Neural Networks for Images (67103): Exercise 3

Generative Adversarial Networks (GANs)

By: Amit Halbreich, ID: 208917393

Theoretical Questions – Answers

1. The formula:

$$P(x) = \left|\frac{\nabla \det(M^{-1}(x)}{\nabla x}\right| \cdot P_z(M^{-1}(x))$$

is discussed in the context of $GLOW$ architecture. In $GLOW$, the mapping M is constructed by composing a series of easily invertible transformations $g_k \circ g_{k-1} \circ \ldots\ldots \circ g_1$.

Each transformation consists of three steps: activation normalization, invertible 1x1 convolution, and affine coupling layer. To compute $P(x)$ in $GLOW$, we denote z as a random variable following a standard Gaussian distribution $p(z)$, and x as a random variable following $p(x)$. The invertible transformation f relates x and z,

with $x = f(z)$ and $z = f^{-1}(x)$.

By the change of variables formula, $P(x)$ is related to $p(z)$ through the equation:

$$p(x) = p(z) \cdot \left|det(J_f(z))\right|$$

where J is the Jacobian matrix of f at z. Taking the logarithm of both sides,

$$log(p(x)) = log(p(z)) + log(\left|det(J_f(z))\right|).$$

Since f is composed of multiple transformations, the Jacobian determinant becomes the product of the determinants of the individual Jacobian matrices. Hence:

$$log\big(p(x)\big) = log\big(p(z)\big) + \sum_{i=1}^{k} log(\left|det(\frac{dg_i}{dg_{i-1}})\right|)$$

Now, let's discuss why $P(x)$ cannot be computed in $GANs$ and $GLOs$:

In $GANs$, the Generator network G is trained to transform a latent code z into data samples x. However, this transformation cannot be explicitly inverted, meaning that given x, the corresponding z cannot be easily determined. Additionally, the Discriminator network D in $GANs$ aims to distinguish between real and generated data but does not estimate the likelihood of samples. Therefore, the computation of $P(x)$ is not possible in $GANs$.

Similarly, in $GLOs$, the process of transforming z to x is not invertible. This means that given x, it is not straightforward to obtain the corresponding latent code z. Moreover, the latent code z in $GLOs$ does not follow a fixed distribution. As a result, estimating $P(x)$ is not feasible in $GLOs$.

In summary, both $GANs$ and $GLOs$ focus on learning the transformation from latent space to data space without explicitly computing the density $P(x)$.

2. In the early stages of GAN training, G is still producing poor quality images, and D can easily differentiate between these generated images and real images. This means that D's

gradients with respect to G will be very large, which can lead to the gradients of G becoming very small, which in turn can make it difficult for G to learn to generate realistic images.

To see this analytically, let's consider the objective function of the GAN, which can be written as:

$$V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data(x)}}[log\ D(x)] + \mathbb{E}_{\boldsymbol{z}\sim p_z(z)}[log\ (1 - D(G(z)))]$$

Where $\boldsymbol{x}\sim p_{data(x)}$ is the distribution of the real images and $\boldsymbol{z}\sim p_z(z)$ is the prior distribution of the noise vector z.

We can compute the gradients of $V(D,G)$ with respect to G as follows:

$$\nabla_G V(D,G) = \nabla_G \mathbb{E}_{\boldsymbol{z}\sim p_z(z)}[log\ (1 - D(G(z)))]$$

By rearranging the terms, by applying Linearity of Expectation law, we have:

$$\nabla_G V(D,G) = \mathbb{E}_{\boldsymbol{z}\sim p_z(z)}[\nabla_G[log\ \big(1 - D\big(G(z)\big)\big)]]$$

Using the chain rule, we can write:

$$\nabla_G V(D,G) = \frac{\nabla_{D(G(z))}\ \big[log\ \big(1 - D\big(G(z)\big)\big)\big]}{\nabla_G D(G(z))}$$

Now, let's consider what happens to the gradients of $log\ \big(1 - D\big(G(z)\big)\big)$ with respect to $D\big(G(z)\big)$. As $D\big(G(z)\big)$ approaches 1 (which is what happens as D becomes a decisive classifier). We can write:

$$\frac{\partial[log\ (1 - D\big(G(z)\big)]}{\partial[D\big(G(z)\big)]} = -\frac{1}{1 - D(G(z))}$$

As $D\big(G(z)\big)$ approaches 1, this gradient becomes very large in magnitude. Now, let's consider what happens to the gradients of $D\big(G(z)\big)$ with respect to G as $D\big(G(z)\big)$ approaches 1. We can write:

$$\frac{\partial[D\big(G(z)\big)]}{\partial[G(z)]} = \frac{\nabla_x D(x)|_{x=G(z)}}{\nabla_G G(z)}$$

As D becomes a decisive classifier, $\nabla_x\ D(x)|_{x=G(z)}$ approaches 0, since the discriminator will be almost certain that the generated image is fake. This means that the gradients of $D(G(z))$ with respect to G will be very small in magnitude.

Now, let's consider the scenario where D has converged quickly to a decisive classifier with logit values close to 0 and 1. This means that D's output before the sigmoid activation function approaches small values for real images and large values for fake images.

When the output of D is close to 0 or 1, the gradient of the logarithm term, $\nabla_G\ log(1 - D(G(z)))$, becomes small. This is because the logarithm function has a small derivative when its argument is close to 0 or 1. As a result, the gradients of D with respect to G - $\nabla_G\ V(G)$, also become small.

In this scenario, the small gradients indicate that D's output is not highly sensitive to changes in G's parameters. The lack of meaningful gradients hinders G's learning process,

making it difficult for G to receive informative feedback on how to improve its generated images. As a result, G's convergence is slower, and it may struggle to generate high-quality images.

To summarize, in the scenario where D converges quickly to a decisive classifier with logit values close to 0 and 1, the gradients of D with respect to G become small. This lack of meaningful gradients hampers G's learning process and can lead to slower convergence and challenges in generating high-quality realistic images.

Putting these results together, we see that as D becomes a decisive classifier, the gradients of $V(D,G)$ with respect to G will become very small, which can make it difficult for G to learn to generate better realistic images. This is known as the "$Vanishing\ Gradients$" problem in $GANs$.

3. (A) The inner problem $\min_G V(D,G)$ aims to find the best generator G (Optimize G such that $V(D,G)$ is minimized – G's loss is minimized while D's parameters remains fixed and G's parameters explicitly are updated) that can fool the discriminator D into believing that its generated samples are real, while the discriminator tries to correctly classify the real and fake samples. The main idea of the inner problem is search for the best Generator given the present Discriminator in the specific epoch. The best Generator is considered to be the Generator that generates the most realistic and accurate fake images. This is done by minimizing the binary cross-entropy loss between the discriminator's output and the ground truth labels.

3. (B) Once the inner problem is converged, the outer problem $\max_D[\min_G V(D,G)]$ aims to find the best discriminator D that can distinguish the real samples from the fake ones generated by G. We strive to optimize D such that G will minimize the value $V(D,G)$ – minimize both of the losses of G & D while G is also being optimized in the process. The main goal of the outer problem is causing D to reach as many correct distinguishes between fake/real images ground truth labels and achieving equilibrium of the architecture in terms of loss of G & D and network stability. It is done by maximizing the binary cross-entropy loss between the discriminator's output and the ground truth labels. Solving the outer problem involves finding the optimal values for the discriminator's parameters given the fixed generator G from the inner problem. This can be a challenging optimization problem, especially for a limited capacity neural network.

3. (C) The main drawback of the inverted formulation $\max_{\mathrm{D}}[\min_G V(D,G)]$ is that it assumes that the discriminator can achieve the exact optimal solution for its inner problem, which may not always be possible in practice. In some cases, the discriminator may get stuck in a local minimum, leading to a suboptimal solution for the entire GAN. This can be particularly problematic if the generator is over-optimizing for this suboptimal discriminator, in this case we will not reach equilibrium point in which G generates realistic images of the given dataset and D won't classify fake and real images correctly. This will result in poor sample quality and instability in the training process.

PRACTICAL QUESTIONS – ANSWERS

Building the DCGAN architecture and making it work as expected was very challenging process. I built an architecture for both MNIST and CelebA64 datasets and in this exercise, I will relay to results received on both datasets – I had an exploration process on both datasets and tried to achieve good results in both.

