IFT6135 Representation Learning (Programming Assignment 4)

Names: Isaac Sultan, Breandan Considine, Hossein Askari **Due Date:** April 20^{th} 2018

1 Architecture

We implemented a GAN with latent dimension 10^2, in order to generate novel faces from the *CelebFaces Attributes Dataset*. The dataset downloaded contains nearly 10000 images. A DCGAN architecture was used. Three techniques for increasing feature-map size were inspected:

(a) Deconvolution (transposed convolution) with paddings and strides.

(b) Nearest-Neighbor Upsampling followed by regular convolution.

(c) Bilinear Upsampling followed by regular convolution.

In **(a)**, a transposed convolution layer, the forward and backward passes of a traditional convolution layer are reversed. It can be thought of as a gradient of some convolution with respect to its input. With this procedure, the spatial transformation of a regular convolution (with the same parameters) are reversed - granting a feature map equal or larger than its input. It has been noted that **(a)** produces an overlap pattern in two dimensions [5], which is most prominent when outputting unusual colors.

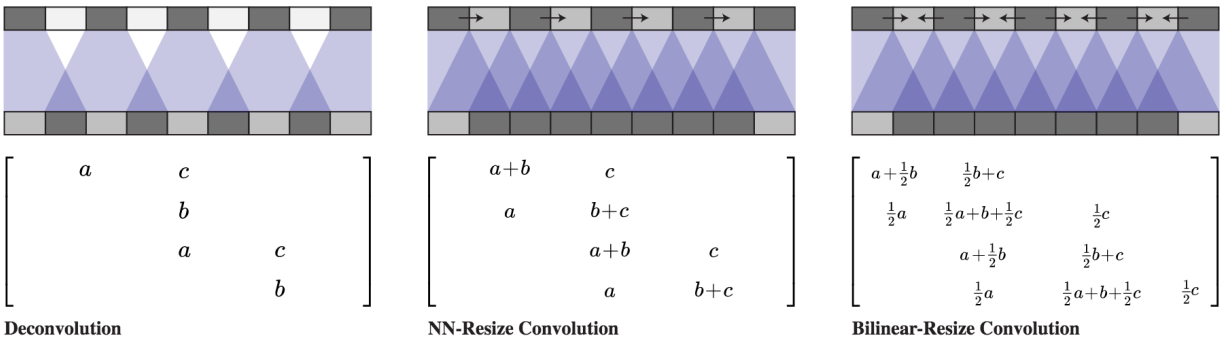


Figure 1: A comparison of three techniques for increasing feature-map size[5]. From left to right, we see the Deconvolution (transposed convolution) with paddings and strides. Then the Nearest-Neighbor Upsampling followed by regular convolution is illustrated. Finally, the Bilinear Upsampling followed by regular convolution.

In order to avoid the aforementioned visual artifacts, two additional procedures may be implemented. In **(b)**, the nearest-neighbor upsampling selects only the value of the nearest point for interpolation. To implement this, an image may have deconvolution applied using a fixed kernel with all values equal to 1. In **(c)**, the closest 4 pixels of each pixel are considered, returning the weighted average of these values.

In Figure 1, we demonstrate the arithmetic involved in **(a)**, **(b)** and **(c)**. From left to right, we first demonstrate the Deconvolution (transposed convolution) with paddings and strides. Then the Nearest-Neighbor Upsampling followed by regular convolution is illustrated. Finally, the Bilinear Upsampling followed by regular convolution.

2 Model Variants and Qualitative Evaluations

2.1 Baseline DCGAN

We decided to implement our model with **deconvolution**, taking inspiration from the DCGAN [6] architecture. The model has five transposed convolutional layers, followed by batch normalization on all layers except the output layer. Both the discriminator and generator employ 'Leaky ReLu' non-linearities until the output layer. The discriminator then employs a sigmoid activation function and the generator uses hyperbolic tangent respectively. We employ the 'ADAM' optimizer with an initial learning rate of $2e^{-4}$. Figure 2 visualizes the Generator architecture for this model.

