CIS680 HW3

March 13, 2019

1 Variational Autoencoders

1.1 Autoencoder for CUFS

Figure 1: Loss over training iterations

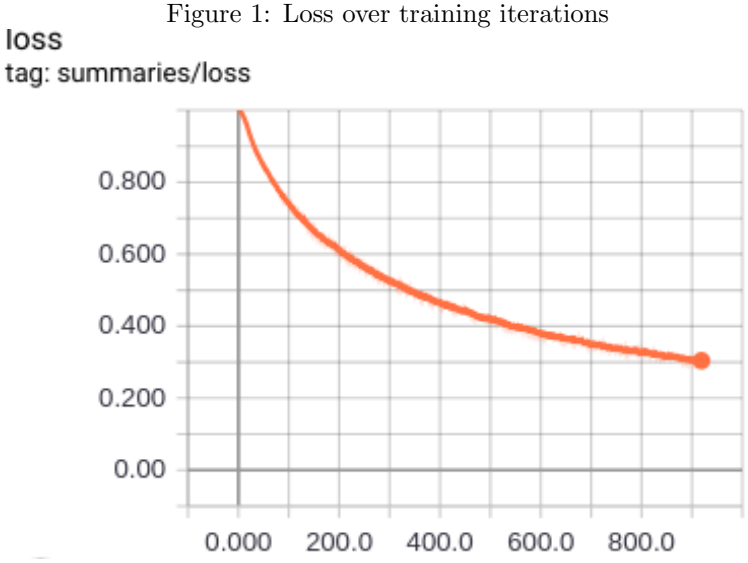
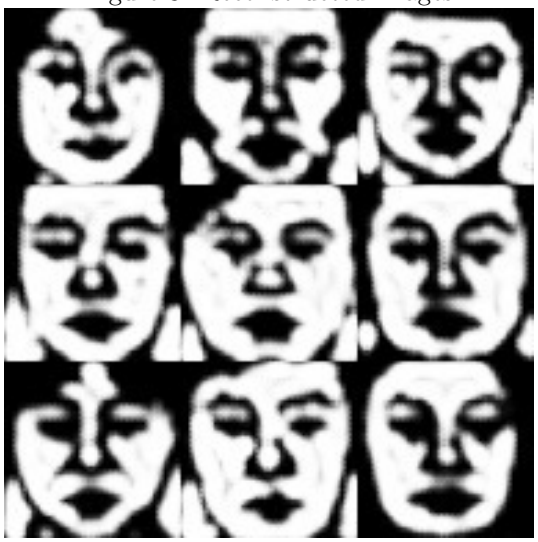


