Assignment Report

Task 1:

optimizer = Adam,

learning rate = 0.001

Performance metric: **Mean Squared Error, Mean Absolute Error, Peak Signal-to-Noise Ratio (PSNR)**

Loss Function: binary_crossentropy,

Note: Since my computer doesn't have GPU power and google colab limit also breached so I am using small number of batch size with less amount of dataset.

Experiment 1:

Epochs: 10, Bottleneck dimension: 256, Split ratio: 80-10-10, train-val-test split, Number of images: 2000

Dimension: 256 * 256 *3

When we took batch size of 32 and ran the autoencoder for 10 epochs with 2000 images then we get following results where loss didn't reduce much after 5th iteration. But improved significantly in 10th iteration for training data.

This model was implemented by normalizing image to grayscale and then processing on different dense layers.

```
Epoch 1/10
50/50 [==============================] - 49s 973ms/step - loss: 0.0700 - val_loss: 0.0646
Epoch 2/10
50/50 [==============================] - 49s 972ms/step - loss: 0.0616 - val_loss: 0.0622
Epoch 3/10
50/50 [==============================] - 50s 1s/step - loss: 0.0599 - val_loss: 0.0621
Epoch 4/10
50/50 [==============================] - 53s 1s/step - loss: 0.0594 - val_loss: 0.0618
Epoch 5/10
50/50 [==============================] - 48s 963ms/step - loss: 0.0591 - val_loss: 0.0645
Epoch 6/10
50/50 [==============================] - 50s 994ms/step - loss: 0.0589 - val_loss: 0.0614
Epoch 7/10
50/50 [==============================] - 49s 990ms/step - loss: 0.0579 - val_loss: 0.0615
Epoch 8/10
50/50 [==============================] - 51s 1s/step - loss: 0.0599 - val_loss: 0.0634
Epoch 9/10
50/50 [==============================] - 50s 989ms/step - loss: 0.0584 - val_loss: 0.0616
Epoch 10/10
50/50 [==============================] - 49s 972ms/step - loss: 0.0573 - val_loss: 0.0639
7/7 [==============================] - 1s 90ms/step
MSE: 0.06233366948809229, MAE: 0.200835470608585
```

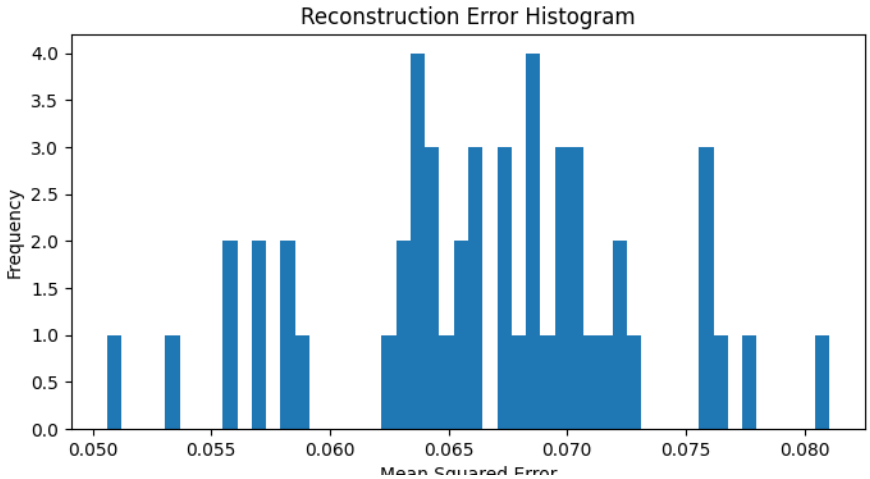
Experiment 2:

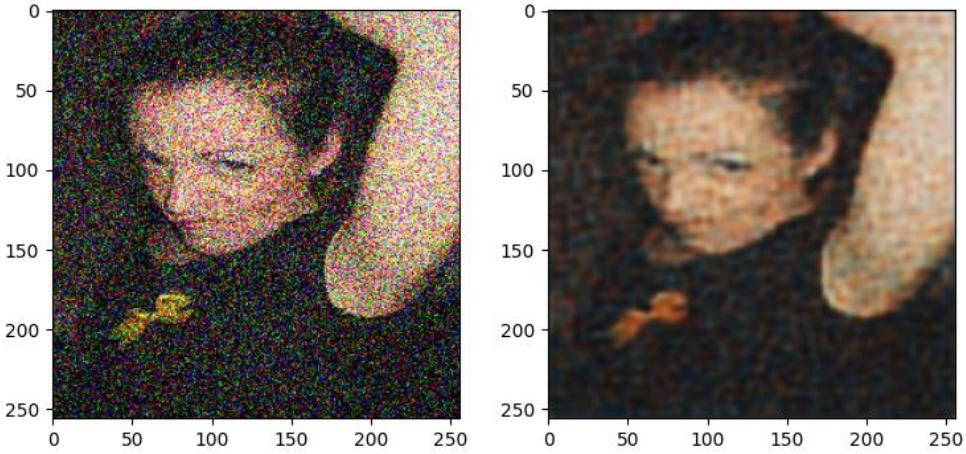
In this experiment I have used following parameters:

Batch size: 16, Epochs: 10, Bottleneck dimension: 256, Split ratio: 80-10-10, train-val-test split

Number of images: 500, Noise factor: 0.1, Dimension: 256 * 256 *3

When I added noise to image then training of autoencoders got slowed down significantly. That's why I had to use less number of images.





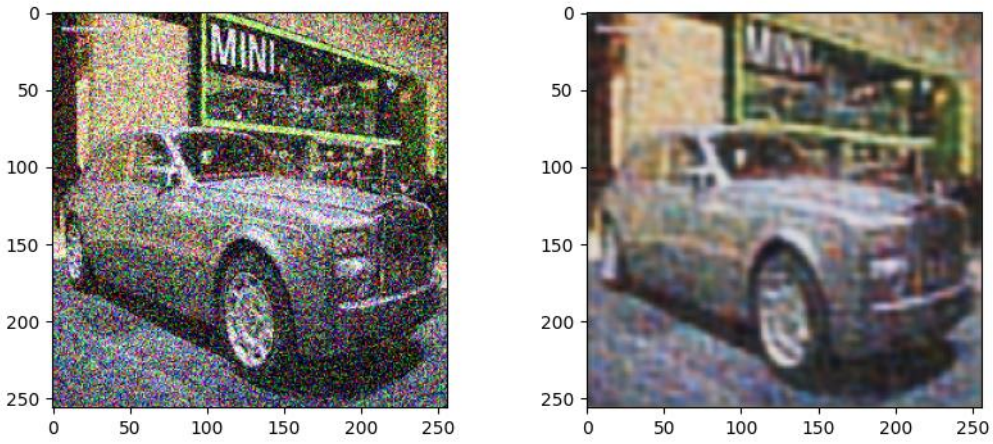
```
(400, 256, 256, 3)
(50, 256, 256, 3)
(51, 256, 256, 3)
Epoch 1/10
25/25 [==============================] - 115s 5s/step - loss: 0.6440 - val_loss: 0.6026
Epoch 2/10
25/25 [==============================] - 115s 5s/step - loss: 0.6062 - val_loss: 0.5936
Epoch 3/10
25/25 [==============================] - 113s 5s/step - loss: 0.5961 - val_loss: 0.5862
Epoch 4/10
25/25 [==============================] - 117s 5s/step - loss: 0.5927 - val_loss: 0.5858
Epoch 5/10
25/25 [==============================] - 123s 5s/step - loss: 0.5913 - val_loss: 0.5833
Epoch 6/10
25/25 [==============================] - 118s 5s/step - loss: 0.5904 - val_loss: 0.5821
Epoch 7/10
25/25 [==============================] - 122s 5s/step - loss: 0.5891 - val_loss: 0.5797
Epoch 8/10
25/25 [==============================] - 121s 5s/step - loss: 0.5877 - val_loss: 0.5788
Epoch 9/10
25/25 [==============================] - 121s 5s/step - loss: 0.5866 - val_loss: 0.5778
Epoch 10/10
25/25 [==============================] - 122s 5s/step - loss: 0.5859 - val_loss: 0.5775
2/2 [==============================] - 3s 1s/step
2/2 [==============================] - 3s 1s/step
MSE: 0.06650811388982054, MAE: 0.21188063205964933
```

Experiment 3:

Batch size: 16, Epochs: 10, Bottleneck dimension: 256, Split ratio: 70-10-20, train-val-test split

Number of images: 500, Noise factor: 0.1, Dimension: 256 * 256 *3

It is being trained on a less amount of data on a noised image and training loss and validation loss is significantly high, however as we increase number of experiments with more images then loss start to reduce significantly.