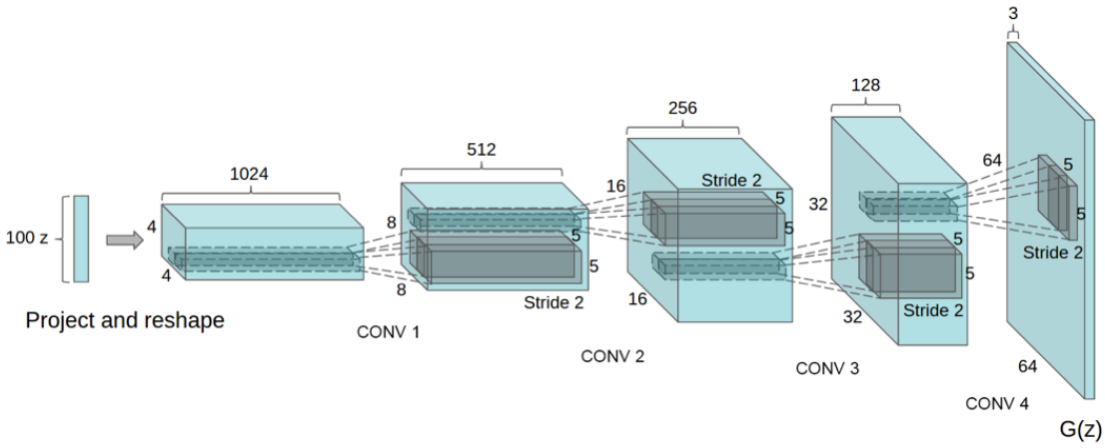


Figure 2: The generator of the original DCGAN model [6]

.

Our model was trained on images from the 'CelebA' dataset[1]. After running the model for

[1] https://www.dropbox.com/sh/8oqt9vytwxb3s4r/AADSNUu0bseoCKuxuI5ZeTl1a/Img?dl=0&preview=img_align_celeba.zip

50 epochs, the best images generated by the DCGAN were on epoch 41 to 50 and especially epoch 49 as shown in Figure 3.

Figure 3: DCGAN on epoch 49 with zdim = 100

As commonly observed in the literature, there was a tendency towards what it is deemed 'mode collapse' - a low diversity of samples produced by the generator. In Table 1 we demonstrate this phenomenon by displaying early images generated by our model. As clearly seen, the samples display a lack of diversity, with the same or similar face repeated. While

this phenomenon is unpredictable, there are measures which can be taken to reduce its severity. As shown in Figure 3 and Table 1 we found that training for longer period of time seems to help with reducing the mode collapse effect. However, we also found that this might not be true for every combination of hyperparameters and latent space dimensions.

