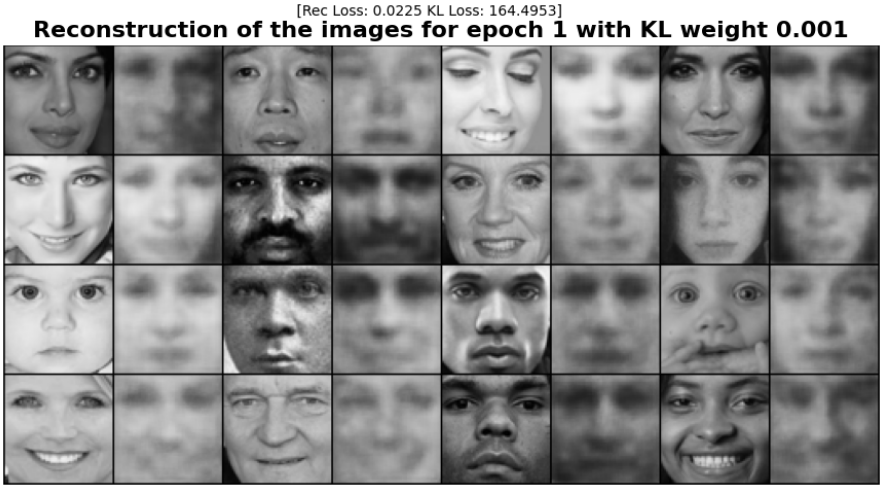
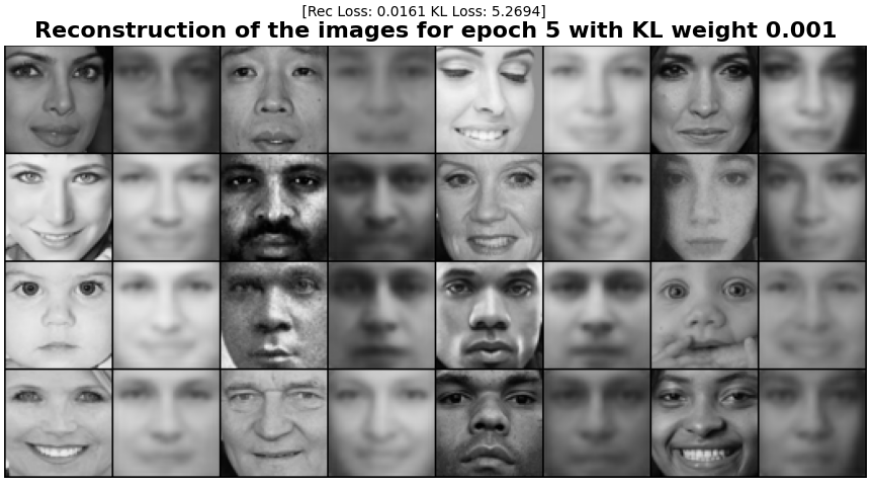
The aim of this lab is to get into Deep Generative Models, especially with Variational Autoencoders (VAE) and Generative Adversarial Networks (GAN). We will create these models and train them with 50000 extracted images from the Cohn-Kanade Database. These images have a size of 32x32 pixels and are gray-scale. They consist of human faces with different emotions.

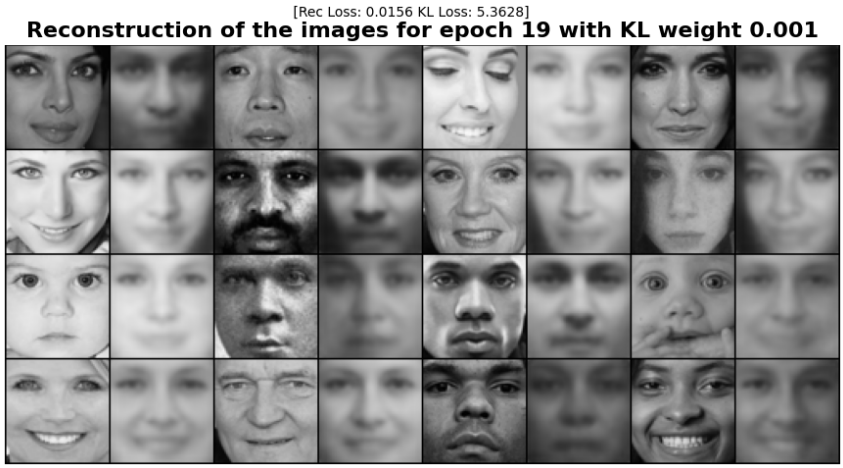
1. Variational Autoencoders

Following the example of the MNIST, we train a VAE with the same architecture but the latent space has 32 dimensions. We use the same optimizer, which is the Adam with a learning rate of 0.001 and a weight decay of 1e-5.

