Latent Space Interpolation in the Variational Autoencoder and the Denoising Diffusion Probabilistic Model

Max Hitzemann

Bachelor's thesis
2026:K19

Lund University

Faculty of Science
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Bachelor's Theses in Mathematical Sciences 2026:K19
ISSN 1654-6229

LUNFMS-4088-2026

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/

Abstract

This work investigates image synthesis of two generative models: the Variational Autoencoder (VAE) and the Denoising Diffusion Probabilistic Model (DDPM). One approach for synthesis is to interpolate between real samples within the models' latent spaces. To understand how we can interpolate in the models' latent space, we derive the loss functions of the models. We trained the models on MNIST and Fashion-MNIST dataset. With that knowledge, we explain why continuous linear interpolation in the latent space often finds the shortest interpolation path in the VAE: because of a the geometric structuredness and continuity of the latent space. In the DDPM, linear interpolation in the latent space tends to fail for samples with different features because the model does not learn a smooth geometric structure for the latent representation of the data. We show how increasing the dimensionality of the latent space from two to three in the VAE results in more clarity of the synthesized samples. In the two dimensional latent space we train a degree two polynomial by the difference between each image in the pixel space and by encouraging proximity to high density regions in the latent space. This resulted in shorter interpolation path compared to linear interpolation in the latent space.

Popular Summary

Interpolation is a method to estimate new data points given a discrete amount of samples. In image processing, interpolation can be used to find intermediate states between two pixels in an image or between two separate images. Nowadays, image interpolation is used for image processing such as enhancing the quality of medical images or generating intermediate frames in videos to increase the frame rate. To achieve the best possible results, modern interpolation techniques increasingly rely on artificial intelligence, specifically models that are capable of image synthesis. These models are called generative models.

One of the models that we will investigate for this purpose is the variational autoencoder. The first research on autoencoders began in the 1980s. The general idea was to reduce the data to its most important features in order to simplify training generative models. In practice, an encoder maps a high-dimensional data point to a point in a lower-dimensional space called the latent space. The encoded point is then decoded to reconstruct the original data point.

Based on this idea, variational autoencoders were introduced in 2013 as an extension of the traditional autoencoder. Instead of mapping each input to a single point in latent space, the encoder encodes each input to a distribution in latent space. The benefit of using the variational autoencoder is that interpolation is no longer restricted to discrete data points in the latent space. Because each encoded data point corresonds to a distribution in the latent space, it becomes possible to interpolate between two encoded data points while still obtaining meaningful reconstructions of the data.

The second model is the denoising diffusion probabilistic model. First introduced in 2020, the model consists of processes; the forward diffusion and the reverse diffusion. In the forward process, noise is gradually added to an image until it becomes indistinguishable from pure noise. In the reverse process, the generative model learns to predict the noise that was added at each step in the forward process. By interpolating between diffused data points, intermediate data points can be generated. These data points are then generated through the reverse diffusion process.

We will compare these two models based on how well they produce meaningful and visually coherent image interpolations. This will be done by generating image interpolations with the two models and investigating their respective strengths and weaknesses. To further challenge the models, we will attempt to interpolate between samples with little or no structural similarity. For the variational autoencoder, we will also investigate different non-linear interpolation path in order to improve the to increase the quality of the resulting interpolation.

Acknowledgements

I would like to express gratitude to my supervisor, Ted Kronvall, for his guidance and valuable feedback throughout this thesis. His expertise and encouragement helped widen the perspective and the scope of the thesis. I am very grateful for the endless support of my parents. They have helped me in every step of my life and have always been there for me. My sister Lara was an essential part of discussing and testing my understanding of the theory while also giving me valuable advice for the writing process. To my friends, thank you for being epic in every aspect and making my time in Lund so much more enjoyable. Lastly, I would like to say a big thank you to my partner Liv. You were always there for me for thesis-related and unrelated problems. I would not have wanted to do it without you.

Contents

1 Introduction **11**
- 1.1 Research Aim . . . 12
- 1.2 Usage of Artificial Intelligence . . . 12

2 Model Background and Loss Function Derivation **13**
- 2.1 Variational Autoencoders . . . 13
  - 2.1.1 Bottleneck . . . 13
  - 2.1.2 Encoder . . . 14
  - 2.1.3 Decoder . . . 14
  - 2.1.4 Reparameterization Trick . . . 14
  - 2.1.5 Evidence Lower Bound . . . 15
- 2.2 Denoising Diffusion Probabilistic Model . . . 18
  - 2.2.1 Forward Diffusion . . . 18
  - 2.2.2 Variance Schedule . . . 18
  - 2.2.3 Reverse Diffusion . . . 19
  - 2.2.4 Evidence Lower Bound . . . 20

3 Implementation and Experimentation **23**

4 Image Interpolation **25**
- 4.1 Linear Image Interpolation . . . 25
- 4.2 VAE Interpolation . . . 26
- 4.3 DDPM Interpolation . . . 36

5 Conclusion **41**

6 Appendix **45**
- A Derivation of Binary Cross Entropy in the ELBO . . . 45
- B Extended Derivation Reparameterization . . . 46
- C Computing the Resulting Gaussian Density . . . 47
- D DDPM Image Interpolation . . . 48

Chapter 1

Introduction

Most neural networks have been used over the years for tasks such as classification and regression. In the recent years, generative machine learning models have become increasingly popular, instead of being used for mostly classification tasks, generative models are able to synthesize realistic samples. In many domains, the creation process has found a wide use in aspects of everyone's life. Popular use of a generative models nowadays include generating images, coding or translating.

Image generation can be done in different ways. Text-to-image models as the DALL-E [8] or stable diffusion [10] are at the forefront of modern generative models. Another way is that models interpolate in the latent space between two images for image synthesis. The interpolation between two images has a widespread use from generating smooth transitions between frames in a video to simple and more playful uses as finding out how the kid of two celebrities could look like.

This thesis introduces image synthesis using interpolation. In its simplest form we can linear interpolate between two images by scaling the intensity of the individual pixels. The obvious problem is that we are not generating images that will have realistic combinations of each of their features. A latent space is a representation of data in which features and structure of the original data are encoded. Latent spaces can possess continuity and geometric structure, enabling interpolation paths that synthesize realistic images. We consider an interpolation realistic if every image in the interpolation path can be interpreted as a representative sample from the set of realistic images. In addition, we want our interpolation path as short as possible to avoid unnecessary transitions.

For the latent space interpolation, we will explain two generative models which we use in the thesis, namely the variational autoencoder (VAE) and the denoising diffusion probabilistic model (DDPM). As reference for the model description for the variational autoencoder we use [2] and for the denoising diffusion model [5]. We derive the loss functions for each model to provide insight into their underlying mechanisms and their use in latent space interpolation. Although these models are seminal rather than state-of-the-art, we include them to illustrate the fundamental principles of modern generative models.

The VAE learns to encode data into a lower-dimensional latent space. Distances in this space are not strictly euclidean. The shortest path along a curve generally depends on the curvature of the data manifold. Consequently, we implement, visualize, and analyze both linear and nonlinear interpolations within this space.

The DDPM does not learn a low-dimensional latent representation. Instead, it operates in a high-dimensional Gaussian latent space corresponding to an intermediate state of the diffusion process. We implement, visualize, and analyze linear interpolation within this latent space.

Linear interpolation in both the VAE and the DDPM have been studied previously. Research has tried to avoid the problems in the latent interpolation of the VAE by for example cubic spline interpolation [13], interpolation aware training [14] or manifold learning [15].

To our knowledge, our approach for learning the polynomial in the latent space of the VAE has not been documented yet. A similar approach has been done for GAN [9]

1.1 Research Aim

The research aim of this thesis is to

- Understand the training of generative models and why it works.
- Compare the latent space linear interpolation of the models to linear image interpolation.
- Compare the latent space linear interpolation between the VAE and DDPM.
- Analyze the effect on latent space linear interpolation in the VAE when increasing dimensionality of the latent space.
- Examine whether we can improve on the latent space interpolation in the VAE by estimating the nonlinear shortest path

1.2 Usage of Artificial Intelligence

Generative AI has been used in the thesis strictly as an aid. ChatGPT and Gemini were used as a discussion tool to help understanding the theory behind the models. As a supportive asset for implementing code in Python and typesetting in Latex. These tools assisted in improving the phrasing of sentences. As well as for finding incorrect grammar or typing errors in the thesis.

Chapter 2

Model Background and Loss Function Derivation

We define $\mathcal{X}$ to be the general set from which we take the input data. This set is a subset of $\mathbb{R}^n$ for $n > 1$ and contains all data points under consideration. Say for example, we have the dataset $\bar{\mathcal{X}} \subset \mathcal{X}$. $\bar{\mathcal{X}}$ can be a dataset of images of armchairs. $\mathcal{X}$ is the space of all images of existing armchairs but also possible non-existing images of armchairs. Let $x \in \bar{\mathcal{X}}$ be a data sample. We denote $\hat{x} \in \mathcal{X}$ as the reconstruction of the sample x generated by the model.

We further define $\mathcal{Z}$ to be the latent space associated with the machine learning model with $\mathcal{Z} \subset \mathbb{R}^s$ and $n \geq s > 0$. Points in the latent space are denoted by $z \in \mathcal{Z}$

2.1 Variational Autoencoders

The general class of autoencoders are machine learning models composed of three main parts; an encoder, a bottleneck and a decoder. The encoder is a mapping $e : \bar{\mathcal{X}} \to \mathcal{Z}$ that compresses the data into a low-dimensional space, expressed as $z = e(x)$. This space is part of the bottleneck and is called the latent space $\mathcal{Z}$. The decoder is a mapping $d : \mathcal{Z} \to \mathcal{X}$ that reconstruct the data from its latent representation, described as $\hat{x} = d(z)$. The training objective is to minimize the reconstruction loss $||d(e(x)) - x||$. More on principles of autoencoders can be found in [1, Ch. 14].

Variational autoencoders (VAE) are a special kind of autoencoders. VAE operate on the same principles, but instead of simply reducing the data into the lower dimensional latent space, it also learns a probabilistic encoding of the data. This approach is based on variational Bayesian methods. The VAE can therefore not just reconstruct the input data but also generate new data by sampling from the latent space. To achieve good results from sampling at any point in the latent space, we need continuity and geometric structuredness in the space. There is more information about this in [2].

2.1.1 Bottleneck

The bottleneck is the connection between the encoder and the decoder. In the bottleneck we find the latent space, the most important feature of the autoencoder.

The purpose of the latent space is to form a compressed representation of the input data. While training conventional autoencoders, the model may often map similar inputs to nearby

regions in the latent space to minimize the reconstruction loss. This leads to isolated regions in the latent space that represent the different data samples. The VAE learning objective encourages continuity and geometric structure of the latent space. Continuity means that sampling from similar regions in the latent space should yield reconstructions with similar features and structure. By geometric structure, we means that decoding any point in the latent space should result in valid samples from the data space $\mathcal{X}$.

As a result, we will be able to sample from any point in that latent space $\hat{x} = d(z)$ for all $z \in \mathcal{Z}$ such that it is likely that $\hat{x} \in \mathcal{X}$. Moreover, points that are close to each other in the latent space should reconstruct into samples with similar features and structure.

It is common to assume that the latent variables $z \in \mathcal{Z}$ follow a prior distribution $p(z)$ which is taken as a standard normal distribution, $\mathcal{N}(\mathbf{0}, \mathbf{I})$, as discussed in Section 3 of [2]. This promotes a continuous latent space centered at the origin.

2.1.2 Encoder

In the VAE the encoder learns the features and structure of the input data by mapping it to a lower-dimensional latent representation using a neural network. A common architecture of the encoder consists of multiple fully connected perceptron layers where subsequent layer should decrease in nodes.

We can describe the encoder as the mapping $e : \mathcal{X} \to \mathcal{Z}$, for $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{Z} \subset \mathbb{R}^s$ for $n >> s$. The probability density function $p(z|x)$ is the posterior probability that a point z in the latent space was generated by the input x. This posterior is is generally intractable because its evaluation requires computation of the marginal likelihood $p(x) = \int p(x|z)p(z)dz$. Since the likelihood $p(x|z)$ is parameterized by the decoder neural network with nonlinear hidden layers, the exact posterior is analytically intractable, see [2]. The VAE uses a variational Bayesian method to approximate the posterior probability by approximating it with a Gaussian distribution $p(z|x) \approx q(z|x) = \mathcal{N}(\mu, \sigma^2\mathbf{I})$.

2.1.3 Decoder

The VAE also contains a second neural network called a decoder. The decoder learns the mapping $d : \mathcal{Z} \to \mathcal{X}$ to decode any point in the latent space. The architecture is often designed as a mirror of the encoder, expanding the number of nodes until it matches the dimension of the input data. The conditional distribution $p(x|z)$ is the likelihood to reconstruct the data x from the latent space z. The decoder parameters are trained to maximize that probability. We will discuss more details in the following sections.

