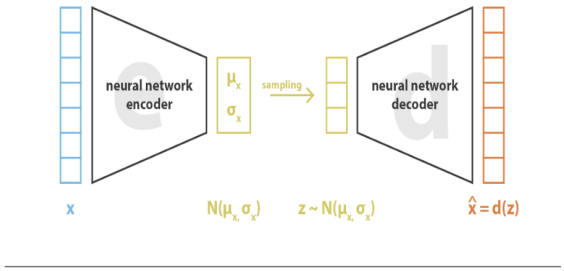
Francesca Mees, NIA: 230849
Arne Berresheim, NIA: 230829

Practical session 4:

Deep Generative Models: Variational Autoencoders and Generative Adversarial Networks

1. Exercise

The CK dataset contains gray-scale images of human faces in .mat format. Variational Autoencoders are designed such that the input data is sampled from a parametrized distribution, and the encoder and decoder are trained jointly such that the output minimizes a reconstruction error.



$$\text{loss} = \|x - \hat{x}\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

Variational Autoencoder Illustration

First, we defined the encoder and decoder networks.

1. The encoder is composed of 3 Conv-BN-ReLu blocks and a fully-connected layer.
2. The decoder is composed of a fully-connected layer and 3 BN-ReLu-Conv blocks.

Then, we use the Encoder and Decoder modules to create the VAE class.

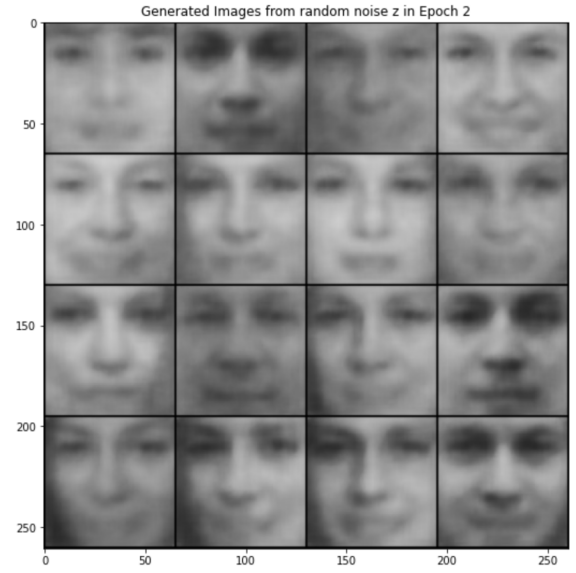
```
CK VAE Definition
VAE(
  (encoder): Encoder(
    (layer1): ConvBNReLU(
      (conv): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (bn): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
    )
    (layer2): ConvBNReLU(
      (conv): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (bn): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
    )
    (layer3): ConvBNReLU(
      (conv): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (pool): AvgPool2d(kernel_size=2, stride=2, padding=0)
    )
    (fc): Linear(in_features=16384, out_features=128, bias=True)
  )
  (decoder): Decoder(
    (fc): Linear(in_features=64, out_features=16384, bias=True)
    (layer3): BNReLUConv(
      (bn): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (pool): UpsamplingNearest2d(scale_factor=2.0, mode=nearest)
    )
    (layer2): BNReLUConv(
      (bn): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv): Conv2d(32, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (pool): UpsamplingNearest2d(scale_factor=2.0, mode=nearest)
    )
    (layer1): BNReLUConv(
      (bn): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv): Conv2d(16, 1, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    )
  )
)
```

Francesca Mees, NIA: 230849
Arne Berresheim, NIA: 230829

After defining the train function for the VAE, we proceeded with training the VAE on CK. We used the Adma optimizer which is typically used in VAEs.

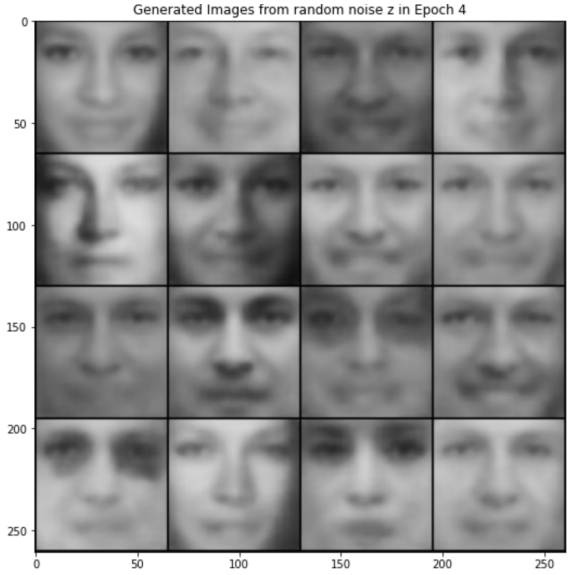
Epoch 2

```
Epoch [2/20], Step [196/196], Rec. Loss: 0.0175, KL Loss: 8.1378
```



