ML Assignment - 3 Report

Abhirama Subramanyam
penamakuri.1@iitj.ac.in

July 10, 2020

The objective of this assignment is to implement a simple encoder-decoder neural network with one hidden layer using LFW-George Bush Images, and compare the reconstruction loss with a simple linear PCA (by varying no. of principal components)

1 Encoder Decoder Model

1. **Convolutional Auto Encoder** (one hidden layer)
   Encoding part of model consists of a conv layer (3x3 kernel, 1 input channel, 8 output channels, zero padding), followed by decoder consisting of a upconv layer (3*3 kernel, 8 input channels, 1 output channel)

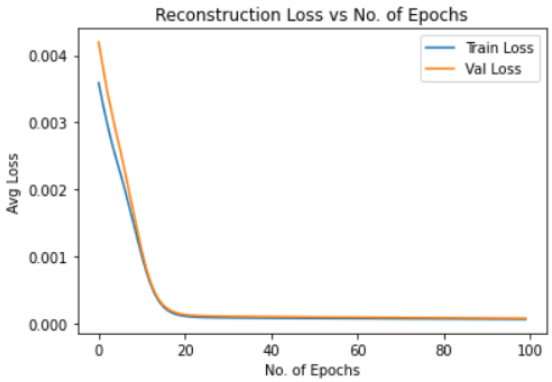


Figure 1: Train/Val Loss (reconstruction) vs No. of epochs

2. **Linear Auto Encoder** (one hidden layer)
   Encoding part of model consists of Linear layer that maps 62500 input vector to 512/256 dimensional hidden representation, followed by a decoder consisting of Linear Layer that maps 512/256 dimensional hidden representation to 62500 dimensional output.

   (a) **512** Hidden Representation - Fig. 2

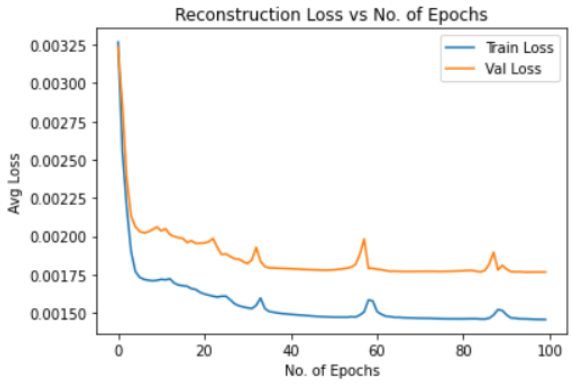


Figure 2: (512 Model) Train/Val Loss (reconstruction) vs No. of epochs

This model was selected for further experiments as loss converged to is lower than 256 representation model.

(b) **256** Hidden Representation - Fig. 3

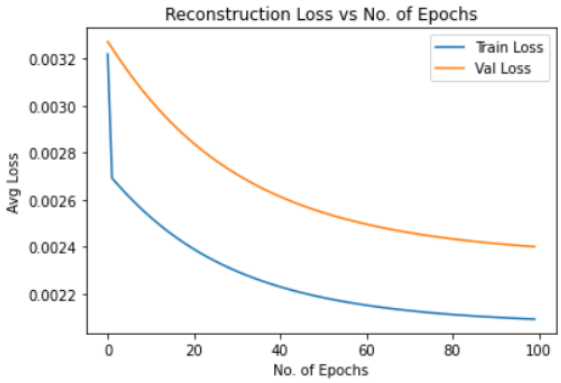


Figure 3: (256 Model) Train/Val Loss (reconstruction) vs No. of epochs

2 Implementation Details

1. 70:20:10 split for train,validation and test respectively.
2. MSELoss.
3. Adam Optimizer - Learning rate 0.001

3 PCA

PCA is implemented with sklearn package, by varying number of principal components used from 10 to 250, and reconstruction loss on test set was plotted, and this reconstruction loss is compared with reconstruction loss from both conv autoencoder and linear autoencoder, Fig. 4

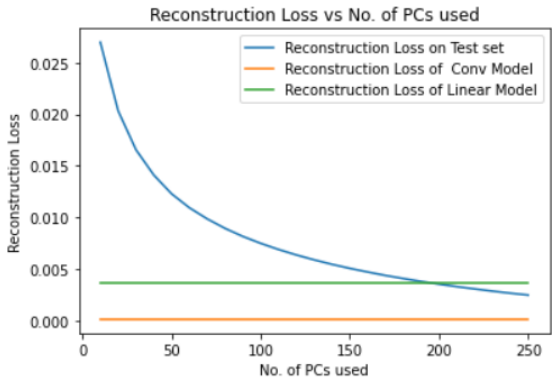
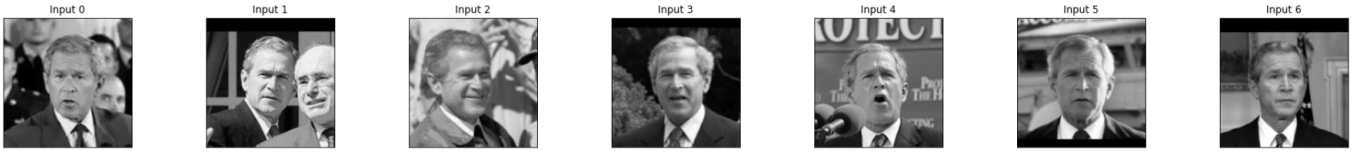