2.1.4 Reparameterization Trick

Backpropagation is an algorithm to compute the gradient of the loss function to adjust the weights in the learning process. To enable backpropagation, the sampling operation must be expressed in a differentiable form. The encoder approximates the intractable posterior with a Gaussian distribution

$$q(z|x) = \mathcal{N}(\mu_x, \sigma_x^2\mathbf{I}),$$

where μ_x and σ_x are the output of the encoder network.

Applying a reparameterization trick expresses sampling in a differentiable form:

$$z = \mu_x + \sigma_x \epsilon, \text{ with } \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

With this reparameterization, z is differentiable with respect to μ_x and σ_x, and it allows backpropagation to be applied to the loss function to train these parameters. During the training, ϵ is independently sampled for each data point in the batch for every forward pass.

2.1.5 Evidence Lower Bound

The loss function of the VAE is the evidence lower bound (ELBO). It encourages the empirical posterior distribution to match the prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ while still allowing accurate reconstruction of samples with features and structure. This is possible with a combination of the Kullback-Leibler divergence (KL divergence), measuring the distance of two distributions, and the reconstruction loss that minimizes the distance between reconstructed and original samples.

Derivation of ELBO Using the KL Divergence

There are multiple ways of deriving the ELBO. In this thesis we will use the KL divergence. The KL divergence is a measure for quantifying the difference between an approximated distribution and a true distribution, in the continuous case defined as

$$D_{KL}\left(p(x)||q(x)\right) = \int_{\mathbb{R}} p(x) \log \frac{p(x)}{q(x)} dx,$$

with $p(x)$ and $q(x)$ being the corresponding density functions. In the VAE we know the approximate posterior distribution. Applying the KL divergence directly would not yield any meaningful expression because we do not know the true posterior distribution of $p(z|x)$. As stated before, we will approximate $p(z|x)$ with $q(z|x)$. Therefore, we apply the reverse KL divergence such that

$$D_{KL}(q(z|x)||p(z|x)) = \int_{\mathbb{R}^s} q(z|x) \log \frac{q(z|x)}{p(z|x)} dz \tag{2.1.1}$$

$$= \int_{\mathbb{R}^s} q(z|x) \left(-\log \frac{p(z|x)}{q(z|x)} \right) dz \tag{2.1.2}$$

$$= -\int_{\mathbb{R}^s} q(z|x) \log \frac{p(z,x)}{q(z|x)p(x)} dz \tag{2.1.3}$$

$$= -\int_{\mathbb{R}^s} q(z|x) \log \frac{p(z,x)}{q(z|x)} dz + \int_{\mathbb{R}^s} q(z|x) \log p(x) dz \tag{2.1.4}$$

$$= -\underbrace{\mathbb{E}_{q(z|x)} \left(\log \frac{p(z,x)}{q(z|x)} \right)}_{=:\text{ELBO}} + \log p(x) \tag{2.1.5}$$

Now we will show why this loss is called the evidence lower bound. In our model, the marginal likelihood $p(x)$ measures the probability density that the trained generative model produces the data x when sampling from its latent prior and decoder. The ELBO is the lower bound of the model log-likelihood of the data. Thus, we want to show:

$$\log\left(p(x)\right) \geq \text{ELBO}, \forall x. \tag{2.1.6}$$

We can rewrite 2.1.5 to

$$\log p(x) - D_{KL}(q(z|x)||p(z|x)) = \text{ELBO}.$$

If we can show that $D_{KL}(q(z|x)||p(z|x)) \geq 0$ then 2.1.6 holds true. We will show that the reverse KL divergence $D_{KL}(q(x)||p(x)) \geq 0$ holds for all probability distributions P and their approximations Q.

$$D_{KL}(q(x)||p(x)) = \int_{\mathbb{R}} q(x) \log \frac{q(x)}{p(x)} dx \tag{2.1.7}$$

$$= -\int_{\mathbb{R}} q(x) \log \frac{p(x)}{q(x)} dx \tag{2.1.8}$$

$$\geq -\log \left(\int_{\mathbb{R}} q(x) \frac{p(x)}{q(x)} dx \right) \tag{2.1.9}$$

$$= -\log \left(\int_{\mathbb{R}} p(x) dx \right) = 0 \tag{2.1.10}$$

The inequality in 2.1.9 is a result from the Jensen's inequality. This shows that the ELBO is the lower bound.

Consequently, we want to maximize the ELBO for the model to perform optimally.

$$\mathbb{E}_{z\sim q(.|x)} \left(\log \frac{p(z,x)}{q(z|x)} \right) = \int_{\mathbb{R}^s} q(z|x) \log \frac{p(z,x)}{q(z|x)} dz \tag{2.1.11}$$

$$= \int_{\mathbb{R}^s} q(z|x) \log \frac{p(x|z)p(z)}{q(z|x)} dz \tag{2.1.12}$$

$$= -\int_{\mathbb{R}^s} q(z|x) \log \frac{q(z|x)}{p(x|z)p(z)} dz \tag{2.1.13}$$

$$= -\int_{\mathbb{R}^s} q(z|x) \log \frac{q(z|x)}{p(z)} dz + \int_{\mathbb{R}} q(z|x) \log p(x|z) dz \tag{2.1.14}$$

$$= -D_{KL} \left(q(z|x)||p(z) \right) + \mathbb{E}_{q(z|x)} \left(\log p(x|z) \right) \tag{2.1.15}$$

The first term in 2.1.15 represents the KL divergence between the approximated posterior distribution $q(z|x) = \mathcal{N}(\mu_x, \sigma_x^2 \mathbf{I})$ and the latent space distribution assumed to be $p(z) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. To maximize the ELBO means to minimize the difference in distribution between the posterior and the latent space distribution. In other words, encouraging the approximation to be close to a standard normal distribution.

The second term represents the reconstruction loss, which measures the distance of the original sample x and the reconstructed sample from the latent space $\hat{x}$. For MNIST data, we chose the negative binary cross entropy (BCE) between the reconstructed sample and the original, see appendix A. To maximize the ELBO means to maximize the negative BCE which happens when $\hat{x}$ and x are identical.

To sum up, to maximize the ELBO, the KL divergence trains μ_x and σ_x towards the mean and standard deviation of a standard normal distribution whilst the reconstruction loss ensures that the resulting sampling from the latent space yields reconstructions that preserve the features and structure of the original data.

This expression can be greatly simplified for use in the model. The details are omitted as the focus of the thesis is the derivation of the loss function for understanding and not for

usage. For a detail explanation of how to simplify this expression for the loss function, take a look at [3, Section 2.4–2.5].

2.2 Denoising Diffusion Probabilistic Model

Diffusion models are nowadays used in most state-of-the-art generative models for image synthesis, see [10] or [8]. Both models use diffusion-based generative modeling combined with transformer-based text encoders to enable image generation from text prompts.. The development of modern diffusion models began with [5]. In this chapter we follow and explain in detail the derivation presented in the paper.

The Denoising Diffusion Probabilistic Model (DDPM) can be used for image denoising and synthesis through interpolation. Diffusion models have two processes; the forward diffusion and the reverse diffusion. In the forward process, we gradually add Gaussian noise to an image until it is indistinguishable from pure noise. The generative model then learns to predict the noise added at each step t in the reverse process.

2.2.1 Forward Diffusion

During the forward diffusion process, the model progressively adds noise to the sample. This noise is Gaussian distributed $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Instead of constantly adding the same amount of noise, the addition of noise follows a schedule; it starts with a small amount of noise and as t increases, the noise added increases. The forward process itself has no learnable parameters. It strictly follows the schedule. The state of each image at step t can be computed directly.

A Markov process is defined by the Markov property, meaning that every conditional distribution of the next state only depends on the current state. The forward diffusion process is explicitly constructed as a Markov process and therefore satisfies the memorylessness property.

2.2.2 Variance Schedule

We can define this forward process therefore as the Markov chain:

$$x_0 \xrightarrow{q(x_1|x_0)} x_1 \to ... \to x_{t-1} \xrightarrow{q(x_t|x_{t-1})} x_t \to ... \to x_T$$

where each $q(x_t|x_{t-1})$ for $1 \leq t \leq T$ describes the transition kernel in the Markov chain, x_0 is the original sample and x_T is the pure noised sample. Through the Markov chain, the complex data distribution is gradually transformed and converges to a tractable standard Gaussian prior.

We define $q(x_t|x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I})$. The specific choice of β_t is described in 3.Through this we can sample x_t recursively at any step t. We can sample from x_t by explicit Gaussian reparameterization.

$$x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t, \text{ for } \epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

which we show in the following lines.

We claim that

$$x_t|x_{t-1} \stackrel{d}{=} q(x_t|x_{t-1}). \tag{2.2.1}$$

As we defined $q(x_t|x_{t-1})$ to be Gaussian and we know that ϵ_t is Gaussian, it is sufficient to show that $x_t|x_{t-1}$ and $q(x_t|x_{t-1})$ have the same mean and covariance, which implies they are

equal in distribution.

$$\mathbb{E}[x_t|x_{t-1}] = \mathbb{E}\left[\sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t|x_{t-1}\right] \tag{2.2.2}$$

$$= \sqrt{1-\beta_t}\mathbb{E}[x_{t-1}|x_{t-1}] + \sqrt{\beta_t}\mathbb{E}[\epsilon_t|x_{t-1}] = \sqrt{1-\beta_t}x_{t-1} \tag{2.2.3}$$

$$\text{Var}[x_t|x_{t-1}] = \text{Var}\left[\sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_t|x_{t-1}\right] \tag{2.2.4}$$

$$= \text{Var}\left[\sqrt{\beta_t}\epsilon_t|x_{t-1}\right] = \beta_t \,\text{Var}[\epsilon_t|x_{t-1}] = \beta_t\mathbf{I} \quad \square \tag{2.2.5}$$

This allows us to sample x_t directly from x_0, since the sum of independent Gaussians is again Gaussian. This is a great idea because instead of computing the full Markov chain for the forward process, we will be able to compute x_t directly for any t, reducing the computational cost significantly.

We can consider the recursive computation starting at x_1.

$$x_1 = \sqrt{1-\beta_1}x_0 + \sqrt{\beta_1}\epsilon_1 \tag{2.2.6}$$

$$x_2 = \sqrt{1-\beta_2}x_1 + \sqrt{\beta_2}\epsilon_2 = \sqrt{1-\beta_2}\left(\sqrt{1-\beta_1}x_0 + \sqrt{\beta_1}\epsilon_1\right) + \sqrt{\beta_2}\epsilon_2 \tag{2.2.7}$$

$$= \sqrt{1-\beta_2}\sqrt{1-\beta_1}x_0 + \sqrt{1-\beta_2}\sqrt{\beta_1}\epsilon_1 + \sqrt{\beta_2}\epsilon_2 \tag{2.2.8}$$

To simplify the expression, we define $\alpha_t := 1 - \beta_t$. Continuing with the recursive computation up until x_t results in:

(2.2.9)

$$x_t = \sqrt{\alpha_t}\sqrt{\alpha_{t-1}}...\sqrt{\alpha_1}x_0 + \sqrt{1-\alpha_t}\epsilon_t + \sqrt{1-\alpha_{t-1}}\sqrt{\alpha_t}\epsilon_{t-1} + ... + \sqrt{1-\alpha_1}\sqrt{\alpha_2}\sqrt{\alpha_3}..\sqrt{\alpha_t}\epsilon_1$$

(2.2.10)

$$= \left(\prod_{s=1}^{t}\sqrt{\alpha_s}\right)x_0 + \sum_{s=1}^{t}\left(\sqrt{1-\alpha_s}\prod_{k=s+1}^{t}\sqrt{\alpha_k}\right)\epsilon_s$$

We define $\bar{\alpha}_t := \prod_{s=1}^{t}\alpha_s$, which yields:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sum_{s=1}^{t}\left(\sqrt{1-\alpha_s}\frac{\sqrt{\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_s}}\right)\epsilon_s \tag{2.2.11}$$

which can finally be simplified to

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon \tag{2.2.12}$$

The detail of the simplification can be found in the Appendix B. We can compute x_t for any t given the original sample x_0 which makes the forward process computationally efficient. Further, this formula gives us a way to reconstruct x_0 given x_t and the predicted noise ϵ_θ. We will therefore refer to it as the reconstruction formula.