Figure 2: Real images

Figure 3: Reconstructed images



1.2 VAE for CUFS

Figure 4: KL over training iterations

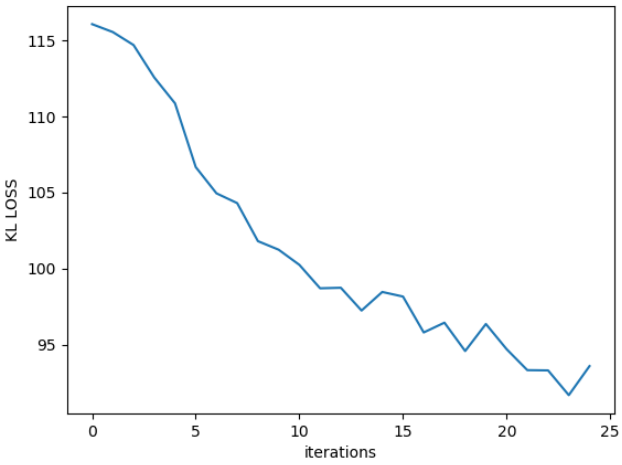


Figure 5: Loss over training iterations

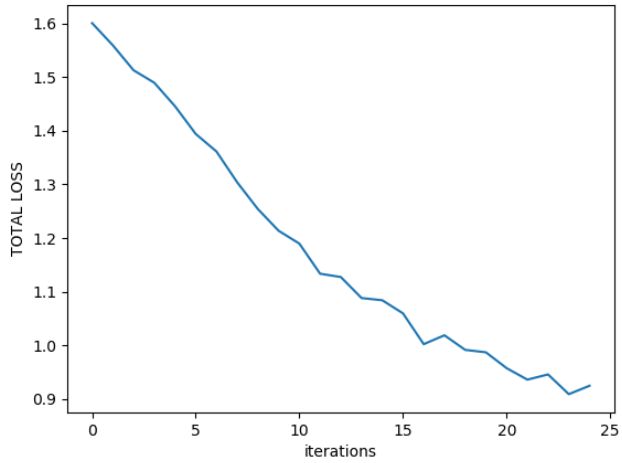


Figure 6: Reconstruction loss over training iterations

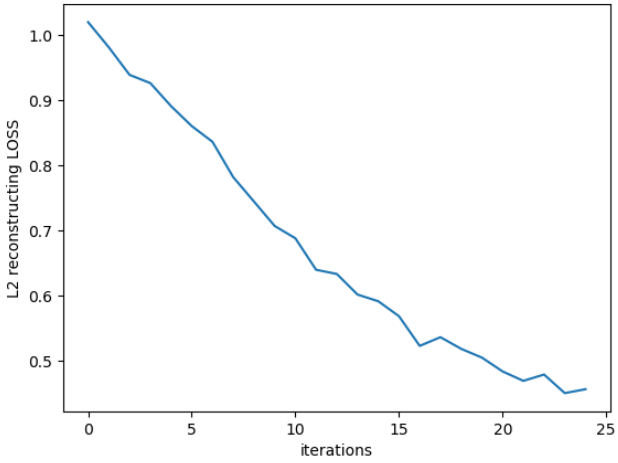


Figure 7: The first column corresponds to true images, second column to the corresponding generated images, the third column to randomly generated images.

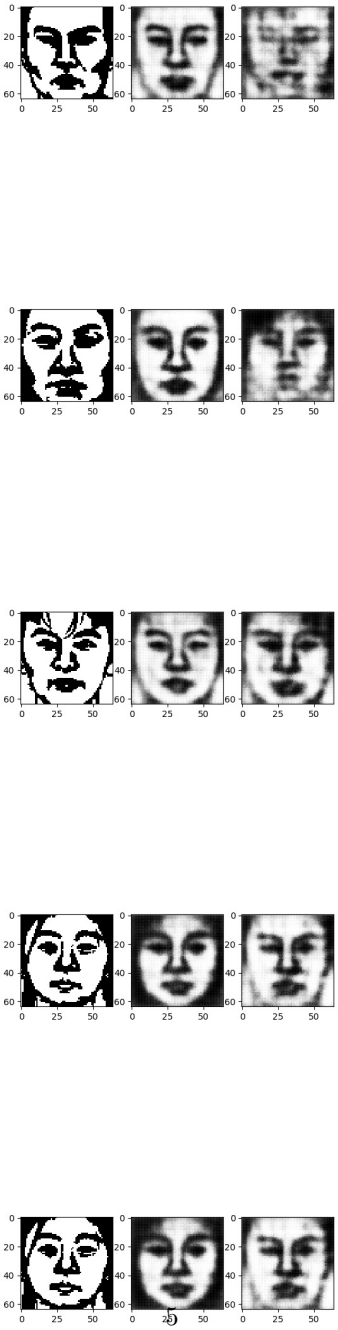


Figure 8: After computing z's from two images .

