

# רשתות נוירונים לתמונות | תרגיל 3

שם: תמוז גיטלר | ת"ז: 312243439

20 במאי 2023

# Generative Adversarial Networks (GANs)

## Practical Questions:

### Question 1 - Loss Saturation

Model parameters:

Laten dimension	Out channel	Batch size	epochs	learning rate
<b>100</b>	<b>1</b>	<b>64</b>	<b>15</b>	0.0002

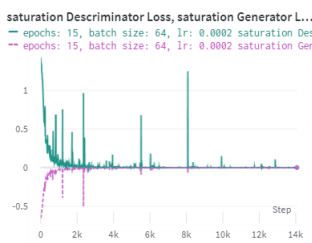
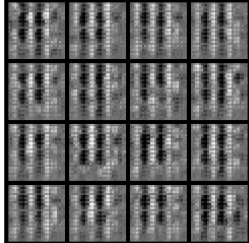
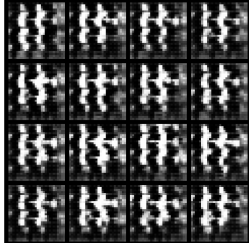
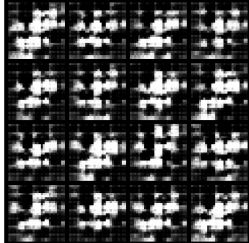
1. In the following tables the pink loss referse to the generators loss and the green loss is the Discriminator loss.
2. I decided to sample a fixed random noise and used it at the end of every epoch to generate 16 image. this helped me to better understand if the Generator was able to learn while using different loss functions.
3. I will present the generated images in epoch 0, 7 and at the last epoch.
4. I also used in the training procces a boolean variable named “saturation” and ill explain about it below

```
# calc G loss
if saturation:
    g_loss = -criterion(output, fake_pred)
else:
    g_loss = criterion(output, real_pred)
```

6. At the beggining i tried to apply multiple training optimization steps to the Discriminator per a single iteration for the Generator, but the Discriminator was to powerfull which lead to bad results by the Generator.

#### i) original GAN cross entropy

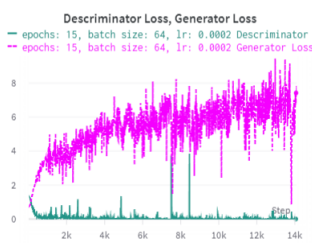
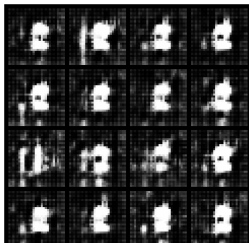
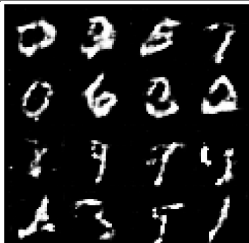
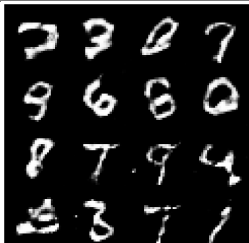
We train the Generator by minimizing  $\log(1-D(G(z)))$ , which do not provide sufficient gradients for updating the model weights, thus “saturation” = True.

Loss	Generated Images in epoch 0
 <p>saturation Discriminator Loss, saturation Generator L...</p> <p>— epochs: 15, batch size: 64, lr: 0.0002 saturation Des</p> <p>— epochs: 15, batch size: 64, lr: 0.0002 saturation Gen</p>	
Generated Images in epoch 7	Generated Images in epoch 14
	

## ii) non-saturating version of this loss (as mentioned in class)

We train the Generator by maximizing the  $\log(D(G(z)))$ , thus “saturation” = False.

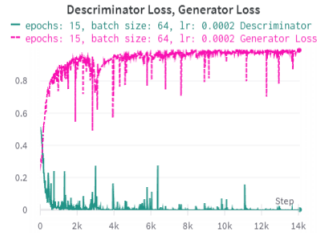
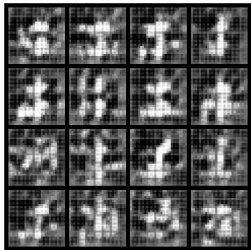
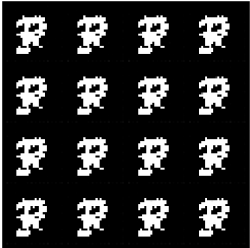
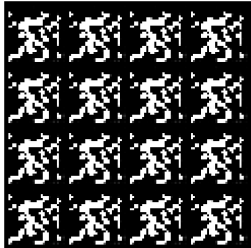
As seeing below, the generator was able to produce better images from the same latent vector!

