Understanding Autoencoders and Variational Autoencoders

Regis Konan Marcel Djaha
African Masters of Machine Intelligence
African Institute for Mathematical Sciences
Basque Center for Applied Mathematics
Bilbao, Spain
rkmdjaha@aimsammi.org

Iñigo Urteaga
BCAM, Basque Center for Applied Mathematics
IKERBASQUE, Basque Foundation for Science
Bilbao, Spain
iurteaga@bcamath.org

Abstract

This work provides a comprehensive exploration of autoencoders (AEs) and variational autoencoders (VAEs), including beta-VAEs, which are fundamental unsupervised learning methods in artificial intelligence and machine learning. By delving into their theoretical underpinnings, practical implementations, and experimental analyses on benchmark datasets including MNIST, Fashion-MNIST, and FreyFace, the study enhances understanding of these models' capabilities in data representation, dimensionality reduction, image generation, and latent space exploration. Insights into network architectures, the influence of hyperparameters, and the significance of priors contribute to advancing the effective utilization of AEs and VAEs across diverse application domains.

1 Introduction

In recent years, the rapid evolution of artificial intelligence and machine learning has given rise to a multitude of sophisticated algorithms and techniques for modeling and understanding data. Autoencoders(AEs) and their more elaborate variants, Variational autoencoders(VAEs) are two prominent examples of advanced techniques in the field of unsupervised learning [18, 14]. These models are used to understand data and creating automated systems capable of generating, detecting anomalies, reconstructing, reducing dimensionality, and interpreting information effectively.

Despite their increasing importance, the theory underlying these algorithms can often seem complex and difficult to grasp. This work aims to deepen both the theoretical and practical understanding of these techniques, making these tools more accessible and enabling their effective use in various application domains such as image and video processing. In the first part, we present the theoretical underpinnings of AEs and VAEs, highlighting their ability to learn efficient representations of data by reducing dimensionality, while preserving essential information for accurate reconstruction. We explore the main differences between the two approaches, in particular the latent distribution constraint imposed by VAEs, and the variational inference algorithm used for their training. The second part focuses on the practical application of AEs and VAEs, with experiments in three benchmark datasets (MNIST, Fashion-MNIST, and FreyFace). Finally, in the third part, we discuss future developments and challenges in this domain.

Our main contributions are:

1. Exploring the Mathematical Foundations of AEs and VAEs: we explain how these models learn efficient data representations, and highlight the differences between them, particularly the latent distribution constraint of VAEs.
2. Practical Applications of AEs and VAEs:
   (a) We study the impact of network architecture in AEs and VAEs

Submitted to the African Institute for Mathematical Sciences for a Master's degree in Machine Intelligence (AIMS-AMMI 2023), Senegal.

(b) We study the impact of VAE priors
(c) We study the impact of β on VAE loss and performance

2 Methods

Autoencoders (AEs) and Variational autoencoders (VAEs) are unsupervised machine learning models. In this study, we examine in detail the theoretical principles underlying them, as well as their architectures and the training and evaluation methods used to measure their performance.

2.1 Autoencoders (AEs)

An autoencoder is a type of deep neural network that learns to compress data from the input layer into a compressed representation (the code), and then decompresses that code into an output that closely matches the raw input data. PCA (Principal Component Analysis), a dimensionality reduction technique, relies on linear transformations to identify the principal components [12]. In contrast, AEs can capture non-linear relationships, making them more adept at handling complex data [12]. This ability to model non-linear structures allows AEs to surpass the limitations of linear techniques such as PCA. AEs consist of encoding and decoding layers, which we now describe in detail.

2.1.1 Architecture

Encoder: In a (deep) autoencoder, a deep neural network is used to transform the input vector, denoted as $\mathbf{x}_{in} \in \{0,1\}^{D_x}$, into a lower-dimensional latent variable $\mathbf{z} \in \mathbb{R}^{D_z}$.
Decoder: Subsequently, another deep network is utilized to decode the latent variable z and reconstruct the vector $\mathbf{x}_{out} \in (0,1)^{D_x}$.
The encoder and decoder transformations are achieved through a neural network-based multi-layered, deterministic mapping represented as:

Encoder	Decoder
$x_1 = s_f(W_1 x_{in} + b_1)$	$x'_1 = s_g(W'_1 \mathbf{z} + b'_1)$
$x_2 = s_f(W_2 x_1 + b_2)$	$x'_2 = s_g(W'_2 x'_1 + b'_2)$
$\vdots$	$\vdots$
$x_{n_{layers}} = \mathbf{z} = W_{n_{layers}} x_{n_{layers}-1} + b_{n_{layers}}$	$x'_{n_{layers}} = \mathbf{x}_{out} = s_g(W'_{n_{layers}} x'_{n_{layers}-1} + b'_{n_{layers}})$

The choice of the number of layers, the dimensionality and the activation function used in each layer is very important for the performance of AEs. These architectural choices significantly influence the ability of the AE to learn useful representations and reconstruct the input.

2.1.2 Mathematical Formula to determine the Number of features per-layer

Equation (1) dynamically tunes the number of D_{x_i} features at each layer of the neural network. It is essential in our implementation, as it allows adaptive adjustments to the network architecture. Specifically,

$$D_{x_i} = D_x - \left[\frac{(D_x - D_z)\cdot i}{n_{layers}}\right] \quad \text{for} \quad i = 0, 1, \cdots n_{layers}\ , \tag{1}$$

where D_{x_i} represents the number of features at layer i; D_x is the initial number of features; D_z is the dimension of latent space or variable; i is the layer index, ranging from 0 to n_{layers}; and n_{layers} is the total number of layers.

2.1.3 Activation Function for each layer

In neural network-based autoencoders, the choice of which activation function to use is of significant importance. For the encoder, because the latent space is considered to be $\mathbb{R}^{D_\mathbf{z}}$, we do not use an activation function. However, for the decoder, because we often need to constrain the space of the output x_{out}, we need to carefully pick the correct activation function. The most commonly used functions for this purpose are the ReLU and sigmoid functions depending on the type of dataset.

Sigmoid: The sigmoid function, denoted as σ, can take all values in the range $[0, 1]$. This characteristic is a consequence of its mathematical formulation: $\sigma(x) = \frac{1}{1+\exp(-x)}$. The sigmoid activation function is ideal for the output layer when input observations range exclusively between 0 and 1, or when they have been normalized to fit within the interval [0, 1].

ReLU: The ReLU activation function can assume all values in the range $[0, \infty)$. Its formula is ReLU(x) $= \max(0, x)$. This is an excellent choice for the output layer when the input observations x cover a wide range of positive real values.

Identity function: If the input $\mathbf{x}_{in}$ can encompass negative and positive real values, i.e., $x_{in} \in \mathbb{R}$, opting for the Rectified Linear Unit (ReLU) becomes impractical, and the identity function emerges as a significantly more suitable alternative: $Identity(x) = x$.

2.1.4 Training an autoencoder: the loss function and backpropagation

The primary objective of an AE is to minimize the reconstruction error, i.e., an AE solely focuses on this error and does not consider any external factors or labels during training. For measuring the reconstruction loss, we can use the Binary cross-entropy (BCE) or Mean Squared Error (MSE). Further detailed descriptions of these loss functions are provided in Appendix A.1. Training an AE using backpropagation involves encoding the input data $\mathbf{x}_{in}$ into a latent space $\mathbf{z}$, decoding $\mathbf{z}$ to reconstruct the data as $\mathbf{x}_{out}$, and optimizing the reconstruction loss using stochastic gradient descent to minimize the difference between $\mathbf{x}_{in}$ and $\mathbf{x}_{out}$. In Appendix A.1, we show an example of training an AE using backpropagation with BCE loss on different datasets.

2.1.5 Latent space visualization

The latent space (see Figure 1) known as the code, the bottleneck layer, or the representation layer, is the compressed representation obtained from the encoder. It is a lower-dimensional space that captures the essential features of the input data. The bottleneck layer acts as a bottleneck or constraint that forces the model to learn a more compact and efficient representation.

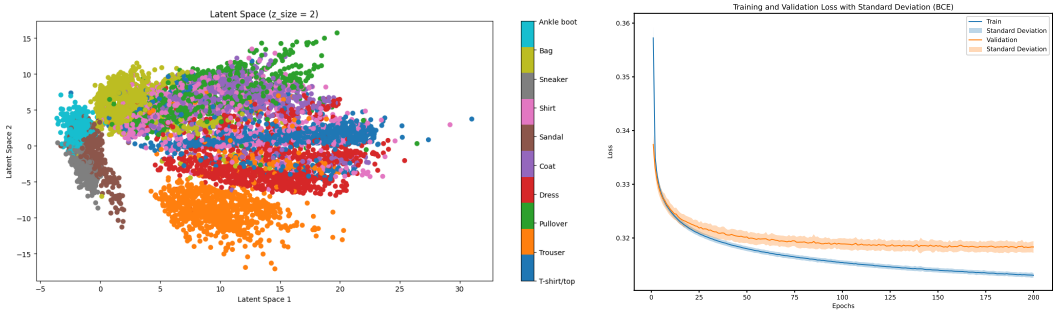

Figure 3: Latent space visualization (Figure 1) and Evolution of the BCE loss (Figure 2) of an AE using the FashionMNIST dataset with a 2-dimensional latent space and two additional layers.

In Figure 1, we observe that certain regions of an AE's latent space show a clear separation between the different classes. For example, the "Trousers" class (orange) forms a distinct cluster, indicating that the AE has effectively learned to differentiate trousers from other items. Similarly, the "Ankle boot" class (cyan) and the "Bag" class (olive) also show good separation from the rest of the classes. However, there are regions where classes overlap significantly, such as the central region where several classes (e.g., "Shirt," "T-shirt/top," "Pullover") are mixed. This overlap suggests that the AE has difficulty differentiating between these similar categories, which may be due to their similarities.

Training the neural networks conforming the AE can be challenging. There is a risk that the network may not generalize effectively, potentially leading to overfitting, as shown in Figure 2, where training loss goes down while validation loss increases with more epochs. To mitigate this issue, it's important to use regularization techniques to improve the quality of the latent space. Regularization methods, such as those imposed within VAEs, can help prevent overfitting by imposing constraints on the latent space, ensuring better generalization and clearer class separation.

2.2 Variational Autoencoders (VAEs)

VAEs are an extension of AEs that allow for more complex data generation and representation. Introduced by Kingma and Welling [14], VAEs differ from AEs in that they model the latent space with a probabilistic distribution. In an AE, each input data point is deterministically mapped to a single fixed point in the latent space, whereas in a VAE, each input data point is represented by a full distribution in the latent space. This approach enables VAEs to generate new data samples by sampling from the learned distribution, making them particularly powerful for data generation.

2.2.1 VAE model and architecture

The VAE latent space. The most common choice is to model the VAE's latent variable corresponding to each input observation with a Gaussian distribution $\mathbf{z}(\mathbf{x}_{in}) \sim \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}(\mathbf{x}_{in}), \boldsymbol{\Sigma}(\mathbf{x}_{in}))$,

where $\boldsymbol{\mu}$ is the mean vector, and $\boldsymbol{\Sigma}$ is the covariance matrix. The mean $\boldsymbol{\mu}$ is determined by a function $f_{\boldsymbol{\mu}}(\mathbf{x}_{in})$ that maps the input $(\mathbf{x}_{in})$ to $\mathbb{R}^{D_z}$, representing the mean of the Gaussian distribution. The covariance matrix Σ is typically chosen as $\sigma^2(\mathbf{x}_{in})I$, where I is the identity matrix and $\sigma^2(\mathbf{x}_{in})$ is computed by the function $f_\sigma(\mathbf{x}_{in})$ in the range $(0, \infty)$. This reflects the variability of the latent variable $\mathbf{z}$ for each observation. Note that, $\boldsymbol{\mu}$ and σ^2 are deterministic conditioned on input data.

Encoder. Consider deep neural networks designed to transform the input vector, denoted as $\mathbf{x}_{in}$, into lower-dimensional $\boldsymbol{\mu}(\mathbf{x}_{in}) \in \mathbb{R}^{D_z}$ and $\sigma^2(\mathbf{x}_{in}) \in (0, \infty)$, each with equations:

$$\begin{aligned} x_1 &= f_\mu(x_{in}) = s_f(W_1 x_{in} + b_1), & x'_1 &= f_\sigma(x_{in}) = s_f(W'_1 x_{in} + b'_1), \\ x_i &= f_\mu(x_{i-1}) = s_f(W_i x_{i-1} + b_i), & x'_i &= f_\sigma(x_{i-1}) = s_f(W'_i x'_{i-1} + b'_i), \\ &\cdots & &\cdots \\ x_{n_{layers}} &= \boldsymbol{\mu} = f_\mu(x_{n_{layers}-1}) = W_{n_{layers}} x_{n_{layers}-1} + b_{n_{layers}}, & x'_{n_{layers}} &= \sigma^2 = f_\sigma(x'_{n_{layers}-1}) = W'_{n_{layers}} x'_{n_{layers}-1} + b'_{n_{layers}} \end{aligned}$$

Sampling the latent variable. Parameters $(\boldsymbol{\mu}, \sigma)$ of the conditional distribution are acquired when the data point $\mathbf{x}_{in}$ is input into the encoder. Specifically, the encoder network outputs $\boldsymbol{\mu}(\mathbf{x}_{in})$ and $\sigma(\mathbf{x}_{in})$, which parameterize a Gaussian distribution $q_\phi(\mathbf{z} \mid \mathbf{x}_{\text{in}}) = \mathcal{N}(\mathbf{z}; \mu_\phi(\mathbf{x}_{\text{in}}), \text{diag}(\sigma^2_\phi(\mathbf{x}_{\text{in}})))$. As a result, the latent space distribution is determined as a function of the data point $\mathbf{x}_{in}$. We can sample the corresponding latent variable $\mathbf{z}$ from this latent space distribution, i.e., $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x}_{in})$. This sampled latent variable $\mathbf{z}$ captures the essential features of the input data $\mathbf{x}_{in}$ in a lower-dimensional space. The decoder receives this latent variable $\mathbf{z}$ as input, as described below.

