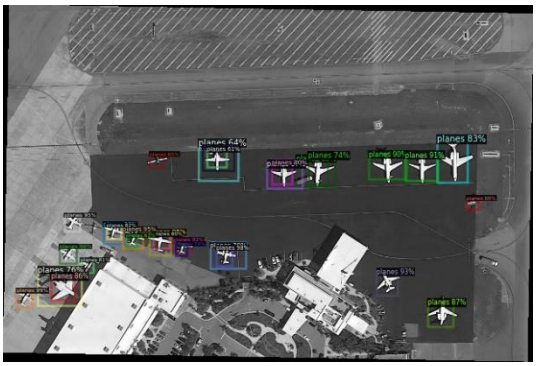


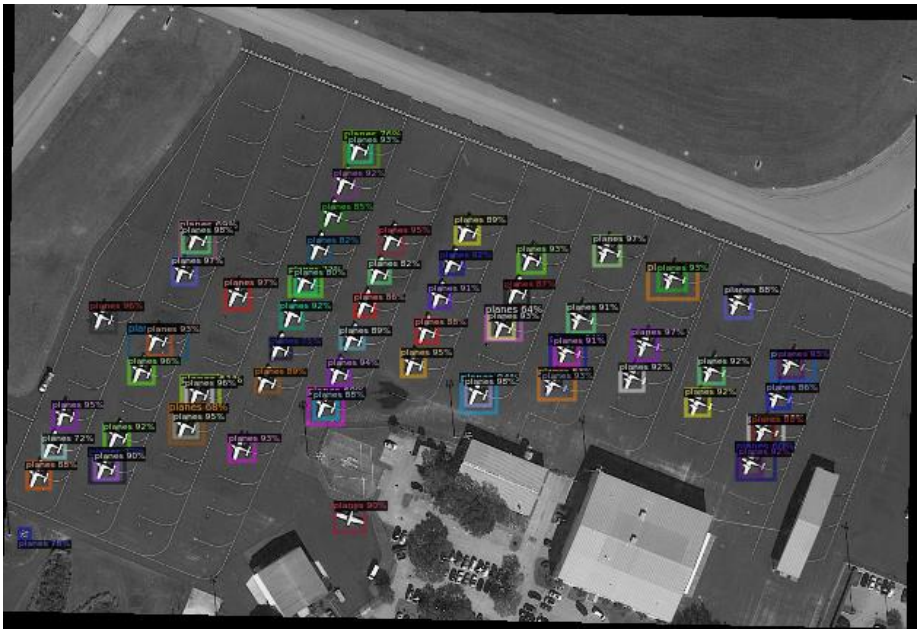


Ablation Study:

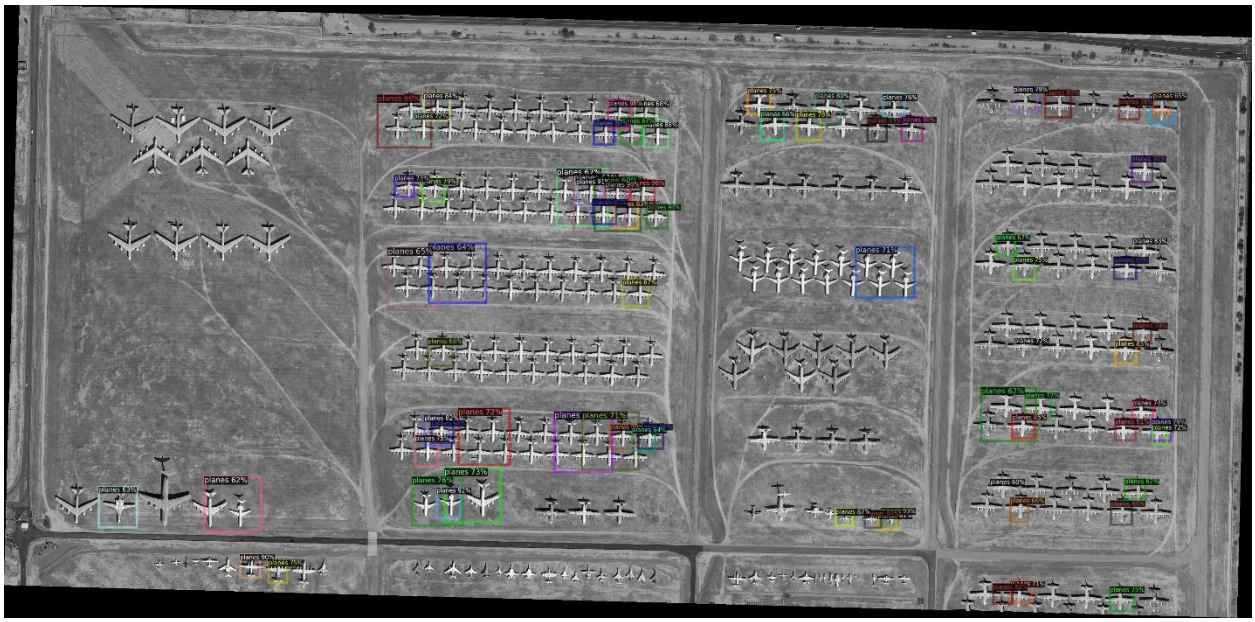
Increasing the max_iter from 500 to 1000 gave the significant improvement in training as it is shown by these results mentioned below. When max_iter is 500 the total_loss was around 1.22 and when max_iter is 1000, the lotal loss was around 0.99. Also, the segmentation results for two are mentioned below.

