TASK 1 - How easy is it to create a deepfake?

Deepfake generator

1. Explain what reenactment is

Reenactment is a technique that uses machine learning algorithms to make convincing fake videos of people saying things they have not said, or doing something they have not done. Reenactment is achieved by training a deep learning model on a dataset og images and videos of the people that are going to be in the fake video. This information is used to generate images and videos that simulate the people's movements and speech as convincingly as possible. Facial reenactment is thus the modifications done to people in the form of changes in movement of lips, head and facial expressions

2. Demo.ipynb

The following pictures were used:

Wearing glasses, face obscured by diving gear, mouth covered by clothing, and weird facial expression

3. Extract frames from fake video

Here, I used mostly the same frames and compared the same frame in different videos. I also Compared different frames from the same video in some cases. The frame number is in the image descriptions

Trump:

All are frame 32

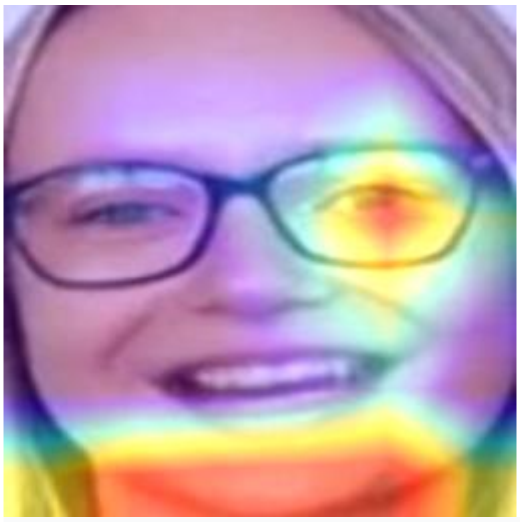
Obama from the same video, frames 32 and 79

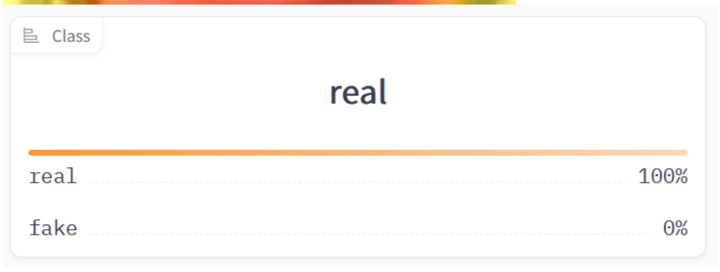
DiCaprio, same video frames 32 and 45

4. Feed images to detector

5. Report deepfakes and results with explainability map. show for at leat 4 videos, explain results

Trump:

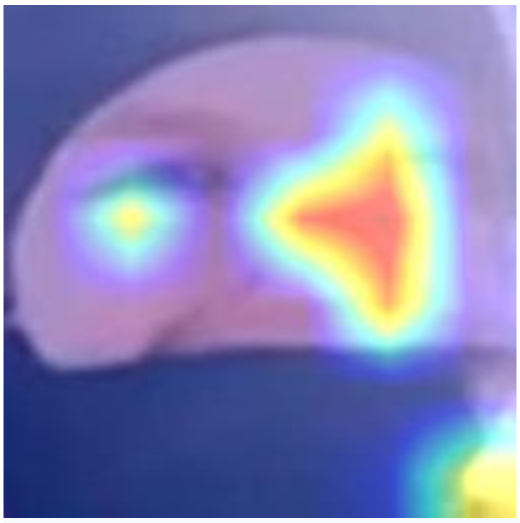


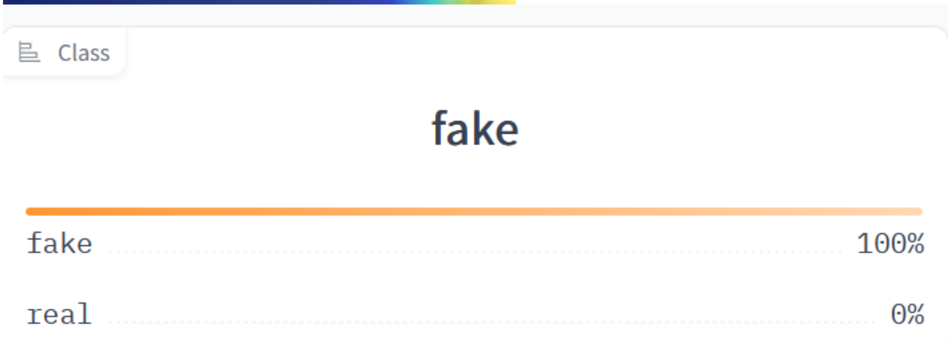


Class

real

```
real                                   100%
fake                                     0%
```