Initially, we set a weight of 0.001 for the Kulback Leibler divergence and we get the following outputs. The images below consist of the reconstruction of certain inputs (reference images are at the odd columns and their respective reconstructed image to its right).

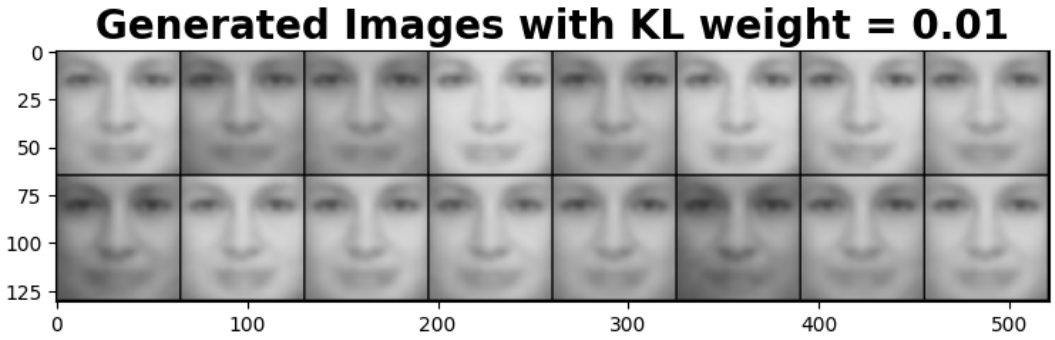




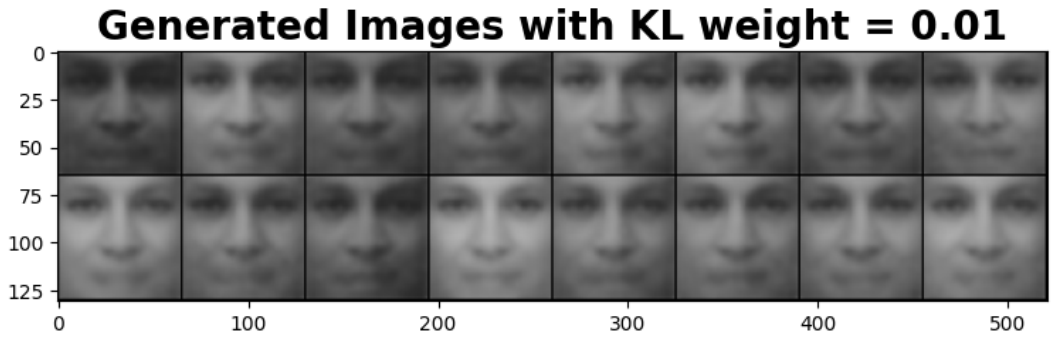


Despite having a low reconstruction loss, one can infer that the reconstructed images are far away to be compared to its respective reference. This is mainly due to the weight given to the KL divergence that is making an overly regularized latent space. This can be seen at the generation with random noise when the network was trained with KL weight 0.01:

Mean = -1 and Variance = 1:



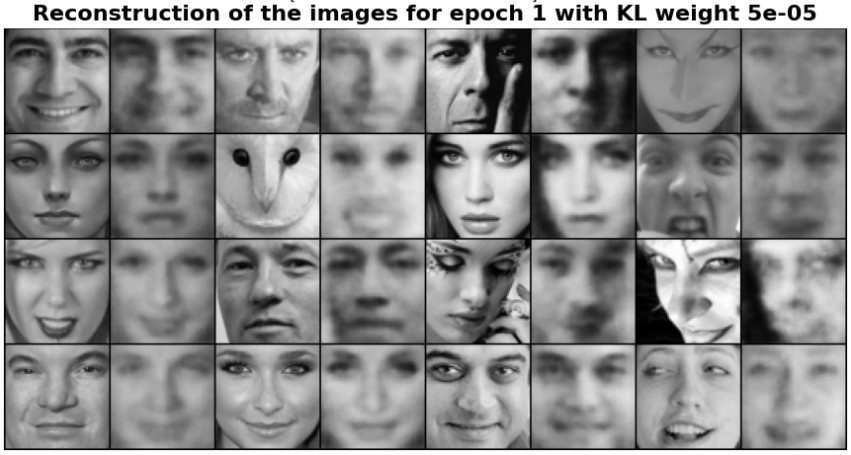
Mean = 1 and Variance = 1:



