Variational Autoencoders

SAiDL 2024 Assignment

Sasmit Datta

1 Introduction

Variational Autoencoders (VAEs) represents a seminal approach in the field of generative models, allowing the learning of complex data distributions like images through latent space exploration. This report entails an analysis of two distinct VAEs with the same architecture but different prior distributions: $\mathcal{N}(0, 1)$ and $\mathcal{N}(1, 2)$.

Furthermore, I try to improve generations from $\mathcal{N}(1, 2)$ by deployment of various loss functions - **BCE**, **MAE**, **MSE**, **perceptual loss**, and **weighted perceptual loss with additional loss term** and other methods like changing initialization.

I also implement a **FID Score Calculator** to compare the quality of generations from different models.

2 VAEs

VAEs consist of an encoder which learns to map data to a latent distribution and a decoder that reconstructs data from this distribution.

2.1 Formulation

Given data $\mathbf{x}$ with latent variables $\mathbf{z}$, the encoder approximates the true posterior $p(\mathbf{z}|\mathbf{x})$ with a variational approximation $q_\phi(\mathbf{z}|\mathbf{x})$. The decoder then reconstructs the data from the latent space, modeling $p_\theta(\mathbf{x}|\mathbf{z})$. The VAE is trained by maximizing the evidence lower bound (ELBO) on the marginal likelihood of each data point:

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = -\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$$

Here, D_{KL} denotes the Kullback-Leibler divergence, providing conditioning by penalizing the deviation of $q_\phi(\mathbf{z}|\mathbf{x})$ from the prior $p(\mathbf{z})$. So when we pass samples from our prior $p(\mathbf{z})$ to the decoder, we can generate novel data.

2.2 Reparameterization Trick

The reparameterization trick enables gradient-based optimization of VAEs by expressing the random variable $\mathbf{z}$ as a deterministic function of $\mathbf{x}$ and random variable ϵ.

3 Model

3.1 Architecture

Fig 1 is the architecture of the model used.

3.2 Common Hyper-parameters

Common hyper-parameters I use to train these models are:

- Optimizer: **Adam** with default Pytorch settings except for the learning rate.
- Learning Rate = 0.0001
- Epochs = 15

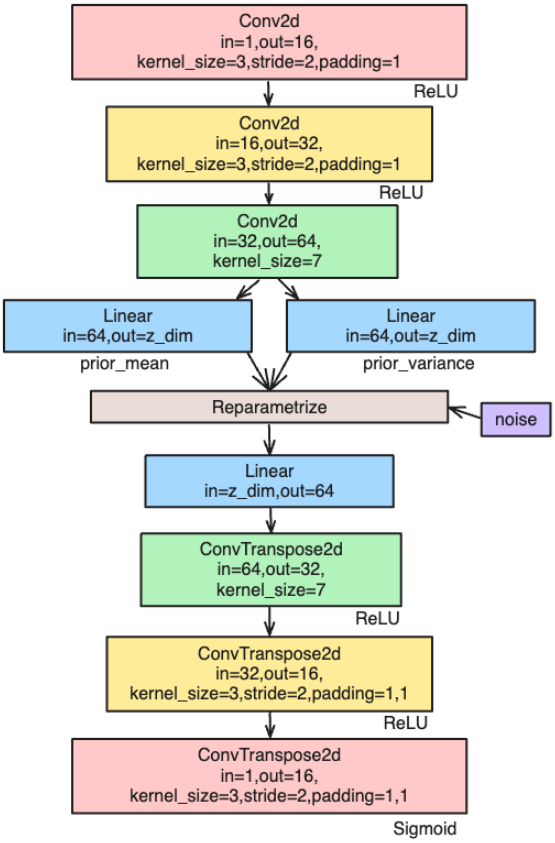

Figure 1: VAE Architecture

- Batch Size = 64
- Latent Dimension = 2

4 Perceptual Loss

Perceptual loss evaluates the similarity between images using high-level feature representations from a pre-trained CNN, focusing on perceptual rather than pixel-wise differences.

Perceptual loss is defined as the difference in feature representations from a CNN between a generated image $\hat{\mathbf{x}}$ and a real image $\mathbf{x}$:

$$\mathcal{L}_{\text{perceptual}}(\mathbf{x}, \hat{\mathbf{x}}) = \sum_{l} \|\Phi_l(\mathbf{x}) - \Phi_l(\hat{\mathbf{x}})\|_2^2 \tag{1}$$

where Φ_l represents the activation of the l-th layer of the CNN.

4.1 Weighted Perceptual Loss with Additional Loss Term

Given reconstructed image $\hat{\mathbf{x}}$ and a target image $\mathbf{x}$, the weighted perceptual loss incorporating an additional loss term can be formulated as:

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \lambda \sum_{l} \|\Phi_l(\mathbf{x}) - \Phi_l(\hat{\mathbf{x}})\|_2^2 + \mathcal{L}_{\text{additional}}(\mathbf{x}, \hat{\mathbf{x}}) \tag{2}$$

where λ is the weighting factor, Φ_l denotes the activation of the l-th layer in a pre-trained CNN, and $\mathcal{L}_{\text{additional}}$ represents the additional loss term, such as binary cross-entropy (BCE) or mean squared error (MSE).

4.2 Implementation

For the pre-trained CNN, I use a straightforward MNIST classifier (Fig 2) trained over the course of **5 epochs** using an **Adam optimizer** with **learning rate** set to **0.001** and a **batch size** of **64**, getting a **99% accuracy** on the test set. I use the features of **2'nd** and **4'th ReLU activations** of this model to calculate the perceptual loss. $\boldsymbol{\lambda}$ is set to **0.01** for the **weighted perceptual loss with an additional loss term** in my experiments.

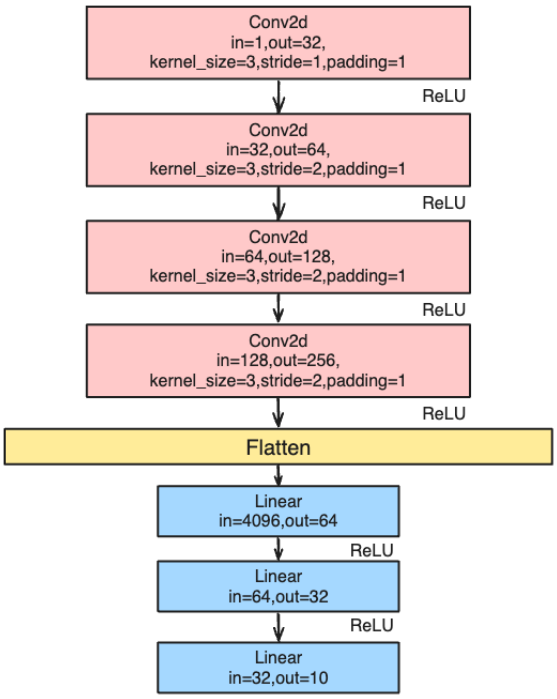

Figure 2: Perceptual Loss MNIST Classifier

5 Baselines

I used the `rsample()` method of `torch.distributions.normal.Normal()` to sample latent vectors, the native KL-Divergence method to calculate $D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ and BCE Loss for $\mathcal{N}(0,1)$ and $\mathcal{N}(1,2)$ baseline models.

$$\text{BCE} = -\sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right], \tag{3}$$

where N is the number of pixels in the image, y_i represents the true pixel values of the original image, and $\hat{y}_i$ denotes the predicted pixel values from the reconstructed image. All the latent space plots can be found in Section 12.

5.1 $\mathcal{N}(0, 1)$ Prior

For an isotropic gaussian prior, and a gaussian posterior whose parameters we have to predict, $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ can be written as:

$$D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) = -\frac{1}{2}\sum_{j=1}^{n}\left[1 + \log\sigma_j^2 - \sigma_j^2 - \mu_j^2\right] \tag{4}$$

where n is the dimension of the latent space. The model's interpolation and latent space plots can be found in Fig 3 and Fig 15 respectively.

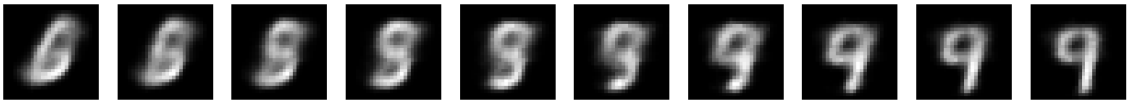

Figure 3: Interpolation: $\mathcal{N}(0, 1)$ Prior

5.2 $\mathcal{N}(1, 2)$ Prior

For a $\mathcal{N}(1, 2)$ prior, and a gaussian posterior whose parameters we have to predict, $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ can be written as:

$$D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})) = -\frac{1}{2}\sum_{j=1}^{n}\left[1 - 2\log 2 + \log\sigma_j^2 - \frac{1}{4}\sigma_j^2 - \frac{1}{4}(1-\mu_j)^2\right] \tag{5}$$

where n is the dimension of the latent space. Derivation of the above result can be found in Section 11. The model's interpolation and latent space plots can be found in Fig 4 and Fig 16 respectively.

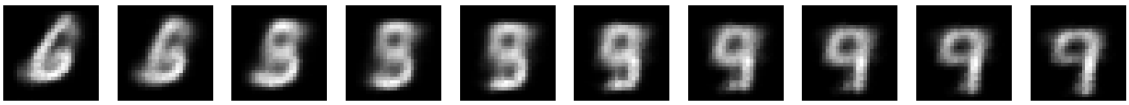

Figure 4: Interpolation: $\mathcal{N}(1, 2)$ Prior

Both the VAEs visually seem to have the same performance producing strong and sharp 0's and 1's and distinguishable 9's while losing its performance for other digits.

6 Improving Generations

6.1 Kaiming Normal Initializaton

Kaiming Normal initialization optimizes weight initialization for layers with ReLU activations, preventing gradient issues in deep neural networks. Kaiming Normal initialization sets the initial weights W from a normal distribution $W \sim \mathcal{N}(0, \sigma^2)$ where:

$$\sigma = \sqrt{\frac{2}{n_{\text{in}}}} \tag{6}$$

where n_{in} represents the number of input units to a layer.

The latent space generations and interpolations seem to be slightly better due to the encoder and decoder both having ReLU activations for its layers. The subsequent models are initialized this way. (Latent space plot: Fig 17).

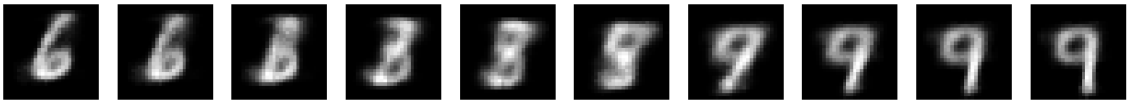

Figure 5: Interpolation: Kaiming Initialization

6.2 MSE Loss as Reconstruction Loss

Mean Squared Error (MSE) loss in Variational Autoencoders (VAEs) quantifies the pixel-wise square of the difference between the original and reconstructed data.

$$\text{MSE} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 \tag{7}$$

where $\mathbf{x}$ is the original data point, and $\hat{\mathbf{x}}$ is the reconstructed data point.

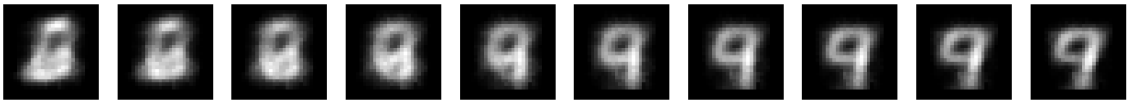

Figure 6: Interpolation: MSE Loss

The generations (Fig 6 and Fig 18) are blurrier and more faded than the ones of BCE Loss. This can be attributed to the fact that the square of the pixel-wise error can lead to a smoother gradient in the loss landscape, thus minimizing the penalty for small deviations.

6.3 MAE Loss as Reconstruction Loss

The Mean Absolute Error (MAE) Loss, also known as L1 loss, is defined to quantify the average magnitude of the absolute differences between original and reconstructed data. It is formulated as:

$$\text{MAE} = \sum_{i=1}^{N} |\mathbf{x}_i - \hat{\mathbf{x}}_i|, \tag{8}$$

where N denotes the total number of pixels in an image, $\mathbf{x}$ represents the original image, and $\hat{\mathbf{x}}$ denotes the reconstructed image.

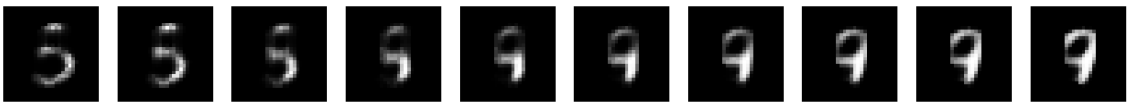

Figure 7: Interpolation: MAE Loss

Samples (Fig 7) using MAE Loss are sharper as MAE directly measures absolute error. Since we are working with values between 0 and 1, it avoids the diminishing effect MSE has due to squaring.

MAE's insensitivity to outliers, unlike BCE and MSE, results in less varied, detailed reconstructions, as it tends to gloss over the finer details that contribute significantly to the overall image quality. (Latent space plot: 19).

6.4 Perceptual Loss as Reconstruction Loss

Perceptual loss (Fig 8 and Fig 20) alone fails to give good samples of our digits since it prioritizes perceptual, higher-level features over pixel-wise reconstruction. Therefore, I paired it with BCE and MSE losses to get better samples.

Figure 8: Interpolation: Perceptual Loss

6.5 Weighted Perceptual with BCE Loss as Reconstruction Loss

Pairing BCE Loss with Perceptual Loss gives us more coherent samples with lesser artefacts (Fig 9). It also is able to produce more varied samples unlike BCE Loss - good 7's and even 2's (Fig 21) since the perceptual loss component is able to help the model focus on the overall structure of the digits and not only the pixels it consists of.

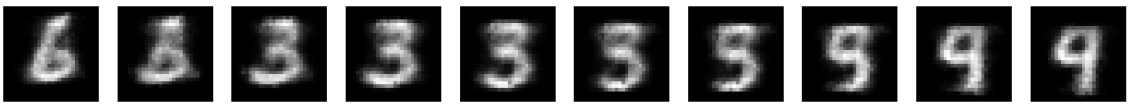

Figure 9: Interpolation: Perceptual Loss with BCE

6.6 Weighted Perceptual Loss with MSE Loss as Reconstruction Loss

Like perceptual loss paired with BCE, perceptual loss with MSE is able to produce varied results. But its generations are more blurred and less distinct like the generations from the model with just MSE Loss.

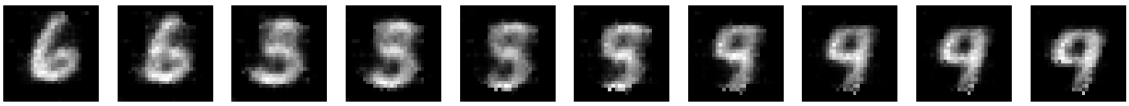

Figure 10: Interpolation: Perceptual Loss with MSE

6.7 BCE Loss for MNIST

For many image reconstruction tasks, MSE Loss is the go-to choice due to its effectiveness at reducing pixel-level intensity differences. However, for the MNIST dataset, with its images that are essentially binary, BCE Loss is favored. This preference stems from BCE's proficiency in binary classification challenges, making it superior to MSE for distinguishing the stark, black-and-white contrast inherent in MNIST digits.

7 Reconstructions

I compare reconstructions of BCE Loss (Fig 11) and Perceptual Loss with BCE (Fig 12) in this section. Like discussed in the previous section, perceptual loss makes model generation more varied.

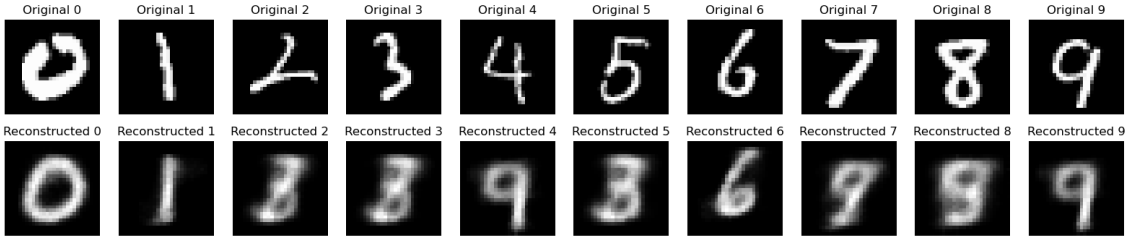

Figure 11: Reconstruction: BCE Loss

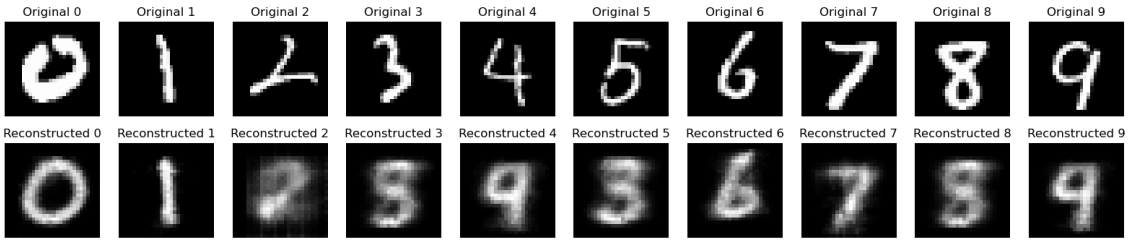

Figure 12: Reconstruction: Perceptual Loss with BCE

8 Higher Dimensional Latent Spaces

Two-dimensional latent space offers a very compact representation which may be too limited for capturing the complexity of the dataset. So, I trained a VAE with BCE Loss on a 10 dimensional latent space..

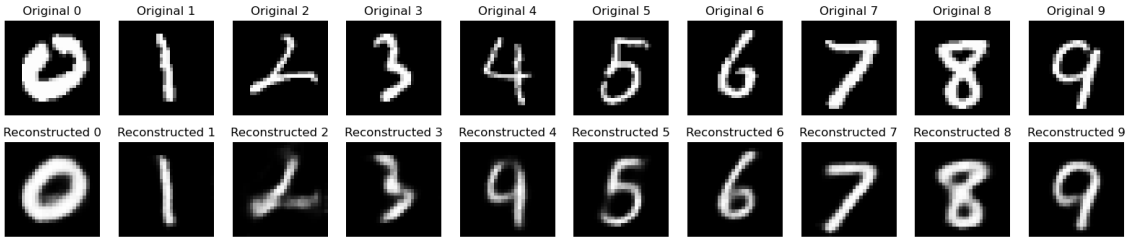

Figure 13: Reconstruction: BCE Loss - 10D Latent Space

Reconstructions improves dramatically (Fig 13 and Fig 14) with almost negligible parameters hike - 0.546%. Original parameter count is 210,821.

Figure 14: Interpolation: BCE Loss - 10D Latent Space

9 FID Score

The Fréchet Inception Distance (FID) score quantifies the similarity between generated and real images, utilizing feature vectors extracted from an intermediate layer of the Inception v3 network. A lower FID indicates closer resemblance a more realistic generation. The FID score is calculated as:

$$\text{FID} = \|\boldsymbol{\mu_r} - \boldsymbol{\mu_g}\|_2^2 + \text{Tr}(\boldsymbol{\Sigma_r} + \boldsymbol{\Sigma_g} - 2(\boldsymbol{\Sigma_r}\boldsymbol{\Sigma_g})^{1/2}), \tag{9}$$

where $\boldsymbol{\mu_r}$, $\boldsymbol{\Sigma_r}$ and $\boldsymbol{\mu_g}$, $\boldsymbol{\Sigma_g}$ are the mean vectors and covariance matrices of the real and generated images' feature vectors from the Inception v3 network's intermediate layer, respectively.

9.1 Implementation

I used the **activations** of the **Mixed-7c layer** of the Inception v3 model to get feature maps of our images.

For each VAE, I **generated random samples** the size of the MNIST test dataset which is **10,000 images**, **resized** them to **299×299**, **normalize** the values to be be **between -1 and 1**, added two extra channels with the same pixels and got our outputs from the model.

9.2 Results

VAE Model	FID Score
$\mathcal{N}(0, 1)$	131.7012
$\mathcal{N}(1, 2)$	132.0186
Kaiming Initialization - BCE Loss	150.64744
MSE Loss	143.3465
MAE Loss	98.2500
Perceptual Loss	323.5575
Perceptual Loss - BCE Loss	180.7325
Perceptual Loss - MSE Loss	176.8897
10D Latent Space - BCE Loss	79.7245

Table 1: FID Scores of All Models

10 Conclusion

There are a host of other strategies like beta-VAEs, adding noise to the image, using skip connections in forward functions that we can use to get better generations.

These were just some popular techniques I used in an attempt to improve samples from $\mathcal{N}(1, 2)$ distribution.

11 KL Divergence with $\mathcal{N}(1, 2)$ Prior

Equation for multivariate gaussian:

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{2^{n/2}|\boldsymbol{\Sigma}|} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\intercal \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right) \tag{10}$$

where $\mathbf{x}, \boldsymbol{\mu} \in \mathbb{R}^n$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$.

We assume that the mean is vector of 1's and the covariance is a diagonal matrix of 4's. (I am taking second parameter of $\mathcal{N}$ to be variance in this derivation, as opposed to the standard practice in this report). Therefore:

$$p_1 = q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are the mean and standard deviation vectors of the surrogate posterior, and

$$p_2 = p_\theta(\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) = \mathcal{N}(\mathbf{1}, \mathbf{4})$$

Therefore, $D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right)$ can be written as

$$\begin{aligned}
&\frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} - n + \mathrm{tr}(\boldsymbol{\Sigma}_2^{-1}\boldsymbol{\Sigma}_1) + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\intercal \boldsymbol{\Sigma}_2^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)\right] \\
&= \frac{1}{2}\left[\log\frac{2^{2n}}{|\boldsymbol{\Sigma}_1|} - n + \frac{1}{4}\mathrm{tr}(\boldsymbol{\Sigma}_1) + \frac{1}{4}(\mathbf{1} - \boldsymbol{\mu})^\intercal(\mathbf{1} - \boldsymbol{\mu})\right] \\
&= \frac{1}{2}\left[2n\log 2 - \sum_{j=1}^{n}\log\sigma_j^2 - n + \frac{1}{4}\sum_{j=1}^{n}\sigma_j^2 + \frac{1}{4}\sum_{j=1}^{n}(1 - \mu_j)^2\right] \\
&= -\frac{1}{2}\sum_{j=1}^{n}\left[1 - 2\log 2 + \log\sigma_j^2 - \frac{1}{4}\sigma_j^2 - \frac{1}{4}(1 - \mu_j)^2\right]
\end{aligned}$$

Hence, proved.

12 Latent Space Plots

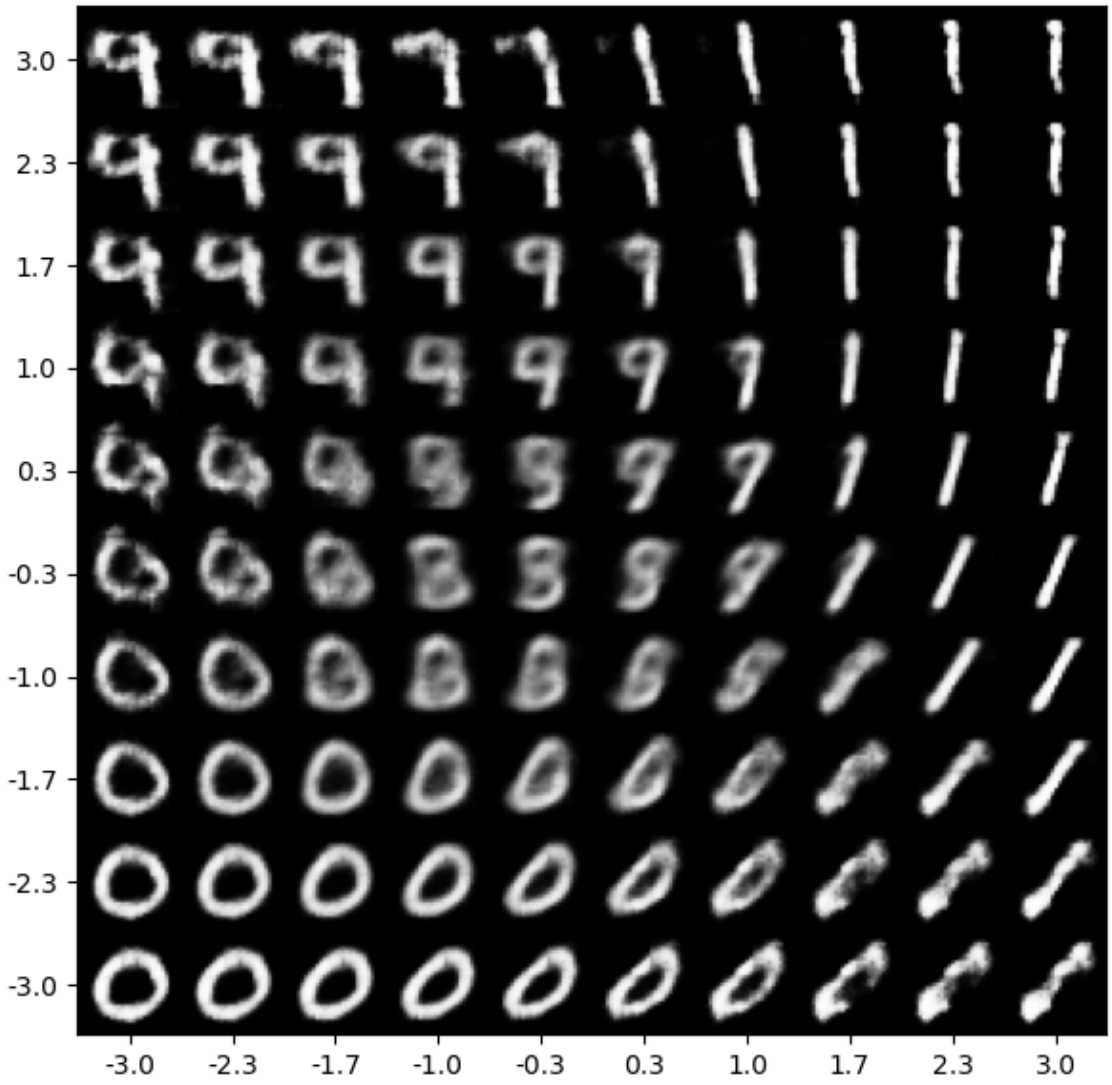

Figure 15: Latent Space: $\mathcal{N}(0, 1)$ Prior

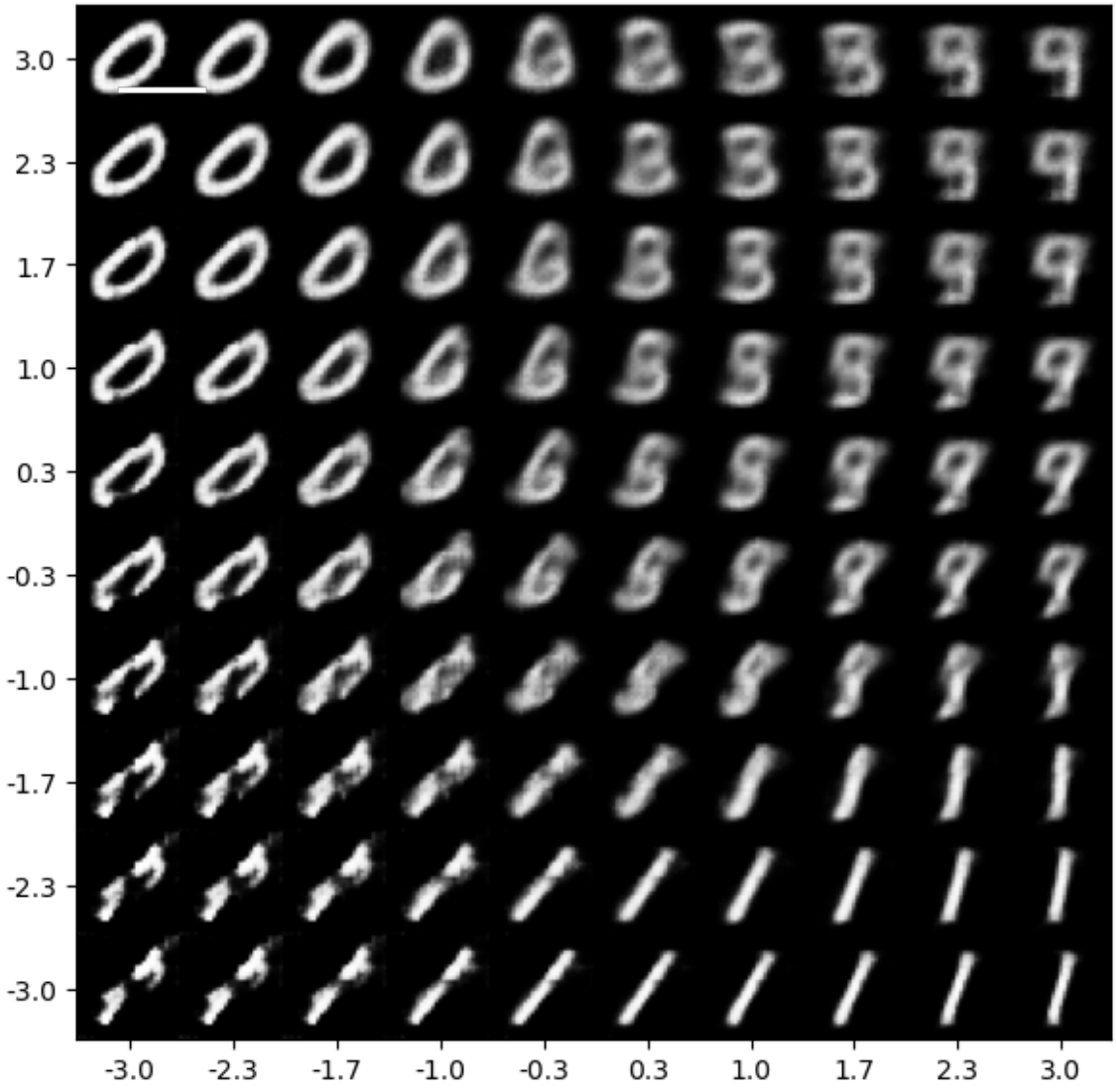

Figure 16: Latent Space: $\mathcal{N}(1, 2)$ Prior

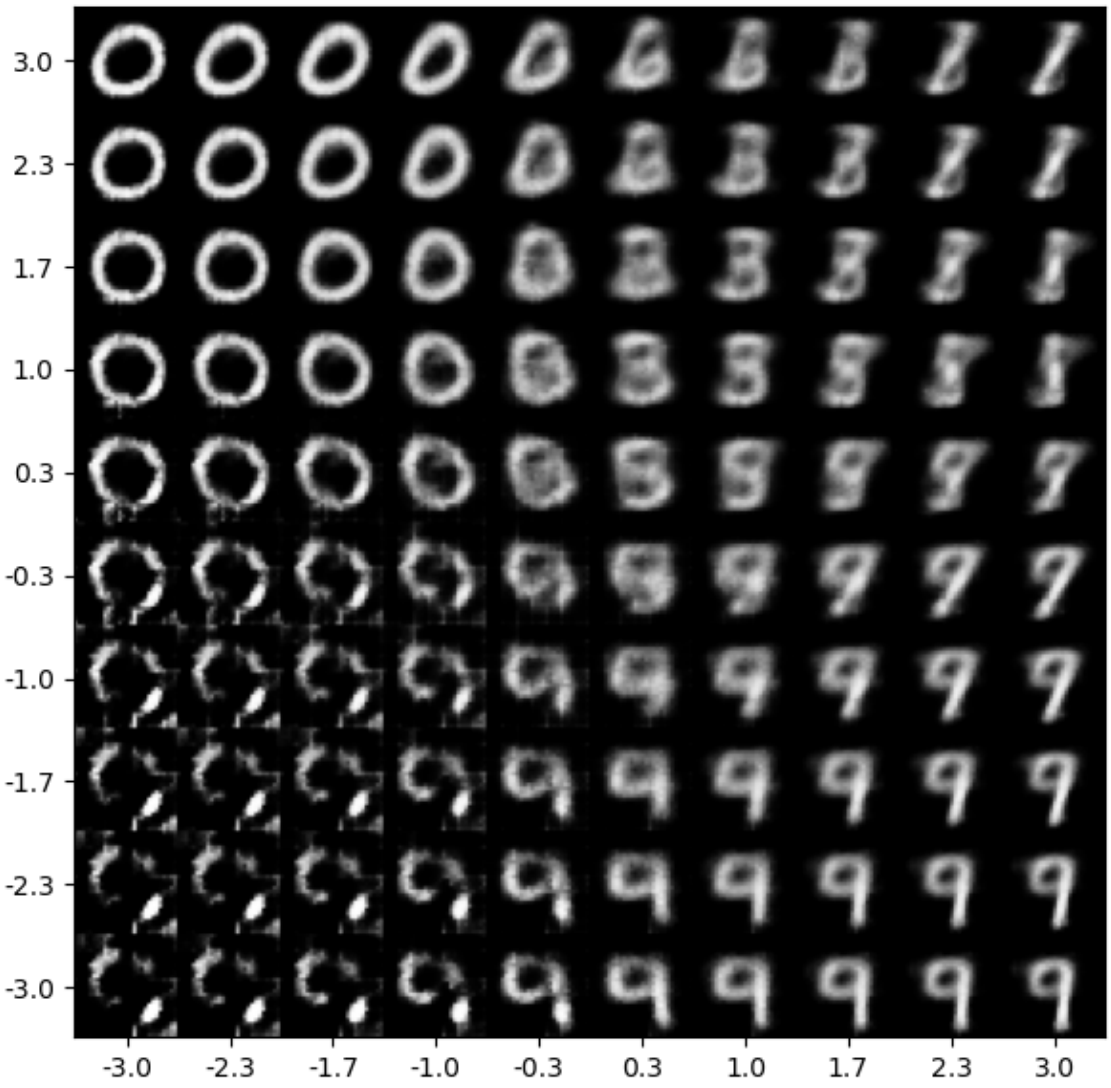

Figure 17: Latent Space: Kaiming Initialization with BCE Loss

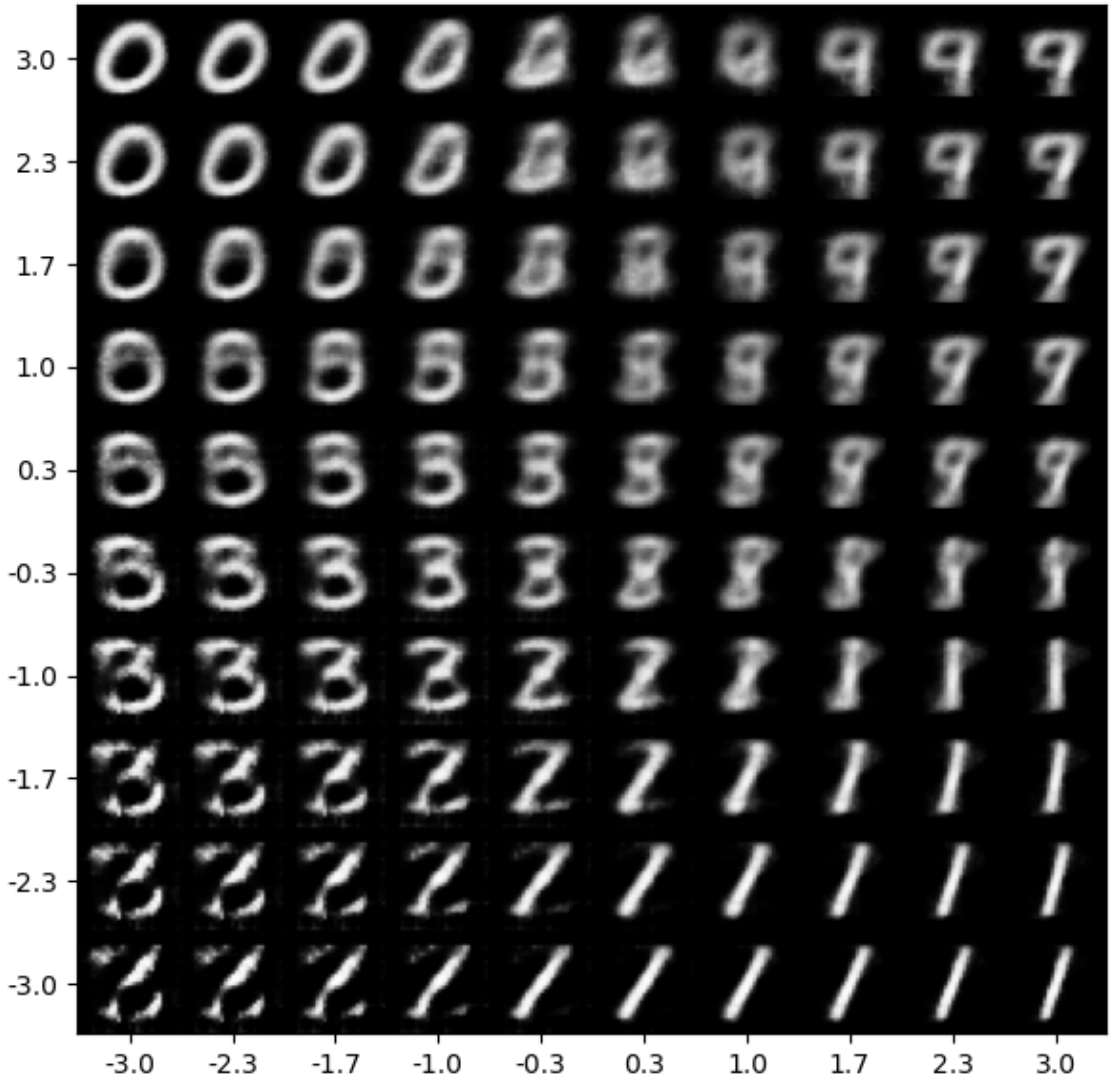

Figure 18: Latent Space: MSE Loss

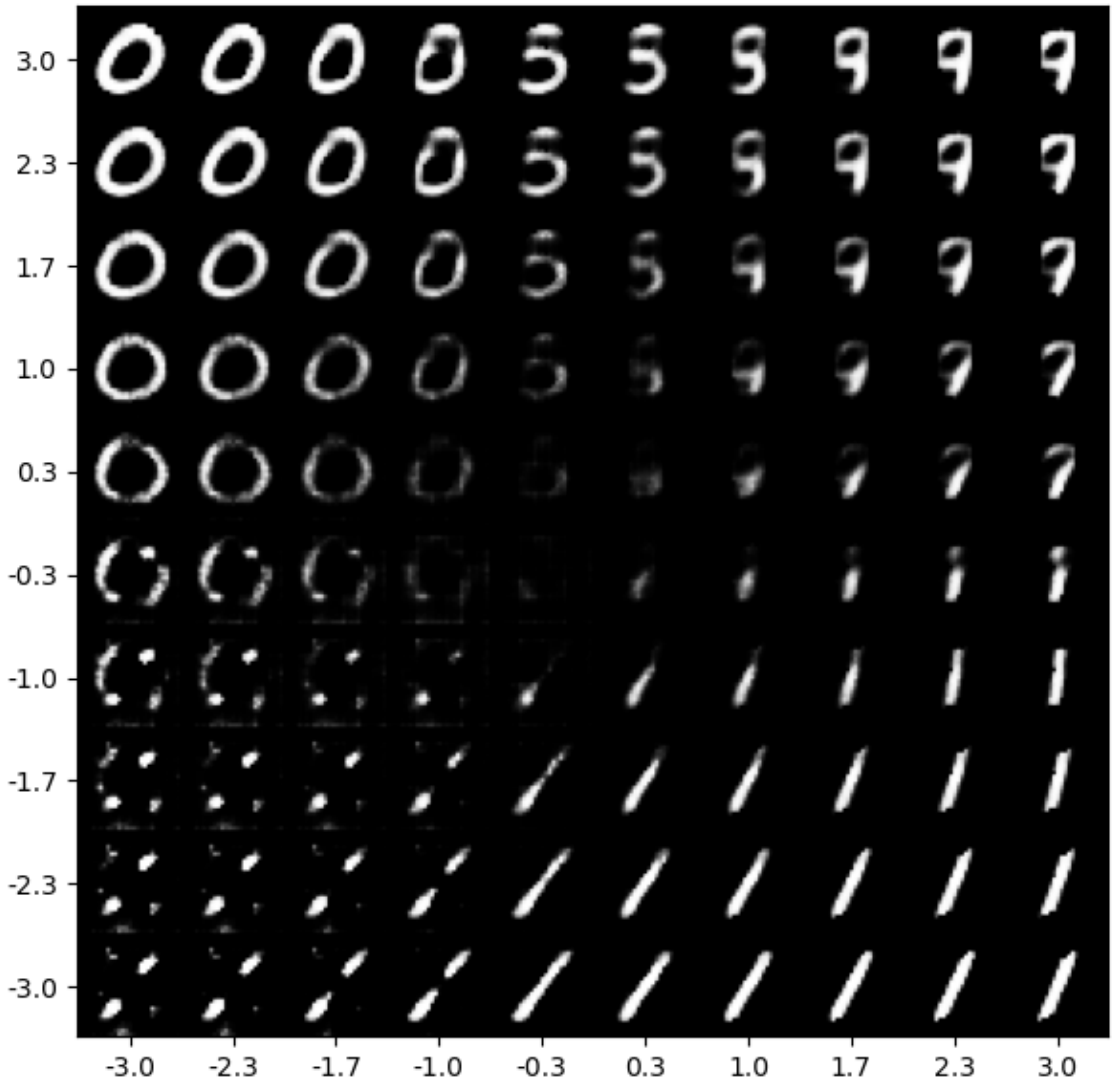

Figure 19: Latent Space: MAE Loss

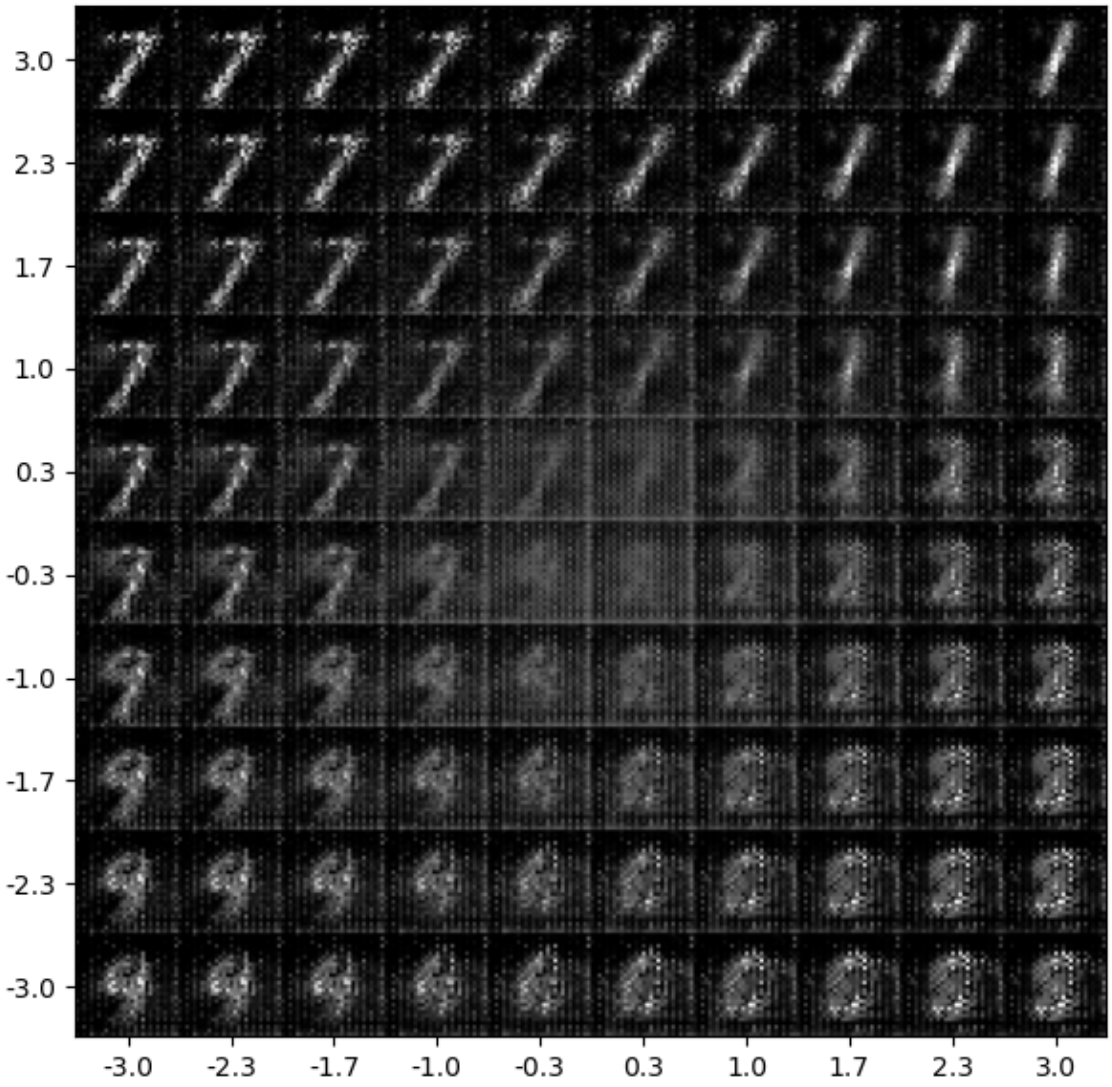

Figure 20: Latent Space: Perceptual Loss

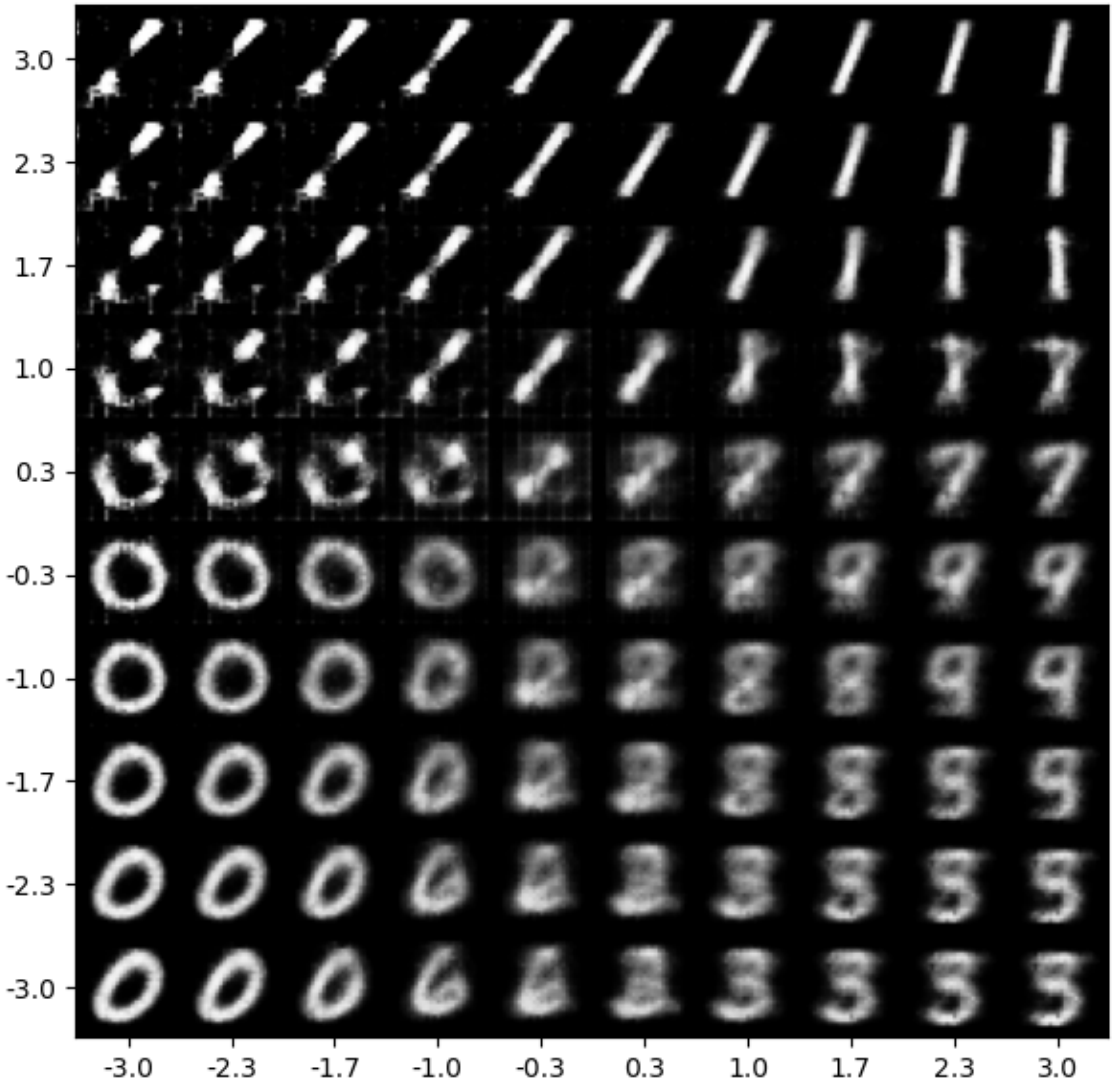

Figure 21: Latent Space: Perceptual Loss with BCE

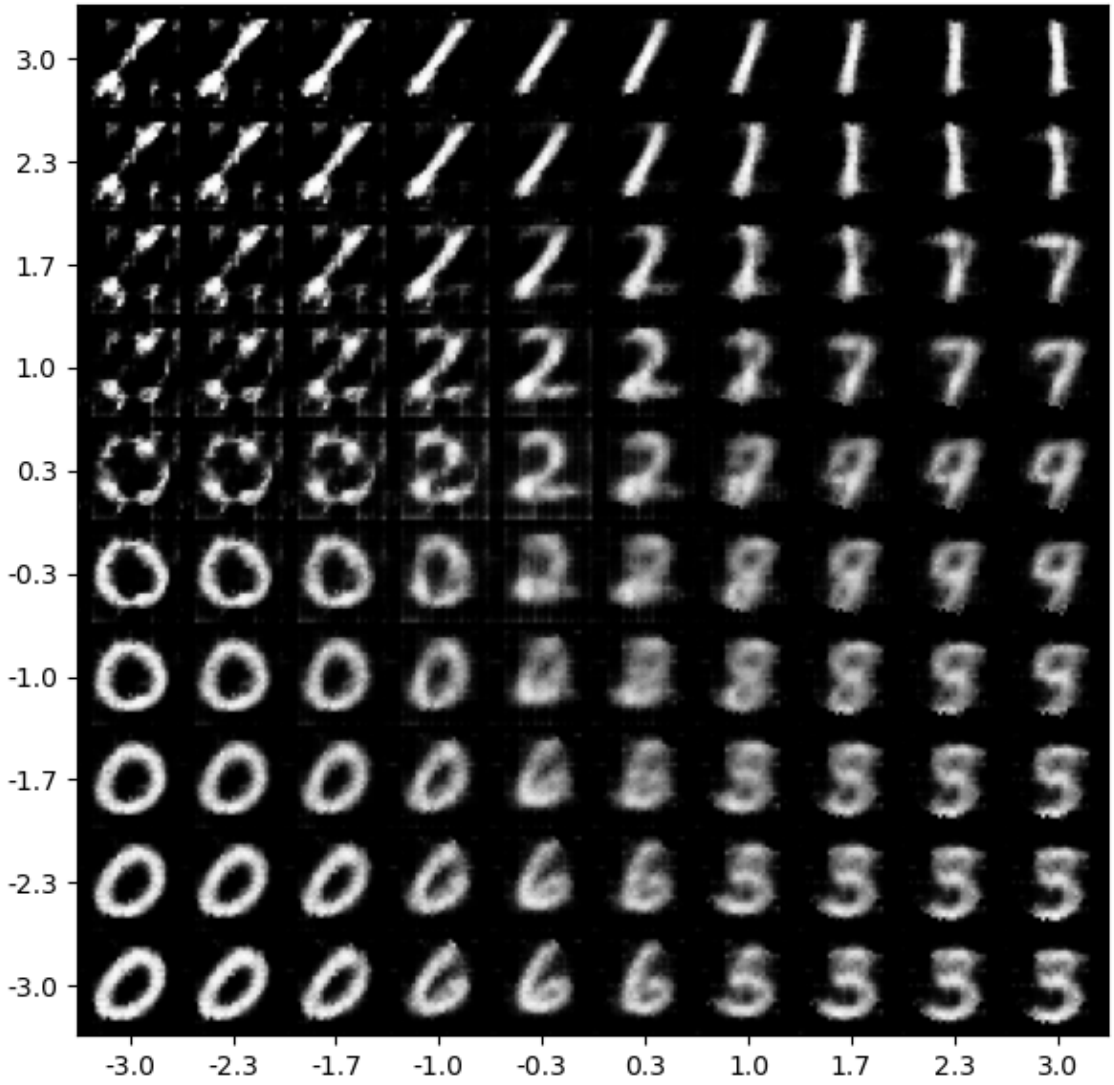

Figure 22: Latent Space: Perceptual Loss with MSE