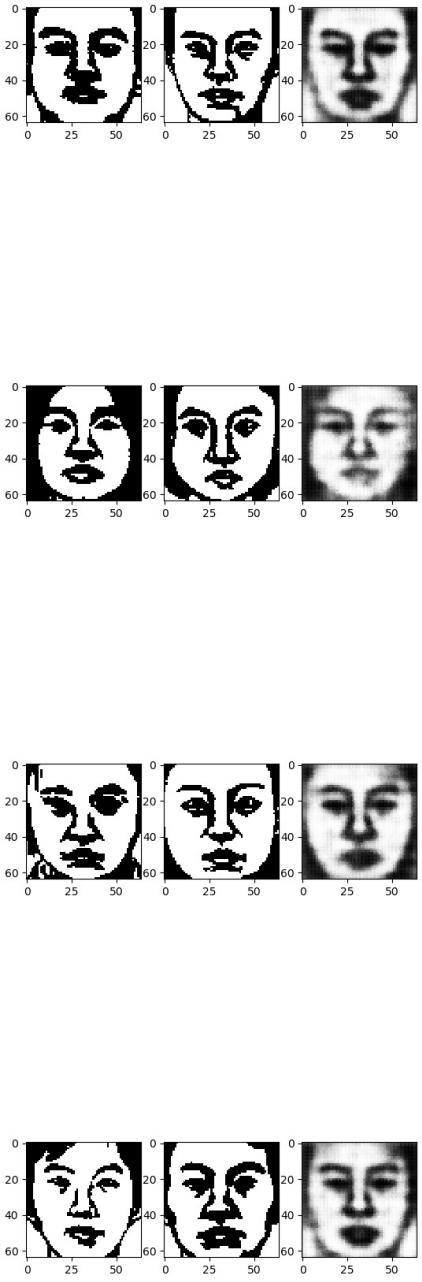
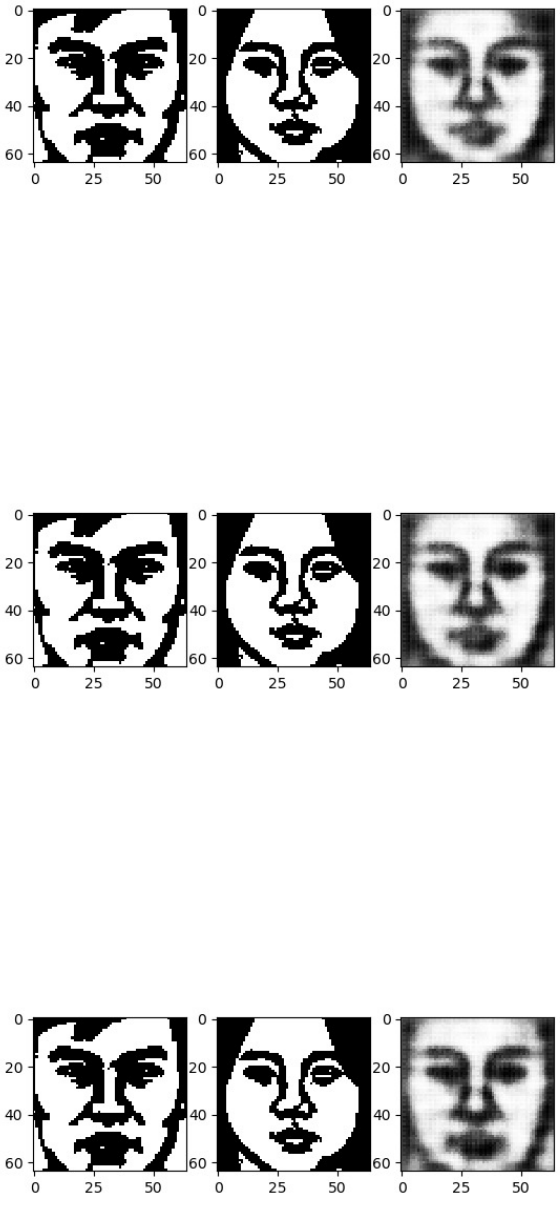


