AIP ASSIGNMENT - 2

Q1)

N-cut using Intensity similarity measure:

1)The intensity between two pixels is taken as a similarity measure

2)At each pixel the intensity is the average intensity of three channels(R,G,B)

3)The similarity of pixel ‘i’ and pixel ‘j’ is inversely proportional absolute difference of intensities at pixel i and j

4)The weight matrix is calculated using both feature proximity term and spatial proximity term as given below

```
weights[i][j] = exp(-(inten_dist**2)/sigma_i) * exp(-(distance*2)/sigma_x)
```

5)Used cutoff distance(euclidean distance) for weights, so that the weights are calculated only if the distance between pixels < cutoff distance ,else weights are set to zero.

6)Had to vary threshold distance for every test image to get better results

7)Tried various thresholds like mean,median and 0 for ‘y’ vector(second smallest eigen vector) for better N-cut results

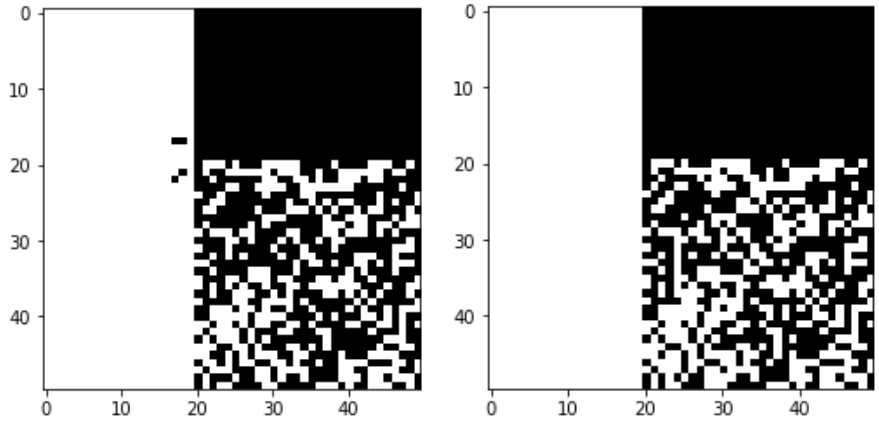
8)Some images gave better results for median threshold, while some gave better results for slightly more than/less than zero threshold(eg:0.0005 or -0.0005)

9)test 2,3,4,5 Images are resized to run in reasonable amount of time

10)Used sigma_i = 10,sigma_x = 15 for all test images

Results:

a) *test1.jpg*

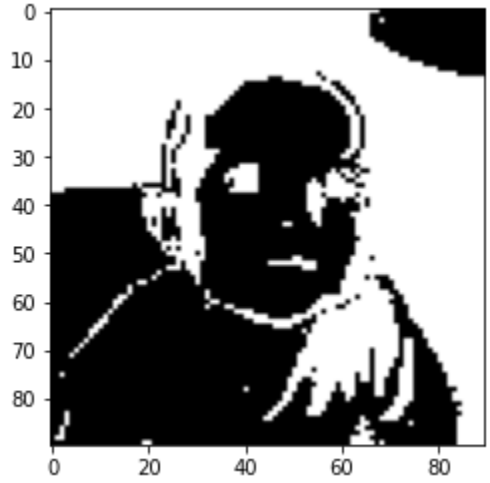


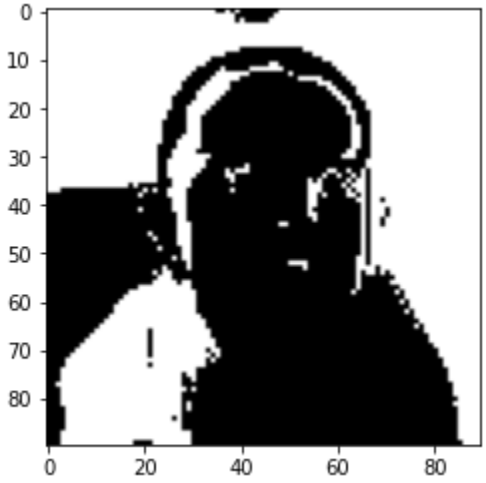
Parameters:

threshold distance = 10 threshold distance = 10

eigen vector threshold = 0.0002 eigen vector threshold = -0.0006

b) *test2.jpg (resized to 90*90)*

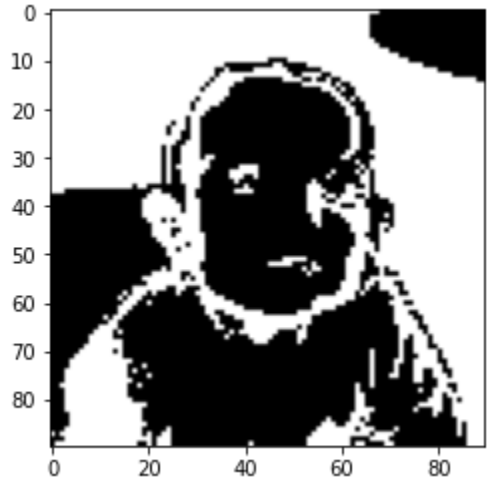


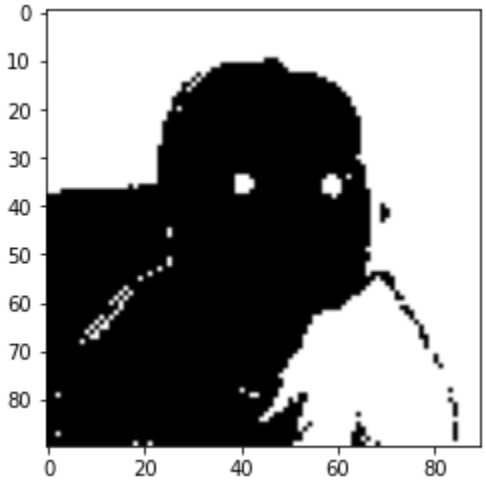


Parameters:

threshold distance = 15
eigen vector threshold = 0.0002

threshold distance = 20
eigen vector threshold = -0.0006

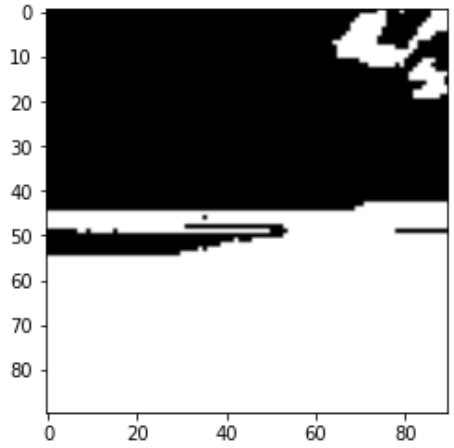


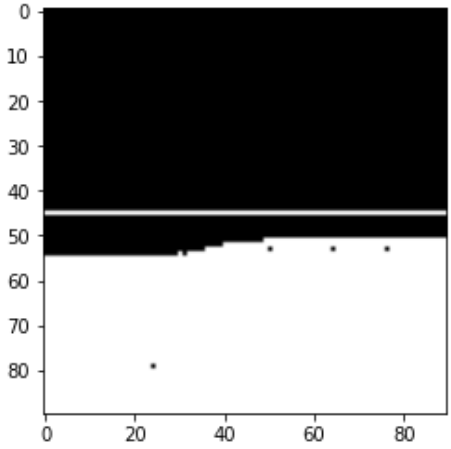


threshold distance = 10
eigen vector threshold = 0.0002

threshold distance = 15
eigen vector threshold = median

c)*test3.jpg (resized to 90 * 90)*

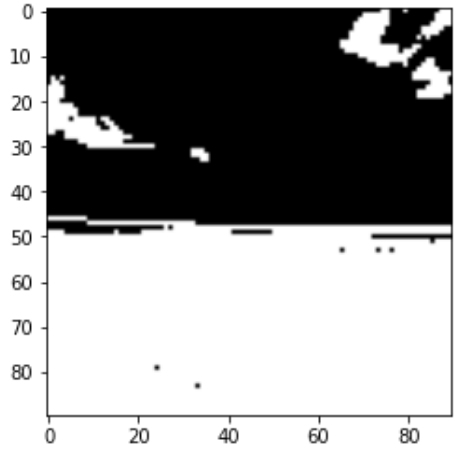


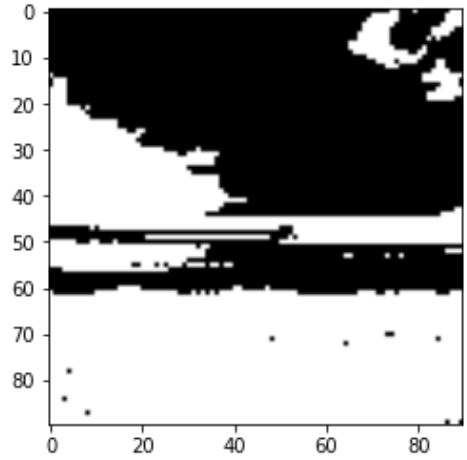


Parameters:

threshold distance = 20
eigen vector threshold = 0.0002

threshold distance = 15
eigen vector threshold = 0

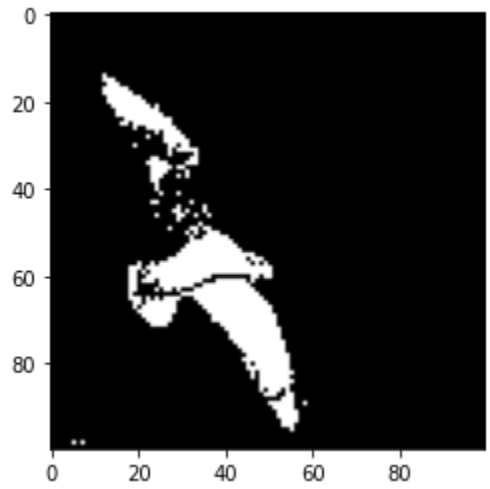




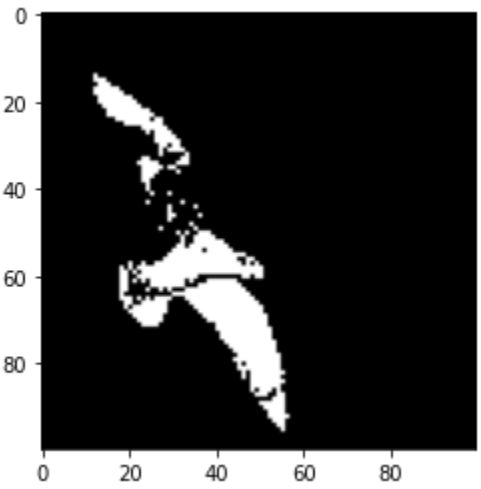
threshold distance = 20
eigen vector threshold = -0.0002

threshold distance = 10
eigen vector threshold = median

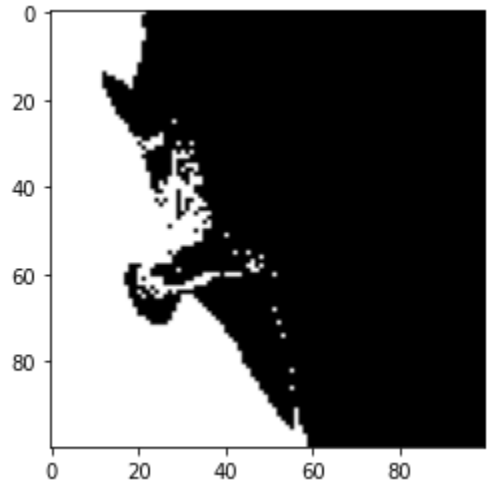
4)*test4.jpg (resized to 90 * 90)*



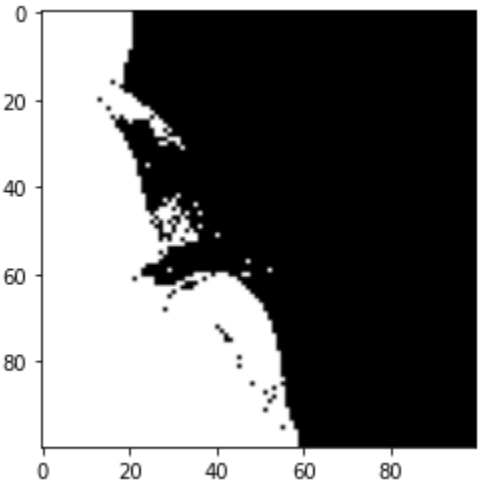
threshold distance = 20
eigen vector threshold = 1e-5
N-cut value=0.000116