Decoder. Subsequently, another deep network is employed to decode *sampled* $\mathbf{z}$ values to reconstruct the output vector $\mathbf{x}_{out} \in (0, 1)^{D^{\mathbf{x}}}$, via a multi-layered and deterministic transformation:

$$\begin{aligned} \tilde{x}_1 &= s_g(\tilde{W}_1 \mathbf{z} + \tilde{b}_1), \\ \tilde{x}_2 &= s_g(\tilde{W}_2 x'_1 + \tilde{b}_2) \\ &\vdots \\ \tilde{x}_{j=n_{layers}} &= x_{out} = s_g(\tilde{W}_j \tilde{x}_{j-1} + \tilde{b}_j) \end{aligned}$$

2.2.2 Training a VAE: the Evidence Lower Bound

The objective of a VAE is to minimize the reconstruction error. However, the approach differs from direct reconstruction loss minimization, because we aim to preserve the probabilistic nature of the model. Consequently, we instead seek to maximize the reconstruction likelihood $\log p_\theta(\mathbf{x})$ of observed data x. Directly computing $p_\theta(\mathbf{x})$ is intractable for VAEs due to the complexities involved. To address this, VAEs use the Evidence Lower Bound (ELBO), which is the result of minimizing the Kullback-Leibler (KL) divergence between an approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ [5, 21] :

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}|\mathbf{x})) = \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} \, d\mathbf{z} \tag{2}$$

The ELBO, derived from the properties of KL divergence and applying Bayes' theorem (further details can be found in appendix (B.1)), ensures that:

$$\log p_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z}|x)} \right] = \mathcal{L}(\phi, \theta, \mathbf{x}, \mathbf{z}) \tag{3}$$

Where $\mathcal{L}(\phi, \theta, \mathbf{x}, \mathbf{z})$ represents the Evidence Lower Bound. The ELBO can be written as:

$$\mathcal{L}(\phi, \theta, \mathbf{x}, \mathbf{z}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x}|\mathbf{z})] \tag{4}$$

where D_{KL} denotes the Kullback-Leibler divergence between the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and the prior $p_\theta(\mathbf{z})$, and $\mathbb{E}$ represents the expectation with respect to $q_\phi(\mathbf{z}|\mathbf{x})$. To compute the ELBO, we use Monte Carlo sampling to estimate the expectation term. To maximizize the ELBO via backpropagation, we use the reparameterization trick to be able to compute its gradients efficiently.

Monte Carlo estimation. The Monte Carlo approximation is used because in practice, it is intractable to compute the ELBO's expectation analytically due to the non-linearity in the encoder and decoder mappings. Instead, the expectation is approximated by drawing L samples $\mathbf{z}^{(l)} \sim q_\phi(\mathbf{z}|\mathbf{x})$:

$$\mathcal{L}(\phi, \theta, \mathbf{x}, \mathbf{z}) \approx \hat{\mathcal{L}}_B(\phi, \theta, \mathbf{x}, \mathbf{z}, L) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(\mathbf{x}|\mathbf{z}^{(l)}) \tag{5}$$

The KL Divergence loss measures how closely the approximate posterior $q_\phi(\mathbf{z}|\mathbf{x})$ matches the prior $p_\theta(\mathbf{z})$, acting as a regularizer to keep the latent space distributions well-behaved. Meanwhile, the reconstruction loss, minimized by averaging the log-likelihood over L samples, evaluates how the model reconstructs the original data x. When $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z}|\mathbf{x})$ follow multivariate Gaussian distributions, the KL divergence can be computed analytically, see Appendix B.2 for details.

Reparametrization trick. We have a latent variable $\mathbf{z}$ that is sampled from a distribution $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ parameterized by ϕ. The goal is to optimize the parameters ϕ and θ of the encoder (inference network) and decoder (generative network), respectively. However, sampling from $q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$ makes it difficult to compute the ELBO's gradients with respect to ϕ. VAEs use the reparametrization technique to get around this issue [14, 19]. This strategy consists on rewriting a random variable $\mathbf{z}$ as a deterministic function of another random variable $\boldsymbol{\varepsilon}$ and, rather than sampling from the original distribution $\mathbf{z}^{(i,l)} \sim q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$, instead drawing from:

$$\mathbf{z}^{(i,l)} = f_\phi(\boldsymbol{\varepsilon}^{(i,l)}, \mathbf{x}^{(i)}) \tag{6}$$

Where $\boldsymbol{\varepsilon}^{(i,l)}$ is a sample from a stochastic variable drawn from a known and easy to draw from distribution $\boldsymbol{\varepsilon}^{(l)} \sim p(\varepsilon)$. By applying the reparameterization technique, the encoder function $f_\phi(\boldsymbol{\varepsilon}^{(i,l)}, \mathbf{x}^{(i)})$ implements $q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)})$, where we have $\mathbf{z}^{(i,l)} = f_\phi(\boldsymbol{\varepsilon}^{(i,l)}, \mathbf{x}^{(i)})$ and $\boldsymbol{\varepsilon}^{(i,l)} \sim p(\boldsymbol{\varepsilon})$ in the latent space. For a Gaussian latent variable $\mathbf{z}$, the reparameterization can be computed as follows:

$$\mathbf{z}^{(i,l)} = f_\phi(\boldsymbol{\varepsilon}^{(i,l)}, \mathbf{x}^{(i)}) = \boldsymbol{\mu}_\phi^{(i)}(\mathbf{x}_i) + \sigma_\phi^{(i)}(\mathbf{x}_i) \odot \boldsymbol{\varepsilon}^{(l)}$$

where, $\boldsymbol{\varepsilon}^{(l)} \sim$N(0;I), $\boldsymbol{\mu}_\phi$ is the mean and σ_ϕ is the standard deviation produced by the encoder network. In practice, we use $\mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + exp(\frac{\log {\sigma^{(i)}}^2}{2}) \odot \boldsymbol{\varepsilon}^{(l)}$. We can rewrite an estimate for $\hat{\mathcal{L}}_B\big(\phi, \theta, \mathbf{x}^{(i)}, \mathbf{z}, L\big)$ as presented in Equation (5) in terms of the auxiliary samples $\boldsymbol{\varepsilon}^{(l)}$.

$$\hat{\mathcal{L}}_B\Big(\phi, \theta, \mathbf{x}^{(i)}, \boldsymbol{\varepsilon}^{(i,l)}, L\Big) = - D_{KL}(q_\phi(f_\phi(\boldsymbol{\varepsilon}^{(i,l)}, \mathbf{x}^{(i)}))||p_\theta(f_\phi(\boldsymbol{\varepsilon}^{(i,l)}, \mathbf{x}^{(i)}))) \tag{7}$$

$$+ \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}^{(i)}|f_\phi(\boldsymbol{\varepsilon}^{(i,l)}, \mathbf{x}^{(i)})) \tag{8}$$

where f_ϕ is the deterministic part of the encoder, and the ε values are drawn from the distribution $p(\boldsymbol{\varepsilon})$, i.e., $\boldsymbol{\varepsilon}^{(l)} \sim p(\boldsymbol{\varepsilon})$ [14]. This quantity is differentiable with respect to both ϕ and θ, making it suitable for updating parameters in a minibatch gradient ascent algorithm.

Training VAEs with Backpropagation

Backpropagation is the method used in practice to train VAEs [19], with the backpropagation algorithm [20] being utilised to train the network's weights. Training a VAE using backpropagation involves encoding input data $\mathbf{x}_{in}$ into a latent space as parameters $\boldsymbol{\mu}$ and $\log \sigma^2$, reparameterizing the latent variable $\mathbf{z}$ using $\mathbf{z} = \boldsymbol{\mu} + \sigma \odot \boldsymbol{\varepsilon}$, decoding $\mathbf{z}$ to reconstruct the input $\mathbf{x}_{out}$, and optimizing the ELBO: a loss function that combines the reconstruction loss and the KL divergence to regularize the latent space distribution. When training VAEs with backpropagation, the whole network (encoder and decoder) is trained together and not in separate steps. Let us assume that $\psi : (\phi, \theta)$ represents the whole set of VAE weights. Backpropagation trains VAE using the mini-batch stochastic gradient descent with learning rate ρ of the negative ELBO, as per the following steps:

$$\psi := \psi - \rho \nabla_\psi \sum \hat{\mathcal{L}}\Big(\psi, \mathbf{x}^{(i)}, \mathbf{z}, L\Big) \tag{9}$$

We show an evolution example of the ELBO-based training of VAEs with differents datasets in Appendix B.3

2.2.3 β-Variational Autoencoders

A modified version of a VAE can be used to control the trade-off between reconstruction error and KL divergence [11]. This can be achieved by introducing a hyperparameter β in the ELBO:

$$\mathcal{L}(\phi, \theta, \mathbf{x}, \mathbf{z}) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|x)}\{\log p_\theta(\mathbf{x}|z)\} - \beta\, D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \tag{10}$$

This parameter controls the impact on how much the approximate posterior resembles the prior. To be more precise, the behavior returns to that of an AE at $\beta = 0$, and the original ELBO of the VAE is

restored at $\beta = 1$. As β increases, the KL divergence becomes more important. Therefore, using an uncorrelated prior ensures that each dimension in the latent space remains more uncorrelated. From an information-theoretic perspective, the β term imposes constraints on the capacity of latent variable [2]. Consequently, excessively high β values lead to distributions of latent neurons that are identical to the a priori, making the constraint useless for reconstruction (no information gets past it). This viewpoint explains the observed phenomenon of lower-quality reconstructions in VAEs under increased regularization (see Table 2).

3 Experiments

Datasets. To train and evaluate the methods described in Section 2, we conducted experiments on three commonly used datasets in the literature. MNIST[1] and Fashion-MNIST[2] consist of grayscale images of size 28 × 28, representing handwritten digits and clothing items, respectively. Both datasets contain ten different classes. Freyface[3] is a dataset comprising 1965 images of Brendan's face, extracted from sequences of a small video, with each image having a size of 20x28 pixels. We normalize the examples from the MNIST, Fashion-MNIST, and Freyface datasets to a range between 0 and 1 before proceeding with our experiments. We divide the datasets into training and test sets. For the MNIST and Fashion-MNIST datasets, we keep the original split of 60,000 training examples and 10,000 test examples. For the Freyface dataset, we use 1,565 examples for training and another 400 examples for testing.

Model architecture. Our architecture is designed with encoders ranging from 1 to 8 layers, where the number of neurons per layer is predefined based on the specific formula in Equation (1). The corresponding decoder also consists of 1 to 8 layers, mirroring the encoder structure.

K-fold cross-validation (K=10). The AE, VAE, and β-VAE models were retrained ten times, with the dataset split into ten folds, and a different fold was utilized as validation data in each iteration. Consequently, for the MNIST and Fashion-MNIST datasets, we now have $54,000$ training images, $6,000$ validation images, and $10,000$ testing images. Similarly, for the Freyface dataset, we have $1,252$ training images, 313 validation images, and 400 testing images. The reported results represent the mean and standard deviation over all validation and testing folds.

Training procedure and hyperparameters. We used the Adam optimizer, and trained each model for 200 epochs on both the MNIST and Fashion-MNIST datasets with BCE for reconstruction loss, while for the Freyface dataset, we trained for 2000 epochs because it contains images of faces, which are more complex in terms of detail and variations (angles, etc.) with MSE for reconstruction loss. The batch size was set to 100, and the learning rate was maintained at 0.001. We trained all the models on a local machine using an Apple M1 Pro Chip with 10-core CPU and 16-core GPU.

Code. The source code for the implementation and the scripts-notebooks to replicate the following experiments are available in this Github link.

3.1 Comparative Studies between Autoencoders and Variational Autoencoders

The aim with these experiments is to deepen our understanding of their respective strengths and weaknesses, identifying the conditions under which each excels and how they can be optimised for different applications. We select the best fold from the cross-validation process to present the original image and its reconstructions here; see Appendix C.1.1 for results for all folds.