1. (I) I wrote a function in which you can choose which dataset to use and what kind of mode to use – "Non-Saturating", "Saturating" and "L2". For each iteration of the training loop**, I trained the Generator and the Discriminator in relation of 1:3** e.g. I trained the Discriminator three times more than I trained the Generator. I used the following parameters:

$$Learning\ Rate = 0.0002, Batch\ Size = 128, Latent\ Vector\ Size = 100$$

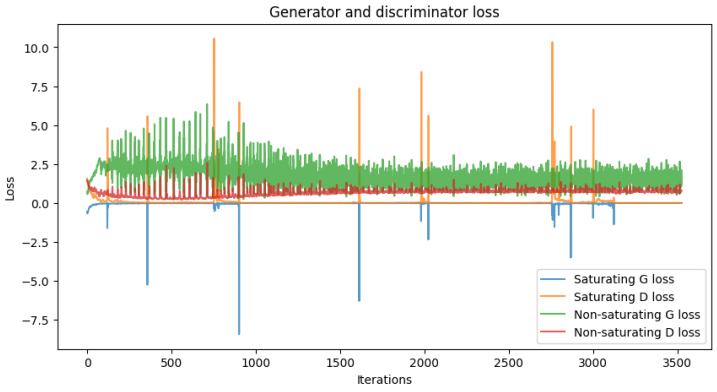
$$Num\ Epochs = 15, Feature\ Map\ Capacity\ (Both\ G\ \&\ D) = 64\ [Layers]$$

The results I got for each one of the modes are presented below (Right: 1:3, Left: 1:1 Ratio).

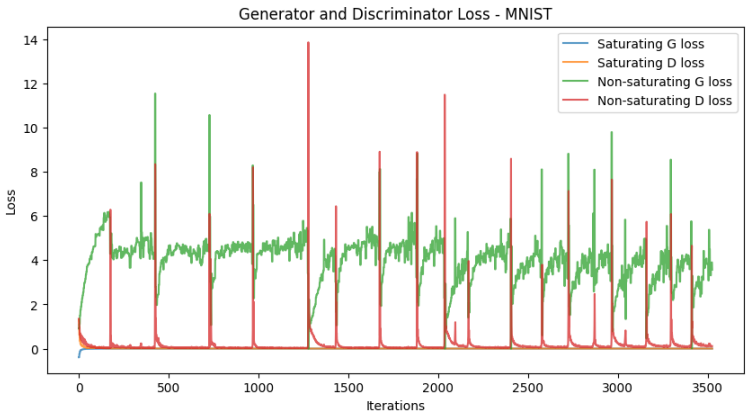
RESULTS ON MNIST DATASET:

