COMP6248 Lab 8 Exercise – Exploring Latent Spaces

Wei Chien Teoh (Eugene)
wct1c16@soton.ac.uk

29 April 2021

Introduction

The results are seeded using pytorch_lightning.seed_everything(0) to provide reproducible results.

1 Exploring the latent space of a VAE

1.1 Systematically sample a VAE

Listing 1: Code to generate latent image.

```
latent_img = np.empty((588, 588))

x_points = np.linspace(-4, 4, 21)
y_points = np.linspace(-4, 4, 21)
xx, yy = np.meshgrid(x_points, y_points)
for i in range(len(xx)):
    for j in range(len(yy)):
        z = torch.tensor([xx[j, i], yy[j,
            i]],
            dtype=torch.float32).view(1, 2)
        output = dec(z)
        img = output.view(28,
            28).detach().numpy()
        latent_img[j*28:j*28+28,
            i*28:i*28+28] = img
```

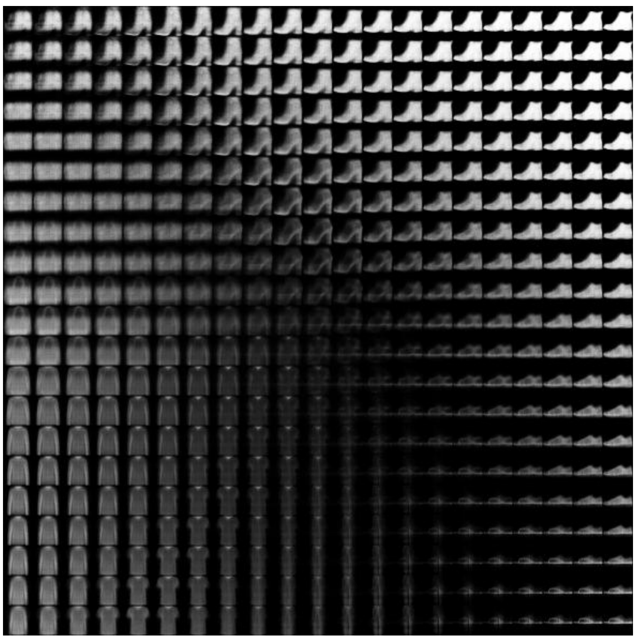

Figure 1: Latent image of VAE.

2 Exploring the code space of a standard auto-encoder

2.1 Systematically sample an Autoencoder

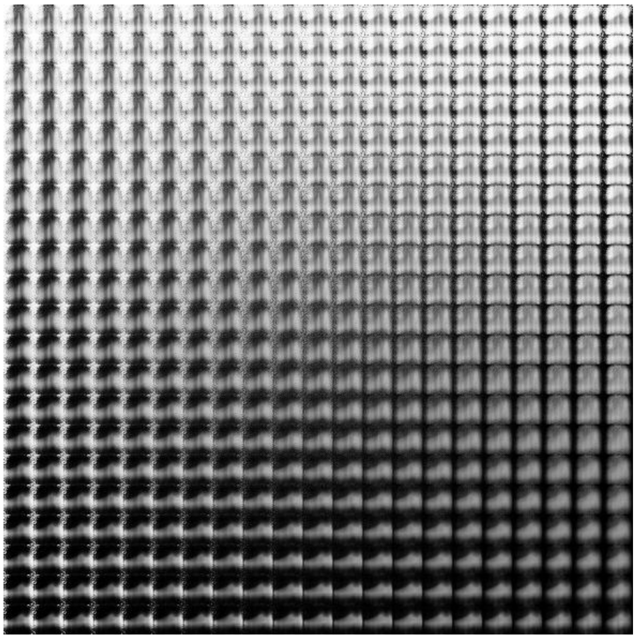

Figure 2: Latent image of autoencoder.

2.2 Compare the latent spaces of the VAE and autoencoder

Fig. 1 shows that VAE is able to learn latent representations of the data such as the structure of shirts, boots, pants, etc. The VAE also attempts to learn orthogonal/uncorrelated structures because of the orthogonality (non-diagonals are zeros) imposed while learning the latent variance.

Fig. 2 rather shows that the autoencoder performs compression of the data into a smaller subspace, thus learning the most important latent features. It can be observed that the latent representations are composed of a linear combination of the structures such as shirts, boots, pants, etc.