threshold distance = 20
eigen vector threshold = 1e-4
N-cut value=1.19e-05

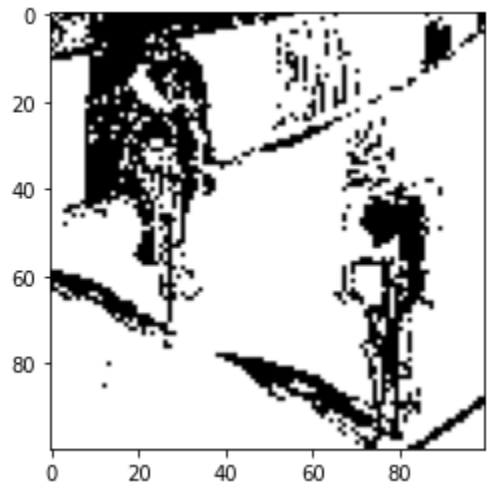


threshold distance = 15
eigen vector threshold = 0.00002
N-cut value=0.00392

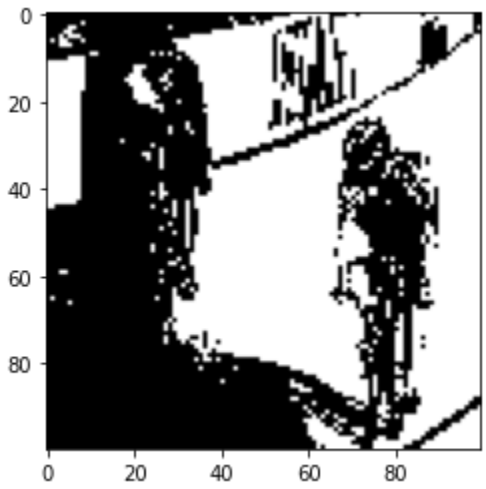


threshold distance = 10
eigen vector threshold = (1e-6) * 5
N-cut value=0.00398

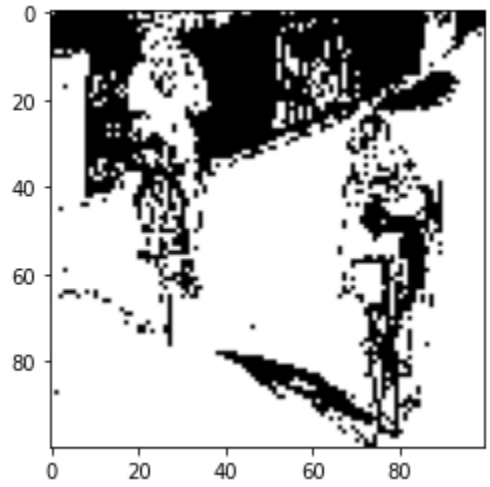
e)*test5.jpg (resized to 100 * 100)*



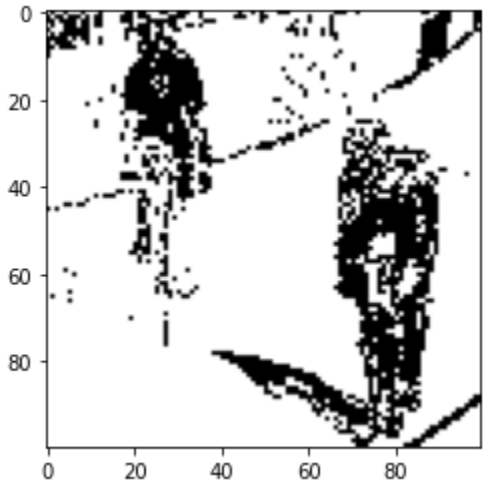
threshold distance = 20
eigen vector threshold = -1e-6
N-cut value=0.00023



threshold distance = 20
eigen vector threshold = median
N-cut value=0.0051



threshold distance = 15
eigen vector threshold = -1e-4
N-cut value=0.00147

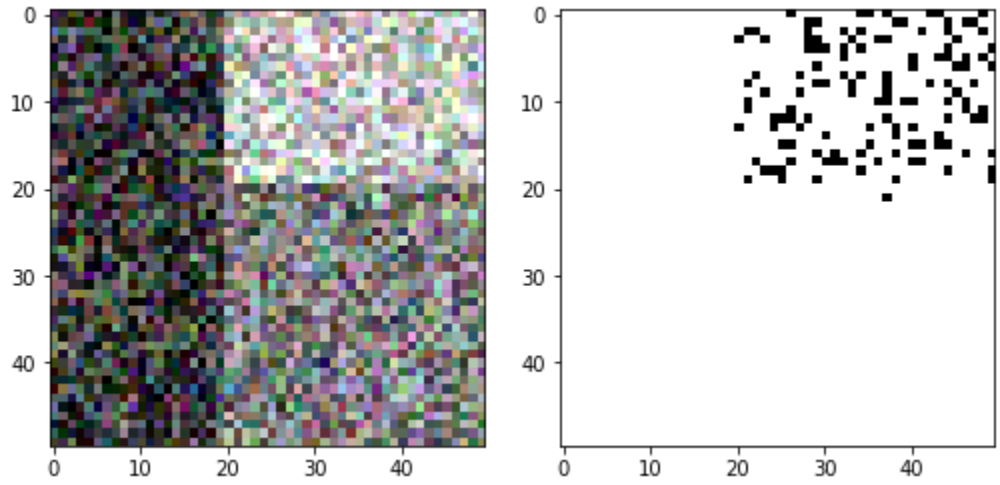


threshold distance = 10
eigen vector threshold = -1e-4
N-cut value=0.000891

GAUSSIAN NOISE ADDED IMAGES:

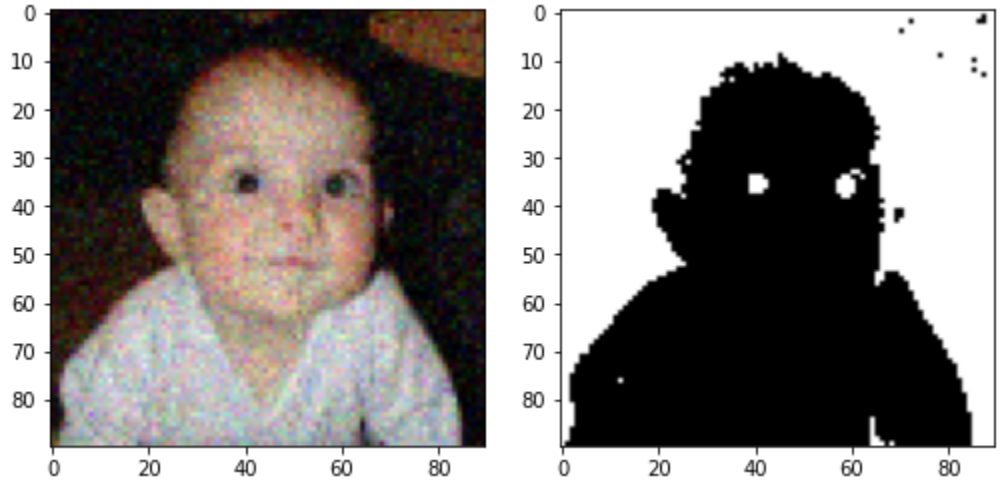
a)test1.jpg

```
Ncut value = 0.0496
Eigen vector threshold=1e-5
```



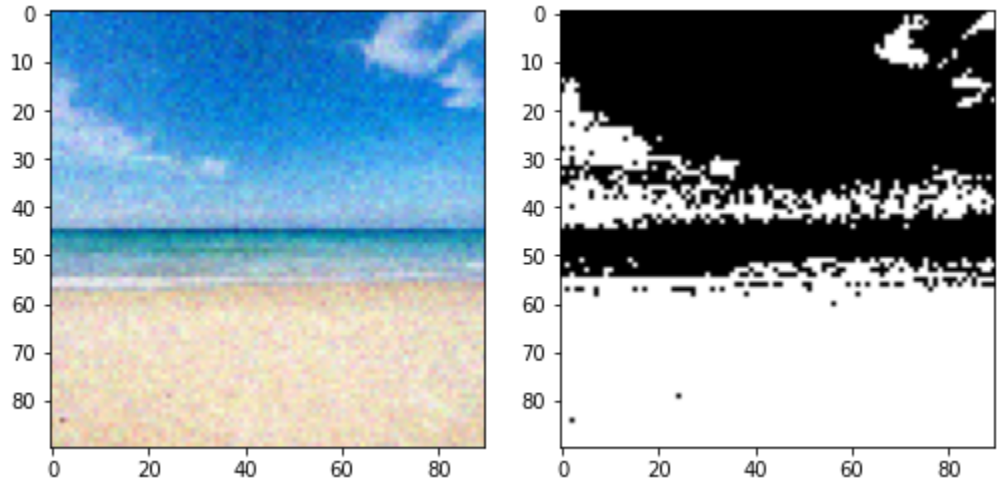
b)*test2.jpg*

```
Ncut value = 0.000231
Eigen vector threshold=1e-4
```



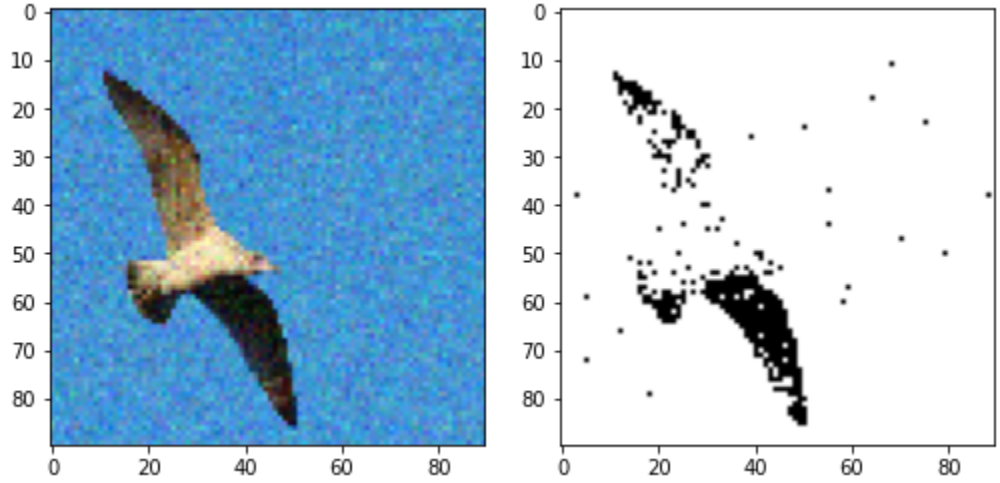
c)test3.jpg

```
Ncut value = 0.013129867267962 84
Eigen vector threshold=median
```



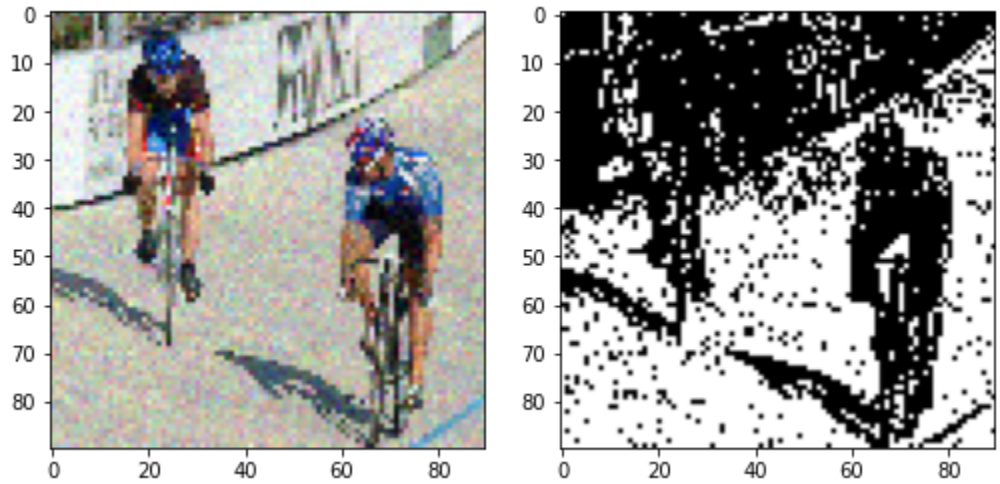
d)test4.jpg

```
Ncut value = 7.0012e-05
Eigen vector threshold= 0.003
```



e)test5.jpg

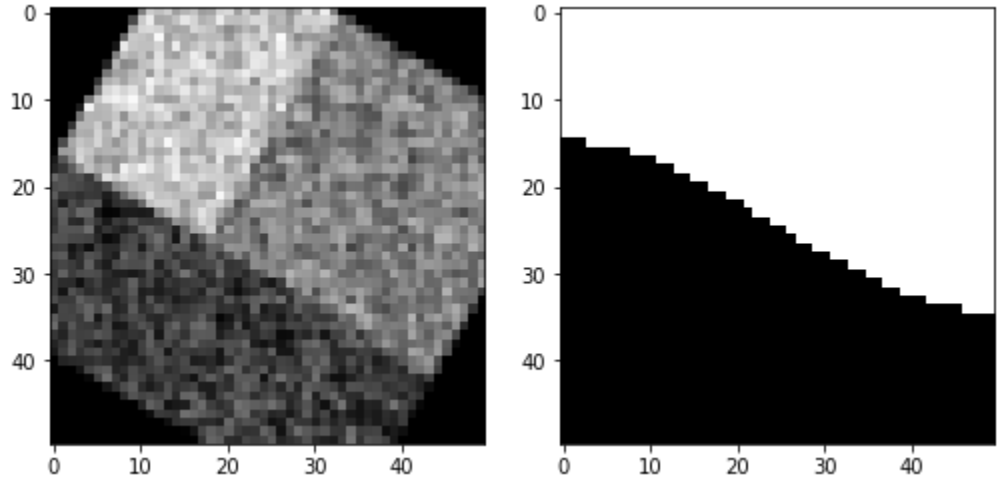
```
Ncut value = 0.0519
Eigen vector threshold= (1e-5)*2
```