2.2.3 Reverse Diffusion

The reverse process aims to approximate the reverse Markov transition at step t. We denote the true transition probability as $q(x_{t-1}|x_t)$. We construct a Markov chain from $T \to 0$, where

x_0 is the denoised sample. Similar to the the VAE, this distribution is intractable so instead we try to approximate it with

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(\mu(x_t,t), \Sigma(x_t,t)) \text{ for } 1 \leq t \leq T.$$

The model learns to predict this distribution for all t. Depending on the size of T and the amount of noise we add at each step, the signal-to-noise ratio (SNR) for large t will be too small for the model to recover any meaningful data. To train the DDPM we aim to minimize the evidence lower bound for the predicted distribution $q(x_{t-1}|x_t)$.

2.2.4 Evidence Lower Bound

For the loss derivation in the DDPM we start with $\log p_\theta(x_0)$, the log-likelihood of the data distribution. Rather than maximizing this probability, we will minimize the negative log-likelihood $-\log p_\theta(x_0)$. The following derivation is based on the steps and results in [5].

We define $p_\theta(x_{0:T}) := p(x_T)\prod_{t=1}^T p_\theta(x_{t-1}|x_t)$ and $q(x_{1:T}|x_0) := \prod_{t=1}^T q(x_t|x_{t-1})$.

$$-\log p_\theta(x_0) = -\log \int_{\mathbb{R}_n} p_\theta\left(x_{0:T}\right) dx_{1:T} = -\log \int_{\mathbb{R}_n} q(x_{1:T}|x_0)\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)} dx_{1:T} \tag{2.2.13}$$

$$= -\log\left(\mathbb{E}_q \frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right) \leq \mathbb{E}_q\left(-\log\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right), \tag{2.2.14}$$

where the inequality results from Jensen's inequality. The right-hand side of the inequality can be rewritten as:

$$\mathbb{E}_q\left(-\log\frac{p_\theta(x_{0:T})}{q(x_{1:T}|x_0)}\right) = \mathbb{E}_q\left(-\log p(x_T) - \sum_{1\leq i\leq T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})}\right) := \text{ELBO}. \tag{2.2.15}$$

We can rewrite this again:

$$\text{ELBO} = \mathbb{E}_q\left(-\log p(x_T) - \sum_{2\leq i\leq T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} - \log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right). \tag{2.2.16}$$

We can now apply the Markov property and Bayes Theorem to rewrite:

$$q(x_t|x_{t-1}) = q(x_t|x_{t-1},x_0) = \frac{q(x_{t-1}|x_t,x_0)q(x_t|x_0)}{q(x_{t-1}|x_0)}.$$

Equation 2.2.16 yields:

$$\text{ELBO} = \mathbb{E}_q\left(-\log p(x_T) - \sum_{2\leq i\leq T}\log\frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t,x_0)}\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} - \log\frac{p_\theta(x_0|x_1)}{q(x_1|x_0)}\right). \tag{2.2.17}$$

The second fraction in the sum is a telescoping sum:

$$\sum_{2\leq i\leq T}\log\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} = \sum_{2\leq i\leq T}\log q(x_{t-1}|x_0) - \log q(x_t|x_0) = \log q(x_1|x_0) - \log q(x_T|x_0)$$

which simplifies 2.2.17 to:

$$\text{ELBO} = \mathbb{E}_q \left(-\log \frac{p(x_T)}{q(x_T|x_0)} - \sum_{2 \leq i \leq T} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_{t-1}|x_t, x_0)} - \log p_\theta(x_0|x_1) \right). \tag{2.2.18}$$

This is now the rewritten variational loss for the DDPM. Using the definition of the KL divergence in 2.2.18, this becomes:

$$\mathbb{E}_q \left(\underbrace{D_{KL}\left(q(x_T|x_0)||p(x_T)\right)}_{L_T} + \sum_{2 \leq i \leq T} \underbrace{D_{KL}\left(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)\right)}_{L_{t-1}} - \underbrace{\log p_\theta(x_0|x_1)}_{L_0} \right). \tag{2.2.19}$$

The first term L_T has no learnable parameters because we know the distribution of $p(x_T)$ which was previously defined as a standard Gaussian. In the DDPM paper [5] they define L_0 as the decoder term, which rescales the input for the neural network. The decoder is derived from a Gaussian distribution. The details can be found in Section 3.3 in the aforementioned paper.

Consequently, only L_{t-1} has learnable parameters for the neural network. We need to find the probability distribution of $q(x_{t-1}|x_t, x_0)$. We claim that $q(x_{t-1}|x_t, x_0) = \mathcal{N}(\tilde{\mu}(x_t, x_0), \tilde{\beta}\mathbf{I})$. To show this one first applies Bayes Theorem to rewrite:

$$q(x_{t-1}|x_t, x_0) = c \cdot q(x_t|x_{t-1}) q(x_{t-1}|x_0). \tag{2.2.20}$$

Computing results in a Gaussian density with:

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} x_0, \ \sigma^2 \mathbf{I} = \tilde{\beta}_t \mathbf{I} = \frac{(1 - \bar{\alpha}_{t-1})\beta_t}{1 - \bar{\alpha}_t} \mathbf{I}. \tag{2.2.21}$$

See Appendix C for details. With this result, we can now continue with L_{t-1}. We choose to fix the variance of reverse model distribution $p_\theta(x_{t-1}|x_t)$ to $\sigma_t^2 = \tilde{\beta}_t$. This is suggested in DDPM paper [5]. Because we have the KL divergence between two normal distributions, we can simplify this expression to:

$$\mathbb{E}_q(D_{KL}\left(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t)\right)) = \mathbb{E}_q \left(\frac{1}{2} \left(\text{tr}(\mathbf{I}) - k + \frac{1}{\sigma_t^2} \left(\tilde{\mu}_t(x_t, x_0) - \mu_\theta(x_t, t) \right)^2 \right) \right),$$

see [6]. We can express $\tilde{\mu}_t(x_t, x_0)$ with respect to ϵ instead of x_0 by substituting the reconstruction formula 2.2.12 for x_0.

$$\tilde{\mu}_t(x_t, \epsilon) = \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t + \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} \frac{x_t - \sqrt{1 - \bar{\alpha}_t}\epsilon}{\sqrt{\bar{\alpha}_t}} \tag{2.2.22}$$

$$= \frac{\sqrt{\alpha_t}(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t + \frac{\beta_t}{\sqrt{\alpha_t}(1 - \bar{\alpha}_t)} x_t - \frac{\beta_t}{\sqrt{\alpha_t}\sqrt{1 - \bar{\alpha}_t}} \epsilon \tag{2.2.23}$$

$$= \frac{1}{\sqrt{\alpha_t}} \left(\frac{\beta_t + \alpha_t(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right) \tag{2.2.24}$$

$$= \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon \right), \tag{2.2.25}$$

where we used the definition $\beta_t = 1 - \alpha_t$ in the last equality. Because x_t is well defined, μ_θ can alternatively be parameterized as:

$$\mu_\theta = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_t, t) \right). \tag{2.2.26}$$

Consequently, it is enough that the neural network predicts the noise ϵ in x_t at step t. The loss simplifies to:

$$\text{ELBO} = \mathbb{E}_{x_0,\epsilon} \left(\frac{\beta_t^2}{2\sigma_t^2 \alpha_t (1-\bar{\alpha}_t)} ||\epsilon - \epsilon_\theta \left(\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t \right)||^2 \right). \tag{2.2.27}$$

The simplification in the loss comes from the change of $\mathbb{E}$. We can ignore the constants because they will not influence the training of the model.

Chapter 3

Implementation and Experimentation

All implementations were carried out in Python (version 3.10.12) using PyTorch. We trained both models on MNIST [16] and Fashion-MNIST [17].

We implemented a VAE with two encoding/decoding layers. The input dimension is 784 and the hidden layer has dimension 256. We mostly worked with a latent space of dimension 2 but for an experiment we increased the dimension to 3. We used ReLU as the activation function and the sigmoid function for the output layer. In the implementation, the ELBO is formulated as the sum of the reconstruction loss (BCE) and the KL divergence (for more information, see [3, Secs. 2.4–2.5]). This means we minimize the negative ELBO. Training the VAE is very similar to training other neural networks. We use the Adam optimizer [11], a stochastic gradient-based optimization method. The learning rate is PyTorch Adam's default learning rate of 0.001. This optimizer in combination with backpropagation have shown to be an efficient way of training neural networks. We trained the model using random mini-batches of 32 and for 400 epochs.

The DDPM implementation closely follows the implementation proposed in [5]. We decreased the diffusion steps T from 1000 to 300 for both MNIST and Fashion-MNIST data and retained the linear noise schedule boundaries for β_t in the forward process from the paper ($\beta_1 = 10^{-4}, \beta_{300} = 0.02$). We implemented the suggested U-Net backbone with group normalization, sinusoidal timestep embedding and two consecutive convolutional layers per block. We trained the model by comparing the estimated noise at each diffusion step t to the true noise (see 2.2.4). Optimization is carried out with backpropagation, Adam with learning rate 0.0002, with mini-batches of 32 and for 150 epochs.

Chapter 4

Image Interpolation

Interpolation in the data space $\mathcal{X}$ is an estimation method for reconstructing new points given a discrete set of samples from the data space. In its simplest form, linear interpolation between two points can be denoted as:

$$\tilde{x}_\lambda = \lambda x_i + (1-\lambda)x_j, \text{ for } \lambda \in [0,1], \tag{4.0.1}$$

where x_i and x_j are two samples from the data space $\mathcal{X}$.

If $\mathcal{X}$ is the space containing samples exhibiting special features and structure, interpolation is used to synthesize new samples with similar features and structure. In the simplest linear interpolation of two images, the images are blended together by scaling the intensity of each pixel. This often results in samples that exhibit unrealistic features and structures from both images, without any logical combination, i.e. $\tilde{x}_\lambda \notin \mathcal{X}$. To improve on that, one can use the introduced generative models to interpolate in their respective latent spaces to generate more realistic samples. It is assumed that $\mathcal{X}$ sufficiently covers the range of possible samples.

We aim for any $\tilde{x}_\lambda$ on an interpolation path between x_i and x_j to remain in the data space $\mathcal{X}$, with $\tilde{x}_0 = x_i$ and $\tilde{x}_1 = x_j$. This can be expressed as minimizing the loss

$$L = \int_0^1 \left| \frac{\partial x}{\partial \lambda} \right|_2^2 d\lambda. \tag{4.0.2}$$

Minimizing this loss with respect to a chosen norm determines the shortest interpolation path between two points. We aim for smooth transitions between images and for all images along the interpolation path to be contained in the set $\mathcal{X}$.

4.1 Linear Image Interpolation

We apply linear image interpolation as described in 4.0.1 to images from MNIST and Fashion-MNIST. Depending on λ each pixel's intensity scales in the images. For $\lambda = 0$ the equation equals x_j, while for $\lambda = 1$ the equation equals x_i. There is a mixture of pixels from both images for $0 < \lambda < 1$.

As can be observed in 4.1 the middle images corresponding to $0.3 \lessapprox \lambda \lessapprox 0.7$ of the interpolation are a mixture between the two samples. These interpolated images cannot be considered to be in their respective data space $\mathcal{X}$. This suggests that the loss along these linear interpolation paths is not minimized.

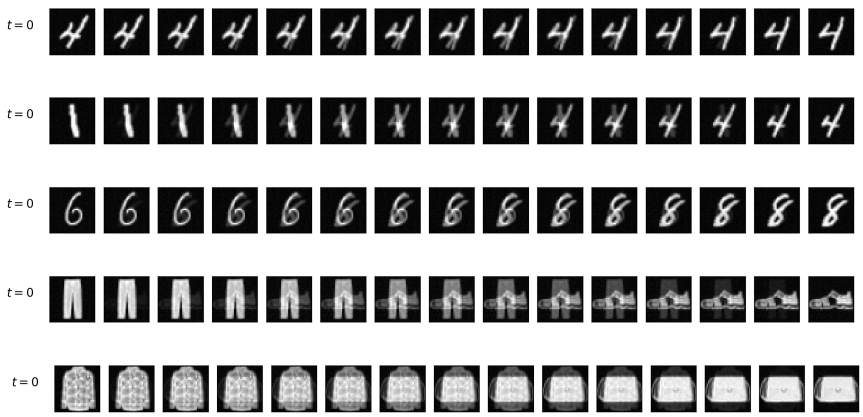

Figure 4.1: Examples of Linear Image Interpolation

4.2 VAE Interpolation

In the VAE, we interpolate in the latent space $\mathcal{Z}$. During training, the model is encouraged to make the latent space continuous and have geometric structure, see 2.1.5. Continuity is enforced by the KL divergence, which encourages the latent space distribution to match a simple prior. Therefore, every point in the latent space can be decoded. The geometric structure is encouraged by the reconstruction loss that ensures that decoded samples corresponding to inputs from $\bar{\mathcal{X}}$ resemble the original sample. Because of the combination of the two losses, latent space should be such that every point decodes into a meaningful and realistic sample from $\mathcal{X}$.