LOSS GRAPHS

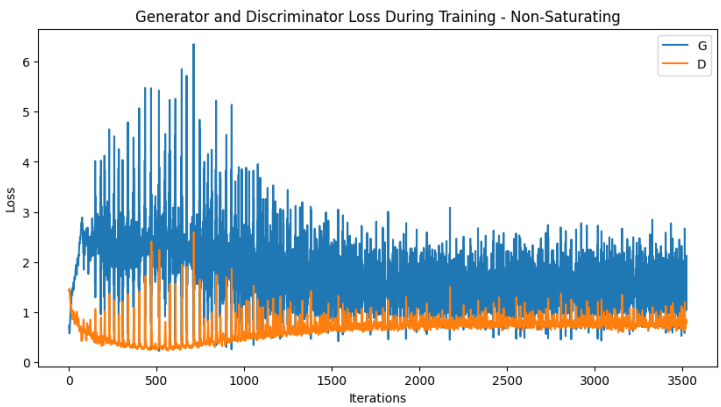
STEP RATIO 1:1 – G:D

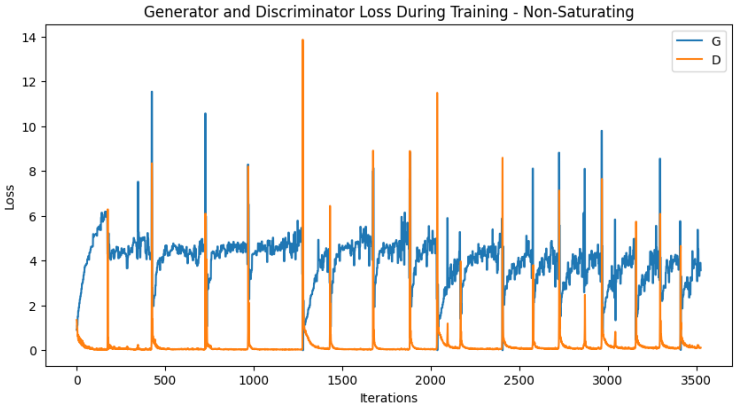


STEP RATIO 1:3 – G:D

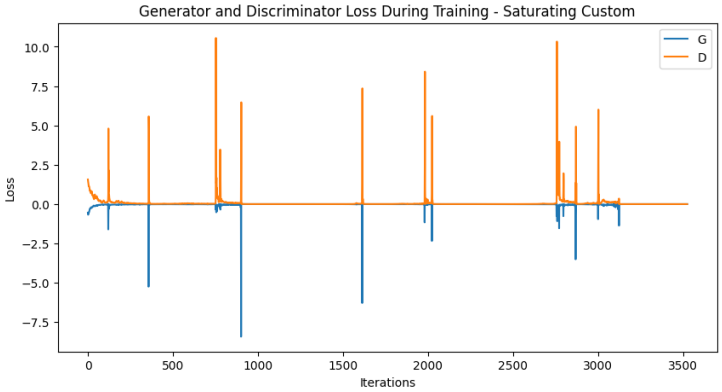


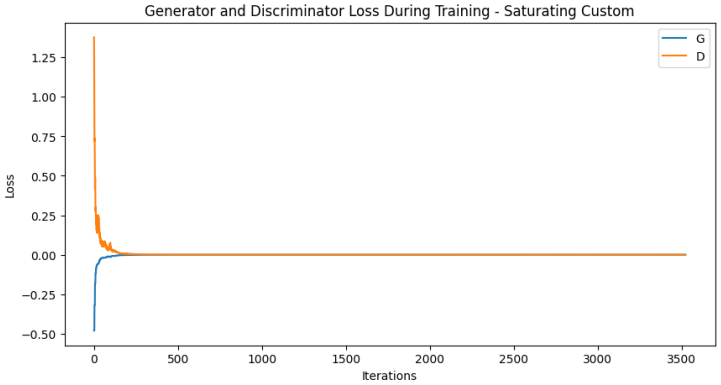
NON – SATURATING MODEL



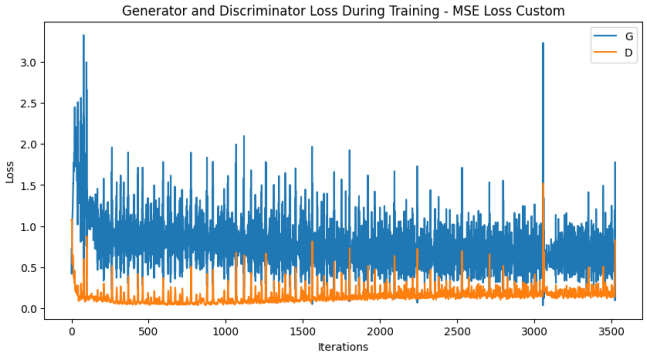


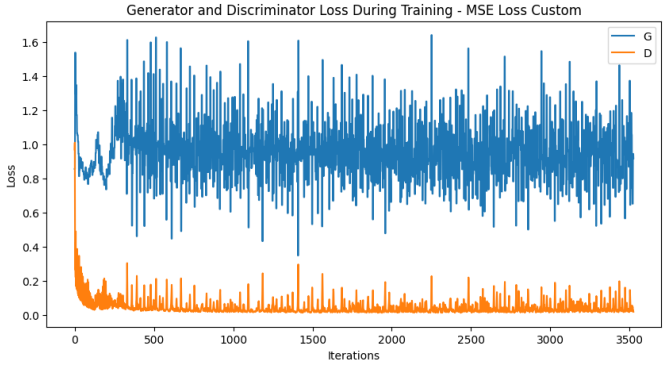
SATURATING MODEL





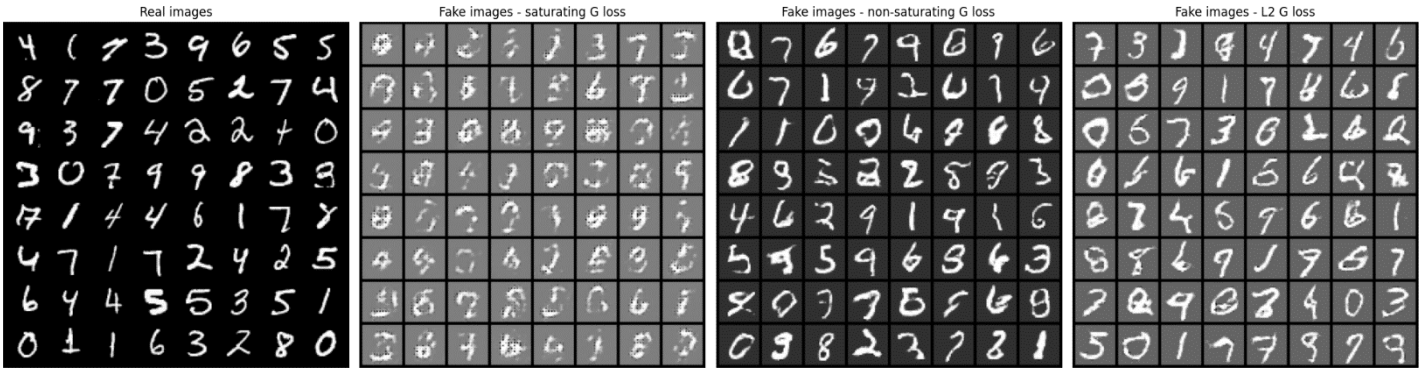
MSE LOSS – ||L2|| NORM



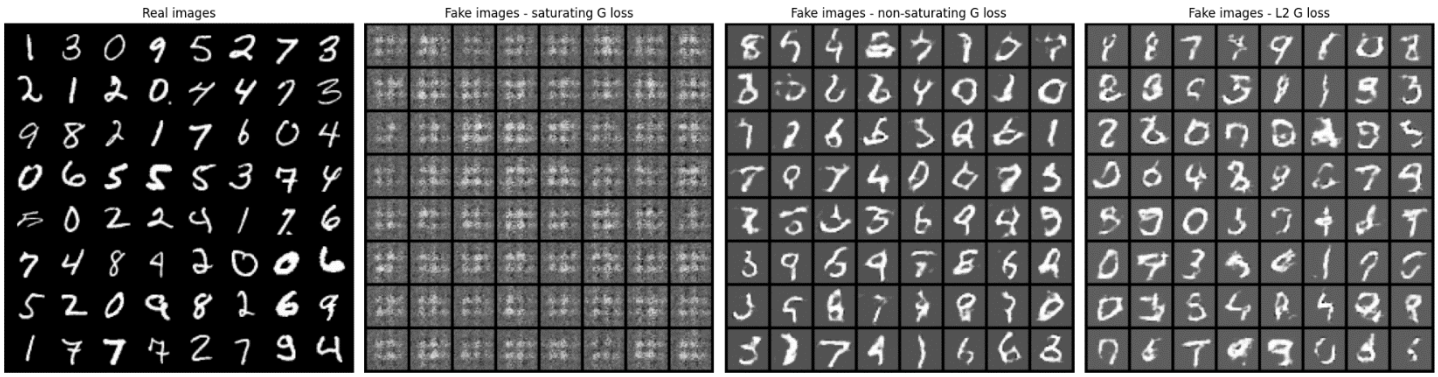


FINAL IMAGES RESULTS - MNIST DATASET

1:1 RATIO OF GENERATOR : DISCRIMINATOR TRAINING

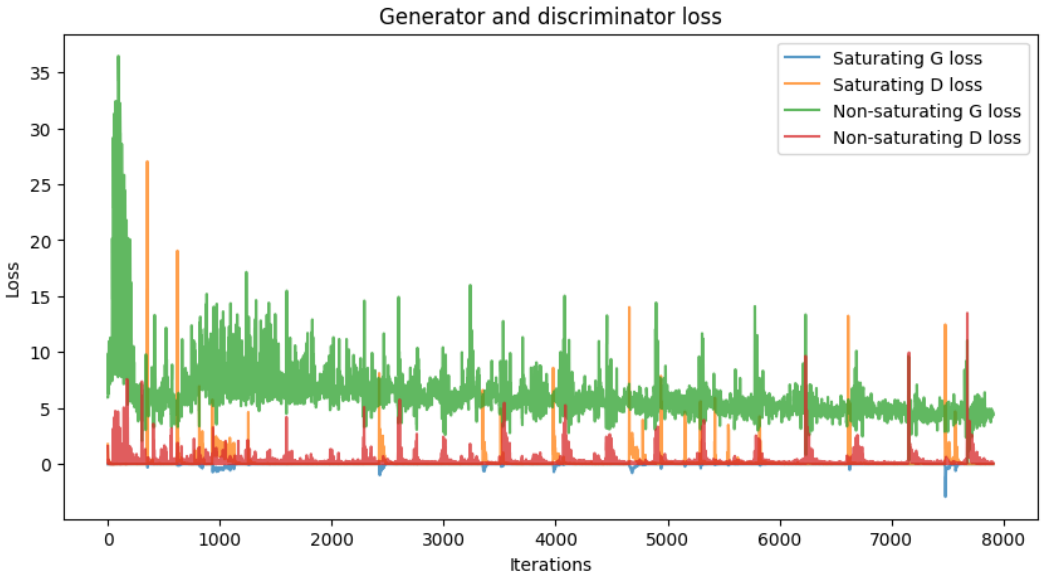


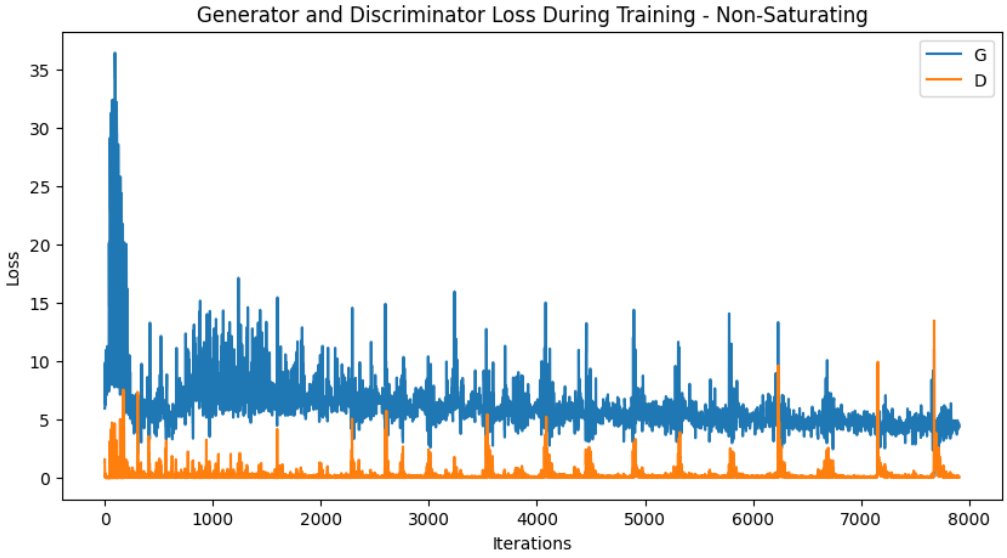
1:3 RATIO OF GENERATOR : DISCRIMINATOR TRAINING

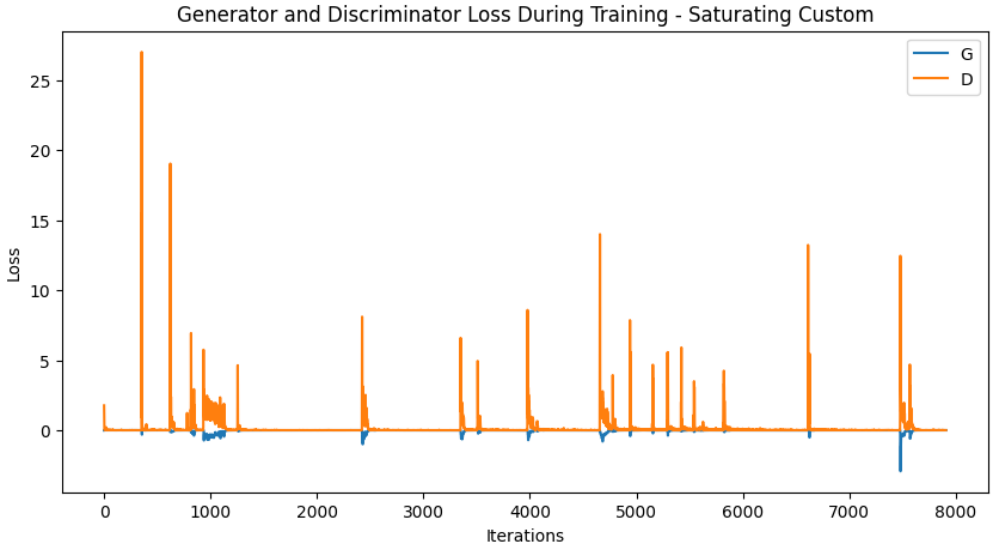


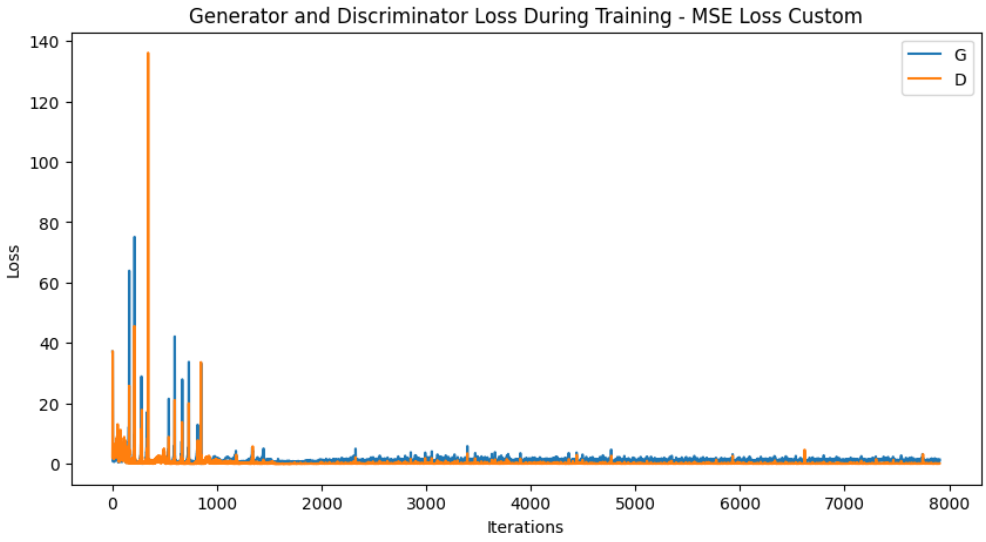
RESULTS ON CELEBA64 DATASET:

GRAPHS – 1:3 G:D STEP RATIO (ONLY)

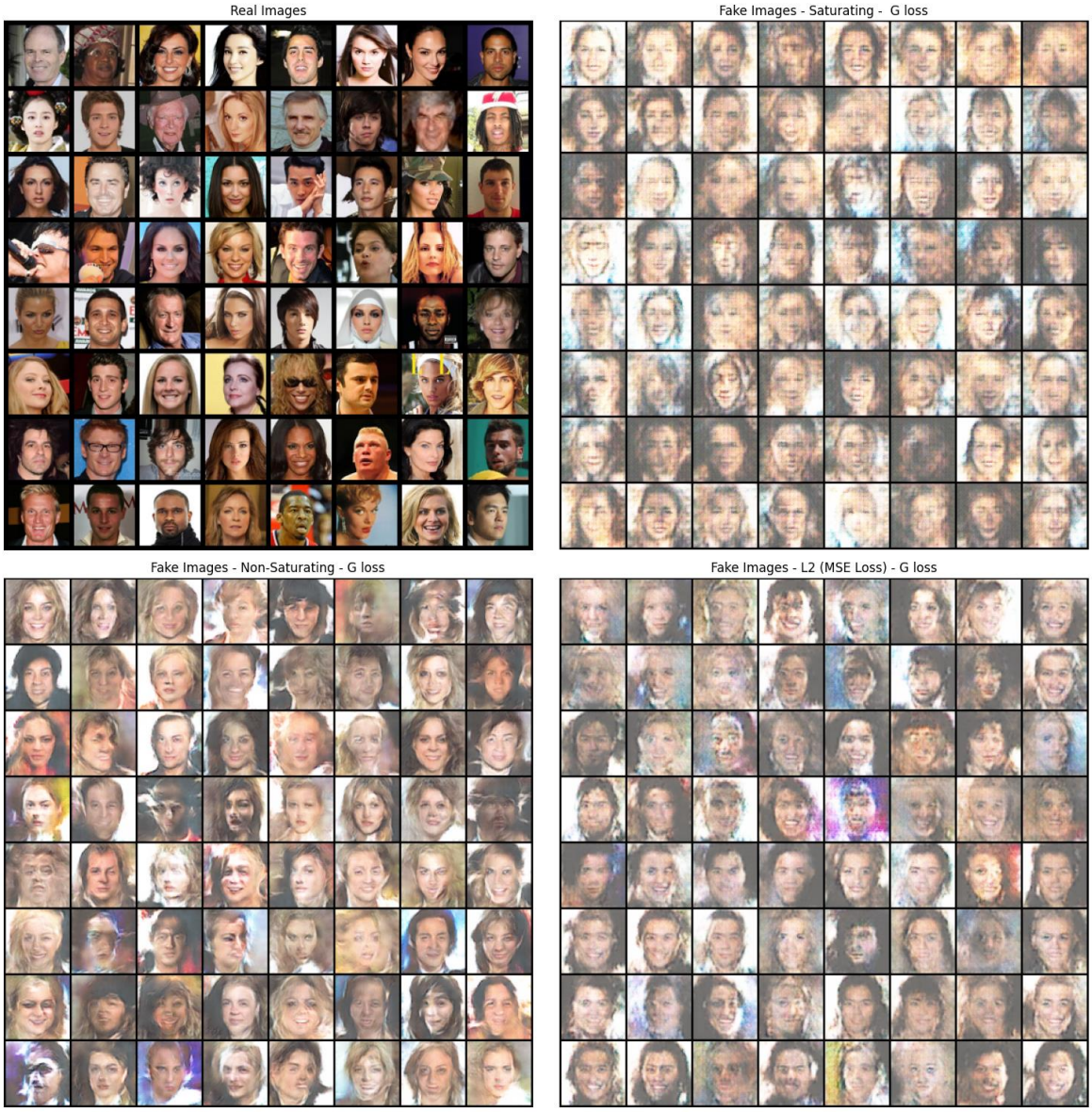








FINAL IMAGES RESULTS – CELEBA64 DATASET



As we can spot, we got the best results when using non-saturating state network and criterion. "L2" state also produced very good results in comparison to the unclear and distorted images that the "Saturating" state network produced.

In the Saturating state graph, we can spot that the DCGAN reached saturation. The Discriminator's Loss values dropped to 0 almost immediately after the training was started and as a result G failed to generate good images of the Digits/Celeb Faces on both datasets.