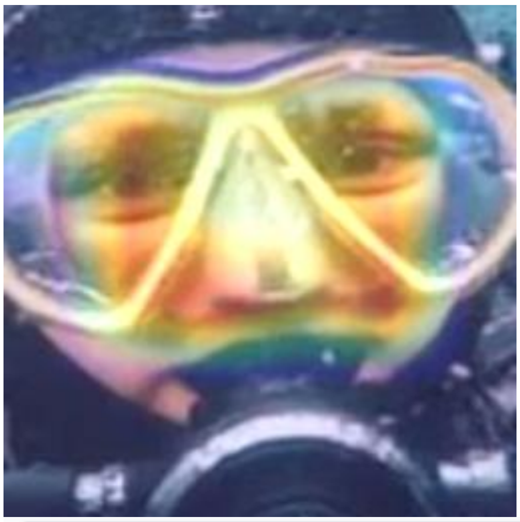


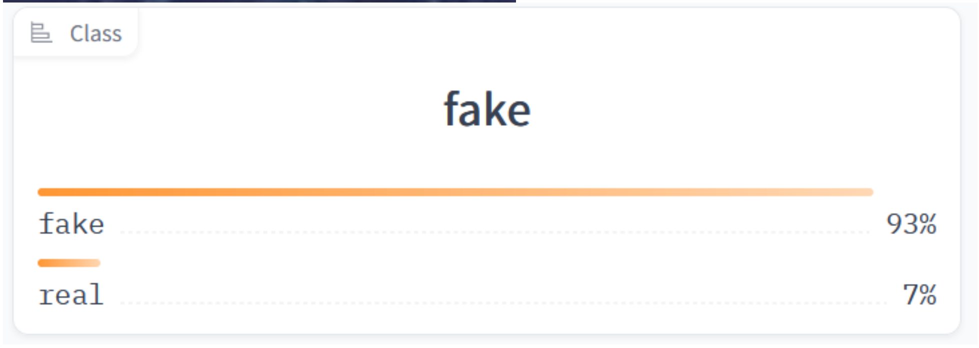


Class

fake

fake	100%
real	0%



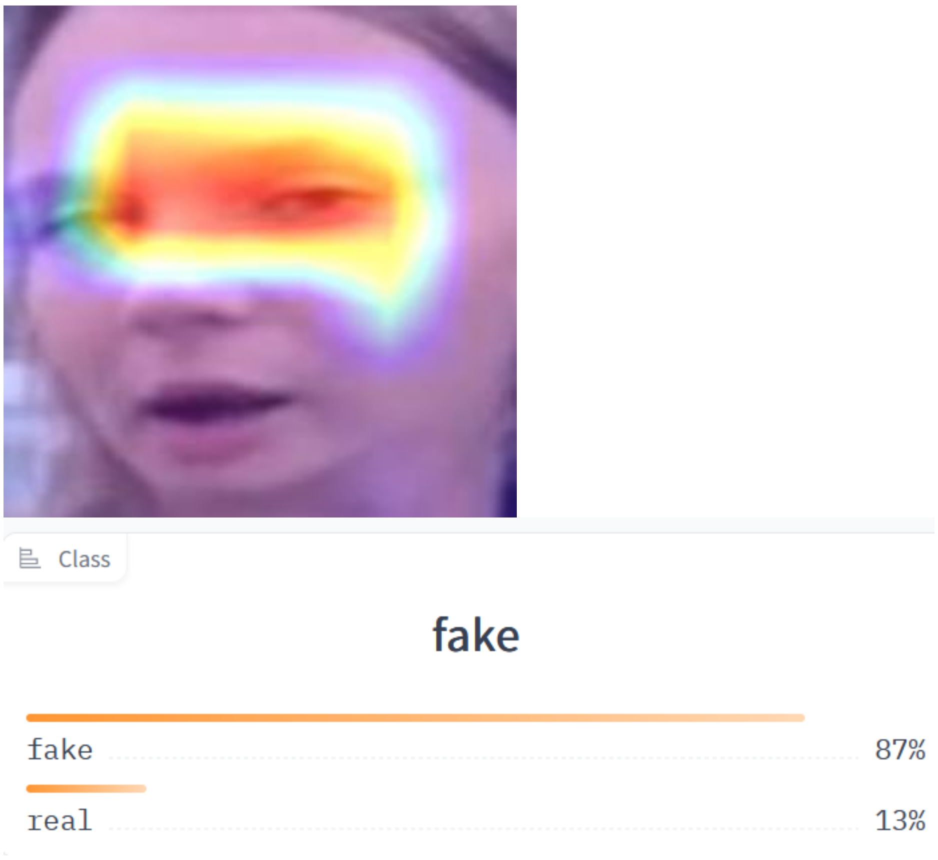


Class

fake

fake	93%
real	7%

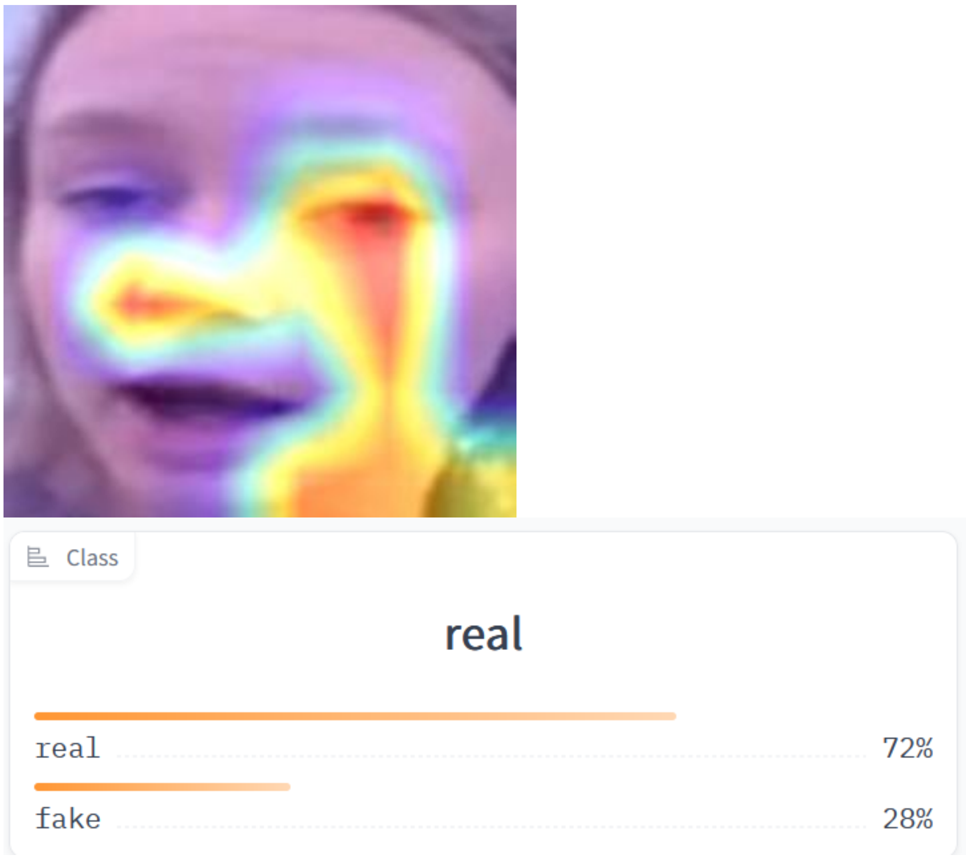
Obama:



Class

fake

fake	87%
real	13%

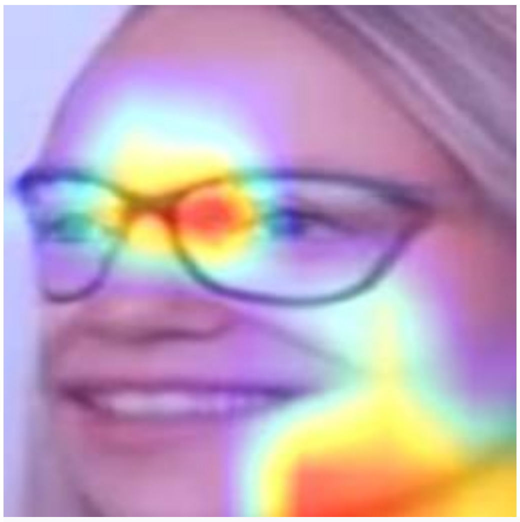


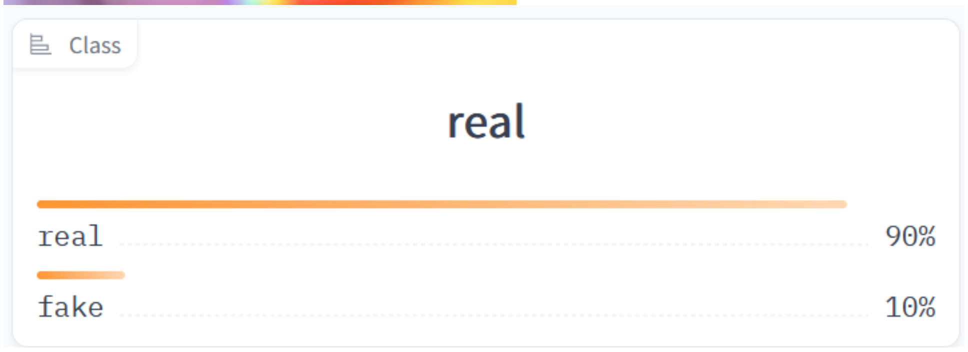
Class

real

real	72%
fake	28%

DiCaprio

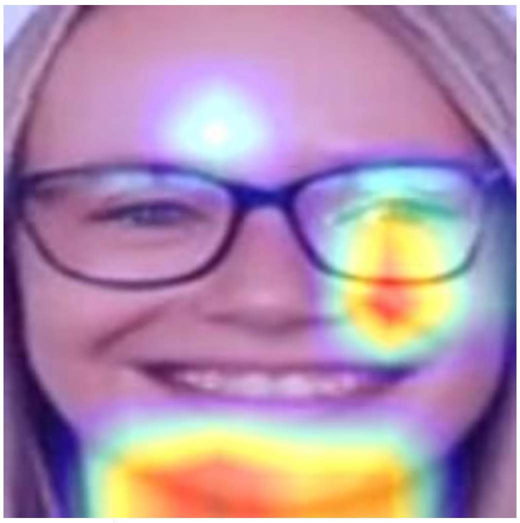


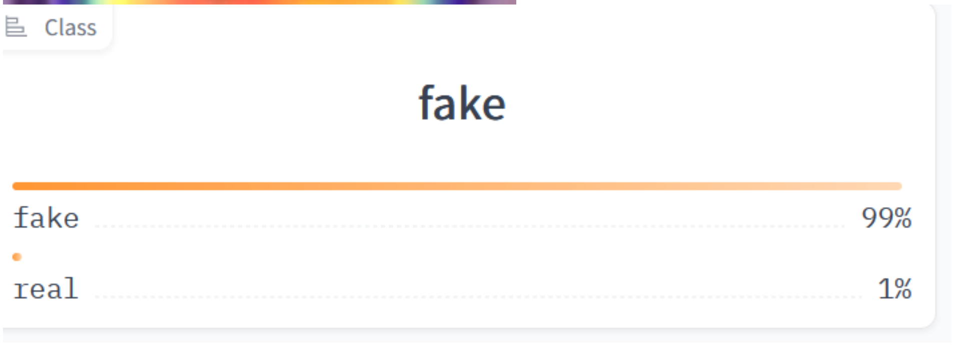


Class

real

```
real                                  90%
fake                                  10%
```





Class

fake

fake 99%

real 1%

6. What are potential benefits and risks associated with deepfake technology?

Benefits:

Deepfakes can be used in the film industry to create realistic special effects, such as bringing back deceased actors or creating younger versions of actors.

Deepfakes can also be used in education to create realistic simulations, such as medical simulations or historical reenactments.

However, in my opinion the **Risks** are greater than the benefits.

Deepfakes can be used to create fake news or propaganda, which can spread misinformation and deceive the public. In todays climate, this is extremely relevant, and I think we will se a lot of this technology used for swaying peoples perception about what is happening in the world.

Deepfakes can be used in frauds, such as creating fake identities or forging documents. It can alsobe used to bypass security measures, such as facial recognition systems or voice biometrics.