ROTATED IMAGES

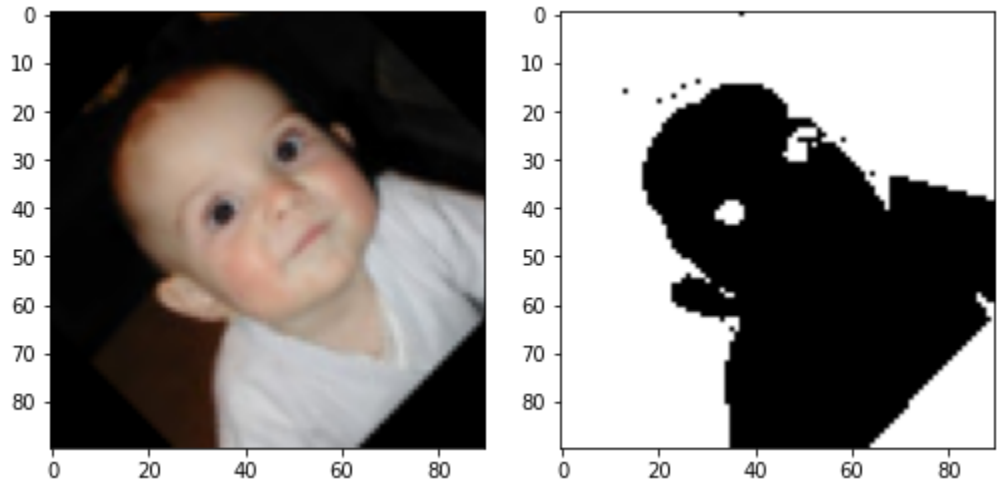
a)test1.jpg

```
Ncut value = 0.04971
Eigen vector threshold= 1e-5
```



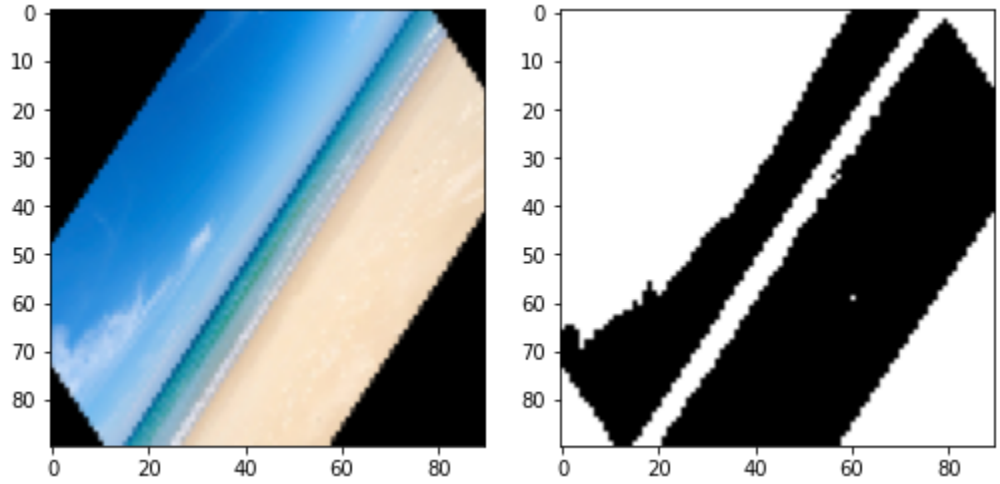
b)test2.jpg

```
Ncut value = 0.000136
Eigen vector threshold= (1e-4)
```



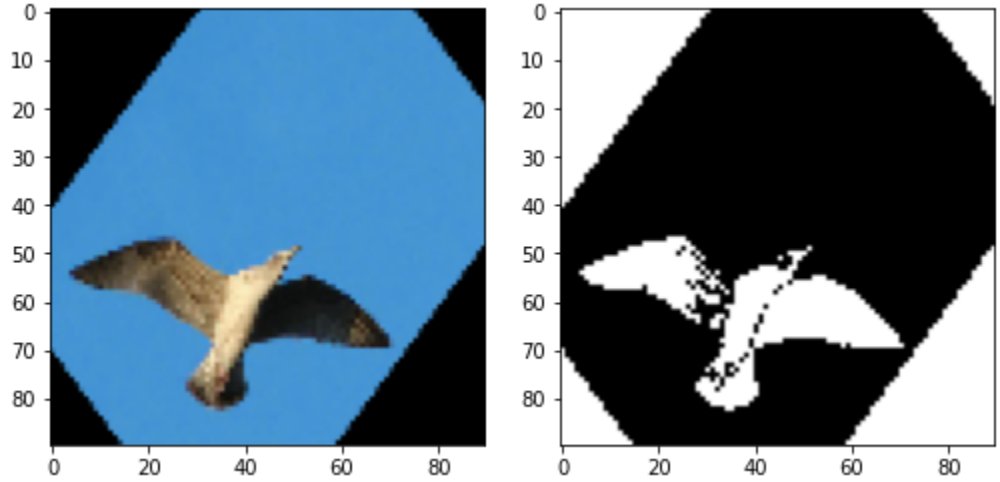
c)test3.jpg

```
Ncut value = 1.0880312801501268e-05
Eigen vector threshold= median
```



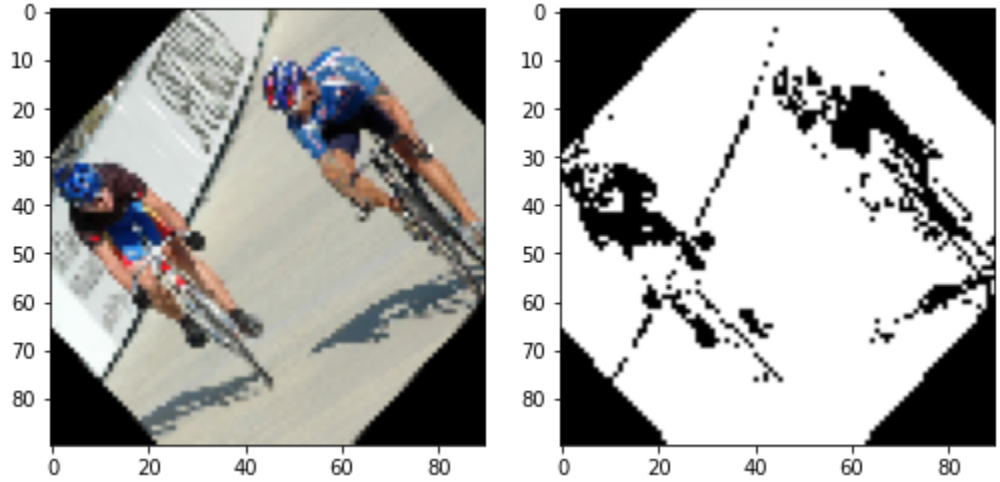
d)test4.jpg

```
Ncut value = 3.807e-05
Eigen vector threshold= 1e-4
```



d)test5.jpg

```
Ncut value = 0.00046
Eigen vector threshold = 1e-6
```



II)N-cut using Color similarity measure:

1)The color difference between two pixels is taken as a similarity measure

2)The feature used for color at each pixel is

```
F(i) = [v, v*s*sin(h),v*s*cos(h)]
```

Where h,s,v are hsv values of the pixel

(This is from https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf)

3)The similarity of pixel 'i' and pixel 'j' is inversely proportional absolute difference of intensities at pixel i and j

```
color_dist=norm(F(i) - F(j))
```

4)The weight matrix is calculated using both feature proximity term and spatial proximity term as given below

```
weights[i][j] = exp(-(color_dist**2)/sigma_i)
```

5)Used rgb2hsv for implementation

(Source :https://scikit-image.org/docs/stable/auto_examples/color_exposure/plot_rgb_to_hsv.html)

6)Had to vary threshold distance for every test image to get better results

7)Tried various thresholds like mean,median and 0 for 'y' vector(second smallest eigen vector) for better N-cut results

8)Some images gave better results for median threshold, while some gave better results for slightly more than/less than zero threshold(eg:0.0005 or -0.0005)

9)test 2,3,4,5 Images are resized to 50*50 as color similarity was taking more time compared to intensity similarity run in reasonable amount of time

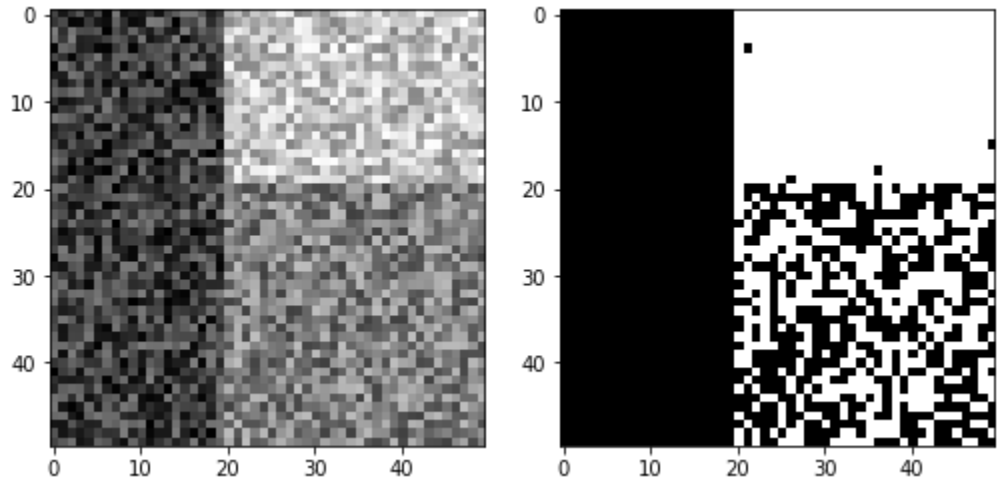
10)Used sigma_i = 10,sigma_x = 15 for all test images

11)The segmentation got affected by resizing but had to do it to run in less time

Results:

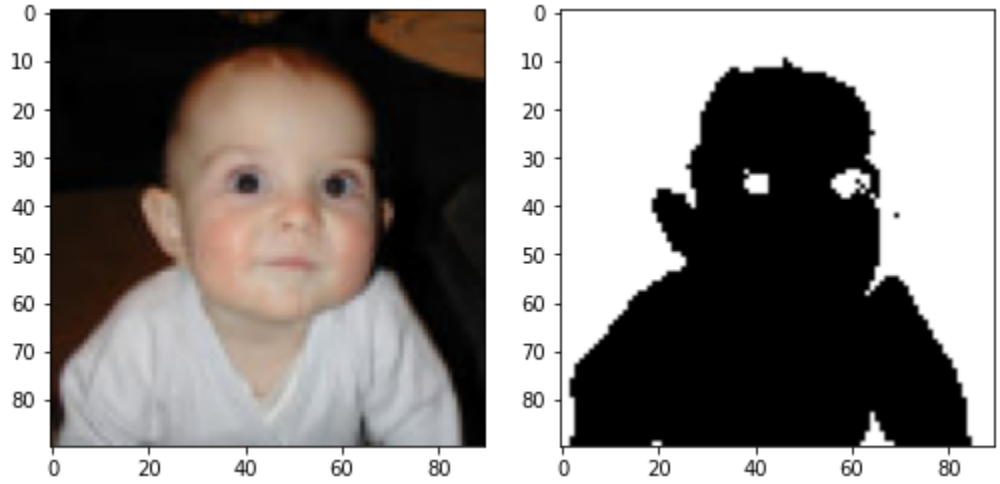
a)test1.jpg

```
Ncut value = 0.491766
Eigen vector threshold = 0.0005
```



