AIP Assignment 2 REPORT

Name: B Srinath Achary **Dept: CSA**
Sr No: 21441 **Programme: MTech AI**

Problem 1

Part A:

Objective: To implement the N-Cut algorithm to segment the images into two segments.

Steps:

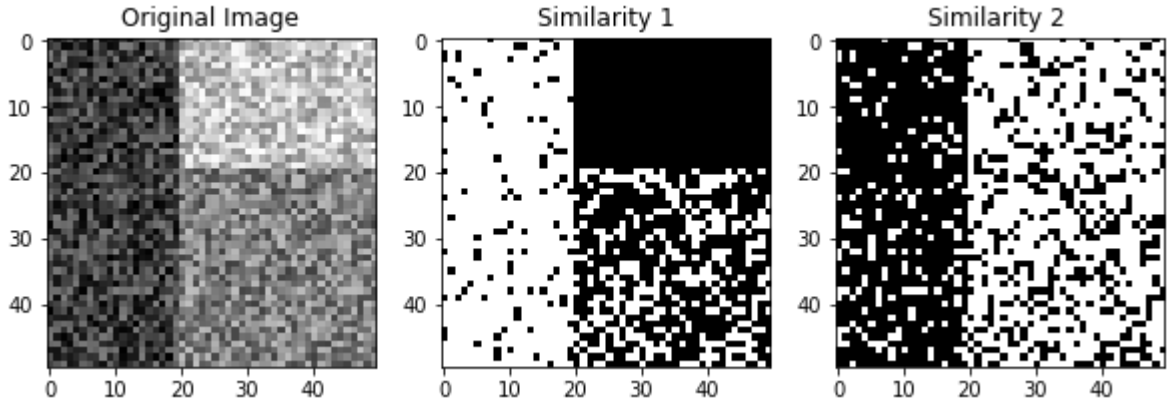
1. At first the image was downsampled to 50x50x3 and as computations were very time-consuming.
2. Then the similarity matrix was computed using the following distance formula:

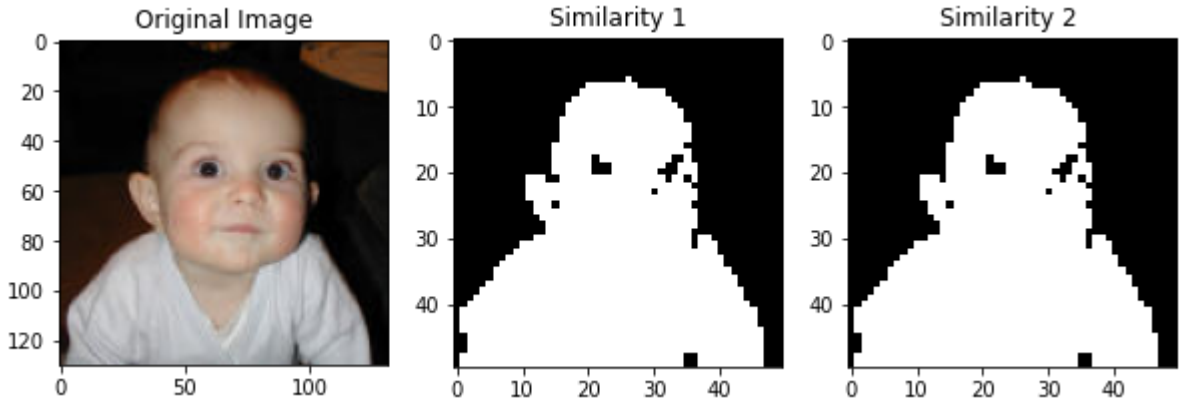
$$w_{ij} = \exp \frac{- \parallel F(i) - F(j) \parallel^2}{\sigma_I}$$

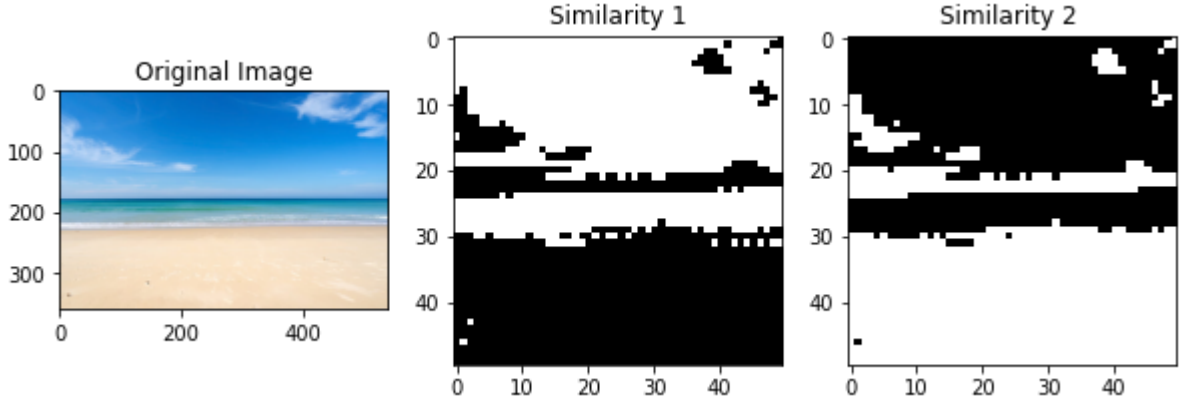
   where w_{ij} is the similarity measure between node i and j
   For Similarity 1: F(i) = (R+G+B)/3 for node i and sigma = 5
   For Similarity 2: F(j) = grayscale intensity of node i and sigma = 10
   Note: Spacial factor was not considered as it gave poor segmentation

