ML2 Assignment 3 Report

Atanu Guin MT19AI002

Question: Implement a simple encoder-decoder neural network with one hidden layer using the following images (images of George Bush from the LFW Face database):
http://vis-www.cs.umass.edu/lfw/lfw-bush.tgz
Use the MSE loss and 70:20:10 train:val:test split to train the network, and compare the reconstruction error with a simple (linear)PCA (use the same split, and vary the number of the top few eigenvectors during reconstruction).
Note: Include some qualitative results/analysis in the submission report.

Answer:

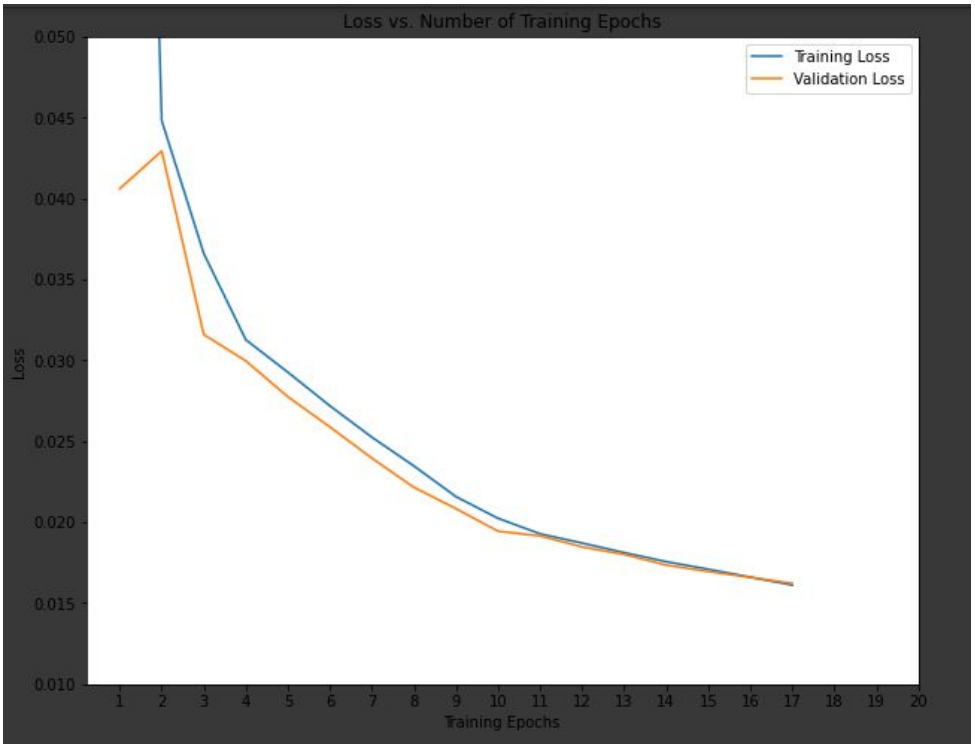
Simple encoder-decoder architecture is used for reconstruction of images from a latent representation. The **LFW Face database** includes face images of George W. Bush. This includes a total of 530 images. As instructed, I have split the dataset into 70:20:10 in train, validation and test dataset which resulted in Train size of 371, Validation size of 106 and Test size of 53. For reproducibility, I have considered random seed value as 100.

Preprocessing:
Since the image size is very small(32*32), first I have transformed the dataset into larger image size of 128*128 and then cropped the image into (64*64) size. This would remove the unnecessary part from each image and will only retain the part which contains the face. Training, validation and testing batches contain 64 images.A training batch is shown below -

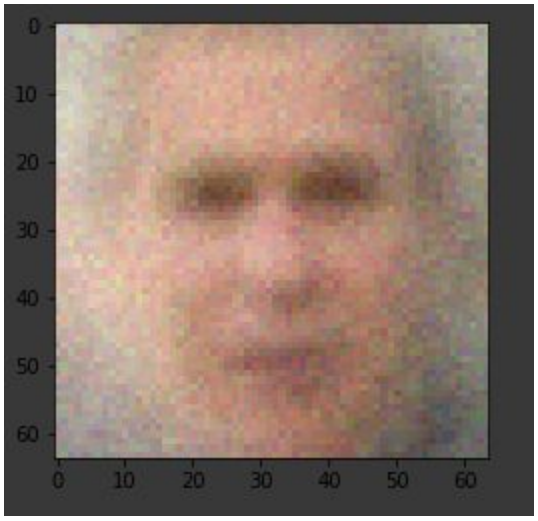
Encoder-Decoder Model:
Encoder decoder model is a simple 2 layer network. The model is implemented in pytorch. Since I have used an image size of 64*64*3, the input size for the model is 12288. The output size is the same as input size as I am trying to reconstruct the same image. The hidden layer is chosen as 10240. I have done much experimenting with the hidden layer size. In order to get the features for a single hidden layer, its size should be comparable to the input size. Lower hidden size could not capture the features of images and much higher hidden size will overfit the model. For activation functions ReLU has been used.

