Adversarial Attackers

Alex, Nathan, Sebastian, Stephan, Waleed

Challenge

- Dataset: subset of „Galaxy10“ dataset (https://astronn.readthedocs.io/en/latest/galaxy10sdss.html)
- Some modifications: 4 classes removed, images cropped to 64x64 pixels

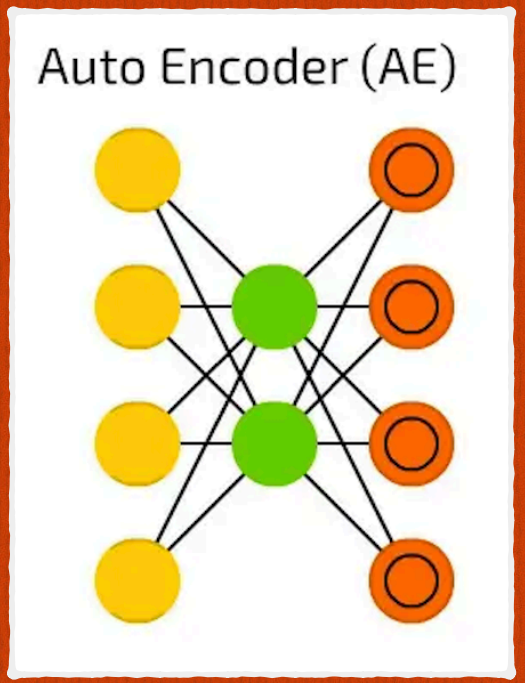
- Task: Find a galaxy that looks as close as possible to our Milky Way
- Bonus task: Generate new galaxies that look as close as possible to our Milky Way

Main Idea

- Use a (variational) autoencoder

- Encoder embeds features into latent space
- Our idea: Features can be compared in latent space using euclidian distance

- Decoder decodes features into pictures



https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

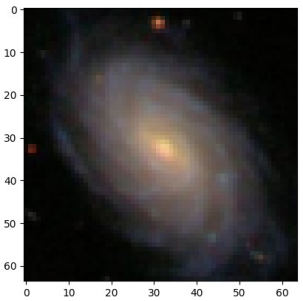
- Input image: 64x64 version of given Milky Way image in Google collab

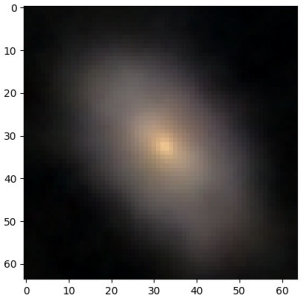
Basic Idea: Autoencoder

```
def build_autoencoder(img_shape, code_size):
    encoder = Sequential(name = 'encoded')
    encoder.add(InputLayer(img_shape))
    encoder.add(Flatten())
    encoder.add(Dense(code_size))

    decoder = Sequential(name = 'decoded')
    decoder.add(InputLayer((code_size,)))
    decoder.add(Dense(np.prod(img_shape))) # hier 64 x 64 x 3
    decoder.add(Reshape(img_shape))

    return encoder, decoder
```

img_shape = (64, 64, 3)