3.1.1 Reconstructed images

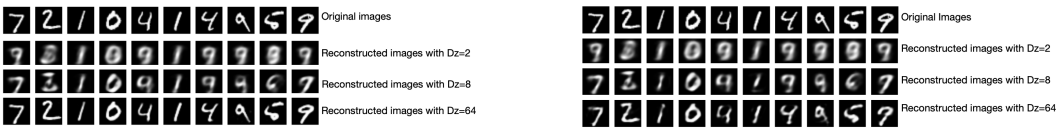

Figure 6: In the top line, we can see the original digits from the MNIST dataset. The line of digits below shows the digits reconstructed using 1-layer AEs (Figure 4) and VAEs (Figure 5).

Impact of Latent Space Dimensionality ($D_z = 2$, 8 and 64) on Single-Layer AE and VAE Performance: using the MNIST dataset, we explore how varying the dimension of the latent space affects the reconstruction performance of single-layer models. We show in Figure 6, a very small

[1]MNIST: `http://yann.lecun.com/exdb/mnist/`

[2]Fashion-MNIST: `https://keras.io/api/datasets/fashion_mnist/`

[3]Freyface: `https://cs.nyu.edu/~roweis/data.html`

dimension of the latent space ($Dz = 2$) degrades the quality of reconstruction for both AEs and VAEs. At $D_z = 8$, some reconstructed images remain blurry for AEs and VAEs. Increasing the dimension of the latent space to $D_z = 64$ improves the quality of reconstructed images for AEs, while for VAEs, it improves slightly. Table 5 and Table 8 in Appendix C.1.1 provide more results showing the impact of the latent space dimensionality on image reconstruction quality, for AEs and VAEs applied to the aforementioned datasets.

Impact of Number of Layers on Performance of AEs and VAEs with Two-Dimensional Latent Space: We investigate how varying the number of layers influences the reconstruction performance specifically configured with a two-dimensional latent space.

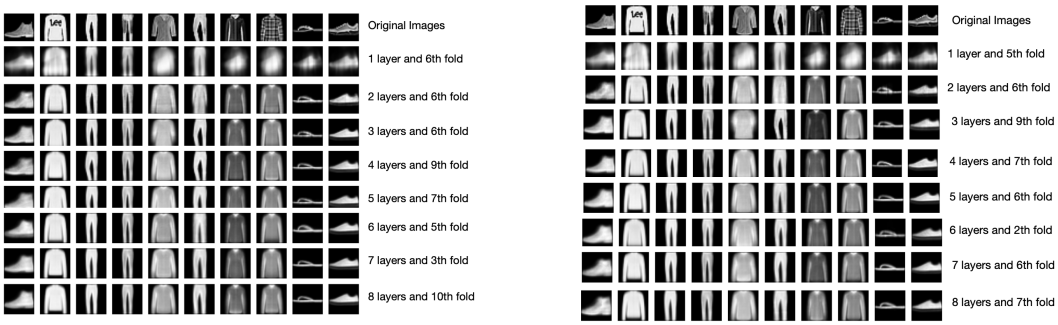

Figure 9: In the top line, we can see the original Fashions from the FashionMNIST dataset. The line of Fashions below contains the best images reconstructed by AEs (Figure 7) and VAEs (Figure 8).

We observe in Figure 9 that the performance of both models, AEs (Figure 7) and VAEs (Figure 8), improves significantly with a higher number of layers. However, despite these improvements, some images still appear blurry. To achieve higher-quality image reconstructions, it is necessary to increase the dimension of the latent space (see additional result in Append C.1). This approach improves the quality of the reconstructions and brings them closer to the original images.

3.1.2 Latent Space and Image Generation

We compare AEs and VAEs by examining their latent space and their performance in generating new images, particularly with a deep latent space (number of layers = 6), using the MNIST dataset. After we have trained each model, we only use the decoder to generate images by providing it with latent vectors to decode. We create an interpolation grid in the latent space, generating latent vectors by varying their values along a defined range. For example, for latent vectors of size two $D_z = 2$, ranging each of its elements from -3 to 3. We generate 100 images corresponding to these interpolated latent vectors. Visualizations in Figures 11 and Figure 15 show different grid-generated images, illustrating how the latent space interpolates between different points in the latent space. We also generate new and diverse images by randomly sampling from the latent space in Figure 12 and Figure 16, where we sample latent vectors from a standard normal distribution $\mathcal{N}(0; I)$. These random latent samples are fed into the decoder to produce the images shown.

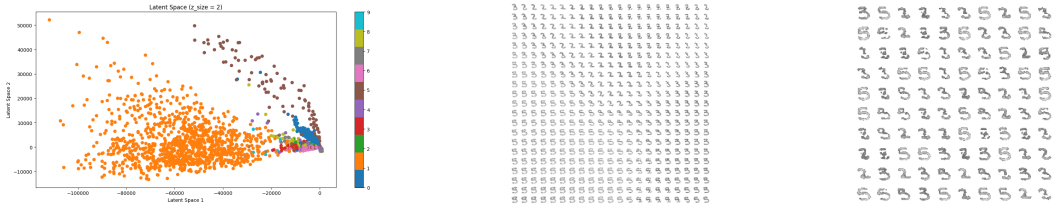

Figure 13: Visualization of AE Latent Space (Figure 10) and Image Generation, Including Both Interval-Specified [-3,3] (Figure11) and Randomly Generated (Figure12).

In the AE's latent space in Figure 10, some classes, such as the orange cluster, are less tightly clustered than others, with less overlap among classes. This indicates that the AE separates the different digit classes in a broad range of the latent space, with points not centered around $(0, 0)$. Furthermore, when utilizing a deep latent space with 6-layers, AEs struggle to randomly generate multiple distinct and high-quality images, see Figures 11 and 12.

The leftmost plot in Figure 14 shows a more structured and organized VAE latent space compared to that of the AE. In this case, most classes form compact and distinct clusters, with less overlap

compared to the AE latent space. The two histograms (Figure 14) on the right of the latent space dimensions compared with a standard normal distribution show that the VAE's latent space dimensions roughly follow a standard normal distribution. That is, VAEs with standard Gaussian priors ensure that the points in the latent space are centered around $(0, 0)$, which facilitates smooth and meaningful interpolation. This structured approach enables a more coherent and rich exploration of the latent space: VAEs effectively generate multiple distinct and high-quality images within the interval [-3, 3]. Overall, the VAE latent space is more suitable for generating well-defined and class-distinguishable latent representations compared to the AE latent space, regardless of the number of layers used.

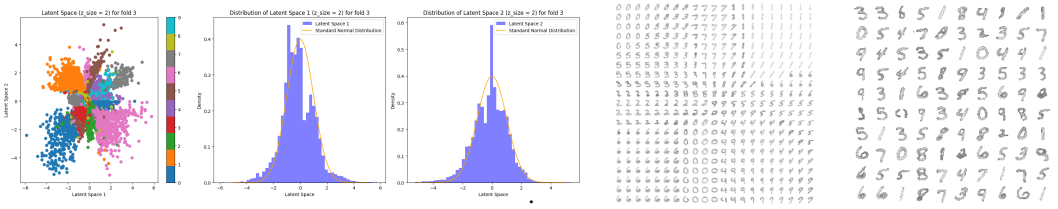

Figure 17: Visualization of VAEs latent space (Figure14) and Image Generation, Including Both Interval-Specified [-3,3] (Figure15) and and Randomly Generated (Figure16)

3.2 The impact of Variational Autoencoders priors

We here investigate how the VAE prior distribution affects the model's ability to generate new data.

3.2.1 Latent space visualization and Generation of Images

This study examines the effects of varying the mean and standard deviation of the latent space distribution on VAEs trained on the MNIST dataset.

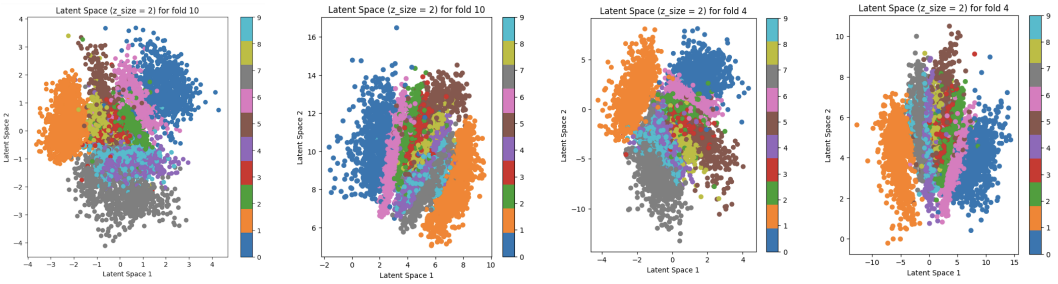

Figure 22: Visualization of the latent space with different VAE priors: $\mathcal{N}(0, \text{diag}(1, 1))$ in Figure 18, $\mathcal{N}((5, 10), \text{diag}(1, 1))$ in Figure 19, $\mathcal{N}(0, \text{diag}(1, 3))$ in Figure 20 and $\mathcal{N}((2, 5), \text{diag}(3, 1))$ in Figure 21

Figure 27: Image generation with different VAE priors: $\mathcal{N}(0, \text{diag}(1, 1))$ in Figure 23, $\mathcal{N}((5, 10), \text{diag}(1, 3))$ in Figure 24, $\mathcal{N}(0, \text{diag}(1, 3))$ in Figure 25 and $\mathcal{N}((2, 5), \text{diag}(3, 1))$ in Figure 26

Figure 22 shows that, whatever the a priori distribution used, the latent space presents structured and organized clusters. Changing the mean of the prior affects the location of the clusters in the latent space, as shown in Figure 19 and Figure21: with $\mathcal{N}((5, 10), \text{diag}(1, 1))$ and $\mathcal{N}((2, 5), \text{diag}(3, 1))$, the clusters become elongated. Furthermore, the prior distribution of $\mathcal{N}(0, \text{diag}(1, 1))$ shown in Figure 18 establishes a latent space where the clusters are relatively well-formed. When the mean of the prior is shifted, as seen in Figure 19, where the prior is $\mathcal{N}((5, 10), \text{diag}(1, 1))$, the latent space still maintains a structured appearance, but with clusters displaced according to the mean change.

Figure 21 demonstrates the impact of modifying both the mean and variance simultaneously. Here, the prior $\mathcal{N}((2,5), \text{diag}(3,1))$ leads to clusters that are not only displaced due to the mean shift but also stretched and elongated according to the variance in each dimension.

To generate new samples, we draw points from these latent spaces, based on the specified prior distribution and then decode these points using the VAE decoder: see Figures 23, 24, 25 and 26. We achieve high-quality image generation across the various prior distributions used. We find that VAEs are particularly effective when the prior distribution is $\mathcal{N}(0, \text{diag}(1,1))$. This classic configuration is advantageous because it is centered at $(0,0)$ and is easy to sample from, which simplifies image generation compared to other prior distributions.

3.2.2 Comparison of Reconstruction Performance

Table 2 below presents the BCE loss comparison between different VAE priors trained on the MNIST dataset: results highlight the impact of various priors on the VAE's reconstruction performance.

Model D_x: 784, n_{layers}: 2, D_z: 2	**Validation**	**Testing**
VAE Prior $\mathcal{N}(0, \text{diag}(1,1))$	$0.184 \pm 0.1983\%$	$0.1837 \pm 0.1662\%$
VAE Prior $\mathcal{N}((5,10), \text{diag}(1,1))$	$0.1837 \pm 0.1249\%$	$0.1830 \pm 0.1123\%$
VAE Prior $\mathcal{N}(0, \text{diag}(1;3))$	$0.1833 \pm 0.1728\%$	$0.1825 \pm 0.1096\%$
VAE Prior $\mathcal{N}((2,5), \text{diag}(3;1))$	$0.1852 \pm 0.3219\%$	$0.1846 \pm 0.3196\%$

Table 1: Comparison of BCE loss between VAE Prior and VAE on the MNIST Dataset.

The results indicate that the VAEs priors $\mathcal{N}((5,10), \text{diag}(1,1))$ and the $\mathcal{N}(0, \text{diag}(1,3))$ both offer stable and efficient performance with lower BCE scores. These priors provide a good reconstruction image. In contrast, more complex priors with higher variance, such as $\mathcal{N}((2,5), \text{diag}(3,1))$, reflect poor image reconstruction.

3.3 The impact of β

We conduct an experiment to study the impact of different values of β (0.02, 1, 1.5, and 10) on a VAE trained on the MNIST dataset.

The choice of β directly affects both the latent space regularization, as shown in Figures 28, 29, 30 & 31; and the characteristics of the generated images, described in Table 2. Lower β values focus more on reconstruction accuracy, while higher β values prioritize the structure and variability of the latent space, impacting the diversity and realism of the generated samples.

3.3.1 Comparison of Reconstruction Performance

The following table summarizes the impact of different β values on the validation and testing loss of the β-VAE:

Model D_x: 784, n_{layers}: 2, D_z: 2	**β-VAE Validation**	**β-VAE Testing**
$\beta = 0.02$	$\mathbf{0.183 \pm 0.204\%}$	$\mathbf{0.183 \pm 0.163\%}$
$\beta = 1$	$0.184 \pm 0.198\%$	$0.184 \pm 0.166\%$
$\beta = 1.5$	$0.184 \pm 0.170\%$	$0.184 \pm 0.151\%$
$\beta = 10$	$0.198 \pm 0.163\%$	$0.198 \pm 0.154\%$

Table 2: β-VAE with BCE loss table.

