Project 1.2 Variational Autoencoders

Ellen Edvinsson, Oscar Granström,
Anton Johansson and Fredrik Lindholm

November 11, 2025

Abstract

This project explores Autoencoders (AE) and Variational Autoencoders (VAE) with focus on implementation and performance using the MNIST dataset. Building on the framework introduced by [Kigma, D, P and Welling, M. 2013], a baseline AE was implemented, and a VAE introduced a probabilistic component for reconstruction and data generation. Training with latent dimensions showed the trade-off between reconstruction accuracy and generalization, with diminishing returns for higher dimensions. The results align with theoretical expectations and provide insight into the models' applications and limitations.

1 Introduction

1.1 Vanilla Autoencoders

Autoencoders efficiently compress data into latent variables which consist of fewer variables than the original input. These can then be decoded into representations of the original input. This process consists of two parts, first the encoding part and secondly the decoding part which mirrors the encoding of input. Autoencoders use neural networks for this process. The method works by minimizing the reconstruction error between the original input and the output after encoding and decoding. This can be done using mean square error, or binary cross-entropy. It is important that the latent variables are forced to encorporate some information loss as otherwise the model would simply be multiplying the data by one [Anwar, A. 2021].

The limitation of this method is that the sole focus is on minimizing the error of the reconstruction, without incorporating probabilistic modeling, resulting in a method in which the latent space does not have generative capacity.

1.2 Variational Autoencoders

Variational autoencoders function similarly to normal autoencoders in that they encode input into a latent variable space with a lower dimension than the original input, as well as decode the encoded data in order to reconstruct a representation of the original input. However, it is achieved in a vastly different way. VAEs are probabilistic and can be used to generate new meaningful outputs based on random latent variables. A simplified explanation of the process of VAE's would be that they extend normal autoencoders to learn latent variables which follow a prior distribution by using a probabilistic framework [Kigma, D, P and Welling, M. 2013]. The steps of variational autoencoders are as follows:

1.2.1 Objective

The objective is to compress input data x into a latent variable space z using a probabilistic model where z is assumed to be sampled from a known probability distribution function which will be called $p(z)$. x is generated by some conditional probability $p_\theta(x \mid z)$. The problem that arises is the intractability of $p_\theta(x) = \int p_\theta(x \mid z)p(z)dz$.

1.2.2 Variational Approximation

Variational inference is used order to combat the intractability of $p_\theta(z \mid x)$. In variational approximation $p_\theta(z \mid x)$ is substituted by the variational posterior $q_\phi(z \mid x)$. The marginal likelihood $logp_\theta(x)$ can be expressed by: $\log p_\theta(\mathbf{x}^{(i)}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$
Where $\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$, the lower bound can be expressed as follows: $\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p_\theta(\mathbf{z}))+$

$E_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right]$ The KL divergence term ensures that the posterior $q_\phi(z \mid x)$ is kept close to the $p_\theta(z)$ which often is a Gaussian distribution of $\mathcal{N}(0,1)$. The term $E_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right]$ defines how well the input gets reconstructed. When solving the KL divergence term, which is to be maximized, we run into the problem of intractability:

$$-D_{KL}(q_\phi(z)\|p_\theta(z)) = \int q_\theta(z)\left(\log p_\theta(z) - \log q_\theta(z)\right) dz$$

1.2.3 Reparemeterization trick

In order to combat the intractability, which means that the computations are impossible or incredibly difficult to solve by known algorithms, VAEs use the parametrization trick. The trick works by replacing $q_\phi(z \mid x)$ with an equivalent distribution that is not parametrized by ϕ. To accomplish this, we assume that $q_\phi(z \mid x)$ is a Gaussian distribution, $z \sim p(z \mid x) = \mathcal{N}(\mu, \sigma^2)$. We define an auxiliary noise variable $\epsilon \sim \mathcal{N}(0,1)$. For this univariate Gaussian case, z can be reparameterized to $z = \mu + \sigma\epsilon$. By doing this we can express

