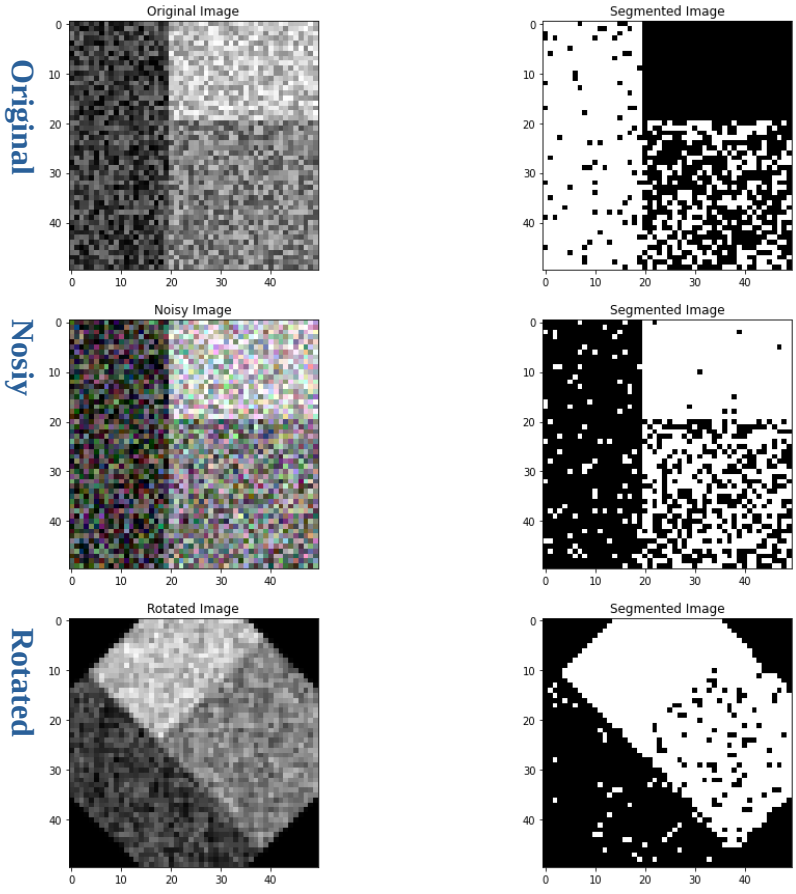
**AIP ASSIGNMENT – 2**

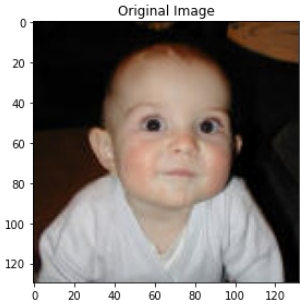
# Q1) N-Cut

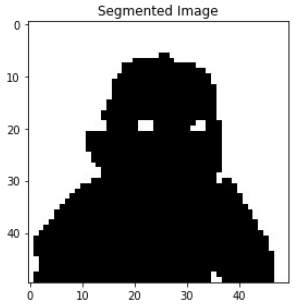
## (a) Using Intensities for Feature Similarity :

- Pixel intensity is calculated by taking the average of RGB values
- Weight [i][j] depends on the Feature Similarity calculated between two pixels i and j, multiplied by the Spatial Proximity calculated between these two pixels
- Calculation of W matrix takes a lot of time proportional to image size and hence all the images are resized to (50, 50)
- Initially both these terms were calculated to find the elements of W matrix and torch.lopbcg was used to calculate the eigenvalues of the generalized eigen value form $( D - W ) y = \lambda D y$ with nter = 100
- This approach worked well for test_1 , test_2 and test_3 and failed for others even after trying out various thresholds for y ($2^{nd}$ smallest eigenvector) such as median, mean, 0 etc.,
- So, only the Feature Similarity term was used for W calculation and scipy.sparse.eigsh was used for eigenvalue calculation. This approach gave better results
- This is because, due to the bound on the distance calculation, W matrix is not accurate and also eigen value calculation using torch.lopbcg is inaccurate ( it depends on initial guess and also no. of iterations)
- Weight matrix used to take ~ 8 mins on a (130, 130) sized image but when resized to (50, 50 ) and Spacial Proximity term is ignored, it took ~ 2 mins
- **Results**