Figure 9: After computing z's from two images and a series of reconstructions of the combinations of z's. First row is with $50\% - 50\%$ weightage, second row is with $30\% - 70\%$ weightage, and the third row is with $70\% - 30\%$ weightage



VAE - CELEBA
Batch size: 200
Optimiser: Adam
Number of epochs: 5
Learning rate: 0.001
Please refer to the code for the full architecture

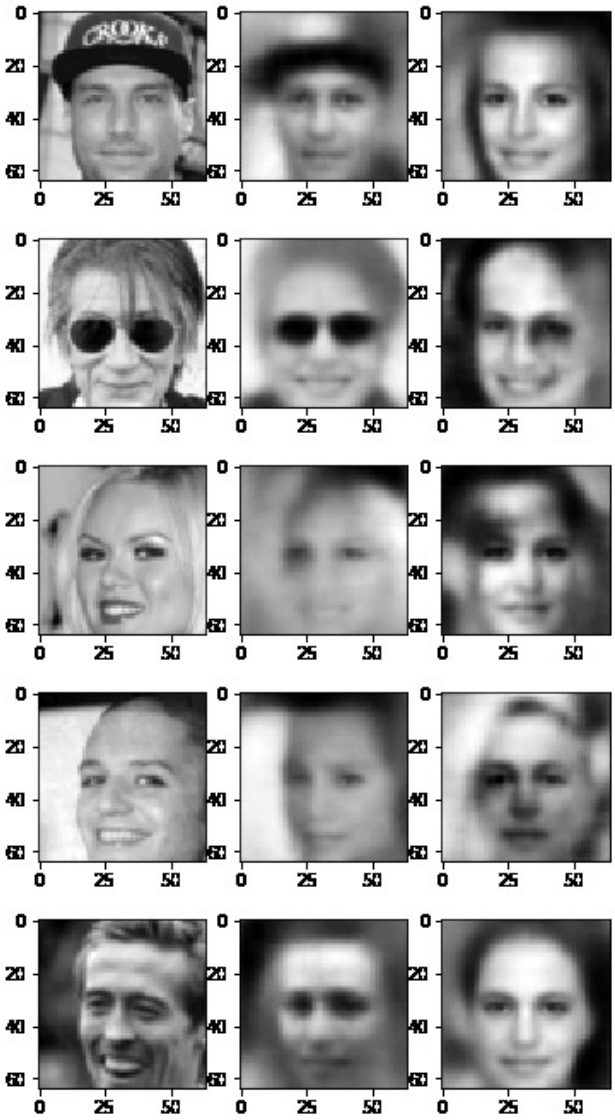


Figure 10: Generated Images

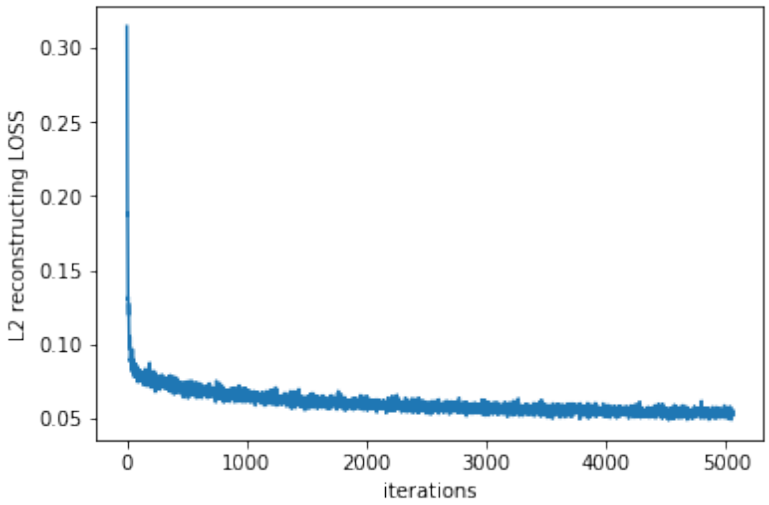


Figure 12: l2 loss vs iterations

The first column corresponds to true images, second column to the corresponding generated images, the third column to randomly generated images.

