data preparation process

- We use deep learning model (Zero-DCE++) for image Enhancement

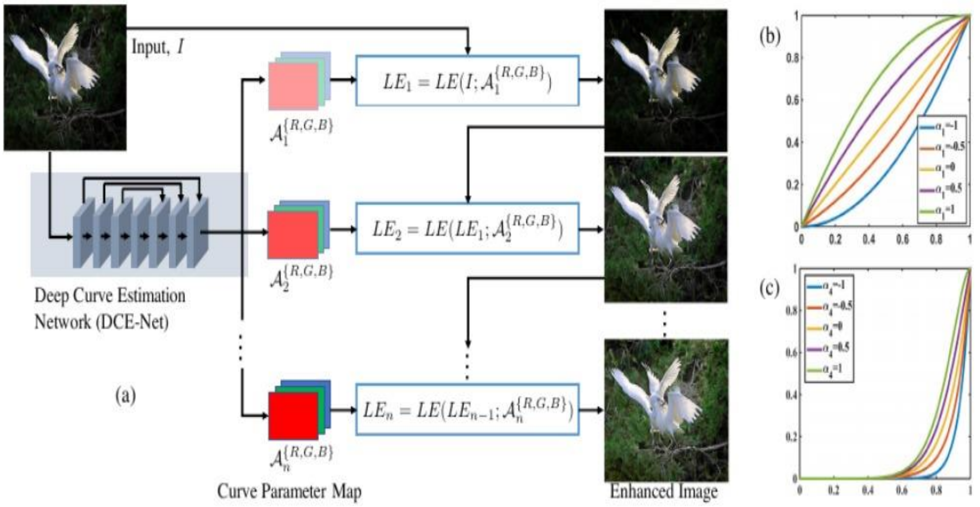
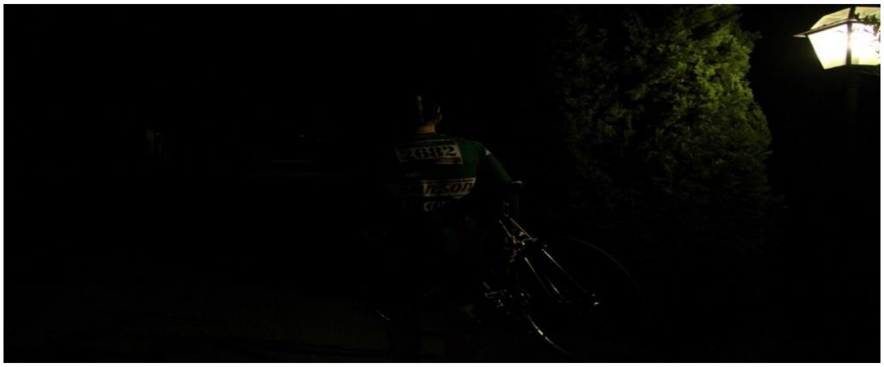
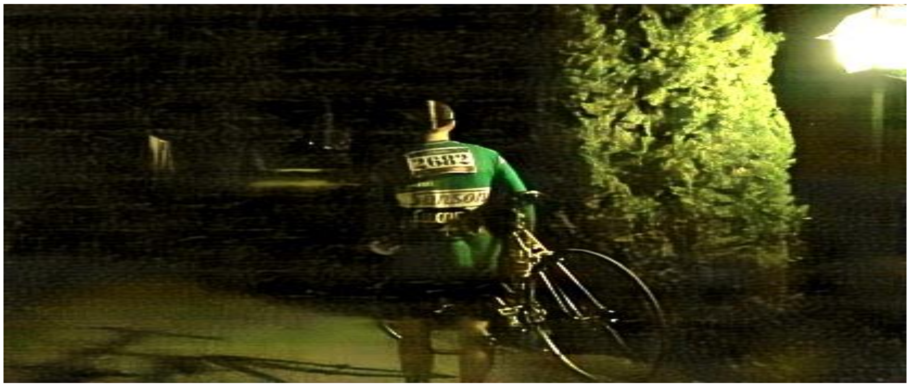


Image Enhancement

- Image before Enhancement



- Image after Enhancement



Description of the models

- ## Enhancement model (Zero-DCE++)

  The pipeline of our method. (a) The framework of Zero-DCE. A DCE-Net is devised to estimate a set of best-fitting Light-Enhancement curves (LE-curves: $LE(I(x);\alpha)=I(x)+\alpha I(x)(1-I(x))$) to iteratively enhance a given input image. (b, c) LE-curves with different adjustment parameters α and numbers of iteration n. In (c), $\alpha1$, $\alpha2$, and $\alpha3$ are equal to -1 while n is equal to 4. In each subfigure, the horizontal axis represents the input pixel values while the vertical axis represents the output pixel values.

