Generative AI Theory & Applications
Report 2: Variational Autoencoders 2

Syed Abraham Ahmed

March 7, 2025

Contents

1 Sample Generation Analysis

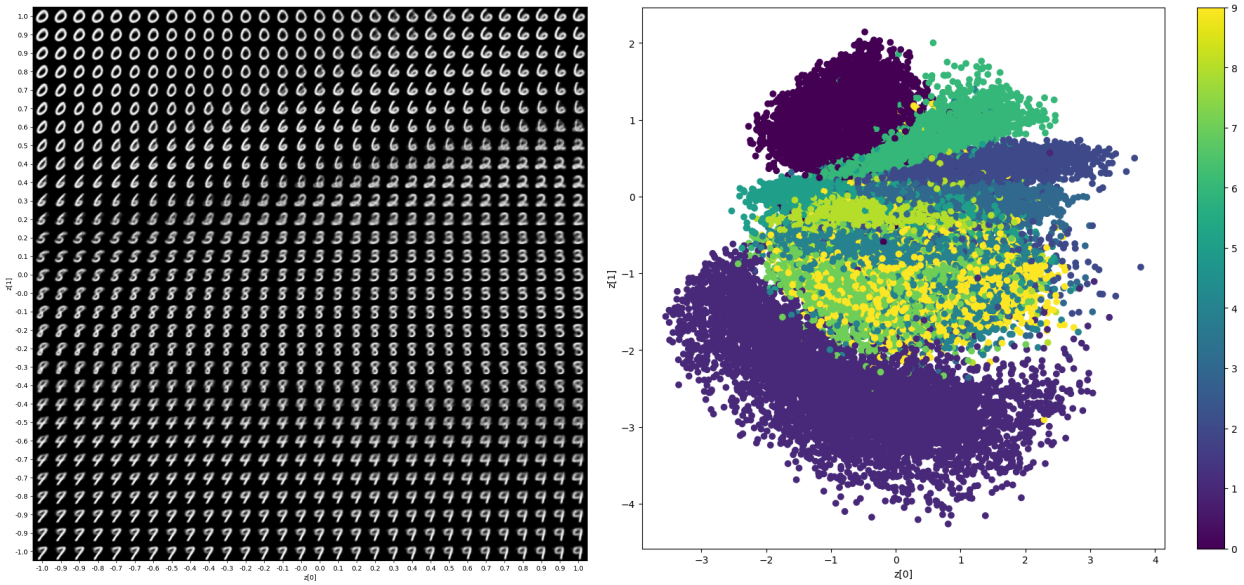
1.1 Additional Steps to Generate a Sample

A Variational Autoencoder (VAE) consists of an encoder and a decoder, but imposes a probabilistic framework on its latent space. The encoder outputs parameters (mean μ and log variance $\log \sigma^2$) describing a normal distribution over the latent vectors, rather than a single deterministic point.

In order to generate a sample, we must:

1. **Sample** a latent vector $\mathbf{z}$ from the (approximate) prior distribution, often assumed to be $\mathcal{N}(0, I)$. One may use the models' learned distribution parameters (μ, σ^2) if the intention is to reconstruct or slightly change an existing input.
2. **Decode** the sampled random latent vector by passing it through the decoder, which outputs a generated sample in the data space (a 28×28 image in the case of MNIST).

The key difference from a standard autoencoder is that we sample from the learned distribution in latent space, rather than taking a single deterministic encoding.



(a) Sample Latent Space Plots (b) Latent Space Plot with Labeled Training Set

Figure 1: Latent Space Plot Analysis

Note: I may interchangeably reference the axis labels as (z_1 & $z[0]$) and (z_2 & $z[1]$). They both label their corresponding axis in the z latent space.

1.2 10 Random Samples

Below is Python block showing sampling 10 random latent vectors from a standard normal distribution and decode them into generated images. We're currently using the same model provided from Keras and using the same neural network architecture for the encoder and decoder [1].

```python
import numpy as np
import matplotlib.pyplot as plt

def generate_new_images(vae, num_images=10, latent_dim=2):
    # Sample from a normal distribution
    z_random = np.random.normal(size=(num_images, latent_dim))

    decoded = vae.decoder.predict(z_random, verbose=0)

    plt.figure(figsize=(2 * num_images, 2))
    for i in range(num_images):
        img = decoded[i].reshape(28, 28)
        plt.subplot(1, num_images, i + 1)
        plt.imshow(img, cmap="gray")
        plt.axis("off")
    plt.suptitle("Randomly Generated Digits (2D Latent Space)")
    plt.show()

generate_new_images(vae)
```

Listing 1: Generating & displaying 10 random samples from a trained VAE.

