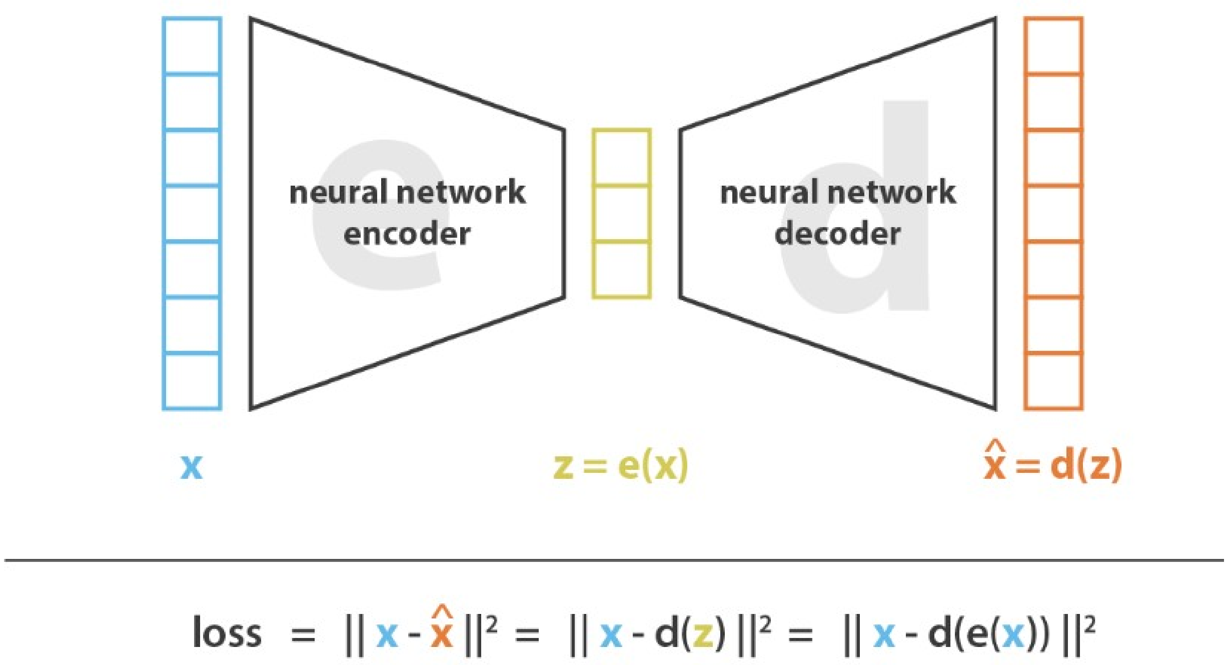
Deep Learning

TP2 : Generative models

SIA_TP1
TRAN Jeremy

1 : Autoencoder

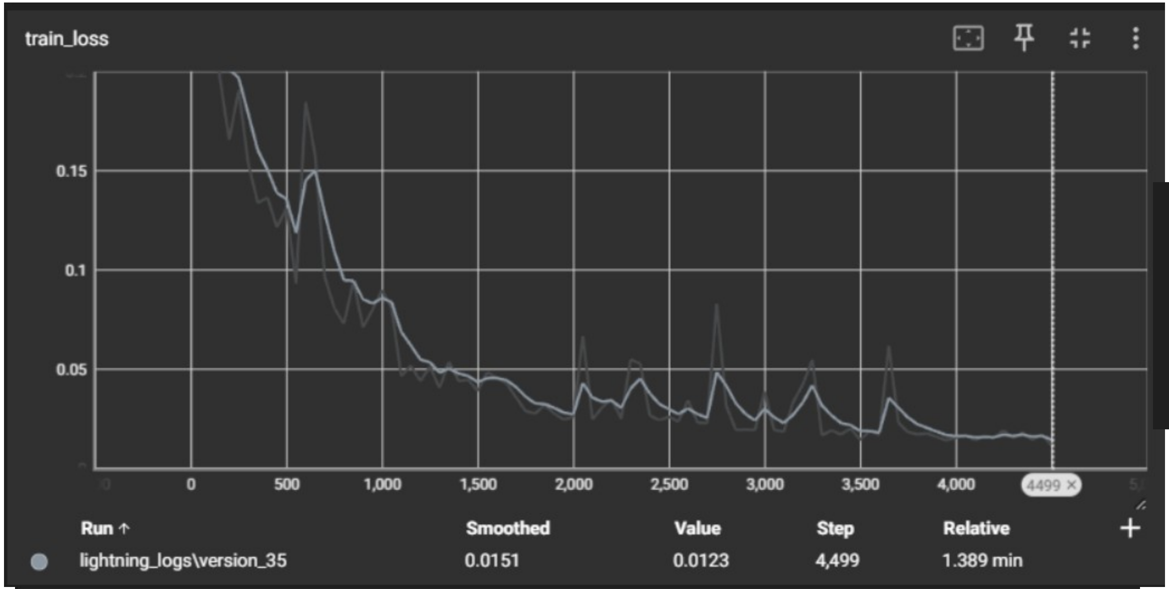

neural network encoder
neural network decoder
e
d
x
z = e(x)
$\hat{x} = d(z)$
$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$

The process is to compress the input then reconstruct it to learn its features :

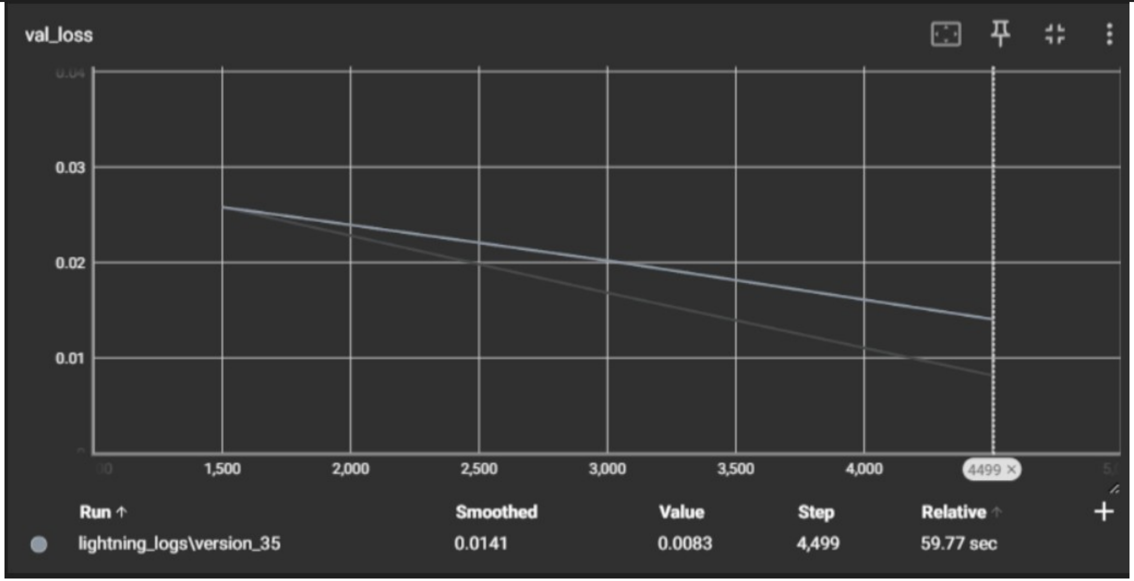