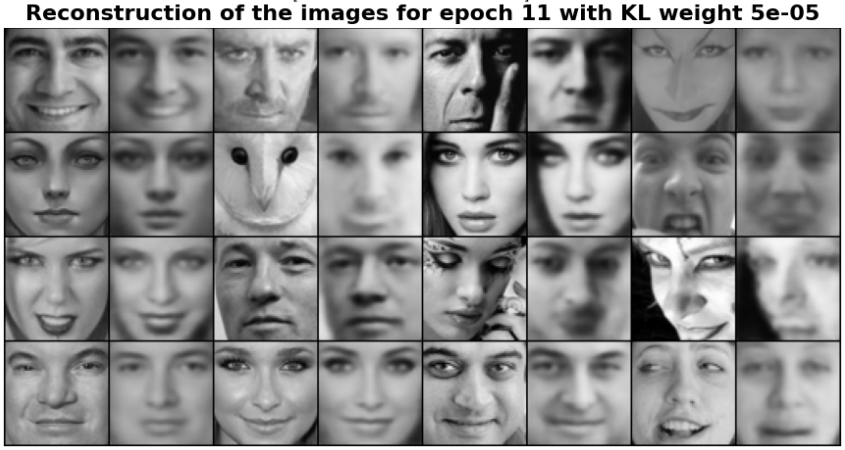
Consequently the interpolation across this latent space is only a change from a darker tone to a lighter one.

Therefore, in order to extract meaningful features and having a well structured latent space we decided to use a KL weight of 0.00005 and the outputs are the following:

[Rec Loss: 0.0143 KL Loss: 89.4859]
Reconstruction of the images for epoch 1 with KL weight 5e-05