TASK 2 - From Data to Dollars: Using GANs to Generate Bags for Your Online Shop

I found this challenge very time consuming and difficult. Especially since I have no background in machine learning. Therefore, I know nothing about neural networks, GANs etc. I spent a lot of time reading and trying to figure out what I was doing. In the end, I managed to get some results and write this report, but I think that for this course, Imaging Security, that has no prerequisite of MALIS, deep learning or similar, this challenge was too hard.

BUT, I had a lot of fun and learned very much. My hope is that our backgrounds are taken into consideration here, since this is not a machine learning course.

2. Report of experiments

Architecture

I experimented with many different architectures. For the generator and Discriminator I added blocks to increase the depth of the neural network. This can allow the generator to learn more complex features and generate higher quality images

My generator and discriminator blocks consist of two main types of layers:

Convolutional layers: Layers that apply convolution operations to the input. In the case of the generator, convolutional layers are used to gradually increase the size of the image until it reaches the desired size. In the case of the discriminator, convolutional layers are used to extract features from the input image.

Batch normalization layers: Layers that normalize the inputs to each layer, typically by subtracting the mean and dividing by the standard deviation. This helps to stabilize the training process by reducing the internal covariate shift.

In addition to these layers, I'm also using ReLU activation functions to introduce nonlinearity to the network and dropout layers to prevent overfitting.

Some of these were given in the notebook, some I added myself. See the .ipynb

I also tried to use a hyperbolic tangent (tanh) activation function, but concluded that the Sidmoid performed better in my case

Parameters

For the parameters, I started by decreasing the learning rate to allow the model to converge more slowly and potentially reach a better optimum. This improved my results.

Then, i tried to increase the batch size to allow the model to see more examples in each iteration. It helped a bit

And also increased epochs, the number of times I iterate through the dataset, to allow the model more time to learn from the data. I got better images with a large epoch.

Then, I changed from cpu to gpu by using cuda.

I also tried to ncrease the dimensionality of the noise vector to make it easier for the generator to create diverse images. However, this had little effect and I ended up leaving it at 64

My findings are that none of the parameters alone could be tweaked in a way that improved my results. I had to run many times and experiment with different architectures and parameters. Even then, I never felt I got very good results, but eventually I settled for just an okay image of bags.

Here are some examples of results I got from my experiments. I chose to download the best images from each iteration (if relevant), in addition to the learning curves. I will only provide the parameters I changed.

The Plot

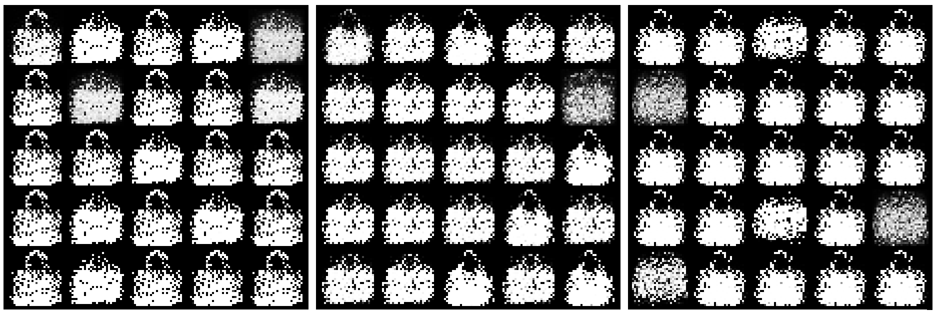
The plot of disc-loss and gen-loss shows how the loss of the discriminator and the generator change during the training process. The discriminator loss measures how well the discriminator can distinguish between real and fake images, while the generator loss measures how well the generator can fool the discriminator by generating realistic images.

