Exploring the Architecture and Latent Structure of Variational Autoencoders: From Theory to Practical Applications

Bryce Jiang

May 20, 2024

Abstract

This project investigates the architecture and latent structure of Variational Autoencoders (VAEs). It begins with an in-depth exploration of the mathematical foundations underlying VAEs, providing a robust theoretical framework for understanding their functionality. Following a reproduction of the results from the original VAE paper, an experiment was conducted on the MNIST dataset to determine a minimal complexity architecture for the encoder and decoder. The study then extends to the CelebA dataset, where a VAE was implemented to achieve attribute manipulation in human face images. The analysis focuses on identifying which attributes most significantly contribute to perceived attractiveness. The primary goal of this project is to thoroughly explore and elucidate the architecture and latent structure of VAEs, contributing valuable insights into their design and application.

1 Introduction

Variational Autoencoders (VAEs) represent a powerful class of generative models that have gained prominence in the field of machine learning. Combining the strengths of probabilistic modeling and neural networks, VAEs offer a probabilistic approach to latent space modeling, enabling the generation of diverse and meaningful samples.

Before the emergence of VAEs, traditional autoencoders were the primary framework for unsupervised representation learning. Autoencoders are neural network architectures designed to learn efficient representations of input data by encoding it into a lower-dimensional latent space and subsequently decoding it back to resemble the original input. By reducing dimensionality of the input, autoencoders offer a convenient way of data compression and reconstruction (Vincent et al. [2008]). However, one limitation of traditional autoencoders is their deterministic nature—given the same input, they always produce the same representation, which may not capture the inherent uncertainty in real-world data. The deterministic mapping also causes empty regions in the latent space not correspond to any data lack inherent meaning, making autoencoders unable to generate new data from the latent space.

Introduced by Kingma and Welling [2013], VAEs emerged as an extension to traditional autoencoders by introducing a probabilistic framework, modeling the continuous latent space as a probability distribution. This not only allows VAEs to generate diverse and realistic samples but also provides a principled way to regularize the latent space, making them more effective in capturing complex data distributions and learning meaningful representations. The elegant framework makes VAEs versatile tools for tasks ranging from data denoising to image generation and representation learning.

In the rest of this paper, I will delve into the mathematical foundations underlying VAEs in Section 2, providing a detailed explanation of the key concepts for optimizing VAEs. Section 3 will then focus on building a VAE model, including the dataset used, the architectural specifications, and the evaluation metrics employed to assess its performance. In Section 4, I will report an experiment to ascertain a minimal complexity model for the MNIST dataset, in which parameters such as hidden layer sizes were varied to identify a model architecture that achieves optimal performance in terms of both computational efficiency and effective reconstruction. In Section 5, the project progresses to applying a VAE to the CelebA dataset, aiming to manipulate facial attributes in images by maneuvering the

latent space. The analysis then aims to discern which specific attributes correlates the most to the perception of attractiveness.

2 Mathematical Foundations

The foundational assumption at the core of the VAE lies in the belief that high-dimensional data X can be effectively generated from a lower-dimensional latent representation z. If we posit that z is a continuous random variable following a certain distribution $p_\theta(z)$ with underlying parameters θ, the generation process can be framed as X coming from the conditional distribution $p_\theta(X|z)$. In real-world applications, however, it is more common to invert this process—to infer the latent representation z given observed data X, denoted as $p_\theta(z|X)$. This inversion task forms the crux of the VAE's objective, where the model seeks to learn an efficient mapping from the observed data space to the latent space. Following Bayesian principles, the inference of latent space can be represented by the equations below.

$$p_\theta(z|X) = p_\theta(X|z)p_\theta(z)/p_\theta(X)$$
$$p_\theta(X) = \int p_\theta(z)p_\theta(X|z)dz$$

Although we can give an expression of the posterior distribution $p_\theta(z|X)$, it is intractable because we do not know the distributions $p_\theta(z)$ and $p_\theta(X|z)$. To make the inference of z possible, we propose another distribution $q_\phi(z|X)$ to approximate $p_\theta(z|X)$.

$$p_\theta(z|X) \approx q_\phi(z|X)$$

The approximation can be accomplished by minimizing the Kullback-Leibler (KL) divergence between the true posterior distribution and the proposed distribution. To achieve this optimization task, we naturally turn to the ELBO loss function and the AEVB algorithm.