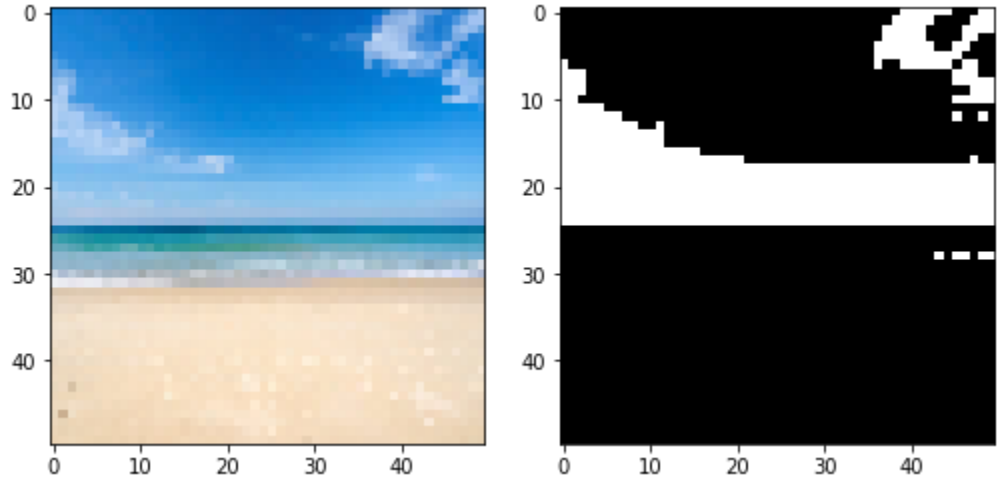
b)test2.jpg

```
Ncut value = 0.3134
Eigen vector threshold = 1e-6
```



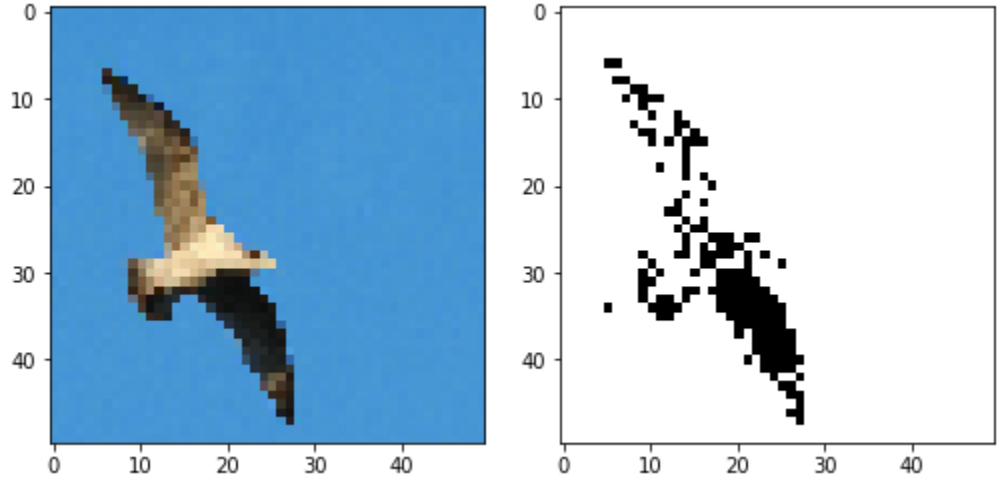
c)test3.jpg

```
Ncut value = 0.000354
Eigen vector threshold = 0.005
```



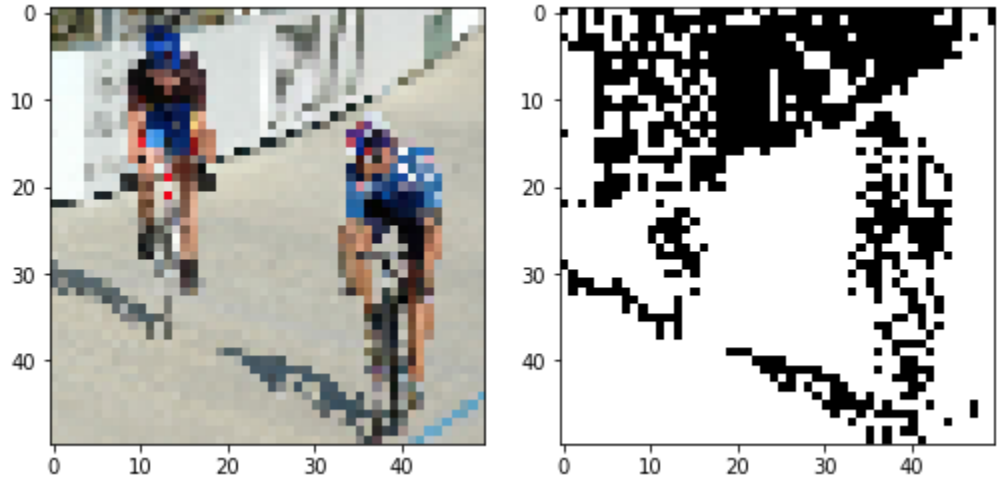
d)test4.jpg

```
Ncut value = 0.4648
Eigen vector threshold = -0.0001
```



e)test5.jpg

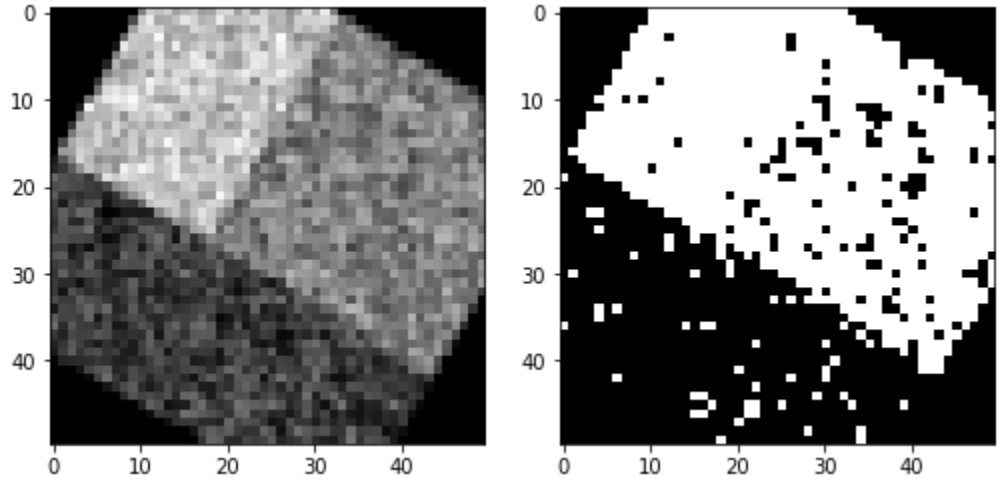
N cut= 0.38616
Eigen vector threshold = 0



ROTATED IMAGES:

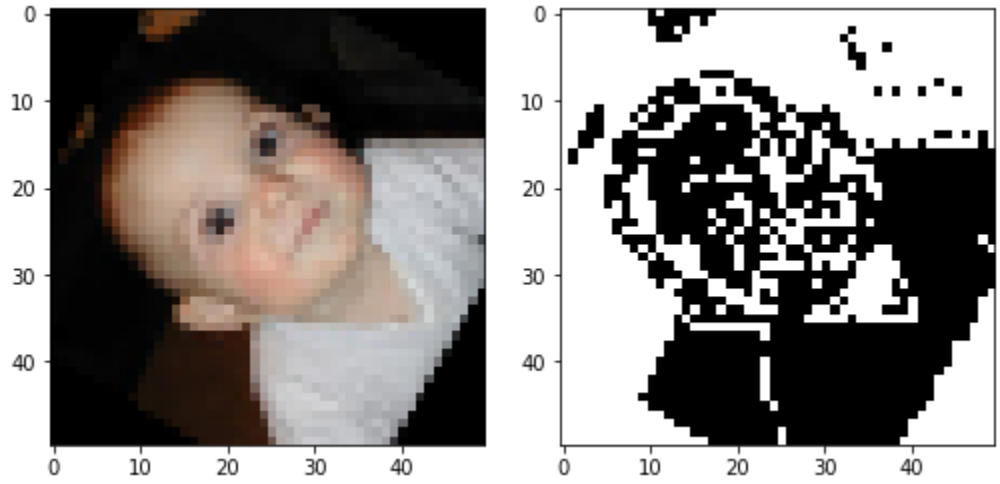
a)test1.jpg

Ncut value = 0.42697
Eigen vector threshold = median



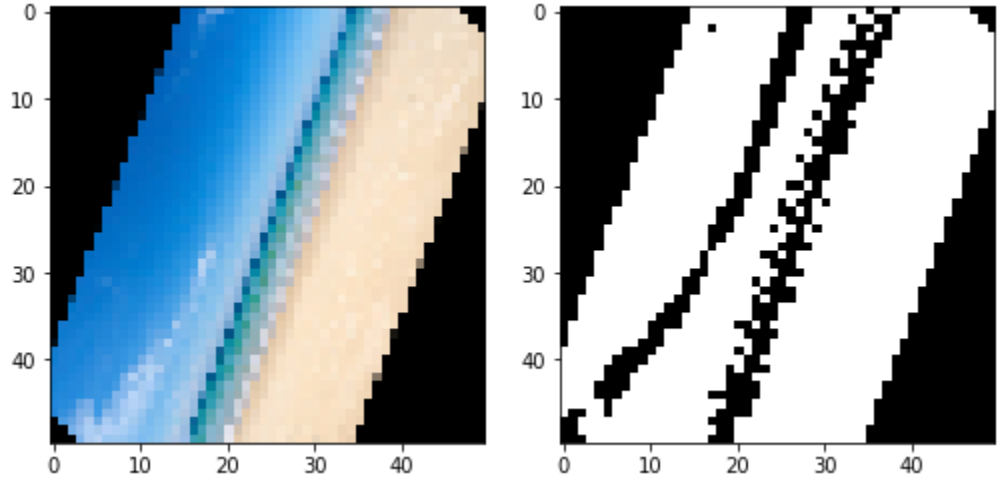
b)test2.jpg

```
Ncut value = 0.4504
Eigen vector threshold = -0.0001
```



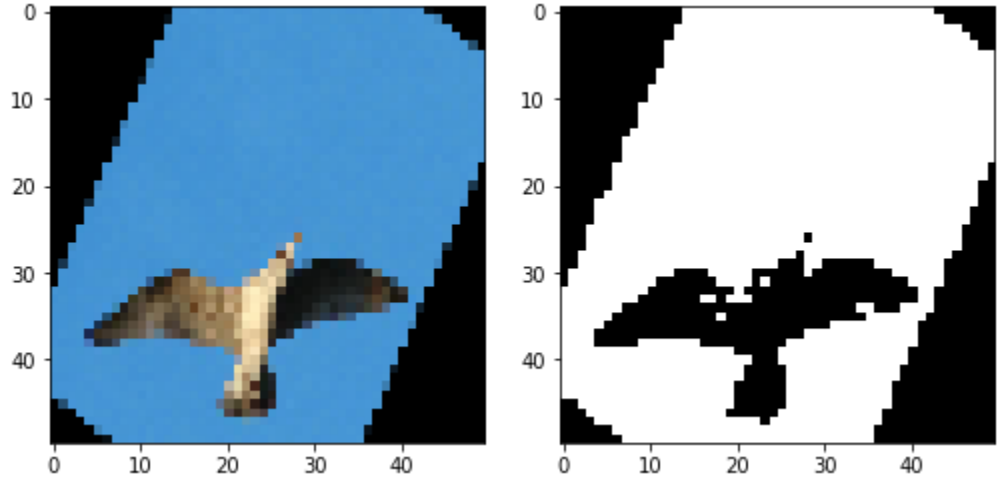
c)test3.jpg

```
Ncut value = 0.4819
Eigen vector threshold = -0.0001
```