We should therefore be able to pick any point in the latent space and decode it to a meaningful sample in the data space. As a consequence of 2.1.15, we can interpolate between samples in the latent space.

The axes of the two dimensional latent space $\mathcal{Z} \in \mathbb{R}^2$ are denoted by z1 and z2 whilst data points are denoted with z_i.

In the simplest form of latent space interpolation, we take two encoded samples z_i, z_j and linearly interpolate between these points.

$$\tilde{z}_\lambda = \lambda z_i + (1 - \lambda) z_j, \text{ for } \lambda \in [0, 1]$$

This would be the shortest interpolation path if the metric in the latent space is Euclidean, meaning $d(\tilde{z}_\lambda)$ minimizes L. Based on the training objective, decoding $\tilde{z}_\lambda$ for all λ should create a new image $\tilde{x} \in \mathcal{X}$. If the VAE fails to learn a smooth geometric structure in the latent space everywhere, there will be low-density regions in which interpolated points do not correspond to meaningful samples and the loss L is higher.

Examining the latent spaces for the data sets, we can see that each point in the latent space represents an encoded data sample.

In Figure 4.2 the model maps samples from the same class, meaning samples with similar features and structure, in similar regions in the latent space. This is the result of the loss function that combines both the reconstruction loss and the KL divergence. We trained the VAE unsupervised, meaning that the model does not learn the class for each sample as a parameter. We can see that for the MNIST data, the model manages to group data with the

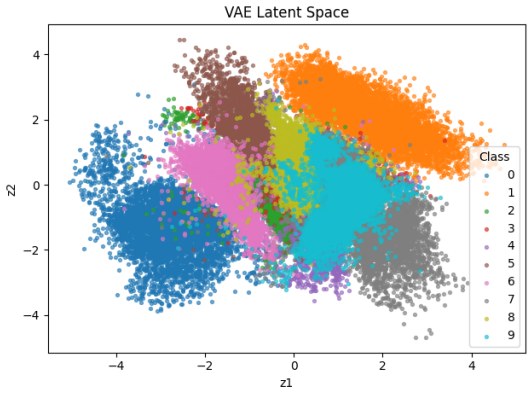

Figure 4.2: Latent Space MNIST

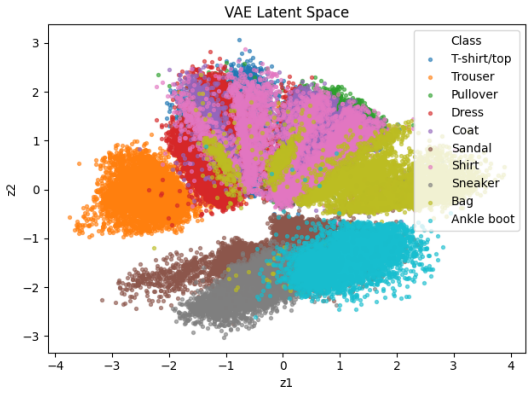

Figure 4.3: Latent Space Fashion-MNIST

same class in the same high density area. The encoded samples are also centered around the origin, indicating that the KL divergence successfully aligned the latent distribution with the prior.

The high-density regions of the different classes often overlap with those of other classes. Some samples from the same class are mapped in high density regions of another class. Figure 4.3 provides a clear illustration of that. If we take a look at the area where most of the topwear is placed (for z2 $\gtrapprox -1$), we can see that it is difficult to identify distinct regions for each individual item (e.g. pullovers, shirts). This suggests that the model successfully identifies the greater structural characteristics of each clothing class (such as footwear, topwear, and trousers), while struggling to sort more subtle or detailed features.

Consider the following examples of latent space interpolation. The linear latent space

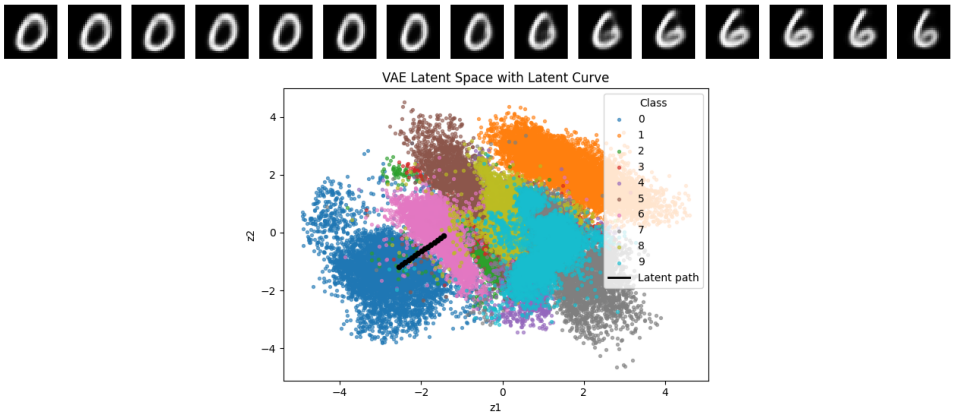

Figure 4.4: Linear Latent Space Interpolation with a Latent Curve Between Samples "0" and "6"

interpolation in 4.4 and 4.5 occurs, respectively, between two high-density areas which primarily contain samples of a single class. This results in smooth transitions between the generated samples.

As shown in Figure 4.6, the unclear boundaries in the latent space may result in that an encoded sample, here a 5, not being clearly identifiable after decoding. This is not a bug but

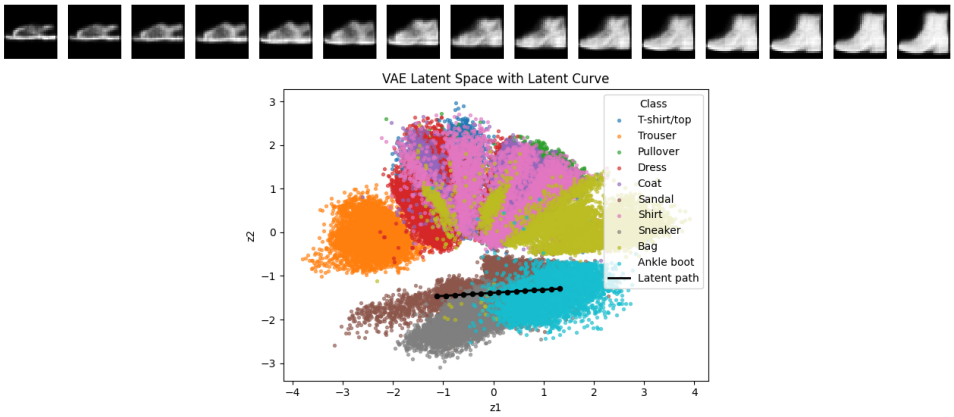

Figure 4.5: Linear Latent Space Interpolation with a Latent Curve Between Samples "Sandal" and "Ankle Boot"

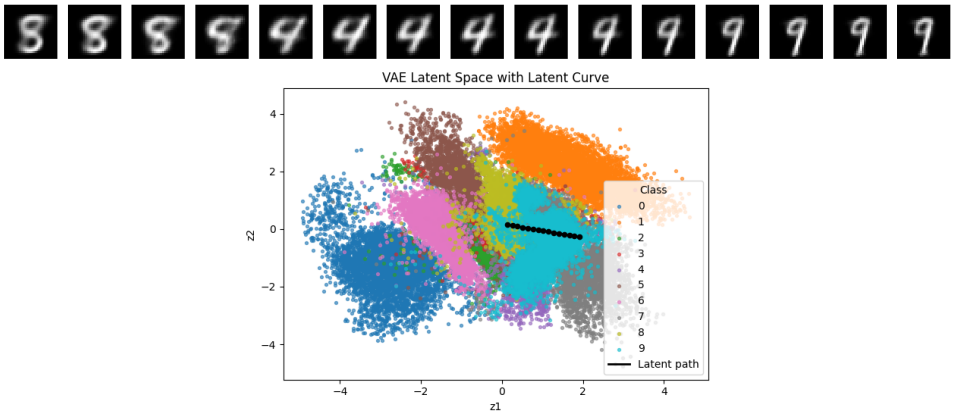

Figure 4.6: Linear Latent Space Interpolation with a Latent Curve Between Samples "5" and "9"

a feature of the VAE. Because the training is unsupervised and we are training it not only for recreating the encoded samples but also for maintaining a meaningful continuous latent space, the boundaries will be overlapping.

We can observe in 4.7 a low-density region separating two high density manifolds in the latent space. The region contains no encoded data samples, which is unsupported by the data distribution. In this specific context, it means that the VAE did not find a mapping from samples that could be interpreted as a mixture of footwear and non-footwear. Presumably, the increase in KL divergence loss was smaller than the corresponding decrease in reconstruction loss

Interpolating through the low density region results in generated samples that cannot be interpreted as valid Fashion-MNIST samples. In these interpolation images, fewer pixels have intensity. The result is a combination of the main features of the input images (legs of trousers and the shoe sole of the sandals). These samples resemble linear image interpolation rather than a smooth semantic transition between generated samples. Moreover, this also implies that the continuity of the latent space is preserved, indicating that the training was successful.

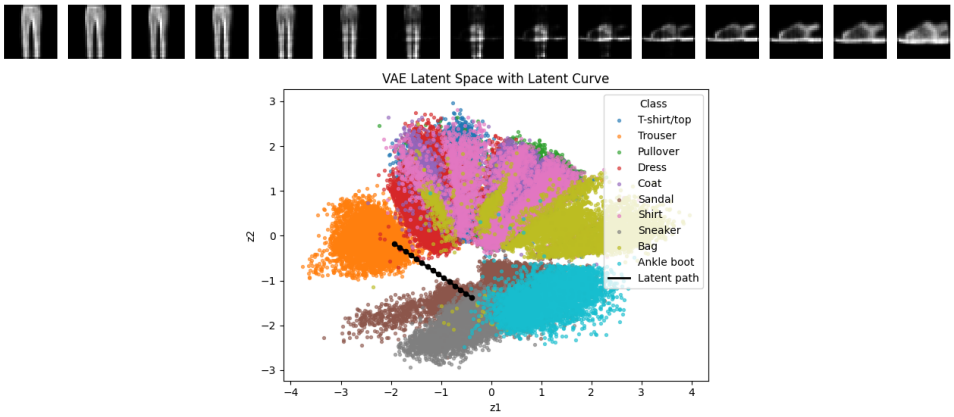

Figure 4.7: Linear Latent Space Interpolation with a Latent Curve Between Samples "Trouser" and "Sandal"

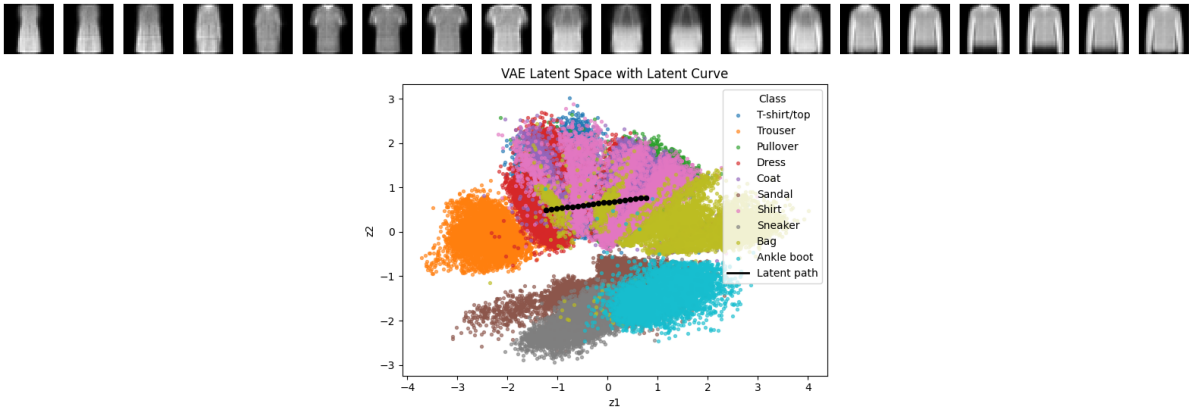

Figure 4.8: Linear Latent Space Interpolation with a Latent Curve Between Samples "Dress" and "Pullover"

Next up, looking at Figure 4.8, the interpolation path crosses multiple different high-density regions. To minimize L, we want the shortest path in the output space. Therefore, we want to cross different regions only if it results in meaningful decoded samples. One could imagine an interpolation path in 4.8 from dress to t-shirt to pullover. In each step the length of the sleeves would increase and the clothing's vertical length would decrease. Here, an additional step from t-shirt to bag appears. Increasing the dimensionality of the latent space allows for more effective arrangement of samples into regions of higher density and consequently for a shorter interpolation path.

In Figure 4.9, because of the extra dimensionality, the VAE has one extra node to learn the features and structure of the sample. This results in a more organized latent mapping of the data set. This results in shorter interpolation path that crosses fewer different classes.