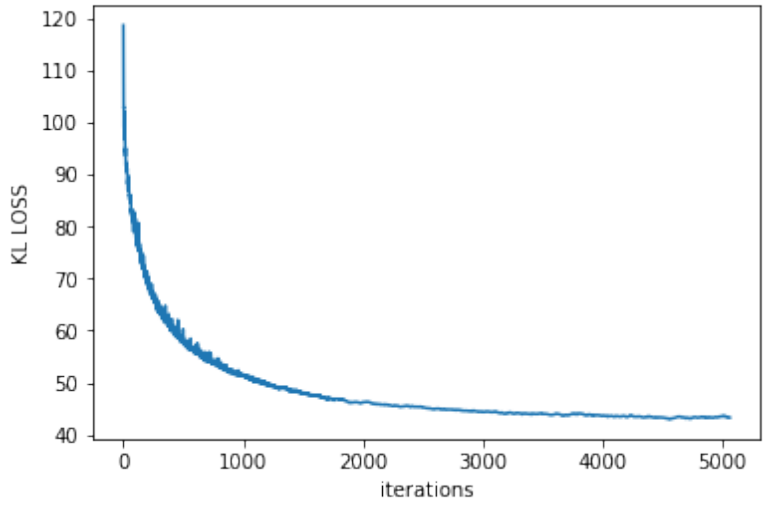


Figure 11: Kl loss vs iterations

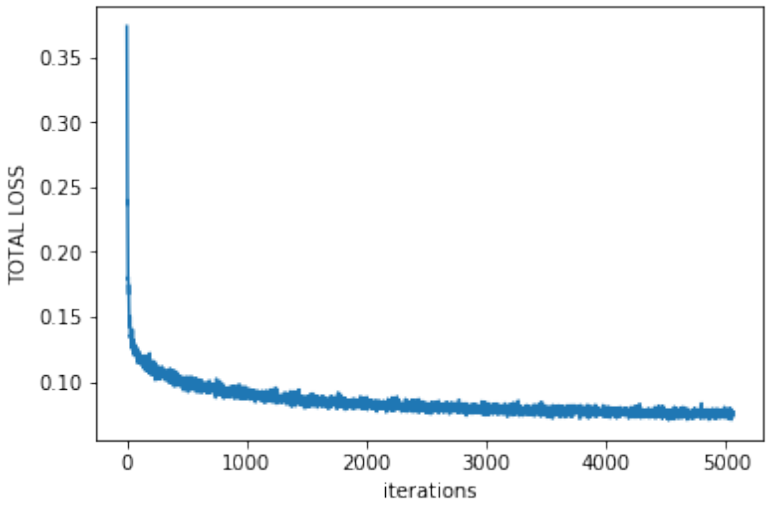


Figure 13: total loss vs iterations

2 GAN

DCGAN on CUFS
Batch size: 47
Optimiser: Adam
Number of epochs: 120
Learning rate: 0.02
Please refer to the code for the full architecture