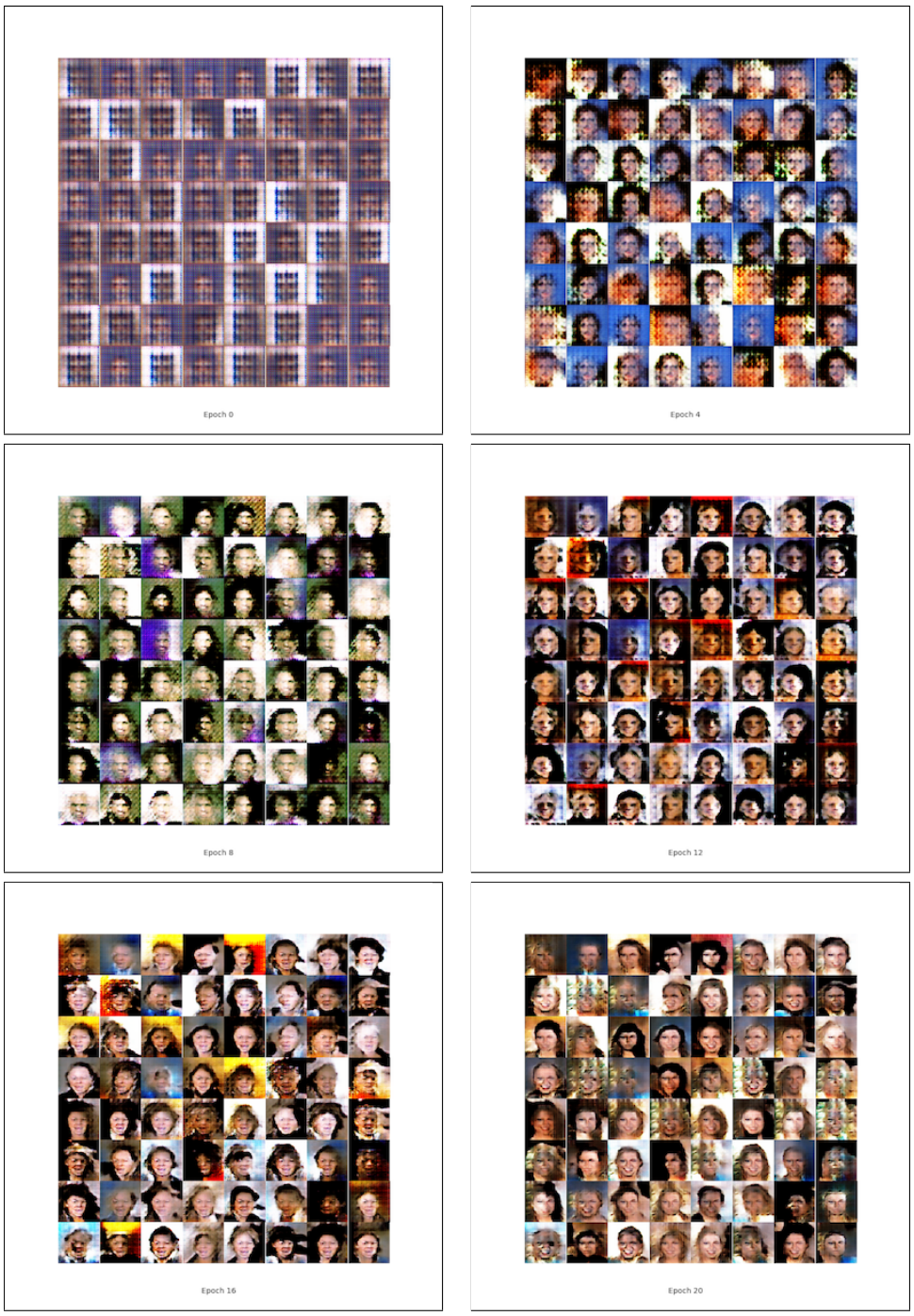


Table 1: This table shows the effect of Mode Collapse. As it can be seen, from top left to bottom right, we can see that the model occasionally outputs similar celebrity faces.

In the following sub-section we discuss more reliable methods to solve the mode collapse.

2.2 Variants

One such measure that is particularly beneficial to training a DCGAN-like model is virtual batch normalization (VBN) [8]. A vanilla batch normalization layer may cause the output of a network layer to be highly dependent on other inputs from the same minibatch. Instead, VBN normalizes each input based on a the statistics of a fixed reference batch and itself. This will help to reduce dependency within a minibatch but is noted to be computationally expensive, requiring two forward passes per minibatch.

Adjusting the objective function of a GAN may also help reduce the effect of mode collapse [1]. We investigated a model with an alternative objective function - the LSGAN [4]. This model replaces the loss function of the discriminator in the first model with a least-squares (LS) loss function. The primary motivation of this model is to reduce the problem of vanishing gradients when updating the generator. As a GAN trains, while the discriminator will correctly classify unrealistic samples, the signals it projects may weaken. By employing a LS loss function, the model will penalize samples that lie far from the correct side of the decision boundary. Mao et al. note a further advantage of the LS loss function: when compared with the sigmoid function (see Figure 4), the LS loss function is flat only at one point. On the other hand, the sigmoid cross entropy loss function will saturate when x is relatively large, causing greater instability.

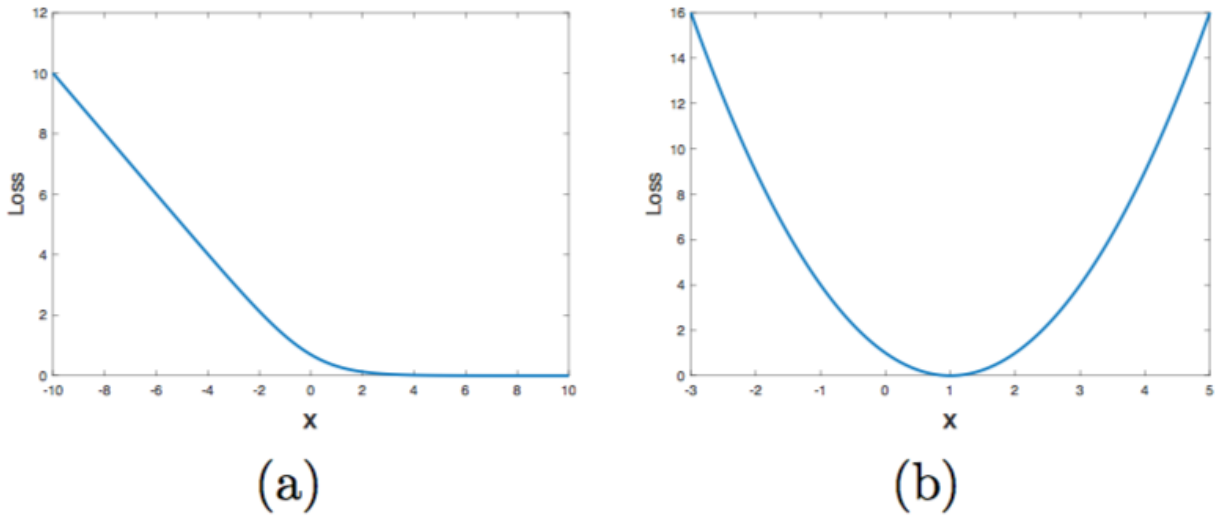


Figure 4: (a): The sigmoid cross entropy loss function. (b): The least squares loss function[4]