- ## Classification model (ResNet50)

  ResNet-50 is a convolutional neural network that is 50 layers deep. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images.

- ## Detection model (YOLO 5)

  is a family of compound-scaled object detection models trained on the COCO dataset, and includes simple functionality for Test Time Augmentation (TTA), model ensembling, hyperparameter evolution, and export to ONNX, CoreML and TFLite.

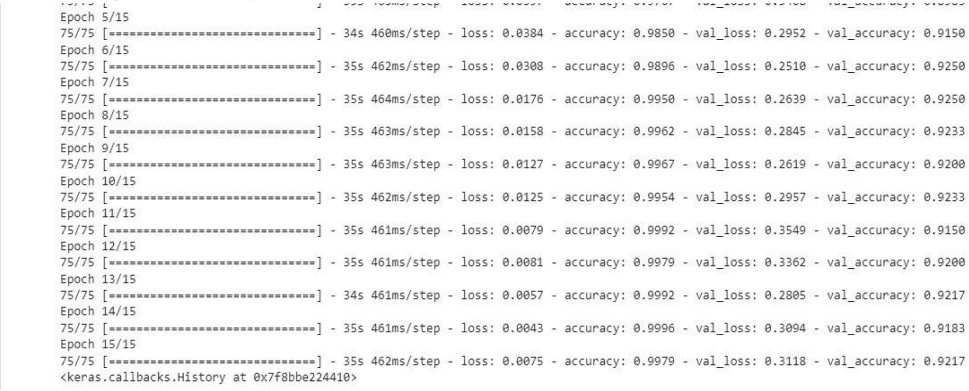
- ## Segmentation technique (K-means)

  K-means is a clustering algorithm. The goal is to partition *n* data points into *k* clusters. Each of the *n* data points will be assigned to a cluster with the nearest mean. The mean of each cluster is called its “centroid” or “center”.

Image Classification (Using ResNet50)