Loss	Generated Images in epoch 0
 <p>Discriminator Loss, Generator Loss</p> <p>— epochs: 15, batch size: 64, lr: 0.0002 Discriminator</p> <p>— epochs: 15, batch size: 64, lr: 0.0002 Generator Loss</p>	
Generated Images in epoch 7	Generated Images in epoch 14
	


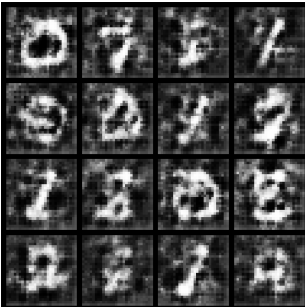
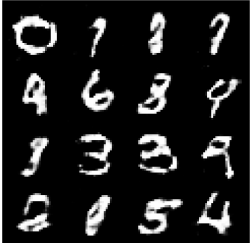

## iii) least-squares loss (L2)

When i used the MSELoss it lead to poor performance - saturation.

the generator failed to receive meaningful feedback from the discriminator on how to improve its generated samples and it struggled to produce diverse and realistic images.

Loss	Generated Images in epoch 0
	
Generated Images in epoch 7	Generated Images in epoch 14
	

After researching i decided to remove the Sigmoid Activation function at the end of the discriminator layers, which allowed him to output values that werent bound to the specific range  $[0,1]$ . This allowed for more informative gradients and prevented the vanishing gradients problem

Loss	Generated Images in epoch 0
	
Generated Images in epoch 7	Generated Images in epoch 14
	

## Question 2 - Model Inversion

The features that were accurately reconstructed were:

1. general structrue - the latent vector was able to learn features that helped the generator to reconstruct

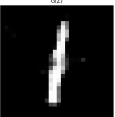
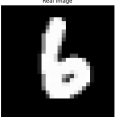
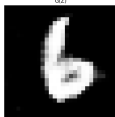
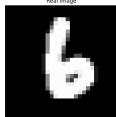
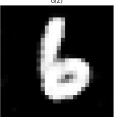
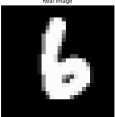

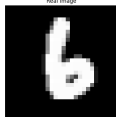
the general structure and capture the overall shape. in the MNIST data set each digit has a distinct global structure, such as a circular shape for "0", which were sucessful learend by the latent vector  $z$ , this esured that the reconstructed images maintained the correct digit identity.

2. straight lines & curves - most of them were well-reconstructed which means that the core structure of each digit was accurately reproduced.

### The features that werent accurately reconstructed were:



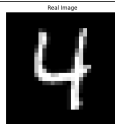
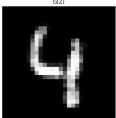

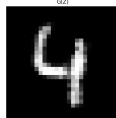
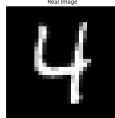
1. Small details - some of the fine details, such as the sharpness of corners or the consistency of line thickness, may not be faithfully reproduced in the reconstructed images.
2. sharpness - the reconstructed images aren't sharp as the original images and consist noise, this noise is often a result of the optimization process.
3. collor intensity - The reconstructed images suffers from a loss of contrast and in some cases fail to capture the smooth transitions between different shades of gray (white of the digit to the black background).

### Example of the reconstructed image proccess:

Epoch 0		epoch 10k	
<small>Q21</small> 	<small>Real Image</small> 	<small>Q21</small> 	<small>Real Image</small> 
epoch 20k		epoch 30k	
<small>Q21</small> 	<small>Real Image</small> 	<small>Q21</small> 	<small>Real Image</small> 

## Question 3 - Image restoration

### denoising:

Blurred Image		epoch 0	
		<small>Q21</small> 	<small>Real Image</small> 
epoch 25k		epoch 50k	
<small>Q21</small> 	<small>Real Image</small> 	<small>Q21</small> 	<small>Real Image</small> 

In this part i decided to use MSELoss function measures the average squared difference between the generated blurred image and the blurred image at each pixel.


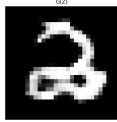





in addition the MSE loss is sensitive to outliers, including noise in the image which can help the optimize to

improve z and generate better images.