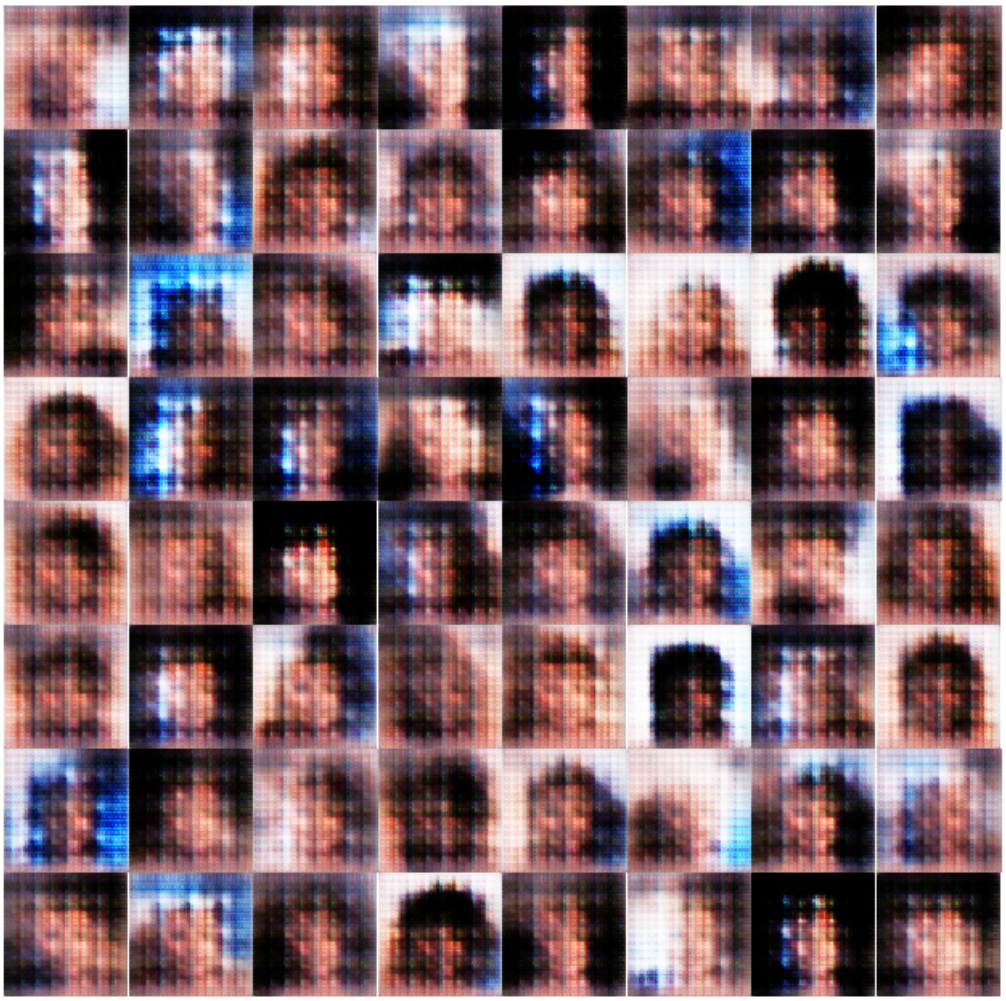


Figure 5: LSGAN on epoch 25 with zdim = 100

While we made no further changes to the original architecture in the first GAN, the resulting images were quite poor, as displayed in Figure 5. While much of our LSGAN's failure was due to insufficient hyperparameter tuning, we believe the changes to the discriminator function of the model are insufficient to reduce mode collapse. This is because the discriminator fails to consider more than one sample at a time, unlike successful mode collapse rectifications such as minibatch discrimination.

2.3 Changes in the Latent Space

We explored making changes in the latent space. Specifically, we changed the z dimension from 100 to 50 to see the effect of this change on the visual variants. Table 2 shows the effect of changing dimension of z. On the left, we have plotted the generated images with z dimension equal to 100. On the right, we have plotted the generated images with z dimension equal to 50. Both models where sampled at epoch 7.