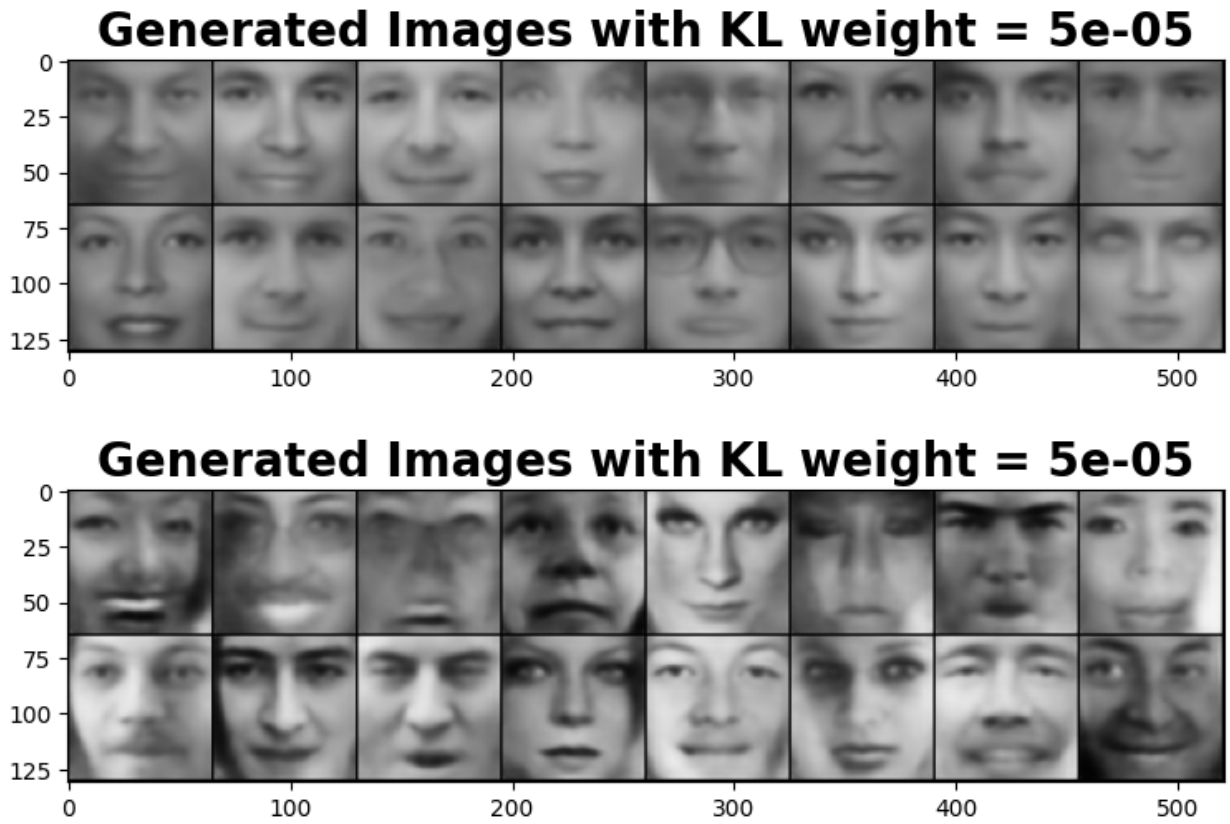


[Rec Loss: 0.0065 KL Loss: 43.758]
Reconstruction of the images for epoch 11 with KL weight 5e-05



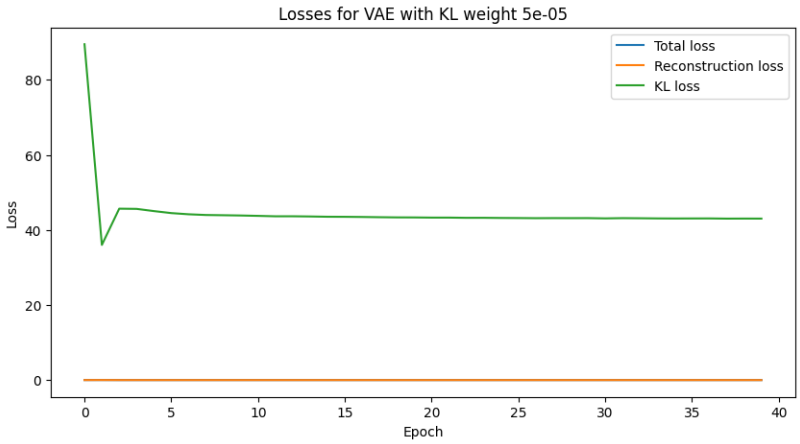
[Rec Loss: 0.0063 KL Loss: 43.3514]
Reconstruction of the images for epoch 19 with KL weight 5e-05

Now the reconstruction is more faithful but one could think that the resulting latent space is not regularized and not structured, meaning that two close points do not need to be similar. For that reason we sample with low variance and a random mean that follows a uniform distribution, in other words: $\mu_i \sim U(-1, 1)$ for $i = 1,\ldots, 32$. Two random outputs with variance 0.05:

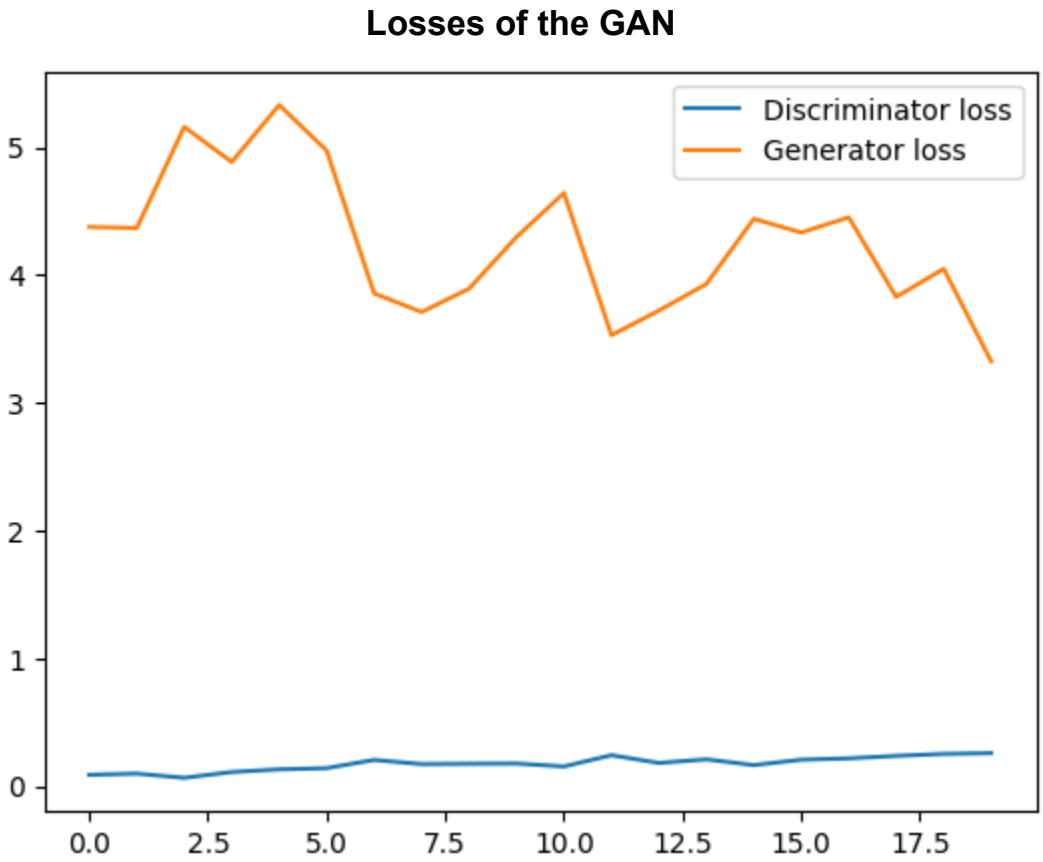


Some images are similar, however, it is easy to see that there are samples that could be considered outliers since they do not bear any resemblance to its neighbors.

The loss plot associated with this model is:



2. Generative Adversarial Network



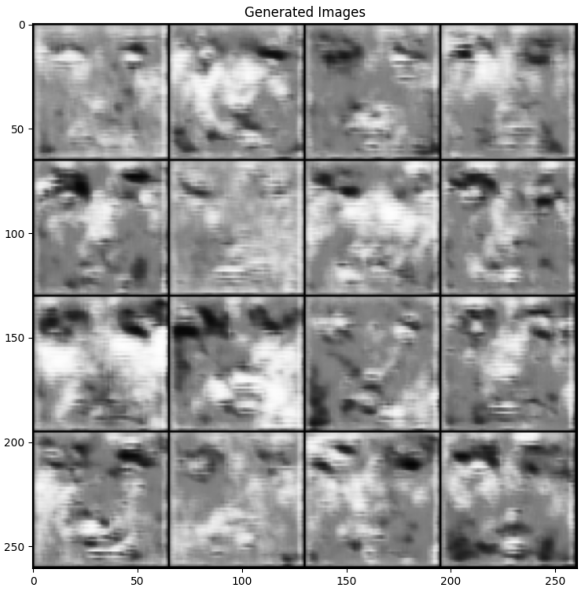
In this first plot we can see how the losses (Discriminator loss and Generator loss) behave during the network training. The particular technique employed in this specific implementation is to alternatively train the discriminator and the generator at each batch. One thing we noticed is that since we were alternating the training role at each batch and we have a pair number of batches, we will always be training each of the networks with the same batches. We took this into account to avoid overfitting in each of the networks and to exploit the full potential of the data

Our main objective is to reduce de generator loss (this entails more realistic images), which is done by having an accurate discriminator to keep it from reaching a steady (good enough) loss. In our particular example, we are given an implementation where the discriminator loss is factored by ½ which should urge the generator to improve itself more drastically.

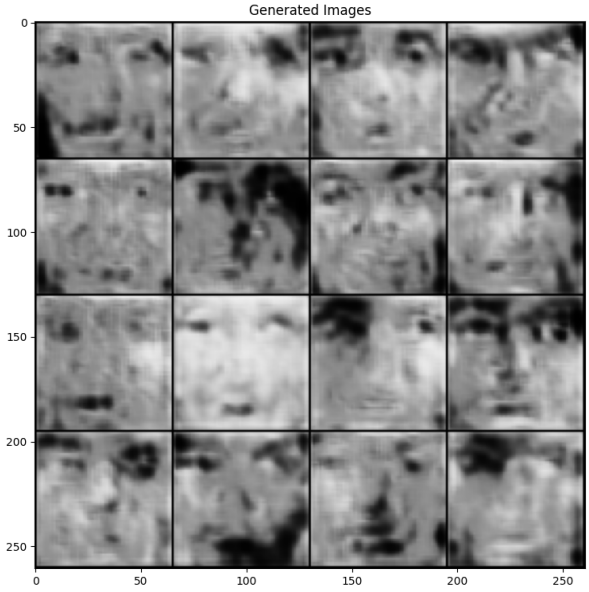
Now it is time to see how our Generation of image improves in optimization and how it is related to both losses.