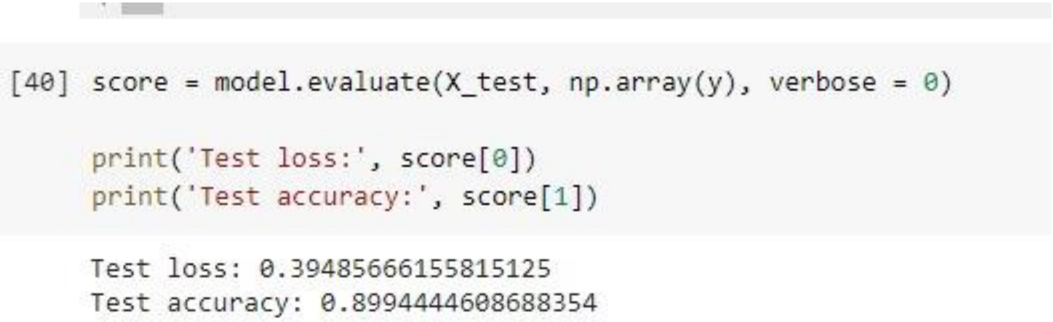
- Accuracy before enhancement

Training accuracy



```
Epoch 5/15
75/75 [==============================] - 34s 460ms/step - loss: 0.0384 - accuracy: 0.9850 - val_loss: 0.2952 - val_accuracy: 0.9150
Epoch 6/15
75/75 [==============================] - 35s 462ms/step - loss: 0.0308 - accuracy: 0.9896 - val_loss: 0.2510 - val_accuracy: 0.9250
Epoch 7/15
75/75 [==============================] - 35s 464ms/step - loss: 0.0176 - accuracy: 0.9950 - val_loss: 0.2639 - val_accuracy: 0.9250
Epoch 8/15
75/75 [==============================] - 35s 463ms/step - loss: 0.0158 - accuracy: 0.9962 - val_loss: 0.2845 - val_accuracy: 0.9233
Epoch 9/15
75/75 [==============================] - 35s 463ms/step - loss: 0.0127 - accuracy: 0.9967 - val_loss: 0.2619 - val_accuracy: 0.9200
Epoch 10/15
75/75 [==============================] - 35s 462ms/step - loss: 0.0125 - accuracy: 0.9954 - val_loss: 0.2957 - val_accuracy: 0.9233
Epoch 11/15
75/75 [==============================] - 35s 461ms/step - loss: 0.0079 - accuracy: 0.9992 - val_loss: 0.3549 - val_accuracy: 0.9150
Epoch 12/15
75/75 [==============================] - 35s 461ms/step - loss: 0.0081 - accuracy: 0.9979 - val_loss: 0.3362 - val_accuracy: 0.9200
Epoch 13/15
75/75 [==============================] - 34s 461ms/step - loss: 0.0057 - accuracy: 0.9992 - val_loss: 0.2805 - val_accuracy: 0.9217
Epoch 14/15
75/75 [==============================] - 35s 461ms/step - loss: 0.0043 - accuracy: 0.9996 - val_loss: 0.3094 - val_accuracy: 0.9183
Epoch 15/15
75/75 [==============================] - 35s 462ms/step - loss: 0.0075 - accuracy: 0.9979 - val_loss: 0.3118 - val_accuracy: 0.9217
<keras.callbacks.History at 0x7f8bbe224410>
```

Testing accuracy



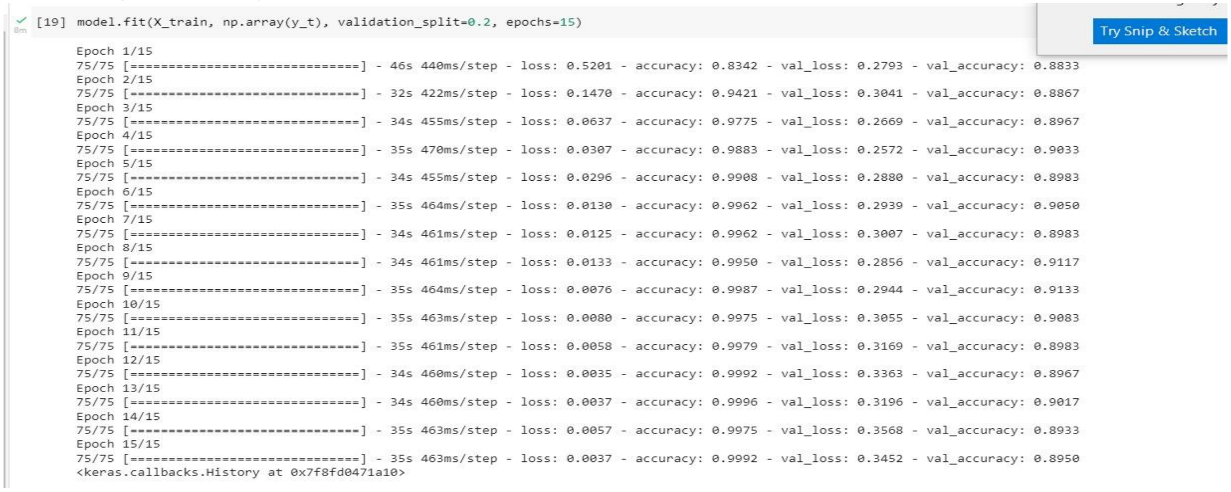
```
[40] score = model.evaluate(X_test, np.array(y), verbose = 0)

     print('Test loss:', score[0])
     print('Test accuracy:', score[1])

     Test loss: 0.39485666155815125
     Test accuracy: 0.8994444608688354
```