- The encoder compresses the data into a latent space, here using down-samplers, adding more of these layers creates a smaller latent space.

- The decodoer reconstructs the image from the latent space data using up-samplers.

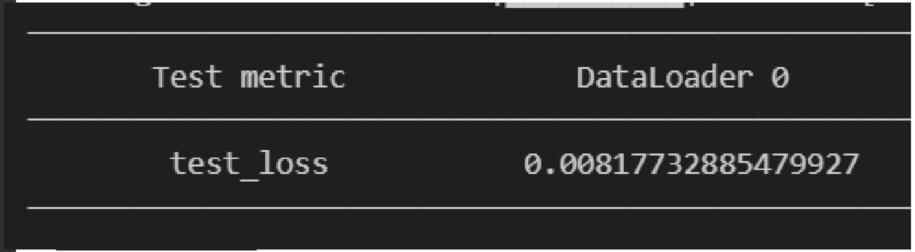
Some results :
Depth 3 AE, 3 epochs



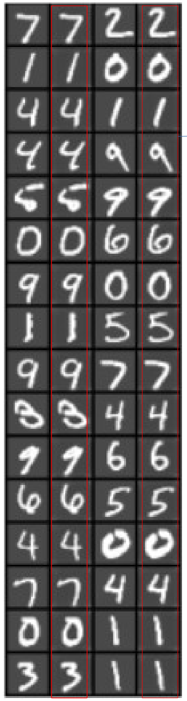
Run ↑	Smoothed	Value	Step	Relative
lightning_logs\version_35	0.0151	0.0123	4,499	1.389 min