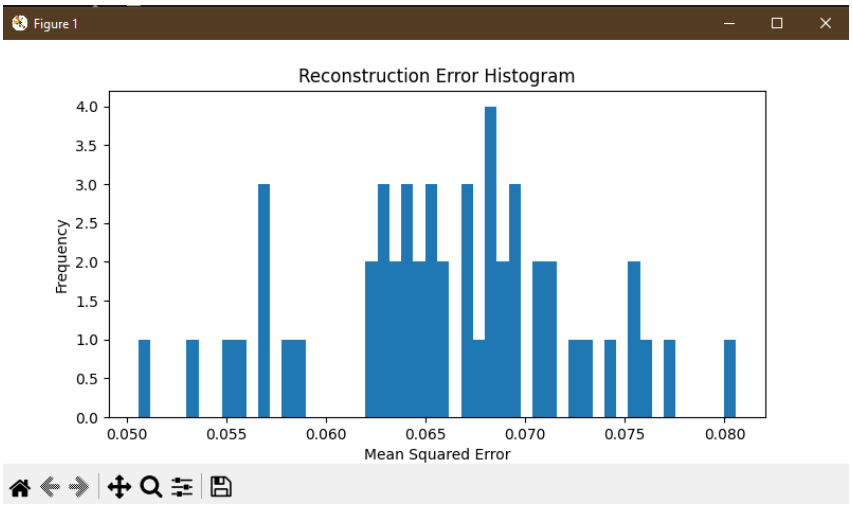


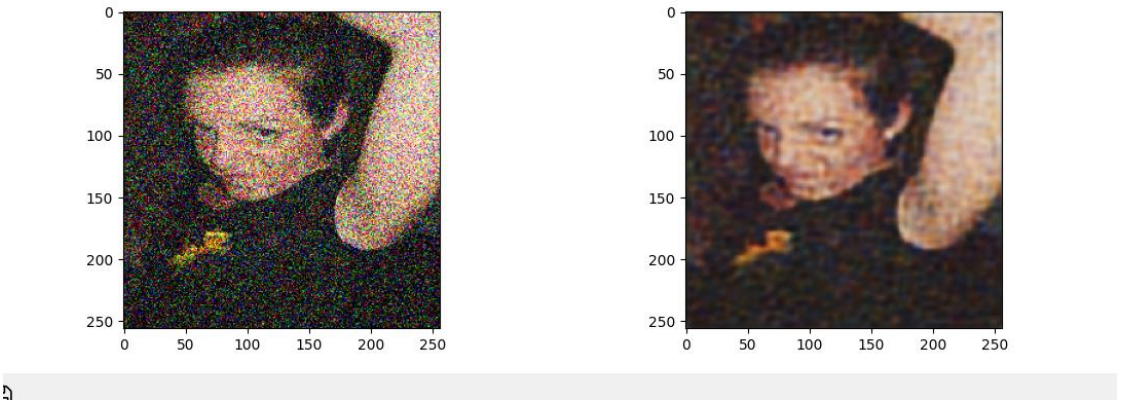

```
(350, 256, 256, 3)
(49, 256, 256, 3)
(102, 256, 256, 3)
Epoch 1/10
22/22 [==============================] - 93s 4s/step - loss: 0.6393 - val_loss: 0.6077
Epoch 2/10
22/22 [==============================] - 96s 4s/step - loss: 0.6054 - val_loss: 0.5931
Epoch 3/10
22/22 [==============================] - 103s 5s/step - loss: 0.5970 - val_loss: 0.5872
Epoch 4/10
22/22 [==============================] - 105s 5s/step - loss: 0.5933 - val_loss: 0.5856
Epoch 5/10
22/22 [==============================] - 107s 5s/step - loss: 0.5915 - val_loss: 0.5833
Epoch 6/10
22/22 [==============================] - 103s 5s/step - loss: 0.5913 - val_loss: 0.5828
Epoch 7/10
22/22 [==============================] - 104s 5s/step - loss: 0.5894 - val_loss: 0.5812
Epoch 8/10
22/22 [==============================] - 104s 5s/step - loss: 0.5878 - val_loss: 0.5797
Epoch 9/10
22/22 [==============================] - 114s 5s/step - loss: 0.5864 - val_loss: 0.5783
Epoch 10/10
22/22 [==============================] - 99s 5s/step - loss: 0.5857 - val_loss: 0.5780
4/4 [==============================] - 9s 2s/step
4/4 [==============================] - 8s 2s/step
MSE: 0.06578830089469222, MAE: 0.21156363455517346
```