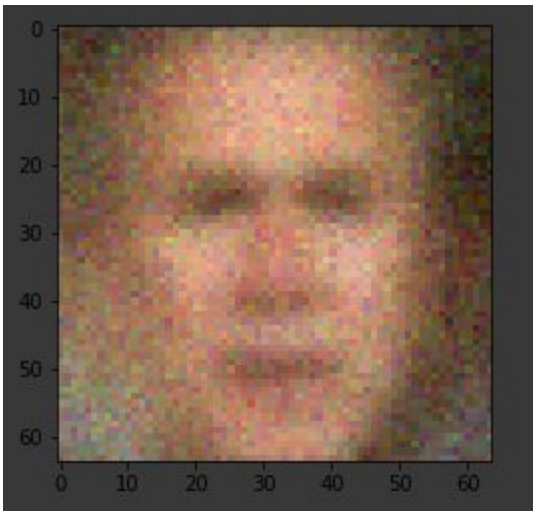
```
Encoder_Decoder(
  (encoder): Sequential(
    (0): Linear(in_features=12288, out_features=10240, bias=True)
    (1): ReLU()
  )
  (decoder): Sequential(
    (0): Linear(in_features=10240, out_features=12288, bias=True)
  )
)
```

Training the model parameters:
For training the model parameters, I have chosen the epoch as 20, learning rate 0.0005 and Adam as optimizer. For avoiding overfitting, I have saved the model parameters based on best validation loss. Since more number of epochs tends to overfit the model, I have used an early stopping method when validation loss starts increasing than training loss.
The best validation loss was measured as 0.0166 which was the same as the training loss, when the model starts overfitting. The same can be shown in the graph below.



Test result: For testing, I have got 53 images. On using the train model on the test data, we have obtained reconstruction error of 1.72%. The test results are shown below.





Principal Component Analysis (PCA):

For train,validation and testing dataset, I have used the PCA for reducing the dimension of the corresponding dataset.

I have chosen the number of Principal Components are 50,40,30,23,17. The reconstruction errors with the data variance are shown below.

```
For 50 components PCA details:
         PCA(copy=True, iterated_power='auto', n_components=50, random_state=42,
    svd_solver='auto', tol=0.0, whiten=False)
For Training:
        Projection Shape: (371, 50)
        Projection Variance: 0.85705847
For Validation:
        Projection Shape: (106, 50)
        Projection Variance: 0.92915004
For Test:
        Projection Shape: (53, 50)
        Projection Variance: 0.9960617
```

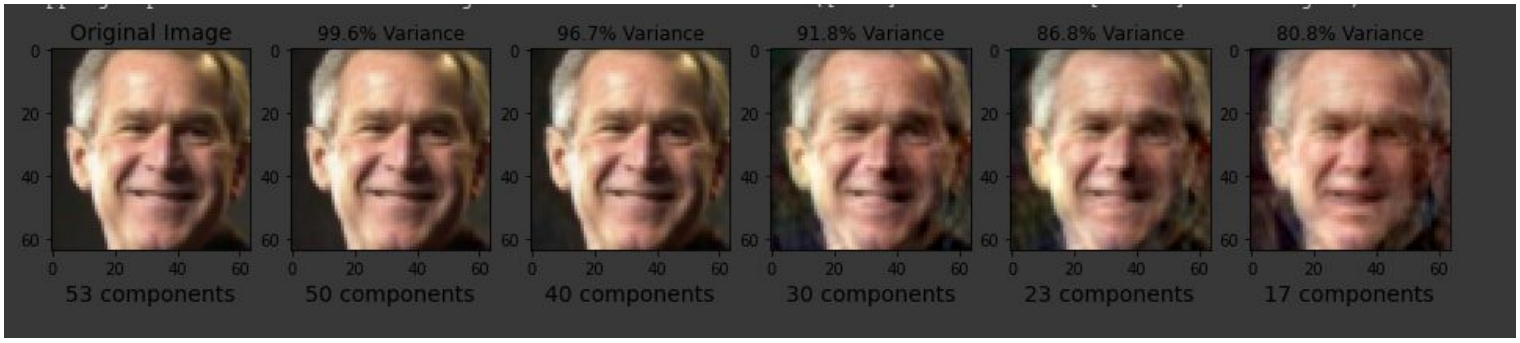
```
For 40 components PCA details:
         PCA(copy=True, iterated_power='auto', n_components=40, random_state=42,
    svd_solver='auto', tol=0.0, whiten=False)
For Training:
        Projection Shape: (371, 40)
        Projection Variance: 0.8303424
For Validation:
        Projection Shape: (106, 40)
        Projection Variance: 0.89949846
For Test:
        Projection Shape: (53, 40)
        Projection Variance: 0.96650434
```