### inpainting:

In this part i decided to use L1Loss function which measures the absolute difference between the generated inpainted region and the inpainted image.

this loss is less sensitive to outliers compared to MSE which helps us to reconstruct specific part of the image (the inpainted part), and it also preserve edges better than MSE loss.

Real Image	epoch 0 (real image here is the inpainted image)	
		
epoch 25k (real image here is the inpainted image)	epoch 50k (real image here is the inpainted image)	
 		

## Theoretical Questions:

### Question 1

Reminder:

#### GLOW: Generative Flow

Reminder:

$M_\theta(z) = x$ ,  $z \sim N(0, I)$ ,  $x \sim P(x)$   
 change-of-  
variable      easy-to-  
sample      target  
dist.      distribution  
 $P(x) = \left| \frac{\nabla M^{-1}(x)}{\nabla z} \right| P_z(M^{-1}(x))$   
 density expansion/  
contraction term

In case of a composition:  
(as the case of multi-layered net.)

$$M = h_k \circ h_{k-1} \circ \dots \circ h_1$$

$$\log P(x) = \underbrace{\log P(z)}_{*1} + \sum_i \log \left\| \frac{\partial f_i}{\partial f_{i-1}} \right\|$$

\*1 - represents the log-probability density of the prior distribution

\*2 reepresents the sum of the log-determinants of the Jacobian matrices of each layer in the transformation function

to compute P(x) for a given input sample x we should:

1. randomly init z from the standard Gaussian distribution
2. calculate f(z)=x where f is the invertible transformation function

3. Compute the log-determinant through each layer -  $J(f)$  ( $J$  - jacobian)
4. Sum the log-determinants computed in the previous step -  $\log|J(f)|$
5. compute  $\log(P(z))$
6. sum the values from section 5 and 6

$P(x)$  cannot be computed in GAN and GLO (generative latent optimization) because they do not explicitly compute the Probability Density Function which is needed to calculate  $P(x)$ . GANS trains a generator network that can generate realistic samples that resemble the training data. while GLO aims to find a latent space where the generated samples resemble the real data as closely as possible.

## Question 2

we have seen in class the following formula :

$$\min_G \max_D V(D, G) = \min_G \max_D E_{p_{\text{data}}} [\log D(x)] + E_{p_z} [\log (1 - D(G(z)))]$$

when deriving this formula with respect to  $G$  we get the following formula:  $E_{p_z} [-D(G(Z)) \cdot \frac{1}{1-D(G(z))}]$

1. when  $D(G(z))=1$  the derivative value aspire to  $-\infty$
1. when  $D(G(z))=0$  the derivative value aspire to 0 - this is the case of vanishing gradients

## Question 3

Note - im saying images and not samples because we are dealing with MNIST/CelebA data set in this exercise.

**A)**

In DCGan, the Generator's objective is to generate images that are indistinguishable from real images. Im going to explain below why minimizing  $V(D, G)$  improves the Generator ability to produce realistic images that can deceive the Discriminator.

Recall that  $V(D, G) = E[\log(D(x))] + E[\log(1 - D(G(z)))]$

$D(G(z))$  represents the discriminator's prediction of the probability that the generated samples are real.

$E[\log(1 - D(G(z)))]$  - represents the expected log-probability assigned by the discriminator to generated images being classified as fake.

in conclusion, the inner problem  $\min_G V(D, G)$  aims to minimize the probability of  $D$  thinking that the generated images are fake, which is equivalent to maximizing the probability of  $D$  thinking the generated images are real!

**B)**

the outer problem ( $\max_D$ ) tries to maximize the adversarial loss  $V$  given the converged inner problem  $\min_G V(D, G)$ . in other words it tries to maximize

$1 - D(G(z))$ , this value will be maximized when  $D(G(z)) \rightarrow 0$  which means that  $D$  detects that the given images are fake and were generated by  $G$ .

I think that the outer problem of maximizing the loss by optimizing  $D$  with fixed  $G$  is easier than the inner problem of minimizing the loss by optimizing  $G$  with a fixed  $D$  because discrimination is generally an easier task compared to generating realistic samples from scratch.

C)

Assuming  $D$  fails to achieve the exact solution it needs to when using the inverted formula,  $D$  will fail to discriminate between generated and real images.  $G$  will not be able to learn how to generate realistic images and it can lead to lower-quality generated samples.