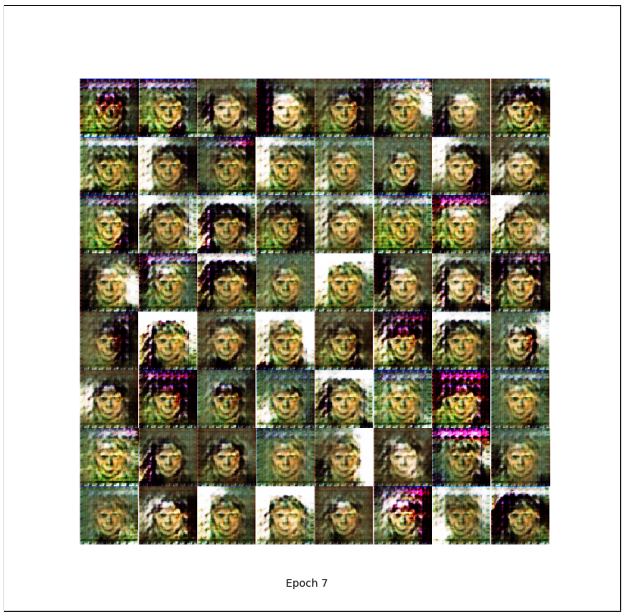


Table 2: This table shows the effect of changing dimension of z. On the left, we have plotted the generated images with z dimension equal to 100. On the right, we have plotted the generated images with z dimension equal to 50

As it can be seen, the model that with a higher dimensional latent space provides better results. We specifically saw this effect for the early stage of the learning. However, after running the model for longer periods of time, the difference became very negligible.

2.4 Interpolation schemes

We used a pretrained model to generate new sample images. Figure 6 shows 64 images that were produced by our pretrained model. Afterwards, we explored moving from any z_0 to z_1. Figure 7 shows the effect of moving in latent space from z_0 to z_1 and generating a new z' which equals to $z' = \alpha z_0 + (1 - \alpha)z_1$. We then interpolated the new z' to the image space. α changes from 0 to 1 in 64 steps. Figure 6 shows our choices for z_0 and z_1 in image space. First we decided which images to sweep and then we captured the corresponding z.

Figure 6: Generated images from our pretrained model. The images with blue and red borders are the ones we want to interpolate in latent space. We also apply same interpolation in image space directly, with results in figure 9.

As it can be seen in Figure 7, the interpolation in z space is fairly smooth, with semantically well-structured images at each point along the interpolation. The image shows α changes from 0 to 1 in 64 steps. It appears the model fits a manifold upon which we can move from z_0 to z_1 smoothly and generate new images along the path which map to realistic images of faces.