$$\int q_\theta(z)\log p(z)\,dz = \int \mathcal{N}(z;\mu,\sigma^2)\log\mathcal{N}(z;0,\mathbf{I})\,dz = -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}\left(\mu_j^2 + \sigma_j^2\right)$$

and

$$\int q_\theta(z)\log q_\theta(z)\,dz = \int \mathcal{N}(z;\mu,\sigma^2)\log\mathcal{N}(z;\mu,\sigma^2)\,dz = -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log\sigma_j^2\right)$$

in order to obtain

$$-D_{KL}(q_\phi(z)\|p_\theta(z)) = \int q_\theta(z)\left(\log p_\theta(z) - \log q_\theta(z)\right) dz = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log\left(\sigma_j^2\right) - \mu_j^2 - \sigma_j^2\right)$$

Now finally we have our resulting estimator:

$$\mathcal{L}(\theta,\phi;\mathbf{x}^{(i)}) \simeq \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i,l)})$$

$$\mathbf{z}^{(i,l)} = \mu^{(i)} + \sigma^{(i)} \odot \boldsymbol{\epsilon}^{(l)}$$

$$\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0,\mathbf{I})$$

1.3 Extensions of VAE

One development since the publication of [Kigma, D, P and Welling, M. 2013] is disentangled VAEs. This autoencoder aims to improve VAEs ability to generalize by achieving greater feature independence in the latent variable space. Disentangled VAEs can achieve this by implementing a term β to the KL-divergence term. This term adjusts the weight of the KL-divergence term, so by adjusting it the model can be tuned in order to find a latent space that has a balance between feature independence, from higher β, and lower reconstruction error by having a lower β [Higgins et al., 2016].

$$\mathcal{L}(\theta, \phi; \mathbf{x}) = E_{q_\phi(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big] - \beta D_{KL}\big(q_\phi(\mathbf{z}|\mathbf{x})\|p(\mathbf{z})\big)$$

1.4 Significance of Latent Dimensionality

The choice of latent dimensionality in a Variational Autoencoder (VAE) is critical for balancing the trade-off between reconstruction accuracy and generalization capacity. Smaller latent dimensions constrain the model's capacity to represent data, potentially leading to underfitting, while larger dimensions may overfit the data and fail to generalize. This study investigates the impact of different latent dimensionalities (2, 10, 20, and 50) on the Evidence Lower Bound (ELBO) and the model's ability to reconstruct and generate meaningful samples.

2 Methods

2.1 Model Architecture

Firstly, an AE was implemented, before being extended to a VAE.

Vanilla Autoencoder The AE was designed as a baseline model. The implementation's architecture was, for the encoder, a fully connected feedforward network with one hidden layer. Input data was flattened to a vector of size 784 and passed through a hidden layer of 512 neurons to produce a deterministic latent vector. The decoder had the same hidden layer configuration, ending with a sigmoid activation function to match the pixel intensity of the data.

Variational Autoencoder The VAE extends the AE by introducing a probabilistic component to the latent space, making reconstruction and generation of data possible. The encoder outputs two parameters, μ and $\log \sigma^2$, representing a Gaussian distribution. A latent variable is then sampled from this using the reparametrization trick, enabling backpropagation through the sampling process. The VAE then optimizes a composite loss function that combines the reconstruction loss and the KL divergence term. This approach lets the VAE generate more structured and realistic output as sampling is now done directly from the latent space.

2.2 Training with Varying Latent Dimensions

To evaluate the effect of latent dimensionality, the VAE was trained with latent dimensions of 2, 10, 20, and 50. Each configuration was trained for 50 epochs using the Adam optimizer with a learning rate of 3×10^{-4}. The original paper on VAE mentions using Adagrad as the optimizer, but the Adam optimizer was released by the same author a year later. ELBO values were logged for each epoch to analyze convergence. The model's architecture and training procedure remained consistent across all experiments.

2.3 Data

The MNIST dataset was used to train and evaluate the models. It consists of 28x28 grayscale images of handwritten digits ranging from 0 to 9. Each image was normalized to have pixel values between 0 and 1, for consistency during training. The dataset was divided into training and testing subsets to assess reconstruction accuracy and generative capabilities of the models.