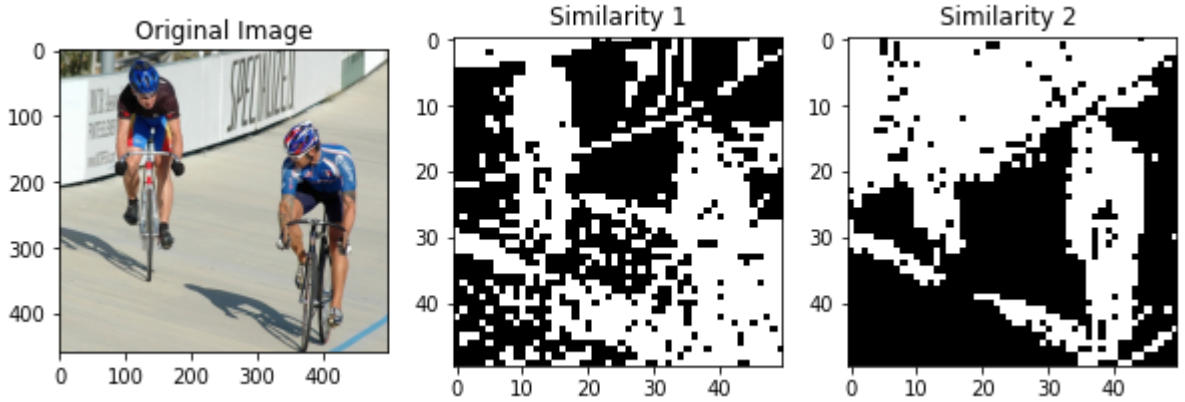
3. Then the eigenvector corresponding to the second smallest eigenvalue of the following generalized eigensystem was calculated.
4. Then the median of that eigenvector was used as a threshold to separate classes.

Results:









Issues faced: Error in the convergence while calculating the eigenvector using eigensolver from scipy for the Test4 image.

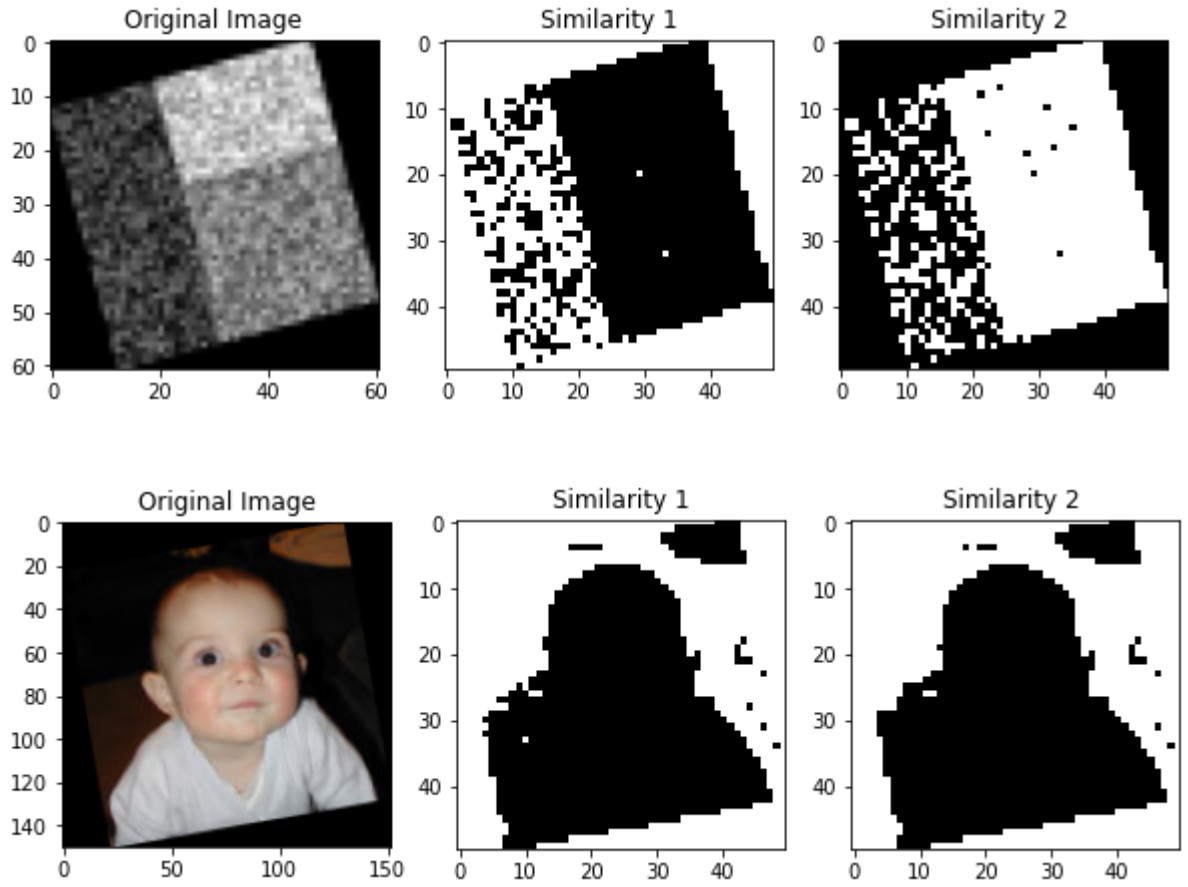
Analysis:

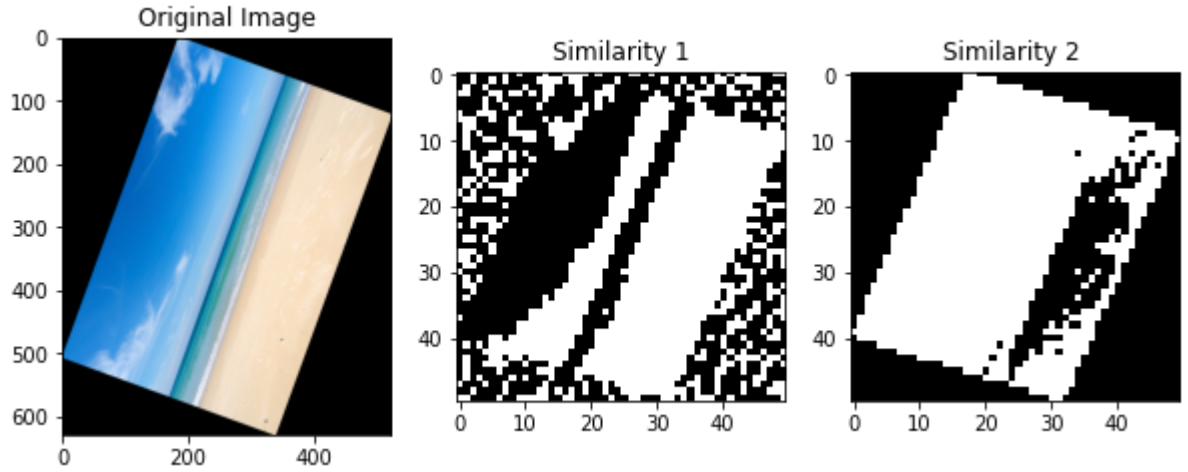
- **For Image 1:** Segmentation using Similarity 2 is better than Similarity 1.
- **For Image 2:** Both yield the same results and both are segmenting with high accuracy
- **For Image 3:** Both yield the same results but accuracy is not that high. Both are separating clouds from the sky and considering it with beach class.