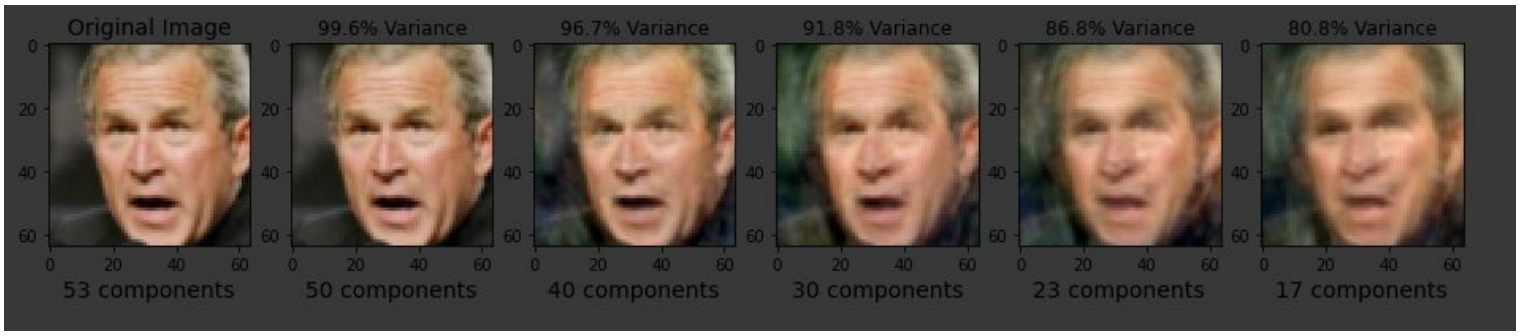
```
For 30 components PCA details:
         PCA(copy=True, iterated_power='auto', n_components=30, random_state=42,
    svd_solver='auto', tol=0.0, whiten=False)
For Training:
        Projection Shape: (371, 30)
        Projection Variance: 0.79378796
For Validation:
        Projection Shape: (106, 30)
        Projection Variance: 0.8588522
For Test:
        Projection Shape: (53, 30)
        Projection Variance: 0.91800964
```

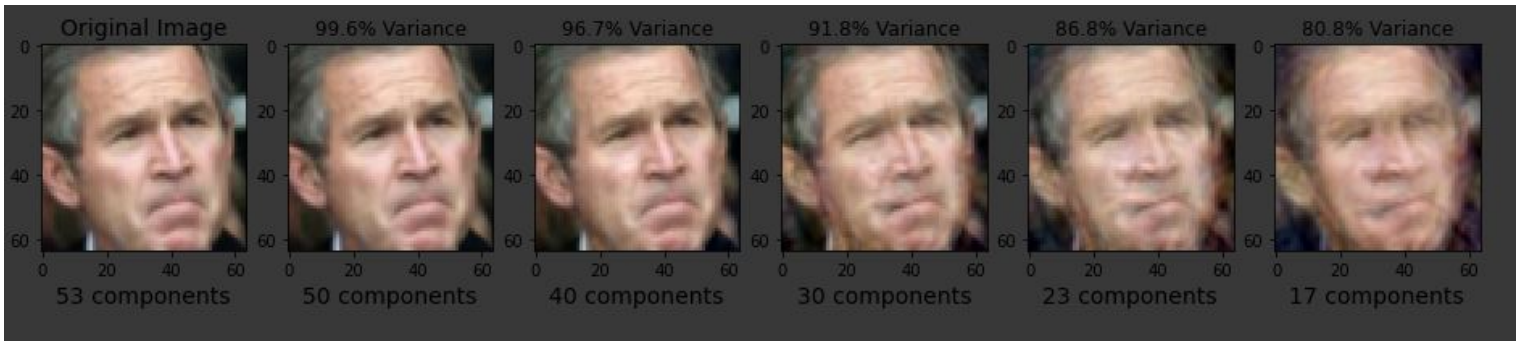
```
For 23 components PCA details:
         PCA(copy=True, iterated_power='auto', n_components=23, random_state=42,
    svd_solver='auto', tol=0.0, whiten=False)
For Training:
        Projection Shape: (371, 23)
        Projection Variance: 0.75903004
For Validation:
        Projection Shape: (106, 23)
        Projection Variance: 0.8180418
For Test:
        Projection Shape: (53, 23)
        Projection Variance: 0.8678451
```

```
For 17 components PCA details:
         PCA(copy=True, iterated_power='auto', n_components=17, random_state=42,
    svd_solver='auto', tol=0.0, whiten=False)
For Training:
        Projection Shape: (371, 17)
        Projection Variance: 0.71349615
For Validation:
        Projection Shape: (106, 17)
        Projection Variance: 0.7679257
For Test:
        Projection Shape: (53, 17)
        Projection Variance: 0.8079023
```

The reconstruction images with original images are shown below.







Analysis: As the number of components are decreasing, we can see the blurry images. The data variance is decreasing with the component loss.

Qualitative Analysis:

1. For the Encoder-Decoder model even though the reconstruction error is small, the single hidden layer is only able to capture the generalized features and not able to learn more specific features.
2. The size of the hidden layer tries to learn the features of the images. If we use very small hidden sizes, the reconstructed images become identical.
3. Encoder-decoder model tries to find a common feature of the whole batch. If there are a variety of images in a batch, the reconstructed images become different. I have used the same model on MNIST dataset and got decent reconstructed images.

4. PCA is the dimension reduction technique which uses the maximum variance components for image reconstruction whereas encoder-decoder models are trying to learn the features through epochs. So the generalizability factor is more prevalent in encoder-decoder models.