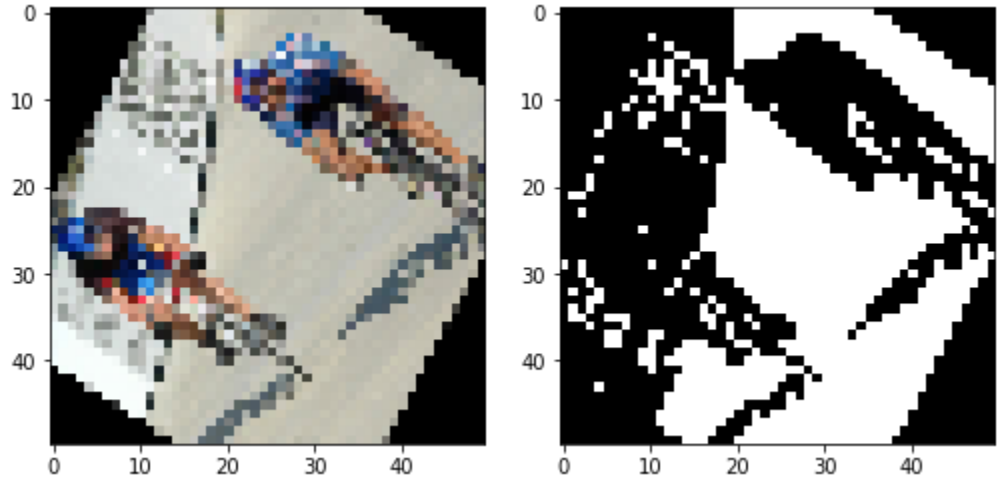
d)test4.jpg

```
Ncut value = 0.4176
Eigen vector threshold = 0.0005
```



d)test5.jpg

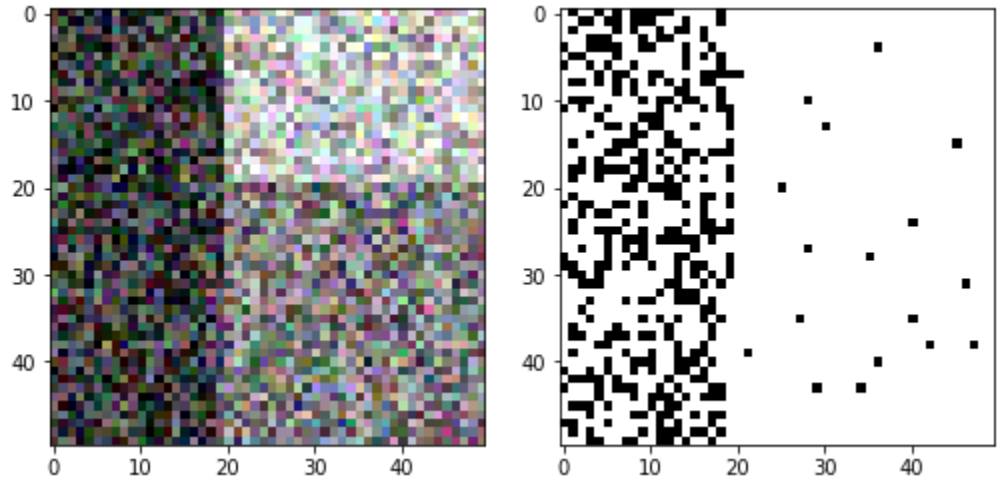
```
Ncut value = 0.486
Eigen vector threshold = median
```



GAUSSIAN NOISE ADDED IMAGES:

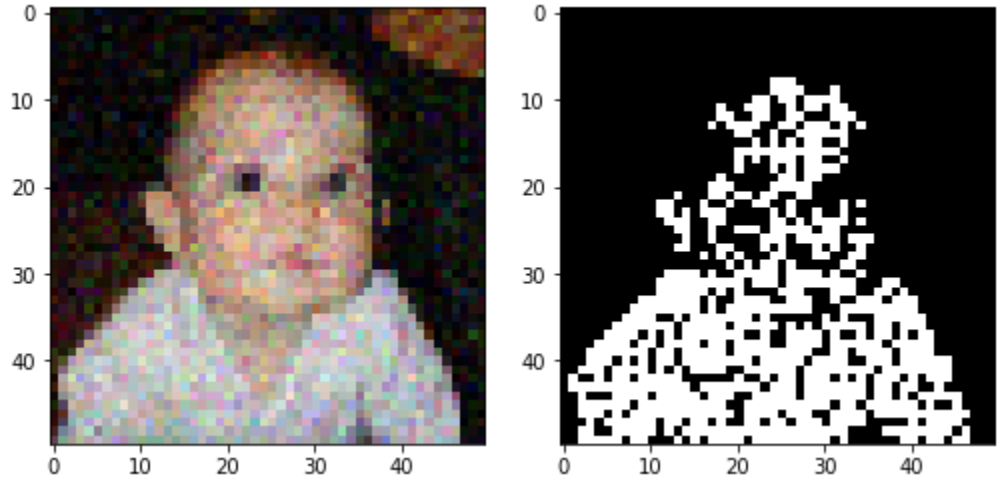
a)test1.jpg

```
Ncut value = 0.2067
Eigen vector threshold = 0.0055
```



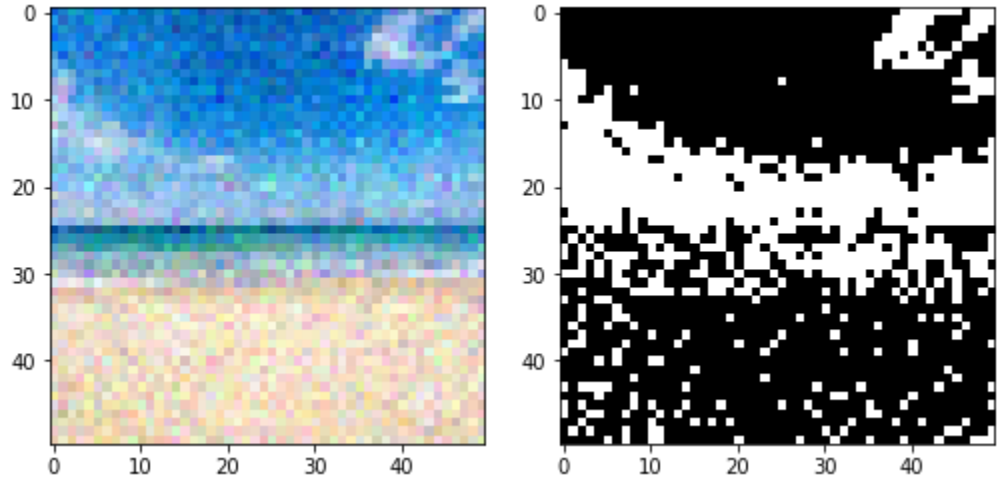
b)test2.jpg

```
Ncut value = 0.33
Eigen vector threshold = (-1e-3)4
```



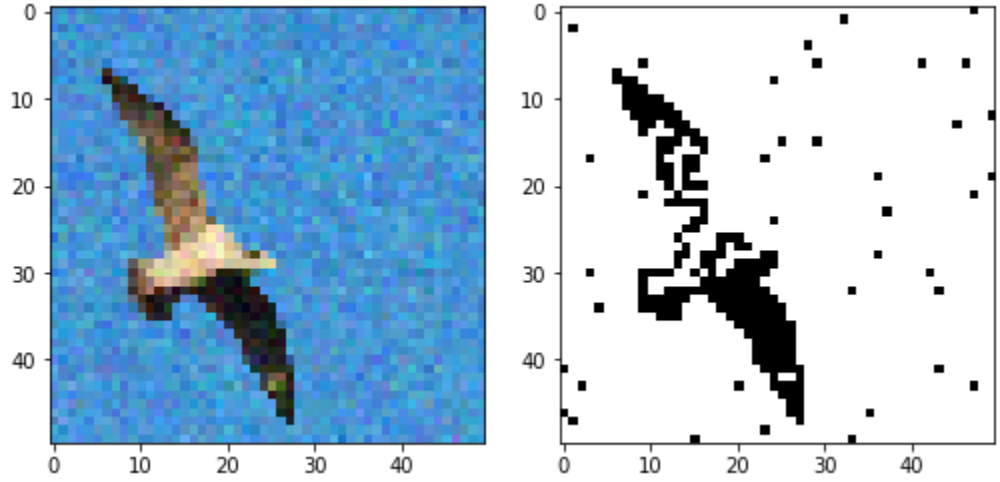
c)test3.jpg

```
Ncut value = 0.037
Eigen vector threshold = 0.001
```



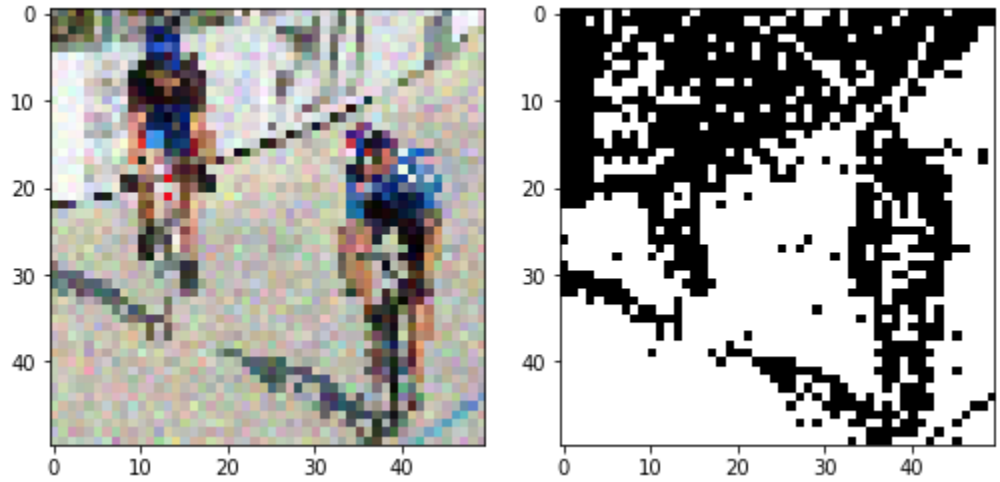
d)test4.jpg

```
Ncut value = 0.478
Eigen vector threshold = 1e-4*5
```



e)test5.jpg

```
Ncut value = 0.5018
Eigen vector threshold = 1e-4
```



Observations:

1)As the image is resized to a lower scale(had to do it to run in reasonable amount of time) , the segmentation accuracy was decreasing. This can be seen from above images as 90*90 resizing gave better segmentation than 50*50

2)Different images needed different thresholds for good segmentation and most of the thresholds were around 0 and median, rarely mean got better than median and 0.

3)Gaussian Noise images and rotated images performed slightly worse on color similarity but performed decent in intensity similarity measure(as you can see above).

Problems faced:

1)The biggest issue was running time for eigen vector calculation, had to use torch.lobpcg() for eigen vector calculation
(source: https://pytorch.org/docs/stable/generated/torch.lobpcg.html)
which uses niter argument to approximate eigen vector upto 'niter' iterations.

2)Weight matrix calculation was time taking again due to large image sizes.
For 500*500 image, the weight matrix size was 250000*250000 which was huge,hence had to resize.

Q2)

A) Segmentation using ResNet Based FCN:

1)Implemented a custom class dataset
(Source:https://towardsdatascience.com/how-to-use-datasets-and-dataloader-in-pytorch-for-custom-text-data-270eed7f7c00
for `SegmentationPascal(Dataset)`

2) Trainval is splitting using .txt files provided

3)Applied standard data transforms and loaded using data loader

4)Loaded FCN_resnet_50 model using default weights

5)Pixel wise and meanIOU was calculated for test data

6)Overall Average accuracy was calculated overall all test Images(mean of all test image pixel wise accuracies)

7)Mean IOU was calculated for each class and averaged over all classes which gives mean IOU for an Image. Overall mean IOU was calculated by averaging over all Image Mean IOUS.

8)For visualization, used a palette which maps class value into corresponding RGB in the predicted Mask.

9)No major problems encountered in resnet fcn model.