Run ↑	Smoothed	Value	Step	Relative
lightning_logs\version_35	0.0141	0.0083	4,499	59.77 sec

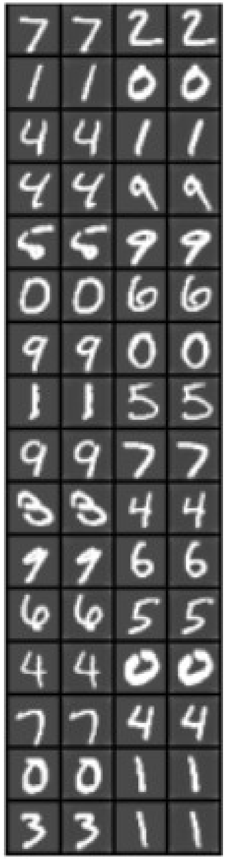


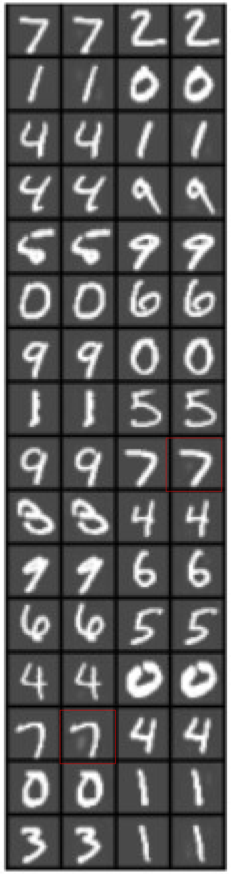
```
        Test metric              DataLoader 0

         test_loss           0.008177328854799927
```

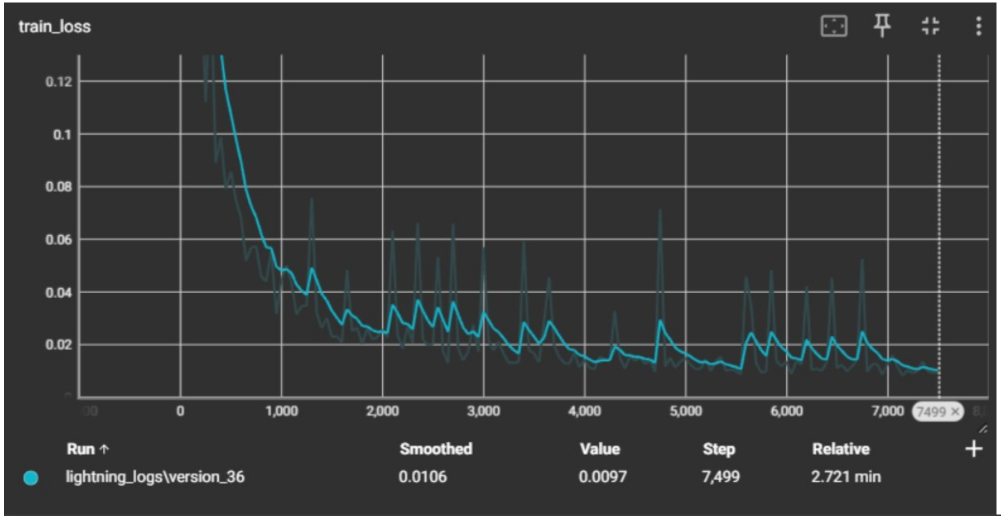


The resulting images are very close to the originals

Test using different training parameters :