Experiment 4:

Batch size: 16, Epochs: 5, Bottleneck dimension: 256, Split ratio: 70-10-20, train-val-test split

Number of images: 500, Noise factor: 0.1, Dimension: 256 * 256 *3






```
Epoch 1/5
50/50 [==============================] - 108s 2s/step - loss: 0.6273 - val_loss: 0.5967
Epoch 2/5
50/50 [==============================] - 104s 2s/step - loss: 0.5968 - val_loss: 0.5844
Epoch 3/5
50/50 [==============================] - 115s 2s/step - loss: 0.5911 - val_loss: 0.5827
Epoch 4/5
50/50 [==============================] - 122s 2s/step - loss: 0.5884 - val_loss: 0.5785
Epoch 5/5
50/50 [==============================] - 121s 2s/step - loss: 0.5863 - val_loss: 0.5763
```

Experiment 5:

I am taking 70-10-20 split as I get better performance on this split, and I can tune hyper parameters so that it performs better on test dataset.

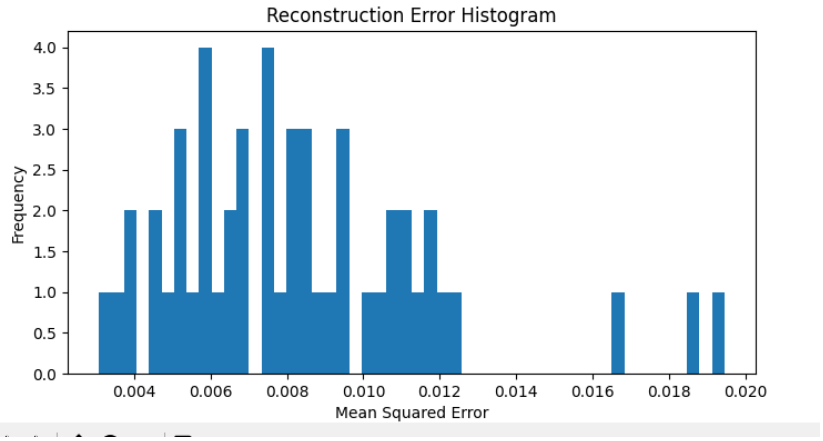
I have also new metric for check the performance Peak Signal-to-Noise Ratio (PSNR) where higher number of PSNR value signifies that this model is performing better.

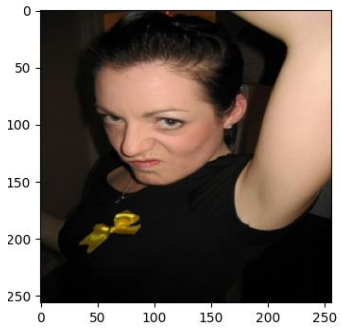
Batch size: 16, Epochs: 5, Bottleneck dimension: 256, Split ratio: 70-10-20, train-val-test split

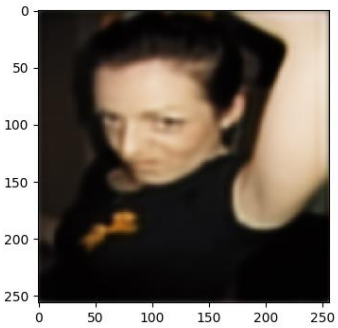
Number of images: 500, Noise factor: 0, Dimension: 256 * 256 *3