Visualisation of test samples when max_ter = 1000.





Visualisation of test samples when max_iter = 500.



As it is seen, that there are so many planes that are not identified in test set when max_iter was 500, therefore max_iter = 1000 improved the performance significantly.

Part2: Semantic Segmentation

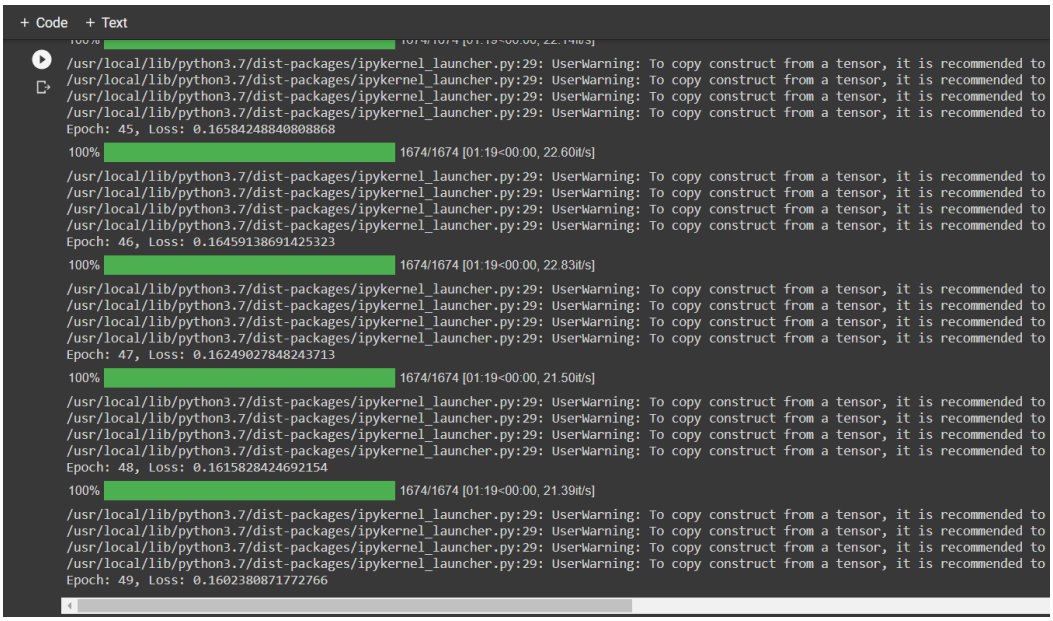
Hyperparameters used were:

- batch_size = 4
- learning_rate = 0.001
- num_epochs = 50
- optimizer = SGD optimizer

Network Architecture:

I added four more down and up sampling layers in the network. The input layer was given (3,16) and then each down sampling layer incred the output to the twice of input. After down sampling, the up sampling was done which decreases the output to half of the input. Making model like this, increased the feature channels which helped in detecting features. The optimizer was not changed and SGD optimizer was used. Number of epoch were made to 50 as loss became