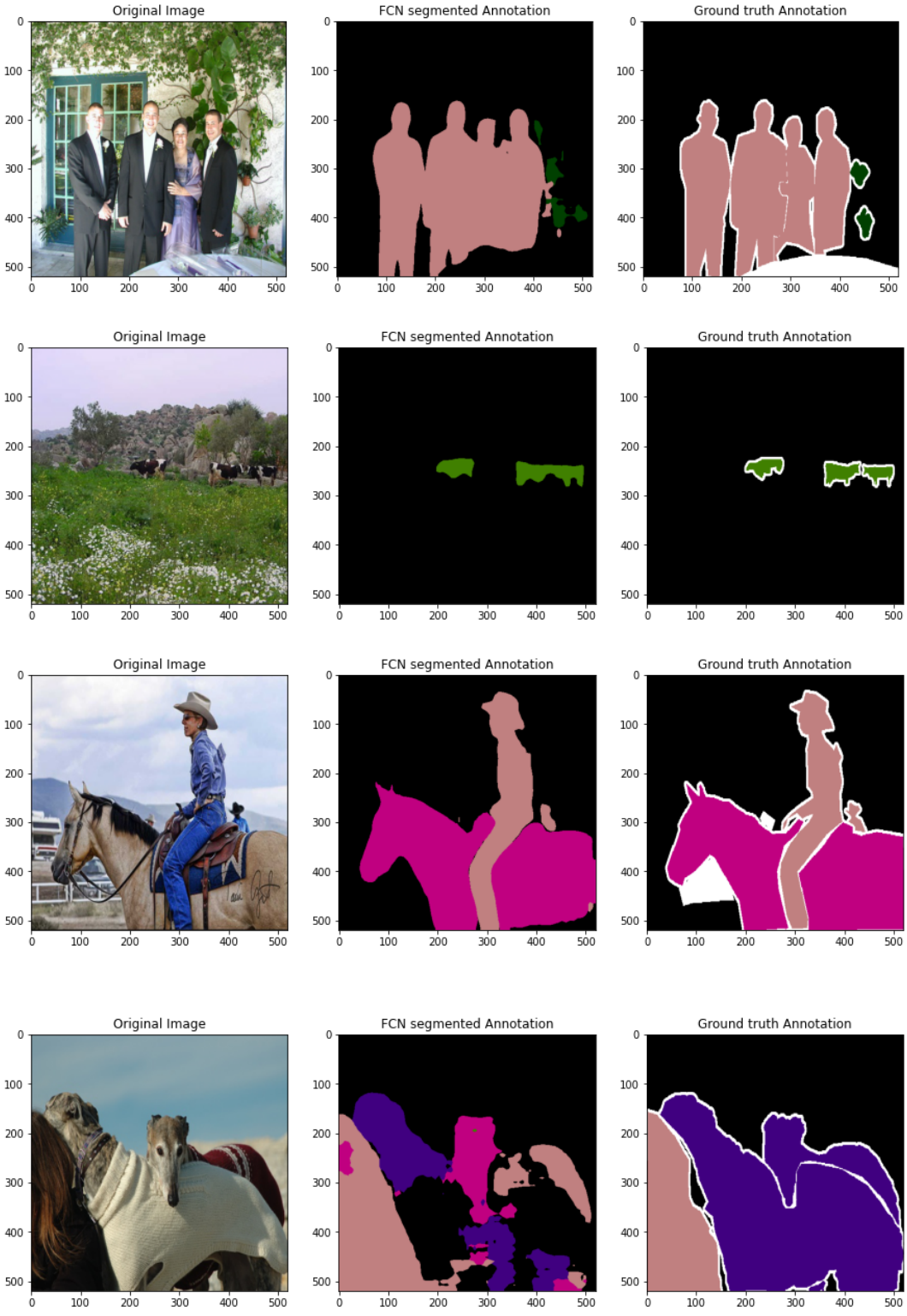
10)Both validation metrics below were decent.

Results:

Overall Pixel wise Accuracy : 87.93%

Overall Mean IOU : 92.10%

Some output Images:



B)Segmentation using MobileNetv2

1)Implemented a custom class dataset

(Source:https://towardsdatascience.com/how-to-use-datasets-and-dataloader-in-pytorch-for-custom-text-data-270eed7f7c00

for `SegmentationPascal(Dataset)`

2) Trainval is splitting using .txt files provided

3)Applied standard data transforms and loaded using data loader

4)Loaded mobilenetv2, removed last fully connected layer and added 5 sequential layers where Each sequential layer has

`nn.ConvTranspose2d,nn.BatchNorm2d,nn.ReLU`

5)Each layer upsamples the input by 2 and the weights are initialized by using bilinear_kernel function(source:https://d2l.ai/chapter_computer-vision/fcn.html)

6)Model No.of trainable and non-trainable parameters:

```
Total params: 15,486,591
Trainable params: 13,262,719
Non-trainable params: 2,223,872
```

7)The model was trained till 500 epochs

8)

Training Criteria Used:

```
criterion=nn.CrossEntropyLoss(ignore_index=255) #cross entropy loss
optimizer_ft=optim.SGD(model.parameters(),lr=0.001,momentum=0.9)  #SGD optimizer
exp_lr_scheduler=lr_scheduler.StepLR(optimizer_ft,step_size=7,gamma=1)  #exp_lr_scheduler with 0% depreciation(gamma=1)
```

In the above gamma was set to one to freeze the learning rate as it was depreciating for larger epochs(almost zero)

9)Visualization here was done similar to 2)A)

Results:

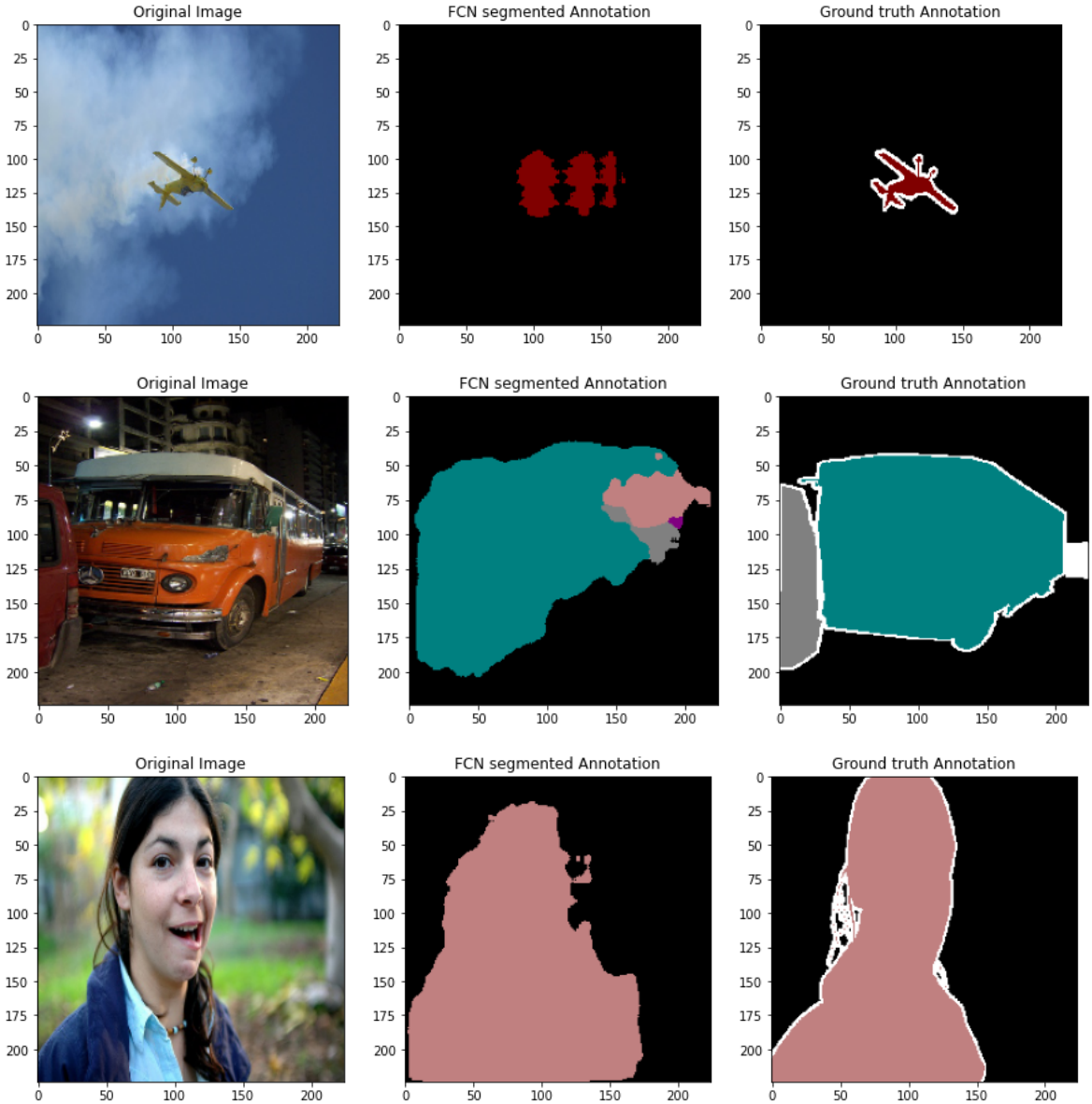
At Epoch 499:

```
train Loss: 0.2274631004052449
val Loss: 1.0781706271734943
```

Overall Average Accuracy is: 71.35626404579689

Overall MeanIOU : 81.14429013466703

Some Outputs visualized:

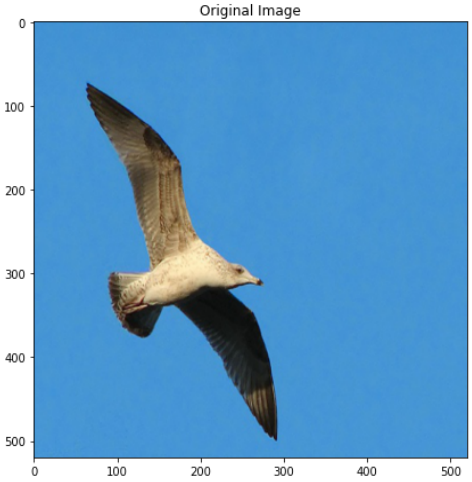


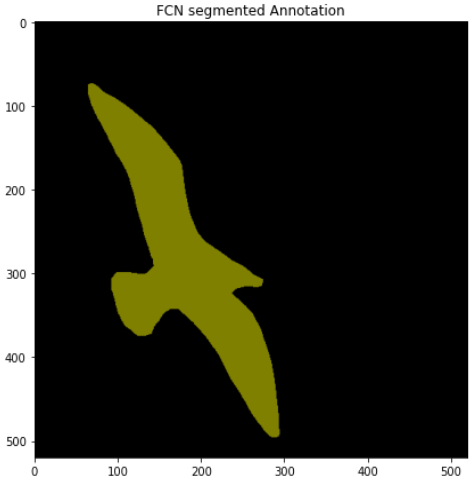
Comparison between two models:

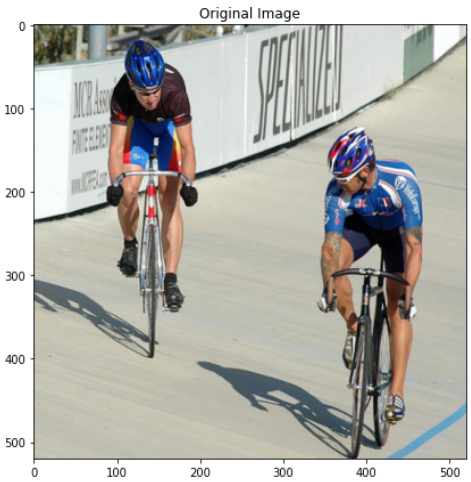
1)FCN_resnet50 was clearly better than mobilenetv2 interms of accuracy and meaniou.
This can be due to less training data for mobilenetv2 and skip connections was not implemented

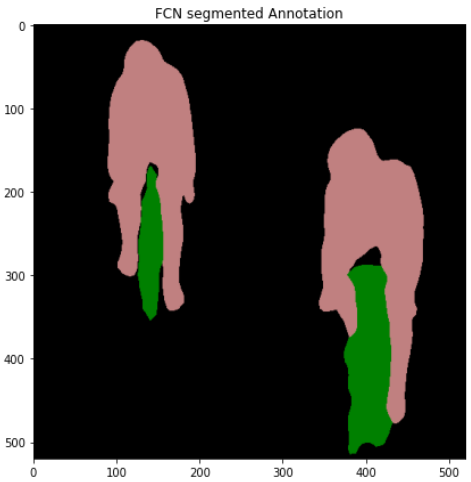
Comparsion between N-cut and FCN models for test4 and test5 Images:

1)FCN_resnet50 outputs for test4 and test5 images:

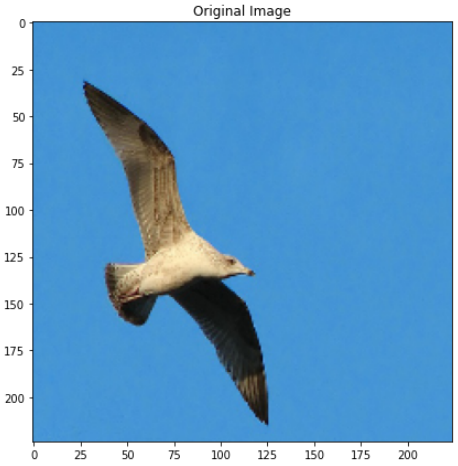


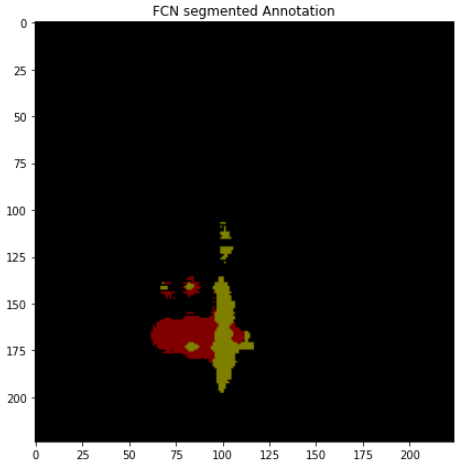


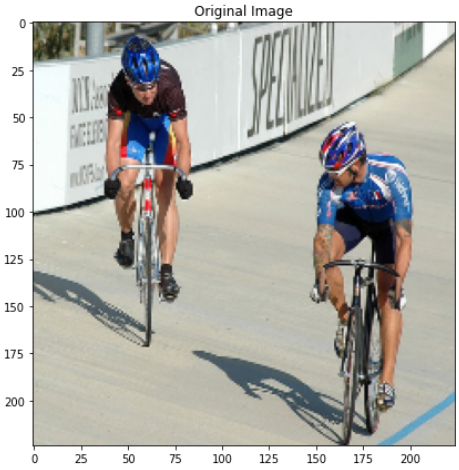


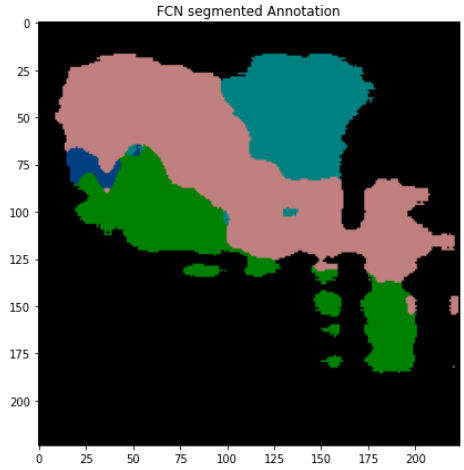


2)Mobilenetv2 outputs for test4 and test5 images:

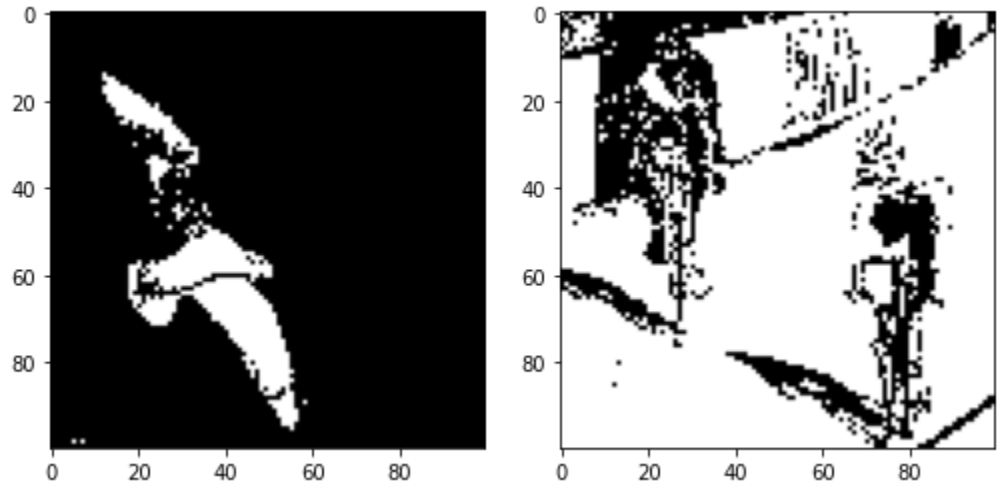








3)N cut outputs for test4 and test5



Observations:

1)FCN_resnet performed well among the three approaches(as expected as its test accuracy was high)

2)Cannot compare Ncuts much over accuracies with FCNResnet as Ncuts here divides into 2 classes while FCNresnet and mobilenetv2 divides into 21 classes.

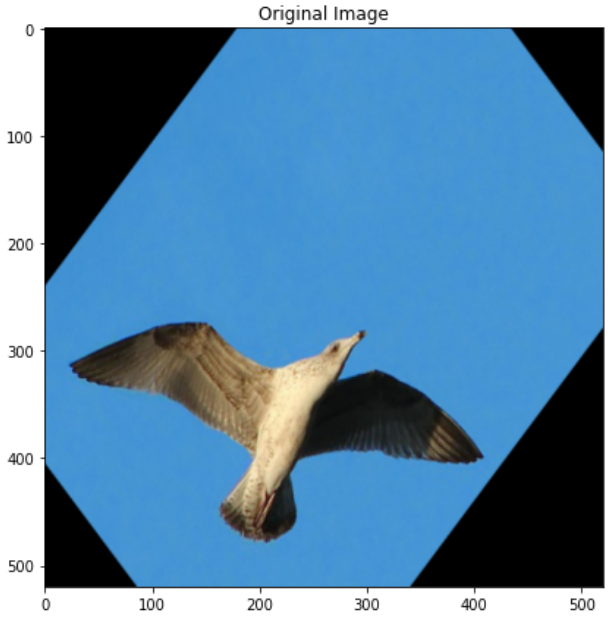
3)Ncuts took highest time among all the three for segmentation. Mobile_netv2 was trained till 500 epochs in 41mins while FCN_Resnet50 was used directly without training.

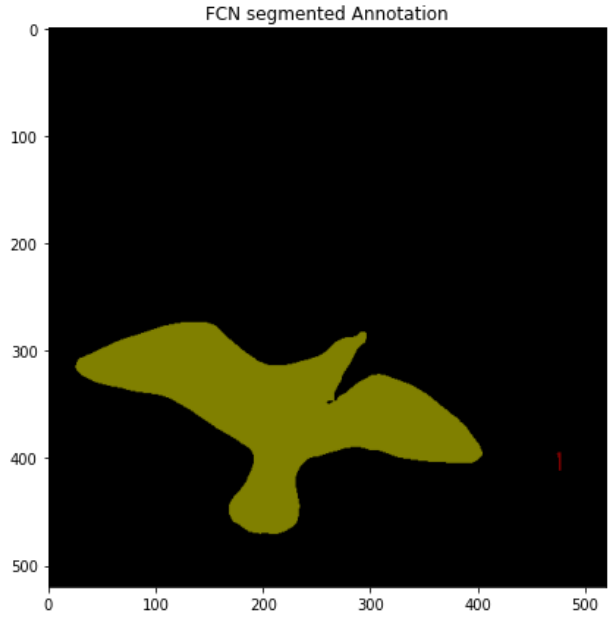
4)Similar performances was observed over gaussian noised and rotated images as well

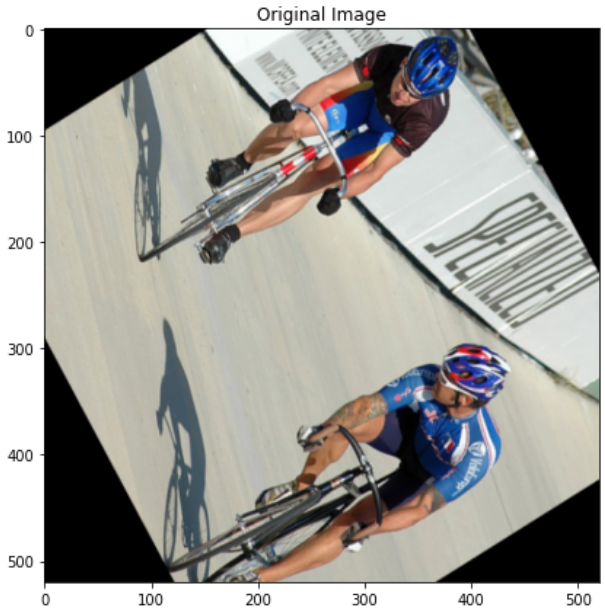
Comparison for Gaussian and rotated Images test4 and test5:

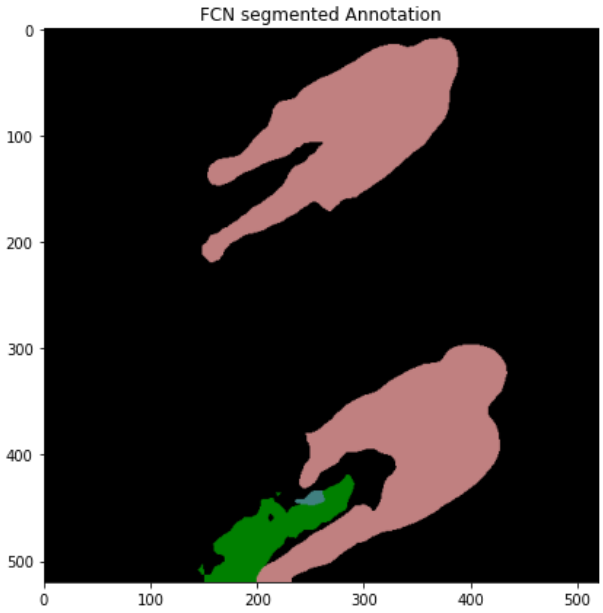
1)FCN_resnet50 outputs for test4 and test5 images:

Rotated Images:

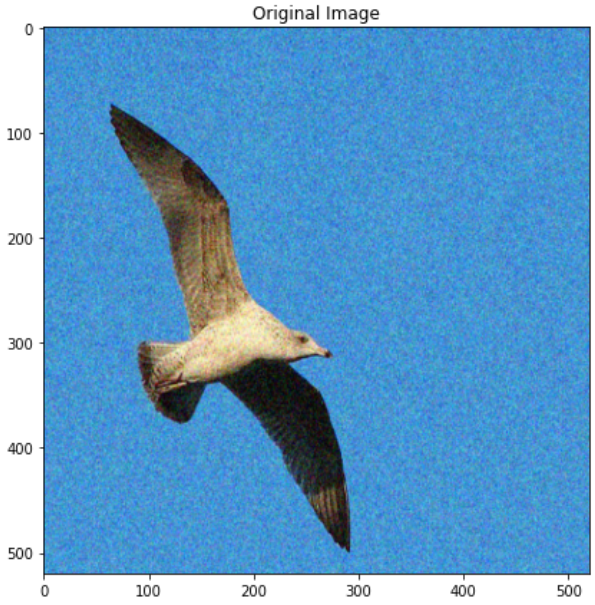


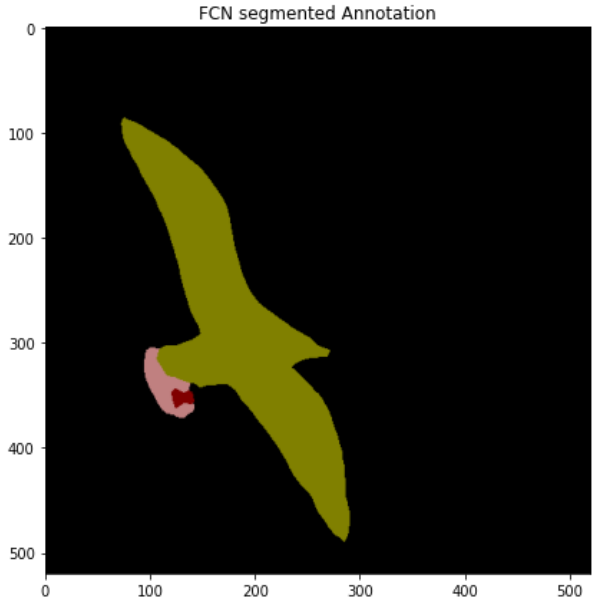


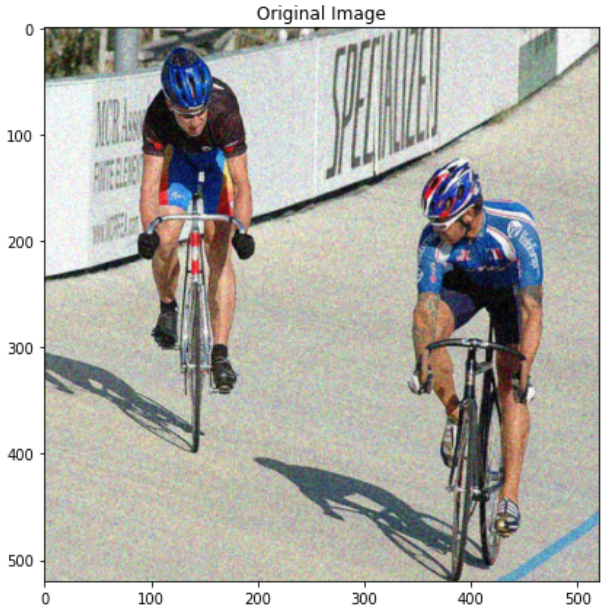


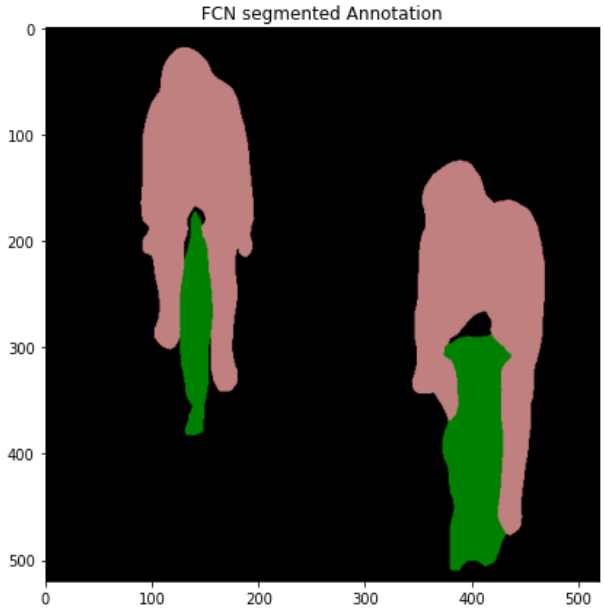


Gaussian Noised Images:



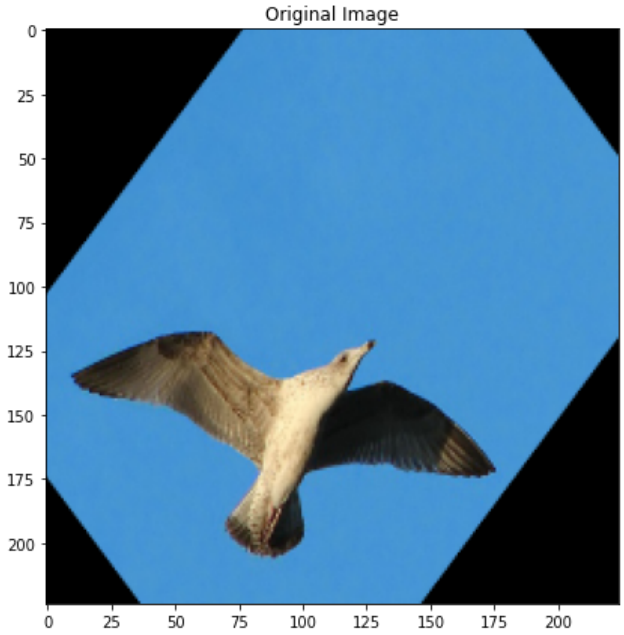


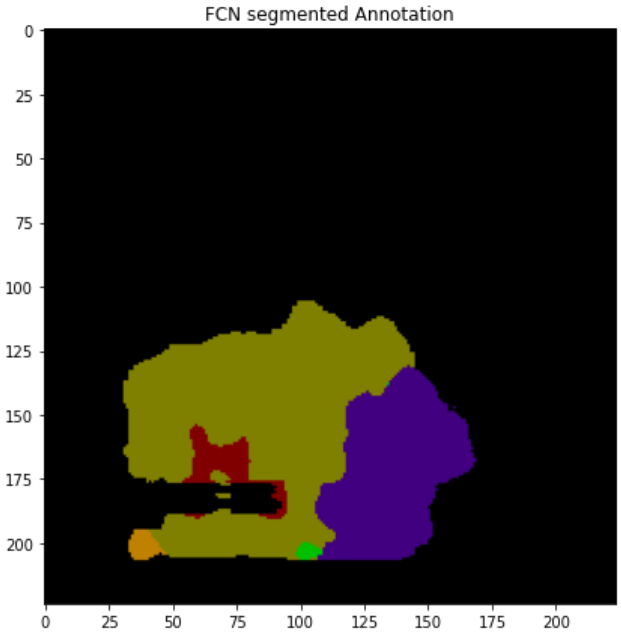


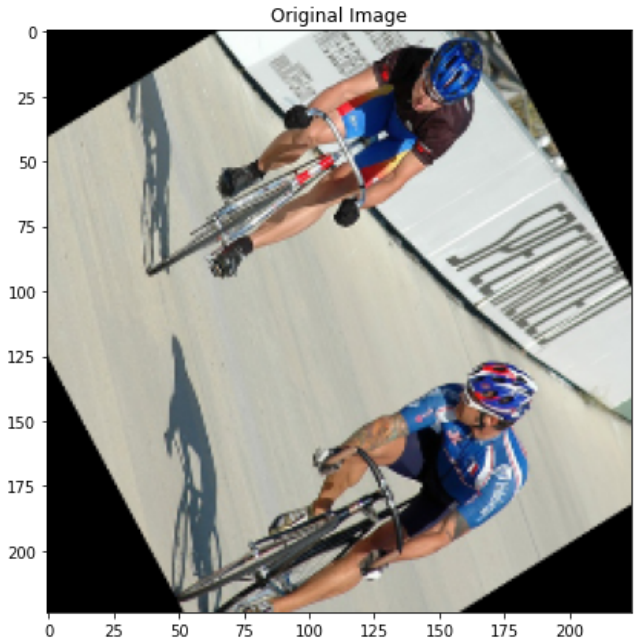


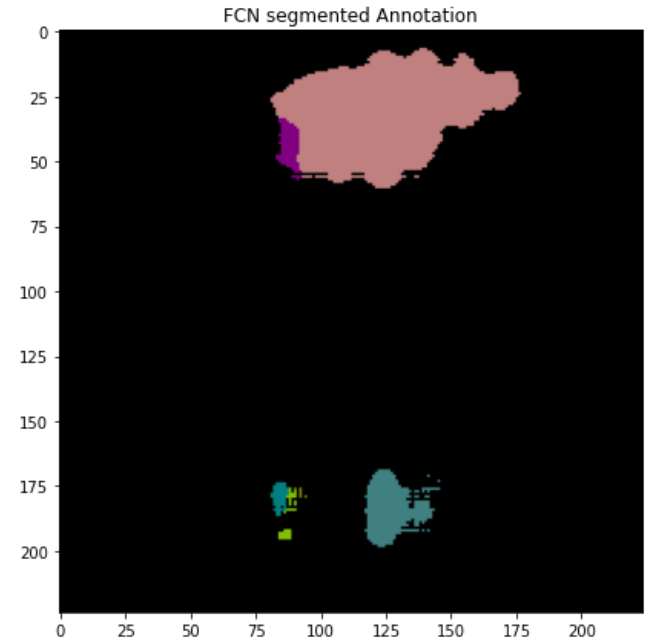
2)**Mobilenetv2 outputs for test4 and test5 images:**

Rotated Images:

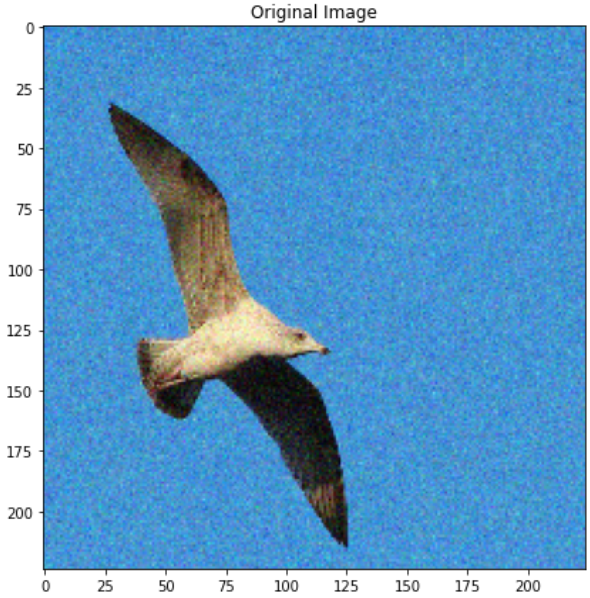


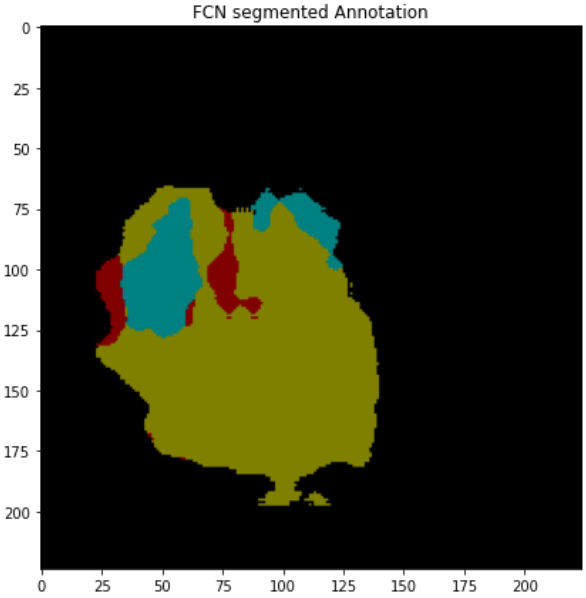


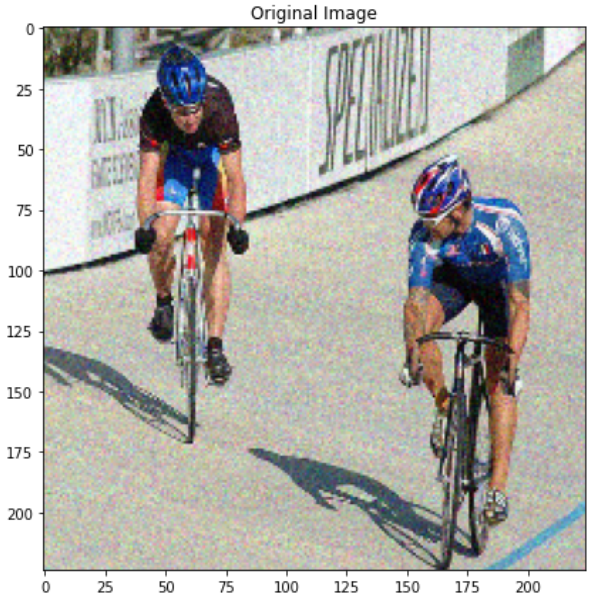


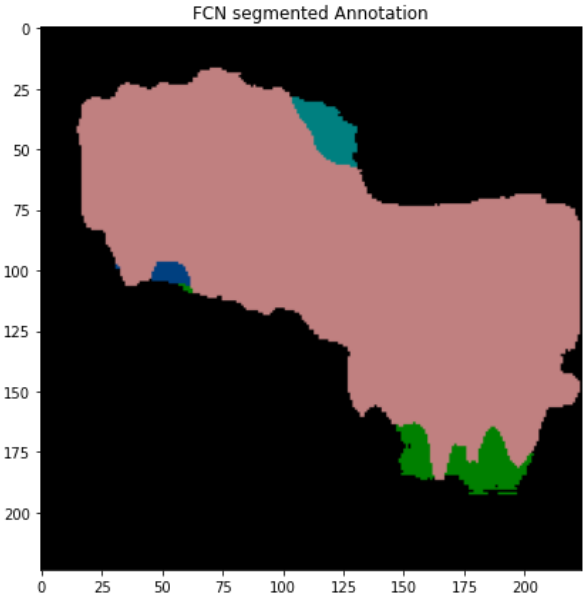


Gaussian Noised Images:



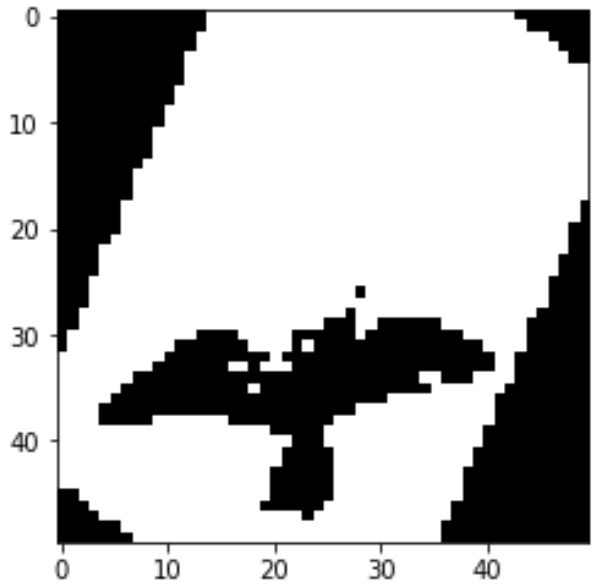


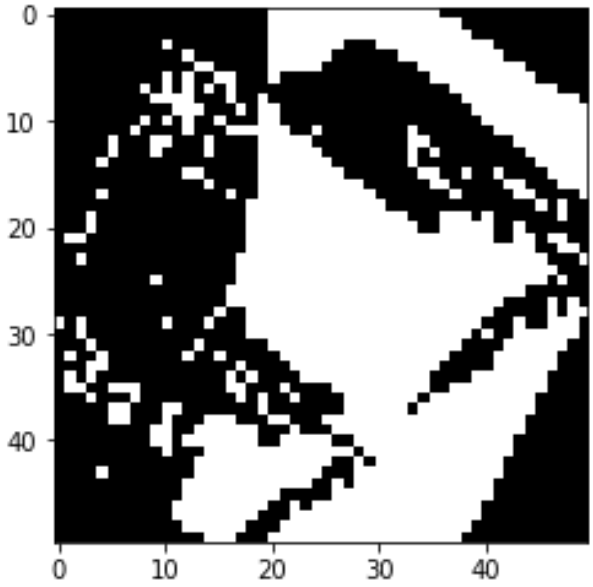




3)N cut outputs for test4 and test5:

Rotated Images:





Gaussian Noised Images: