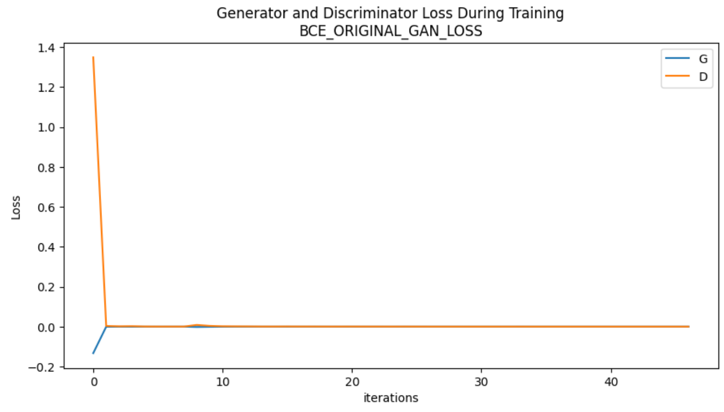
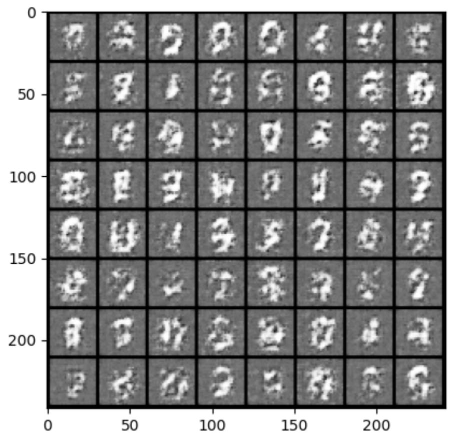
Ex 3 GANS

Amitai ovadia 312244254

Question 1 loss saturation:

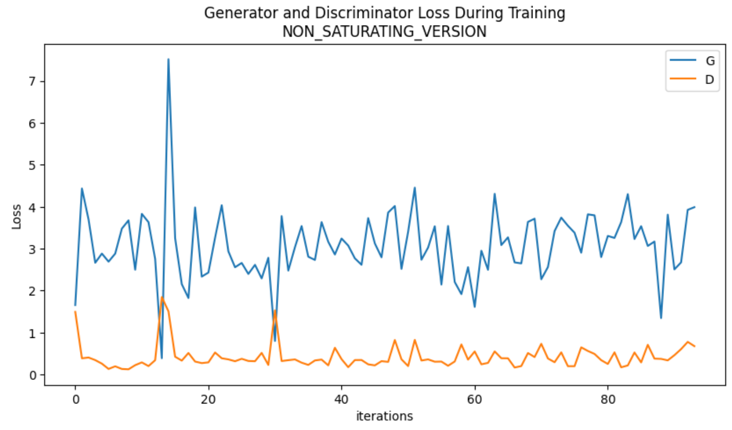
I constructed a Generator and Discriminator based on the Torch GAN architecture and tried 3 different losses.
Using 3 training steps for the Discriminator for every 1 step of the generator.
Trained for 40 epoches. Also i used z latent space of size 100.
(i) the original GAN cross entropy :

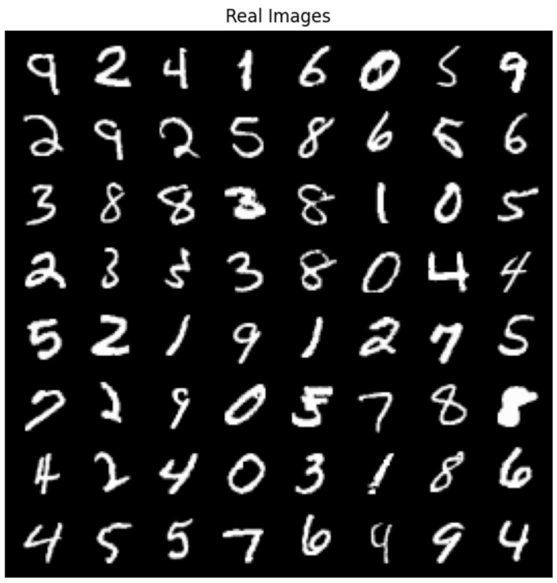


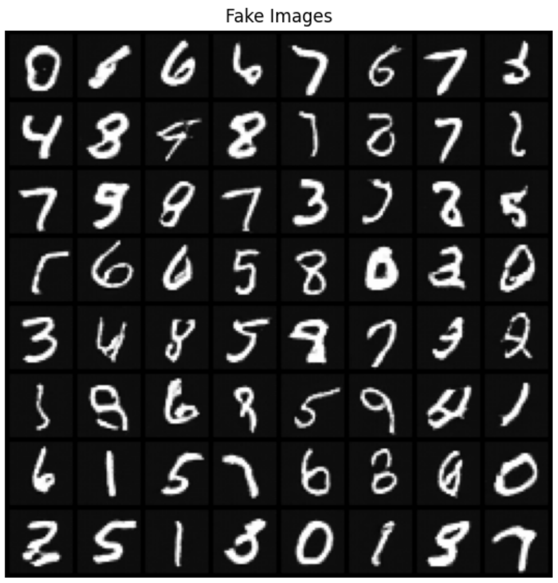


As can be seen: full saturation. The Generator wasn't able to train probably because of the vanishing gradients problem

(ii) non-saturating version of this loss:
Meaning that now instead of G minimizing $log(1 - D(G(z)))$ we just maximize $log(D(G(z)))$, which is better because there we get non zero gradients:

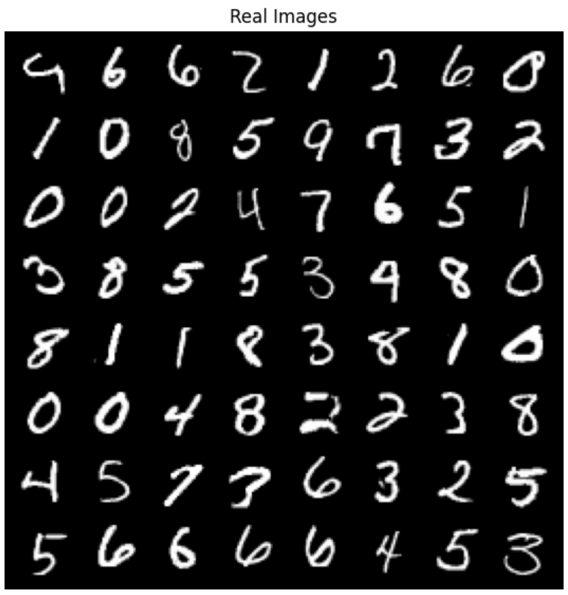


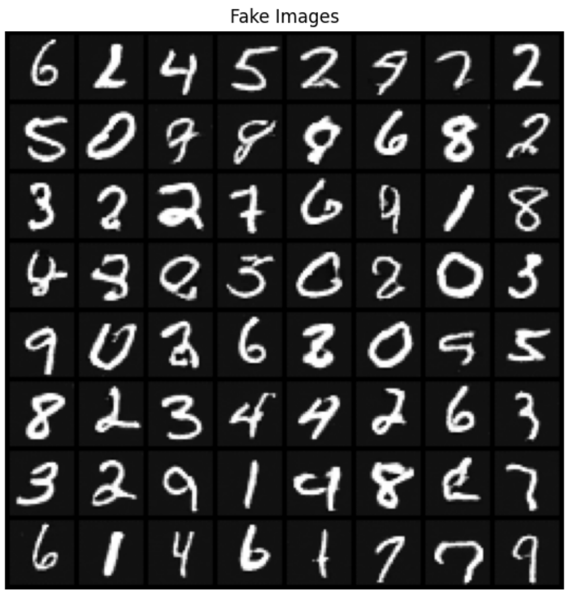




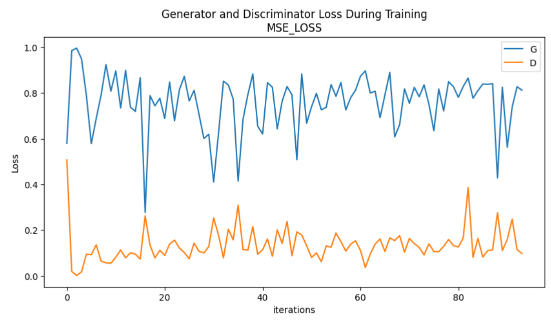
We can see that the results are pretty good for the non-saturated loss.

(iiI) MSE loss: here we do maximize $|D(G(z)) \ - \ 0|^2$ or in other words, G tries to make D output 1 on $G(z)$.





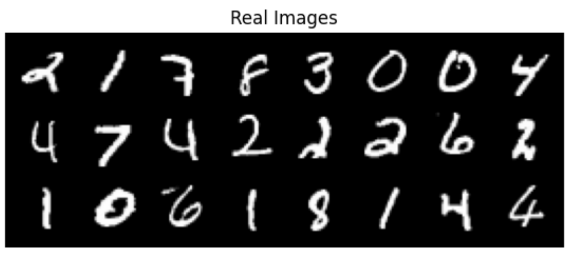
And the loss function behaved like this:

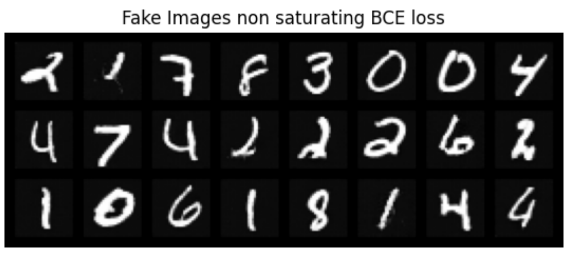


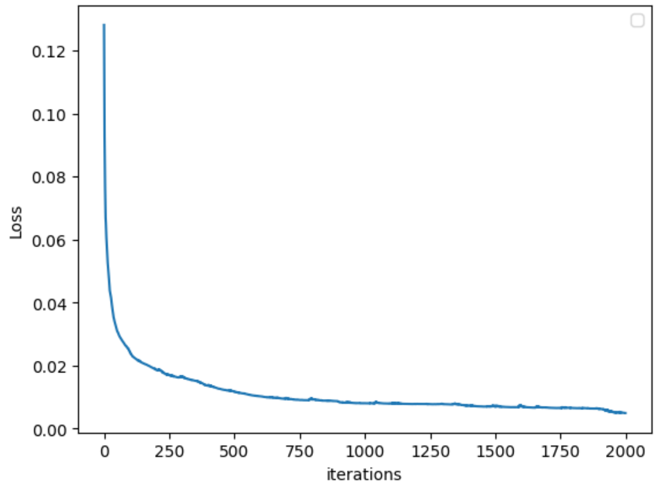
Basically we can conclude that both the MSE loss and the non saturation loss resulted in pretty good results, probably because the generator wasn't facing a vanishing gradients problem due to saturation. Unlike the original loss from class that resulted in saturation and the inability of the Generator to train properly.

Question 2 Model Inversion:

Here I used an already trained Generator (trained using the BCE loss),
Took an image I from the training set and optimized for the latent space z such that $G(z) = I$, or on other words, tried to find the best z such that G could recreate I using z.
It was done by starting with an initial random guess for z, and the searching through the latent space of z using gradient descent and finding $min_z(||G(z) - I||_2)$.
I did it using 5000 optimization steps to ensure convergence and a high learning rate of 0.1 and Adam optimizer.
This are the results of the G trained using the BCE loss (the non saturating version)







We can see that the results were less noisy than the original images (the '6' and the '0').
So the noise was not reconstructed, vs the general notion of the character that was encoded was actually reconstructed. An explanation to it might be that G acts as a decoder of the relatively small latent space (100) and noise is a thing that cannot be compressed and therefore is not preserved and reconstructed.

