Cyrus Ahn
301377544
CMPT 412
Spring 2023

Project 3
Object Detection, Semantic Segmentation, and Instance Segmentation

(Use 1 Free-late day)

Part 1:

First used default parameters:

```
cfg = get_cfg()
cfg.OUTPUT_DIR = "{}/output/".format(BASE_DIR)

cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("data_detection_train")
cfg.DATASETS.TEST = ()

cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_101_FPN_3x.yaml")
cfg.SOLVER.IMS_PER_BATCH = 2 # default
cfg.SOLVER.BASE_LR = 0.00025 # default
cfg.SOLVER.MAX_ITER = 500  # default: 500
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512  # default
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
```

Ran again with improved parameters:

```
cfg = get_cfg()
cfg.OUTPUT_DIR = "{}/output/".format(BASE_DIR)

cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("plane_train",)
cfg.DATASETS.TEST = ("plane_test",)

cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_R_50_FPN_3x.yaml")
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.0005
cfg.SOLVER.MAX_ITER = 1000
cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 512
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
```

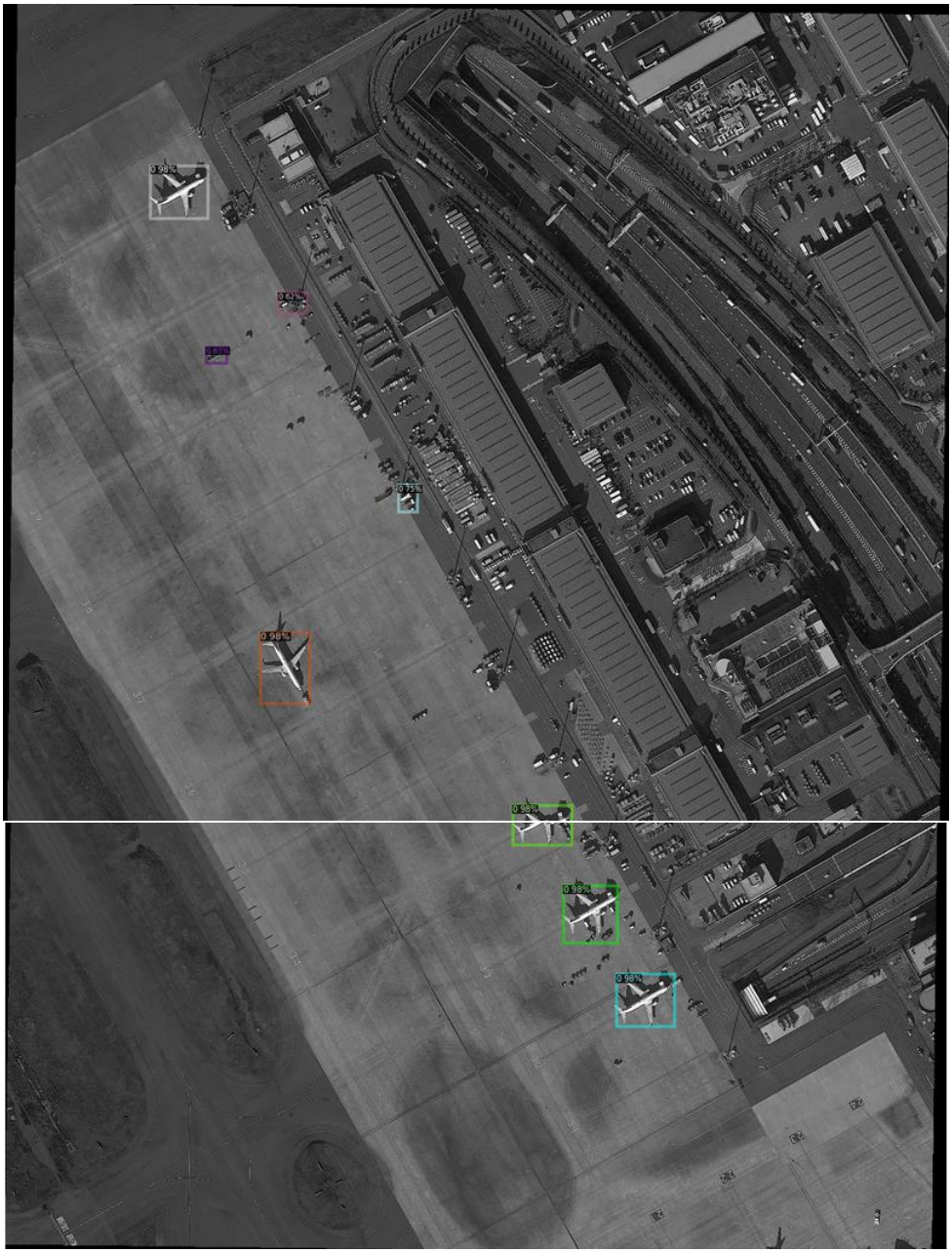
Increasing max iterations from 500 to 1000 allowed for more iterations for training. Losses continued to improve after 500th iteration so useful training was being done.

Increased learning rate to 0.0005. This learning rate seems to be able to get out of local minima and allow for better convergence to achieve a better score

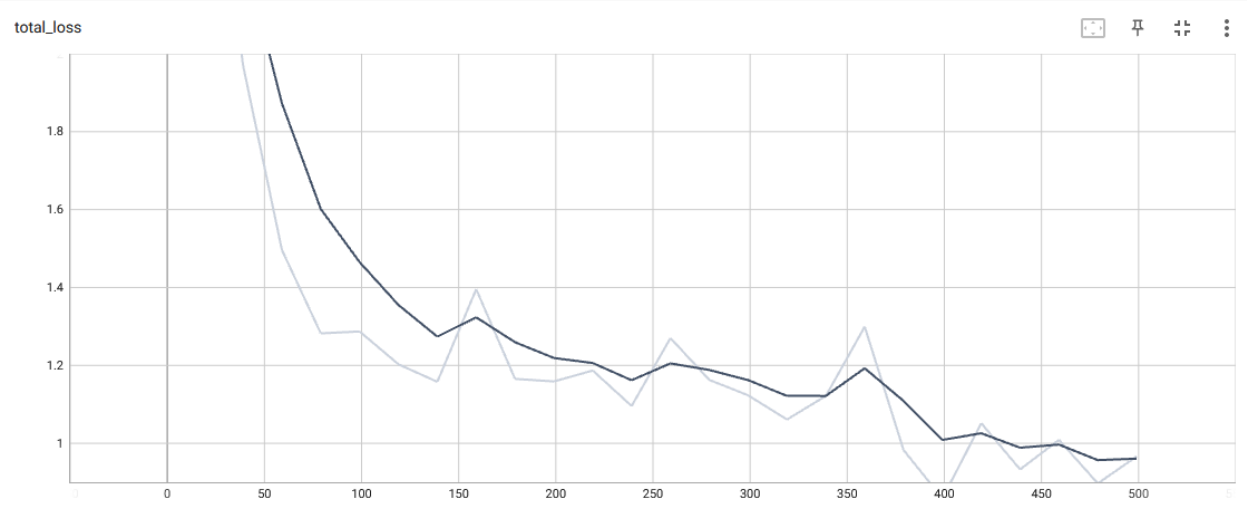
Sample photos of the test set:

(photo was split in 2 from screenshot)

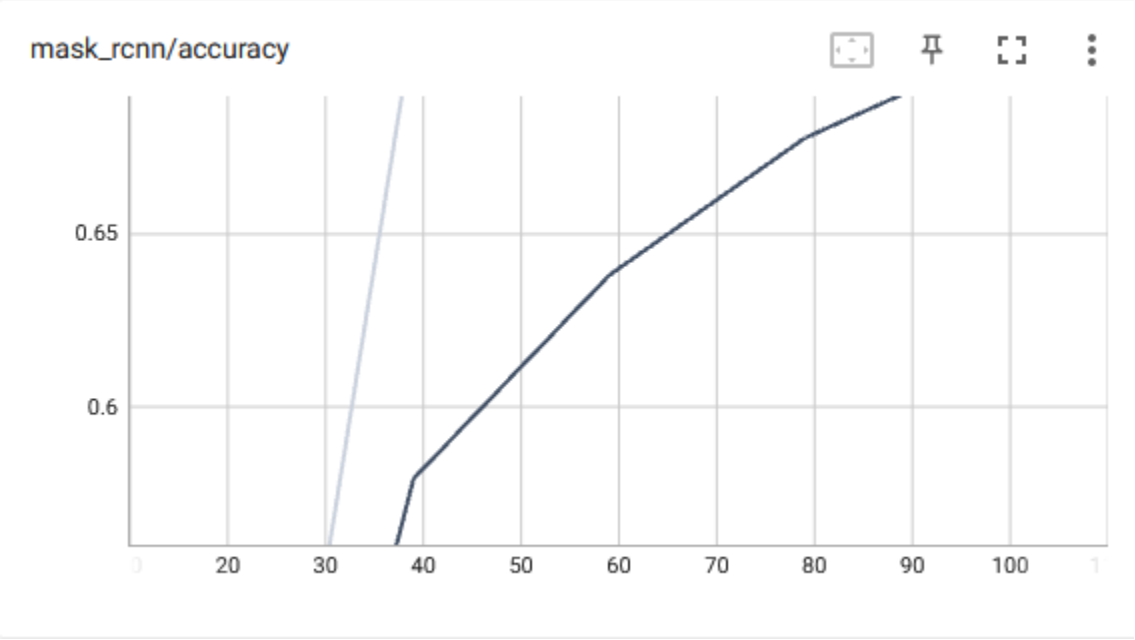
Sample predicted result:



Total loss:



Accuracy:



Albation study:

Result of default parameters:

AP	AP50	AP75	APs	APm	APl
19.591	33.203	20.786	10.727	25.372	52.012

Evaluation result with adjusted parameters:

AP	AP50	AP75	APs	APm	APl
31.704	51.891	35.500	20.843	40.743	61.219

Sample result with adjusted parameters:

The AP50 improved from 33.203 to 51.891

Part 2:

Hyper parameters used:

```
num_epochs = 50
batch_size = 8
learning_rate = 0.001
weight_decay = 1e-5

model = MyModel() # initialize the model
model = model.cuda() # move the model to GPU
loader, _ = get_plane_dataset('train', batch_size) # initialize data_loader
crit = nn.BCEWithLogitsLoss() # Define the loss function
optim = torch.optim.SGD(model.parameters(), lr=learning_rate, weight_decay=weight_decay)
```

Final architecture:

```
    self.input_conv = conv(3, 8)
    self.down1 = down(8, 16)
    self.down2 = down(16, 32)
    self.down3 = down(32, 64)
    self.down4 = down(64, 128)
    self.down5 = down(128, 256)
    self.down6 = down(256, 512)

    self.up1 = up(512, 256)
    self.up2 = up(256, 128)
    self.up3 = up(128, 64)
    self.up4 = up(64, 32)
    self.up5 = up(32, 16)
    self.up6 = up(16, 8)
    self.up7 = conv(8, 4)
    self.output_conv = conv(4, 1, False)

    self.pool1 = nn.BatchNorm2d(128)
    self.pool2 = nn.BatchNorm2d(32)
    self.pool3 = nn.BatchNorm2d(4)

def forward(self, input):
  y = self.input_conv(input)
  y = self.down1(y)
  y = self.down2(y)
  y = self.down3(y)
  y = self.pool1(self.down4(y))
  y = self.down5(y)
  y = self.down6(y)
  y = self.up1(y)
  y = self.pool1(self.up2(y))
  y = self.up3(y)
  y = self.pool2(self.up4(y))
  y = self.up5(y)
  y = self.up6(y)
  y = self.pool3(self.up7(y))
  output = self.output_conv(y)
  return output
```

This architecture added more scale up and down layers to the network. Each scale up layer doubled the output of the previous layer, and each scale down layer halved the input of the previous layer. This allows the network to detect more features from each photo.

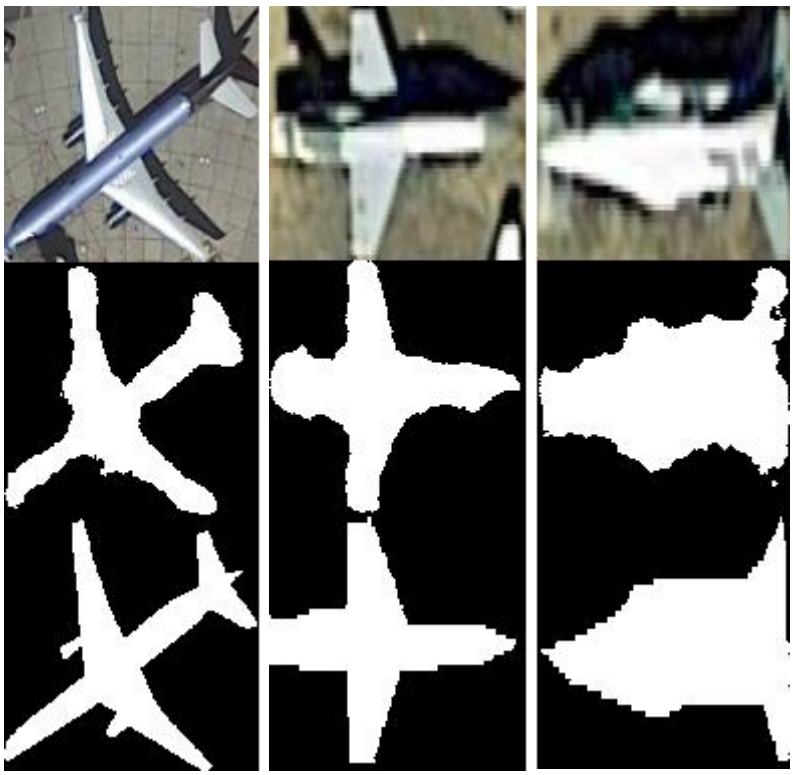
Training loss over 50 epochs:

```
Epoch: 0, Loss: 0.6632471084594727
Epoch: 9, Loss: 0.2731105089187622
Epoch: 19, Loss: 0.2200092226266861
Epoch: 29, Loss: 0.19506429135799408
Epoch: 39, Loss: 0.17832320928573608
Epoch: 49, Loss: 0.16591215133666992
```

Final mean IoU:

```
Mean IoU: 0.7633271900012819
```

Visualization:

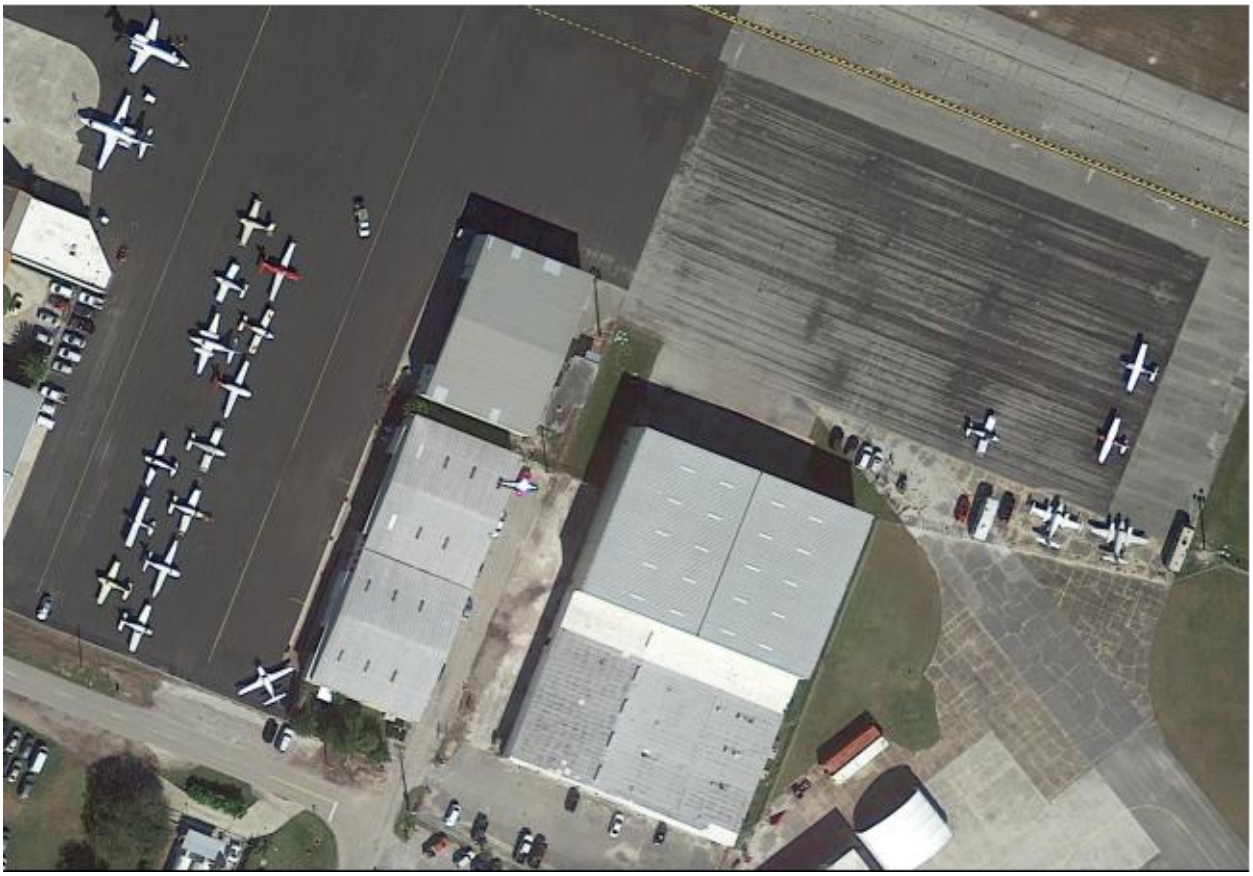


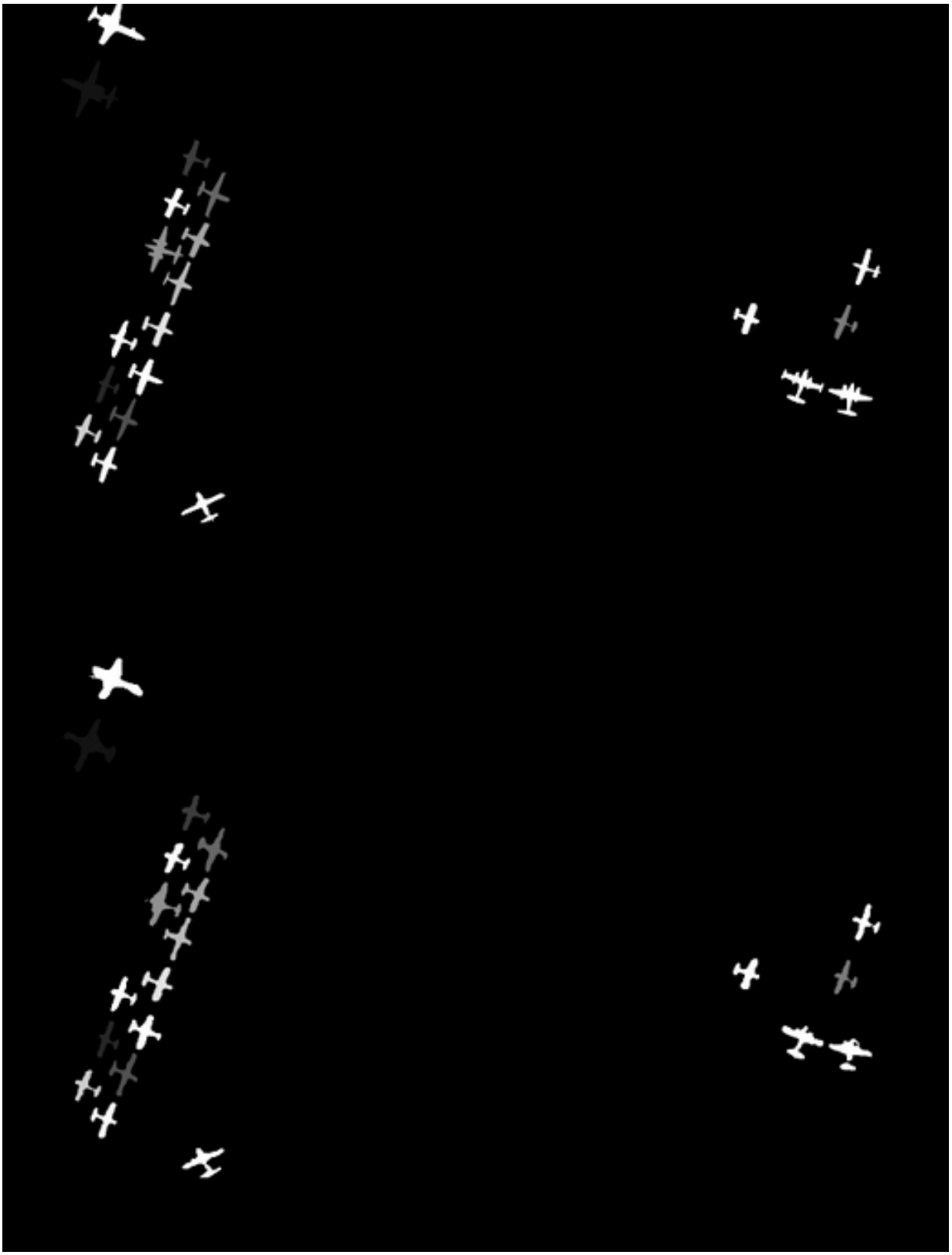
Part 3:

Name in Kaggle: Cyrus Ahn

Best Score: 0.50276

Visualization:






29

Part 4:

Visualization:


plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane
plane

This model with default parameters performed extremely poorly sometimes not detecting any planes in the image.