3 Results

3.1 ELBO Development Across Latent Dimensions

Figure 6 shows the ELBO progression over 50 epochs for different latent dimensions. Smaller dimensions (e.g., 2) exhibited slower convergence and significant fluctuations, reflecting the model's limited capacity to represent the data. Conversely, larger dimensions (e.g., 10, 20, 50) demonstrated faster convergence and stable ELBO values, with diminishing returns beyond 20 dimensions.

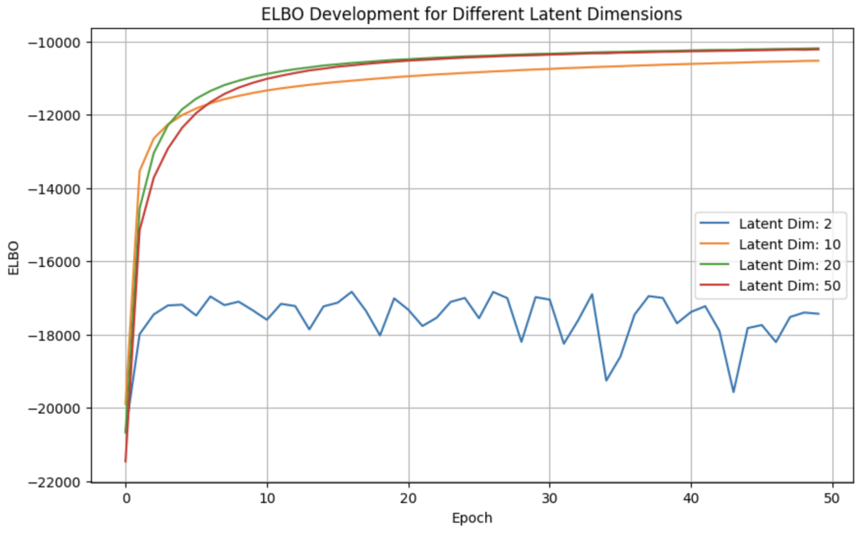


Figure 1: ELBO Development for Different Latent Dimensions

3.2 Final ELBO and Average ELBO

Table 1 summarizes the final and average ELBO values for each latent dimension. The results indicate that while increasing the latent dimension improves ELBO, the improvement plateaus between dimensions 20 and 50.

Latent Dimension	Final ELBO	Average ELBO
2	-17433.81	-17566.50
10	-10528.39	-11216.32
20	-10190.44	-10888.09
50	-10218.01	-11010.50

Table 1: Final and Average ELBO for Different Latent Dimensions

3.3 Random latent variable generation

The following are the outputs of both original samples, as well as a randomly generated latant variable, and combined.

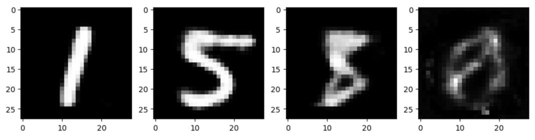


Figure 2: Vanilla AE.

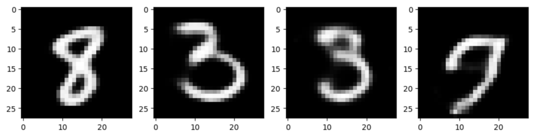
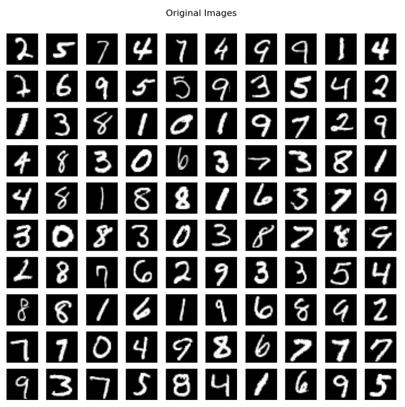


Figure 3: VAE.