Question 3 Image restoration:

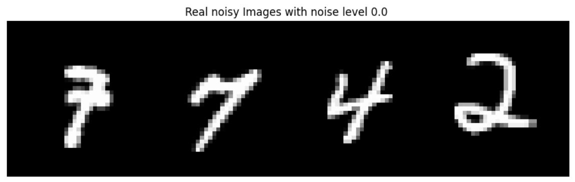
Here again i used G that was trained using the BCE loss. And optimized over the latent space to find the z that minimizes the reconstruction loss in a MSE least squares manner:

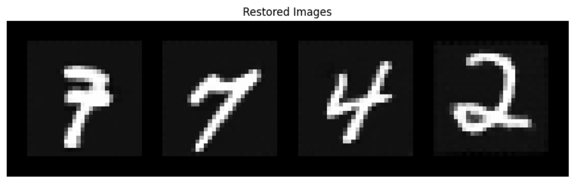
$min_z(||G(z) - I||_2)$

I used the same function as before using 3000 optimization steps

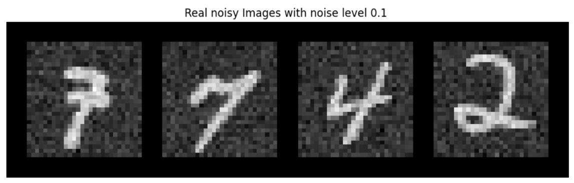
(a) first case: restoration of noisy images (denoising), i added gaussian noise to the image and tried to find the best z that minimizes the reconstruction loss:

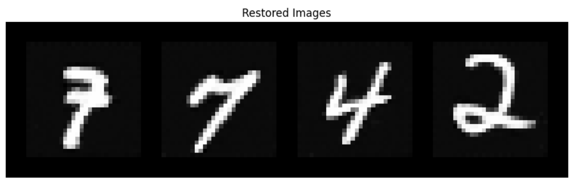
For 0 noise I got:



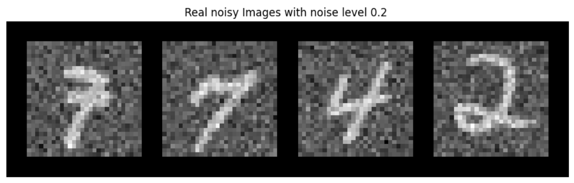


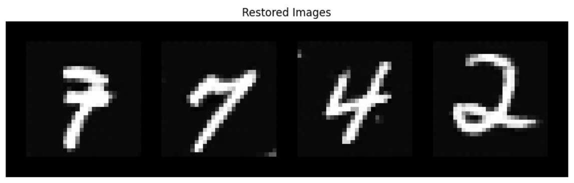
For 0.1 * $N(0, I)$ noise:



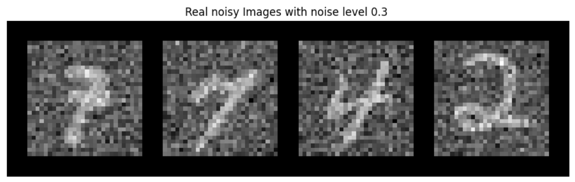


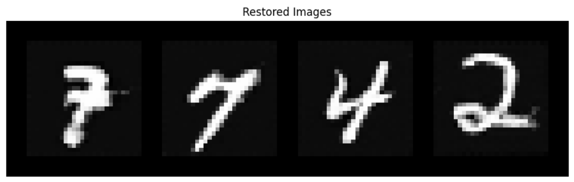
For 0.2 * $N(0, I)$ noise:



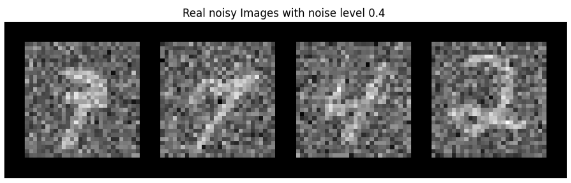


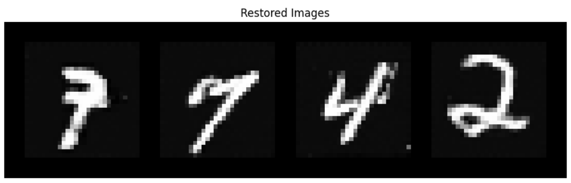
For 0.3 * $N(0, I)$ noise:



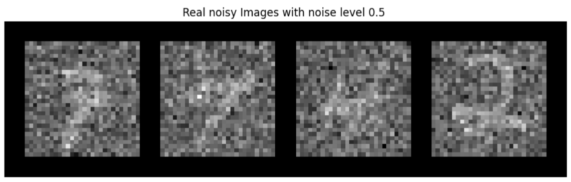


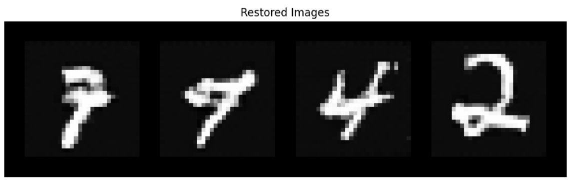
For 0.4 * $N(0, I)$ noise



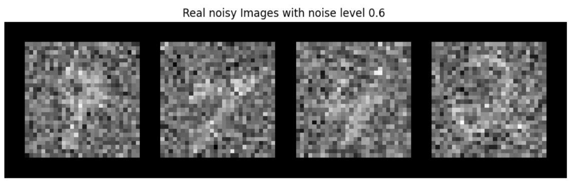


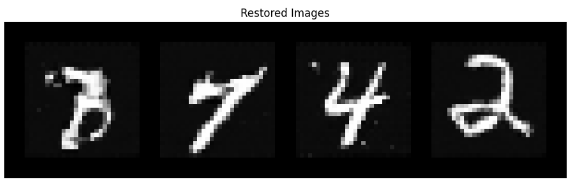
For 0.5 * $N(0, I)$ noise



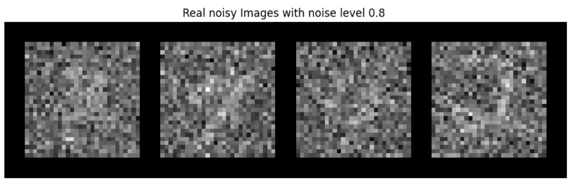


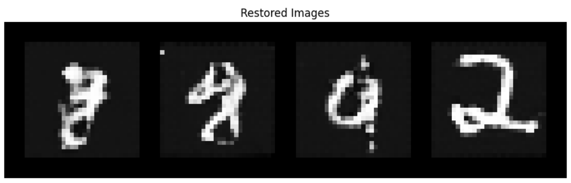
For 0.6 * $N(0, I)$ noise



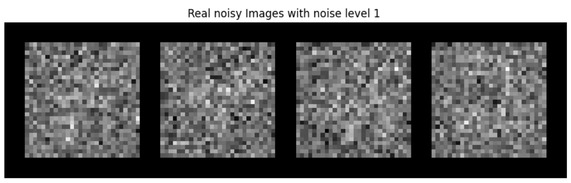


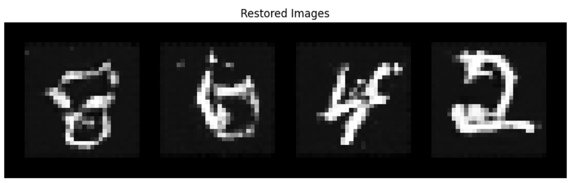
For 0.8 * $N(0, I)$ noise





For 1 * $N(0, I)$ noise

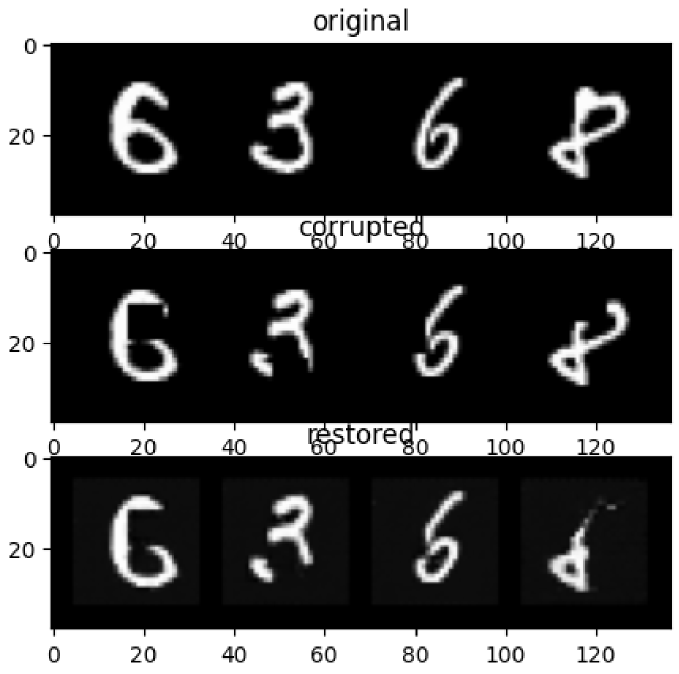




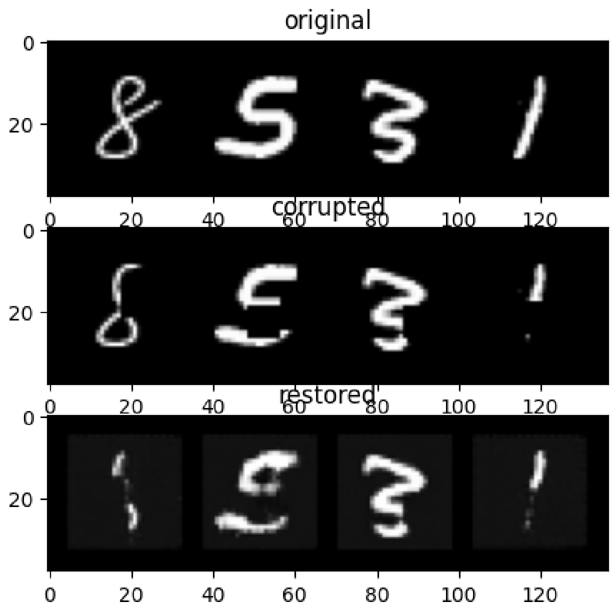