- Accuracy before enhancement

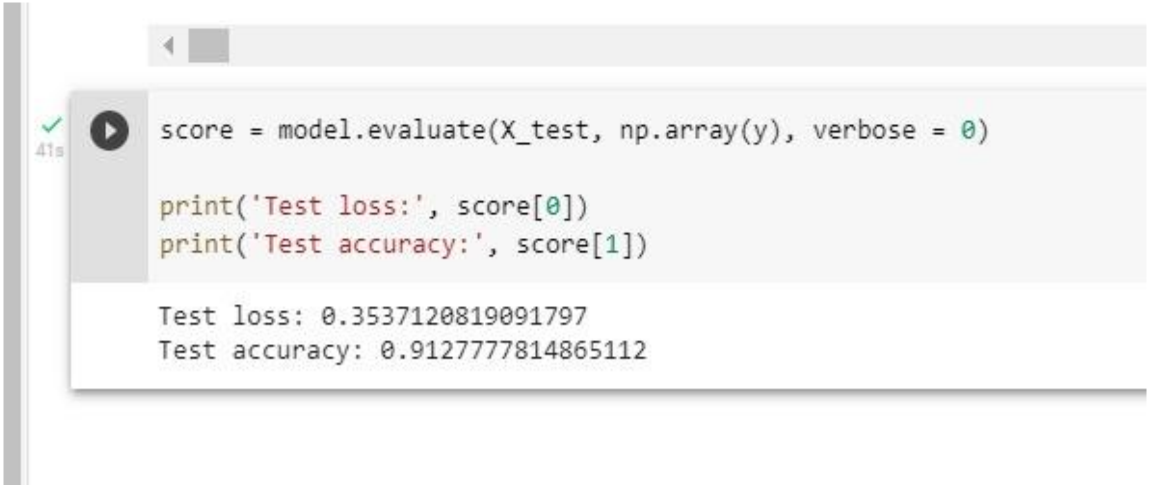
Training accuracy



```
[19] model.fit(X_train, np.array(y_t), validation_split=0.2, epochs=15)

     Epoch 1/15
     75/75 [==============================] - 46s 440ms/step - loss: 0.5201 - accuracy: 0.8342 - val_loss: 0.2793 - val_accuracy: 0.8833
     Epoch 2/15
     75/75 [==============================] - 32s 422ms/step - loss: 0.1470 - accuracy: 0.9421 - val_loss: 0.3041 - val_accuracy: 0.8867
     Epoch 3/15
     75/75 [==============================] - 34s 455ms/step - loss: 0.0637 - accuracy: 0.9775 - val_loss: 0.2669 - val_accuracy: 0.8967
     Epoch 4/15
     75/75 [==============================] - 35s 470ms/step - loss: 0.0307 - accuracy: 0.9883 - val_loss: 0.2572 - val_accuracy: 0.9033
     Epoch 5/15
     75/75 [==============================] - 34s 455ms/step - loss: 0.0296 - accuracy: 0.9908 - val_loss: 0.2880 - val_accuracy: 0.8983
     Epoch 6/15
     75/75 [==============================] - 35s 464ms/step - loss: 0.0130 - accuracy: 0.9962 - val_loss: 0.2939 - val_accuracy: 0.9050
     Epoch 7/15
     75/75 [==============================] - 34s 461ms/step - loss: 0.0125 - accuracy: 0.9962 - val_loss: 0.3007 - val_accuracy: 0.8983
     Epoch 8/15
     75/75 [==============================] - 34s 461ms/step - loss: 0.0133 - accuracy: 0.9950 - val_loss: 0.2856 - val_accuracy: 0.9117
     Epoch 9/15
     75/75 [==============================] - 35s 464ms/step - loss: 0.0076 - accuracy: 0.9987 - val_loss: 0.2944 - val_accuracy: 0.9133
     Epoch 10/15
     75/75 [==============================] - 35s 463ms/step - loss: 0.0080 - accuracy: 0.9975 - val_loss: 0.3055 - val_accuracy: 0.9083
     Epoch 11/15
     75/75 [==============================] - 35s 461ms/step - loss: 0.0058 - accuracy: 0.9979 - val_loss: 0.3169 - val_accuracy: 0.8983
     Epoch 12/15
     75/75 [==============================] - 34s 460ms/step - loss: 0.0035 - accuracy: 0.9992 - val_loss: 0.3363 - val_accuracy: 0.8967
     Epoch 13/15
     75/75 [==============================] - 34s 460ms/step - loss: 0.0037 - accuracy: 0.9996 - val_loss: 0.3196 - val_accuracy: 0.9017
     Epoch 14/15
     75/75 [==============================] - 35s 463ms/step - loss: 0.0057 - accuracy: 0.9975 - val_loss: 0.3568 - val_accuracy: 0.8933
     Epoch 15/15
     75/75 [==============================] - 35s 463ms/step - loss: 0.0037 - accuracy: 0.9992 - val_loss: 0.3452 - val_accuracy: 0.8950
     <keras.callbacks.History at 0x7f8fd0471a10>
```