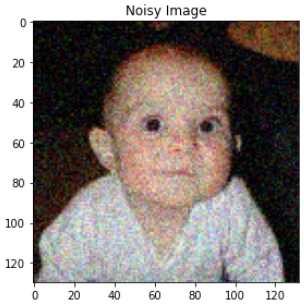


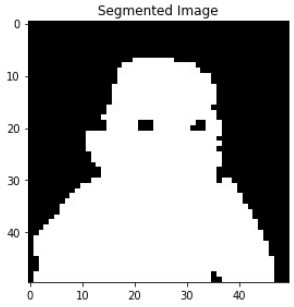
Original



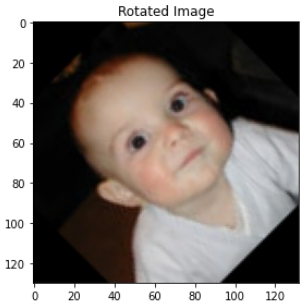



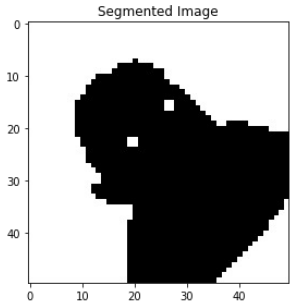

Nosiy



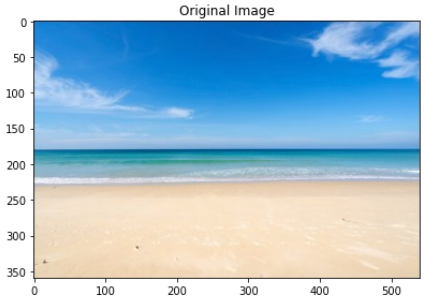



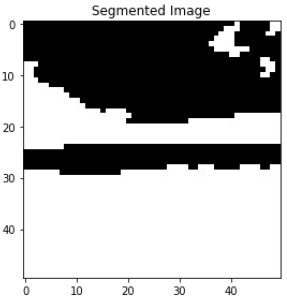

Rotated



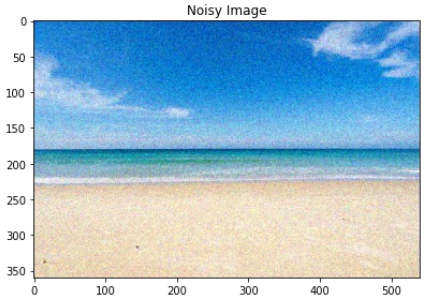



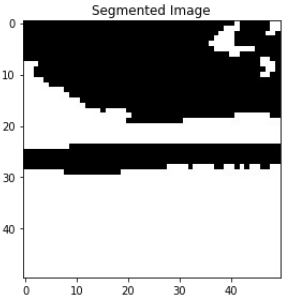

Original



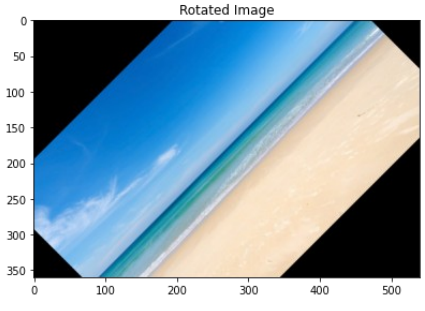



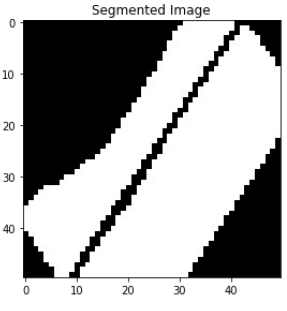

Nosiy



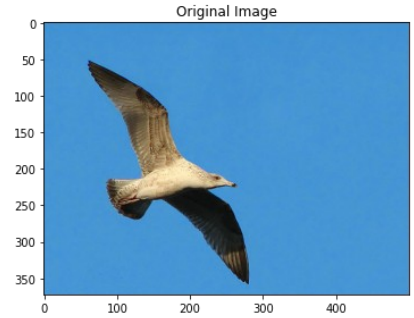



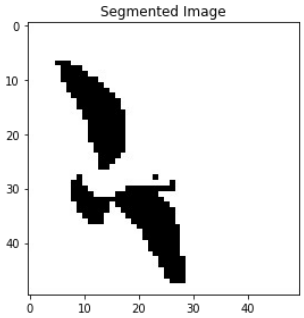

Rotated



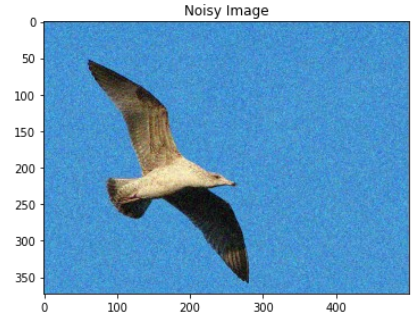



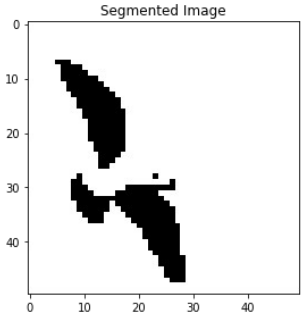
Original



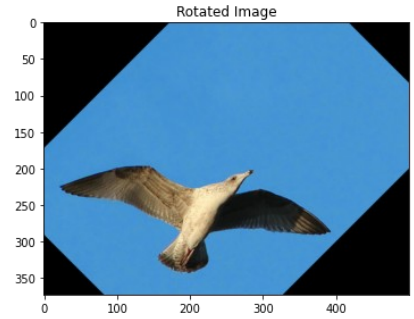



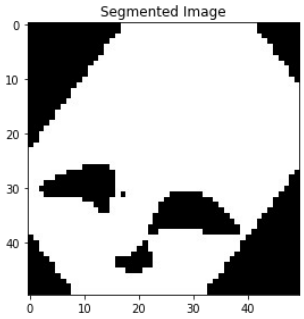

Nosiy



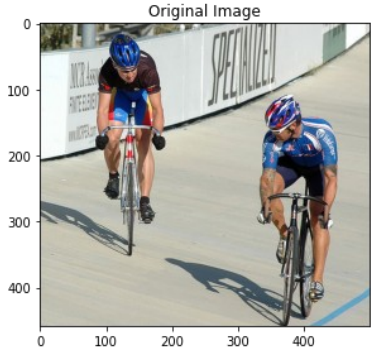



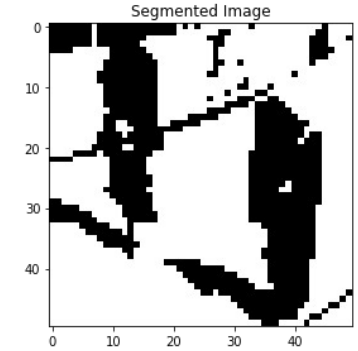