Image generation in different epochs

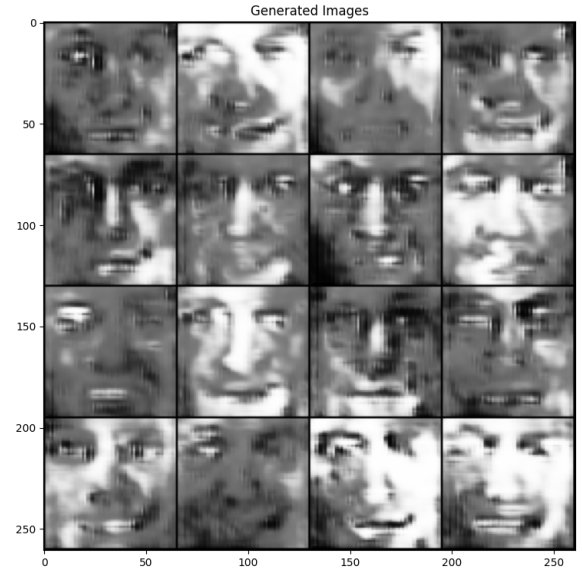
Epoch 1 Gen. Loss: 4.3769, Disc Loss: 0.0884



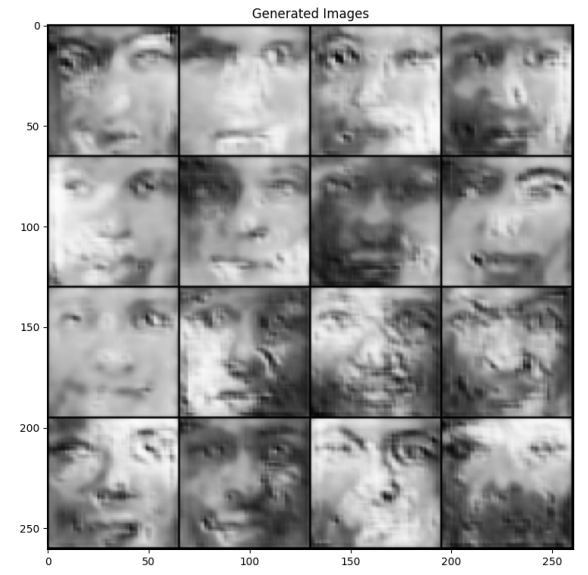
Epoch 3 Gen. Loss: 5.1603, Disc Loss: 0.0648



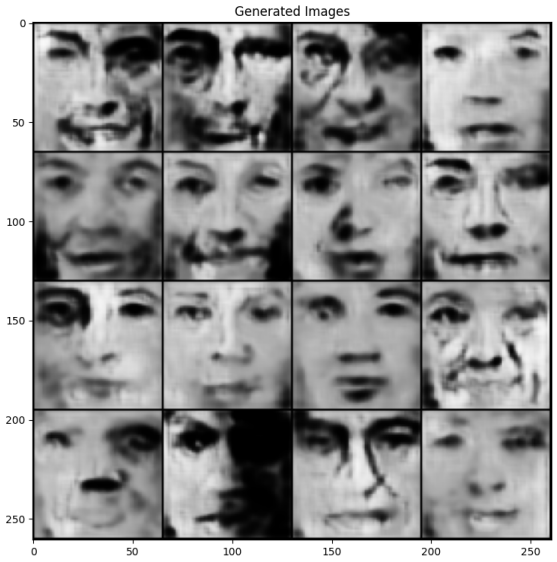
Epoch 5 Gen. Loss: 5.3317, Disc Loss: 0.1312