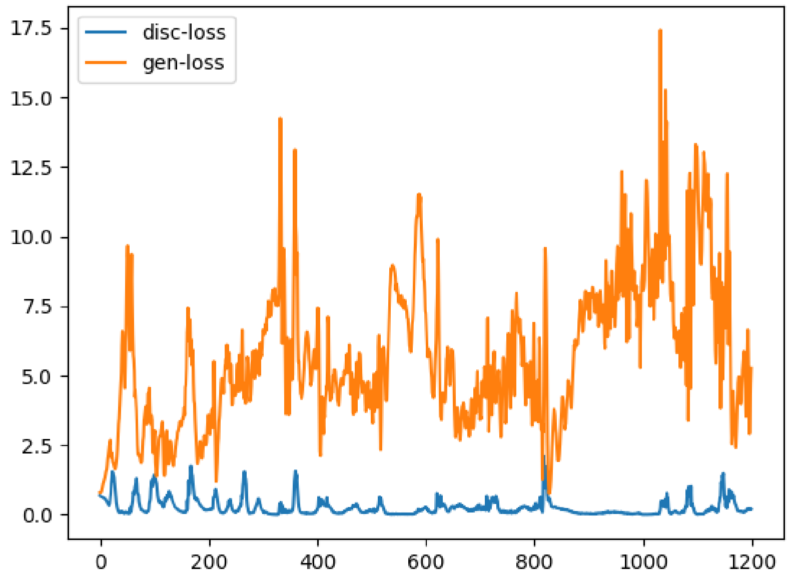
During training, the goal is to minimize the discriminator loss while maximizing the generator loss. If the discriminator loss is decreasing and the generator loss is increasing, it means that the generator is getting better at generating realistic images that can fool the discriminator.

Run No. 9:

3 extra blocks

Parameters: epoch=50, z-dim=2, batch size=256, lr=0.001

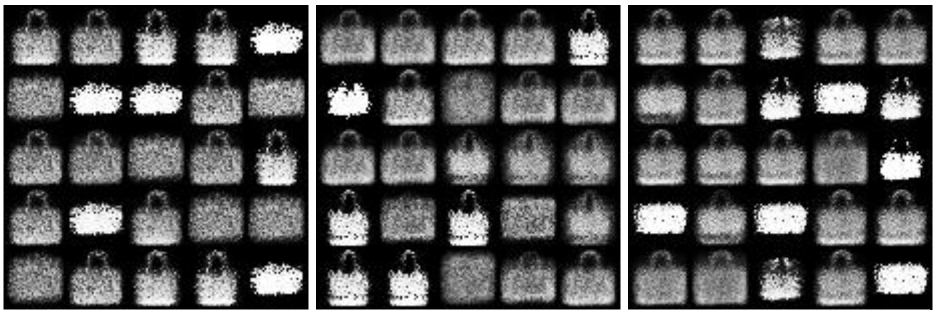


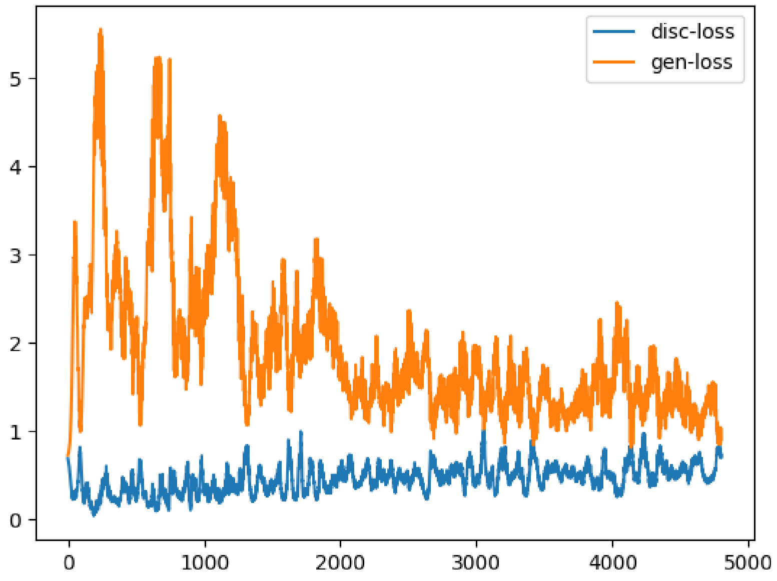


Run No. 16

3 extra blocks, but tweaked the Dicriminator block so it had fewer layers

Parameters: epoch=200, z-dim=64, batch size=256, lr=0.0002

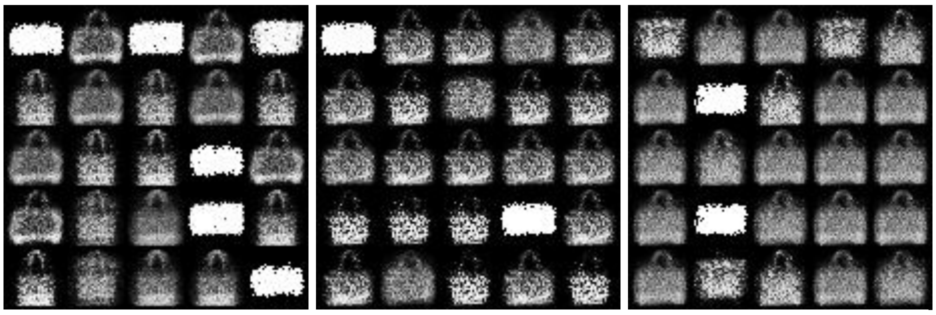