Rotated



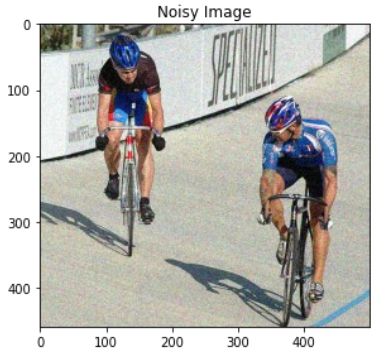



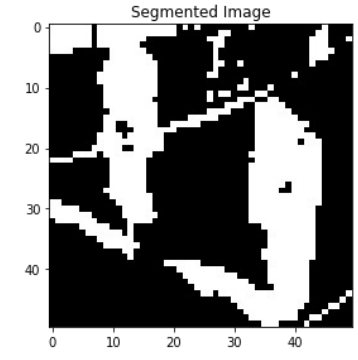

Original



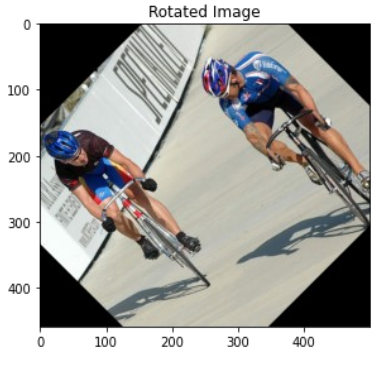



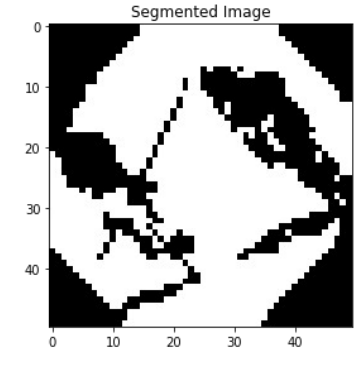

Nosiy



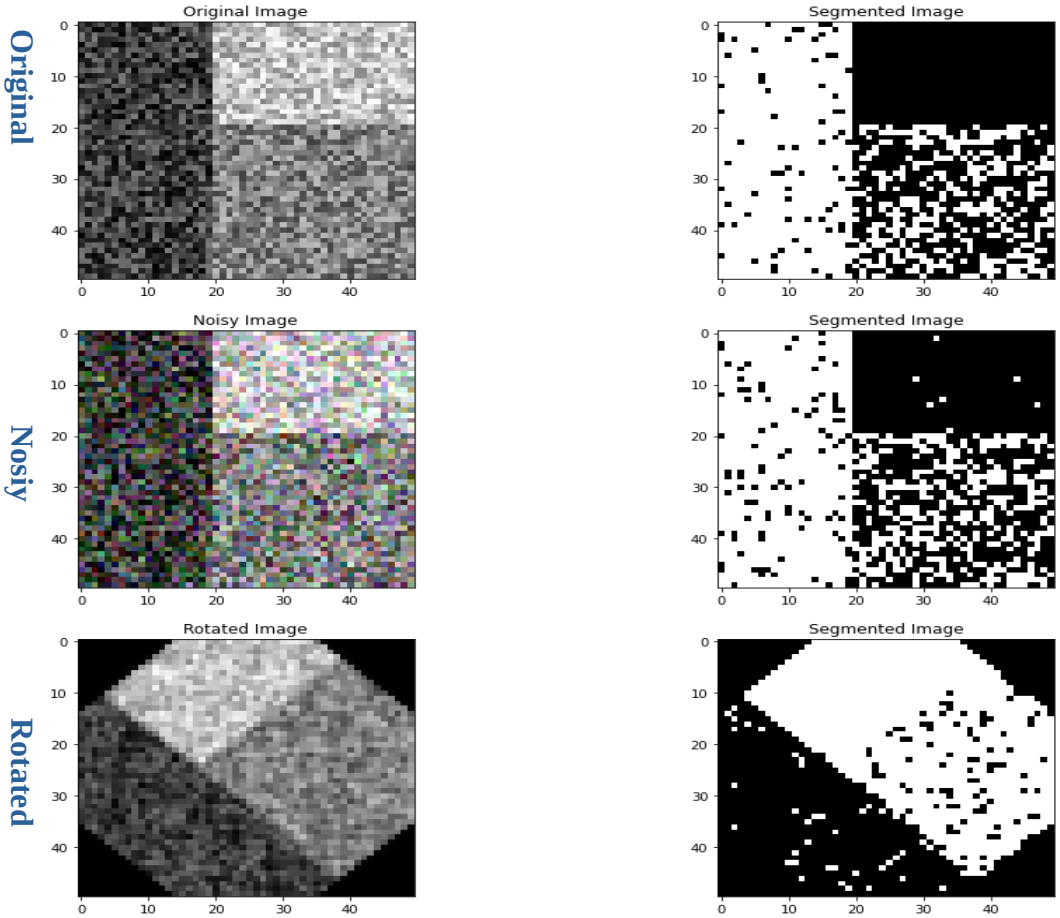

Rotated



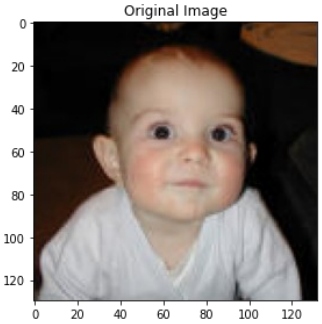



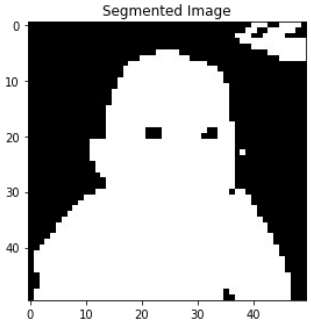
## (b) Using color for Feature Similarity :

- For Feature similarity term, h s v are calculated for each pixel
- Feature vector of each pixel calculated using F(i) = [v, v*s*sin(h),v*s*cos(h)]
- Weights[i][j] = Feature Similarity term calculculated from F(i) and F(j) of the pixels i and j.
- Here also, Spatial Proximity is ignored for better results
- Weight matrix and Eigenvalue computation is taking longer time as norm is involved while calculating the Feature Similarity term
- **Results**