1. $\beta = 0.02$: With a very small β, the model behaves almost like a autoencoder, prioritizing reconstruction accuracy over regularization. This is reflected in the lowest validation and testing losses ($0.183 \pm 0.204\%$ and $0.183 \pm 0.163\%$, respectively), indicating lower reconstruction error.
2. $\beta = 1$: At $\beta = 1$, the VAE achieves a balance between reconstruction and regularization. The reconstruction error slightly increases to 0.184% for both validation and testing sets.
3. $\beta = 1.5$: Increasing β to 1.5 shows a small improvement in both validation and testing losses compared to $\beta = 1$, with $0.184 \pm 0.170\%$ and $0.184 \pm 0.151\%$, respectively. This suggests a slight trade-off towards better latent space regularization with minimal impact on reconstruction quality.

4. $\beta = 10$: At this higher β value, the mean reconstruction error increases to 0.198% for both validation and testing sets, with even lower standard deviations. This higher β value puts more emphasis on the KL divergence term, leading to a less focused reconstruction but a strong regularization of the latent space.

3.3.2 Latent space visualization and Generations Images

With $\beta = 0.02$, the latent space is less regularized, leading to limited diversity and novelty in generated images. Image generation may closely resemble the training data, but the effectiveness of generalization is compromised. At $\beta = 1$, the latent space is well-regularized, striking a balance between reconstruction and generalization. With $\beta = 1.5$, the latent space regularization is slightly improved, which may lead to more coherent and semantically meaningful variations in the generated samples. The generated images might be slightly less accurate in terms of reconstruction, but the overall diversity and structure are enhanced. At $\beta = 10$, the latent space is highly regularized, resulting in a well-structured latent representation.

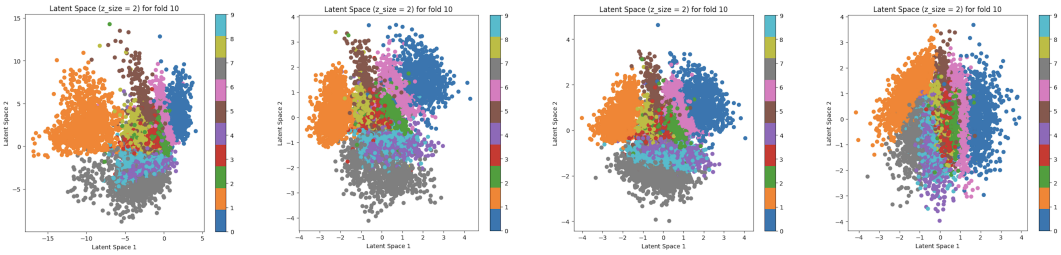

Figure 32: Impact of β in latent space, $\beta = 0.02$ in Figure28, $\beta = 1$ in Figure29, $\beta = 1.5$ in Figure30, $\beta = 10$ in Figure31

Figure 37: Impact of β in Image generation, $\beta = 0.02$ in Figure33, $\beta = 1$ in Figure34, $\beta = 1.5$ in Figure35, $\beta = 10$ in Figure36

4 Discussion and Conclusion

In this work, we have thoroughly explored autoencoders (AEs) and variational autoencoders (VAEs), two powerful and versatile tools in the field of unsupervised machine learning. We presented the theoretical foundations of these models, explaining their ability to learn compressed and meaningful representations of input data. We highlighted the key distinctions between AEs and VAEs: the latent distribution constraint imposed by VAEs, the difference in the resulting inference algorithm, and the flexibility of the probabilistic framework, which allows VAEs to generate new and realistic data.

We examined the practical aspects of implementing AEs and VAEs, focusing on the impact of network architectures, the regularization parameter β, and the VAE priors on model performance. Our experiments on MNIST, Fashion-MNIST, and FreyFace datasets demonstrated how these factors influence the evolution of the loss functions during training and validation, their impact on the latent space and the output reconstruction accuracy.

Furthermore, we identified several promising future research directions, such as improving network architectures for better data reconstruction and generation, exploring new regularization methods, and adapting VAEs to specific applications requiring more performant generative models.

Acknowledgments

We thank La Caixa Junior Leader program's LCF/BQ/PI22/11910028 award for financial support of this work.

References

[1] Andrea Asperti and Matteo Trentin. Balancing reconstruction error and kullback-leibler divergence in variational autoencoders. *IEEE Access*, 8:199440–199448, 2020.

[2] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[3] Monica Casella, Pasquale Dolce, Michela Ponticorvo, Davide Marocco, et al. Autoencoders as an alternative approach to principal component analysis for dimensionality reduction. an application on simulated data from psychometric models. In *PSYCHOBIT*, 2021.

[4] Norma Cristina Cueto Ceilis and Hanna Peters. An empirical comparison of generative capabilities of gan vs vae, 2022.

[5] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[6] John Duchi. Derivations for linear algebra and optimization. *Berkeley, California*, 3(1): 2325–5870, 2007.

[7] Gabriel Fernández Fernández. Extracting relevant physical parameters with ß-variational autoencoders. Master's thesis, Universitat Politècnica de Catalunya, 2020.

[8] B Ghojogh, A Ghodsi, F Karray, and M Crowley. Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder. *Tutorial and Survey*, 2021.

[9] Benyamin Ghojogh, Hadi Nekoei, Aydin Ghojogh, Fakhri Karray, and Mark Crowley. Sampling algorithms, from survey sampling to monte carlo methods: Tutorial and literature review. *arXiv preprint arXiv:2011.00901*, 2020.

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[11] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*, 2016.

[12] Shuiwang Ji, Z Wuang, and Ying Wei. Principal component analysis and autoencoders.

[13] Tom Joy, Sebastian M Schmon, Philip HS Torr, N Siddharth, and Tom Rainforth. Capturing label characteristics in vaes. *arXiv preprint arXiv:2006.10102*, 2020.

[14] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[15] Umberto Michelucci. An introduction to autoencoders. *arXiv preprint arXiv:2201.03898*, 2022.

[16] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

[17] Stephen Odaibo. Tutorial: Deriving the standard variational autoencoder (vae) loss function. *arXiv preprint arXiv:1907.08956*, 2019.

[18] Walter Hugo Lopez Pinaya, Sandra Vieira, Rafael Garcia-Dias, and Andrea Mechelli. Autoencoders. In *Machine learning*, pages 193–208. Elsevier, 2020.

[19] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

[20] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[21] Tim Salimans, Diederik Kingma, and Max Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International conference on machine learning*, pages 1218–1226. PMLR, 2015.

A Appendix

A.1 Description of loss functon: Autoencoders

Mean Squared Error: In the field of regression problem solving, the mean square error (MSE) is the predominant and widely adopted choice for the loss function. The Mean Squared Error is also employed when the input data consists of independent real-valued observations. Maximizing the log-likelihood for a Gaussian distribution leads to using MSE as the loss function.

The formula for mean square error loss for all observations $X_{in} = \{x_{in_n}; n = 1, \cdots, N\}$ and the corresponding AE's outputs $X_{out} = \{x_{out_n}; n = 1, \cdots, N\}$ is:

$$\mathcal{L}_{MSE}(X_{out}, X_{in}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{D}\sum_{d=1}^{D}\left(x_{in_{n,d}} - x_{out_{n,d}}\right)^2 \tag{11}$$

where N is the total number of observations; D is the number of dimensions or components in the vectors x_{in_i} and x_{out_i} for the i^{th} observations; $x_{in_{i,d}}$ is the d^{th} component of the i^{th} observation, and $x_{out_{i,d}}$ represents the d^{th} component of the i^{th} predicted value.

MSE measures the average squared difference between the observation (x_{in}) and predicted value (x_{out}) in each dimension.

Mean Absolute Error/L1 loss: Maximizing the log-likelihood for a Laplace distribution leads to using MAE as the loss function. To calculate the MAE, you take the difference between all observations x_{in} and model prediction x_{out} and average it across the whole dataset.

$$\mathcal{L}_{MAE}(X_{out}, X_{in}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{D}\sum_{d=1}^{D}|x_{in_d} - x_{out_d}| \tag{12}$$

This loss function is used as an alternative to MSE in some cases. MSE is highly sensitive to outliers, which can dramatically affect the loss because the distance is squared. MAE is used in cases when the training data has a large number of outliers to mitigate this.

Binary cross-entropy: For classification problems, the negative log likelihood loss is the most common and widely used choice for the loss function. Specifically, in binary classification scenarios, this loss is generally called the binary cross entropy. Additionally, it's important to observe that when the activation function of the output layer is a sigmoid function, constraining the neuron outputs to a range between 0 and 1, and the input features fall within the range of 0 to 1 or normalised to be between 0 and 1, the binary cross-entropy, viewed as the negative log likelihood for a Bernoulli distribution, becomes a suitable choice for the loss function. Here, the output of the model is interpreted as probabilities assigned to binary classes.
The formula for the mean of Binary Cross-Entropy (BCE) losses across all observations, with averaging over individual elements, is expressed as:

$$\mathcal{L}_{BCE_{mean}}(X_{out}, X_{in}) = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{D}\sum_{d=1}^{D} x_{in_{i,d}}\log(x_{out_{i,d}}) - (1 - x_{in_{i,d}})\log(1 - x_{out_{i,d}})) \tag{13}$$

where N is the total number of observations; D is the number of dimensions or components in the vectors x_{in_i} and x_{out_i} for the i^{th} observations; $x_{in_{i,d}}$ is the d^{th} component of the i^{th} observation and $x_{out_{i,d}}$ represent the d^{th} component of the i^{th} predicted probabilities. BCE measures the dissimilarity between observation (x_{in}) and predicted probabilities (x_{out}). The summation or averaging

is performed over all observations and components, assuming that observations are independent and identically distributed (iid) across the dataset. The predictive accuracy of the model is closely linked to the predicted probabilities (x_{out}) it assigns. When the predicted probabilities (x_{out}) are low, the associated Binary Cross Entropy (BCE) loss increases, indicating that the predictions are poor. Conversely, when the model makes good predictions with higher probabilities, the resulting BCE loss decreases significantly.

Evolution of BCE Loss in Autoencoders on MNIST, and FreyFace

This figure 40 examines the evolution of BCE loss in an AE with 1-layer and a latent space dimension of $Dz = 2$, applied to the MNIST and FreyFace datasets. All training and validation plots go down.

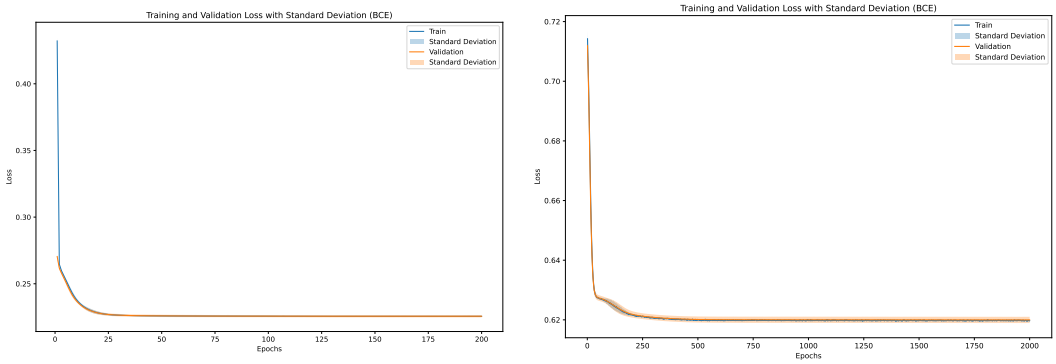

Figure 40: Evolution of the Loss Function (BCE) during Training and Validation of the Autoencoder on MNIST (Figure 38) and FreyFace (Figure 39).