The result of Figure 4.10 indicates that increasing dimensionality does not necessarily fill the low-density region. The resulting interpolation path for the two-and three-dimensional latent space are almost the same.

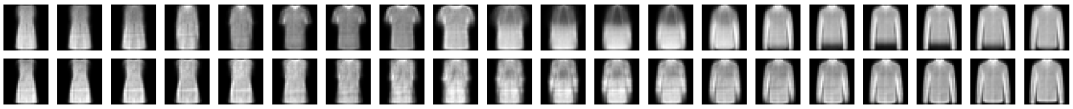

Figure 4.9: Comparison of Linear Latent Space Interpolation of 2D and 3D Latent Space

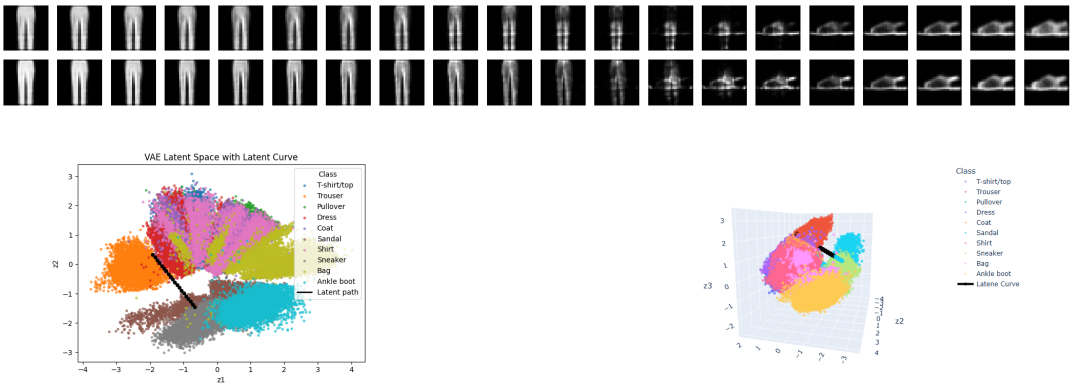

Figure 4.10: Linear Latent Space Interpolation: Comparison of 2D and 3D with a Latent Curve

By expanding the latent space dimensionality, we reduced the interpolation steps. However, this did not result in avoiding low-density regions, preventing us from minimizing L for interpolation paths across the entire manifold.

Next, we will fit a non-linear interpolation line in the latent space of the VAE. We choose a second degree polynomial with the scaling factor α such that the two encoded samples z_i and z_j are the roots of the polynomial. This is possible by applying a rotation such that the line connecting z_i and z_j is aligned with the z1-axis. All subsequent methods trying to optimize the polynomial have been trained with the same set of initial α values as well as the same number of images.

$$\tilde{z} = \alpha\,(z - z_i)\,(z - z_j)\,, \text{ for } z_i \leq z \leq z_j.$$

The parameter α is optimized through backpropagation. For the loss function, we used the squared L_2-norm for consecutive decoded samples.

This approach yielded mixed results. We noticed that restricting the comparison to only higher pixel intensities in the decoded samples improved the results. Therefore, we introduced a threshold for pixel intensities above 0.5. Moreover, using different initializations of α and increasing the interpolation points helped finding a potential global minimum that minimized the loss.

Figure 4.11 shows how the trained polynomial interpolation learns a shorter interpolation path from a trouser to a bag. In the linear interpolation, multiple bag-like images appear at different stages. According to our criteria for shortest interpolation, this step is unnecessary and increases the loss L. The trained interpolation instead finds a path that does not cross the same class twice. This example illustrates of how the trained latent interpolation can decrease the effective interpolation length in the data space even if the length of the interpolation

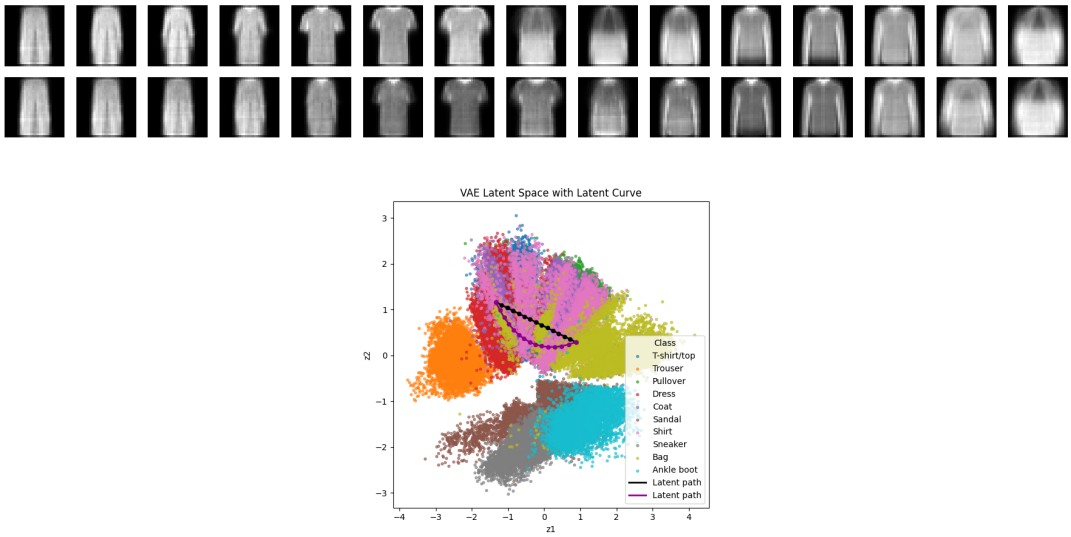

Figure 4.11: Comparison of Linear and Trained Polynomial Interpolation in the Pixel Space Between Samples "Dress" and "Bag"

curve increases in the latent space. Consequently, Euclidean distance in latent space does not directly correspond to length in the output space. Therefore, we suggest that for optimal latent interpolation, non-linear interpolation should be considered.

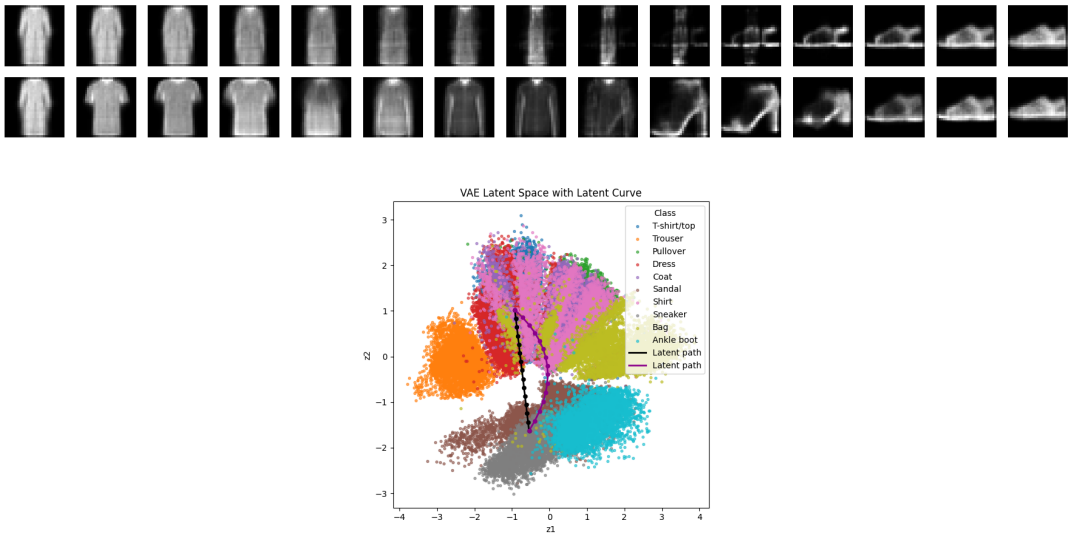

Figure 4.12: Comparison of Linear and Trained Polynomial Interpolation in the Pixel Space Between Samples "Dress" and "Sneaker"

Figure 4.12 shows that the trained interpolation curve circumvents the low-density region. The decoded samples along the polynomial interpolation path show no sign of previously mentioned samples from a low density region. All generated samples are recognizable, which

corresponds to a lower interpolation loss L.

There are also examples in which the curve did not align well with the underlying data manifold, see Figure 4.13. Although the interpolation curve differs significantly from the linear interpolation in the latent space, the resulting interpolation path still crosses a low density region.

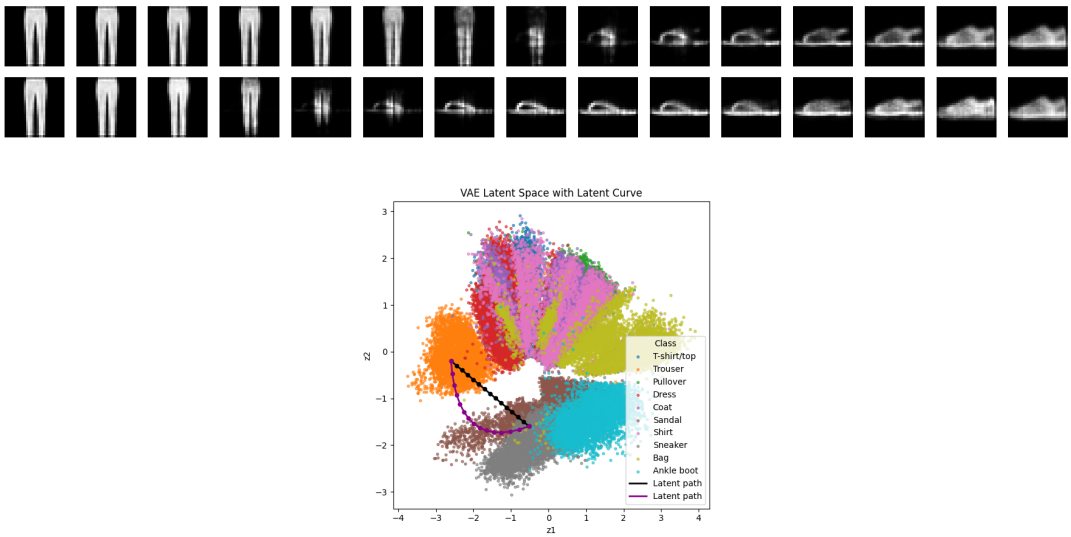

Figure 4.13: Comparison of Linear and Trained Polynomial Interpolation in the Pixel Space Between Samples "Trouser" and "Sneaker"

This method generally provide shorter latent interpolation paths but we suggest using multiple initializations of α to find the smallest loss. Tuning hyperparameters (e.g. threshold, initial α, number of interpolation steps) might improve the results for different data sets.

Instead of adjusting α by comparing the consecutive samples from the latent space, we search a curve that remains close to encoded samples in latent manifold. This corresponds to minimizing the distance between points on the latent interpolation curve and its nearest encoded samples.

$$\underset{i}{\operatorname{argmin}} \, ||\tilde{z}_i(\alpha) - z_k||_2^2 \tag{4.2.1}$$

where $\tilde{z}_i(\alpha)$ is a point on the interpolation curve based on α and z_k is an encoded sample. Varying the initial values of α to escape local minima, we obtain an interpolation path that circumvents the low-density region, see Figure 4.14.

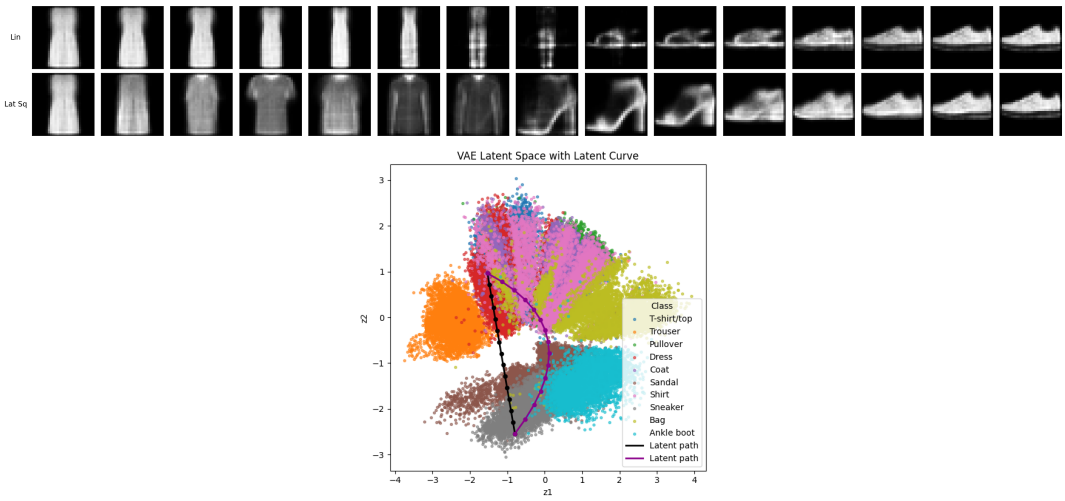

Figure 4.14: Comparison of Linear and Trained Polynomial Interpolation in the Latent Space Between Samples "Dress" and "Sneaker"

As observed in Figure 4.15, this approach may fail to find the shortest interpolation path in a high density region where samples overlap. This is because we are not comparing decoded images but strictly how close each $z_i(\alpha)$ is to the nearest encoded sample.