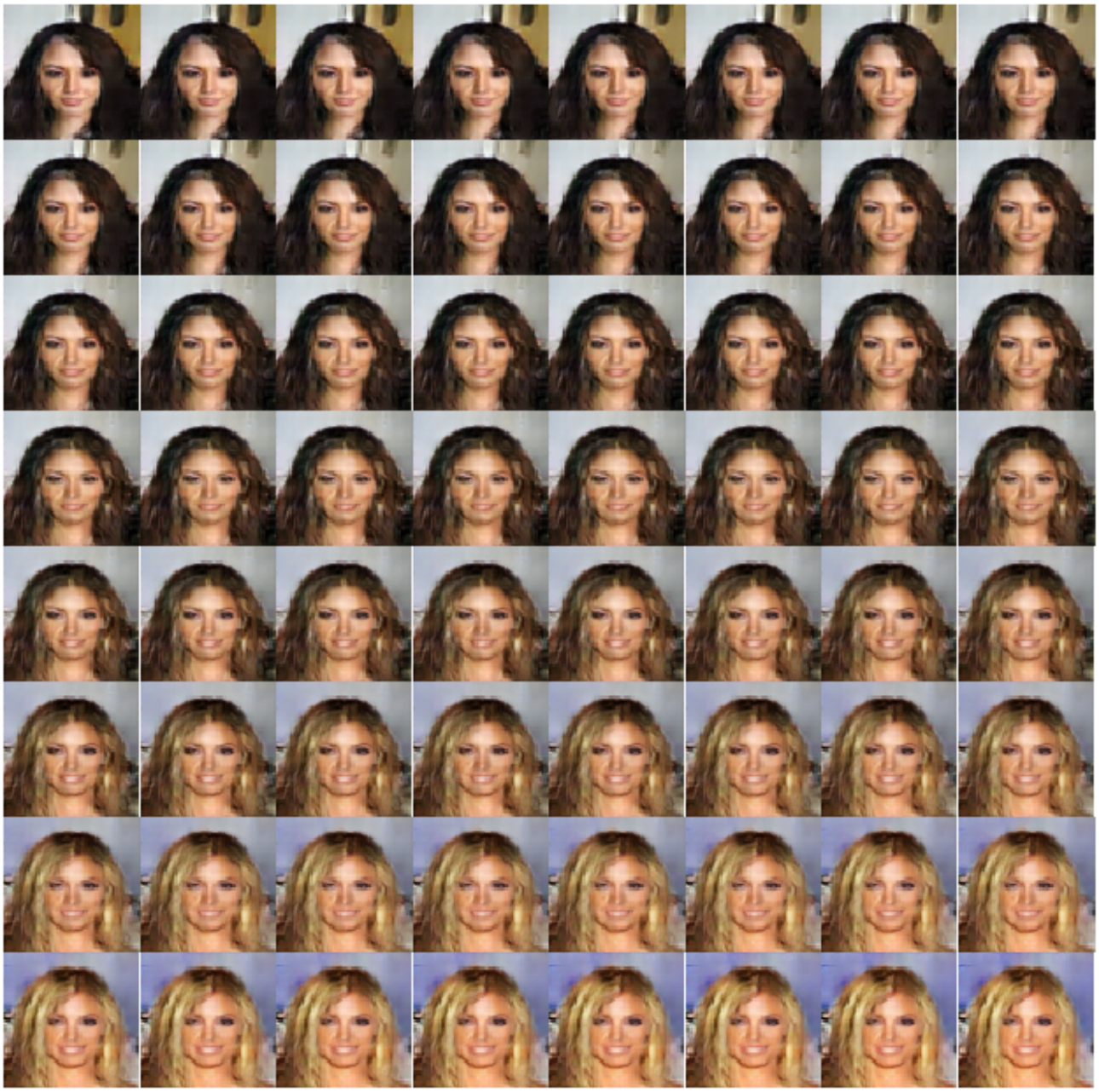


Figure 7: This image shows the effect of moving in latent space from z_0 to z_1 and generating a new z' which equals to $z' = \alpha z_0 + (1-\alpha)z_1$. We then interpolated the new z' to the image space. α changes from 0 to 1 in 64 steps.

In the second experiment for interpolation, we visualize sweeping from x_0 to x_1 with the following equation $x' = \alpha x_0 + (1-\alpha)x_1$. Figure 9 shows the interpolation in x space. As before, α changes from 0 to 1 in 64 steps. As it can be seen the image shows the effect of moving in x space from x_0 to x_1 in which x_0 and x_1 correspond to the z_0 and z_1 discussed earlier. The image shows that moving in x space does not produce any new images, resulting

in negligible changes to the image.