B Description of ELBO: Varational Autoencoder

B.1 Evidence Lower Bound (ELBO)

An Evidence Lower Bound (ELBO), also known as the variational lower bound that uses the approximate posterior $q_\phi(\mathbf{z}|x)$ and the conditional distribution $p_\theta(\mathbf{z}|x)$ is derived in the Equations (26) and (31). It aims to provide a tractable approximation of the intractable posterior distribution. The KL divergence between the approximate and the real posterior distributions is given by:

$$D_{KL}(q_\phi(\mathbf{z}|x)||p_\theta(\mathbf{z}|x)) = \int q_\phi(\mathbf{z}|x) \log \frac{q_\phi(\mathbf{z}|x)}{p_\theta(\mathbf{z}|x)} \, d\mathbf{z} \tag{14}$$

Note that KL divergence is always positive and equation (14) can also be written as follows

$$D_{KL}(q_\phi(\mathbf{z}|x)||p_\theta(\mathbf{z}|x)) = -\int q_\phi(\mathbf{z}|x) \log \frac{p_\theta(\mathbf{z}|x)}{q_\phi(\mathbf{z}|x)} \, d\mathbf{z} \geq 0 \tag{15}$$

Applying Bayes' theorem :

$$p_\theta(\mathbf{z}|x) = \frac{p_\theta(x|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(x)} \tag{16}$$

Substitution 15 in 16, we obtain :

$$D_{KL}(q_\phi(\mathbf{z}|x)||p_\theta(\mathbf{z}|x)) = -\int q_\phi(\mathbf{z}|x) \{\log \frac{p_\theta(x|\mathbf{z})p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|x)p_\theta(x)}\} \, d\mathbf{z} \geq 0 \tag{17}$$

This can be simplified by applying logarithmic laws, resulting in :

$$-\int q_\phi(\mathbf{z}|x) \{\log \frac{p_\theta(x|\mathbf{z})p_\theta(z)}{q_\phi(\mathbf{z}|x)} - \log p_\theta(x)\} \, d\mathbf{z} \geq 0 \tag{18}$$

By expanding, we get:

$$-\int q_\phi(\mathbf{z}|x)\left(\log\frac{p_\theta(x|\mathbf{z})p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|x)}\right)\,d\mathbf{z}+\int q_\phi(\mathbf{z}|x)\log p_\theta(x)\,d\mathbf{z}\geq 0 \tag{19}$$

Since, $P_\theta(x)$ does not involve $\mathbf{z}$, then $\log(P_\theta(x))$ is a constant, it can be moved out.

Hence,

$$-\int q_\phi(\mathbf{z}|x)\left(\log\frac{p_\theta(x|\mathbf{z})p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|x)}\right)\,d\mathbf{z}+\log p_\theta(x)\int q_\phi(\mathbf{z}|x)\,d\mathbf{z}\geq 0 \tag{20}$$

We see that $\int q_\phi(\mathbf{z}|x)\,d\mathbf{z}$ integrates to 1, since this is simply integrating the density function of a multivariate probability distribution.

$$D_{KL}(q_\phi(\mathbf{z}|x)||p_\theta(\mathbf{z}|x))=-\int q_\phi(\mathbf{z}|x)\left(\log\frac{p_\theta(x|\mathbf{z})p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|x)}\right)\,d\mathbf{z}+\log p_\theta(x)\geq 0 \tag{21}$$

Then, carrying the integral over to the other side of the inequality, we get,

$$\log p_\theta(x)\geq\int q_\phi(\mathbf{z}|x)\left(\log\frac{p_\theta(x|\mathbf{z})p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|x)}\right)\,d\mathbf{z} \tag{22}$$

Now, let's discuss the two methods for computing 22.

First Method for Computing ELBO

Applying rules of logarithms on the right side of 22, we get,

$$\log p_\theta(x)\geq\int q_\phi(\mathbf{z}|x)\left(\log p_\theta(x|\mathbf{z})+\log p_\theta(\mathbf{z})-\log q_\phi(\mathbf{z}|x)\right)\,d\mathbf{z} \tag{23}$$

Recognizing the right-hand side of the above inequality as

$$\log p_\theta(x)\geq\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|x)}\left\{\log p_\theta(x|\mathbf{z})+\log p_\theta(\mathbf{z})-\log q_\phi(\mathbf{z}|x)\right\} \tag{24}$$

Applying rules of logarithms, we obtain,

$$\log p_\theta(x)\geq\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|x)}\left(\log(p_\theta(x|\mathbf{z})p_\theta(\mathbf{z}))-\log q_\phi(\mathbf{z}|x)\right) \tag{25}$$

Hence,

$$\log p_\theta(x)\geq\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|x)}\left(\log(p_\theta(x,\mathbf{z}))-\log q_\phi(\mathbf{z}|x)\right) \tag{26}$$

ELBO can therefore be written as follows

$$\mathcal{L}(\phi,\theta,x,\mathbf{z})=\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|x)}\left(\log(p_\theta(x,\mathbf{z}))-\log q_\phi(\mathbf{z}|x)\right) \tag{27}$$

Equation (27) represents the first equation of the Evidence Lower Bound (ELBO).

Second Method for Computing ELBO

From Equation (22), applying the rules of logarithms, it also follows that:

$$\log p_\theta(\boldsymbol{x})\geq\int q_\phi(\mathbf{z}|x)\left(\log\frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|x)}+\log p_\theta(x|\mathbf{z})\right)\,d\mathbf{z} \tag{28}$$

By expanding, we obtain:

$$\log p_\theta(x)\geq\int q_\phi(\mathbf{z}|x)\log\frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|x)}\,d\mathbf{z}+\int q_\phi(\mathbf{z}|x)\log p_\theta(x|\mathbf{z})\,d\mathbf{z} \tag{29}$$

By taking the inverse of the first term on the right side,

$$\log p_\theta(x)\geq-\int q_\phi(\mathbf{z}|x)\log\frac{q_\phi(\mathbf{z}|x)}{p_\theta(\mathbf{z})}\,d\mathbf{z}+\int q_\phi(\mathbf{z}|x)\log p_\theta(x|\mathbf{z})\,d\mathbf{z} \tag{30}$$

Hence

$$\log p_\theta(x)\geq-D_{KL}(q_\phi(\mathbf{z}|x)||p_\theta(\mathbf{z}))+\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|x)}\{\log p_\theta(\boldsymbol{x}|z)\} \tag{31}$$

ELBO can therefore be written as follows

$$\mathcal{L}(\phi,\theta,x,\mathbf{z})=-D_{KL}(q_\phi(\mathbf{z}|x)||p_\theta(\mathbf{z}))+\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|x)}\{\log p_\theta(\boldsymbol{x}|z)\} \tag{32}$$

Equation (32) represents the second equation of the Evidence Lower Bound (ELBO).

ELBO 1

Let's first consider equation (27),

$$\mathcal{L}(\phi, \theta, x, \mathbf{z}) = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|x)} \left(\log(p_\theta(x, \mathbf{z})) - \log q_\phi(\mathbf{z}|x) \right) \tag{33}$$

The Monte Carlo approximation [9] can be used to approximate this expectation, where we draw L samples $\{\mathbf{z}_{i,l}\}_{l=1}^{L}$, corresponding to the $i-th$ data point, from the conditional distribution.

$$\mathbf{z}^{(i,l)} \sim q_\phi(\mathbf{z}|x^{(i)}) \quad \forall \{1, 2, \cdots, L\} \tag{34}$$

In general, the Monte Carlo approximation [9] approximates expectations as follows:

$$\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}^i|x)}(g(\mathbf{z}^{(i)})) \approx \frac{1}{L} \sum_{l=1}^{L} g(\mathbf{z}^{(i,l)}) \tag{35}$$

where $g(\mathbf{z}^{(i)})$ is a function of $\mathbf{z}^{(i)}$.

Monte Carlo estimation of ELBO 1

$$\mathcal{L}(\phi, \theta, x, \mathbf{z}) = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|x)} \left(\log(p_\theta(x, \mathbf{z})) - \log q_\phi(\mathbf{z}|x) \right) \tag{36}$$

The Monte Carlo approximation is used because in practice, ELBO makes it intractable to compute the expectation analytically. Instead, the expectation is approximated by drawing L samples $\mathbf{z}^{(i,l)} \sim q_\phi(\mathbf{z}|x^{(i)})$:

$$\frac{1}{L} \sum_{l=1}^{L} \Big(\log(p_\theta(x^{(i)}, \boldsymbol{z}^{(\boldsymbol{i,l})})) - \log q_\phi(\boldsymbol{z}^{(\boldsymbol{i,l})}|x^{(i)}) \Big) \tag{37}$$

Therefore, ELBO becomes

$$\mathcal{L}(\phi, \theta, x, \mathbf{z}) \approx \hat{\mathcal{L}}_A \Big(\phi, \theta, x^{(i)}, \mathbf{z}, L \Big) = \frac{1}{L} \sum_{l=1}^{L} \Big(\log(p_\theta(x^{(i)}, \boldsymbol{z}^{(\boldsymbol{i,l})})) - \log q_\phi(\boldsymbol{z}^{(\boldsymbol{i,l})}|x^{(i)}) \Big) \tag{38}$$

ELBO 2

Let's second consider equation (32),

$$\mathcal{L}(\phi, \theta, x, \mathbf{z}) = -D_{KL}(q_\phi(\mathbf{z}|x)||p_\theta(\mathbf{z})) + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|x)} \{\log p_\theta(\boldsymbol{x}|z)\} \tag{39}$$

Using the Monte Carlo approximation [9], we can estimate the second term in the equation by taking L samples $\{\mathbf{z}_{i,l}\}_{l=1}^{L}$ from $q_\phi(\mathbf{z}|x^i)$.

Monte Carlo estimation of ELBO 2

The expectation is approximated by drawing L samples $\mathbf{z}^{(i,l)} \sim q_\phi(\mathbf{z}|x^{(i)})$:

$$\frac{1}{L} \sum_{l=1}^{L} \log p_\theta(x^{(i)}|\mathbf{z}^{(i,l)}) \tag{40}$$

Therefore, ELBO becomes

$$\mathcal{L}(\phi, \theta, x, \mathbf{z}) \approx \hat{\mathcal{L}}_B \Big(\phi, \theta, x^{(i)}, \mathbf{z}, L \Big) = -D_{KL}(q_\phi(\mathbf{z}|x^{(i)})||p_\theta(\mathbf{z})) + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(x^{(i)}|\mathbf{z}^{(i,l)}) \tag{41}$$

The first term of the equation can be converted to an expectation and, again, we can use the Monte Carlo approximation [9] to compute it and draw L samples. $\{\mathbf{z}_{i,l}\}_{l=1}^{L}$ from $q_\phi(\mathbf{z}|x^{(i)})$.

$$-D_{KL}(q_\phi(\mathbf{z}|x^{(i)})||p_\theta(\mathbf{z})) = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|x)} \{\log \frac{q_\phi(\mathbf{z}|x)}{p_\theta(\mathbf{z})}\} \approx -\frac{1}{L} \sum_{l=1}^{L} \log q_\phi(\mathbf{z}^{(i,l)}|x^{(i)}) - \log p_\theta(\mathbf{z}^{(i,l)}|x^{(i)}) \tag{42}$$

Then, ELBO can also write

$$\hat{\mathcal{L}}_B\Big(\phi,\theta,x^{(i)},\mathbf{z},L\Big) = -\frac{1}{L}\sum_{l=1}^{L}\log q_\phi(\mathbf{z}^{(i,l)}|x^{(i)}) - \log p_\theta(\mathbf{z}^{(i,l)}|x^{(i)}) + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(x^{(i)}|\mathbf{z}^{(i,l)}) \tag{43}$$

The first term in Equation (41) may be computed analytically if we have some families of distributions, such as the Gaussian distribution for $q_\phi(\mathbf{z}^{(i,l)}|x^{(i)})$ and $p_\theta(\mathbf{z}^{(i,l)}|x^{(i)})$. In what follows, we simply extend Equation. (41) further for Gaussian distribution.

B.2 KL divergence for special case of Gaussian distribution

For univariate or multivariate Gaussian distributions, we can analytically calculate the KL divergence in the first term of Equation (41). To do this, we require the two lemmas below:

The KL divergence between two univariate Gaussian distribution $p_1 \sim \mathcal{N}(\mu_1;\sigma_1^2)$ and $p_2 \sim \mathcal{N}(\mu_2;\sigma_2^2)$ is:

$$D_{KL}(p_1||p_2) = \log\frac{\sigma_2}{\sigma_1} + \frac{\sigma_1^2+(\mu_1-\mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \tag{44}$$

Proof. See ([8]) for proof.

The KL divergence between two multivariate Gaussian distribution $p_1 \sim \mathcal{N}(\mu_1;\Sigma_1)$ and $p_2 \sim \mathcal{N}(\mu_2;\Sigma_2)$ with dimensionality p is:

$$D_{KL}(p_1||p_2) = \frac{1}{2}\left(\log\frac{|\Sigma_2|}{|\Sigma_1|} - p + \mathrm{tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2-\mu_1)^{\mathrm{T}}\Sigma_2^{-1}(\mu_2-\mu_1)\right) \tag{45}$$

Note that $\Sigma = \sigma^2 I$.

Proof. See ([6], Section 9) for proof.

Example: Gaussian Distributions

We now illustrate the VAE with Gaussian distributions. Suppose the prior distribution is a centered isotropic multivariate Gaussian, , denoted by $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z};0;I)$, with no parameters. In addition, consider the conditional distribution, which is modelled as a multivariate Gaussian, denoted by $p_\theta(x^{(i)}|\mathbf{z}) = \mathcal{N}(x,\mathbf{z};\boldsymbol{\mu};\Sigma)$. Then, we can assume that the true posterior is simular to a multivariate Gaussian with a diagonal covariance structure, which leads us to choose this distribution as an approximation for the posterior $q_\phi(\mathbf{z}|x^{(i)}) = \mathcal{N}(\mathbf{z};\boldsymbol{\mu}^{(i)};\sigma^{2(i)}I)$. By applied Equation (45), then the Kullback-Leibler divergence simplifies to:

$$D_{KL}(q_\phi(\mathbf{z}|x^{(i)})||p_\theta(\mathbf{z})) = -\frac{1}{2}\sum_{j=1}^{J}\{1+\log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\} \tag{46}$$

Let J be the dimensionality of $\mathbf{z}$. This equation (46) was demonstrated by [17]. Then, the resulting ELBO, i.e Equation (41) becomes:

$$\hat{\mathcal{L}}_B\Big(\phi,\theta,x^{(i)},\mathbf{z},L\Big) = \frac{1}{2}\sum_{j=1}^{J}\{1+\log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\} + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(x^{(i)}|\mathbf{z}^{i,l})\} \tag{47}$$

where and σ_j^2 and μ_j are parameters into the approximate distribution, q, and j is an index into the latent vector $\mathbf{z}$, and L is the number of samples.

