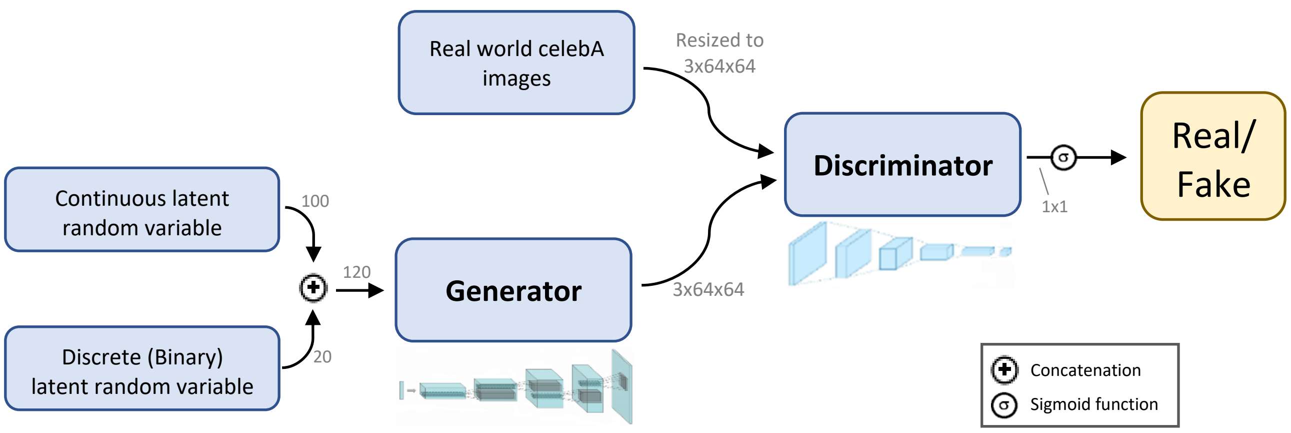
DL 097200 – HW3 Report

Group Members: Yotam Martin, ID: 308044296 || Gal Goldstein, ID: 204403174

1. Model architecture description and illustration:
   Our inspiration to the model architecture came mainly from this paper. Yet, we made some changes from the article as we will discuss soon.