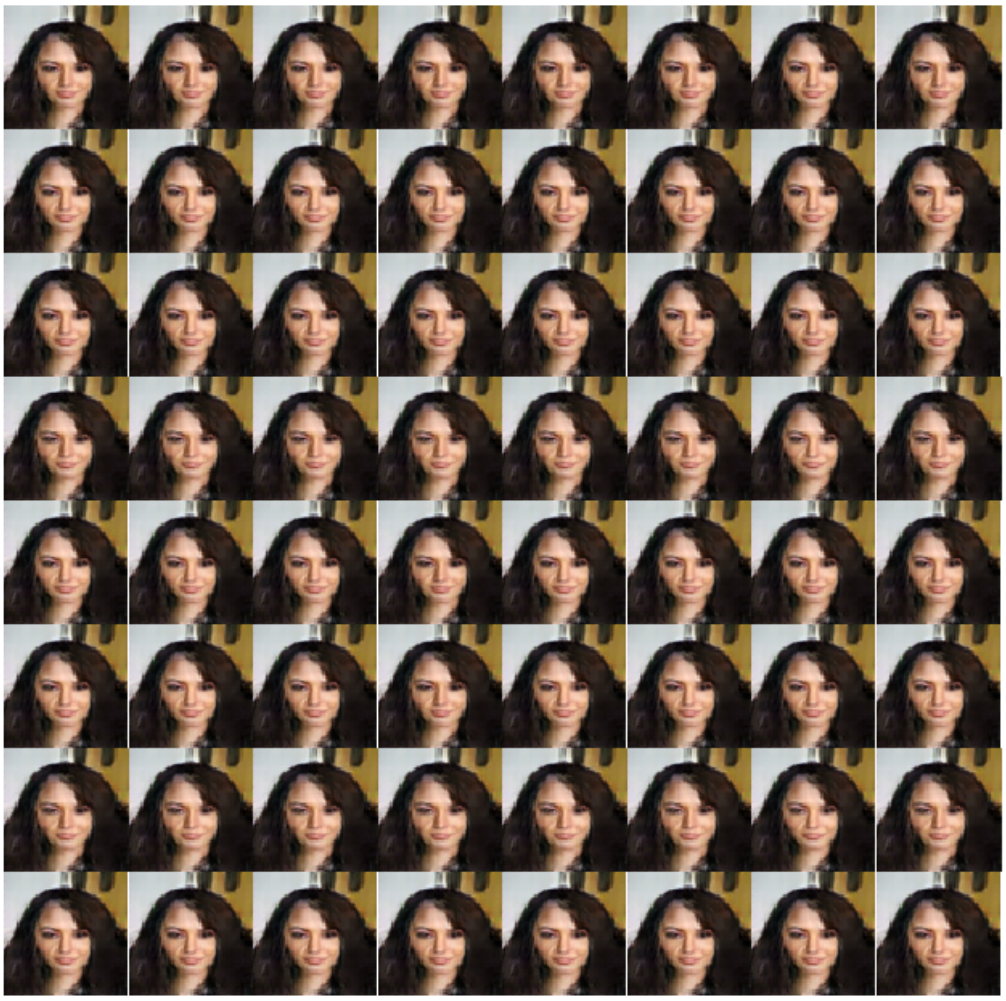
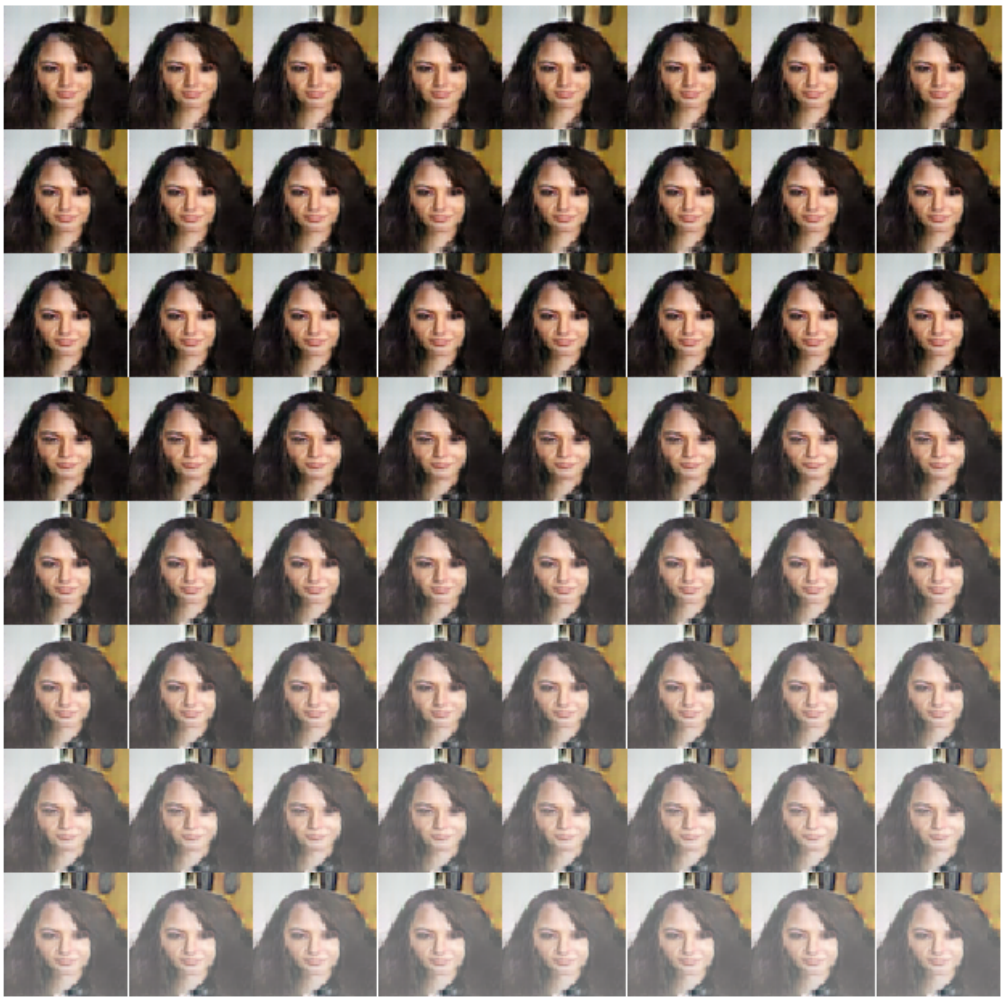


Figure 8: This image shows the effect of moving in x space from x_0 to x_1. x_0 and x_1 correspond to the z_0 and z_1 discussed earlier. The image shows that moving in x space does not produce any new image. α changes from 0 to 1 in 64 steps.