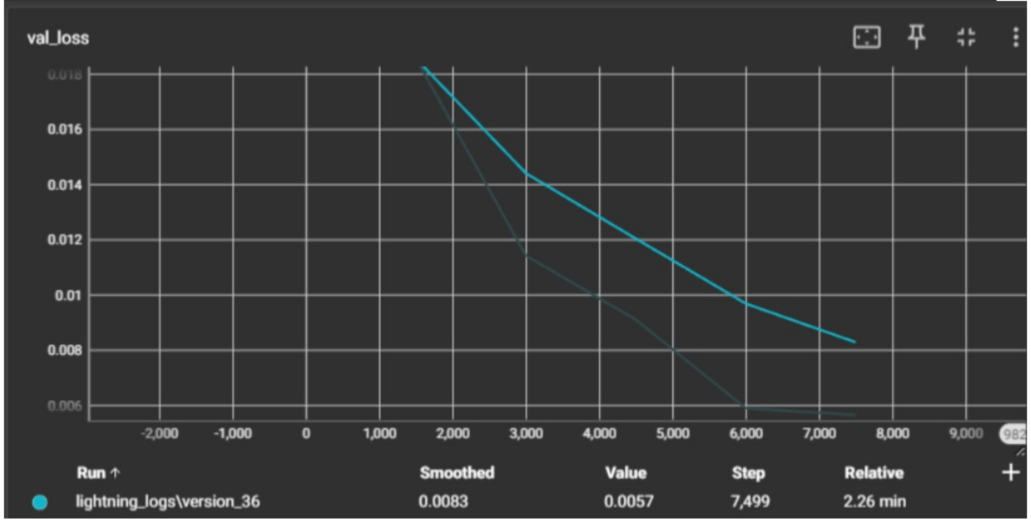






train_loss

Run ↑	Smoothed	Value	Step	Relative
● lightning_logs\version_36	0.0106	0.0097	7,499	2.721 min

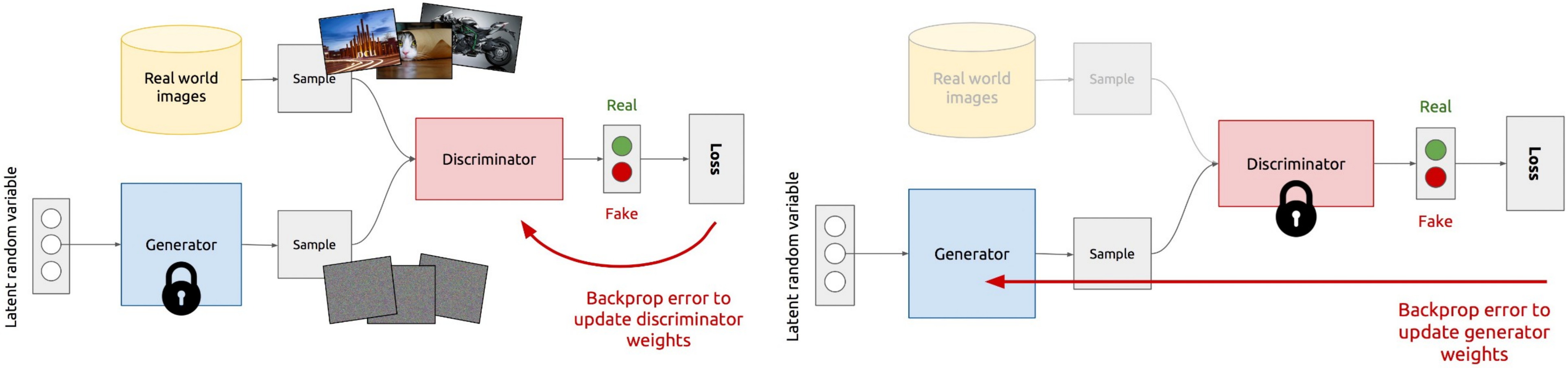


val_loss

Run ↑	Smoothed	Value	Step	Relative
● lightning_logs\version_36	0.0083	0.0057	7,499	2.26 min