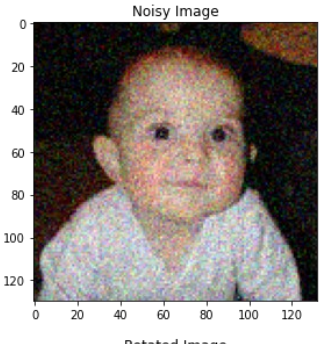


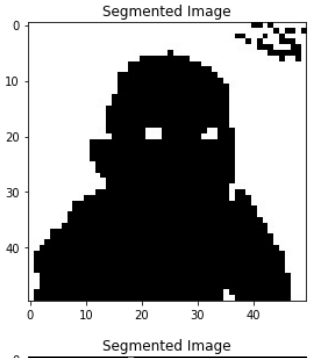
Original



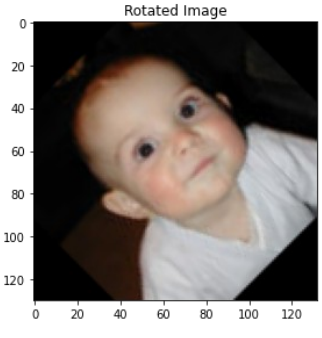



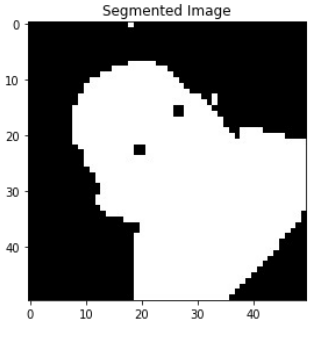

Nosiy



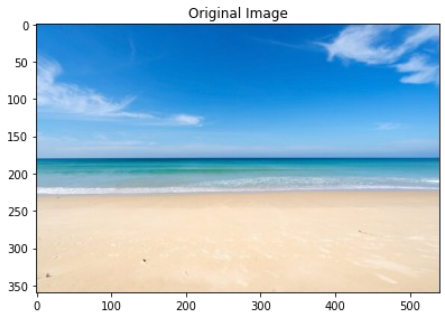



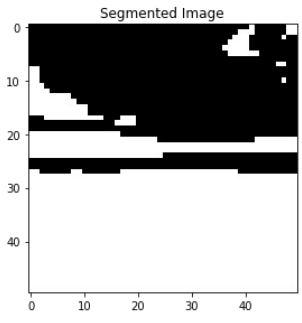

Rotated



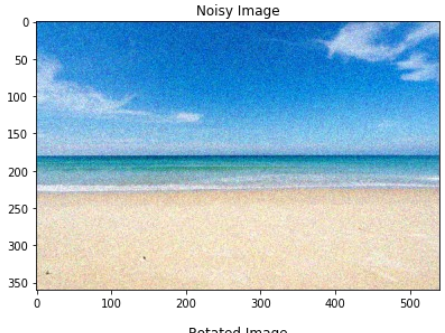



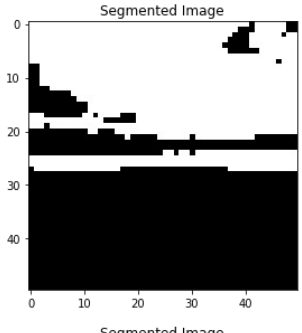

Original



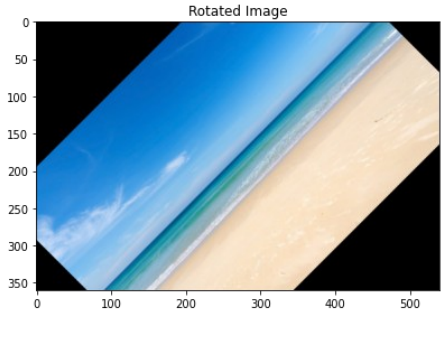



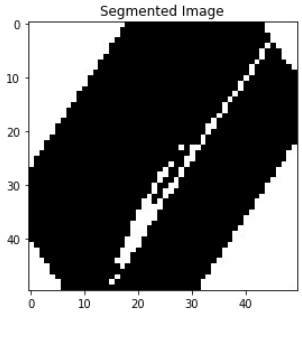

Nosiy



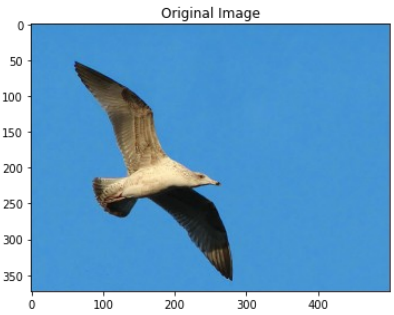



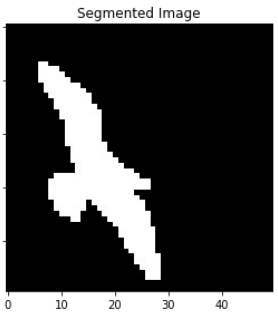

Rotated



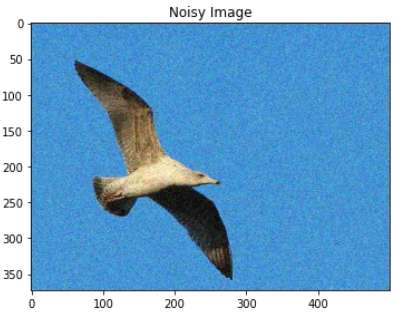



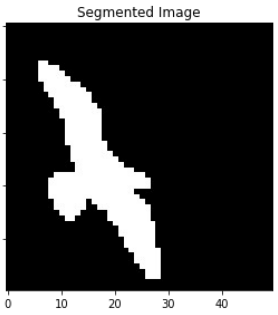
Original



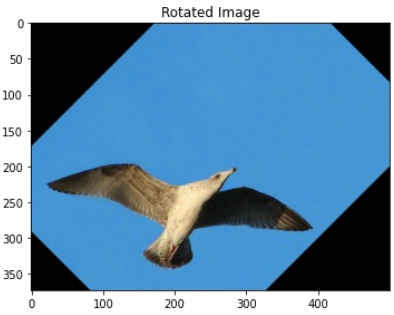




Nosiy



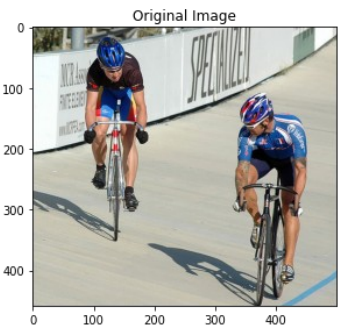




Rotated



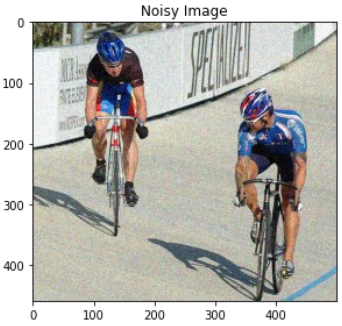


Original



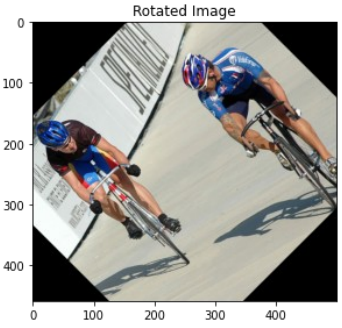


Nosiy





Rotated




# Q2) Fully Convolutional Networks

## (a) FCN_ResNet50:

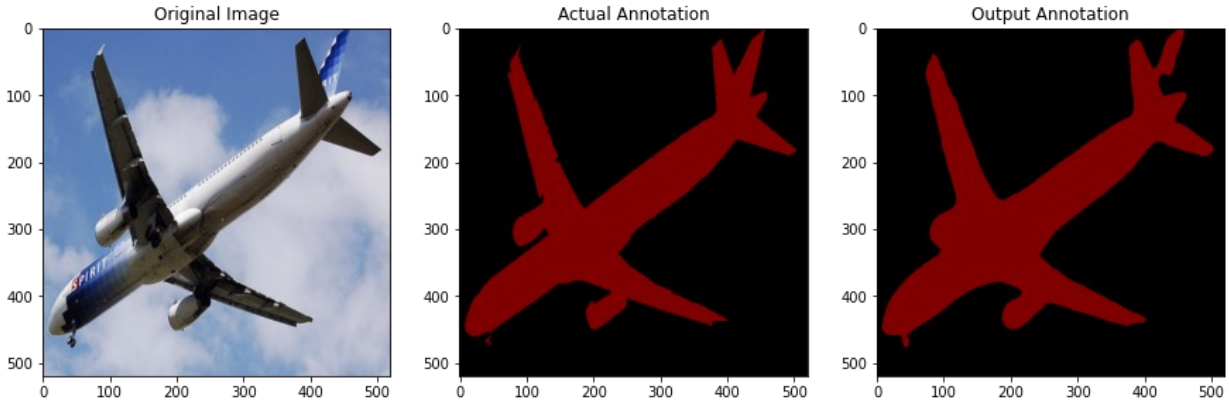
- Custom Dataset is created which operates differently for different mode (train/val/test). Data is taken as per the text files that are given. Dataset sizes are `{'train': 209, 'val': 213, 'test': 210}`
- All images and masks are resized to (520, 520) and normalized to ImageNet Dataset's mean and standard deviation
- FCN_ResNet50 model is used with pretrained weights.
- Model ouputs a (21, 520, 520) array (in this case) and we take argmax and thus convert it to (520, 520) with its pixel values as predicted classes. We apply Palette on this array for visualization purpose
- Pixel wise accuracy is calculated for each image and averaged over all the images
- MeanIOU accuracy is calculated for every class for every image and and averaged over all classes and all images
- test_4 and test_5 images of Ncut problem are tested with the model and very good results are obtained even with rotation and gaussian noise
- Applying this model is very straight forward once you write Custom Dataset for dataloaders and Palette for visualization
- **Results**

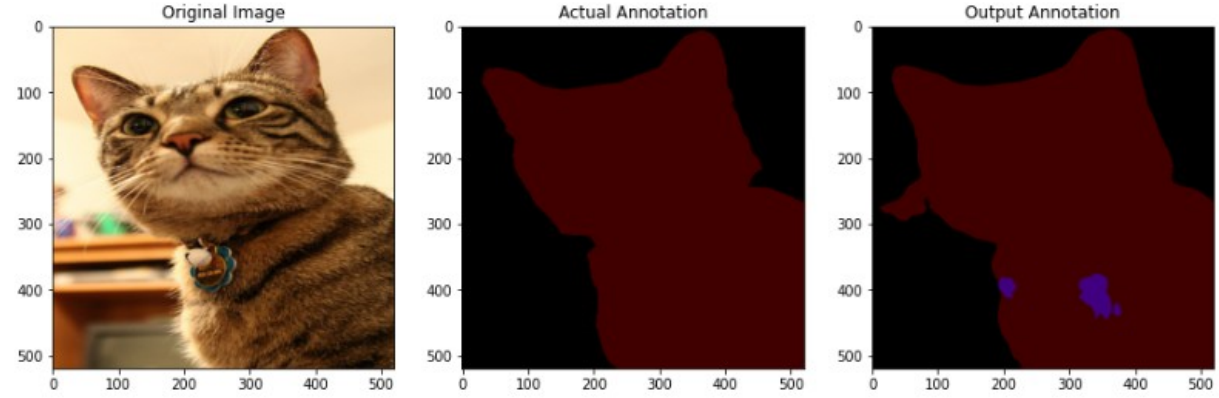
  Pixel Wise Accuracy: `91.04458474218089`