2.1 The ELBO loss and AEVB algorithm

In the original VAE paper, Kingma and Welling [2013] used the method of minimizing KL divergence to propose the Evidence Lower Bound (ELBO), which is then employed in the Auto-Encoding Variational Bayes (AEVB) algorithm as the loss function to enhance the efficiency of both inference and learning processes. The authors started by formulating the KL divergence between the posterior distribution $p_\theta(z|X)$ and the proposed approximation $q_\phi(z|X)$. Through mathematical derivation and transformation, they arrived at an equation wherein the summation of the KL divergence and a variational lower bound remains constant, as shown below.

$$\begin{aligned} log(p_\theta(x)) &= D_{KL}(q_\phi(z|x)||p_\theta(z|x)) + \sum_z q_\phi(z|x)log(\frac{p_\theta(x,z)}{q_\phi(z|x)}) \\ &= D_{KL}(q_\phi(z|x)||p_\theta(z|x)) + L(\theta,\phi;x) \end{aligned}$$

The task of minimizing the KL divergence between $p_\theta(z|x)$ and $q_\phi(z|x)$ is then equivalent to maximizing the variational lower bound $L(\theta,\phi;x)$, which is called the Evidence Lower Bound. To make the ELBO more meaningful, the authors rewrote $L(\theta,\phi;x)$ as a sum of the expected log-likelihood of the data under the generative model, called the reconstruction loss, and the KL divergence between the approximate posterior and the prior distribution in the latent space, which can be viewed as a regularization loss.

$$L(\theta,\phi;x) = \sum_z q_\phi(z|x)log(\frac{p_\theta(x,z)}{q_\phi(z|x)}) = \sum_z q_\phi(z|x)log(\frac{p_\theta(x|z)p_\theta(z)}{q_\phi(z|x)})$$

$$= \sum_z q_\phi(z|x) \Big[log(p_\theta(x|z)) + log(\frac{p_\theta(z)}{q_\phi(z|x)}) \Big]$$
$$= E_{q_\phi(z|x)}[log(p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p_\theta(z))$$

The reconstruction term is often approximated using the MSE between the input data and the reconstructed data, capturing the fidelity of the generative model in reproducing the observed data. Simultaneously, the KL divergence term can be simplified in the case of Gaussian distributions, aligning with the assumptions commonly made in VAEs.

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

After expressing the two terms in the ELBO using MSE and Gaussian KL divergence, the comprehensive loss function for optimizing VAEs emerges.

$$L = MSE(x, reconstructed\ x) - \frac{1}{2} \sum_{j=1}^{J} (1 + log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$$

In the loss function, J represents the total number of latent distributions, which are assumed to be independent Gaussian, and μ_j and σ_j^2 represents the mean and variance of one latent distribution. Minimizing this loss function is equivalent to maximizing the ELBO, thereby minimizing the KL divergence between the true posterior distribution and the proposed approximation.

Utilizing the ELBO loss function, the authors then came up with the Auto-Encoding Variational Bayes (AEVB) algorithm that involves key steps for training VAEs. It starts with sampling from input data and mapping it to the latent space using the encoder neural network. Latent variables are then sampled, and the decoder network reconstructs the data. The algorithm computes the reconstruction loss and the KL divergence, forming the ELBO loss. This ELBO loss is then optimized using techniques like Stochastic Gradient Descent (SGD), iteratively updating model parameters in both the encoder and decoder networks to maximize the ELBO and enhance the VAE's generative capabilities.

2.2 The reparameterization trick

In VAEs, the latent space is typically modeled as a probability distribution, allowing for the generation of diverse and meaningful samples. However, when training VAEs using the AEVB, a problem arises during backpropagation when gradients need to flow through the sampling operation. Directly sampling from a distribution makes it challenging to compute gradients because the operation is non-differentiable.