Testing accuracy



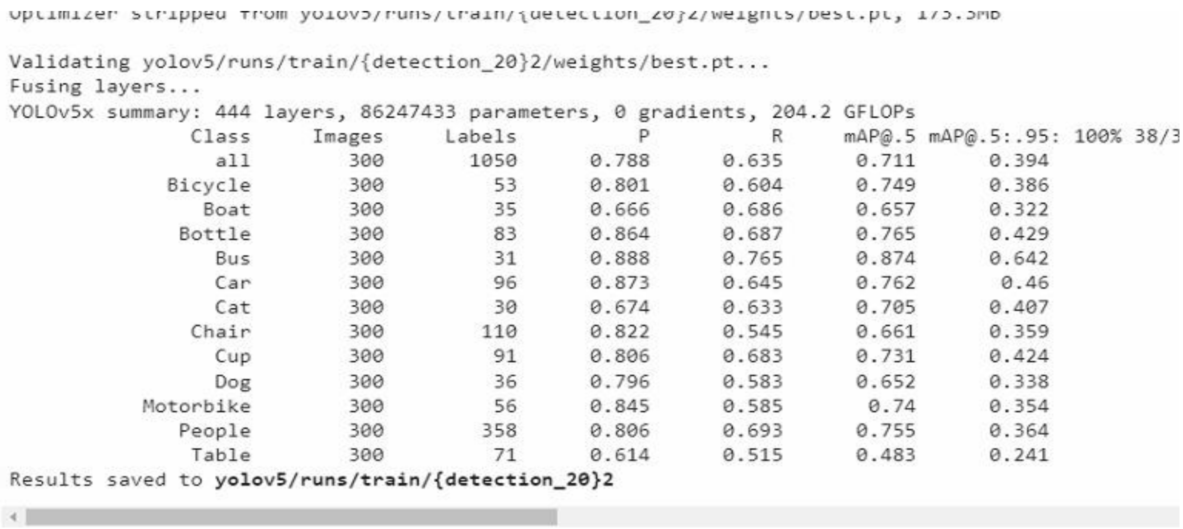
```
score = model.evaluate(X_test, np.array(y), verbose = 0)

print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Test loss: 0.3537120819091797
Test accuracy: 0.9127777814865112
```

Object detection

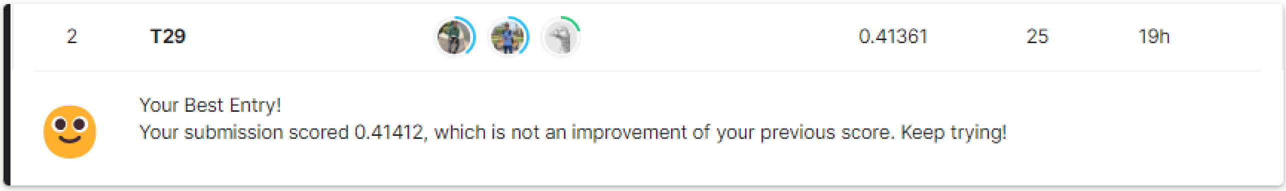
- Training accuracy



```
Optimizer stripped from yolov5/runs/train/{detection_20}2/weights/best.pt, 173.5MB

Validating yolov5/runs/train/{detection_20}2/weights/best.pt...
Fusing layers...
YOLOv5x summary: 444 layers, 86247433 parameters, 0 gradients, 204.2 GFLOPs
              Class     Images     Labels          P          R      mAP@.5 mAP@.5:.95: 100% 38/3
                all        300       1050      0.788      0.635      0.711      0.394
            Bicycle        300         53      0.801      0.604      0.749      0.386
               Boat        300         35      0.666      0.686      0.657      0.322
             Bottle        300         83      0.864      0.687      0.765      0.429
                Bus        300         31      0.888      0.765      0.874      0.642
                Car        300         96      0.873      0.645      0.762       0.46
                Cat        300         30      0.674      0.633      0.705      0.407
              Chair        300        110      0.822      0.545      0.661      0.359
                Cup        300         91      0.806      0.683      0.731      0.424
                Dog        300         36      0.796      0.583      0.652      0.338
          Motorbike        300         56      0.845      0.585       0.74      0.354
             People        300        358      0.806      0.693      0.755      0.364
              Table        300         71      0.614      0.515      0.483      0.241
Results saved to yolov5/runs/train/{detection_20}2
```