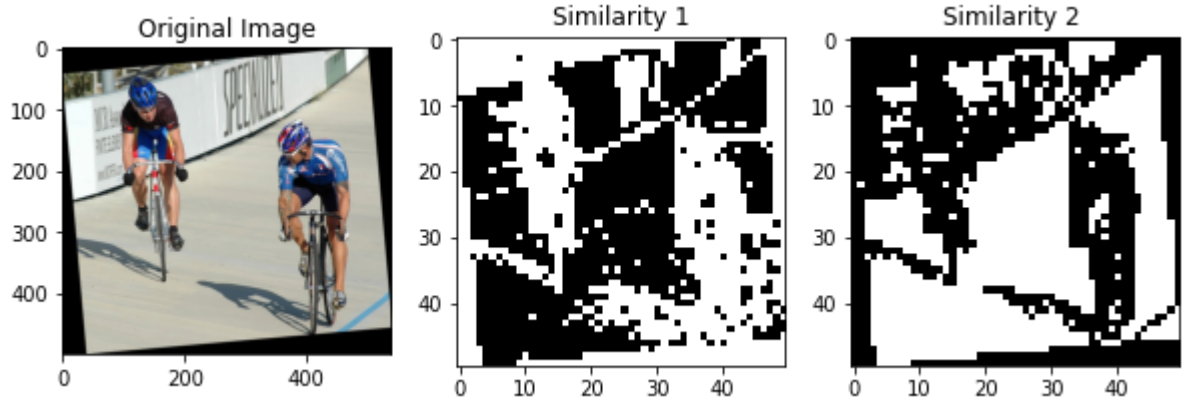
- **For Image 5:** Both yield poor results. In both cases, it is segmenting one of the humans with the background. This may be because there are a lot of variations and is difficult to segment it into 2 classes meaningfully.

PART B:

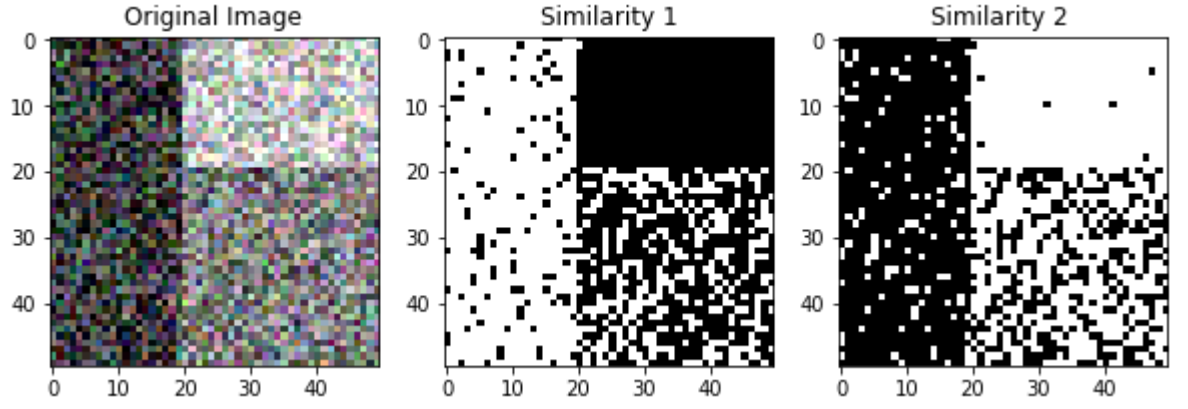
With rotation:

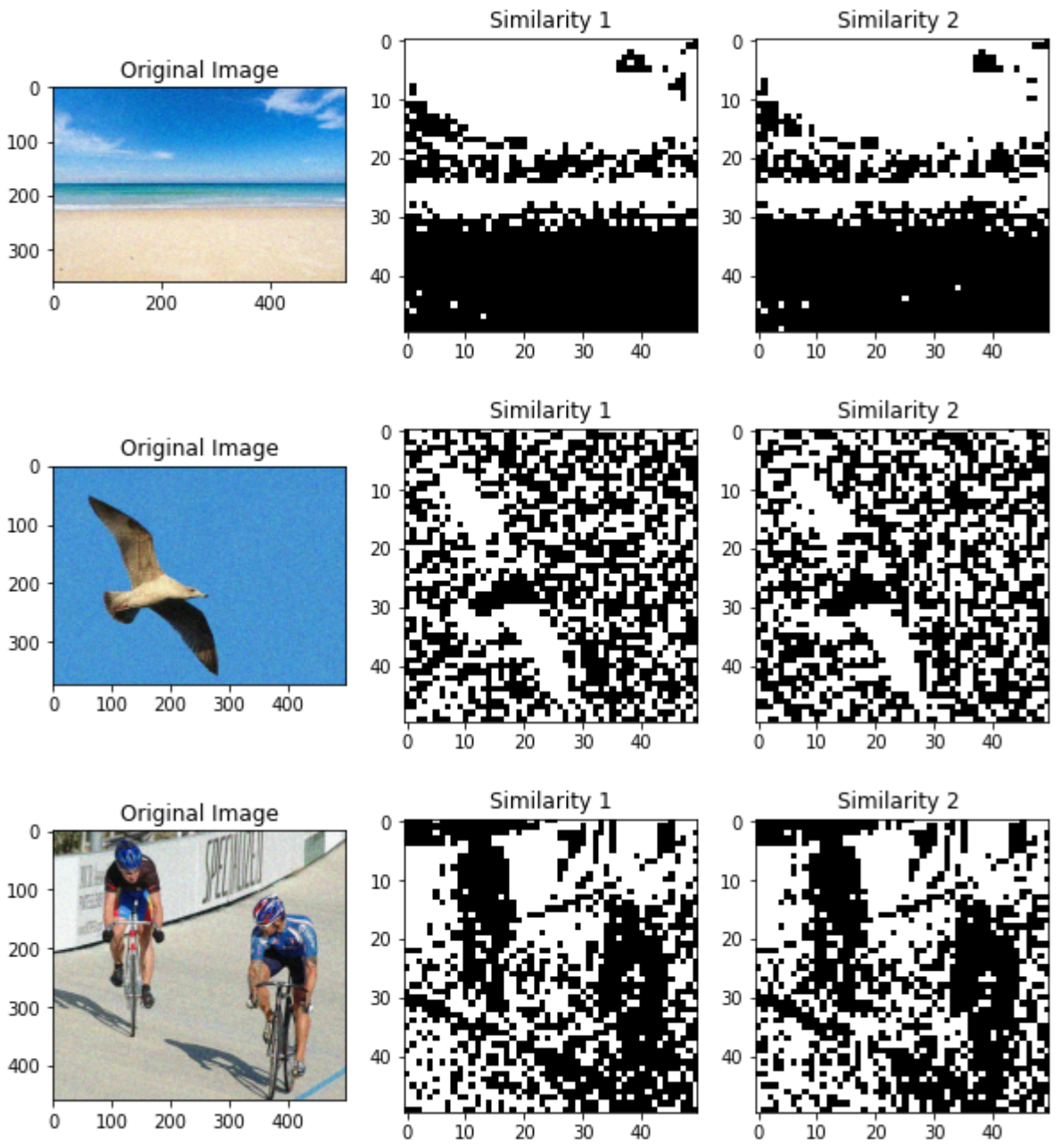






With Gaussian Noise:





PROBLEM 2:

Part A:

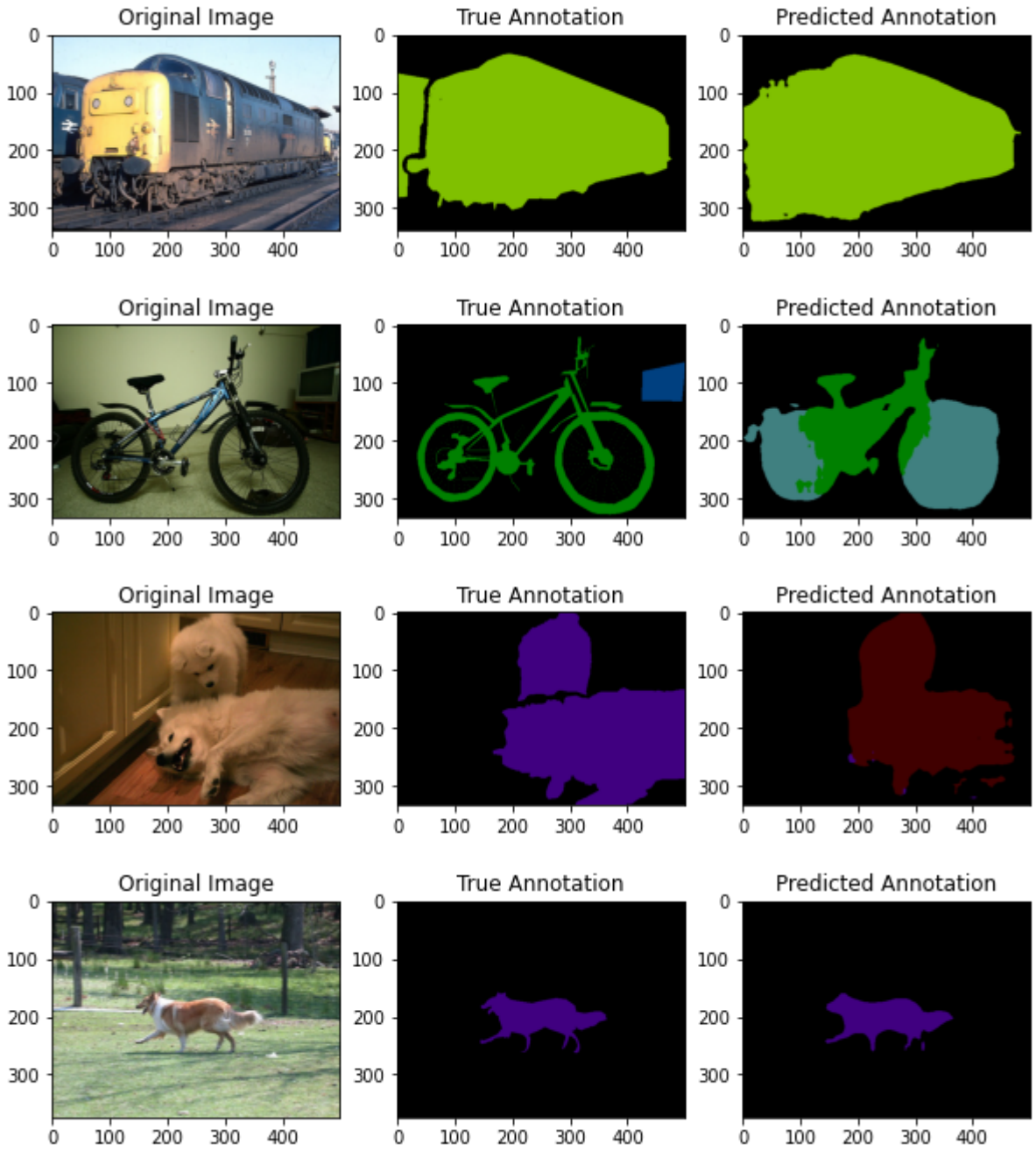
Objective: To test the given dataset on ResNet50-based FCN