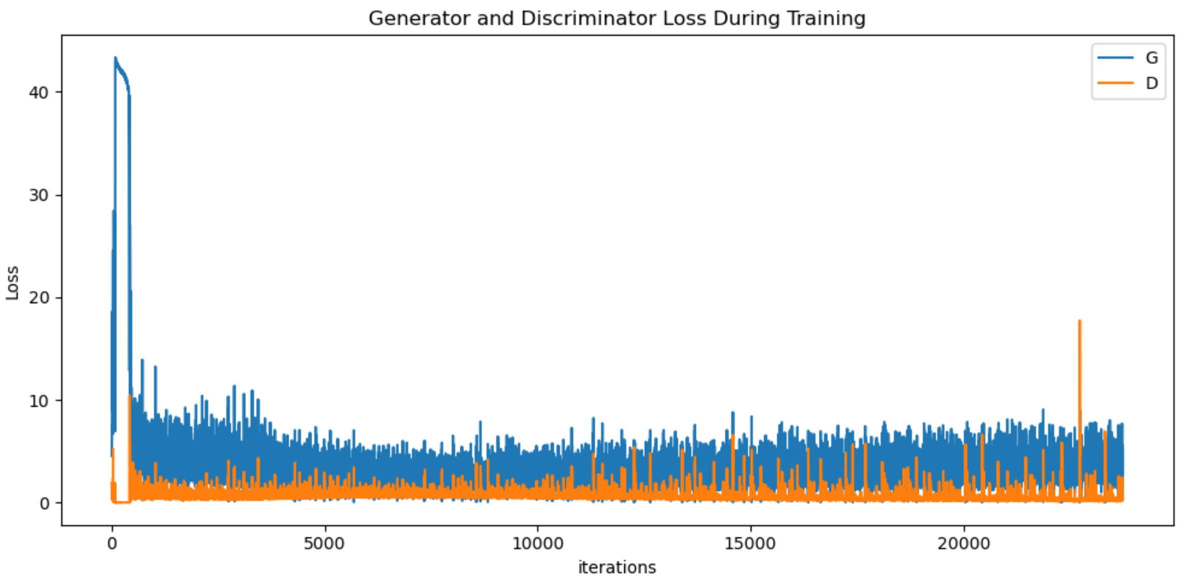


- The original aligned images size is 3x178x218. For each real image we did a preprocess that includes resizing to 3x64x64, normalizing values to [-1,1] range and covert to tensors.
- The fake images creation starts at defining two latent random variables: one is a continuous one – sampling of 100 normal standard variables, and the second is binary – sampling of 20 binary variables. Then, we concatenated them to one latent variable with length of 120. The latent vector (aka "Z") goes into the generator which is simply consist of convolutional-transpose layers (known also as de-convolution layer). Given an input vector Z the generator produces an output image with dimensions of 3x64x64 tensor, same size as the resized real images.
- Now, the fake and real images are ready to get into the discriminator. Alternately, batches of real images and of fake images will enter the discriminator which is simply consist of convolutional layers. Its input is 3x64x64 tensor and the output is scalar, which then pass-through Sigmoid function to get the probability of each image to be real or fake.
- As we saw in class, the training is separate for the generator and the discriminator. First the discriminator is trained by the real and fake images, while the generator weights are fixed. Then, the discriminator weights are fixed, and the generator is trained – generating fake images and try to fool the discriminator.

Training procedure – chosen hyper parameters:
We tried many configurations with different values for the hyper parameters. Our choices were influenced by the desire to achieve D(x) probability that get close to 0.5 (as we saw in class), but also from our own opinion on how real the fake images look. We will show the chosen values:
Loss function: Binary cross entropy - this function fit to our case where we want to compare two binary vectors of labels and predictions.
Images resize: 3x64x64. We found this size as large enough to catch the humans characteristics but still small enough so we can be able to train on ~400K real and fake images. This size is the new size of the real images, as well as the output size of the generator which is the fake images size. We tried few runs with size of 3x128x128, and we did not see any improvement.
Optimizer: Adam, **learning rate: 0.0002**.
Batch size: 128.
Epochs: 15.

Training procedure – optimization further details:
Images handling: To save run time, we resized all images to the same size, re-saved them as tensors (.pt files) instead of images (.jpg files), place them on our own machine disk, and then moved them to memory (RAM) for training.

2. Discriminator and generator losses plots:

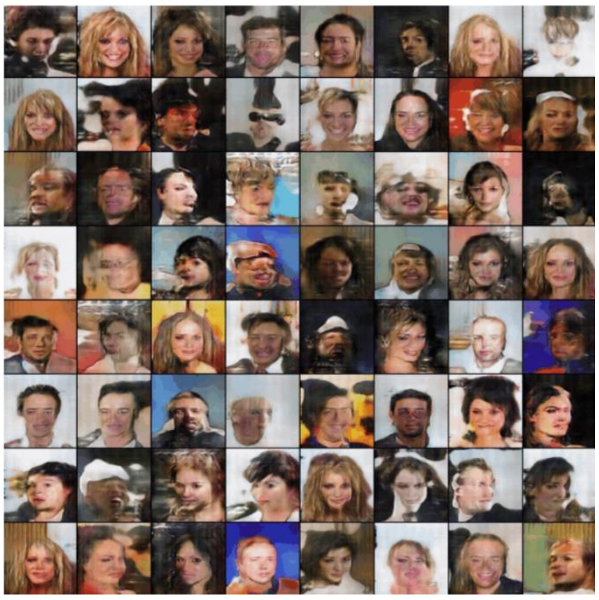


3. Summary of attempts & Visualizations:

We will discuss about attempts to bring insightful visualizations:

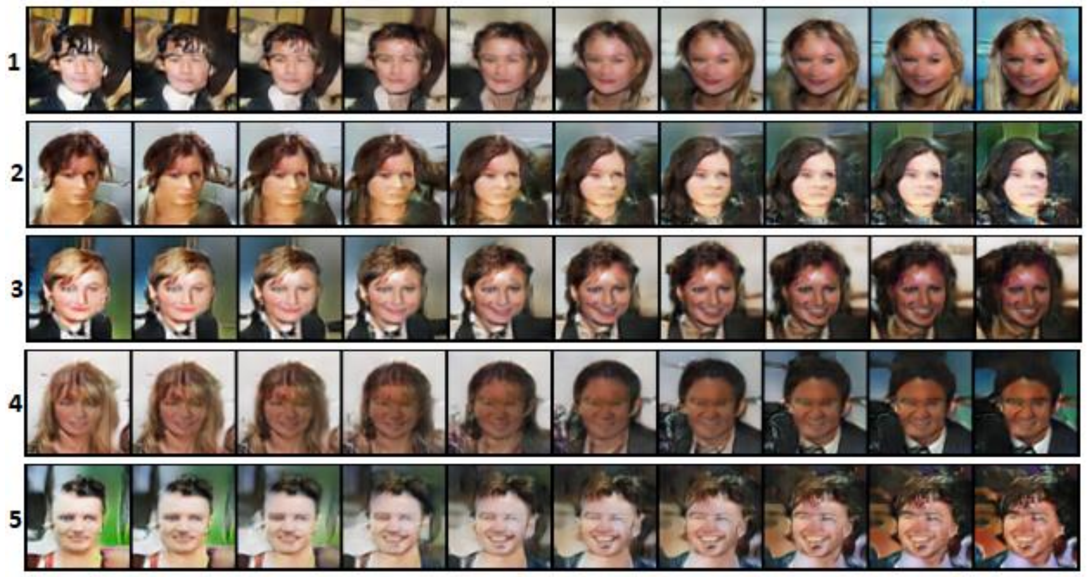
- Gradual change of the discrete latent vector:

  To find the influence of the discrete part in the latent vector, we took our final model weights, and generate one fixed randomly continuous vector (of length 100).
  Then we created 21 different discrete vectors (each of length 20) such that the i'th vector contain $\boldsymbol{i}$ 1's and then $\mathbf{20} - \boldsymbol{i}$ 0's. i between 0 to 20 (including). For example, here is the vector for i=2 [1, 1, 0, …, 0, 0, 0, 0]. For clarification: the first discrete vector (i=0) is only zeros and the last discrete vector (i=20) is only ones.
  Each discrete vector is concatenated to the fixed continuous vector.
  Then we generate 21 images, for the 21 different discrete vectors.
  We did it 64 times (64 different faces) and collect it to gif, that shows the influence of adding another "1" to the discrete vector.
  Discrete values change GIF (click)