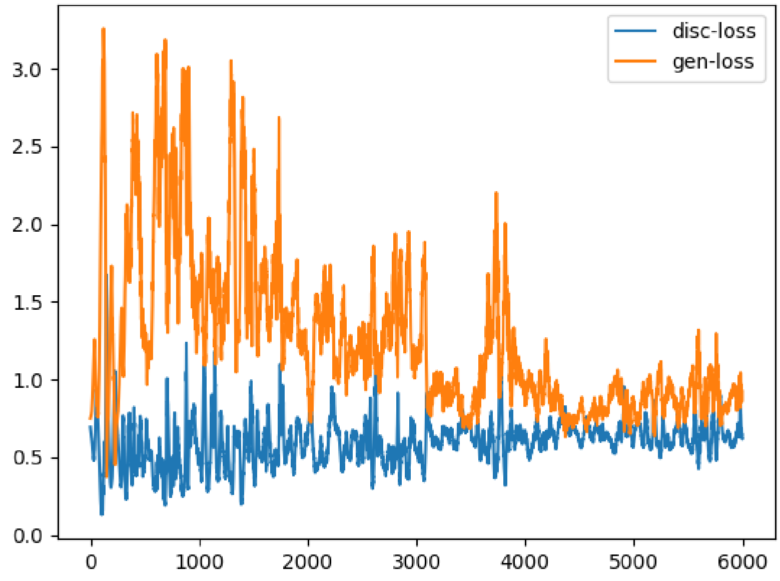


Run No. 21

Same blocks as No. 16

Parameters: epoch=500, z-dim=64, batch size=512, lr=0.0002



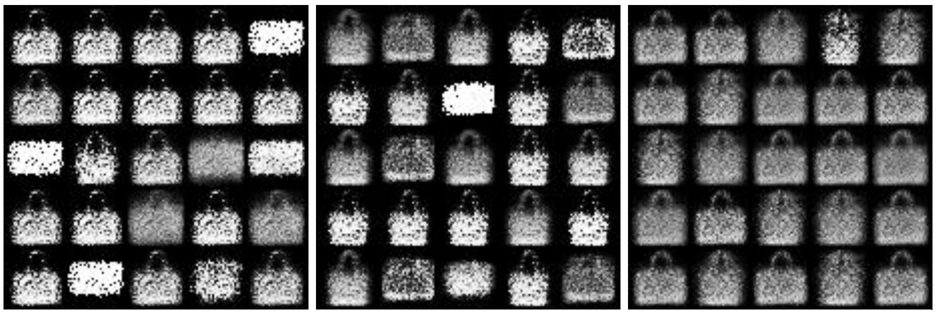


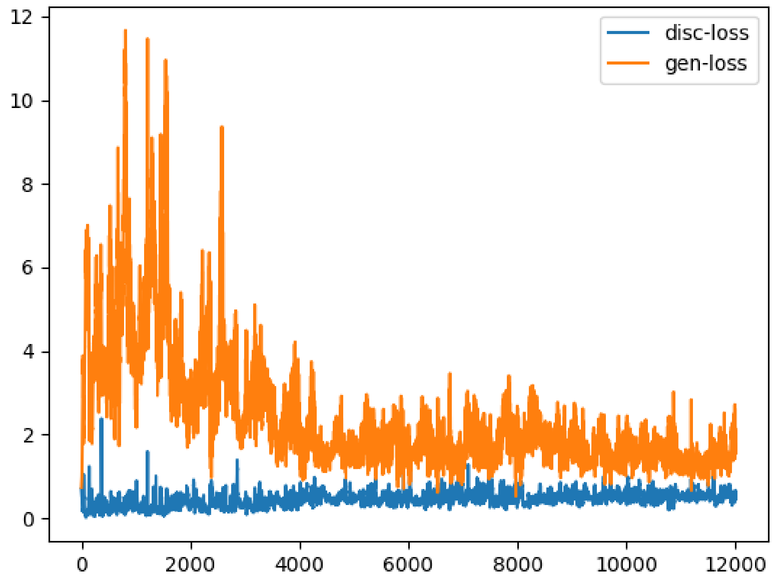
By Now I learned that a very big n_epochs was the way to go. Still, I was not satisfied with my results and kept running and trying different things.