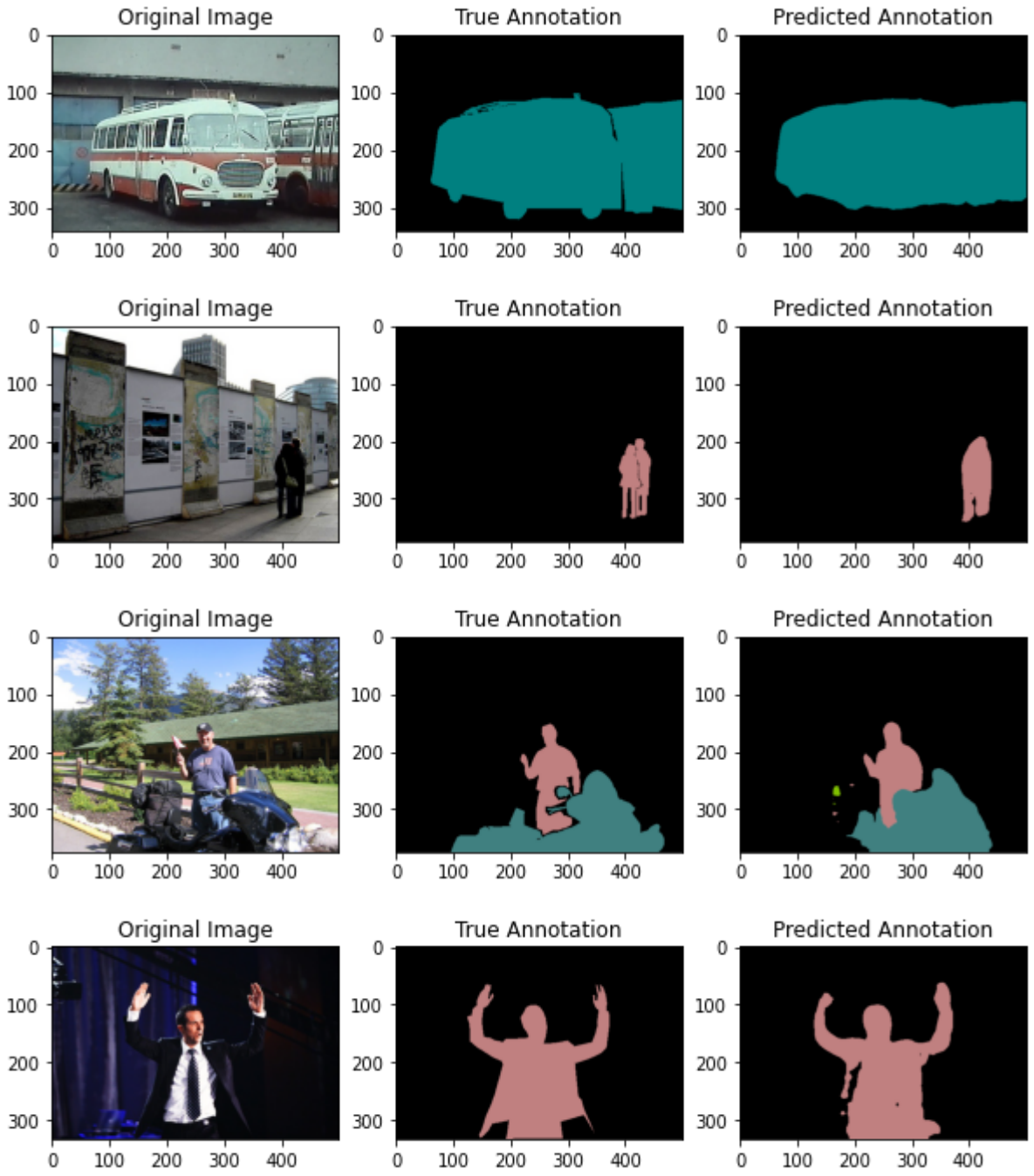
Steps:

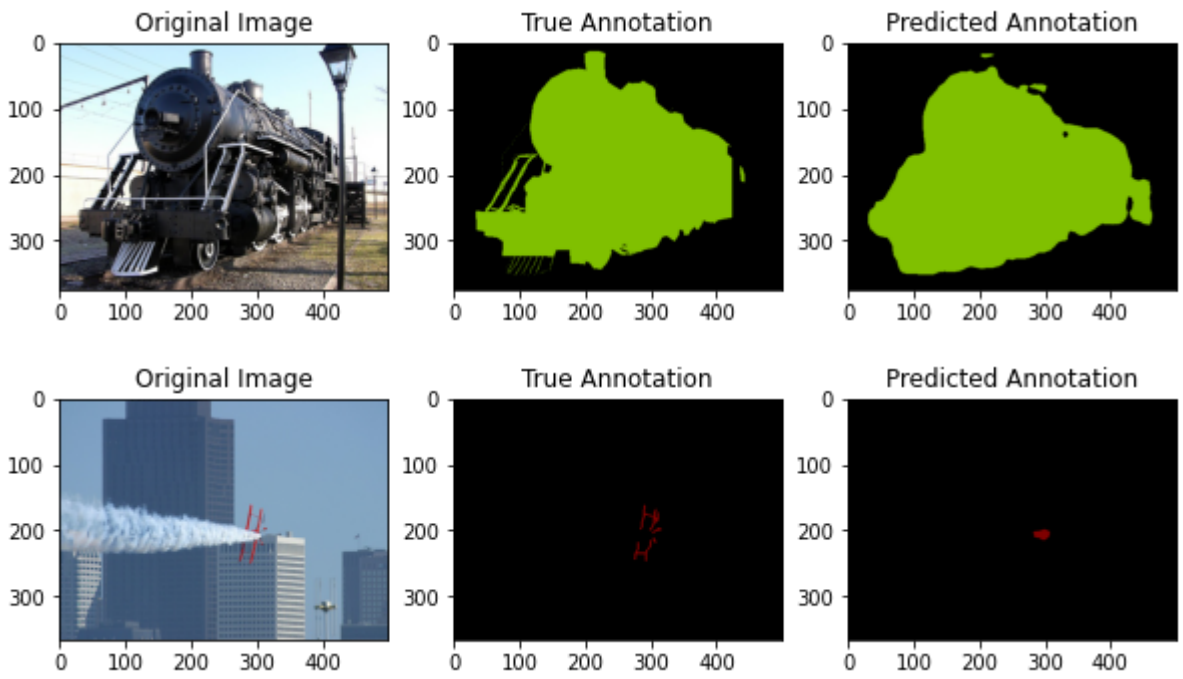
1. Imported all the test images and their corresponding annotations into a list.
2. Defined a dictionary for Pascal VOC color codes to classes.