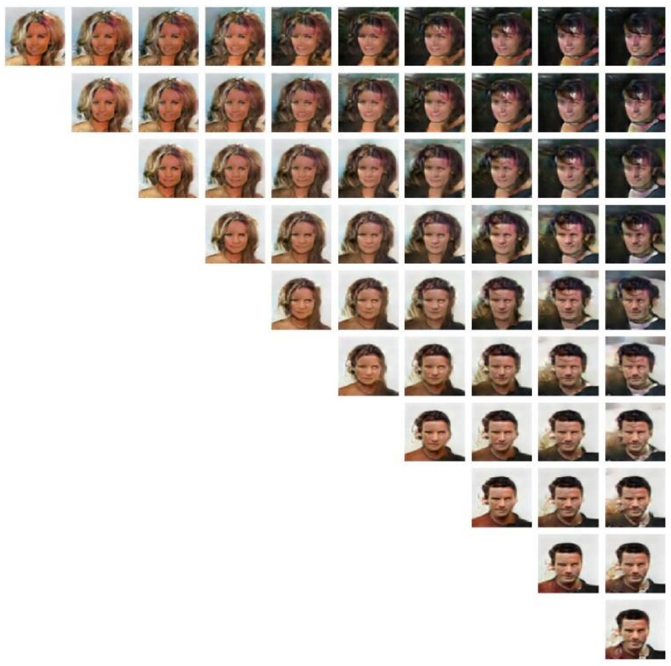


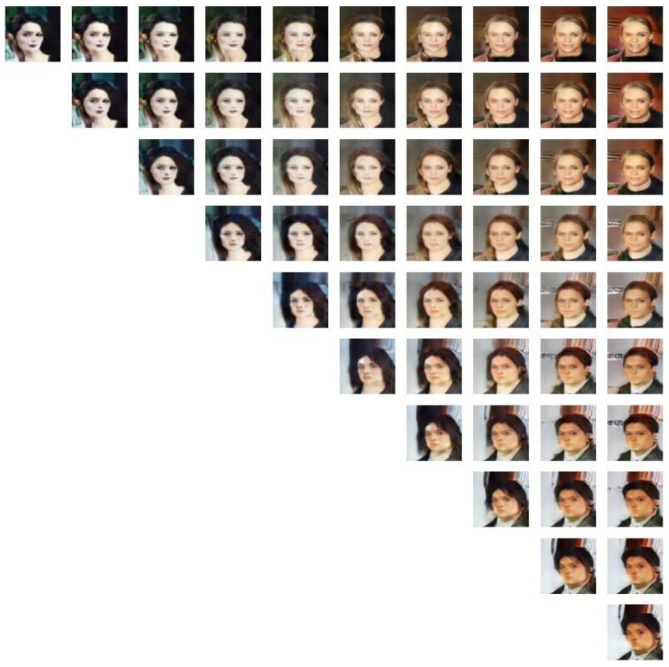
  As can be seen, the images change only slightly from one to the other. We think this means the discrete part of the latent vector does not encode any interesting and significant characteristic of the human face. By freezing the continuous part and explore the discrete part we found that the discrete part does not donate any significant improvement to the fake images' quality.

- <u>2 and 3 images interpolation:</u>
  We created a function that shows smooth passing from one fake image to another one. **This method helps to understand the latent vectors space**. The images at the edges are generated from two random latent vectors (with continuous and discrete parts, as detailed above), and the other images are generated from latent vector which is linear interpolation between the latent vectors of the two images at the edges.

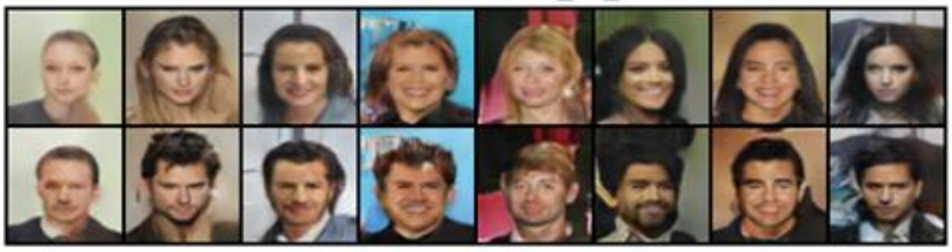


Then we upgraded the function and created the interpolation between 3 different fake images.