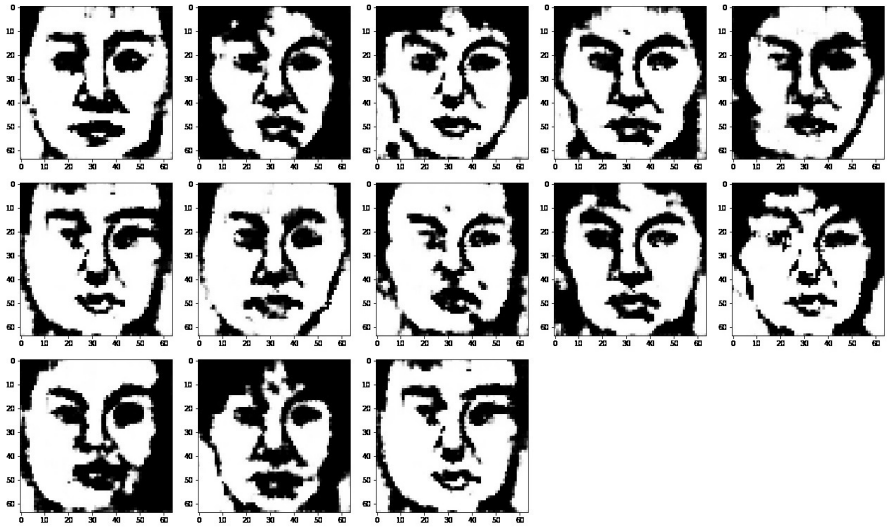


Figure 14: Generated Images

we can see that the images generated show some degree of similarity. This problem is commonly termed as mode collapse. The reason we believe is due to the less number of training images.

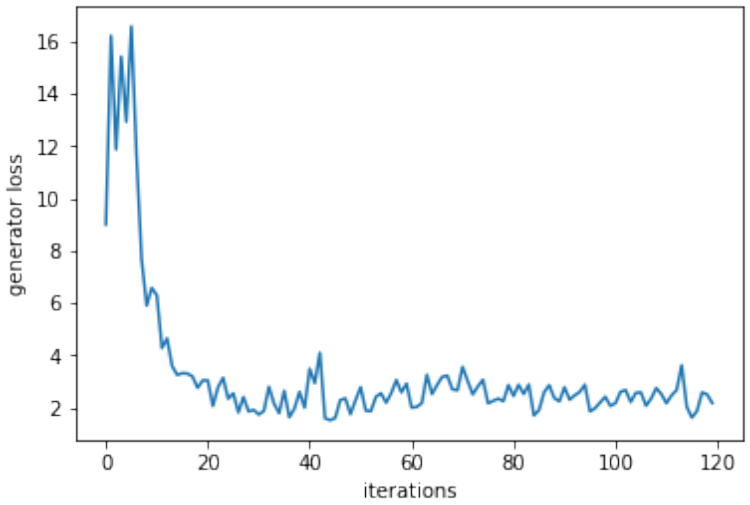


Figure 15: average Generator loss per epoch vs epochs

DCGAN on Celeba
Batch size: 200

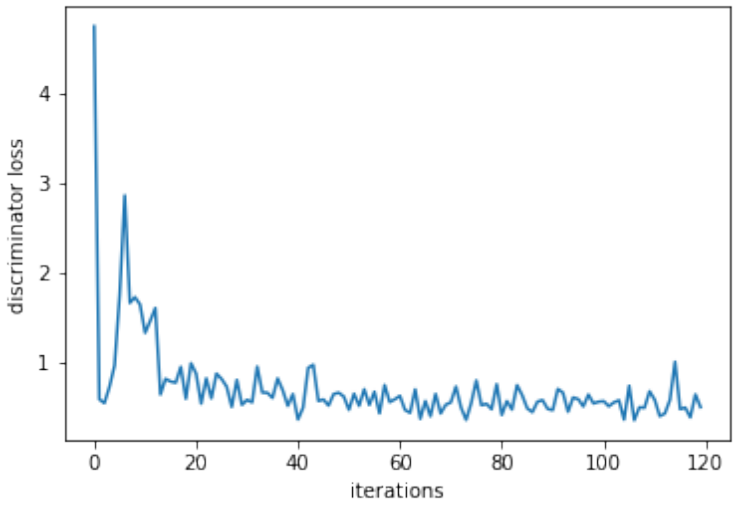


Figure 16: average Discriminator loss per epoch vs epochs

Optimiser: Adam
Number of epochs: 10
Learning rate: 0.0003
Please refer to the code for the full architecture

Figure 17: Generated Images

The problem in the previous question has been resolved. A diverse set of images were generated. The reason we believe is due to the large amount of training data, the generator cannot fool the discriminator too easily and is forced to learn a better representative distribution of the data.