Figure 4: Reconstruction Loss vs No. of PC's used

PCA models trained using 50,100,200 principal components respectively are used for quantitative comparision.

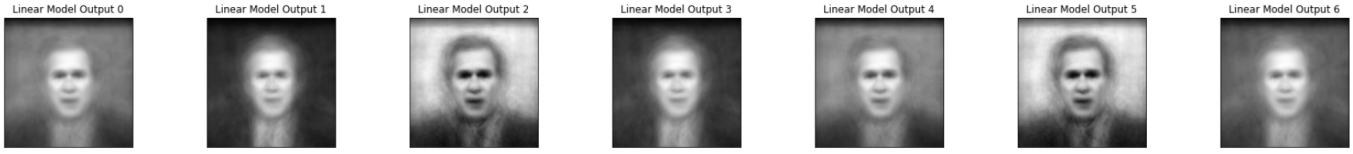
4 Quantitative Examples

1. Test Samples

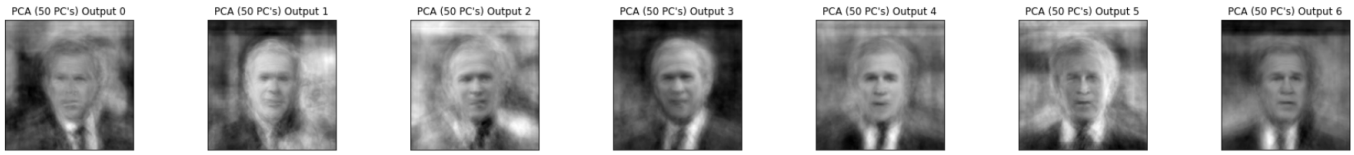


2. Results from Conv Model

3. Results from Linear Model

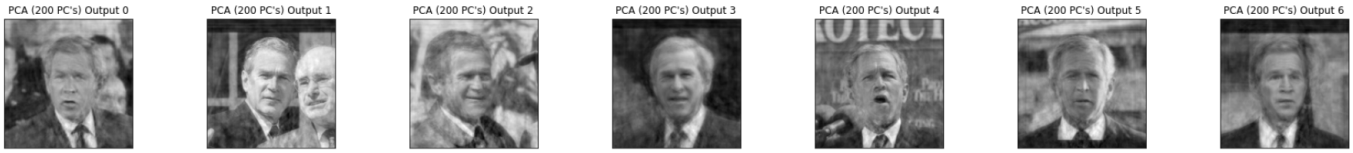


4. Results from PCA (with 50 PC's)



5. Results from PCA (with 100 PC's)

6. Results from PCA (with 200 PC's)



5 Learnings and Observations

1. Reconstruction Loss is lower in Conv Autoencoder than the linear Autoencoder.
2. Reconstructed images from Conv Model look far better than other models (used above) which can be clearly seen from the Quantiative Examples section.
3. Reconstructed images from Linear Model looks wierd (looks similar (not exactly)).
4. Reconstructed images quality increases as no. of principal components used.
5. Much better experience in writing custom dataloaders when compared to Assignment-2.
6. Learnt to implement conv(down and up) full end to end pipeline.
7. Learnt to plot multiple images (subplots) inside a same plot.

6 References

1. https://pytorch.org/tutorials/beginner/finetuning_torchvision_models_tutorial.html
2. https://gist.github.com/AFAgarap/4f8a8d8edf352271fa06d85ba0361f262
3. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

4. https://stackoverflow.com/questions/37799865/valueerror-num-must-be-1-num-2-not-3

5. https://stackoverflow.com/questions/61811946/train-valid-test-split-for-custom-dataset-using-pytorch-and-torchvision

6. https://github.com/utkuozbulak/pytorch-custom-dataset-examples

7 Files in the zip folder

1. *P19CSE001_ML_Assignment_3.ipynb*

2. *P19CSE001_ML_Assignment3_report.pdf*

3. *P19CSE001_Plots* - contains all the plots obtained and reported.