The first two images in each figure are reconstructions of two different samples, image three is a reconstructed image based on a combination of the two latent variables from image 1 and two, and image 4 is a reconstruction based on completely random latent variables.



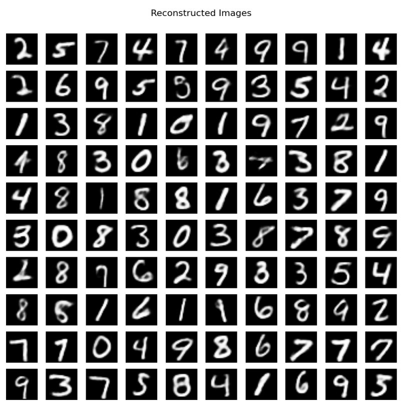


Figure 4: Original 16 Dimensions

Figure 5: Reconstructed 16 Dimensions

Figure 4 and Figure 5 shows our VAE's reconstruction of randomly sampled images from the dataset. The latent space for these figures were set to 16. Similar figures for 4 and 64 Dimensions can be found in the appendix.

4 Conclusion

4.1 Completely random latent variable space

The results of the latent variable generation tests in figure 2 and 3 align with the theory of autoencoders and variational autoencoders. Firstly, the image generated from randomized latent values is devoid of any meaningful features. This is to be expected as the encoder and decoder have been trained minimize the reconstruction error between original, i.e. meaningful images, reconstructed images. The model therefore only reconstructs meaningful images from latent spaces that are very similar to the latent spaces of the training data.

However, the VAE image decoded from random latent variables manages to produce a meaningful output, the digit nine for this specific run. This also aligns with the theory of variational autoencoders as the method adapts probabilistic learning, in order to generalize better.

4.2 Merged latent variable space

These results illustrate how reconstructed images are affected by taking the average of two latent variable spaces that were encoded from original samples. These results align well with the theory of vanilla autoencoders and variational autoendcoders. The merged image of two vanilla AE latent spaces resemble a graphical merge between a 0 and a 6.

In contrast to this the merged image of two VAE latent spaces more closely resembles one of the two samples that were merged.

5 Discussion

5.1 Impact of Latent Dimensionality

The results demonstrate a clear relationship between latent dimensionality and the model's performance. Smaller dimensions limited the model's capacity, leading to suboptimal ELBO values and unstable training dynamics. Larger dimensions improved ELBO but exhibited diminishing returns beyond 20 dimensions, suggesting that additional capacity did not significantly enhance the model's ability to represent the data.

5.2 Balancing Generalization and Reconstruction

The findings align with the theoretical trade-off in VAEs: smaller dimensions enforce stronger regularization, aiding generalization but limiting reconstruction fidelity. Larger dimensions allow for better reconstruction at the risk of overfitting, as evidenced by the slight increase in average ELBO for 50 dimensions.

5.3 Future Directions

Further experiments could explore intermediate latent dimensions (e.g., 15, 30) or investigate the role of regularization techniques like β-VAEs to mitigate overfitting in larger dimensions.

References

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *arXiv preprint* **arXiv:1312.6114**. Available at `https://arxiv.org/abs/1312.6114`.

Higgins, I., Matthey, L., Glorot, X., Pal, A., Uria, B., Blundell, C., Mohamed, S., and Lerchner, A. (2016). Early visual concept learning with unsupervised deep learning. *arXiv preprint* **arXiv:1606.05579**. Available at `https://arxiv.org/abs/1606.05579`.

Anwar, A. (2021). Difference Between Autoencoder (AE) and Variational Autoencoder (VAE). Available at `https://towardsdatascience.com/difference-between-autoencoder-ae-and-variational-autoencoder-vae-ed7be1c038f2`.