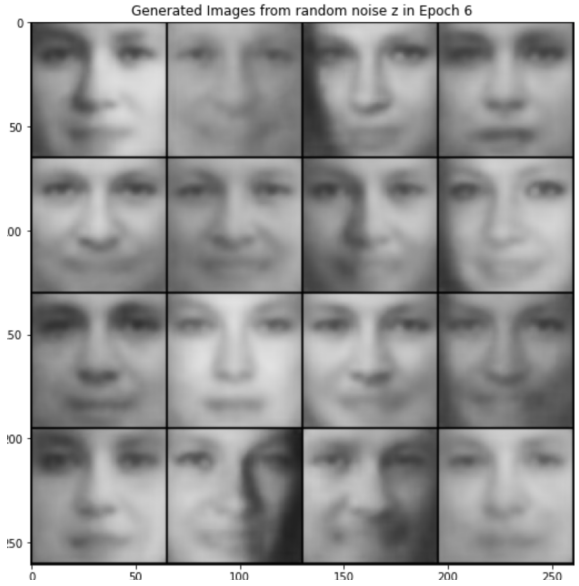
Epoch 4

```
Epoch [4/20], Step [196/196], Rec. Loss: 0.0166, KL Loss: 21.7185
```



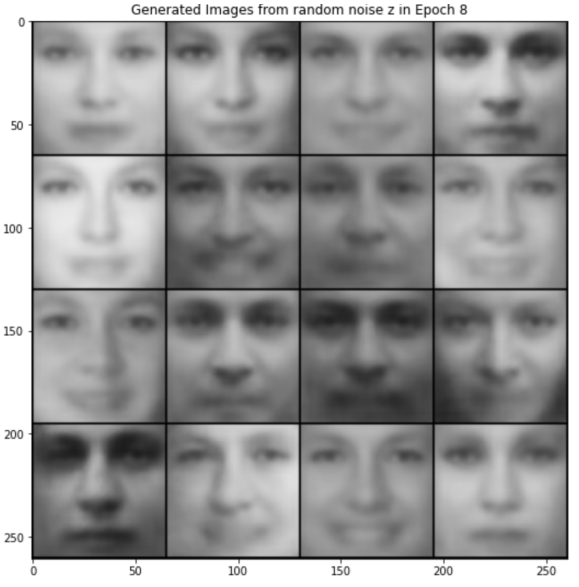
Epoch 6

```
Epoch [6/20], Step [196/196], Rec. Loss: 0.0165, KL Loss: 11.6402
```



Epoch 8

```
Epoch [8/20], Step [196/196], Rec. Loss: 0.0164, KL Loss: 14.5125
```

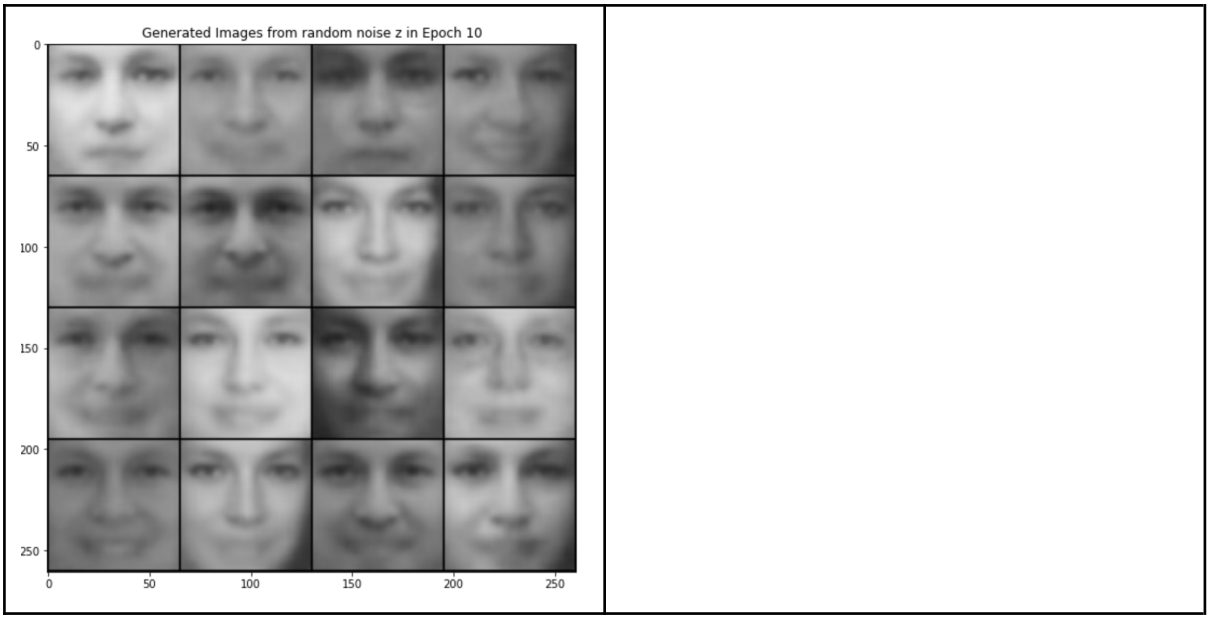


Epoch 10

```
Epoch [10/20], Step [196/196], Rec. Loss: 0.0163, KL Loss: 6.1432
```