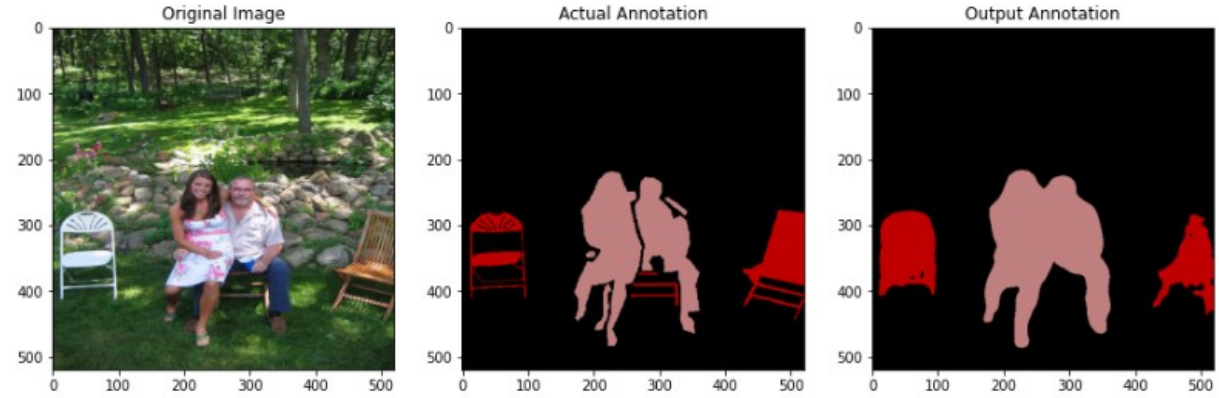
  MeanIOU Accuracy: `92.1655431827674`

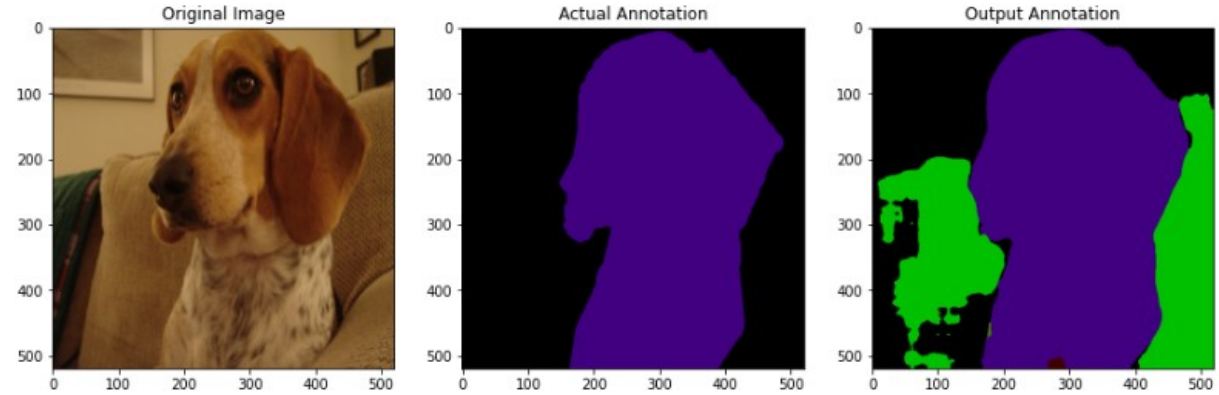
  test_4 and test_4 image visualizations are very good with FCN_ResNet50 model

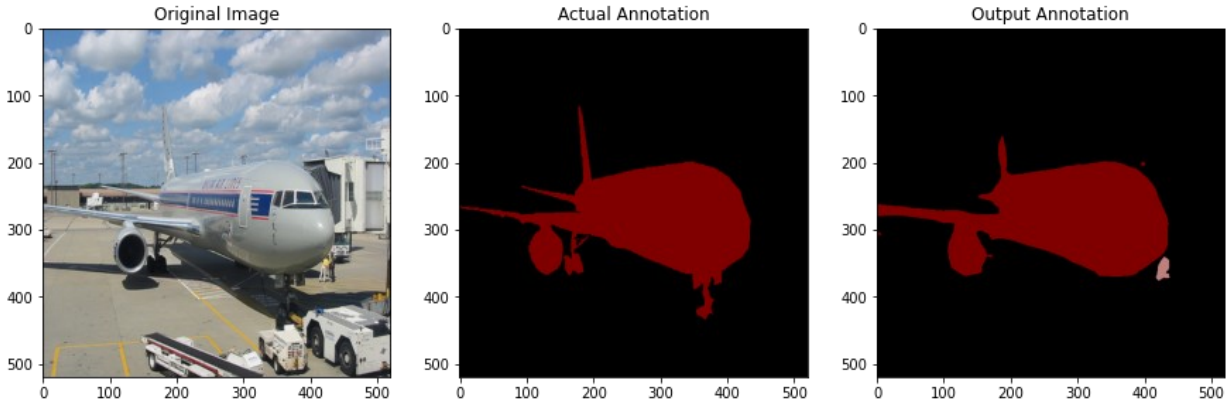
  Few images that are visualized are:

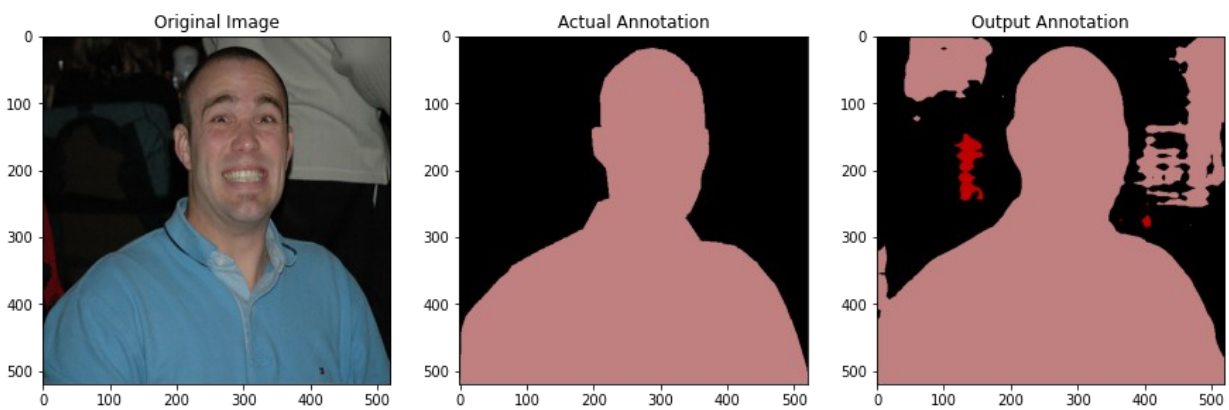



Original Image
Actual Annotation
Output Annotation

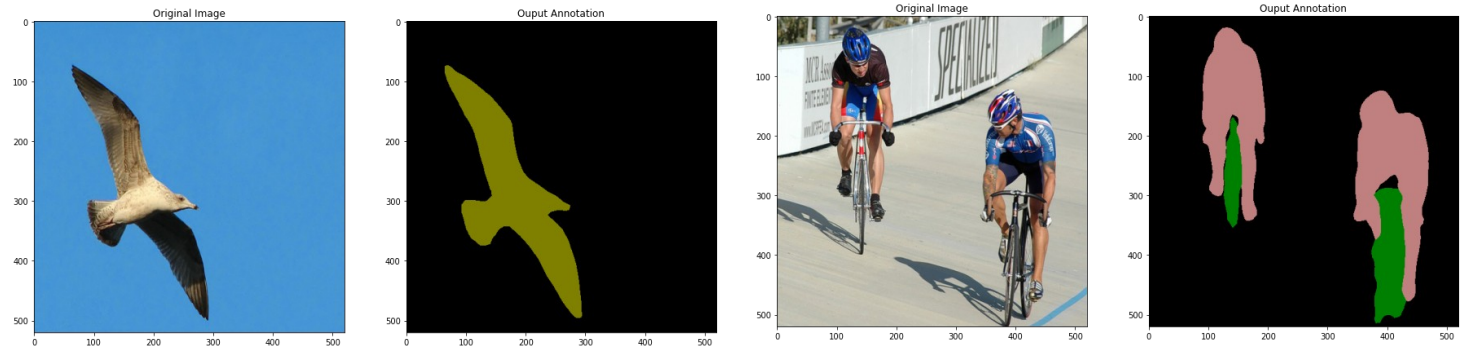

Original Image
Actual Annotation
Output Annotation


Original Image
Actual Annotation
Output Annotation

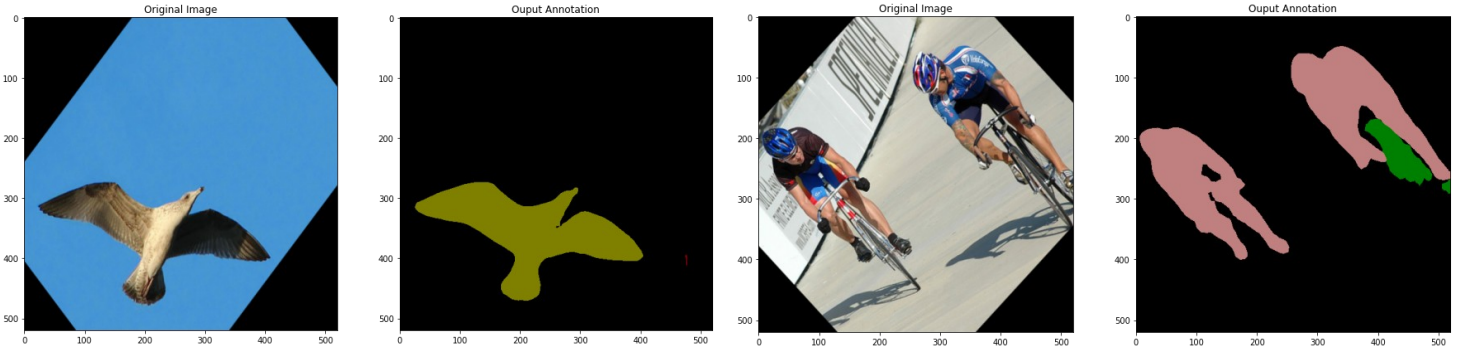

Original Image
Actual Annotation
Output Annotation


Original Image
Actual Annotation
Output Annotation

## Results on test_4 and test_5 images of Ncut problem:




### Rotated:




### Noisy:


## (b) FCN_MobileNetV2:

- Custom Dataset is created which operates differently for different mode (train/val/test). Data is taken as per the text files that are given. Dataset sizes are `{'train': 209, 'val': 213, 'test': 210}`
- All images and masks are resized to (224, 224) and normalized to ImageNet Dataset's mean and standard deviation
- InterpolationMode.NEAREST is used while resizing the mask so that the colour corresponding to a label does not change
- MobileNetV2 model is loaded with pretrained weights and the last FC layer is removed and 5 blocks of (Conv, BN, ReLU) layers are added. Weights for these convolutional layers are initialized using bilinear interpolation

- An output size of (21, 224, 224) is obtained in the last layer. Here is a brief summary of the layers that are added:

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
          Conv2d-154          [-1, 1280, 7, 7]         409,600
     BatchNorm2d-155          [-1, 1280, 7, 7]           2,560
           ReLU6-156          [-1, 1280, 7, 7]               0
  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
----------------------------------------------------------------
```

The first three layers are existing in the MobileNetV2 and an FC layer is removed after this layer and the remaining layers are added

- Trained the model on provided training data using the following hyperparameters

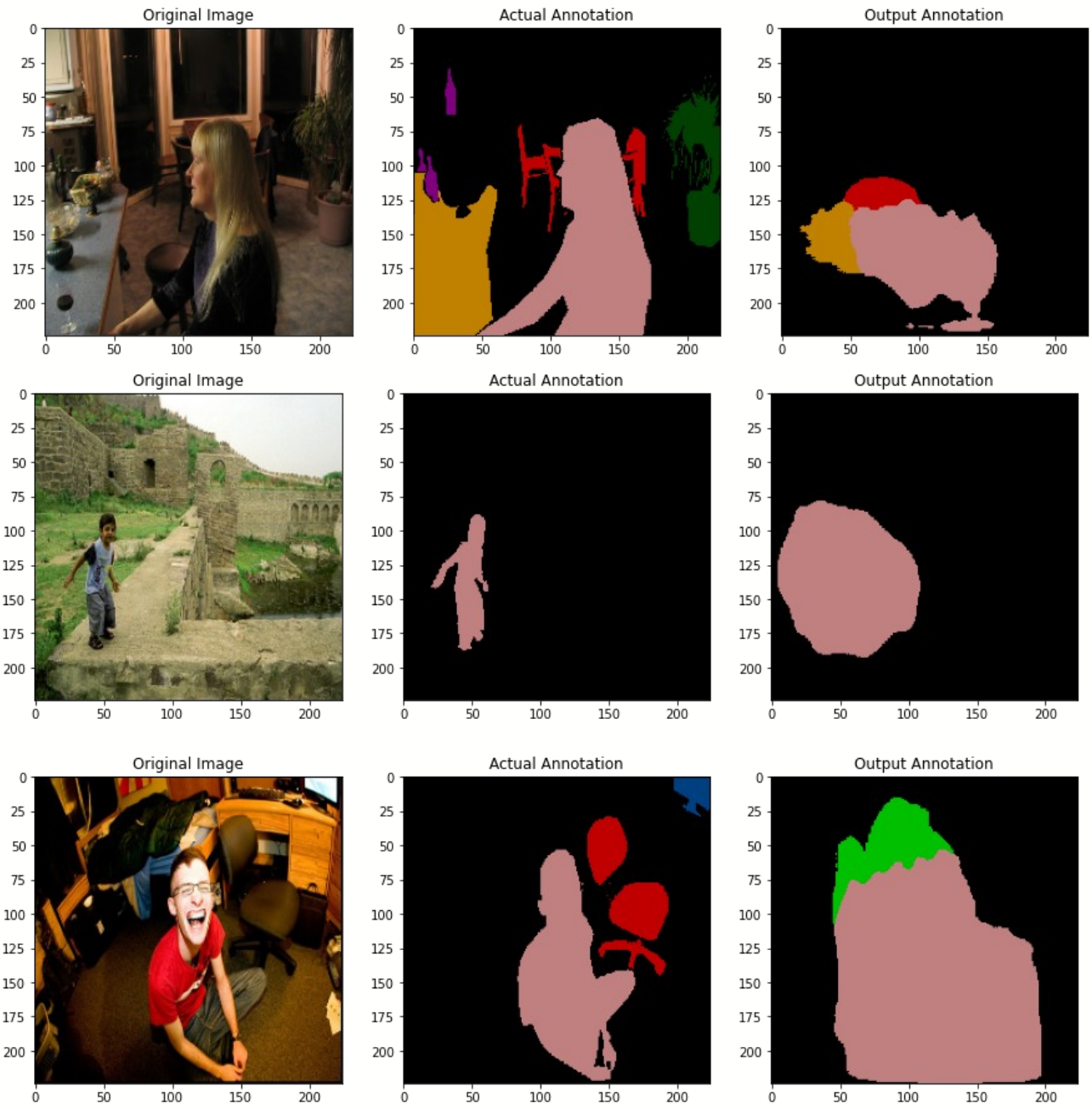
```
criterion = torch.nn.CrossEntropyLoss(ignore_index = 255)