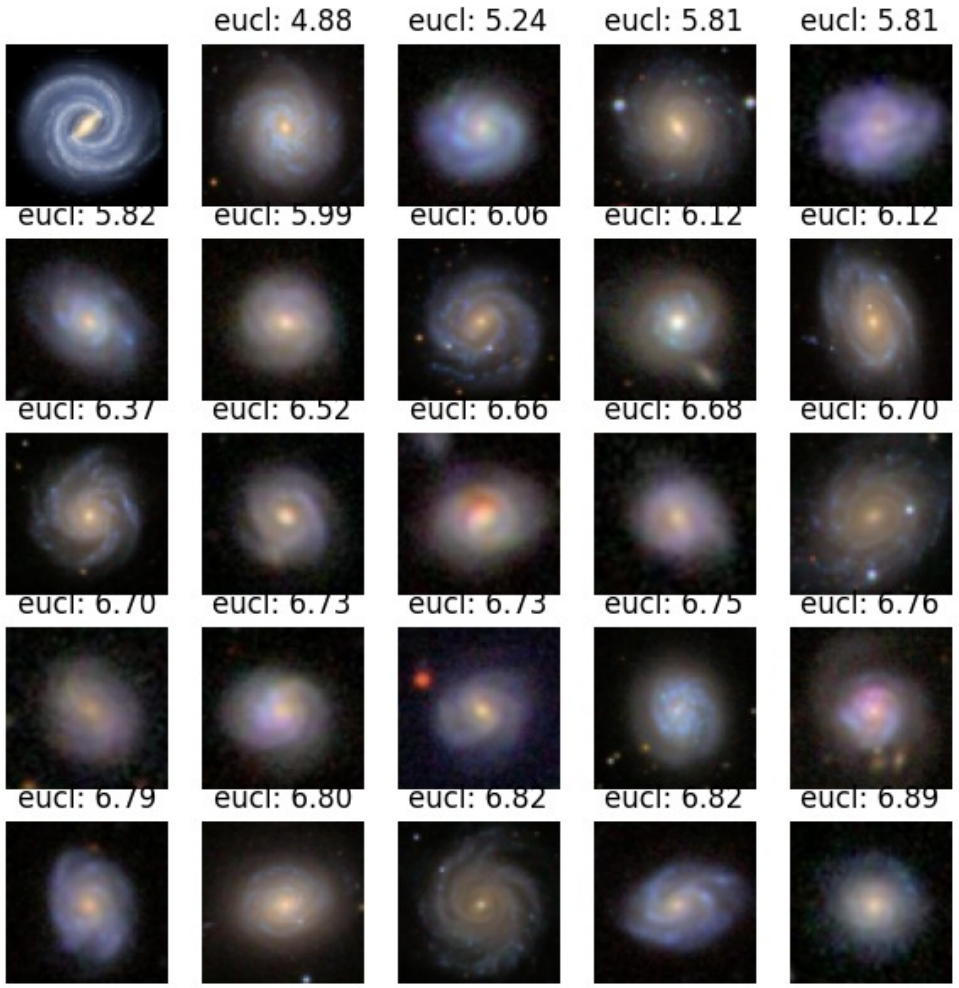
Latent space dimension = 64



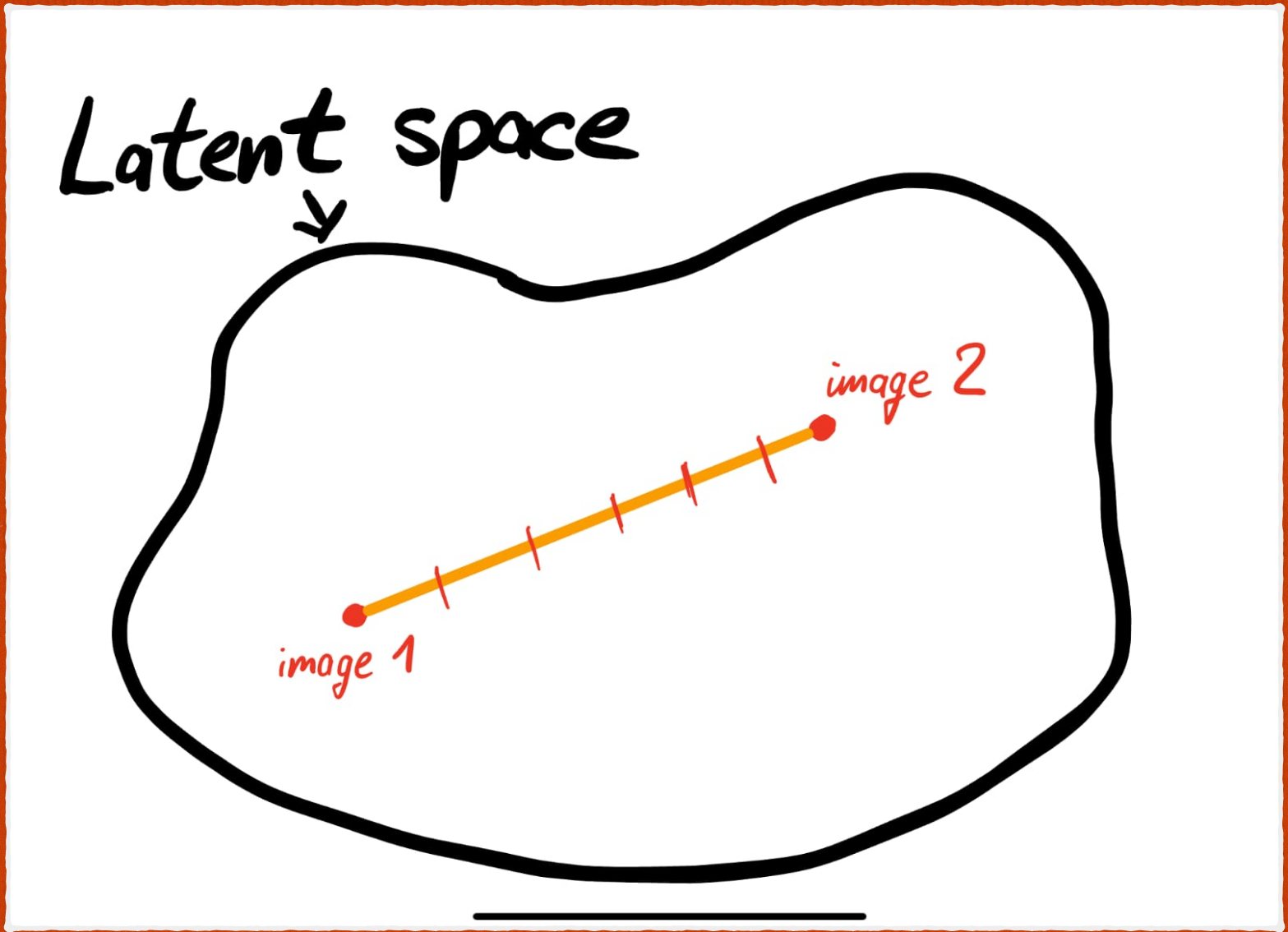


Here could be your latent space

Autoencoder - similarity search



Autoencoder - sampling idea



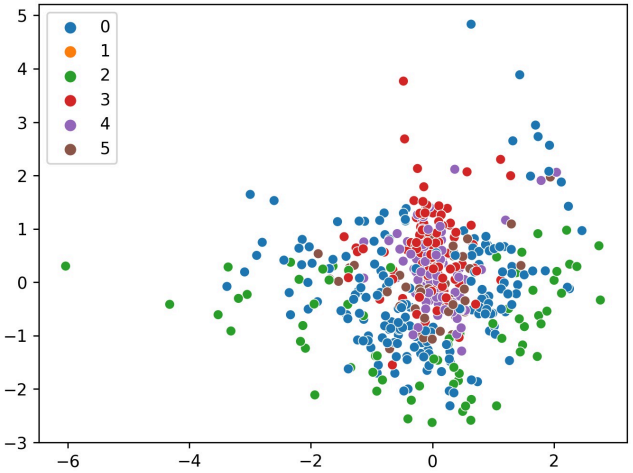
Autoencoder - sampling

Variational autoencoder

```
ConvVAE(
  (enc1): Conv2d(3, 8, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (enc2): Conv2d(8, 16, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (enc3): Conv2d(16, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (enc4): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2))
  (fc1): Linear(in_features=64, out_features=128, bias=True)
  (fc_mu): Linear(in_features=128, out_features=2, bias=True)
  (fc_log_var): Linear(in_features=128, out_features=2, bias=True)
  (fc2): Linear(in_features=2, out_features=64, bias=True)
  (dec1): ConvTranspose2d(64, 64, kernel_size=(4, 4), stride=(1, 1))
  (dec2): ConvTranspose2d(64, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (dec3): ConvTranspose2d(32, 16, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (dec4): ConvTranspose2d(16, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
  (classifier): Linear(in_features=2, out_features=6, bias=True)
)
```

Latent space dimension = 2

Latent space



Sampling