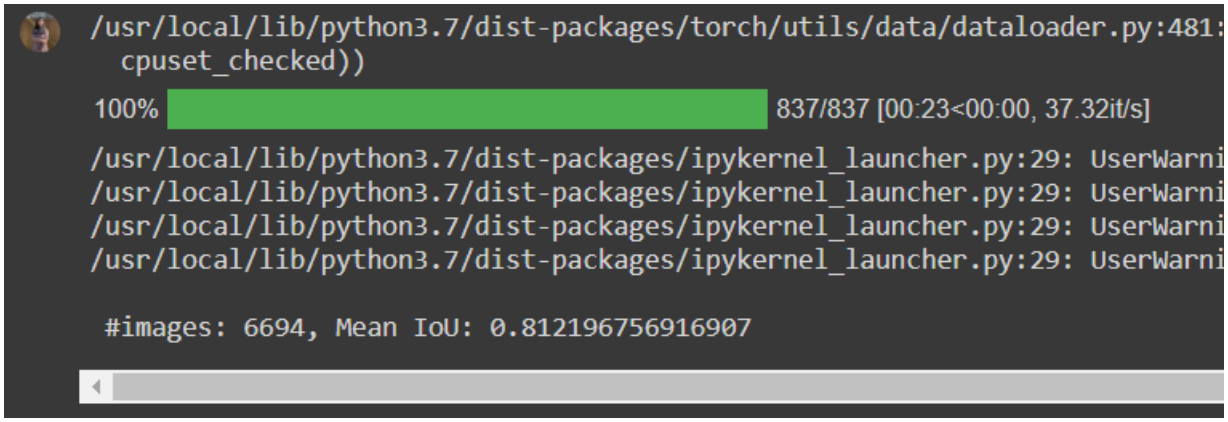
stable. The loss at first epoch was 0.62 which got decreased to 0.16.The screenshots of the los are given below:



The lost function is the default loss function, nn.BCEWithLogistsLoss()

Final IoU:

Mean IoU : 0.81



Visualised images from test set:

Image1: (original , ground truth, predicted)

Image2: (original , ground truth, predicted)

Image3: (original , ground truth, predicted)

Part3: Instance Segmentation:

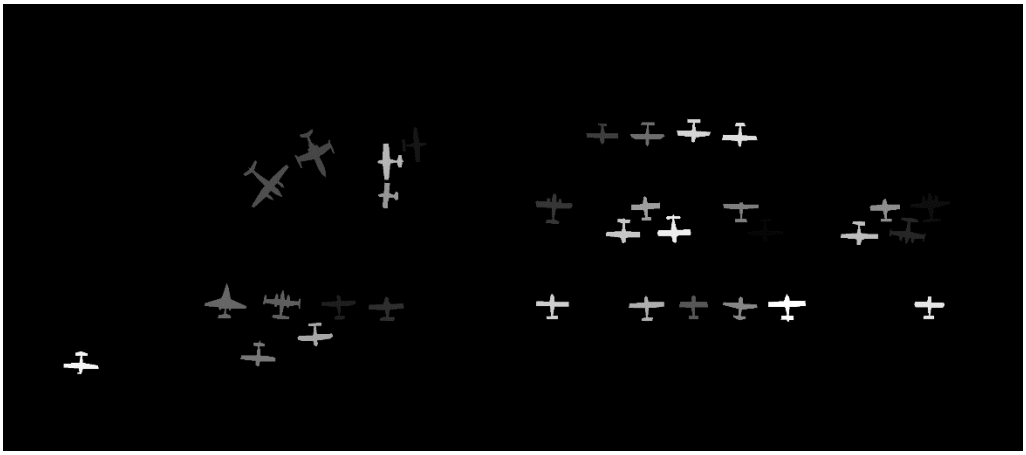
Kaggle submission:

Kaggle submission under the name RitikaGoyal

Best Score on Kaggle: 0.60771

Visualisation of results:

Image1: (original, ground truth, prediction)