Although G's loss got to almost zero, it started from a negative number loss and reached to 0 so G's loss isn't actually low for the "Saturating" state – as we can see we got a blurry, unclear and distorted images that doesn't look like actual Celeb Faces and the digits for MNIST are sometimes distorted and does not resemble a real digit. This is also called the "Vanishing Gradient" problem.

When we used "Non-Saturating" and "L2" states for the network we reached good results and the "Vanishing Gradient" problem was solved – the DCGAN architecture did not reach saturating point, G and D were able to train and fit themselves for each other – we can see that by looking at the graphs of MSE L2 Loss" and "Non-Saturating" states in which there are regional jumps in the losses of G and D and this is because each time the Generator / Discriminator improved and optimized itself and decreased its Loss value, the corresponding other would result in a higher loss value and as a result strived to be improved and optimized as well and vice-versa.

2. DCGAN INVERSION: Denote an image I as image from the dataset $MNIST\ /\ CelebA64$, and a pretrained Generator G we wish to find the latent vector z that satisfies the equation:

$$G(z) = I$$

Taking the pretrained Generator G we used to in the training of Question 1 in a "Non-Saturating" state that provided us with the best results out of all the states. I generated a random Latent Vector z and a random image I from the current batch. I initialized a new optimizer for z and gave it only the Latent Vector z with its random values. Because it is a reconstruction task I used MSE Loss function as criterion.