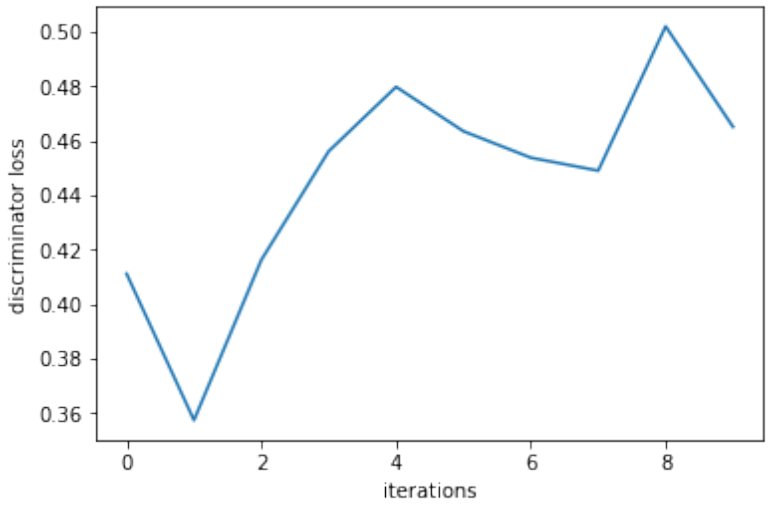


Figure 19: average Discriminator loss per epoch vs epochs

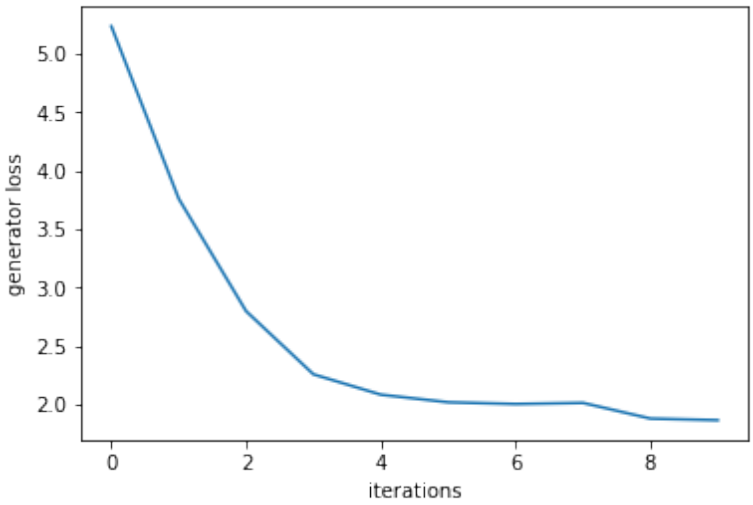


Figure 18: average Generator loss per epoch vs epochs

LSGAN on Celeba

For this part, we tried training LS GAN. The architecture is similar to the

DCGAN with the loss modified.

$$Discriminatorloss = -[(1 - D(X))^2 - (1 - D(G(Z))^2]$$
$$Generatorloss = -[(1 - D(G(Z)))^2]$$

Where X - true images, G(Z) - generated images, D - refers to the discriminator.
We **minimise** the losses using Adam optimizer. Please refer to the code for the implementation details.
Parameters: Batch size: 200
Optimiser: Adam
Number of epochs: 10
Learning rate: 0.0001