Kaggle competition MCRMSE score

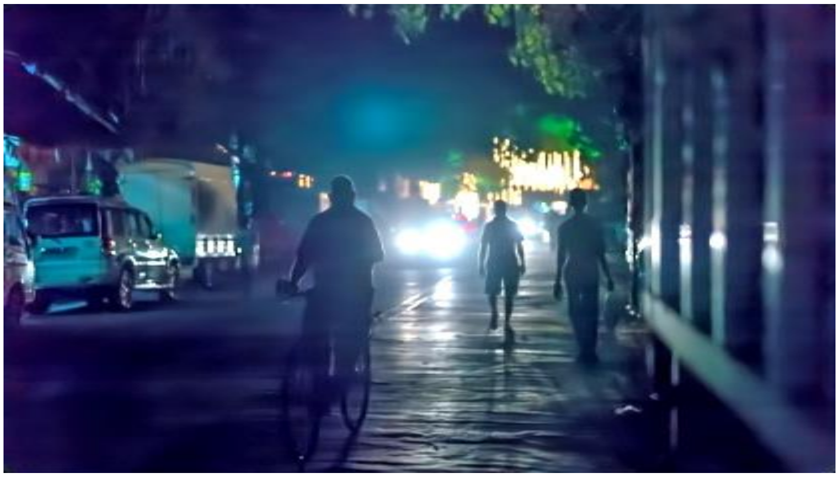


2 **T29** 0.41361 25 19h

Your Best Entry!
Your submission scored 0.41412, which is not an improvement of your previous score. Keep trying!

Object detection

- Image before detection



- Image after detection

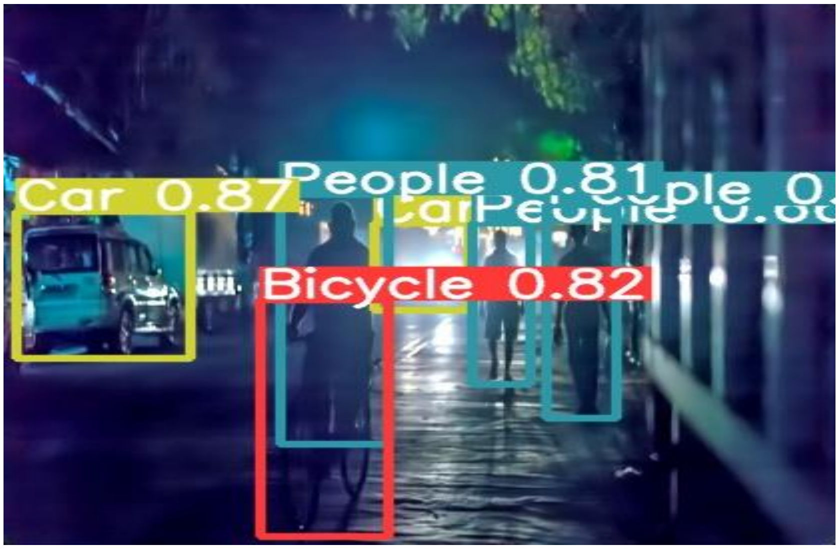
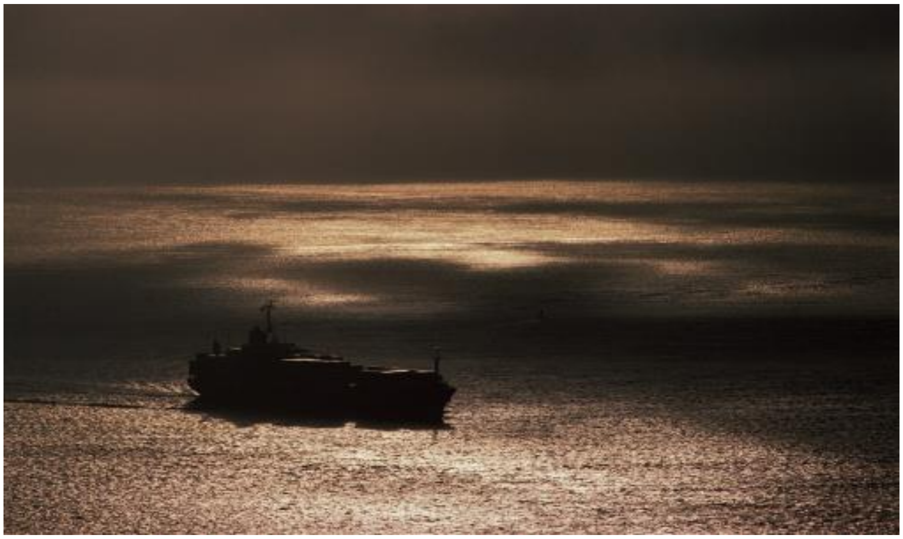


Image Segmentation

- Image before Segmentation



- Image after Segmentation