Parameters for $z - Latent\ Vector$ estimation:

$$Learning\ Rate = 0.05[MNIST]/\ 0.005[CelebA64], Latent\ Vector\ Size = 100$$

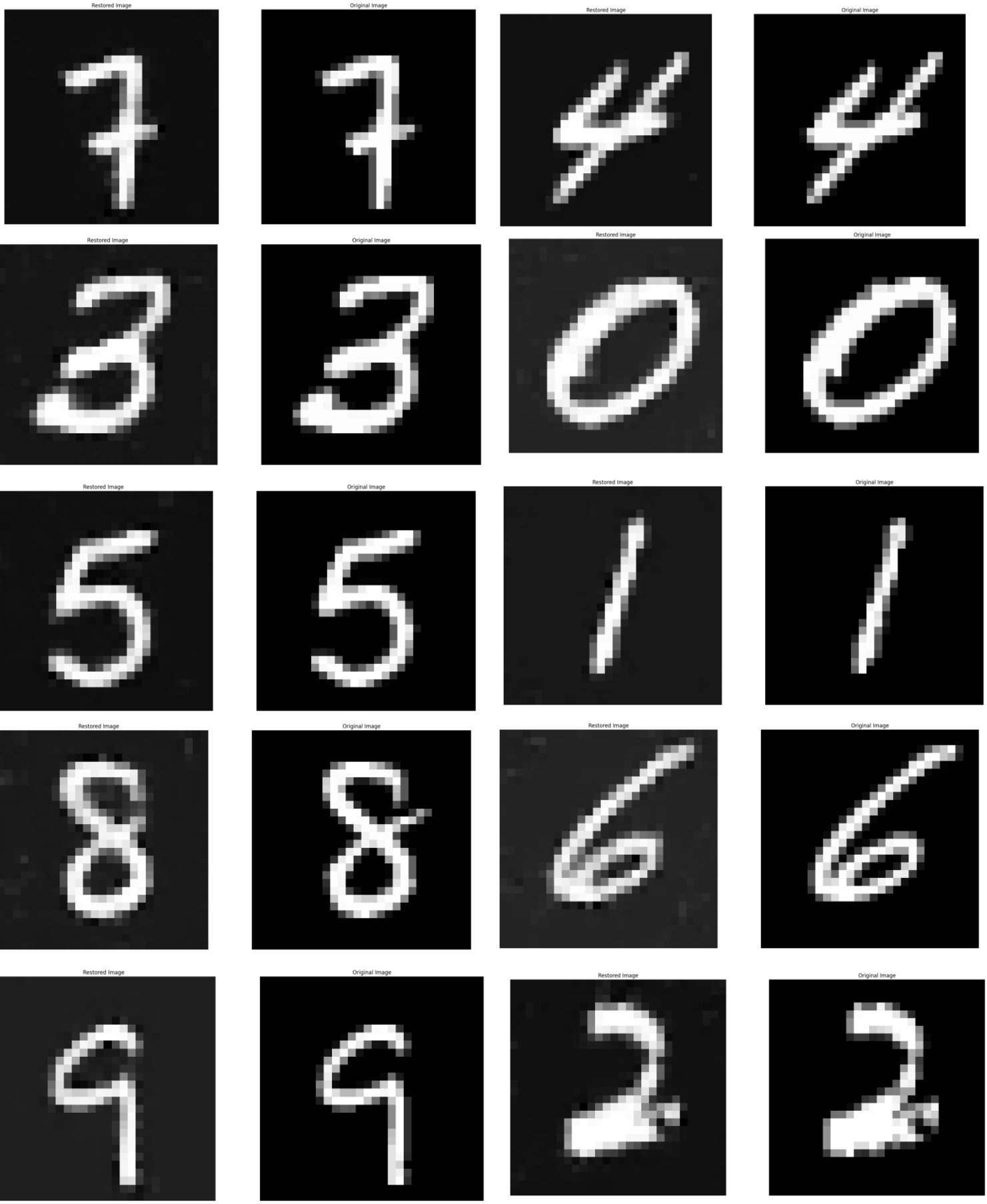
$$Num\ Iterations = 5{,}000, Optimizer\ Type = Adam, Criterion = MSE\ Loss$$

Explanation about the process of finding the closest latent vector z that will perform the best image restoration: We start with a randomly generated z latent vector and give the optimizer only the vector itself with the its random values, then we use the pretrained Generator G "Non-Saturating" Model to generate an image such that:

$$Generated\ Image = G(z)$$

In each iteration we compute the $MSE\ Loss$ and optimize the latent vector z using back propagation. Finally we return the Latent Vector which minimizes the $MSE\ Loss$ and the generated image after restoration applying the Generator on the z that got minimal MSE Loss such that $I \cong Generated\ Image = G(z)$.

FINAL IMAGES RESULTS – MNIST DATASET



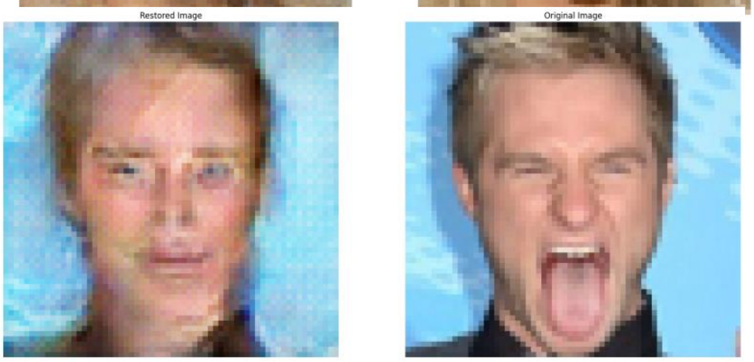
FINAL IMAGES RESULTS – CELEBA64 DATASET

Figure 1

Figure 2

Figure 3

Figure 4



2. RESULTS ANALYSIS:

As we can spot the DCGAN Inversion process on the MNIST dataset was a lot more efficient in the z Latent Vector loss terms and as a result we got better result when trying to reconstruct digits. On CelebA64 dataset we managed to reconstruct the faces images partially only, the loss values that we got at the end of the restoration process are higher and we can see that reconstructing actual human faces is much more complex and harder task to accomplish. This is due to the fact that human faces features are a lot more versatile and include complex shapes, patterns and textures that the Generator finds hard to restore when he generates an image. We have also seen in Question 1 that the fake images that the Generator produces even in "Non-Saturating" state seems a bit odd, distorted and one may recognize that the generated fake images are indeed fabricated and doesn't include real human face.

In contrary to human faces, MNIST handwritten digits images are more likely to be reconstructed successfully because the digits' images contain only limited set of colors/patterns and mainly consists of straight lines or curves, round/oval shapes – all of these factors lead to better results and good reconstruction (it's almost unrecognizable to the naked eye that the generated image is not a real image, but a fake/fabricated image).

I used $Learning\ Rate = 0.05$ for $MNIST\ dataset$ and $Learning\ Rate = 0.005$ for $CelebA64\ dataset$ – because these learning rates for $z\ Optimizer$ produced the best results after a lot of tries.

About CelebA64 human faces reconstruction - the main facial features that the Generator reconstructs successfully are:

- Face shape and borders/lines.
- Hair, skin, lips and eye colors.
- Background color.
- Smiles.

The facial features that the Generator reconstructs not so good:

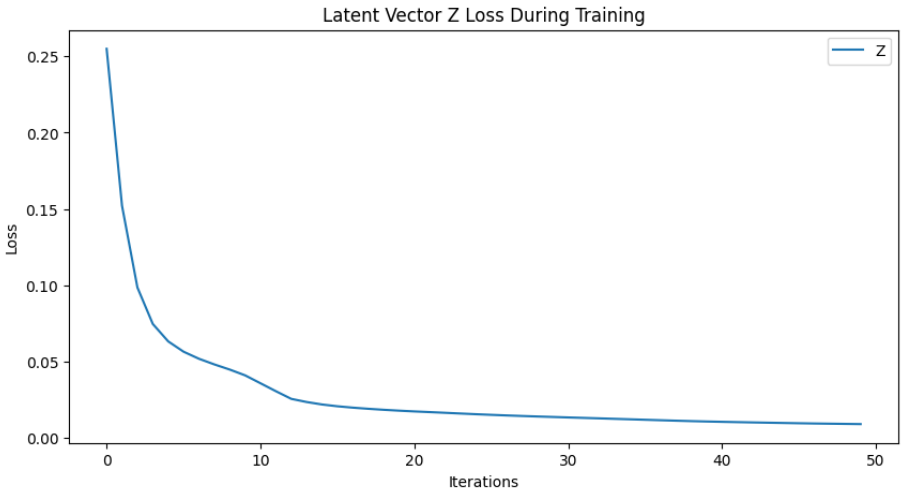
- Unique Facial expressions – for instance doesn't recognize the enthusiastic guy with his tongue out.
- Accessories: such as earrings, glasses hats etc..
- Details of facial features such as nose shape, ears etc..
- Generally, the reconstruction is very blurry and unclear, and the generated image doesn't resemble the human in the original image overwhelmingly.