Epoch 12

Francesca Mees, NIA: 230849
Arne Berresheim, NIA: 230829



As we reached the output size limit we were not able to print all face clusters in the notebook. The rest of the interpolations can be found in the results folder with the naming convention 'vae_interpolation_ck_EX', where X is the corresponding epoch.

We can clearly see a big improvement in the generated faces after some Epochs. The results become very close to real pictures, although they are still quite blurry. In the first two Epochs there is a big decline in the KL loss as the desired distribution is adjusted rapidly. In Epoch 4 the KL loss is comparably high as we can see this in the interpolations as well, when compared with later Epochs. Also in Epoch 4 the reconstruction loss had an above average decline. This is because between the reconstruction and the KL-loss there is a tradeoff, as this is a min-max problem, in which we are optimizing two functions. Generally the KL-loss does not have a steady decline as compared to the KL-loss.

2. Exercise

In the following, we will train and evaluate GAN in CK dataset. Similar to VAEs, GANs also have a generator which receives a random vector $\mathbf{z} \sim N(0, I)$ to generate a synthetic image. However, instead of using an encoder , GANs are trained employing a binary classifier implemented with a neural network. Generator and discriminator losses are jointly optimized.

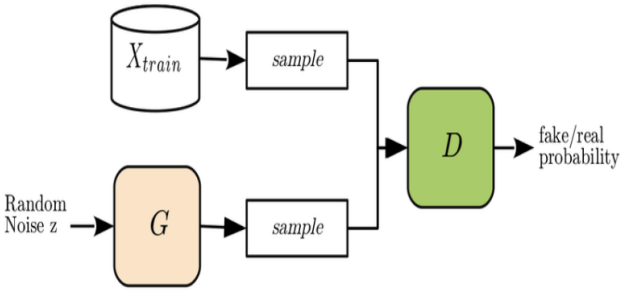


Image Caption

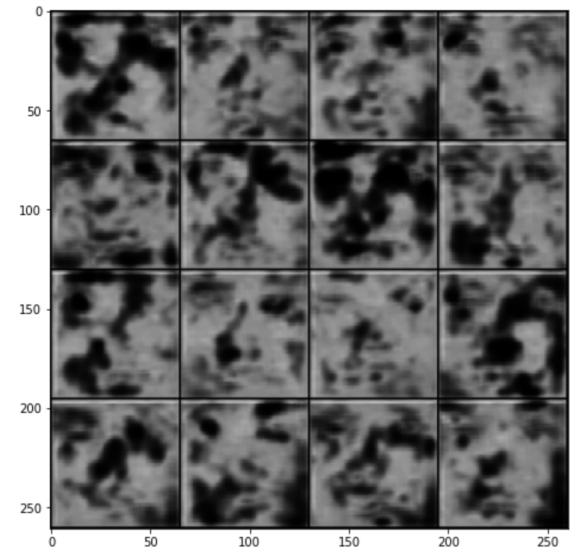
(1) The generator network will have the same definition as our VAE decoder.

Francesca Mees, NIA: 230849
Arne Berresheim, NIA: 230829

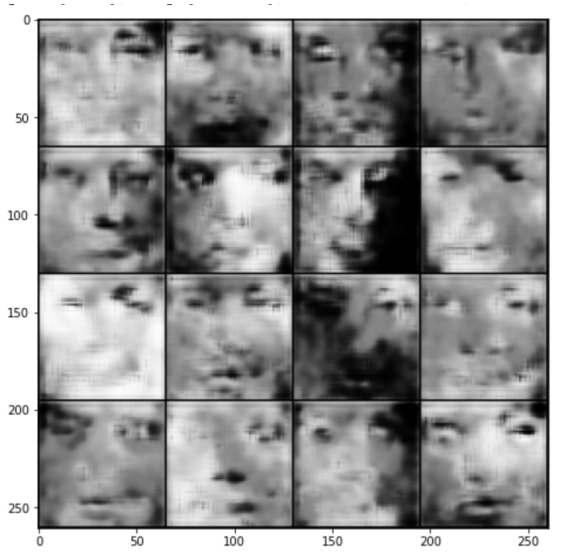
(2) The discriminator will have a similar definition than our VAE encoder. The difference is that the last fully-connected layer will be a binary classifier which will output a single value representing the probability of a fake/real image.