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)
```

- Colour Palette is added to visualize the outputs and visualization is done in a similar way
- **Results:**

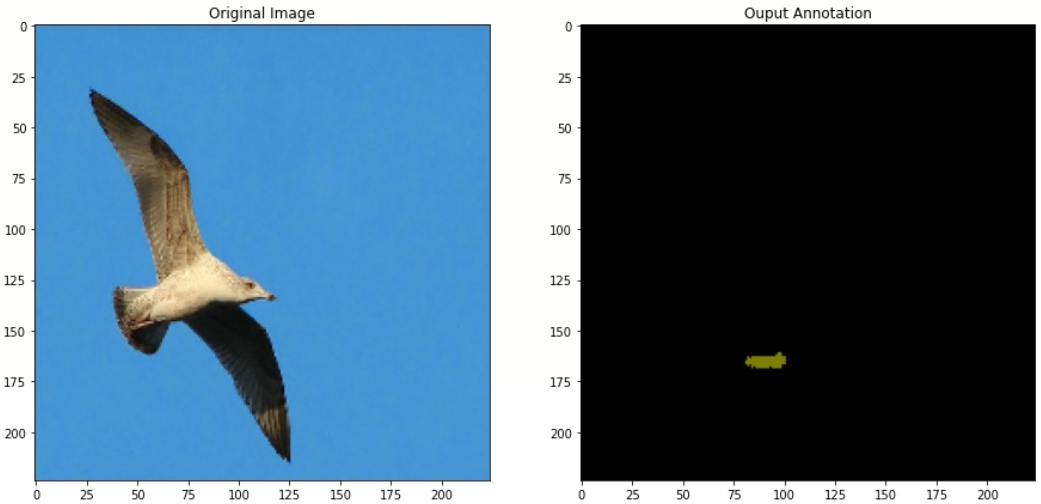
  Pixel Wise Accuracy on test data : `77.15387550109331`

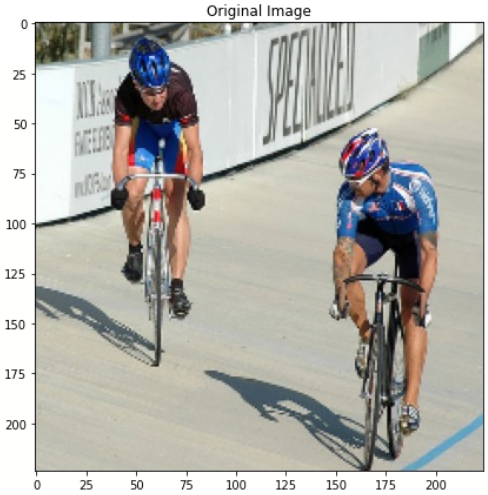
  MeanIOU Accuracy on test data: `86.16131539517966`

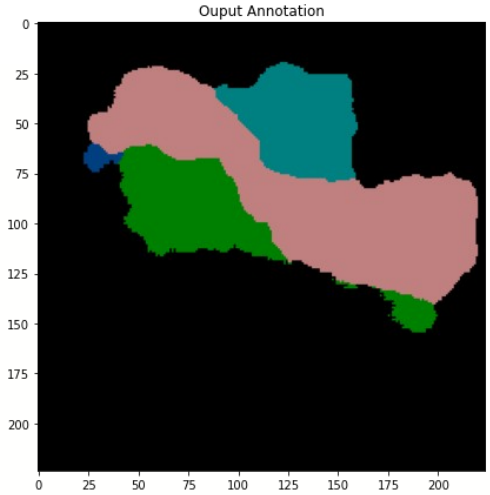
Here are the few outputs that are visualized:



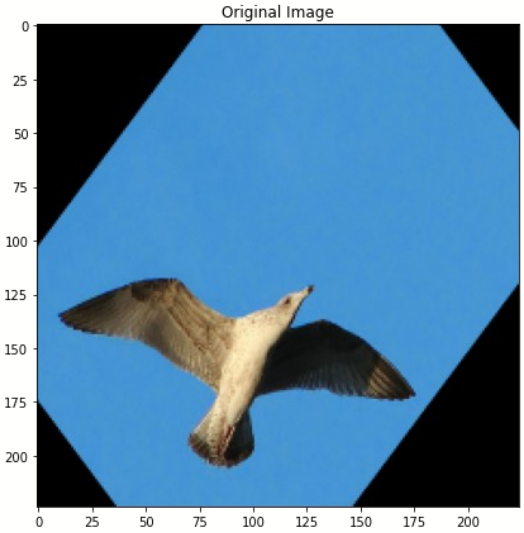

## Results on test_4 and test_5 images of Ncut problem:

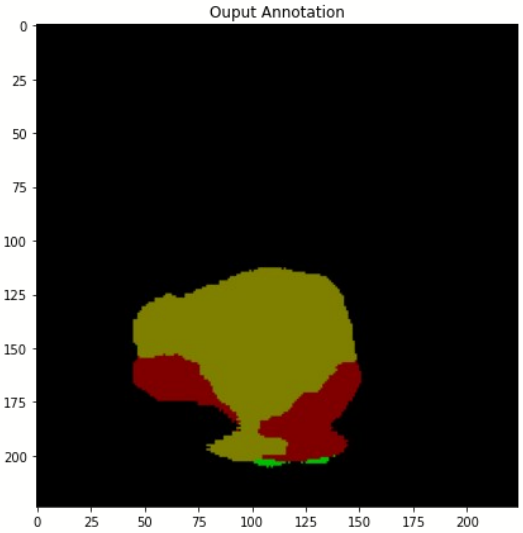


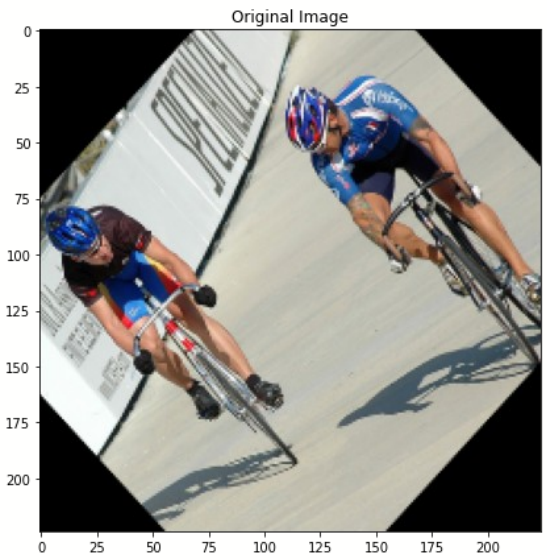


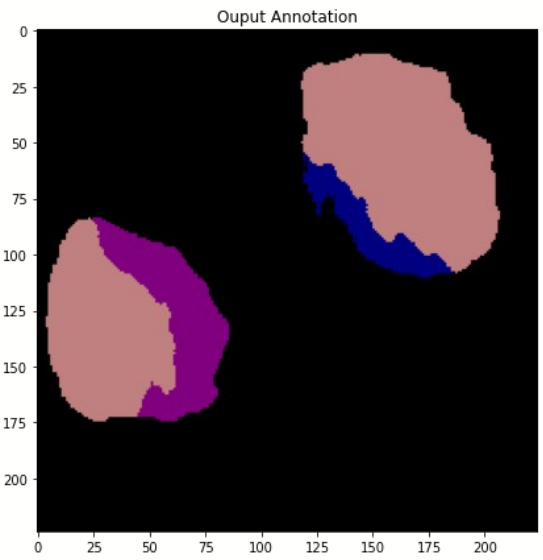




## Rotated:

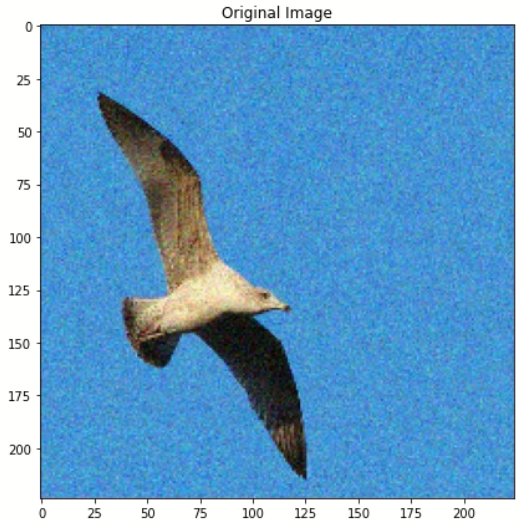


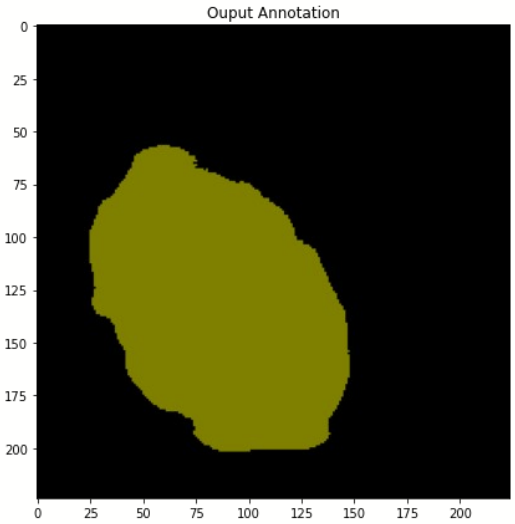