In the original VAE paper, Kingma and Welling [2013] ingeniously proposed the reparameterization trick to circumvent this challenge by decoupling the stochasticity from the network's parameters. Instead of directly sampling from the latent distributions, it draws random samples independent of the trained parameters and then expresses the latent representations as a deterministic function of the samples and the parameters. This transformation allows the gradients to flow smoothly through the stochastic nodes, enabling effective backpropagation and optimization of the VAE.

As shown in Figure 1, originally, we cannot compute the gradient of the model parameters ϕ since z is a non-differentiable random sample. After reparameterization, the randomness is separated to the ϵ node and z comes from a deterministic function $g(\phi, x, \epsilon)$, allowing the gradient to flow through z to the parameters ϕ. In this way, the reparameterization trick not only ensures the model can be efficiently trained using standard gradient-based optimization algorithms but also preserves the probabilistic nature of the generative process.

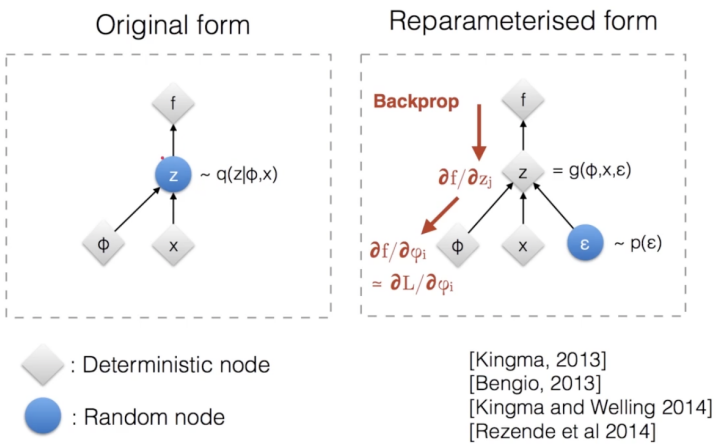


Figure 1: Demonstration of the reparameteriza1on trick.

In the context of VAEs, the trick involves reparametrizing the random sampling operation for Gaussian distribution. Let z be the latent variable sampled from $N(\mu, \sigma^2)$. Rather than sampling directly, we rewrite it as $N(0,1) \times \sigma + \mu$, where $N(0,1)$ is a standard normal distribution. The reparameterization is therefore expressed as: $z = \mu + \sigma \cdot \mu$ where $\epsilon \sim N(0,1)$. Consequently, the gradients of μ and σ can be calculated and the parameters in the encoder and decoder networks can be trained without the influence of the stochastic variable ϵ.

3 Building a VAE Model

3.1 Dataset

In this part of the study, the MNIST dataset is used for training and evaluating VAE models due to its simplicity for training and visualization as well as its widespread availability that facilitates model comparison (Doersch, 2016).

MNIST consists of grayscale images of handwritten digits (0-9), each of size 28x28 pixels. The simplicity of the dataset makes it easy to work with, understand, and experiment on, especially when initially exploring VAEs and generative models. Due to its small size, training VAEs on the MNIST is relatively quick compared to larger datasets. This allows for faster experimentation when fine-tuning models or trying out different architectures. In addition, the compact size of the images makes it easy to visualize the results of VAEs. Researchers can easily inspect the quality of generated samples, reconstructions, and the structure of the learned latent space. Since it has been extensively used in the literature, MNIST serves as a benchmark for comparing different generative models and variations of VAE architectures. Researchers often use MNIST to demonstrate the effectiveness of their proposed methods before applying them to more complex datasets.

When constructing a VAE model for MNIST, Binary Cross Entropy (BCE) loss is commonly employed as the reconstruction loss in the Evidence Lower Bound (ELBO). This choice is particularly apt for MNIST's grayscale digit images, where pixel values represent intensities in the range of 0 to 255. BCE loss is well-suited for the task of comparing pixel-wise reconstructions when each pixel is treated as a binary outcome (activated or not), effectively capturing the probability of pixel activation. This probabilistic interpretation aligns with the nature of MNIST images, making BCE loss a more effective choice than MSE, which may be sensitive to variations in pixel intensities.