B.3 Evolution of ELBO in VAE on FreyFace, MNIST and FashionMNIST

This figure 43 examines the evolution of ELBO in an VAE with 2-layer and a latent space dimension of $Dz = 2$, applied to the MNIST and FashionMNIST datasets, and Figure 43 for the FreyFace dataset. All training and validation plots decrease.

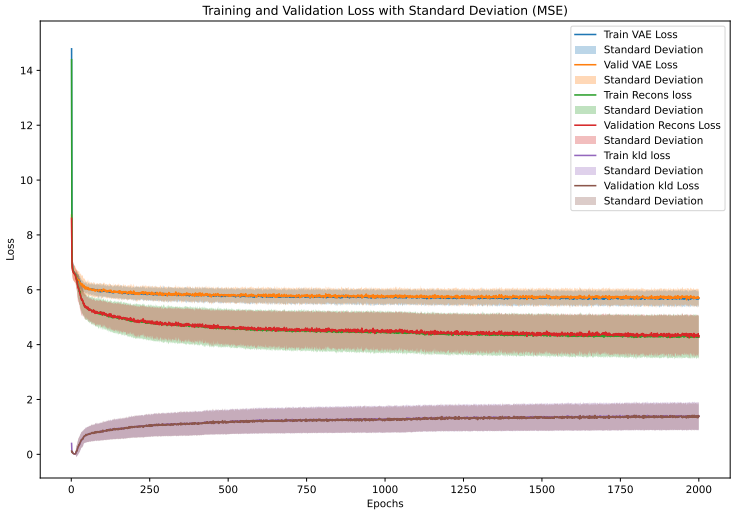

Figure 41: Evolution of ELBO during training and MSE on validation applied to the FreyFace.

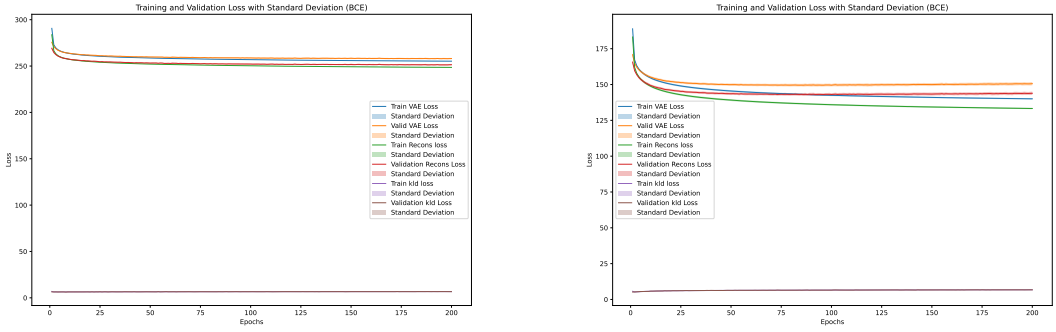

Figure 44: Evolution of ELBO during training and BCE on validation using FashionMNIST (Figure 42) and MNIST (Figure 43).

C Additional Results

C.1 Comparative Studies between Autoencoders and Variational Autoencoders

Impact of Latent Space Dimensionality on Single-Layer Autoencoder Performance:

Figure 48: The top row shows the original digits from the MNIST dataset. The row below presents the reconstructed digits using a 1-layer with Dz=2 in Figure 45, Dz=8 in Figure 46 and Dz=64 in Figure 47.

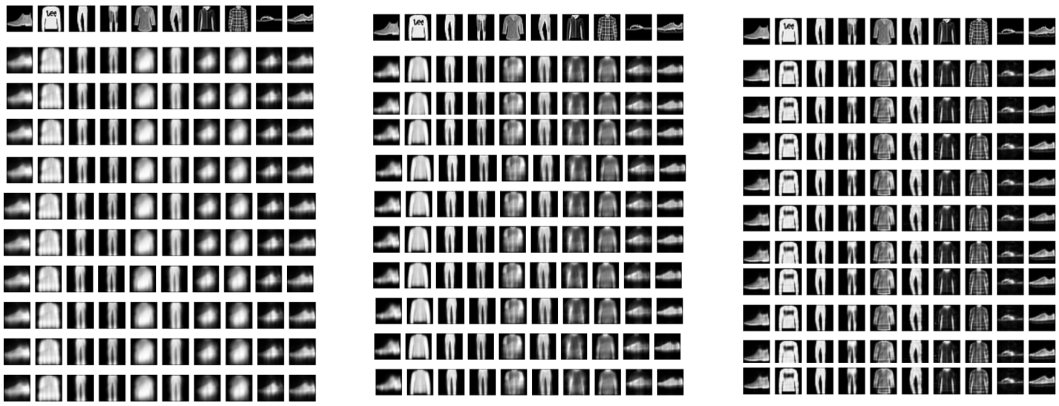

Figure 52: In the top line, we can see the original images from the FashionMNIST dataset. The line below shows the images reconstructed using a 1-layer model with Dz=2 in Figure 49, Dz=8 in Figure 50, and Dz=64 in Figure 51.

Figure 56: The top row displays the original faces from the FreyFace dataset. Below, the reconstructed faces are shown using a 1-layer with Dz=2 in Figure 53, Dz=8 in Figure 54 and Dz=64 in Figure 55.

Impact of Latent Space Dimensionality on Single-Layer Variational Autoencoder Performance:

Figure 60: In the top line, we can see the original digits from the MNIST dataset. The line of digits below contains the digits reconstructed by 1-layer with Dz=2 in Figure 57, Dz=8 in Figure 58 and Dz=64 in Figure 59.

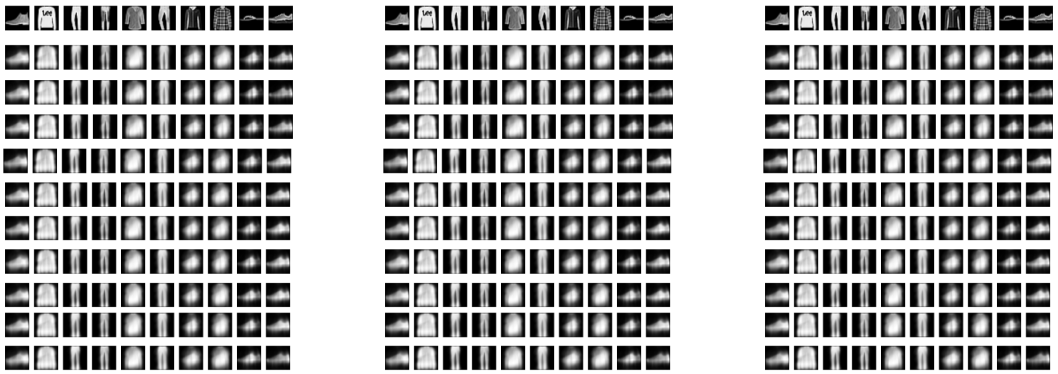

Figure 64: The top row displays the original Fashion images. The row below shows the Fashion images reconstructed using a 1-layer with with Dz=2 in Figure 61, Dz=8 in Figure 62 and Dz=64 in Figure 63.

Figure 68: The top line shows the original Face images. The line below displays the Face images reconstructed using a 1-layer with Dz=2 in Figure 65, Dz=8 in Figure 66 and Dz=64 in Figure 67.

C.1.1 Comparison of Reconstruction Performance

Model ($\mathbf{n}_L$ / $\mathbf{D}_z$)	**MNIST, D_x: 784** **(Valid / Testing)**	**FashionMNIST, D_x: 784** **(Valid / Testing)**	**FreyFace, D_x: 560** **(Valid / Testing)**	**Nbres of Params** **(Unlabel / label)**
AE, 1 / 2	**0.226 ± 0.089% / 0.225 ± 0.041%**	**0.375 ± 0.061% / 0.377 ± 0.088%**	**0.620 ± 0.107% / 0.620 ± 0.007%**	3922 / 2802
VAE, 1 / 2	0.227 ± 0.091% / 0.226 ± 0.066%	0.376 ± 0.065% / 0.378 ± 0.180%	0.624 ± 0.099% / 0.624 ± 0.016%	5492 / 3924
AE, 1 / 8	**0.164 ± 0.056% / 0.163 ± 0.048%**	**0.314 ± 0.077% / 0.316 ± 0.056%**	**0.611 ± 0.110% / 0.612 ± 0.002%**	13336 / 9528
VAE, 1 / 8	0.170 ± 0.051% / 0.168 ± 0.060%	0.319 ± 0.070% / 0.321 ± 0.200%	0.623 ± 0.098% / 0.623 ± 0.022%	19616 / 14016
AE, 1 / 64	**0.073 ± 0.022% / 0.072 ± 0.026%**	**0.272 ± 0.075% / 0.274 ± 0.054%**	**0.604 ± 0.118% / 0.604 ± 0.002%**	101200 / 72304
VAE, 1 / 64	0.108 ± 0.030% / 0.107 ± 0.160%	0.304 ± 0.064% / 0.306 ± 0.320%	0.623 ± 0.102% / 0.624 ± 0.034%	151440 / 108208

Table 3: AE and VAE with BCE in validation and testing table for MNIST, FashionMNIST, and FreyFace datasets.

Model (n_L / D_z)	MNIST, D_x: 784 (Valid / Testing)	FashionMNIST, D_x: 784 (Valid / Testing)	FreyFace, D_x: 560 (Valid / Testing)	Nbres of Params (Unlabel / label)
AE, 1 / 2	**0.056 ± 0.028% / 0.055 ± 0.003%**	**0.045 ± 0.018% / 0.045 ± 0.002%**	**0.008 ± 0.040% / 0.008 ± 0.004%**	3922 / 2802
VAE, 1 / 2	0.056 ± 0.028% / 0.056 ± 0.002%	0.045 ± 0.018% / 0.045 ± 0.006%	0.010 ± 0.042% / 0.010 ± 0.008%	5492 / 3924
AE, 1 / 8	**0.034 ± 0.015% / 0.033 ± 0.002%**	**0.023 ± 0.012% / 0.023 ± 0.003%**	**0.004 ± 0.021% / 0.004 ± 0.001%**	13336 / 9528
VAE, 1 / 8	0.036 ± 0.013% / 0.035 ± 0.003%	0.025 ± 0.023% / 0.025 ± 0.017%	0.009 ± 0.036% / 0.009 ± 0.014%	19616 / 14016
AE, 1 / 64	**0.004 ± 0.002% / 0.004 ± 0.001%**	**0.009 ± 0.006% / 0.009 ± 0.003%**	**0.001 ± 0.006% / 0.001 ± 0.001%**	101200 / 72304
VAE, 1 / 64	0.015 ± 0.008% / 0.014 ± 0.007%	0.020 ± 0.013% / 0.020 ± 0.015%	0.009 ± 0.050% / 0.009 ± 0.011%	151440 / 108208

Table 4: AE and VAE with MSE in validation and testing table for MNIST, FashionMNIST, and FreyFace datasets.

Model (n_L / D_z)	MNIST, D_x: 784 (Valid / Testing)	FashionMNIST, D_x: 784 (Valid / Testing)	FreyFace, D_x: 560 (Valid / Testing)	Nbres of Params (Unlabel / label)
AE, 1 / 2	**0.125 ± 0.043% / 0.125 ± 0.022%**	**0.138 ± 0.043% / 0.138 ± 0.016%**	**0.063 ± 0.002% / 0.063 ± 0.000%**	3922 / 2802
VAE, 1 / 2	0.126 ± 0.043% / 0.126 ± 0.024%	0.139 ± 0.025% / 0.140 ± 0.022%	0.070 ± 0.001% / 0.070 ± 0.000%	5492 / 3924
AE, 1 / 8	**0.083 ± 0.023% / 0.082 ± 0.011%**	**0.088 ± 0.022% / 0.089 ± 0.011%**	**0.043 ± 0.000% / 0.043 ± 0.001%**	13336 / 9528
VAE, 1 / 8	0.086 ± 0.029% / 0.086 ± 0.021%	0.093 ± 0.036% / 0.094 ± 0.032%	0.068 ± 0.002% / 0.068 ± 0.000%	19616 / 14016
AE, 1 / 64	**0.018 ± 0.007% / 0.018 ± 0.003%**	**0.051 ± 0.013% / 0.051 ± 0.006%**	**0.019 ± 0.000% / 0.020 ± 0.001%**	101200 / 72304
VAE, 1 / 64	0.044 ± 0.015% / 0.044 ± 0.012%	0.081 ± 0.028% / 0.081 ± 0.031%	0.068 ± 0.001% / 0.069 ± 0.001%	151440 / 108208

Table 5: AE and VAE with L1 Loss in validation and testing table for MNIST, FashionMNIST, and FreyFace datasets.