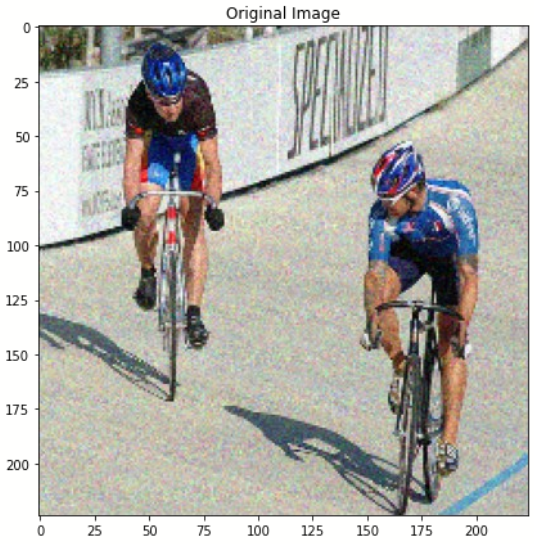



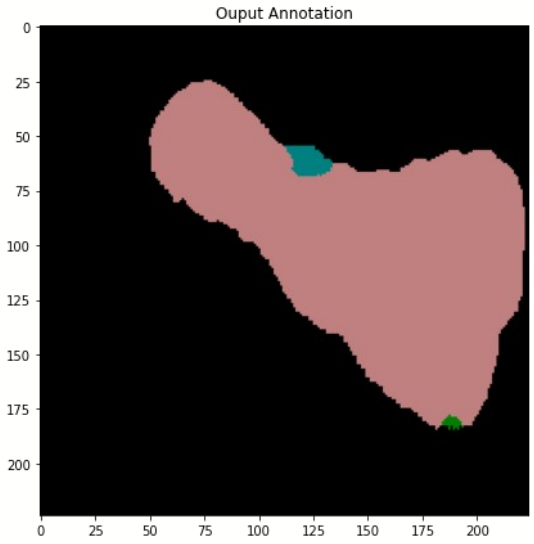




## Noisy:




**Comparison:**

We can observe that the visualizations obtained from the FCN_ResNet50 model are far better than that of FCN_MobileNetV2 model. Here are some of the reasons:

- No skip connections in MobileNetV2 backbone
- FCN_MobileNetV2 is trained on very less data when compared to FCN_ResNet50

**References:**


https://d2l.ai/chapter_computer-vision/fcn.html

https://github.com/sagieppel/Fully-convolutional-neural-network-FCN-for-semantic-segmentation-with-pytorch/blob/master/NET_FCN.py

https://poutyne.org/examples/semantic_segmentation.html

https://pytorch.org/vision/main/_modules/torchvision/datasets/voc.html#VOCSegmentation