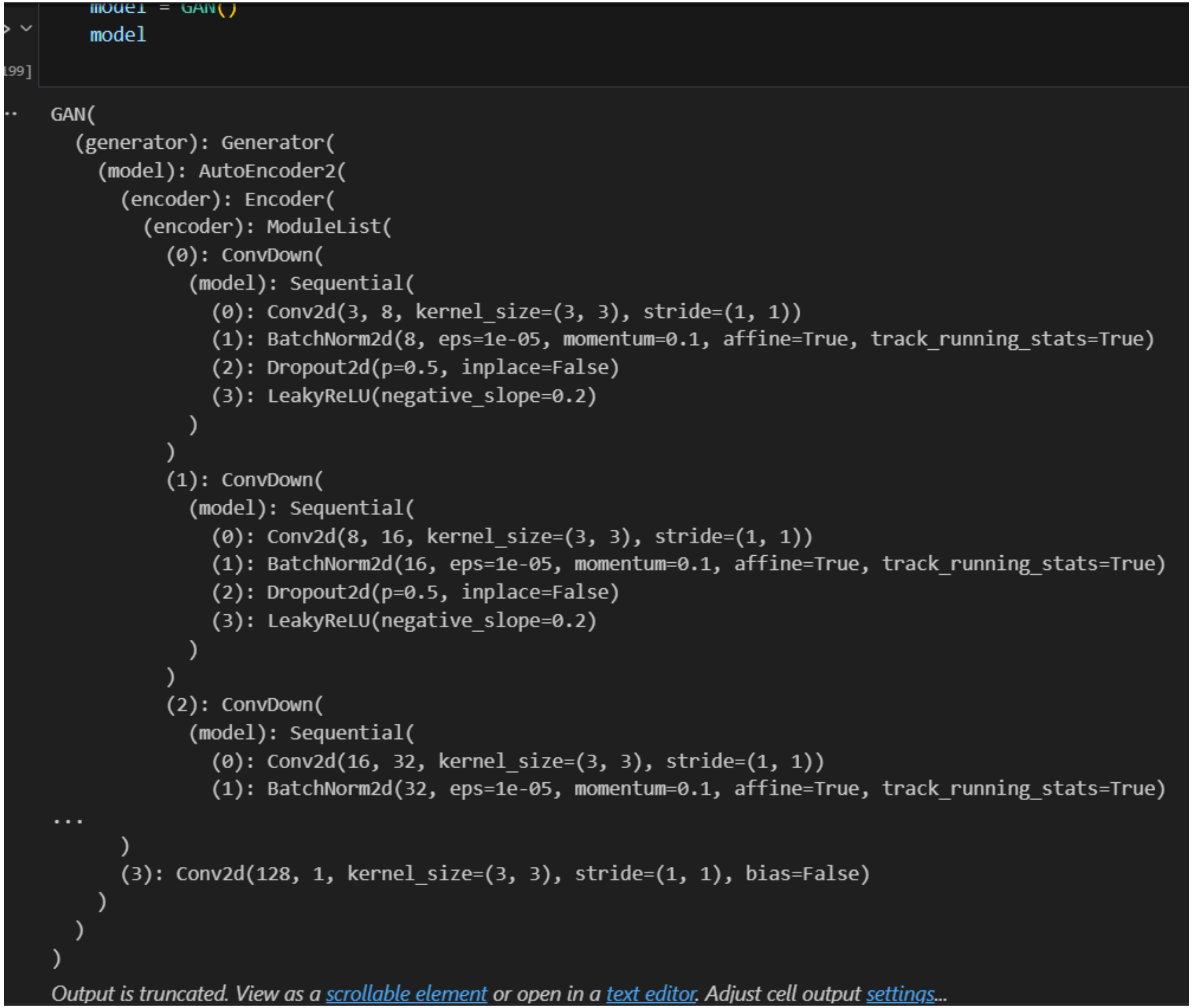
Here the depth is 5 and trained on 5 epochs.
There are visible defects in the image not present in the previous test.

2 : Generative adversarial network



The GAN model uses a generator to create fake images and a discriminator to distinguish between fake and real images.

- In the first step, the discriminator is trained to recognize real pictures
- In the second step, the generator that used to generate random images has its weights updated to output images that gradually get closer to the real images.



```
model = GAN()
model
```

```
GAN(
  (generator): Generator(
    (model): AutoEncoder2(
      (encoder): Encoder(
        (encoder): ModuleList(
          (0): ConvDown(
            (model): Sequential(
              (0): Conv2d(3, 8, kernel_size=(3, 3), stride=(1, 1))
              (1): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
              (2): Dropout2d(p=0.5, inplace=False)
              (3): LeakyReLU(negative_slope=0.2)
            )
          )
          (1): ConvDown(
            (model): Sequential(
              (0): Conv2d(8, 16, kernel_size=(3, 3), stride=(1, 1))
              (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
              (2): Dropout2d(p=0.5, inplace=False)
              (3): LeakyReLU(negative_slope=0.2)
            )
          )
          (2): ConvDown(
            (model): Sequential(
              (0): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1))
              (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
...
      )
      (3): Conv2d(128, 1, kernel_size=(3, 3), stride=(1, 1), bias=False)
    )
  )
)
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

I managed to create the GAN model but a bug probably due to some update prevented me from actually testing it.