As can be expected the quality of reconstruction decreases as we amplify the noise, but it still gives pretty good results even for significant noise like 0.3, 0.4.

(b) Filling missing pixels in images with holes (inpainting):

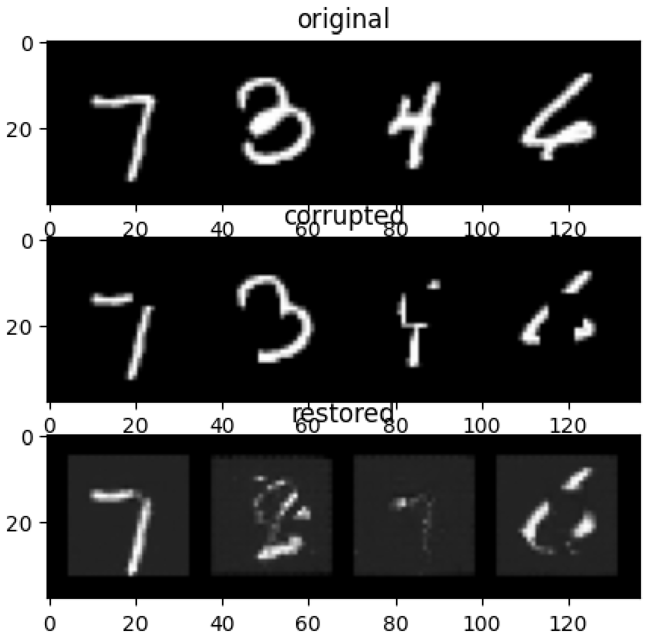
1 hole:



2 holes:



3 holes:



As can be seen, here the reconstruction performed poorly in comparison with the previous task. One possible reason could be that the notion of the character in the image is completely different after the image corruption and therefore much harder to reconstruct.