Impact of Number of Layers on Performance of AEs and VAEs with Two-Dimensional Latent Space

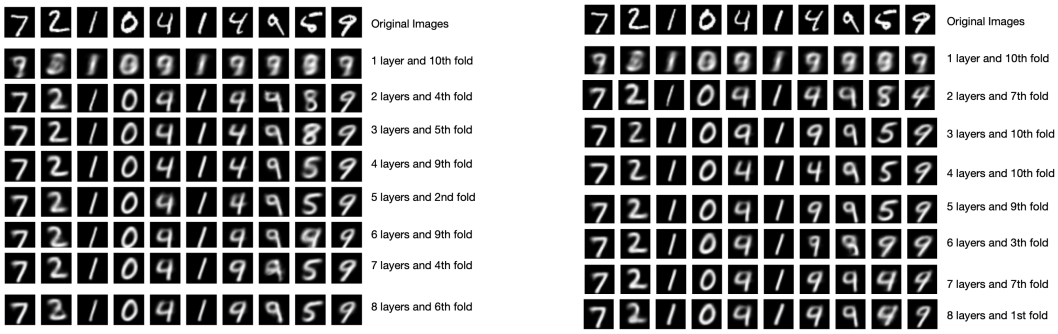

Figure 69: Image reconstruction: the left side shows results from an AE, while the right side displays results from a VAE using the MNIST dataset.

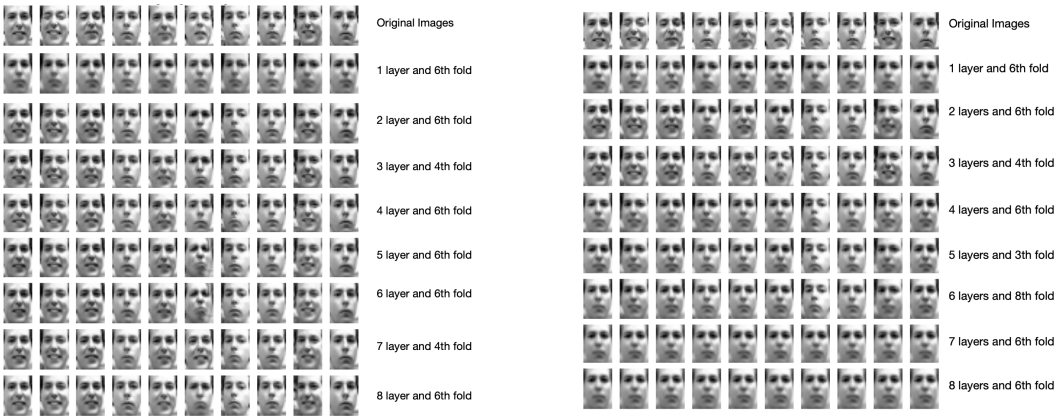

Figure 70: Image reconstruction: the left side presents results from an AE, while the right side shows results from a VAE applied to the FreyFace dataset.

C.1.2 Comparison of Reconstruction Performance

Model (n_L / D_z)	MNIST, D_x: 784 (Valid / Testing)	FashionMNIST, D_x: 784 (Valid / Testing)	FreyFace, D_x: 560 (Valid / Testing)	Nbres of Params (Unlabel / label)
AE, 1 / 2	0.226 ± 0.089% / 0.225 ± 0.041%	0.375 ± 0.061% / 0.377 ± 0.088%	0.620 ± 0.107% / 0.620 ± 0.007%	3922 / 2802
VAE, 1 / 2	0.227 ± 0.091% / 0.226 ± 0.066%	0.376 ± 0.065% / 0.378 ± 0.180%	0.624 ± 0.099% / 0.624 ± 0.016%	5492 / 3924
AE, 2 / 2	0.182 ± 0.127% / 0.181 ± 0.083%	0.318 ± 0.109% / 0.320 ± 0.047%	0.612 ± 0.121% / 0.611 ± 0.020%	619368 / 316968
VAE, 2 / 2	0.184 ± 0.198% / 0.183 ± 0.166%	0.321 ± 0.086% / 0.323 ± 0.023%	0.620 ± 0.003% / 0.620 ± 0.003%	620156 / 317532
AE, 3 / 2	0.166 ± 0.171% / 0.165 ± 0.166%	0.312 ± 0.120% / 0.314 ± 0.090%	0.612 ± 0.001% / 0.609 ± 0.000%	1100668 / 561942
VAE, 3 / 2	0.167 ± 0.107% / 0.166 ± 0.060%	0.314 ± 0.227% / 0.316 ± 0.240%	0.621 ± 0.003% / 0.621 ± 0.003%	1101196 / 562320
AE, 4 / 2	0.167 ± 0.623% / 0.167 ± 0.649%	0.317 ± 0.357% / 0.319 ± 0.354%	0.612 ± 0.001% / 0.609 ± 0.001%	1546072 / 790744
VAE, 4 / 2	0.164 ± 0.189% / 0.164 ± 0.155%	0.317 ± 0.288% / 0.319 ± 0.288%	0.628 ± 0.001% / 0.629 ± 0.000%	1546470 / 791030
AE, 5 / 2	0.172 ± 0.366% / 0.172 ± 0.337%	0.317 ± 0.277% / 0.319 ± 0.259%	0.612 ± 0.001% / 0.611 ± 0.002%	1979636 / 1012628
VAE, 5 / 2	0.173 ± 0.284% / 0.172 ± 0.287%	0.318 ± 0.225% / 0.320 ± 0.174%	0.628 ± 0.001% / 0.629 ± 0.000%	1979956 / 1012858
AE, 6 / 2	0.175 ± 0.210% / 0.174 ± 0.195%	0.321 ± 0.341% / 0.323 ± 0.303%	0.612 ± 0.001% / 0.610 ± 0.001%	2404656 / 1227672
VAE, 6 / 2	0.177 ± 0.299% / 0.177 ± 0.310%	0.322 ± 0.321% / 0.324 ± 0.279%	0.628 ± 0.001% / 0.629 ± 0.000%	2404924 / 1227864
AE, 7 / 2	0.175 ± 0.455% / 0.174 ± 0.414%	0.321 ± 0.535% / 0.323 ± 0.534%	0.613 ± 0.003% / 0.611 ± 0.003%	2827940 / 1446532
VAE, 7 / 2	0.178 ± 0.245% / 0.178 ± 0.278%	0.321 ± 0.384% / 0.323 ± 0.363%	0.628 ± 0.001% / 0.629 ± 0.000%	2828170 / 1446698
AE, 8 / 2	0.180 ± 0.656% / 0.180 ± 0.659%	0.323 ± 0.729% / 0.325 ± 0.732%	0.617 ± 0.003% / 0.616 ± 0.004%	3246992 / 1660736
VAE, 8 / 2	0.181 ± 0.665% / 0.181 ± 0.690%	0.321 ± 0.253% / 0.323 ± 0.261%	0.628 ± 0.001% / 0.629 ± 0.000%	3247194 / 1660882

Table 6: AEs and VAEs with BCE in validation and testing table for MNIST, FashionMNIST, and FreyFace datasets.

Model (n_L / D_z)	MNIST, D_x: 784 (Valid / Testing)	FashionMNIST, D_x: 784 (Valid / Testing)	FreyFace, D_x: 560 (Valid / Testing)	Nbres of Params (Unlabel / label)
AE, 1 / 2	0.056 ± 0.028% / 0.055 ± 0.003%	0.045 ± 0.018% / 0.045 ± 0.002%	0.008 ± 0.040% / 0.008 ± 0.004%	3922 / 2802
VAE, 1 / 2	0.056 ± 0.028% / 0.056 ± 0.002%	0.045 ± 0.018% / 0.045 ± 0.006%	0.010 ± 0.042% / 0.010 ± 0.008%	5492 / 3924
AE, 2 / 2	0.040 ± 0.045% / 0.040 ± 0.032%	0.026 ± 0.033% / 0.026 ± 0.023%	0.005 ± 0.033% / 0.004 ± 0.009%	619368 / 316968
VAE, 2 / 2	0.041 ± 0.046% / 0.041 ± 0.062%	0.027 ± 0.018% / 0.027 ± 0.011%	0.008 ± 0.135% / 0.008 ± 0.143%	620156 / 317532
AE, 3 / 2	0.034 ± 0.044% / 0.034 ± 0.039%	0.024 ± 0.041% / 0.024 ± 0.030%	0.004 ± 0.022% / 0.003 ± 0.012%	1100668 / 561942
VAE, 3 / 2	0.035 ± 0.030% / 0.035 ± 0.021%	0.025 ± 0.084% / 0.025 ± 0.083%	0.008 ± 0.124% / 0.008 ± 0.128%	1101196 / 562320
AE, 4 / 2	0.035 ± 0.210% / 0.035 ± 0.220%	0.026 ± 0.110% / 0.026 ± 0.115%	0.004 ± 0.029% / 0.003 ± 0.047%	1546072 / 790744
VAE, 4 / 2	0.034 ± 0.061% / 0.034 ± 0.055%	0.026 ± 0.106% / 0.026 ± 0.100%	0.012 ± 0.052% / 0.012 ± 0.001%	1546470 / 791030
AE, 5 / 2	0.037 ± 0.118% / 0.037 ± 0.110%	0.026 ± 0.088% / 0.026 ± 0.085%	0.005 ± 0.073% / 0.004 ± 0.109%	1979636 / 1012628
VAE, 5 / 2	0.037 ± 0.095% / 0.037 ± 0.097%	0.026 ± 0.064% / 0.026 ± 0.061%	0.012 ± 0.052% / 0.012 ± 0.001%	1979956 / 1012858
AE, 6 / 2	0.038 ± 0.067% / 0.038 ± 0.065%	0.027 ± 0.107% / 0.027 ± 0.102%	0.004 ± 0.028% / 0.003 ± 0.046%	2404656 / 1227672
VAE, 6 / 2	0.039 ± 0.103% / 0.039 ± 0.109%	0.028 ± 0.086% / 0.027 ± 0.085%	0.012 ± 0.053% / 0.012 ± 0.001%	2404924 / 1227864
AE, 7 / 2	0.038 ± 0.151% / 0.038 ± 0.141%	0.027 ± 0.183% / 0.027 ± 0.183%	0.005 ± 0.112% / 0.004 ± 0.147%	2827940 / 1446532
VAE, 7 / 2	0.039 ± 0.083% / 0.039 ± 0.094%	0.027 ± 0.124% / 0.027 ± 0.125%	0.012 ± 0.052% / 0.012 ± 0.001%	2828170 / 1446698
AE, 8 / 2	0.040 ± 0.225% / 0.040 ± 0.225%	0.028 ± 0.235% / 0.028 ± 0.237%	0.007 ± 0.150% / 0.006 ± 0.191%	3246992 / 1660736
VAE, 8 / 2	0.040 ± 0.227% / 0.040 ± 0.239%	0.027 ± 0.090% / 0.027 ± 0.088%	0.012 ± 0.053% / 0.012 ± 0.001%	3247194 / 1660882

Table 7: AEs and VAEs with MSE in validation and testing for MNIST, FashionMNIST, and FreyFace datasets.