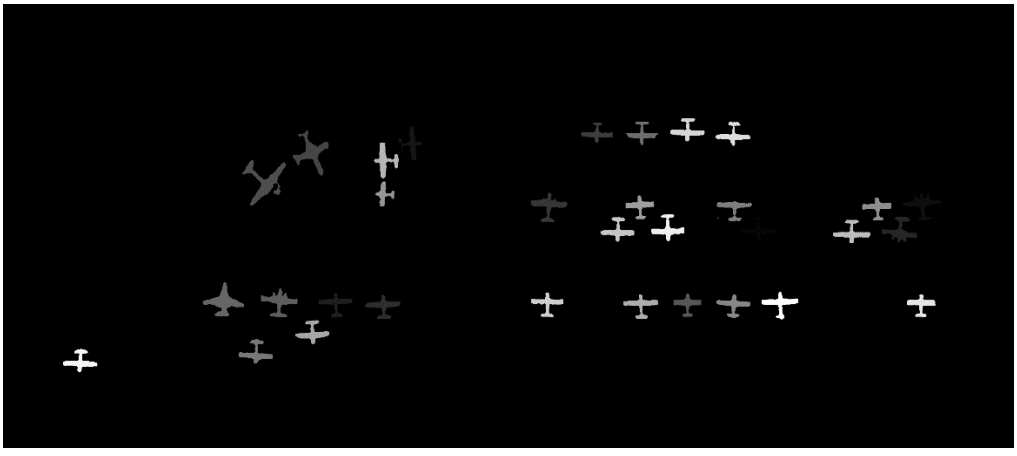
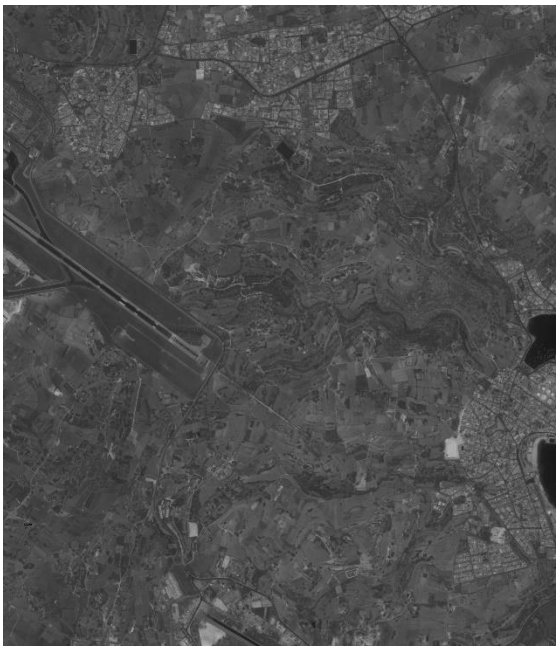
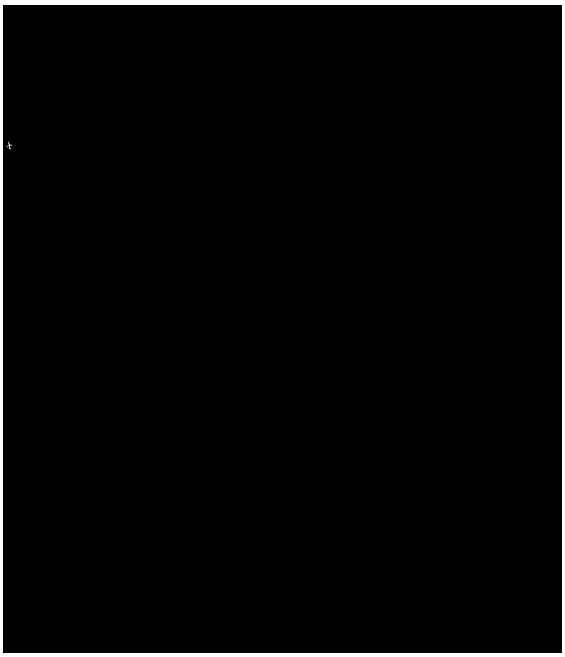


Image2: (original, ground truth, prediction)