We did another experiment to magnify the effect of moving in x space. We keep the variables as before and vary α from 0 to 10 in 64 steps.



Epoch 0

Figure 9: This image shows the effect of moving in x space from x_0 to x_1. x_0 and x_1 correspond to the z_0 and z_1 discussed earlier. The image shows that moving in x space does not produce any new image. α changes from 0 to 10 in 64 steps.

Magnifying the coefficient α still does not produce any new images and it only brightens the image.

3 Quantitative Evaluations

Salimans et al. introduce the Inception score metric which we compute for both models [8]. The inception score produced by the original model on a set of 64 generated images was 2.7501891. One advantage of this metric is improved precision over human qualitative evaluation. It does this by applying the Inception model to all generated images, producing a scalar value. A good Inception score is produced when images have a conditional label distribution with low entropy. On the other hand, judging the quality of a generative model with an Inception score can be misleading, when a different training set from the original Inception model is used [7].

A second metric was used for comparison, the "Maximum mean discrepency" (MMD) [3] statistic. The motivation of this metric is the more similar the statistical moments are, then the more likely samples are to come from the same distribution. MMD matches all statistical moments between a dataset and samples from the model with the kernel trick. However its usage of kernels also makes MMD a computationally expensive metric by comparison. Thus it is often applied within mini-batches to reduce the runtime overhead. In our implementation, we use the squared linear MMD statistic from Gretton et al. (6.1) [2].

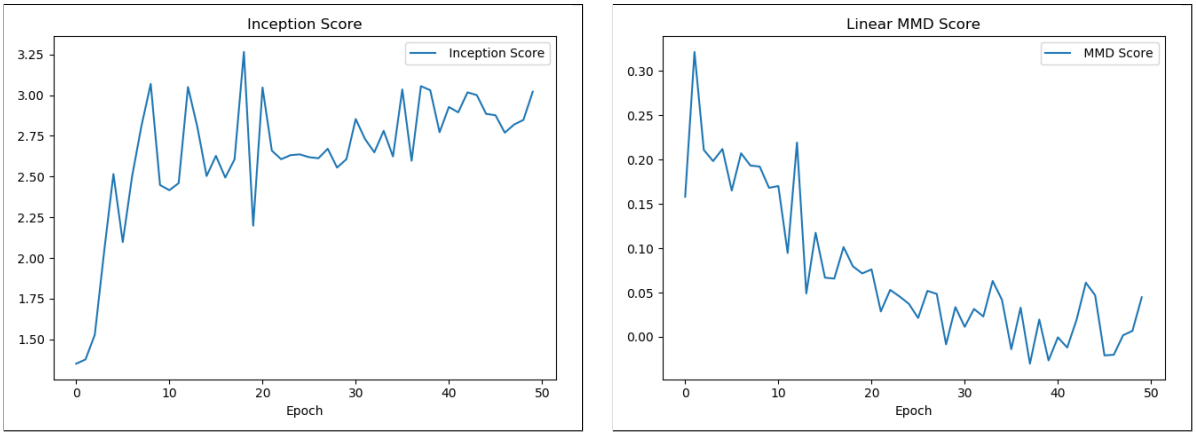


Table 3: Inception Score (higher is better) and the MMD Statistic (lower is better) for generated images after each epoch of training. We evaluated both on a batch size of 1000 and observe that the Inception Score has slightly higher variance that linear MMD. We note that both metrics improve during, suggesting that these scores track subjective image quality over the course of the experiment. Whether this correlation is maintained until convergence, or if it diverges from subjective quality after 50 epochs is a topic for further research.

All source code for the assignment can be found on Github[2].

[2] https://github.com/isaacsultan/ift6135-gan/

References

[1] Ian J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017.

[2] A. Gretton, K. Borgwardt, M. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, March 2012.

[3] Yujia Li, Kevin Swersky, and Richard S. Zemel. Generative moment matching networks. *CoRR*, abs/1502.02761, 2015.

[4] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, and Zhen Wang. Multi-class generative adversarial networks with the L2 loss function. *CoRR*, abs/1611.04076, 2016.

[5] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.

[6] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

[7] Mihaela Rosca, Balaji Lakshminarayanan, David Warde-Farley, and Shakir Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

[8] Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.