3. Created a Custom Dataset class for images and their corresponding masks which were created using the dictionary mentioned in step 2.
4. Then loaded the FCN_ResNet50 with pre-trained weights
5. Then tested the model with the given test set.

Accuracy Metrics:
Pixel Accuracy: **88.98 %**
Mean IoU: **56.10 %**




Original Image
True Annotation
Predicted Annotation
0
100
200
300
400



PART B:

OBJECTIVE: To modify the last FC layers of MobileNetv2 to convert it to an FCN.

Steps:

1. Loaded Mobilenetv2 model and implemented the following changes:
    a. These are the final layers of the original architecture

```
          Conv2d-151          [-1, 320, 12, 13]         307,200
     BatchNorm2d-152          [-1, 320, 12, 13]             640
InvertedResidual-153          [-1, 320, 12, 13]               0
          Conv2d-154         [-1, 1280, 12, 13]         409,600
     BatchNorm2d-155         [-1, 1280, 12, 13]           2,560
           ReLU6-156         [-1, 1280, 12, 13]               0
         Dropout-157                [-1, 1280]               0
          Linear-158                [-1, 1000]       1,281,000
================================================================
Total params: 3,504,872
Trainable params: 3,504,872
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 1.72
Forward/backward pass size (MB): 461.85
Params size (MB): 13.37
Estimated Total Size (MB): 476.93
----------------------------------------------------------------
```

    b. Then removed the final linear layer and Dropout-157

```
          Conv2d-148            [-1, 960, 7, 7]           8,640
     BatchNorm2d-149            [-1, 960, 7, 7]           1,920
           ReLU6-150            [-1, 960, 7, 7]               0
          Conv2d-151            [-1, 320, 7, 7]         307,200
     BatchNorm2d-152            [-1, 320, 7, 7]             640
InvertedResidual-153            [-1, 320, 7, 7]               0
          Conv2d-154           [-1, 1280, 7, 7]         409,600
     BatchNorm2d-155           [-1, 1280, 7, 7]           2,560
           ReLU6-156           [-1, 1280, 7, 7]               0
================================================================
Total params: 2,223,872
Trainable params: 2,223,872
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 152.85
Params size (MB): 8.48
Estimated Total Size (MB): 161.91
----------------------------------------------------------------
```

c. Then added the following Transpose convolution layers and used the Bilinear kernel to initialize weights

```
  ConvTranspose2d-157          [-1, 512, 14, 14]      10,486,272
     BatchNorm2d-158          [-1, 512, 14, 14]           1,024
            ReLU-159          [-1, 512, 14, 14]               0
  ConvTranspose2d-160          [-1, 256, 28, 28]       2,097,408
     BatchNorm2d-161          [-1, 256, 28, 28]             512
            ReLU-162          [-1, 256, 28, 28]               0
  ConvTranspose2d-163          [-1, 128, 56, 56]         524,416
     BatchNorm2d-164          [-1, 128, 56, 56]             256
            ReLU-165          [-1, 128, 56, 56]               0
  ConvTranspose2d-166         [-1, 64, 112, 112]         131,136
     BatchNorm2d-167         [-1, 64, 112, 112]             128
            ReLU-168         [-1, 64, 112, 112]               0
  ConvTranspose2d-169         [-1, 21, 224, 224]          21,525
     BatchNorm2d-170         [-1, 21, 224, 224]              42
            ReLU-171         [-1, 21, 224, 224]               0
================================================================
Total params: 15,486,591
Trainable params: 13,262,719
Non-trainable params: 2,223,872
----------------------------------------------------------------
Input size (MB): 0.57
Forward/backward pass size (MB): 211.42
Params size (MB): 59.08
Estimated Total Size (MB): 271.07
----------------------------------------------------------------
```

Note: All the shapes of the intermediate layers are with respect to the input size: 224x224x3