Run No. 28:

3 blocks with deeper layers

Parameters: epoch=500, z-dim=64, batch size=256, lr=0.0002



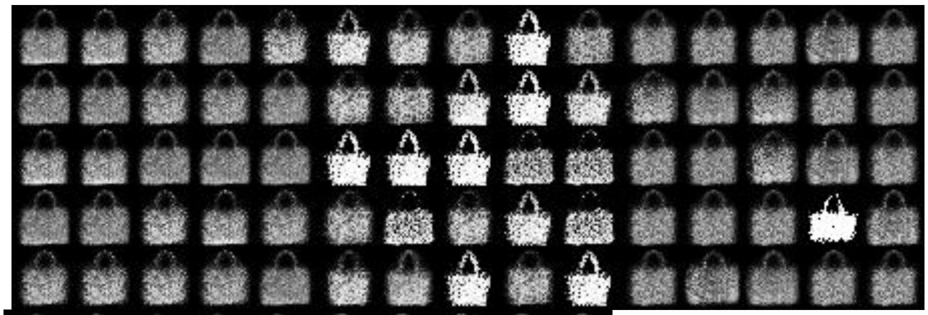


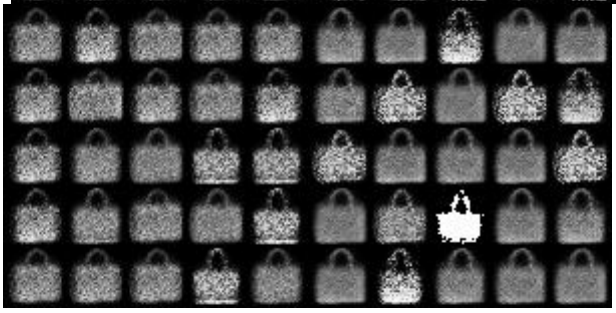
Finally after some more trial and error, I found this result by looking at the best results I had earlier. Then, i increased the learning rate a bit and got this

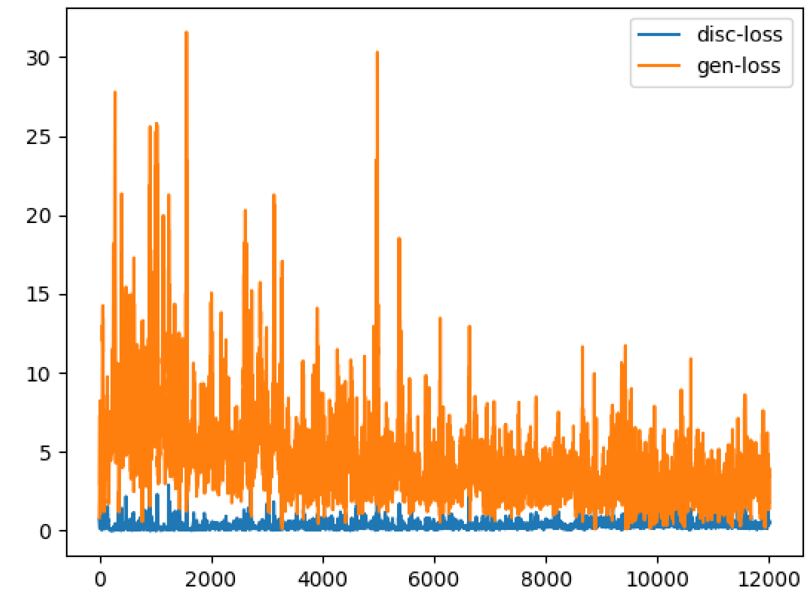
Run No. 31:

3 blocks with deep layers (the ones that are in the .ipynb and not commented out)

Parameters: epoch=500, z-dim=64, batch size=256, lr=0.001






disc-loss
gen-loss
0
5
10
15
20
25
30
0
2000
4000
6000
8000
10000
12000