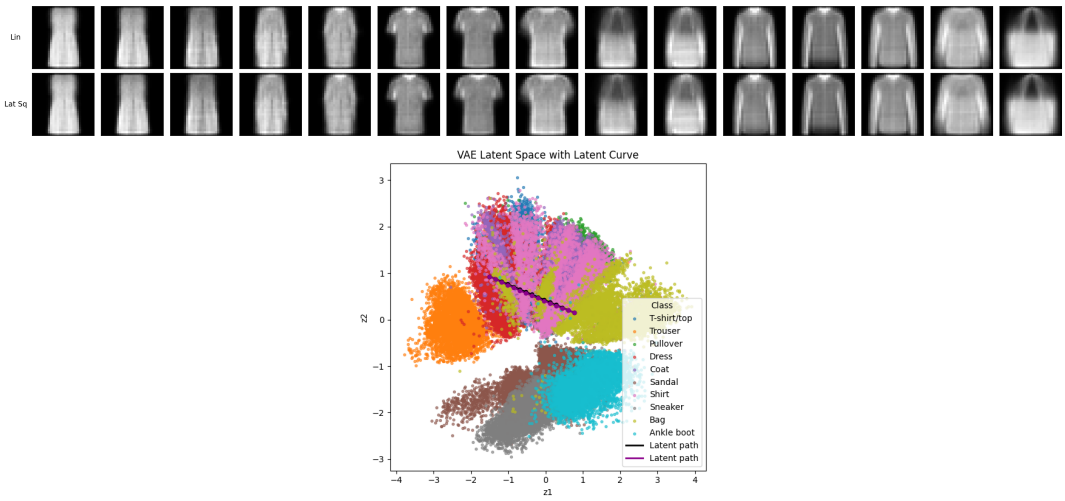

Figure 4.15: Comparison of Linear and Trained Polynomial Interpolation in the Latent Space Between Samples "Dress" and "Bag"

This shows that, without analysis of the pixel space, we cannot learn a curve that finds the shortest interpolation path.

Only looking at the pixel space sometimes resulted in interpolation through low density regions. On the other hand, the training in the latent space failed to find the shortest interpolation

path in high density regions. We propose combining the two parameter trainings to achieve improved latent space interpolation using a second degree polynomial.

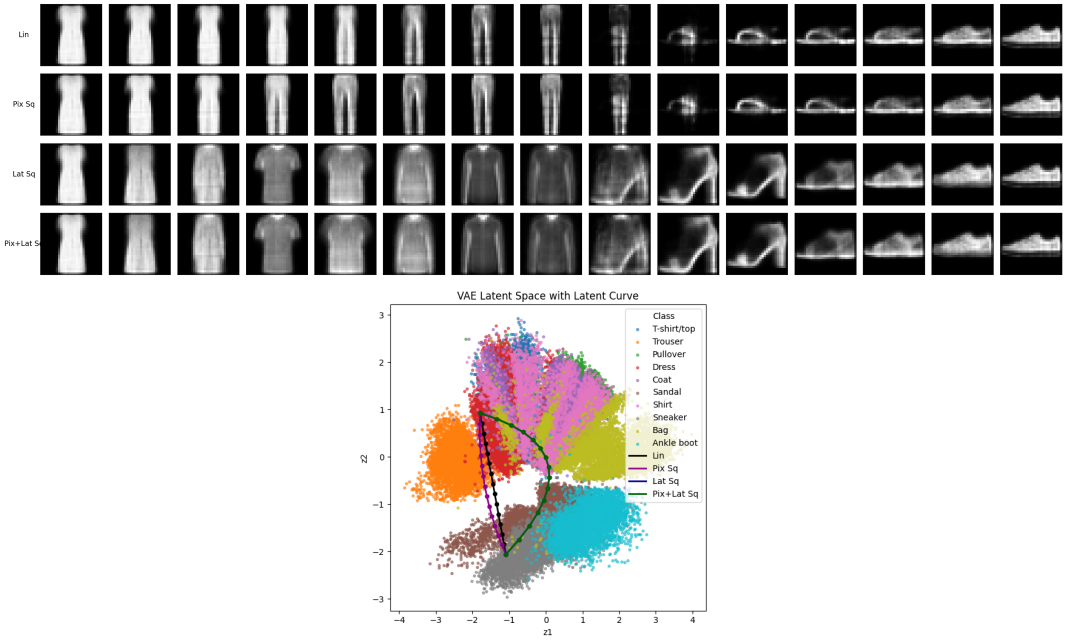

Figure 4.16: Comparison of Linear and Trained Polynomial Interpolation in the Pixel Space, Latent Space, and Combined Between Samples "Dress" and "Sneaker"

Comparing the result in Figure 4.16 shows that, because of additional training, the curve follows the structure of the manifold. As seen in the output images, the learned curve in latent space and the learned curve of the combination of pixel and latent loss result in the same interpolation path. This occurs because the latent loss changes too gradually in the low-density region causing the curve to follow the latent space manifold.

On the other hand, in Figure 4.17 the difference in the latent loss presumably varies very little for different α. As a result, the latent loss curve crosses more class regions during interpolation, producing a longer interpolation path compared to the other curves

Sometimes, there is no clearly defined shortest path when comparing all interpolation curves. A larger sample of images would be needed to reliably compare these functions (see Figure 4.18).

The combination of the two approaches has generally led to the most consistent shortest interpolations. Figure 4.19 illustrates the learning of the interpolation path.

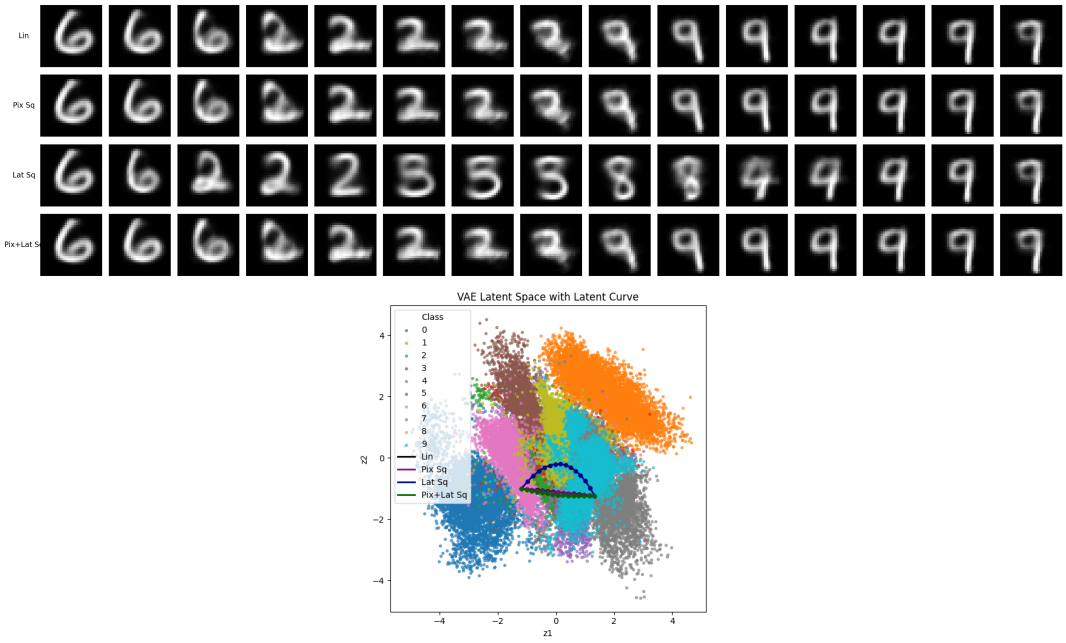

Figure 4.17: Comparison of Linear and Trained Polynomial Interpolation in the Pixel Space, Latent Space, and Combined Between Samples "6" and "9"

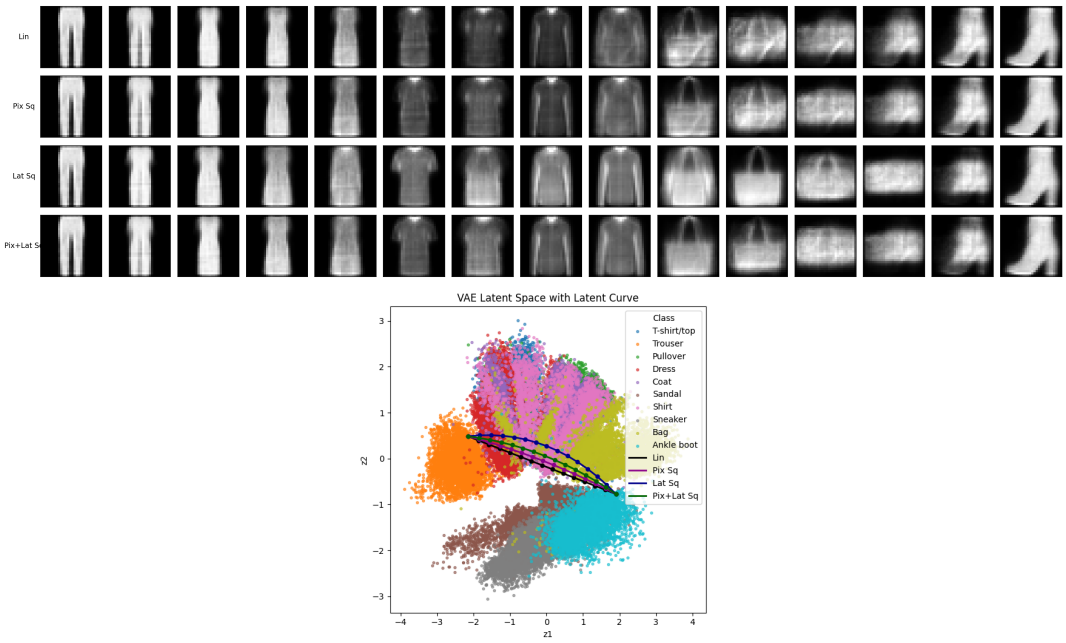

Figure 4.18: Comparison of Linear and Trained Polynomial Interpolation in the Pixel Space, Latent Space, and Combined Between Samples "Trouser" and "Ankle boot"

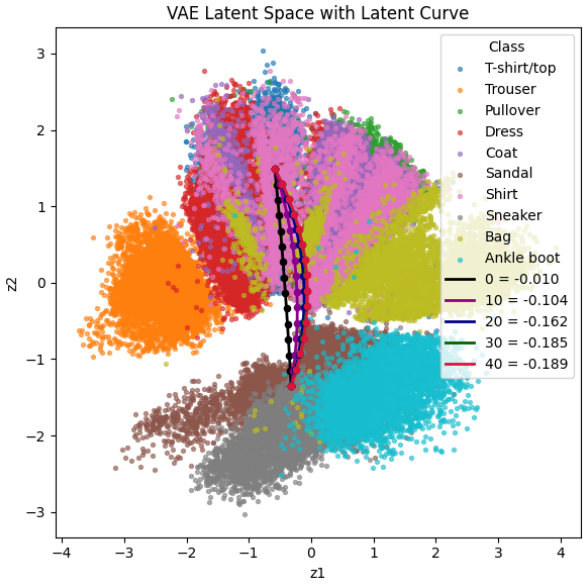

Figure 4.19: Interpolation Curves Across Epochs of the Combined Training Objective

4.3 DDPM Interpolation

In [5], the diffused state of samples for sufficiently large t is called latent space. We can generate samples in this latent space by linearly interpolating between two highly diffused samples. This is possible because the model learns the reverse transitions at each step t, which induces a continuous mapping from noise to data space. The practical training objective reduces to predicting the noise at every step t, which is a result of simplifying the KL divergence terms in 2.2.19. We take two samples $\bar{x}$ and $\hat{x}$ and apply the forward diffusion process up to step t, resulting in noised samples $\bar{x}_t$ and $\hat{x}_t$. As described in [5], we linearly interpolate between these noisy samples and then reconstruct the interpolated points with the reverse process. This can be described mathematically by:

$$\bar{x}_t = \sqrt{\bar{\alpha}_t}\bar{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_j \tag{4.3.1}$$

$$\hat{x}_t = \sqrt{\bar{\alpha}_t}\hat{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_j, \text{ where } j \text{ is fixed.} \tag{4.3.2}$$

We then perform linear interpolation between the two noisy latent samples.

$$\tilde{x}_t^\lambda = \lambda\bar{x}_t + (1-\lambda)\hat{x}_t, \text{ for } \lambda \in [0,1] \tag{4.3.3}$$

We apply the reverse process to the resulting $\tilde{x}_t^\lambda$ yielding:

$$\tilde{x}_{t-1}^\lambda = \frac{1}{\sqrt{\alpha_t}}\left(\tilde{x}_t\lambda - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\tilde{x}_t, t)\right) + \sigma_t z, \text{ where } z \sim \mathcal{N}(\mathbf{0}, \mathbf{I}). \tag{4.3.4}$$

This formula can be derived analogously to the explicit Gaussian reparameterization 2.2.2, using the predicted mean from 2.2.26 and the fixed variance from 2.2.21. Comparing the

reverse process formula in 2.2.12 with the iterative formula 4.3.4 produces qualitatively different reconstructions depending on t. When using the simple reconstruction formula 2.2.12, samples reconstructed from large time steps t remain noisy and unidentifiable. Even for small t, the reconstruction is imprecise and noisy around the digits(see 4.20). The second row in the figure shows the corresponding state of the image in the forward process at step t.