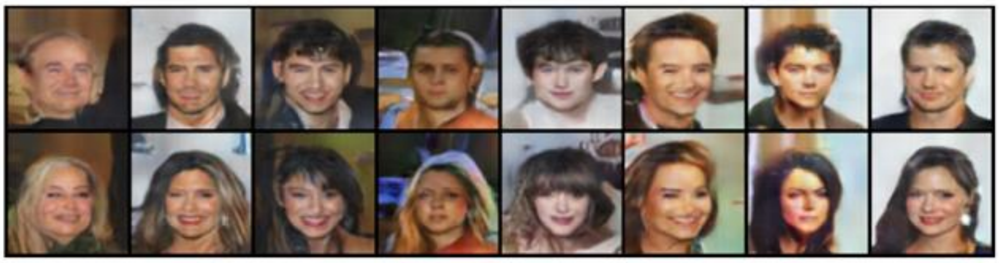


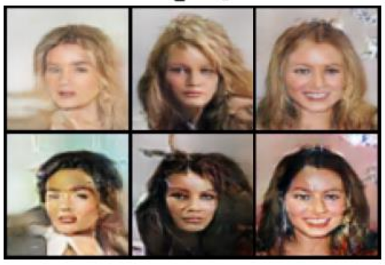


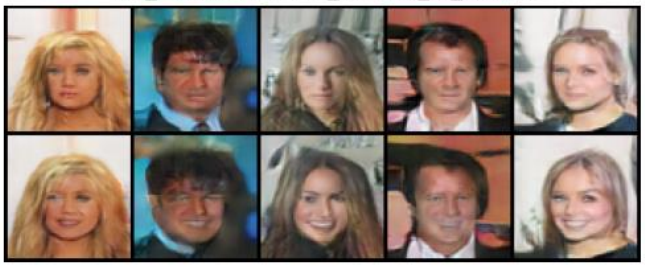
- <u>Vector arithmetic:</u>
  The field of vector arithmetic says that if we will take a vector that we believe to represent one group and subtract second vector which we believe to represent second group, we should get difference vector which supposed to capture the difference between groups.
  So, we started by taking 10 fake images that clearly look like men and another 10 that look like women. We averaged the latent vectors of the first 10 and got our belief to vector that represents the group of men, same for the group of women. Now with the help of vector arithmetic, we found the difference vector "men-women". Therefore, if we will add it to a latent vector of fake image that look like woman, we will get that image in "man version".



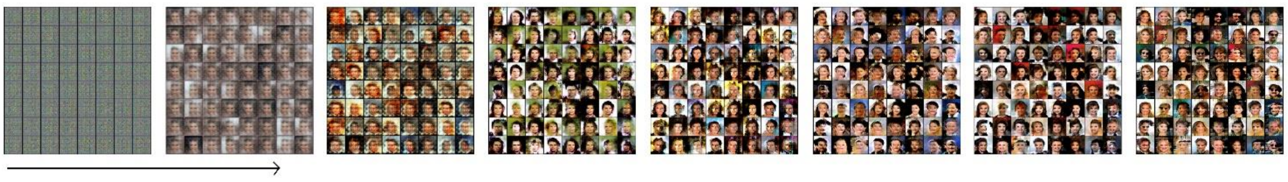
  This worked well. We can see in the first row the fake images we chose, and in the second row the transformation we just detailed, to the second gender.
  Showing below some more vector arithmetic:



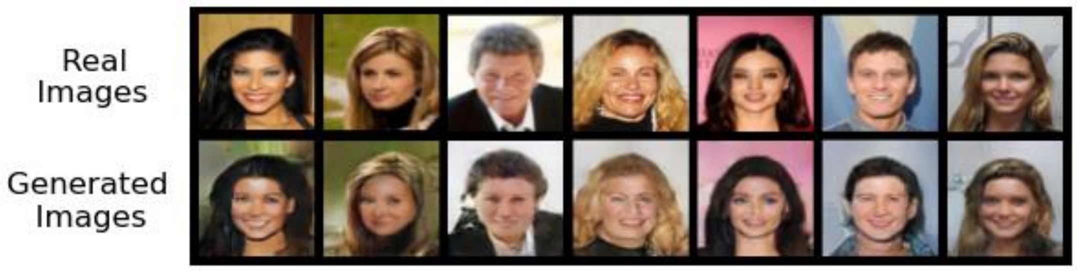




- <u>Generator model progression:</u>
  As before, we took fixed 64 latent vectors, and checked the generator output on them, along the training, from the first epoch until the last one. This gif shows the progression in the generator learning, from no learning to initial general facial features, until the output of our final model.
  [Generator progression GIF (click)]
  Some images from the GIF:



- <u>Reverse Generator:</u>
  Not like in VAE, in GAN there is no latent vector representation to the real images. The latent vectors are used for generating the fake images. We wanted to find a way to represent a real image with latent vector, since we found it as useful for more creative ideas. Inspired by [this paper], we created the reverse generator that given input of real image, it can find vector which is approximation to representative latent vector. This is done by the SGD on the MSE objective function: $\underset{z}{argmin}||G(z) - Image||_2^2$. Where G is our trained generator and "Image" is the image we try to find its latent vector.
  To check the quality of this estimated latent vector, we pass the approximated z found through the generator to produce a fake image (2nd row). The results look like the real images, but not identical at all.



<u>Conclusions:</u>
- The discrete part of the latent vector is much less significant than the continuous part, in the task of learning human face characteristics. This does not surprise us as the discrete part has finite number of combinations, and the continuous has infinite number of combinations (theoretically). In addition, the sizes we chose for each part may be a factor as well.
- The vector arithmetic performed very well.
- The interpolation shed light on the latent vector space and showed the smooth transition between two/three fake images.