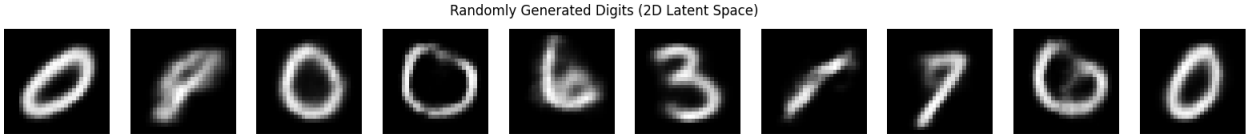


Figure 2: 10 Randomly Generated Images from Random Latent Vectors

1.3 Generative Quality and Common Practice

Generative Quality: When sampling from the model's latent distribution and using the 10 generated images, the images looks coherent but appear blurry, and imperfect. This could be from the dataset being complex or if the VAE capacity is limited. Using the true sampling approach (rather than just taking the mean of the distribution) shows the models' generative power. [2]

Mean vs. Sampling: In many **reconstruction tasks**, we would pass the mean of the latent distribution through the decoder because it often yields a stable, representative reconstruction. For creative or **generative tasks**, sampling is essential. Without sampling, we lose the ability to produce a diverse set of outputs. [2]

From the provided conference paper from ETH Zurich, we can support the claim whether to use the mean or fully sample from $\mathcal{N}(0, I)$ as it depends on whether the goal is reconstruction vs. generating new variations. The general advantage of VAEs is their probabilistic sampling approach.

1.4 Behaviour of the Two Loss Terms

During training, the overall VAE loss is typically (depending on the neural network architecture, we'll consider our provided example) the sum of:

- **Reconstruction Loss** (binary cross-entropy or mean squared error), ensuring the reconstructed sample is close to the input.
- **KL-Divergence Loss** (Kullback-Leibler divergence), encouraging the learned latent distribution to remain close to a standard normal prior and prevent overfitting.

Which Dominates? Initially, the reconstruction loss often dominates because the network must learn to map inputs to reasonably accurate outputs. As training occurs, the KL term becomes critical in shaping the latent space to be smooth and continuous.

Should One Dominate? They should be balanced. If the KL term is strong early on, the network may be forced to conform to the prior at the cost of reconstruction quality. Ignoring the KL term leads to an autoencoder-like model that does not truly learn a proper latent distribution. A balance is aimed for, and in practice, some implement a warm-up or weighting of KL vs. reconstruction to ensure stable training (beta-VAEs) [3].

2 Latent Space Interpolation

An aspect of VAEs is their ability to interpolate in latent space. To achieve a latent space interpolation we:

1. Choose 2 random latent vectors z_1 and z_2, drawn from $\mathcal{N}(0, I)$ or by encoding two different inputs.
2. Generate intermediate points $z_\alpha = (1 - \alpha)z_1 + \alpha z_2$ for $\alpha \in [0, 1]$.
3. Decode each z_α to show a morph from one latent representation to the other.

Below is a code block where we pick two encoded points from the test set or random vectors, then interpolate:

```
import numpy as np
import matplotlib.pyplot as plt

def interpolate_images(vae, z1, z2, n_steps):
    alphas = np.linspace(0, 1, n_steps)
    interpolated_images = []

    for alpha in alphas:
        z_interp = (1 - alpha) * z1 + alpha * z2
        decoded = vae.decoder.predict(z_interp)
        interpolated_images.append(decoded[0])

    plt.figure(figsize=(12, 2))
    for i, img in enumerate(interpolated_images):
        ax = plt.subplot(1, n_steps, i + 1)
        plt.imshow(img.reshape(28, 28), cmap="gray")
        plt.axis("off")
    plt.suptitle("Latent Space Interpolation (Along z[0] axis, [-1, 1])")
    plt.show()

# 2 Random Latent Vectors
z1 = np.random.normal(0, 1, (1, 2))
z2 = np.random.normal(0, 1, (1, 2))

# 2 Vectors of choice
# z1 = np.array([[-1, 0]])
# z2 = np.array([[1, 0]])

interpolate_images(vae, z1, z2, n_steps=10)
```

Listing 2: Interpolating between 2 latent z vectors and showing the morphing process.

We have below 3 interpolation methods using different defined $z_1,\ z_2$ for each interpolation method being 2 random latent vectors, the z_1 axis, and the z_2 axis.