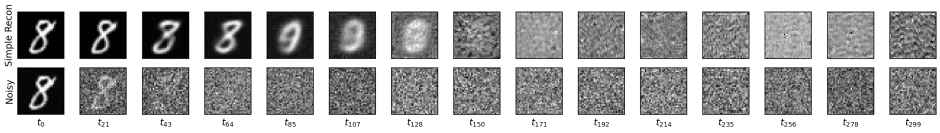

Figure 4.20: DDPM Reconstruction Direct

Figure 4.21 shows the reconstructed images for the same initial sample using formula 4.3.4.

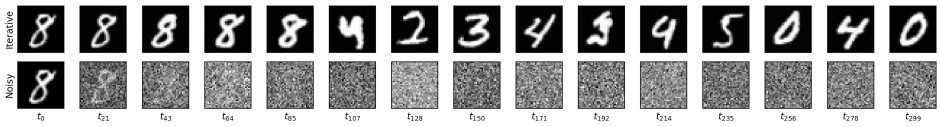

Figure 4.21: DDPM Reconstruction Iteratively

Figure 4.22 shows a comparison of the direct and iterative formulas for computing reconstructed samples. With both formulas, there is no exact reconstruction for arbitrary t. However, the reconstructed image remains in the same class as the original sample up until the signal-to-noise ration (SNR) becomes too small. The model can no longer recover the distinguishing features of the input sample from the noise. When using the iterative formula 4.3.4, the reconstructed sample for large t can still be identified as belonging to the dataset $\mathcal{X}$. This indicates continuity of the learned reverse mapping.

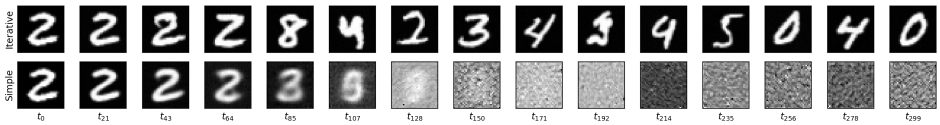

Figure 4.22: DDPM Reconstruction Comparison

We use the iterative formula for interpolation because of its superior denoising quality and to stay consistent with the latent interpolation approach explained in [5].

In 4.23, for $t = 0$, we use standard simple linear interpolation in image space, since the samples are still in their original (non-diffused) state. Up to approximately $t = 50$, we observe an interpolation path that begins and ends at the corresponding endpoint samples. For $t \gtrapprox 50$, reconstructed samples corresponding to $\tilde{x}_t^0$ and $\tilde{x}_t^1$ no longer resemble the original input. For very large t, there appears to be no more change between the interpolation images. This is caused by the SNR, which gradually becomes too small to preserve or reconstruct any interpolation path. Figure 4.24 shows similar results for the Fashion-MNIST dataset.

For most examples, we could not identify interpolation paths that approximately minimize L.We obtained the best results for $30 < t < 40$, but there is no consistent pattern indicating

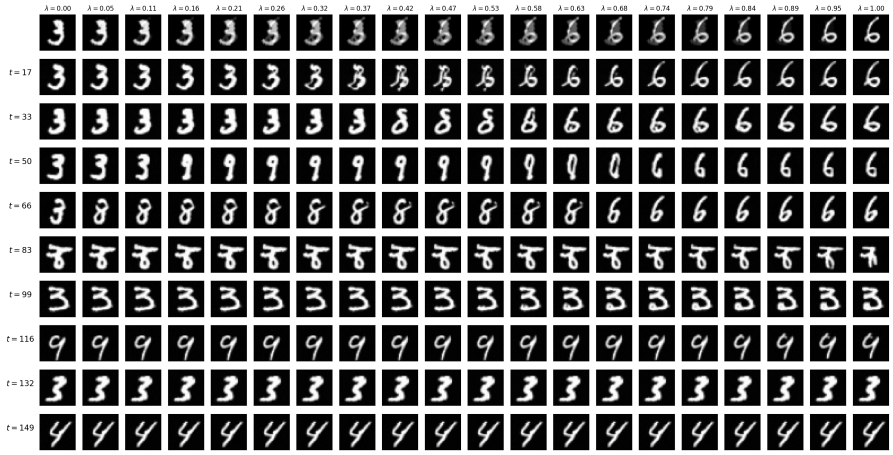

Figure 4.23: DDPM Latent Space Interpolation MNIST

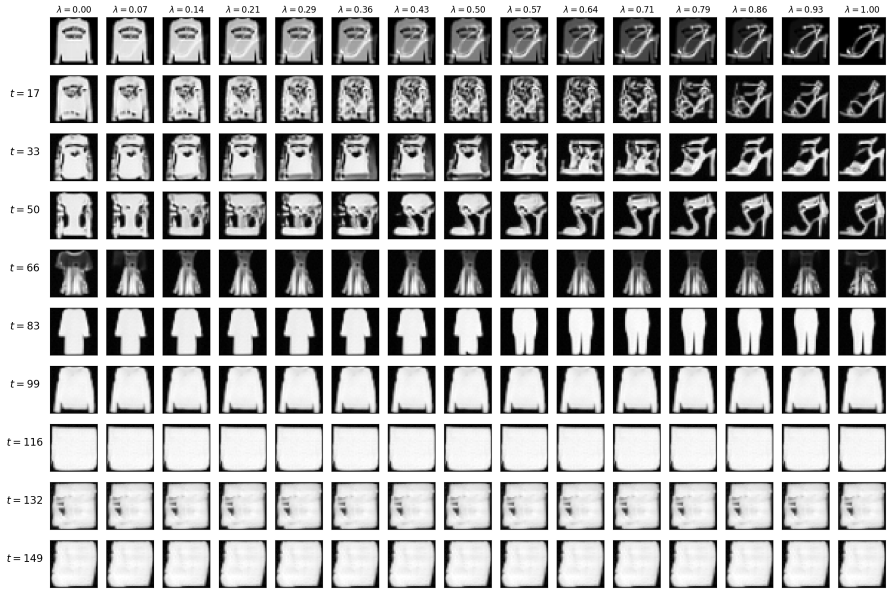

Figure 4.24: DDPM Latent Space Interpolation Fashion-MNIST

whether any particular values of t yield short interpolation paths. Further examples can be found in Appendix D.

Since the model does not explicitly encourages a geometric structure in latent space, interpolation between samples that are far apart in feature space can become unreliable. When interpolating between samples from the same class, interpolation in the DDPM latent space produces reasonably short interpolation paths, see Figure 4.25. The same behavior is observed for interpolating between samples with similar features and structure, see Figure 4.26. The

interpolation images show high visual quality with accurately recreating features of the original images.

Figure 4.25: DDPM Latent Space Interpolation from the same Class

Figure 4.26: DDPM Latent Space Interpolation from Sandal to Shoe

Modern diffusion models aim to encourage a smoother geometric structure in latent space to obtain shorter interpolation paths between samples with clear differences in features and structure, e.g., the Smooth Diffusion Model [7] and Stable Diffusion [10].

Chapter 5

Conclusion

A comparison of linear interpolation in the two models shows that the structured latent space of the VAE generally results in shorter interpolation paths than the DDPM for samples with distinctly different features, provided that the path does not cross a low-density region.

When comparing samples from similar classes, there is no clear tendency as to which model performs better. In the two-dimensional latent space of the VAE, interpolation may cross high-density regions corresponding to different samples, which can result in longer interpolation paths than observed in the DDPM. Further, one could argue that the generative sharpness in the DDPM is desirable. However, this might only hold for the low-dimensional latent space, since increasing dimensionality allows the model to learn more specific features of the input data. Additionally, correcting for the continuous Bernoulli normalization constant may further reduce this gap.

We further observe that increasing the dimensionality of the VAE latent space results in more clearly distinct high-density regions of the latent space and therefore leads to shorter interpolation paths. This reduced the previous observed issue of interpolation paths crossing too many different high-density region.

By switching to a second-degree polynomial with trainable parameter, we were able to circumvent low-density regions in the latent space by minimizing a combined loss defined in both latent and pixel space. This resulted in consistent interpolation path that avoided low-density regions. Consequently, we obtained shorter interpolation paths in the VAE, reducing the previously observed issue of traversing multiple unrelated high-density regions. This increases the amount of sample pairs for which we can consistently find reasonably short interpolation paths.

For future research, it would be valuable to investigate how interpolation behavior changes as the latent space dimension s of the VAE increases, particularly when using a polynomial of degree s for path parametrization. Additionally, it would be interesting to establish a evaluation measurement that is based on objective comparison of image quality (e.g. discriminator AI).

Recent research on diffusion models demonstrates that interpolation within these models produces highly realistic images. Diffusion models combined with VAEs and transformer architectures currently represent the state of the art in image synthesis (see [8], [10]).

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[2] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," arXiv:1312.6114, 2014. Available: https://arxiv.org/abs/1312.6114

[3] D. P. Kingma and M. Welling, "An Introduction to Variational Autoencoders," *Foundations and Trends in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019. doi: 10.1561/2200000056

[4] G. Loaiza-Ganem and J. P. Cunningham, "The Continuous Bernoulli: Fixing a Pervasive Error in Variational Autoencoders," arXiv:1907.06845, 2019. Available: https://arxiv.org/abs/1907.06845

[5] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," arXiv:2006.11239, 2020. Available: https://arxiv.org/abs/2006.11239

[6] J. Soch, "Kullback-Leibler divergence for the normal distribution," *The Book of Statistical Proofs*, Nov. 19, 2020. Available: https://statproofbook.github.io/P/norm-kl.html

[7] J. Guo, X. Xu, Y. Pu, Z. Ni, C. Wang, M. Vasu, S. Song, G. Huang, and H. Shi, "Smooth Diffusion: Crafting Smooth Latent Spaces in Diffusion Models," arXiv:2312.04410, Dec. 2023. Available: https://arxiv.org/abs/2312.04410

[8] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-Shot Text-to-Image Generation," arXiv:2102.12092, 2021. Available: https://arxiv.org/abs/2102.12092

[9] H. Petzka, T. Kronvall and C. Sminchisescu, "Discriminating Against Unrealistic Interpolations in Generative Adversarial Networks," arXiv:2203.01035, 2022. Available: https://arxiv.org/abs/2203.01035

[10] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," arXiv:2112.10752, 2021. Available: https://arxiv.org/abs/2112.10752

[11] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980, 2014. Available: https://arxiv.org/abs/1412.6980

[12] P. A. Bromiley, "Products and Quotients of Gaussian Probability Density Functions," *TINA-Vision Memo No. 2003-003*, 2003. Available: http://www.tina-vision.net/docs/memos/2003-003.pdf

[13] S. Gaikwad, T. Kasilingam, O. Ahmad, R. Mukherjee, and S. Bhowmick, "Deep Learning-Driven Prediction of Microstructure Evolution via Latent Space Interpolation," arXiv:2508.01822, 2025. Available: https://arxiv.org/abs/2508.01822

[14] L. Mi, T. He, C. F. Park, H. Wang, Y. Wang, and N. Shavit, "Revisiting Latent-Space Interpolation via a Quantitative Evaluation Framework," arXiv:2110.06421, 2021. Available: https://arxiv.org/abs/2110.06421

[15] P. Shamsolmoali, M. Zareapoor, H. Zhou, D. Tao, and X. Li, " VTAE: Variational Transformer Autoencoder with Manifolds Learning," arXiv:2304.00948, 2023. Available: https://arxiv.org/abs/2304.00948

[16] Y. LeCun and C. Cortes, "MNIST Handwritten Digit Database," 2010. Available: http://yann.lecun.com/exdb/mnist/