Theoretical Questions:

Question 1

Basically it was best if we could just find $P(x)$ and just sample new images from it directly. In the GLOW method there is an attempt find the optimal function that can take a vector from a simple distribution, and transfer it into a vector from $P(x)$ using a change of variable law of probabilities. We can do it by computing the log-likelihood function of $P(x)$ and optimize it as a loss fucntion

So here is how we compute the log-likelihood of $P(x)$:

$P_{\theta}(x) = |\frac{\nabla M^{-`1}(x)}{\nabla x}| P_z(M^{-`1}(x))$

While M in this setting is a composition of invertible functions $h_k \circ h_{k-1} \circ ... \circ h_1$

Because each h_i is invertible then $M^{-`1}(x)$ computable and also $M^{-`1}(x) = z$

And we get $P_{\theta}(x) = |\frac{dz}{dx}| P_z(z)$

We can apply log the the 2 sides and get:

$log(P_{\theta}(x)) = log(|\frac{dz}{dx}| P_z(z)) = log(|\frac{dz}{dx}|) + log(P_z(z))$ and using the chain rule:

$log(P_{\theta}(x)) = \sum_k log(|\frac{dh_{k-1}}{dh_k}|) + log(P_z(z))$

And again, this can be computed because we can compute $M^{-`1}(x)$.

In the GAN scenario G didn't learn $P(x)$ but only the parameters of G_{θ} to create x that could fool the discriminator and so $P(x)$ cannot be computed in GAN.

In the GLO scenario we match a $\{z_i\}$ a vector from a multi gaussian distribution for every image I_i in our dataset. And we learn an encoder G to generate the image I_i given z_i by optimizing

$min_{\theta,\{z_i\}} \sum_i ||G(z_i) - I_i||$

Here we also learn the parameters θ but without knowing anything about $P(x)$

Question 2

The loss function that G minimizes is $E_z[log(1 - D(G(z)))]$

During the beginning of the GAN training it's easy for the discriminator D to distinguish between the real images and the $G(z)$. Therefore the expression $D(G(z))\sim 0$ in the beginning of the training.
In the last layer of D we have a sigmoid, and for $D(G(z))\sim 0$ the value that feeds into this layer $D_{last}(x)$ must be around $-\ \infty$, but because of how the sigmoid function behaves (it saturates far from $x = 0$) if we try to find the derivative of $D'(G(z))$ it will be almost 0 and the weights wouldn't change and we get vanishing gradients and G will not learn anything

Question 3

a. The problem $min_G V(D, G) = min_G E_x D(x) + E_{p_z}[log(1 - D(G(z)))]$ (we can ignore the first term as it is not dependent on G)
   means that given a fixed D that returns 1 for a real image and 0 to a fake one, We want to find the best G that makes $D(G(z))$ to return 1 or in other words to fool the discriminator D and so the best strategy for G will be to return the images $G(z)$ that make D return 1.
   So this expression solves the maximization of $V(D, G)$ under a fixed D
b. The outer problem, namely $max_D min_G V(D, G)$ means that given a fixed G we want to maximize the expression $E_x D(x) + E_{p_z}[log(1 - D(G(z)))]$.
   In order for this expression to be minimized D needs to 'find out' what are the images that the fixed G is returning that receive high score and lower their score, and $D(G(z)))$ will go down, and $log(1 - D(G(z))$ will go up as D 'wants'.
   This is an easy task for D because for a limited capacity NN G the variety of the images it can generate is small and it's more difficult for it to outsmart D.
c. This can result in mode-collapse. The reason is that the dynamic that G finds a few images that fool D, and then D finds out which are those images easily because there are a few of them, so G then uses a few other images and so on an so forth can just continue. And the result of that is that G will just generate a few images that by no means cover all $P(x)$.