Appendix

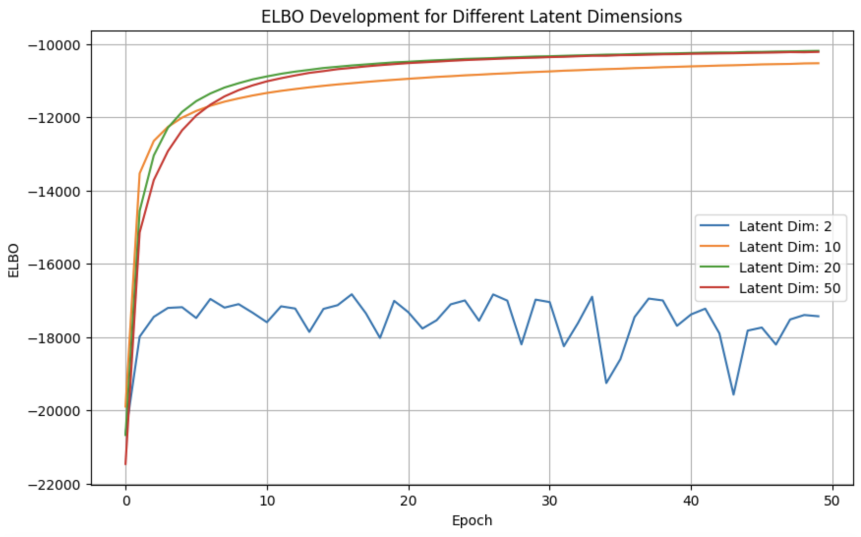


Figure 6: ELBO Development for Different Latent Dimensions

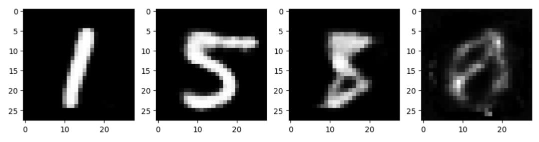


Figure 7: Vanilla AE.

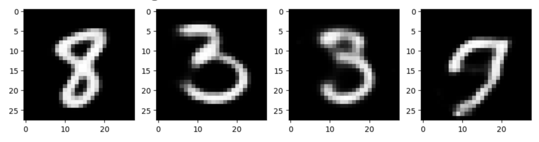


Figure 8: VAE.