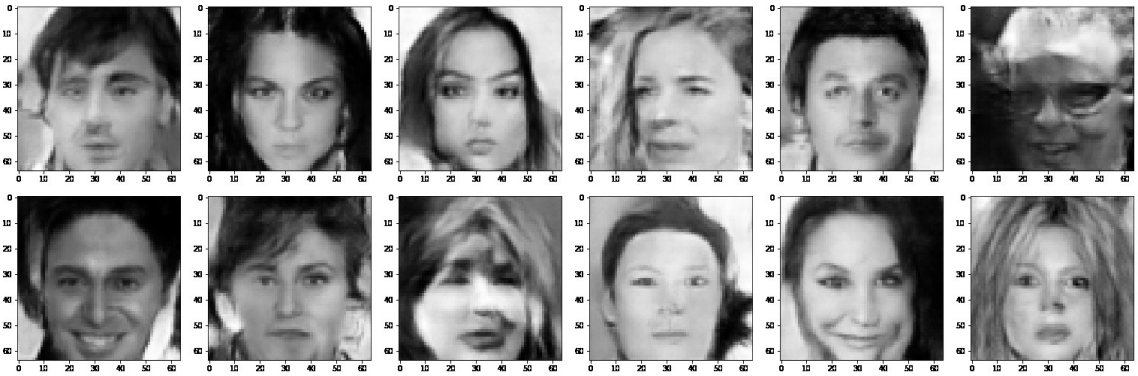


Figure 20: Generated Images

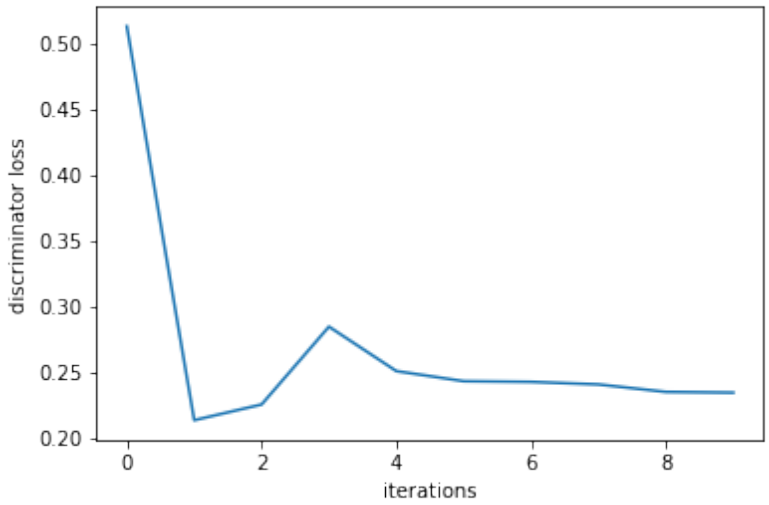


Figure 22: average Discriminator loss per epoch vs epochs

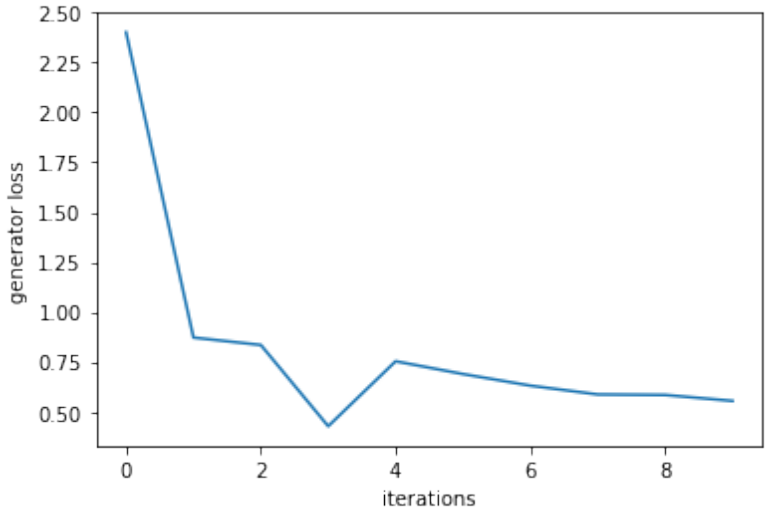


Figure 21: average Generator loss per epoch vs epochs