CONCLUSIONS:

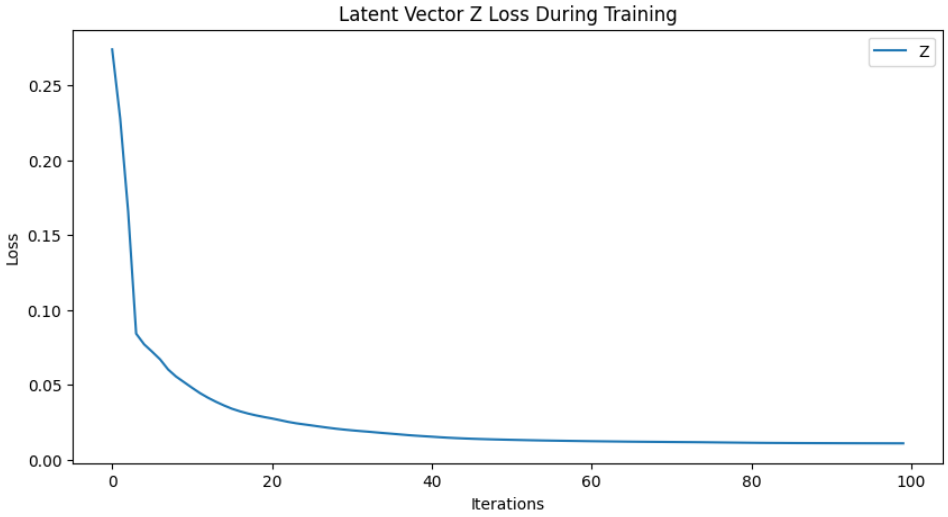
- We may need a deeper or more sophisticated $DCGAN\ Architecture$ in order to reconstruct good images of CelebA or a different architecture such as Style GAN2 or other architecture.
- It is much easier to reconstructing digits than reconstructing human faces using DCGAN Architecture.

Z – Latent Vector Loss Graphs:

For CelebA Dataset (Images in Figure 1)



For MNIST Dataset (Digit #2)



As we can see, we got very low loss values for both datasets, but the actual reconstructed images in the $GAN\ Inversion$ process results are better on digits rather than human faces.

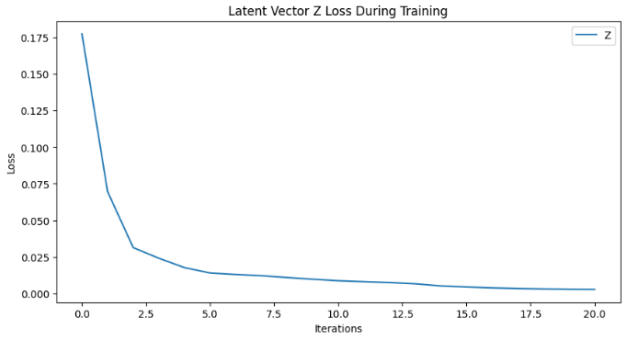
3. A. For this section, consider a noisy image I with normal distribution noise with $m = 0\ and\ std = 0.1$, denote the noise matrix N by - $N \sim N(0,0.1)$. Because it is a reconstruction task, I used $MSE\ Loss$ criterion $(L2\ Norm)\ and\ L1\ Norm$ (sometimes with $L1\ Norm$ The result were better with less noise). We would like to solve the equation for z^*:

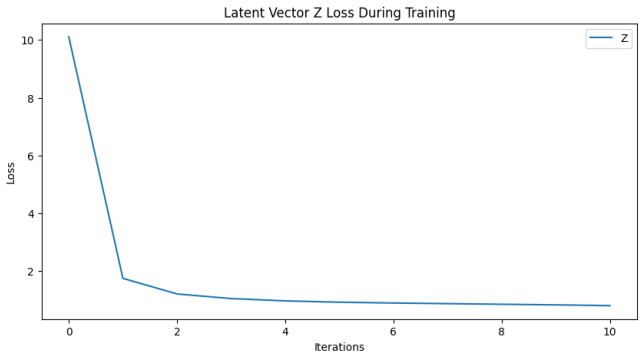
$$L2\colon z^* = argmin_z \|[G(z) + N] - I\|_2{}^2\ or\ \ L1\colon\ z^* = argmin_z \|[G(z) + N] - I\|_1$$

Our goal is to find the latent vector z that if we will activate the Generator G on it and add the noise to the generated image, subtract the original noisy image from it and calculate the $L2$ norm of it, and minimize the described loss by using an optimizer on the latent vector weights, we will receive the best Latent Vector z^* which is considered to be the original latent vector that will reconstruct the original image without the noise – perform image denoising.

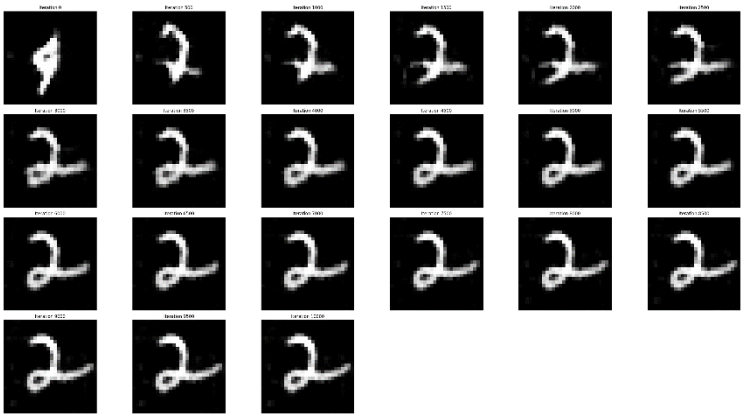
I took the original "Non-Saturating" state Generator G that I used in Question 1.

L2 NORM (MSE) RESULTS – MNIST DATASET L1 NORM

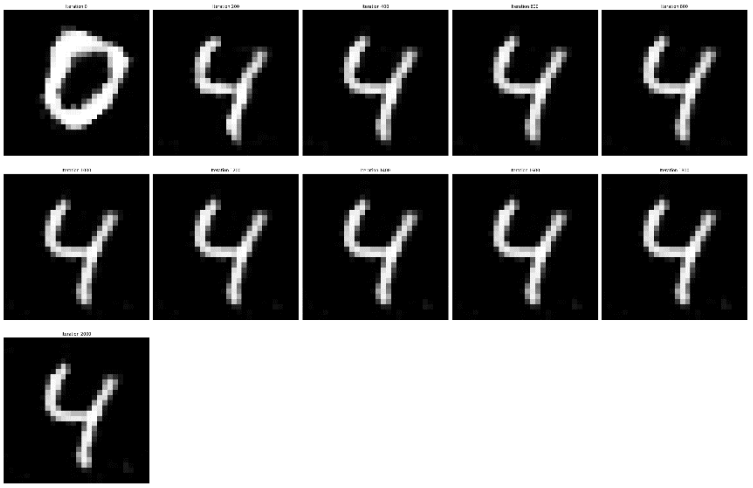




PROGRESS OVER ITERATIONS:



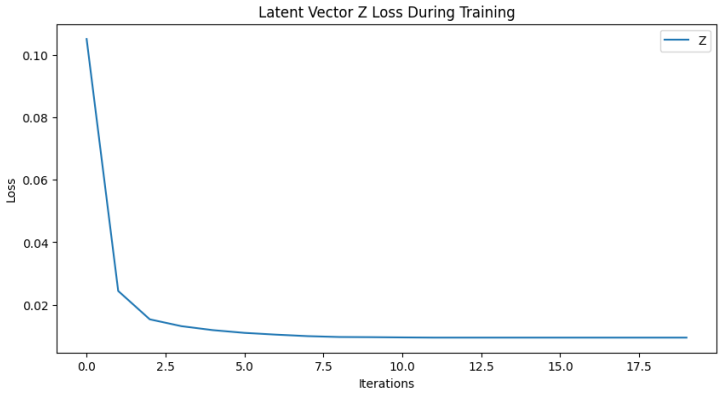
PROGRESS OVER ITERATIONS:

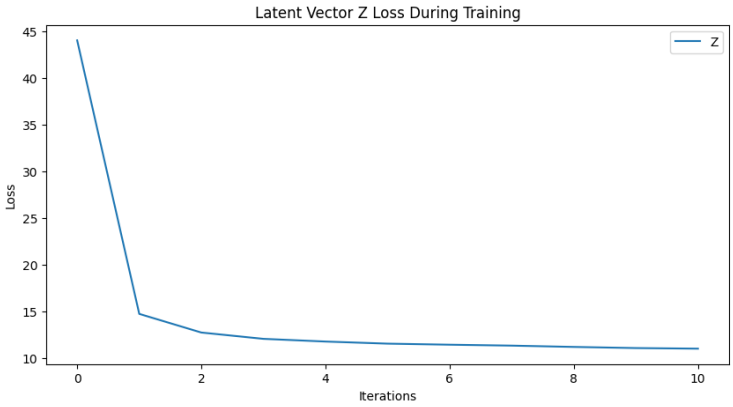


L2 NORM (MSE)

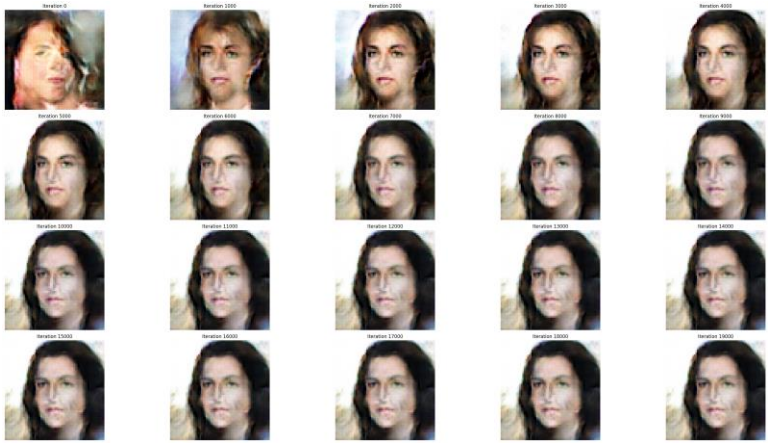
RESULTS – CELEBA DATASET

L1 NORM

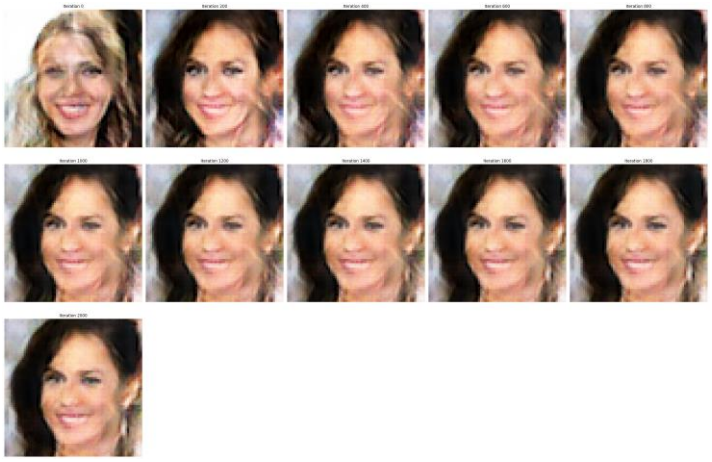




PROGRESS OVER ITERATIONS:



PROGRESS OVER ITERATIONS:



As we can spot reconstruction of digits is an easier task than reconstruction of human faces and this is due to the reasons explained above in Question 2. If we also add noise to the original reconstruction of human faces becomes even harder task because some of the delicate details in the original images are changed and the Generator will find it hard to reconstruct the original image. Despite that, we got images that somewhat resemble the human in the original image. Reconstruction of digits images with normal distribution noise produced better denoising results and the final denoised images are better in comparison to the final denoised images of human faces. Almost similar to the results we got on MNIST dataset back in $Question\ 2$ regarding $DCGAN\ Inversion$.

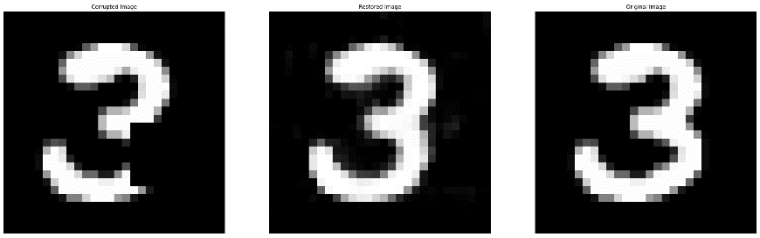
3. B. For this section, consider an inpainting image I (With missing 8x8 random pixels window in the middle of the image). Given a binary mask M where the mask value is 1 where I is not corrupted and 0 where it is corrupted (missing the original pixel) We would like to solve the equation for z^*:

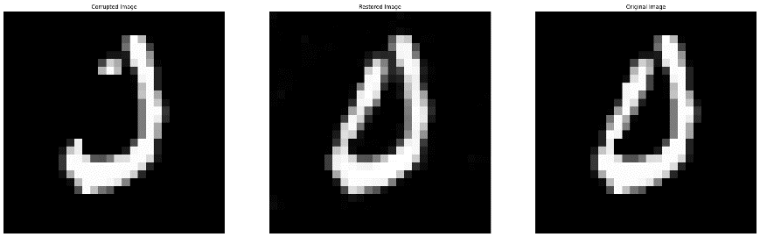
$$z^* = argmin_z \|M \cdot G(z) - I\|_2{}^2$$

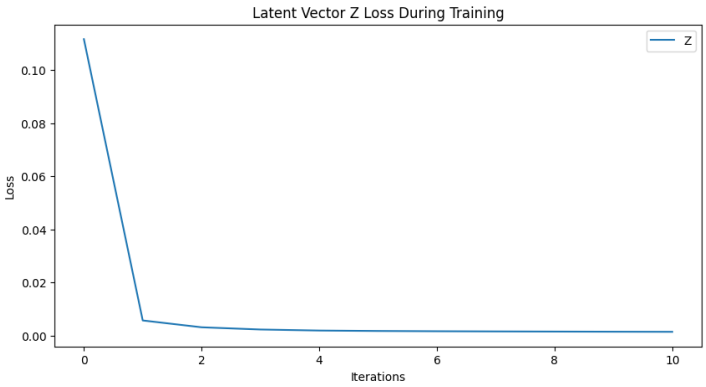
Because it is a reconstruction task, I used $MSE\ Loss$ criterion $(L2\ Norm)$ or $L1\ Norm$ and often the $L1\ Norm$ produced better images for inpainting (mainly in $MNIST$ dataset).

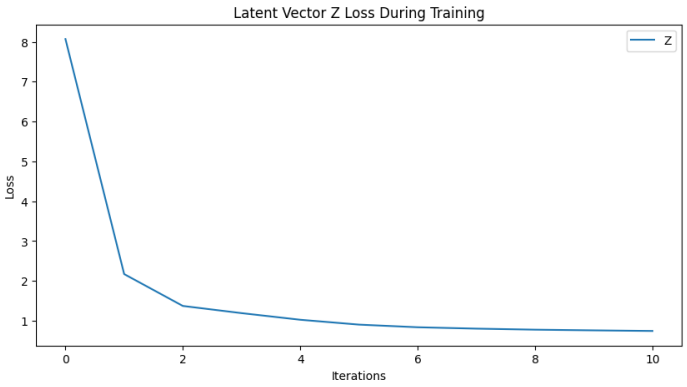
Our goal is to find the latent vector z that if we will activate the Generator G subtract the original inpainting image from it, multiply the result by the mask M, calculate the $L2$ norm of it, and minimize the described loss by using an optimizer on the latent vector weights, we will receive the best Latent Vector z^* which is considered to be the original latent vector that will reconstruct the original image without the missing random 8x8 pixels window – perform image inpainting.