[17] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," arXiv:1708.07747, 2017. Available: https://arxiv.org/abs/1708.07747

Chapter 6

Appendix

A Derivation of Binary Cross Entropy in the ELBO

We want to examine $\mathbb{E}_{q(z|x)}(\log p(x|z))$. We start by investigating $\log p(x|z)$. For Fashion-MNIST or MNIST data, every pixel value is scaled to $[0, 1]$. Therefore, we assume that the reconstruction probability is $\mathbb{P}$("Each pixel is reconstructed with the correct intensity"). If we had a binary representation for each pixel, it would mean that every pixel would either be on ($x_i = 1$) or off ($x_i = 0$). In this case, we can write the reconstruction probability for a single pixel

$$p(x_i|z) = \hat{x}_{z,i}^{x_i} (1 - \hat{x}_{z,i})^{1-x_i}, \tag{A.1}$$

where x_i the value of pixel i in the sample and $\hat{x}_{z,i} \in [0, 1]$ denotes the decoder output for pixel i conditioned on z. We can assume that pixels are independent when conditioned on the latent variable z. This is because all the structural detail is assumed to be explained by the low-dimensional latent variable. Under the conditional independence assumption, this yields

$$p(x|z) = \prod_{i=1}^{d} \hat{x}_{z,i}^{x_i} (1 - \hat{x}_{z,i})^{1-x_i}. \tag{A.2}$$

Applying the logarithm gives

$$\log p(x|z) = \sum_{i=1}^{d} x_i \log \hat{x}_{z,i} + (1 - x_i) \log (1 - \hat{x}_{z,i}). \tag{A.3}$$

In the implementation, we use $x_i \in [0, 1]$ instead of $x_i \in \{0, 1\}$. This means that assuming a discrete Bernoulli distribution for $p(x_i|z)$ is theoretically invalid. We will however continue with A.3 with the support $x_i \in [0, 1]$ considering that most pixels in Fashion-MNIST and MNIST are approximately zero or one, and consequently limiting the practical impact of this theoretical inconsistency. For theoretical correctness, one should instead assume a continuous Bernoulli distribution, which can be seen as the generalization of the discrete Bernoulli to a continuous support on $[0, 1]$. This yields the distribution

$$p(x_i|z) = C(\hat{x}_{z,i}) \hat{x}_{z,i}^{x_i} (1 - \hat{x}_{z,i})^{1-x_i}. \tag{A.4}$$

This in practice can lead to better results. However, this improvement seems not to affect the performance of continuity or geometric structure of samples in the latent space, but rather the qualitative improvement of the samples sharpness, see [4]. We further note that this was discovered after the experimental results had been obtained, and correcting for it is left for future work.

It remains to compute $\mathbb{E}_{q(z|x)}(\log p(x|z))$. We cannot compute it directly, because the integral over $p(x|z)$ is intractable, see 2.1.2. We approximate the expectation as an average over L samples, meaning we compute

$$\mathbb{E}_{q(z|x)}(\log p(x|z)) \approx \frac{1}{L}\sum_{l=1}^{L}\log p(x|z^{(l)}), \tag{A.5}$$

where $z^{(l)} \sim q(z|x)$ is the encoded sample in the latent space obtained via the reparameterization trick 2.1.4. In [2], it is stated that choosing $L = 1$ works for large enough minibatches. Setting $L = 1$, we simplify A.5 to

$$\mathbb{E}_{q(z|x)}(\log p(x|z)) \approx \log p(x|z^{(1)}) = \sum_{i=1}^{d} x_i \log \hat{x}_{z^{(1)},i} + (1-x_i)\log\left(1-\hat{x}_{z^{(1)},i}\right). \tag{A.6}$$

This is equivalent to the negative BCE between a sample and its reconstruction.

B Extended Derivation Reparameterization

We want to find the conditional distribution of $x_t|x_0$ in 2.2.11. Since x_t can be expressed as linear combination of independent Gaussians random variables, $\epsilon_s \overset{i.i.d.}{\sim} \mathcal{N}(0,1)$ for $0 \le s \le t$, $x_t|x_0$ is also Gaussian. It suffices to compute its expectation and covariance to get the distribution.

$$\mathbb{E}[x_t|x_0] = \mathbb{E}\left[\sqrt{\bar{\alpha}}x_0 + \sum_{s=1}^{t}\left(\sqrt{1-\alpha_s}\frac{\sqrt{\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_s}}\right)\epsilon_s|x_0\right] = \sqrt{\bar{\alpha}}x_0 \tag{B.1}$$

(B.2)

Similar to 2.2.2, each random variable ϵ_s satisfies $\mathbb{E}[\epsilon_s] = 0$.

(B.3)

$$\mathrm{Cov}[x_t|x_{t-1}] = \mathbb{E}\left[[x_t - \mathbb{E}[x_t|x_0]]^2|x_0\right] = \mathbb{E}\left[\left[\sum_{s=1}^{t}\left(\sqrt{1-\alpha_s}\frac{\sqrt{\bar{\alpha}_t}}{\sqrt{\bar{\alpha}_s}}\right)\epsilon_s\right]^2|x_0\right]$$

(B.4)

$$= \sum_{s=1}^{t}\left((1-\alpha_s)\frac{\bar{\alpha}_t}{\bar{\alpha}_s}\mathbb{E}\left[\epsilon_s^2\right]\right) = \sum_{s=1}^{t}\left((1-\alpha_s)\frac{\bar{\alpha}_t}{\bar{\alpha}_s}\left[\mathrm{Var}\,\epsilon_t - [\mathbb{E}(\epsilon_t)]^2\right]\right) = \sum_{s=1}^{t}\left((1-\alpha_s)\frac{\bar{\alpha}_t}{\bar{\alpha}_s}\right)\mathbf{I}$$

Next, we simplify the sum for $t \ge 1$.

$$t = 1: \qquad (1-\alpha_1)\frac{\alpha_1}{\alpha_1} = (1-\alpha_1)$$

$$t = 2: \qquad (1-\alpha_1)\frac{\alpha_1\alpha_2}{\alpha_1} + (1-\alpha_2)\frac{\alpha_1\alpha_2}{\alpha_1\alpha_2} = (1-\alpha_2) + \alpha_2(1-\alpha_1) = 1-\alpha_2\alpha_1$$

$$t = 3: \qquad (1-\alpha_3) + \alpha_2(1-\alpha_2) + \alpha_3\alpha_2(1-\alpha_1) = 1-\alpha_3\alpha_2\alpha_1$$

This is a telescoping sum. Hence, for any t, we obtain:

$$\sum_{s=1}^{t}\left((1-\alpha_s)\frac{\bar{\alpha}_t}{\bar{\alpha}_s}\right) = 1-\alpha_t\alpha_{t-1}...\alpha_1 = 1-\bar{\alpha}_t \tag{B.5}$$

Consequently, the resulting distribution is:

$$x_t \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)\mathbf{I}\right), \tag{B.6}$$

which coincides with equation 2.2.12 when conditioned on x_0. Hence, the simplified expression is verified.

C Computing the Resulting Gaussian Density

The application of Bayes' theorem in 2.2.20 simplifies to a proportionality constant c, since the denominator does not depend on x_{t-1} and is therefore constant with respect to this variable. The proportionality constant is determined implicitly by the requirement that the conditional density integrates to one. Both $q(x_t|x_{t-1})$ and $q(x_{t-1}|x_0)$ are Gaussian distributions. In particular:

$q(x_t|x_{t-1}) = \mathcal{N}\left(\sqrt{\alpha_t}x_{t-1}, \beta_t\mathbf{I}\right)$, with corresponding density $g(x_t|x_{t-1})$, (C.1)

$q(x_{t-1}|x_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}x_0, (1-\bar{\alpha}_{t-1})\mathbf{I}\right)$, with corresponding density $f(x_{t-1}|x_0)$. (C.2)

Next, we express the density in C.1 as a function of x_{t-1}. As discussed previously, we do not explicitly track the proportionality constant, but instead concentrate on the quadratic term in the exponent of the Gaussian density.

$$g(x_t|x_{t-1}) = c\exp -\left(\frac{\left(x_t-\sqrt{\alpha_t}x_{t-1}\right)^2}{2\beta_t}\right) = c\exp -\left(\frac{\alpha_t}{2\beta_t}\left(x_{t-1}-\frac{x_t}{\sqrt{\alpha_t}}\right)^2\right) \tag{C.3}$$

It suffices to show that the product of densities $q(x_t|x_{t-1})$ and $q(x_{t-1}|x_0)$ is proportional to the density of $q(x_{t-1}|x_t, x_0)$. Consequently, we seek functions f and g such that

$$q(x_{t-1}|x_t, x_0) \propto g(x_t|x_{t-1})f(x_{t-1}|x_0). \tag{C.4}$$

The means and covariance matrices of the two Gaussian densities, for fixed x_t and x_0, are given by:

$$\mu_g = \frac{x_t}{\sqrt{\alpha_t}} \text{ and } \Sigma_g^2 = \frac{\beta_t}{\alpha_t}\mathbf{I} \tag{C.5}$$

$$\mu_f = \sqrt{\bar{\alpha}_{t-1}}x_0 \text{ and } \Sigma_f^2 = (1-\bar{\alpha}_{t-1})\mathbf{I} \tag{C.6}$$

We use the fact that the product of two Gaussian densities is proportional to another Gaussian density, see [12]. In particular, we use the formula given in that paper to find the mean and the variance of the resulting Gaussian.

$$\Sigma_f^2 + \Sigma_g^2 = \frac{\beta_t}{\alpha_t}\mathbf{I} + (1 - \bar{\alpha}_{t-1})\,\mathbf{I} = \frac{1 - \bar{\alpha}_t}{\alpha_t}\mathbf{I} \tag{C.7}$$

$$\Sigma_{fg}^2 = \frac{\Sigma_f^2 \Sigma_g^2}{\Sigma_f^2 + \Sigma_g^2} = \frac{1 - \bar{\alpha}_{t-1}}{1 - \bar{\alpha}_t}\beta_t \mathbf{I} \tag{C.8}$$

$$\mu_{fg} = \frac{\mu_g \Sigma_f^2 + \mu_f \Sigma_g^2}{\Sigma_f^2 + \Sigma_g^2} = \frac{\sqrt{\alpha_t}\,(1 - \bar{\alpha}_{t-1})}{1 - \bar{\alpha}_t} x_t + \frac{\sqrt{1 - \bar{\alpha}_{t-1}}\beta_t}{1 - \bar{\alpha}_t} x_0 \tag{C.9}$$

We have shown that $q(x_{t-1}|x_t, x_0)$ is proportional to a Gaussian density in x_{t-1}. Consequently, after normalization $q(x_{t-1}|x_t, x_0)$ is Gaussian. The proportionality constant corresponds to the normalization factor required to ensure that the density integrates to one.

D DDPM Image Interpolation

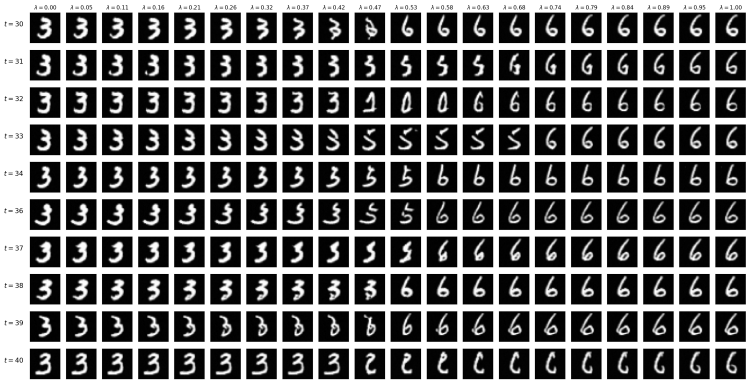

Figure 1: DDPM Latent Interpolation MNIST

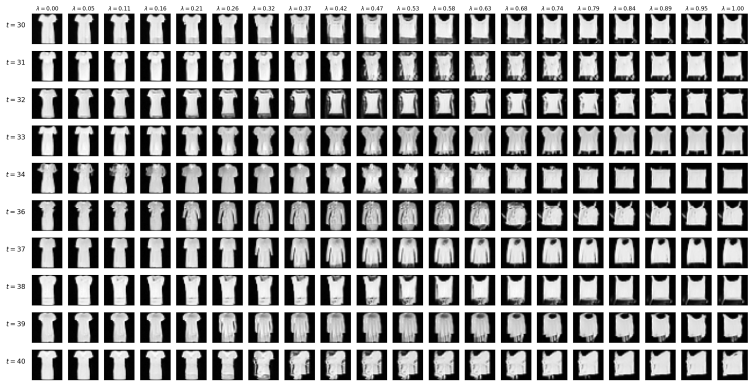

Figure 2: DDPM Latent Interpolation Fashion-MNIST