After defining the train function for the GAN, we proceeded with training the GAN on CK. We used the Adma optimizer which is typically used in GANs.

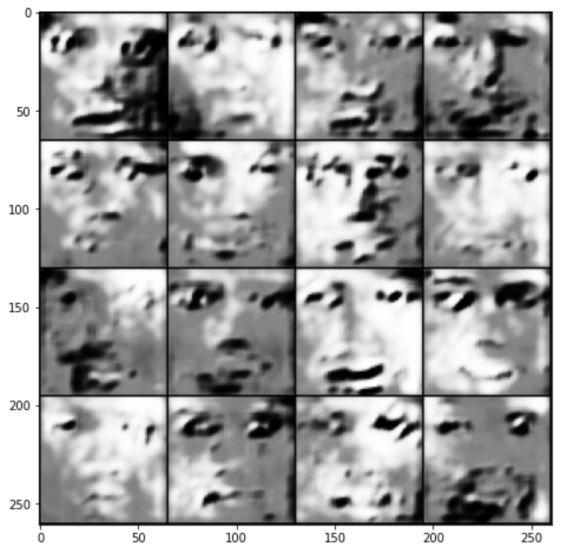
Epoch 2



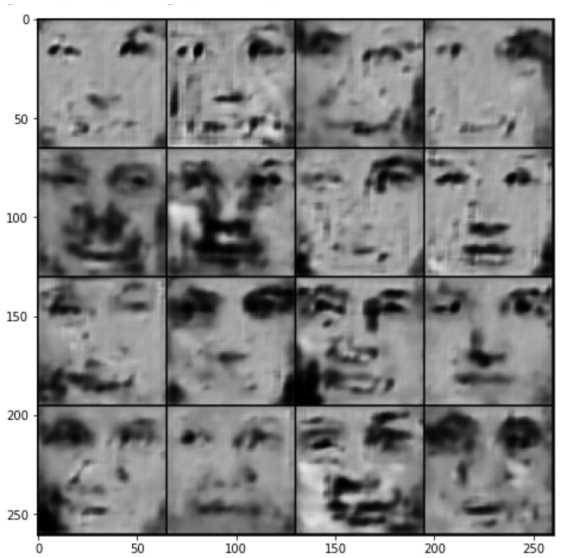
Epoch 4



Epoch 6



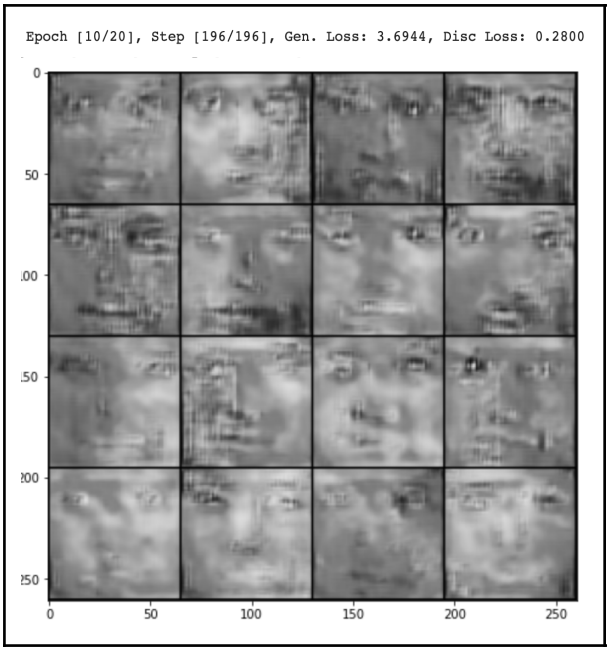
Epoch 8



Epoch 10

Epoch 12

Francesca Mees, NIA: 230849
Arne Berresheim, NIA: 230829



All the results with the interpolations can be found in the results folder with the naming convention ‘vae_interpolation_ck_EX’, where X is the corresponding epoch.

On the first look we can see that the results are a lot worse than the ones we obtained from the VAE. In the second Epoch there is only a quite arbitrary black and white result in which we can detect a nose in some of the images. We can see very clearly that again there is a tradeoff between the losses. In this case it is between the Generator and the Discriminator. In every Epoch the one goes down, the other goes up. After generating the edges of the face more detail in the eyes and nose is generated. This shows that the Generators weights are fighter optimized to imitate the distribution given in the dataset.

In the VAE the images seem more close to the training dataset. If we would give the GAN more Epochs to train the results would look better and most likely more arbitrary when compared with the VAE, as this is overfitting to the given training dataset.