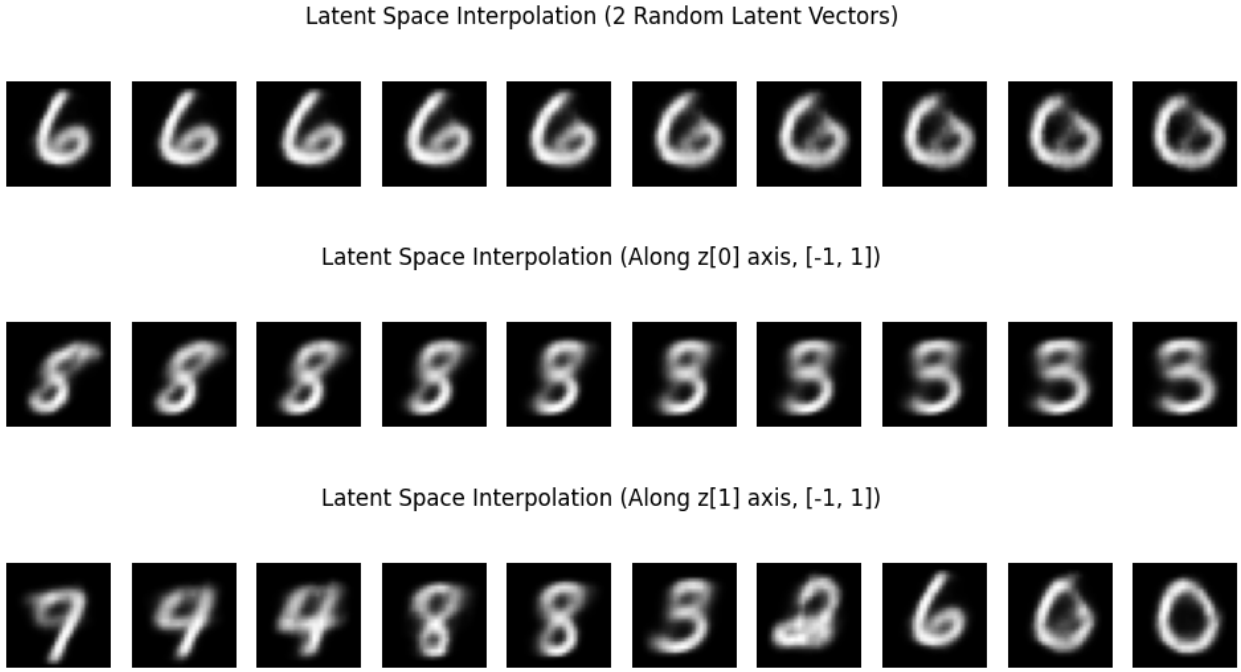


Figure 3: Linear Interpolation Along Different Paths

A carefully chosen pair (z_a, z_b) might show interesting transformations. These latent space interpolation plots show that the VAEs latent space can learn smooth transitions, rather than discrete jumps. For a topic of interest, we look at a subset of the interpolation of the z_2 axis, looking more closely into the 2 to 6 transition below. From Figure 1a and following the z_2 axis, we can kind of see a morph from 2 to 6 in the upper half of the image, which follows our interpolation plot. By a visual esitmation from Figure 1, we will analyze z_2 from 0.2 to 0.4.

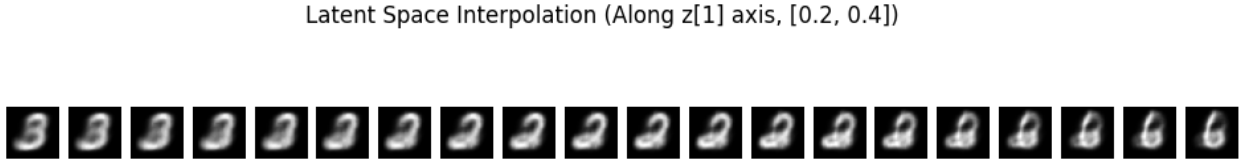


Figure 4: 20 Step Liner Interpolation Along z[1] Axis

The images smoothly transitions from a 3 to 2 to 6. This suggests that the VAE learned some meaningful representation of the digit structures. However, the intermediate images do appear overly blurry and jumping from one attribute to another. This supports the claim that the VAE hasn't learned sharp enough feature representations as well as the latent space not being well regularized.

3 Additional Discussion

3.1 Comparison of Autoencoders vs. Variational Autoencoders

Outside of generation, Autoencoders provide an approach to unsupervised anomaly detection. By training on "standard and normal data", an AE learns compressed reconstruction for inputs, and creates a large error for anomalies. [4]

Variational Autoencoders can also detect anomalies but do so in a probabilistic way. As VAEs learn a probability distribution of the data provided, anomalies can be checked for by a low likelihood. Since VAEs enforce a smooth latent space, they generalize which overtly smooths out the latent space and potentially washes out subtle fine anomaly details. [4] Since VAEs produce a a generalized latent space, they are great for problems with a representation learning requirement where AEs fall short in performance metrics due to their reconstructive nature. [4]

4 References

[1] F.Chollet, Variational AutoEncoder Example, Keras, 2024.DIRECT LINK

[2] E. Palumbo, I. Daunhawer, J. Vogt, MMVAE+: Enhancing the Generative Quality of Multimodal VAEs Without Compromises, ETH Zurich Department of Computer Science, 2022. DIRECT LINK

[3] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, Google DeepMind, 2017. DIRECT LINK

[4] Z. Barban, G. Webb, S. Pan, C. Aggarwal, Deep Learning for Time Series Anomaly Detection: A Survey, University Melbourne, IBM Research Centre, 2023. DIRECT LINK