2. Details for training the model:

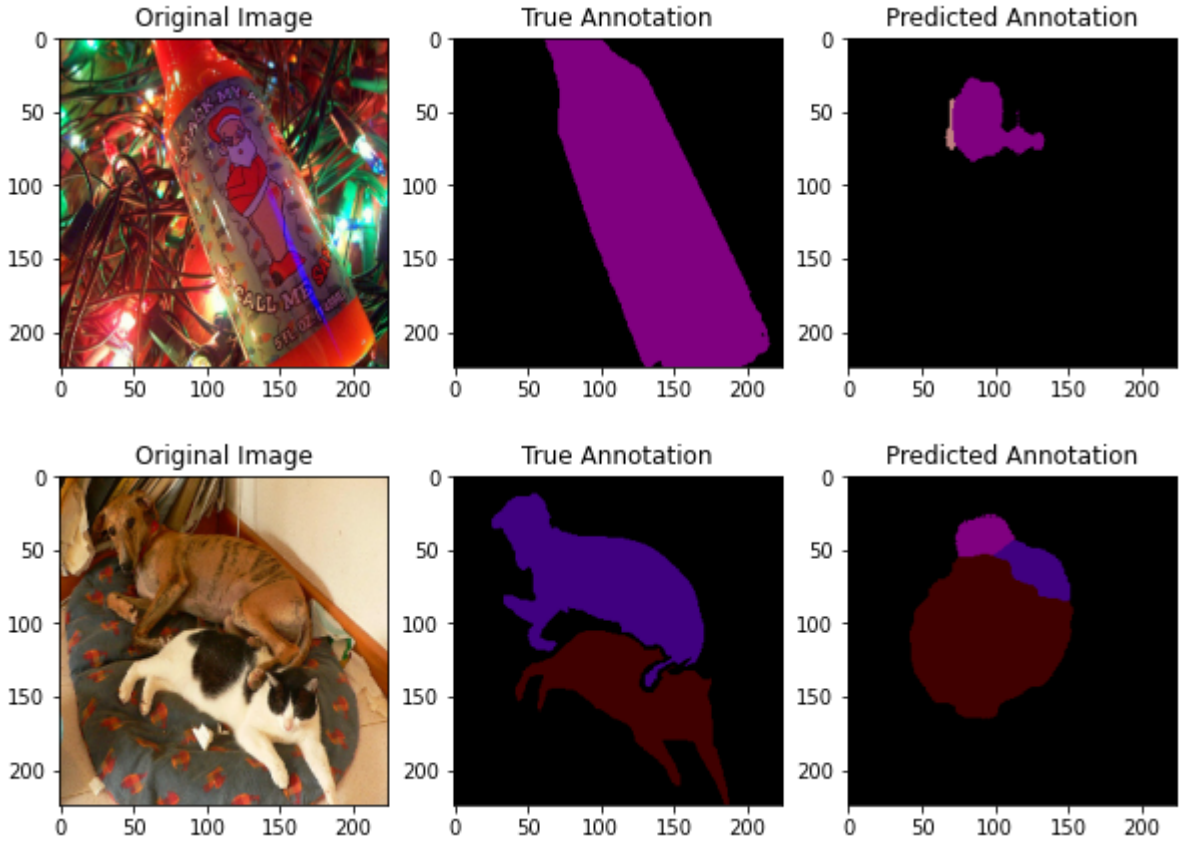
```
criterion = torch.nn.CrossEntropyLoss(ignore_index = 255)
optimizer = torch.optim.SGD(params_to_update, lr = 0.01, momentum = 0.9)
exp_lr_scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size = 7, gamma = 1)
```

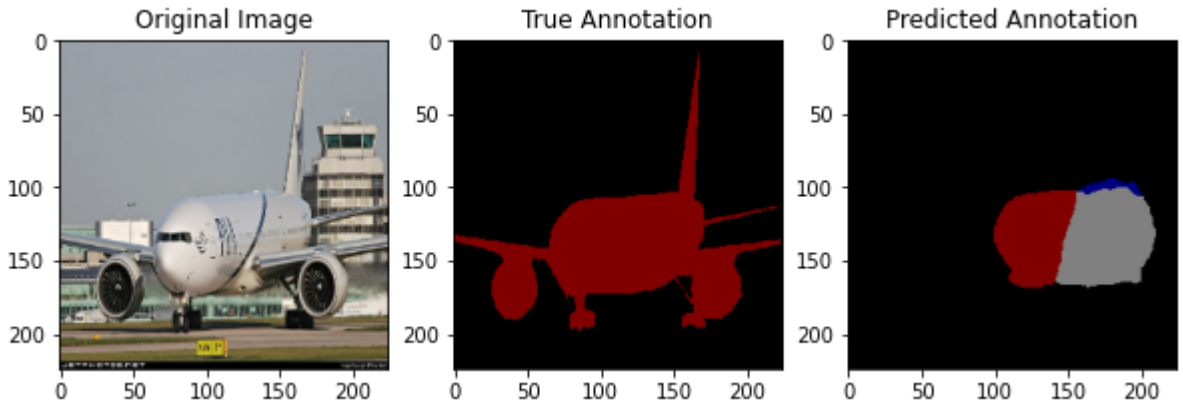
3. Then tested the fine-tuned model on the test set.

Accuracy Metrics:

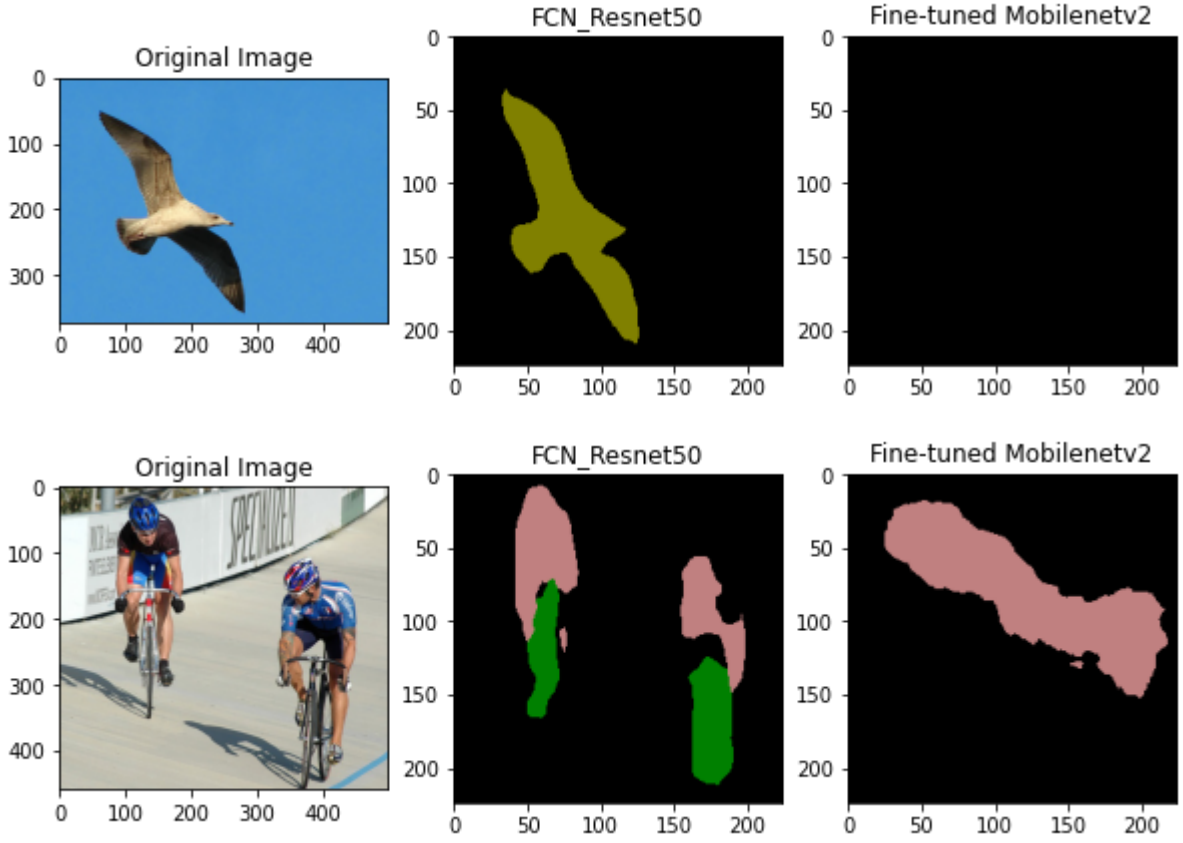
Pixel Accuracy: **77.57 %**

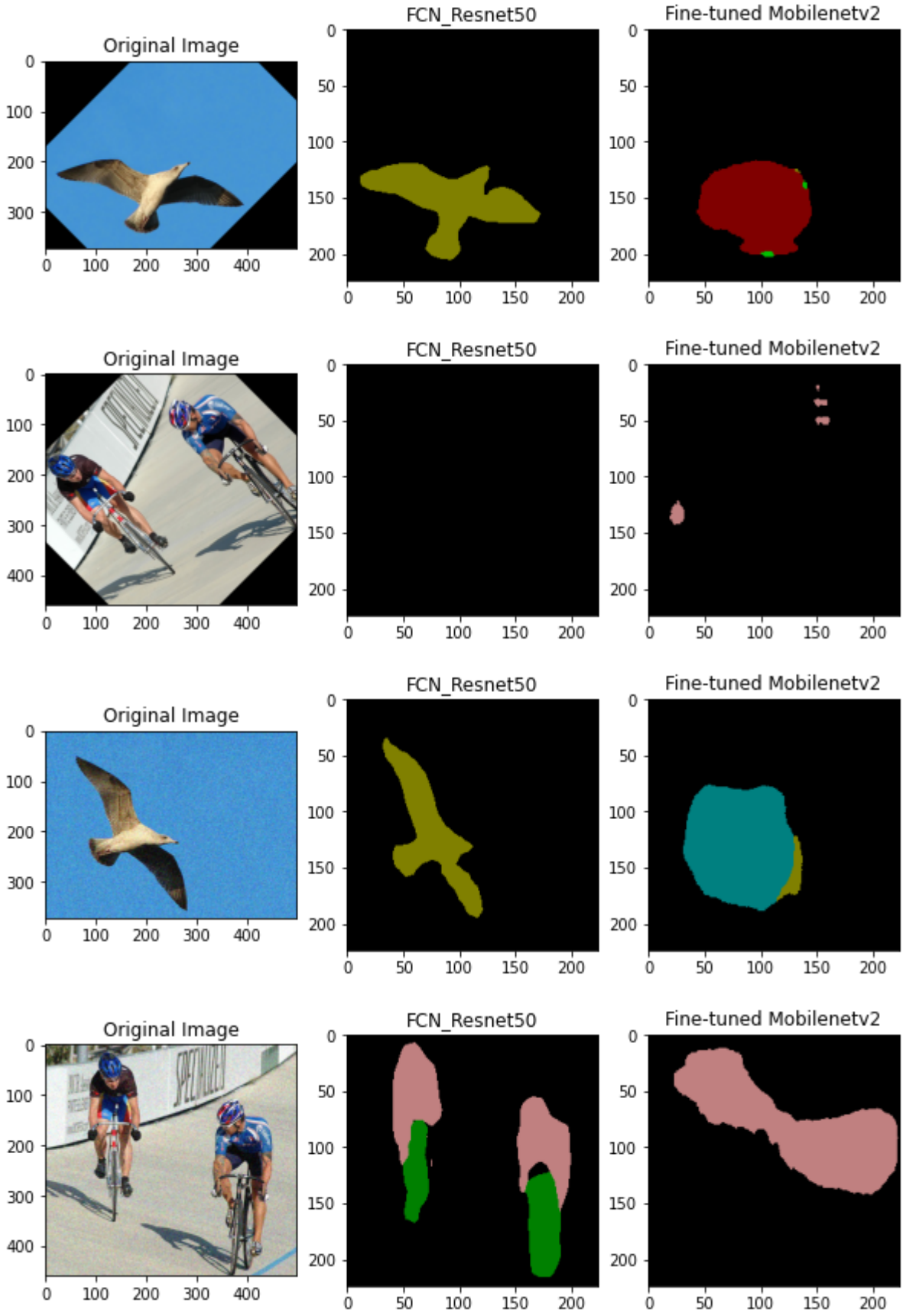
Mean IoU:**31.39** **%**




Original Image
True Annotation
Predicted Annotation

PART C :


Original Image
FCN_Resnet50
Fine-tuned Mobilenetv2
Original Image
FCN_Resnet50
Fine-tuned Mobilenetv2


Original Image
FCN_Resnet50
Fine-tuned Mobilenetv2
Original Image
FCN_Resnet50
Fine-tuned Mobilenetv2
Original Image
FCN_Resnet50
Fine-tuned Mobilenetv2
Original Image
FCN_Resnet50
Fine-tuned Mobilenetv2

Conclusion:

- Segmentation by FCN_Resnet50 outputs reasonable results.
- Segmentation by Fine tuned Mobilenetv2 is very poor.