Peak Signal-to-Noise Ratio (PSNR): 19.7631

MSE: 0.008349359110217372, MAE: 0.06524227693403403







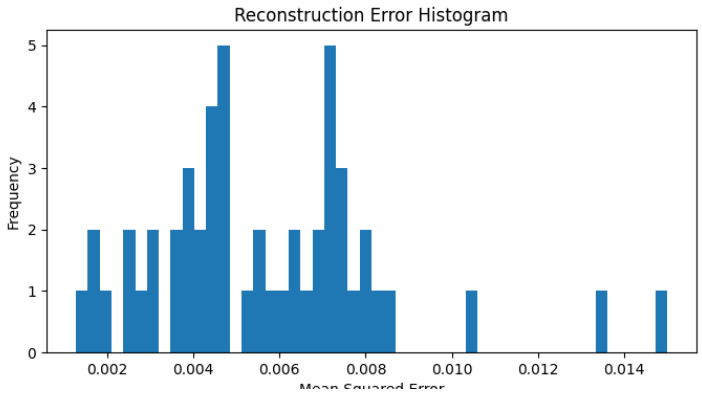

```
(400, 256, 256, 3)
(50, 256, 256, 3)
(51, 256, 256, 3)
Epoch 1/5
25/25 [==============================] - 119s 5s/step - loss: 0.6118 - val_loss: 0.5323
Epoch 2/5
25/25 [==============================] - 116s 5s/step - loss: 0.5359 - val_loss: 0.5175
Epoch 3/5
25/25 [==============================] - 122s 5s/step - loss: 0.5297 - val_loss: 0.5138
Epoch 4/5
25/25 [==============================] - 114s 5s/step - loss: 0.5262 - val_loss: 0.5176
Epoch 5/5
25/25 [==============================] - 118s 5s/step - loss: 0.5239 - val_loss: 0.5122
2/2 [==============================] - 4s 1s/step
2/2 [==============================] - 3s 1s/step
```

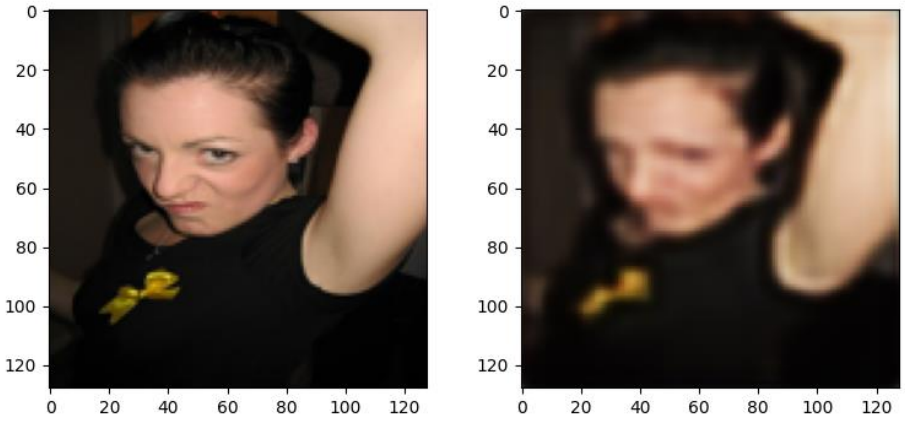
Experiment 6:

Batch size: 16, Epochs: 10, Bottleneck dimension: 256, Split ratio: 80-10-10, train-val-test split

Number of images: 500, Noise factor: 0, Dimension: 128 * 128 *3

As I reduced the dimension then reconstruction did not happen properly.





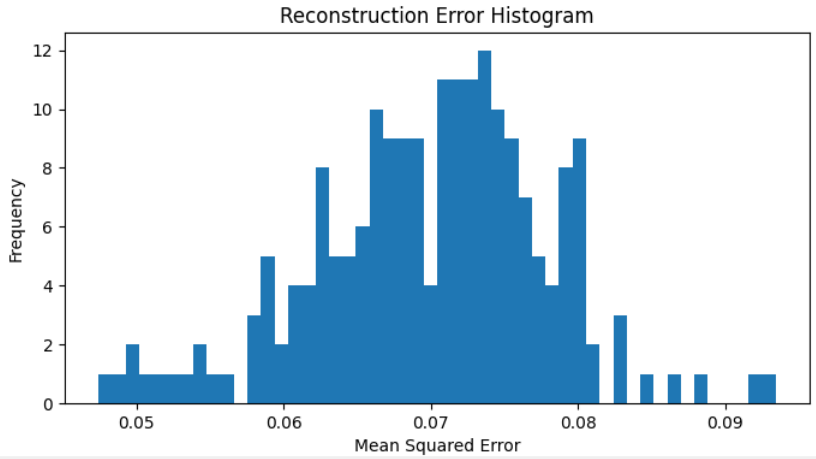
Experiment 7:

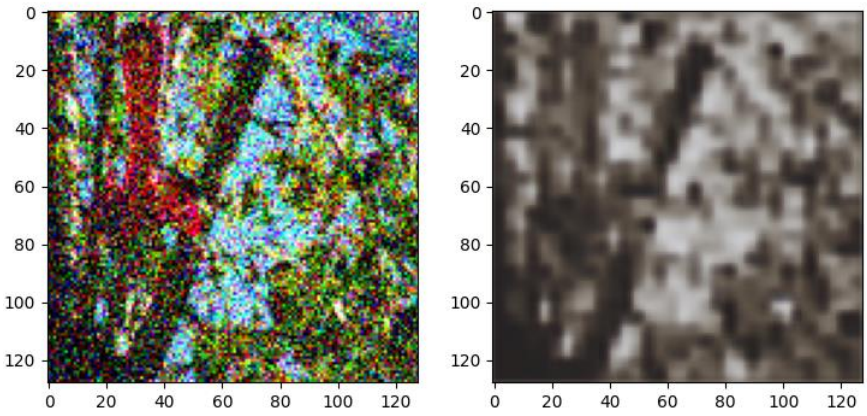
Batch size: 32, Epochs: 20, Bottleneck dimension: 256, Split ratio: 70-10-20, train-val-test split

Number of images: 1000, Noise factor: 0.1, Dimension: 128 * 128 *3,

loss function: mean squared error

I have changed loss function for autoencoder which significantly reduced loss generated.

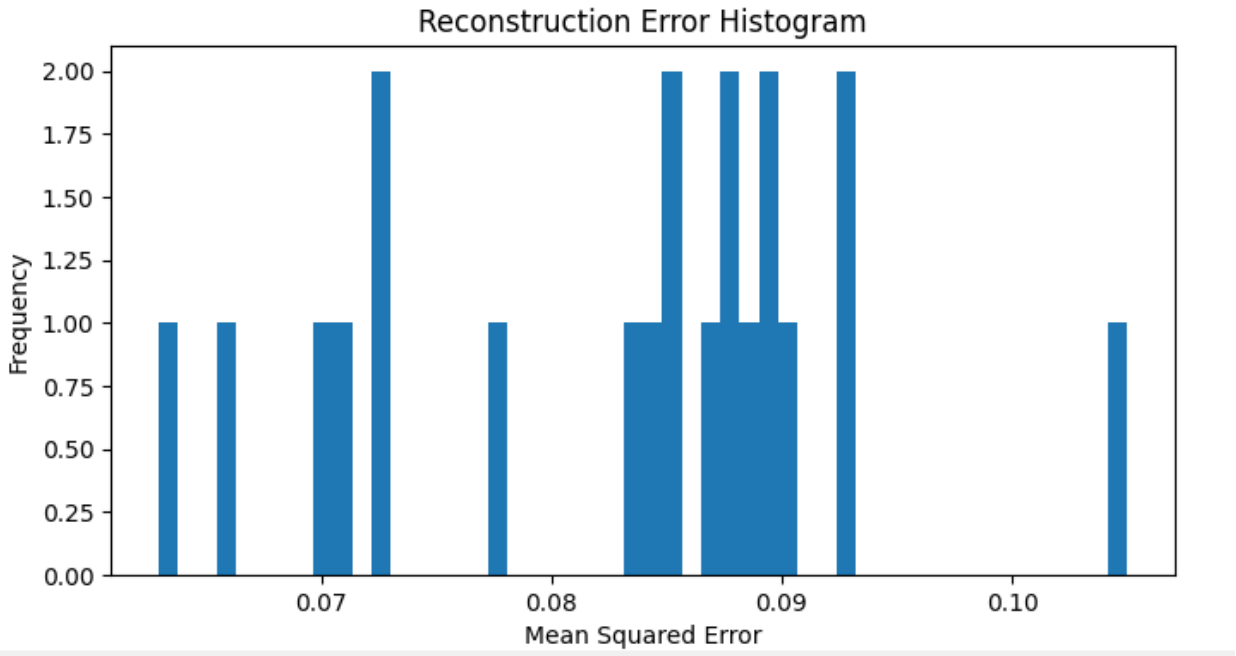




Experiment 7:

Batch size: 32, Epochs: 10, Bottleneck dimension: 128, Split ratio: 70-10-20, train-val-test split