L2 NORM (MSE) RESULTS – MNIST DATASET L1 NORM

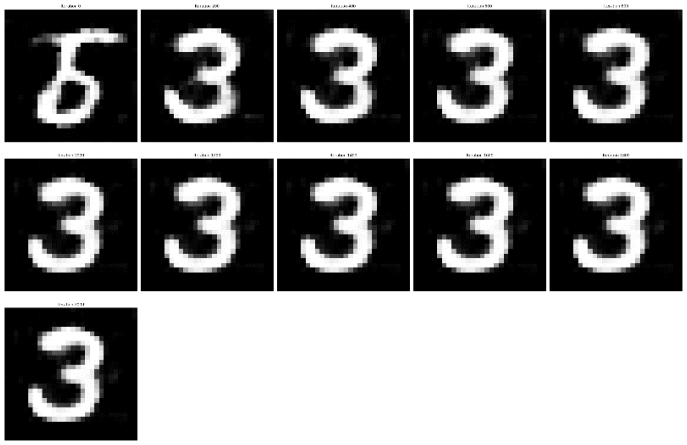




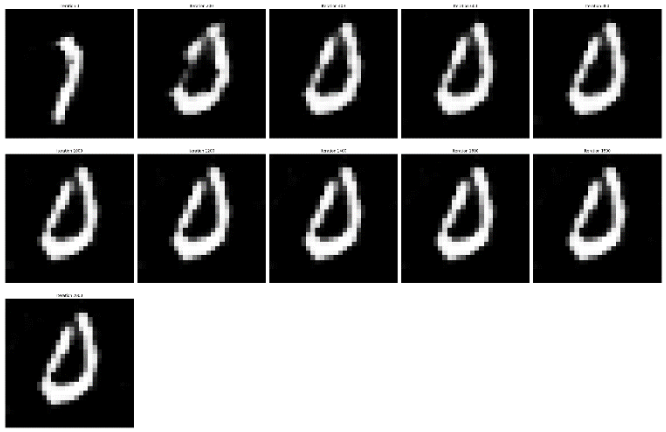




PROGRESS OVER ITERATIONS:



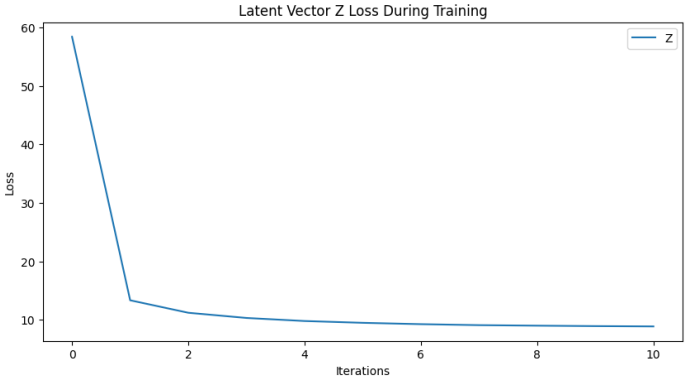
PROGRESS OVER ITERATIONS:

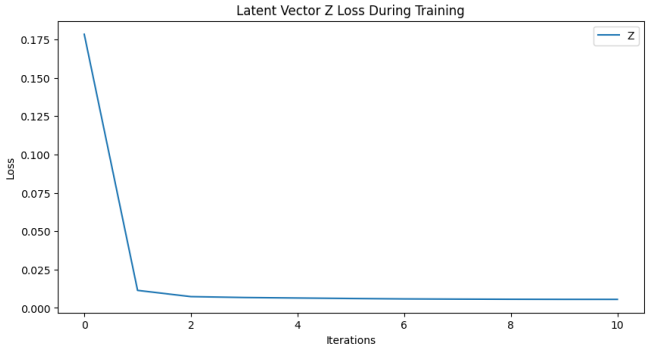


L1 NORM

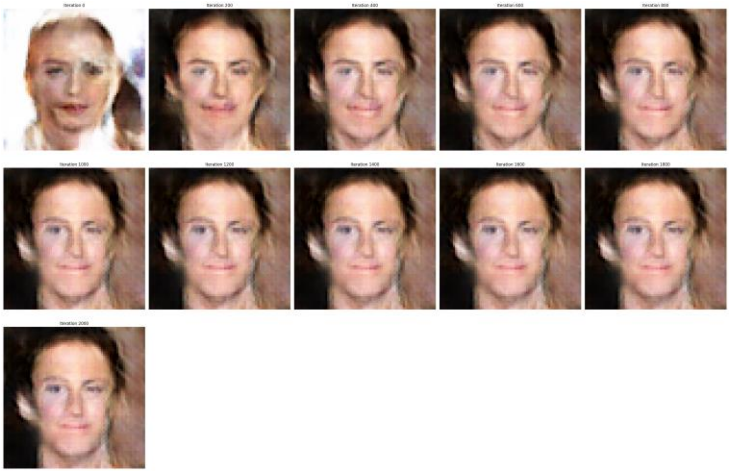
RESULTS – CELEBA DATASET

L2 NORM (MSE)

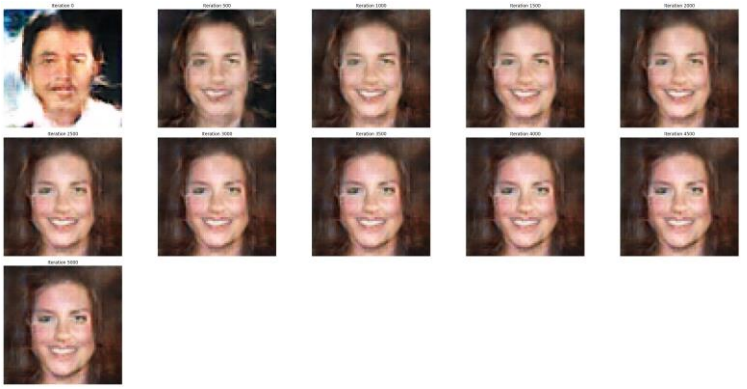




PROGRESS OVER ITERATIONS:



PROGRESS OVER ITERATIONS:



As we can spot reconstruction of digits is an easier task than reconstruction of human faces and this is due to the reasons explained above in Question 2. If we multiply the original image by a binary mask in a key region in the middle of the human face/digit the reconstruction both human faces/digits becomes a hard task because there is a missing part of the image that the Generator needs to complete according to its previous training session. Sometimes even the digits are not reconstructed correctly and what was originally a "4" digit the Generator reconstructs as "9" digit which is incorrect – this can happen because of the resemblance of the two digits. On CelebA dataset the results are somewhat alike to the results we got in section 3.A. – The human faces are a bit distorted and doesn't look exactly like realistic human being, but the inpainting part of the image is completed by the Generator pretty well – it completes the smile if half of it is missing/completes details in the face that were omitted as a result of the activation of the mask on the original image.

For image restoration tasks on the $MNIST$ and $CelebA64$ datasets, the choice between $L2\ Norm$ and $L1\ Norm$ for inpainting and denoising can depend on the characteristics of each dataset and the specific task at hand. Let's analyze each dataset and task individually:

MNIST DATASET

INPAINTING:

In the case of inpainting missing pixels in $MNIST$ images, both $L2\ norm\ and\ L1\ norm$ can yield satisfactory results. The $L2\ norm$, also known as the Euclidean norm, promotes the generation of pixel values that minimize the overall squared difference between the restored and original pixels. This can lead to smoother transitions and visually appealing inpainted regions. However, since $MNIST$ images are relatively simple and contain well-defined shapes and edges, the $L1\ norm$ could be sufficient. The $L1\ norm$ encourages sparse solutions and can better preserve sharp edges and fine details in the restored areas.

DENOISING:

For denoising $MNIST$ images task, the $L2\ norm$ is commonly used and generally performs well. The $L2\ norm$ penalizes the squared difference between the generated image and the corrupted image, which can effectively reduce the impact of noise. The smoothness of the images in $MNIST$, along with the simplicity of the digits, allows the $L2\ norm$ to capture the overall structure of the digits while suppressing the noise.

CELEBA64 DATASET

INPAINTING:

In the case of inpainting missing pixels in $CelebA64$ face images, the $L1\ norm$ is often preferred. Faces in $CelebA64$ exhibit complex textures, fine details, and intricate facial features. The $L1\ norm$ promotes sparsity and tends to preserve sharp edges and fine details, making it suitable for inpainting tasks in this dataset. It can help generate visually plausible results with smoother blending of the inpainted regions while maintaining the natural appearance of facial features.

DENOISING:

Similar to $MNIST$, denoising $CelebA64$ images can be effectively performed using the $L2\ norm$. The $L2\ norm$ can effectively reduce the impact of noise in images with complex textures. However, it's important to consider that $CelebA64$ images contain more diverse facial expressions, poses, and lighting conditions compared to $MNIST$ digits.

In summary, for inpainting tasks, the $L1\ norm$ can be a suitable choice for both $MNIST$ and $CelebA64$ datasets, given their respective characteristics. For denoising, the $L2\ Norm$ is commonly used and generally performs well for both datasets. However it was important during the exercise to examine both the Loss Functions that should be applied on a specific sample of the data through trial and error process.