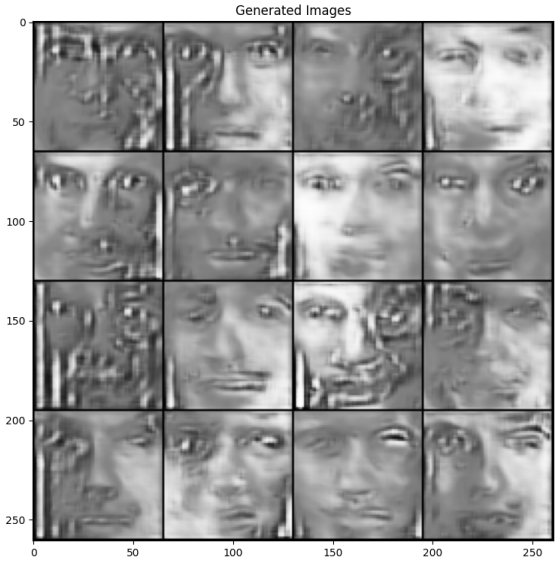
Epoch 7 Gen. Loss: 3.8572, Disc Loss: 0.2049



Epoch 13 Gen. Loss: 3.7225, Disc Loss: 0.1812

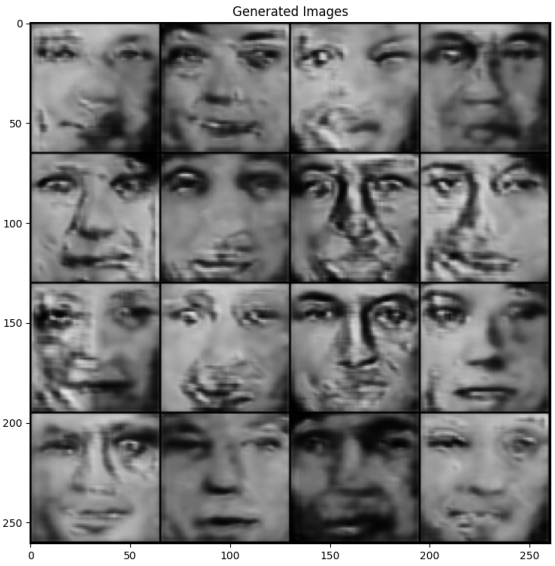


Epoch 15 Gen. Loss: 4.4405, Disc Loss: 0.1642



Epoch 17 Gen. Loss: 4.4527, Disc Loss: 0.2171

Epoch 19 Gen. Loss: 4.0494, Disc Loss: 0.2520



The key aspect of this progression is the interplay between both losses. We see how the most realistic images are also the ones with the lowest loss, as is shown by epoch 13. The rest of the changes seem to come as a result of random variation in light intensity, texture or blurriness which in most cases doesn't deter the discriminator from predicting correctly.

As a matter of fact, we believe the exceptional performance of the discriminator is leading to the generator not finding where to improve. Using an analogy to explain this

behavior, if we were taking exams much more complicated than what we studied for, it would be difficult to know where to start improving on since we would be getting consistent low marks. This works similarly in GAN's. If either of the networks gets significantly better than the other, it breaks the symbiotic system.

We can clearly see that the Variational Autoencoder has a much better and more neat performance that it's counterpart. This is most likely due to VAEs being better at picking up structured distributions of the data, which is crucial for the problem at hand; reproducing faces.

Furthermore, VAEs utilize an encoder-decoder architecture that can effectively learn meaningful representations of facial expressions. The encoder part of the VAE captures the essential features and latent factors underlying the CK faces dataset. By learning a compressed representation of the input data, the encoder enables efficient storage and retrieval of facial expression information. The decoder part reconstructs the input data from the learned latent representation, facilitating the generation of realistic facial expressions.

On the other hand the GANs are not at all well-suited to capture complex data distributions or to generate diverse facial expressions, which can also be seen by the uniform output of our generator.

Future steps in GAN training

The factor we mentioned earlier on the discriminator loss is reflected in it being very low at all stages and the generator not really coming close to it. That is when we considered employing another alternating protocol. We did so by, on the one hand, getting rid of the discriminator loss factor while having a more thought-out condition to alternate between which network to train.

Our idea was to implement a process by which one of the networks stopped training once it reached a certain threshold we deemed reasonable. This threshold would be different in both losses since one (specifically the discriminator) converges to a lower value than the other. By doing so, we would be exclusively training one network until "perfected", at which point the other would train to try to counter the improvements of its adversary.

Unfortunately, due to exams, we really procrastinated and weren't able to train it (rather, we started training and it was taking too long). We added a section at the bottom of our code called **alternative training** expressing these changes.