Number of images: 100, Noise factor: 0.1, Dimension: 128 * 128 *3,

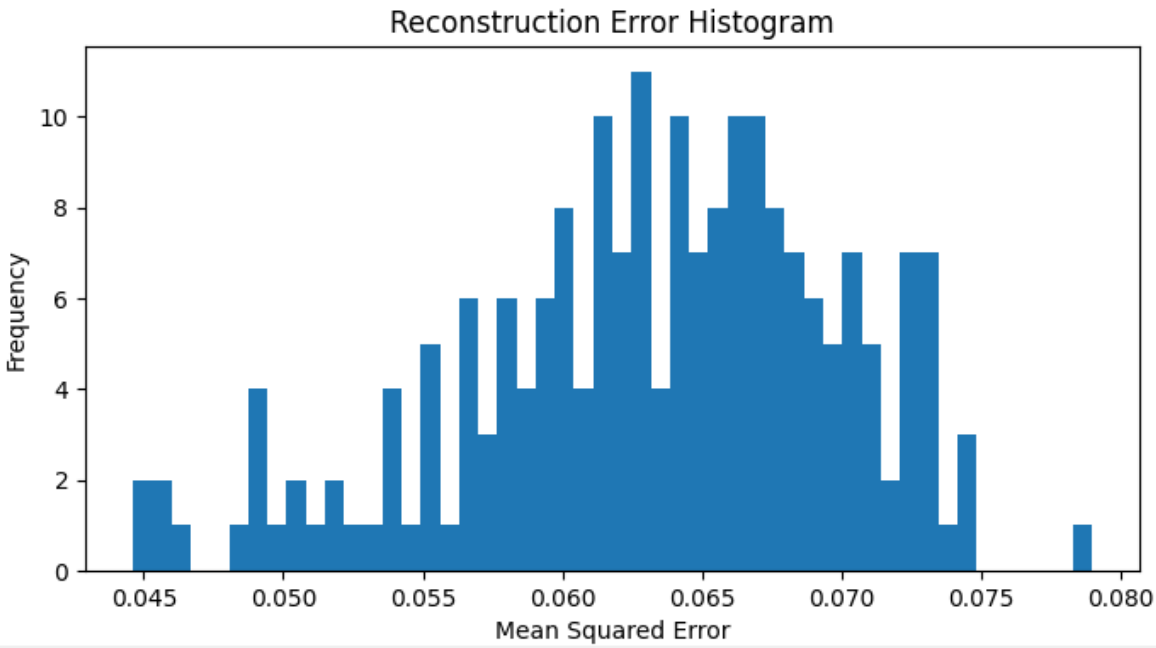


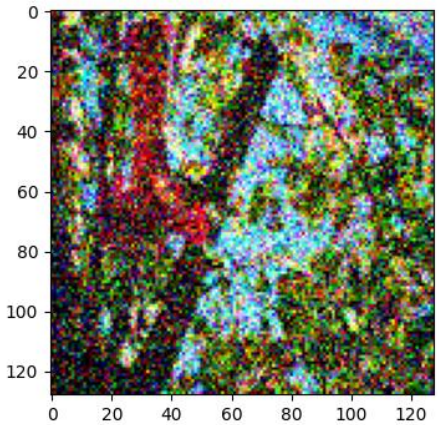
Final Experiment:

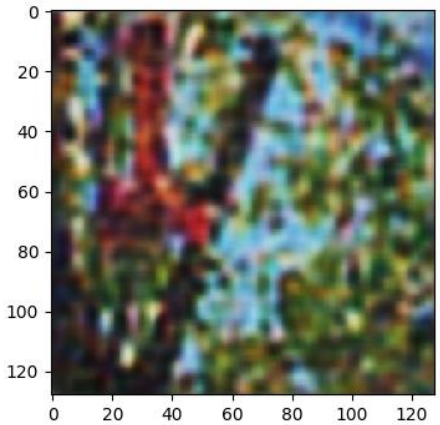
Batch size: 32, Epochs: 20, Bottleneck dimension: 128, Split ratio: 70-10-20, train-val-test split