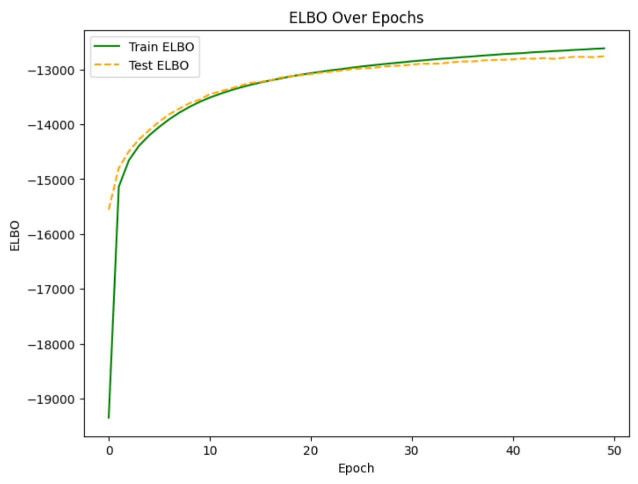


Figure 9: 4 Dimensions ELBO

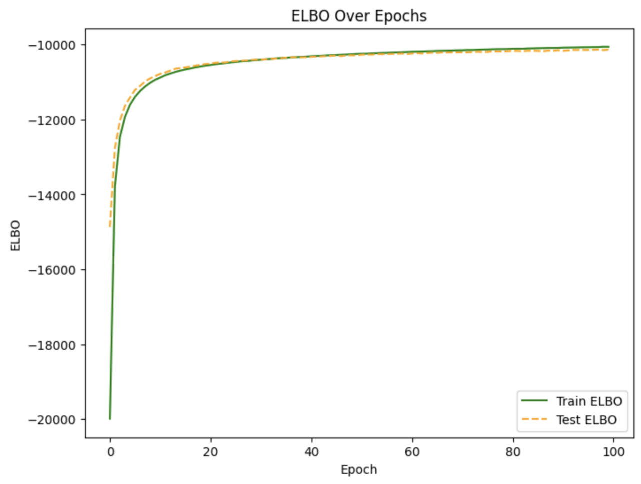


Figure 10: 16 Dimensions ELBO

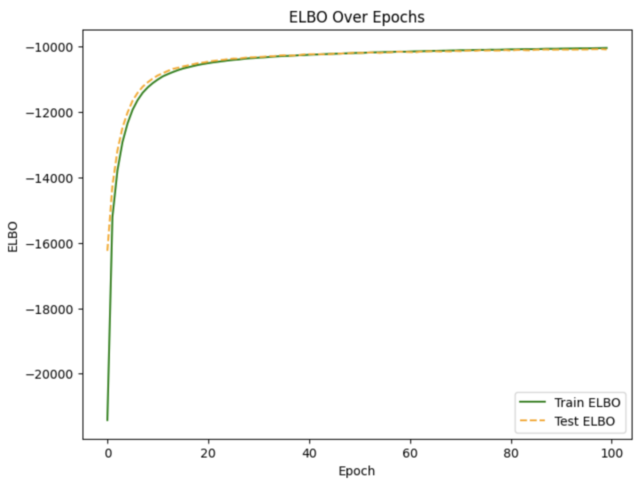


Figure 11: 64 Dimensions ELBO

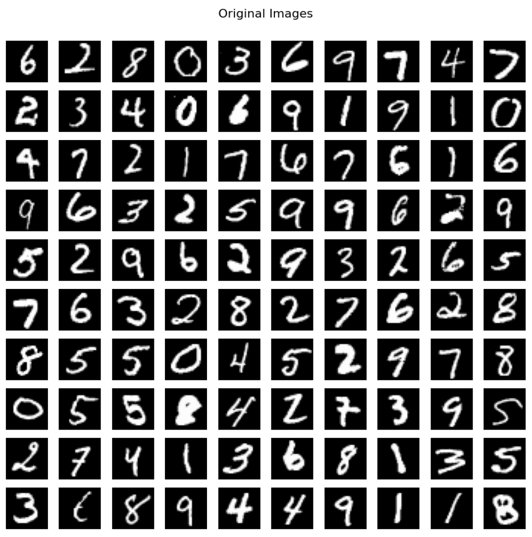


Figure 12: Original 4 Dimensions

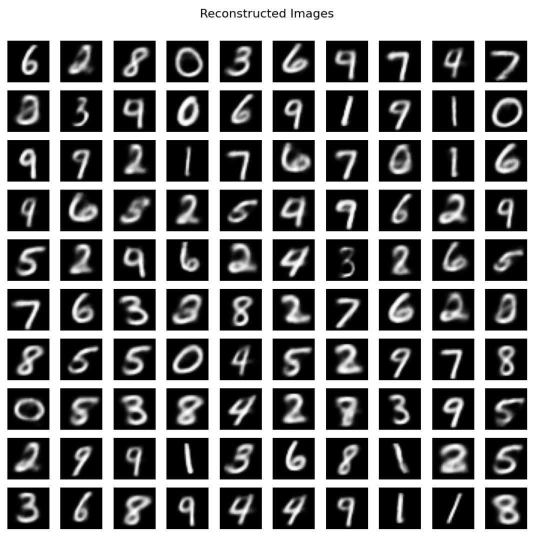


Figure 13: Reconstructed 4 Dimensions

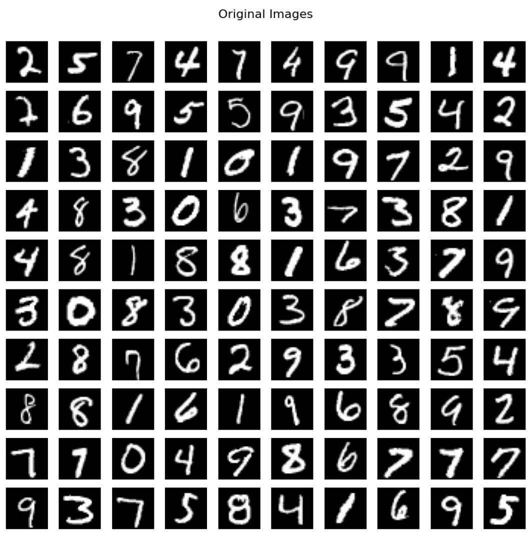


Figure 14: Original 16 Dimensions

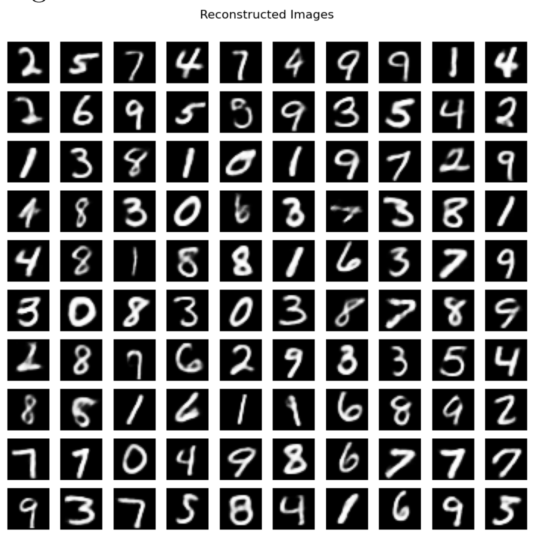


Figure 15: Reconstructed 16 Dimensions

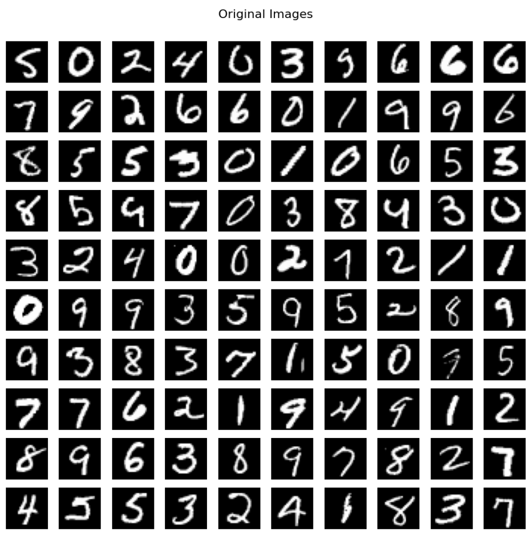


Figure 16: Original 64 Dimensions

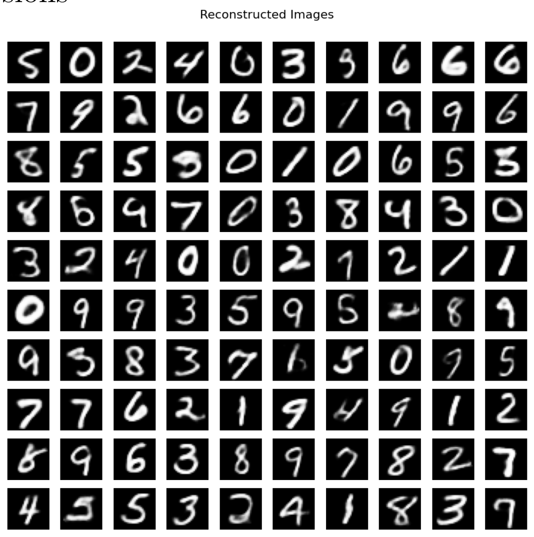


Figure 17: Reconstructed 64 Dimensions

Contributions

The project was completed collaboratively, with each group member contributing in various ways as decided through group communication: This included group discussions, problem solving and report writing.

- **Ellen:** Worked with Fredrik in implementing the AE and VAE architectures, making sure the models were correctly trained and optimized.
- **Fredrik:** Worked with Ellen in implementing the AE and VAE architectures, making sure the models were correctly trained and optimized. Also worked on data handling.
- **Oscar:** Handled processing of the MNIST data and worked on visualizations of the latent space and reconstruction results.
- **Anton:** Worked with Oscar on the data and contributed to testing and visualization.