Model (n_L / D_z)	MNIST, D_x: 784 (Valid / Testing)	FashionMNIST, D_x: 784 (Valid / Testing)	FreyFace, D_x: 560 (Valid / Testing)	Nbres of Params (Unlabel / label)
AE, 1 / 2	0.125 ± 0.043% / 0.125 ± 0.022%	0.138 ± 0.043% / 0.138 ± 0.016%	0.063 ± 0.165% / 0.063 ± 0.030%	3922 / 2802
VAE, 1 / 2	0.126 ± 0.043% / 0.126 ± 0.024%	0.139 ± 0.025% / 0.140 ± 0.022%	0.070 ± 0.100% / 0.070 ± 0.000%	5492 / 3924
AE, 2 / 2	0.091 ± 0.001% / 0.091 ± 0.001%	0.089 ± 0.001% / 0.089 ± 0.000%	0.044 ± 0.154% / 0.040 ± 0.050%	619368 / 316968
VAE, 2 / 2	0.093 ± 0.001% / 0.093 ± 0.001%	0.092 ± 0.000% / 0.092 ± 0.000%	0.060 ± 0.594% / 0.061 ± 0.621%	620156 / 317532
AE, 3 / 2	0.080 ± 0.001% / 0.080 ± 0.000%	0.084 ± 0.001% / 0.085 ± 0.001%	0.042 ± 0.109% / 0.035 ± 0.075%	1100668 / 561942
VAE, 3 / 2	0.082 ± 0.001% / 0.082 ± 0.001%	0.086 ± 0.002% / 0.087 ± 0.002%	0.063 ± 0.551% / 0.063 ± 0.541%	1101196 / 562320
AE, 4 / 2	0.084 ± 0.004% / 0.084 ± 0.005%	0.089 ± 0.003% / 0.090 ± 0.003%	0.041 ± 0.161% / 0.033 ± 0.313%	1546072 / 790744
VAE, 4 / 2	0.082 ± 0.001% / 0.082 ± 0.001%	0.089 ± 0.003% / 0.089 ± 0.002%	0.077 ± 0.182% / 0.078 ± 0.007%	1546470 / 791030
AE, 5 / 2	0.088 ± 0.002% / 0.089 ± 0.002%	0.089 ± 0.002% / 0.089 ± 0.002%	0.044 ± 0.434% / 0.040 ± 0.659%	1979636 / 1012628
VAE, 5 / 2	0.089 ± 0.002% / 0.089 ± 0.002%	0.090 ± 0.002% / 0.090 ± 0.002%	0.077 ± 0.187% / 0.078 ± 0.010%	1979956 / 1012858
AE, 6 / 2	0.090 ± 0.001% / 0.090 ± 0.001%	0.092 ± 0.003% / 0.092 ± 0.003%	0.042 ± 0.140% / 0.037 ± 0.282%	2404656 / 1227672
VAE, 6 / 2	0.092 ± 0.002% / 0.092 ± 0.002%	0.093 ± 0.002% / 0.094 ± 0.002%	0.077 ± 0.185% / 0.078 ± 0.008%	2404924 / 1227864
AE, 7 / 2	0.090 ± 0.003% / 0.090 ± 0.003%	0.092 ± 0.005% / 0.093 ± 0.005%	0.046 ± 0.618% / 0.041 ± 0.845%	2827940 / 1446532
VAE, 7 / 2	0.092 ± 0.002% / 0.092 ± 0.002%	0.092 ± 0.003% / 0.093 ± 0.003%	0.077 ± 0.176% / 0.078 ± 0.013%	2828170 / 1446698
AE, 8 / 2	0.094 ± 0.005% / 0.094 ± 0.005%	0.094 ± 0.006% / 0.094 ± 0.006%	0.054 ± 0.781% / 0.052 ± 1.019%	3246992 / 1660736
VAE, 8 / 2	0.095 ± 0.005% / 0.095 ± 0.005%	0.093 ± 0.002% / 0.093 ± 0.002%	0.077 ± 0.200% / 0.078 ± 0.000%	3247194 / 1660882

Table 8: AEs and VAEs with L1 Loss in validation and testing table for MNIST, FashionMNIST, and FreyFace datasets.

C.1.3 Additional result for Latent space visualization and Generation image

Autoencoders: Latent space visualization

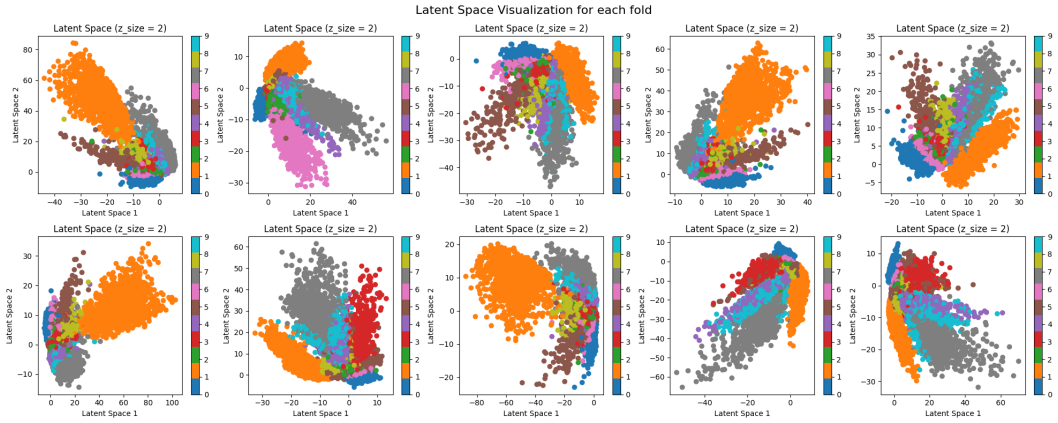

Figure 71: AE Latent Space Visualization from MNIST dataset using a 2-layer with Dz=2, , showing the latent representations for all 10 digits across all 10 folds.

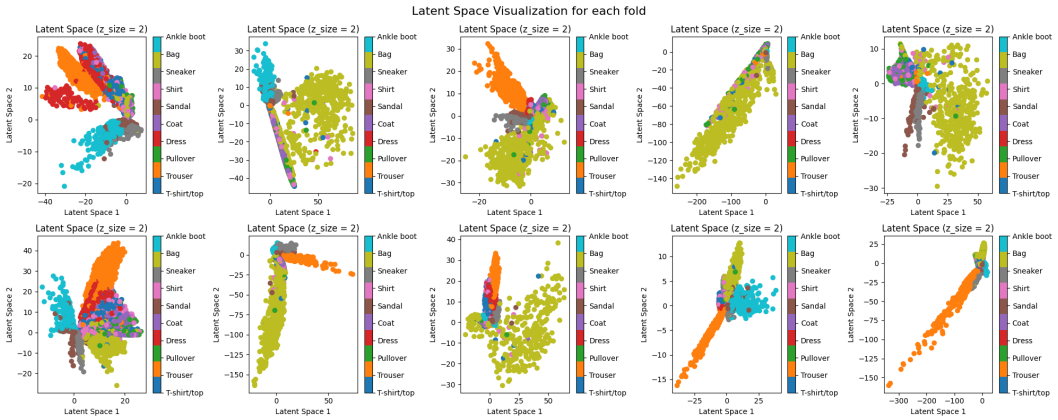

Figure 72: Visualize the latent space of an AE with a 4-layer architecture and Dz=2 using the FashionMNIST dataset, showing the latent representations for all 10 classes across all 10 folds.

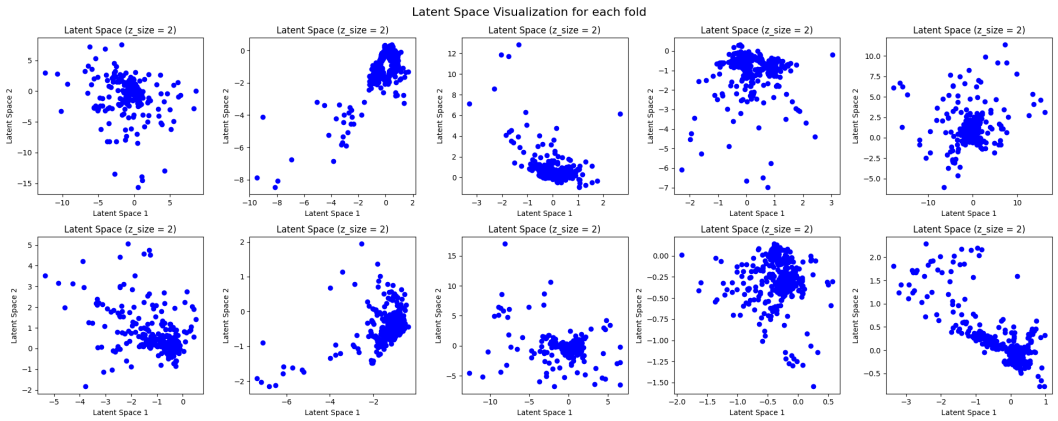

Figure 73: AE latent Space Visualization from FeyFace dataset using a 5-layer with Dz=2.

Variational Autoencoders: Latent space visualization

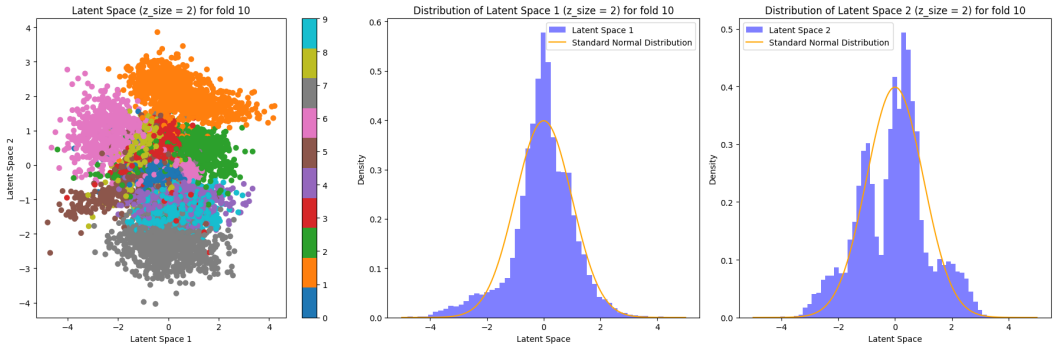

Figure 74: VAE with a 2-layer architecture and Dz=2 using the MNIST dataset, showing the latent space visualization of the best fold.

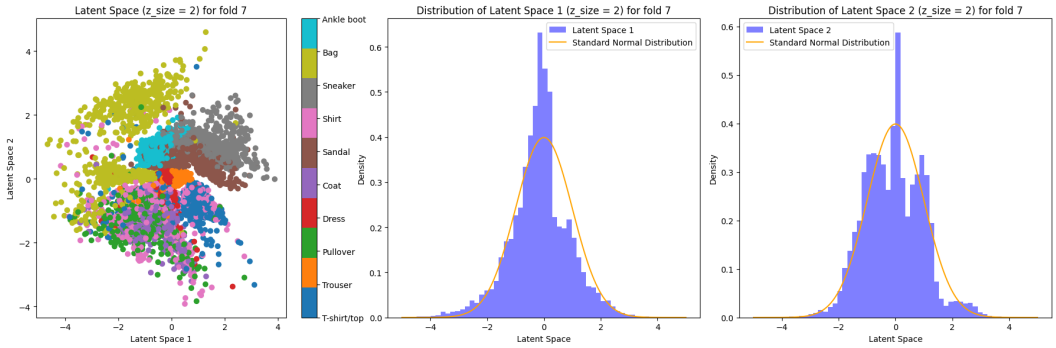

Figure 75: VAE with a 4-layer architecture and Dz=2 using the FashionMNIST dataset, showing the latent space visualization of the best fold.

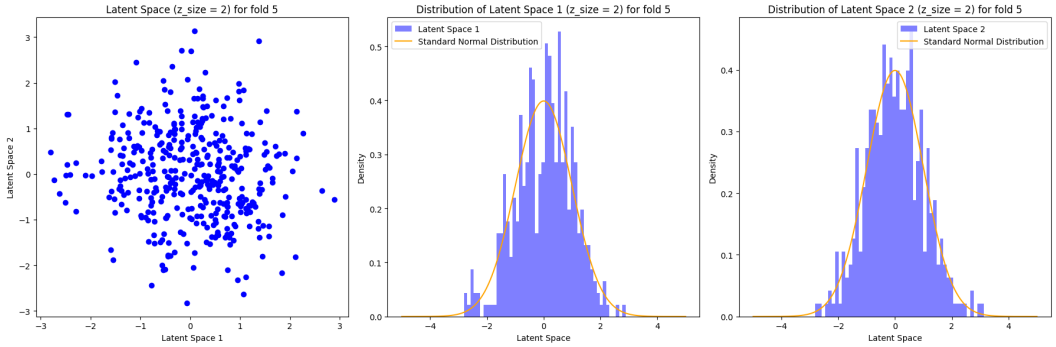

Figure 76: VAE with a 5-layer architecture and Dz=2 using the FeyFace dataset, showing the visualization of the best-fold latent space.

Autoencoders: Generation images

Figure 80: AE image generation with Dz=2, using 2 layers for MNIST in Figure 77, 3 layers for FashionMNIST in Figure 78, and 5 layers for FreyFace in Figure 79 (For FreyFace, we generate only one image).

Variational Autoencoders: Generation images

Figure 84: VAE image generation with Dz=2, using 2 layers for MNIST in Figure 81, 3 layers for FashionMNIST in Figure 82, and 5 layers for FreyFace in Figure 83 (For FreyFace, we generate only one image).

C.1.4 Study the impact of β on VAE loss: Reconstructions images and Latent space visualization

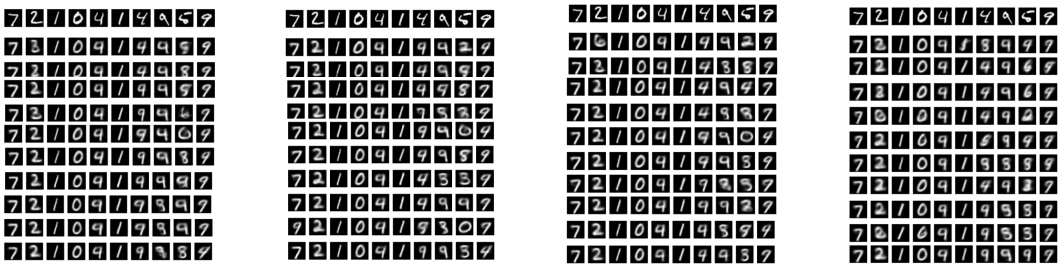

Figure 89: Reconstruction of digits using 2 layers with $\beta = 0.02$ in Figure 85, $\beta = 1$ in Figure 86, $\beta = 1.5$ in Figure 87, and $\beta = 10$ in Figure 88.