Number of images: 1000, Noise factor: 0.1, Dimension: 128 * 128 *3,

It provided better results than previous experiment and reconstructed better image considering training on a smaller number of epochs and training set.







```
(700, 128, 128, 3)
(99, 128, 128, 3)
(202, 128, 128, 3)
Epoch 1/20
22/22 [==============================] - 53s 2s/step - loss: 0.6442 - accuracy: 0.3698 - val_loss: 0.6143 - val_accuracy: 0.4133
Epoch 2/20
22/22 [==============================] - 48s 2s/step - loss: 0.6092 - accuracy: 0.4083 - val_loss: 0.6055 - val_accuracy: 0.3993
Epoch 3/20
22/22 [==============================] - 47s 2s/step - loss: 0.6021 - accuracy: 0.4112 - val_loss: 0.6032 - val_accuracy: 0.4204
Epoch 4/20
22/22 [==============================] - 46s 2s/step - loss: 0.5993 - accuracy: 0.4141 - val_loss: 0.5988 - val_accuracy: 0.4180
Epoch 5/20
22/22 [==============================] - 46s 2s/step - loss: 0.6012 - accuracy: 0.4163 - val_loss: 0.6000 - val_accuracy: 0.3997
Epoch 6/20
22/22 [==============================] - 52s 2s/step - loss: 0.5985 - accuracy: 0.4113 - val_loss: 0.5973 - val_accuracy: 0.4235
Epoch 7/20
22/22 [==============================] - 48s 2s/step - loss: 0.5955 - accuracy: 0.4171 - val_loss: 0.5955 - val_accuracy: 0.4228
Epoch 8/20
22/22 [==============================] - 41s 2s/step - loss: 0.5940 - accuracy: 0.4199 - val_loss: 0.5943 - val_accuracy: 0.4256
Epoch 9/20
22/22 [==============================] - 41s 2s/step - loss: 0.5931 - accuracy: 0.4223 - val_loss: 0.5933 - val_accuracy: 0.4280
Epoch 10/20
22/22 [==============================] - 50s 2s/step - loss: 0.5920 - accuracy: 0.4262 - val_loss: 0.5936 - val_accuracy: 0.4336
Epoch 11/20
22/22 [==============================] - 42s 2s/step - loss: 0.5911 - accuracy: 0.4311 - val_loss: 0.5912 - val_accuracy: 0.4367
Epoch 12/20
22/22 [==============================] - 42s 2s/step - loss: 0.5902 - accuracy: 0.4338 - val_loss: 0.5918 - val_accuracy: 0.4457
Epoch 13/20
22/22 [==============================] - 45s 2s/step - loss: 0.5893 - accuracy: 0.4365 - val_loss: 0.5892 - val_accuracy: 0.4401
Epoch 14/20
22/22 [==============================] - 42s 2s/step - loss: 0.5894 - accuracy: 0.4380 - val_loss: 0.5895 - val_accuracy: 0.4421
Epoch 15/20
22/22 [==============================] - 41s 2s/step - loss: 0.5879 - accuracy: 0.4417 - val_loss: 0.5879 - val_accuracy: 0.4492
Epoch 16/20
22/22 [==============================] - 44s 2s/step - loss: 0.5871 - accuracy: 0.4422 - val_loss: 0.5900 - val_accuracy: 0.4527
Epoch 17/20
22/22 [==============================] - 45s 2s/step - loss: 0.5869 - accuracy: 0.4429 - val_loss: 0.5869 - val_accuracy: 0.4536
Epoch 18/20
22/22 [==============================] - 43s 2s/step - loss: 0.5863 - accuracy: 0.4457 - val_loss: 0.5878 - val_accuracy: 0.4553
Epoch 19/20
22/22 [==============================] - 45s 2s/step - loss: 0.5860 - accuracy: 0.4472 - val_loss: 0.5862 - val_accuracy: 0.4566
Epoch 20/20
22/22 [==============================] - 46s 2s/step - loss: 0.5850 - accuracy: 0.4494 - val_loss: 0.5856 - val_accuracy: 0.4495
7/7 [==============================] - 3s 356ms/step
7/7 [==============================] - 3s 398ms/step
```

Peak Signal-to-Noise Ratio (PSNR): 11.9429

MSE: 0.06324585354260474, MAE: 0.2062094751520004

Task 2:

I have used pretrained encoder from previous task and exported that to same folder where Task 2 python file can fetch it and load it to extract features.

I have also added same layers in encoders as defined in Task 1.

References:

https://keras.io/examples/vision/autoencoder/

https://blog.keras.io/building-autoencoders-in-keras.html

https://www.tensorflow.org/datasets/catalog/stl10

Other multiple resources from internet like github etc.