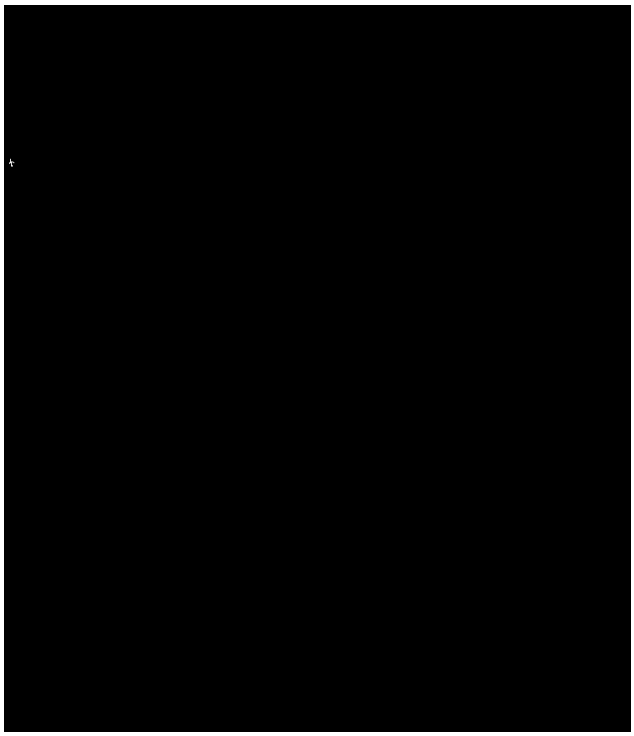
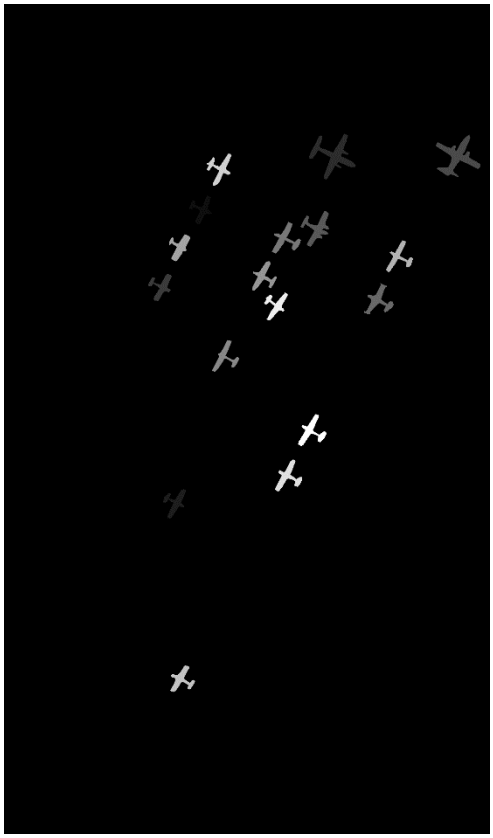
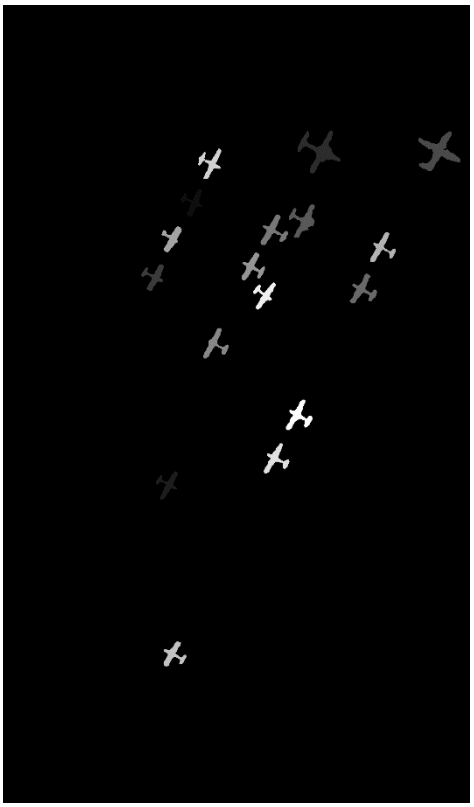


Image3: (original, ground truth, prediction)





Part4: Mask R-CNN

Visualization and the Evaluation:

Image1:

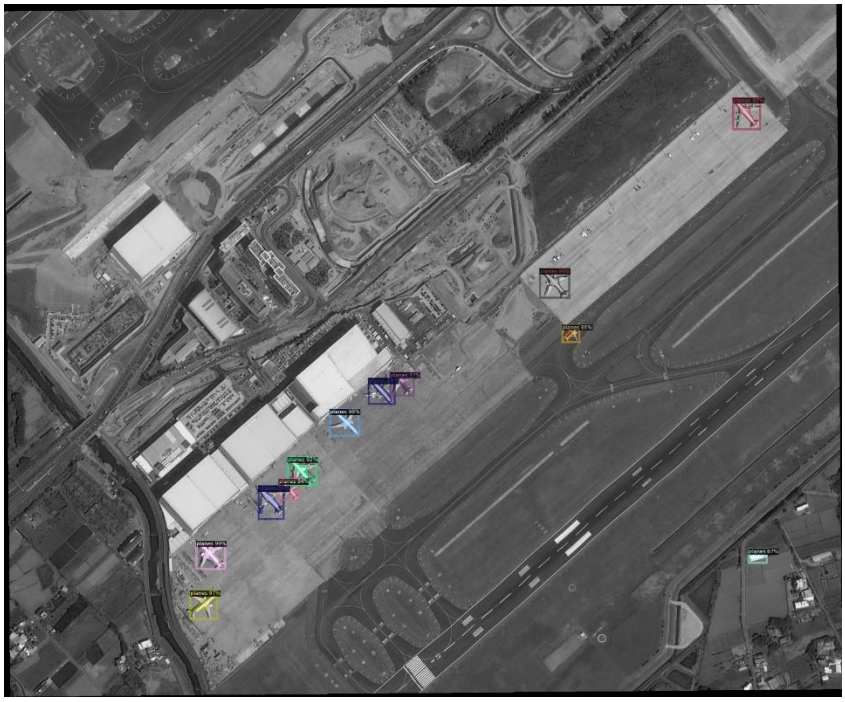
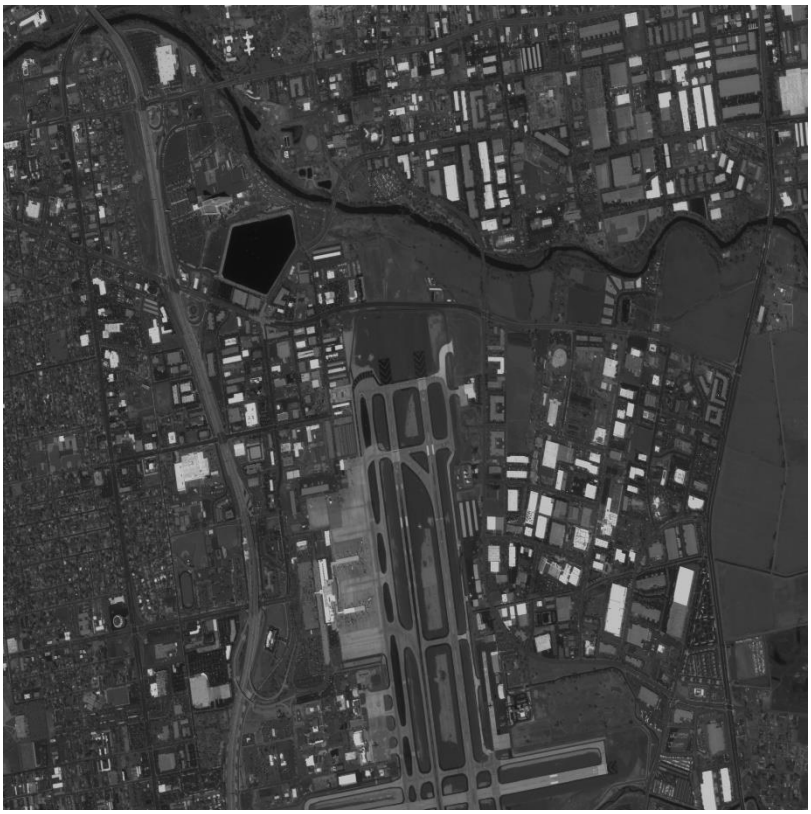


Image2:

Image3:



Observation:

Part1 model works better than this one as it is seen in the visualisation as well that many planes are not detected by this model.

AP50 of part1 is 38.554 and part4 is 15.033. The mean IoU of part 3 is 0.81 but due to less AP50, it shows that this model is less accurate than previous one.

The screenshots are mentioned below:

Part1:

```
[11/06 23:16:15 d2.evaluation.fast_eval_api]: COCOeval_opt.accumulate() finished in 0.01 seconds.
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.142
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.386
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.061
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.158
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.161
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.110
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.011
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.083
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.187
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.167
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.190
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.219
[11/06 23:16:15 d2.evaluation.coco_evaluation]: Evaluation results for bbox:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 14.213 | 38.554 | 6.053  | 15.787 | 16.074 | 11.011 |
OrderedDict([('bbox', {'AP': 14.213204817920975, 'AP50': 38.55438353822251, 'AP75': 6.052737722128967, 'APs':
```

Part4:

```
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.018
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.044
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.252
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.004
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.028
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.053
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.024
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.065
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.289
[11/07 05:57:09 d2.evaluation.coco_evaluation]: Evaluation results for segm:
|   AP   |  AP50  |  AP75  |  APs   |  APm   |  APl   |
|:------:|:------:|:------:|:------:|:------:|:------:|
| 3.979  | 15.033 | 1.173  | 1.774  | 4.441  | 25.165 |
OrderedDict([('bbox', {'AP': 16.838123079673075, 'AP50': 28.01757456538375, 'AP75': 17.776661
```