3.2 Model architecture

The architecture of a VAE model consists of three main components: the encoder, the latent layer, and the decoder, as shown in Figure 2.

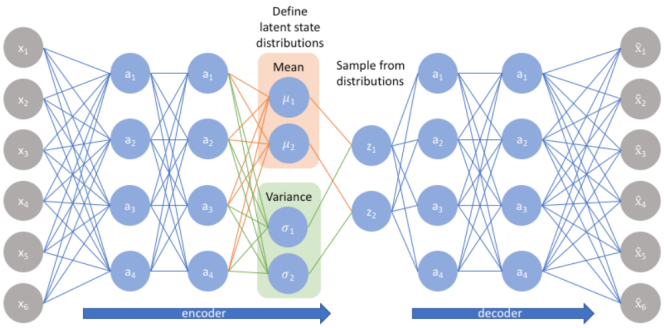


Figure 2: Architecture of VAE.

The encoder is a neural network that maps the input data into parameters that define the latent space distributions. Typically, the latent space is assumed to follow a multivariate Gaussian distribution, and the encoder outputs the parameters of this distribution, including the mean and variance vectors. Sometimes log-variance is used instead of variance in the latent layer since logarithm function transforms the variance into an unbounded range, enabling stabler optimization. The latent layer then samples from the latent distributions using the reparameterization trick, introducing a level of stochasticity crucial for generating diverse outputs during training.

The decoder, on the other hand, takes these sampled latent variables and reconstructs the input data. This process involves mapping the latent variables back to the data space, generating an output that resembles the input. The training objective for a VAE is to maximize the Evidence Lower Bound (ELBO), striking a balance between the reconstruction accuracy and the regularization imposed on the latent space through the KL divergence. This architecture enables VAEs to learn efficient representations of complex data while facilitating the generation of novel and diverse samples (Doersch, 2016).

In my first VAE model, I trained encoder and decoder with one hidden layer and 512 hidden units. The dimensionality of latent space was set to 3, 5, and 20, similar to the original VAE paper. During the training process, minibatches of size M=128 were used.

3.3 Visualizations

Following 30 epochs with 1.75×10^6 training samples, the ELBO loss for the model with a 3-dimensional latent space ($N_z = 3$) stabilized at approximately 132, which is similar to the results in Kingma and Welling's paper. The models with $N_z = 5$ and $N_z = 20$ also achieved comparable performance to the models outlined in the initial paper, as shown in Figure 3. This side-by-side comparison indicates that the models in the original VAE paper were correctly replicated, affirming the fidelity of the implemented VAE architecture and training process to the established benchmarks in the literature.

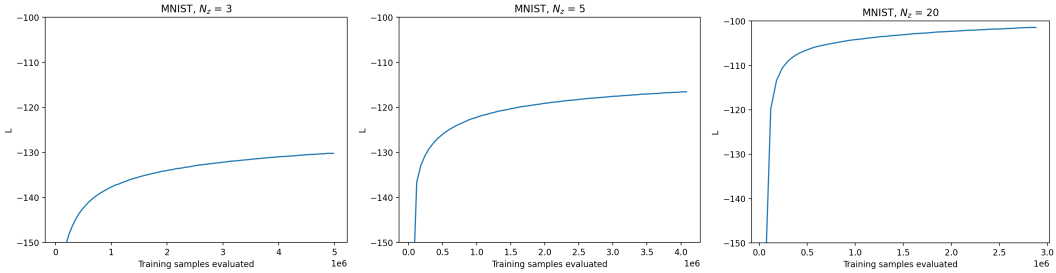
Through appropriately trained VAE models, we can visualize the latent space and produce novel samples from it. A visualization of the 2-dimensional latent space is shown in Figure 4. The plot on the right shows smooth transition between different numbers reconstructed from the latent space and indicates the VAE can perform meaningful and realistic reconstruction on empty regions in the latent space.

By randomly sampling from the latent space and reconstructing the sample using the decoder, the VAE is able to generate new hand-written digits. As presented in Figure 5, they look fairly similar to the training samples from the MNIST dataset.

3.4 Model evaluation

The VAE model is evaluated using the average MSE reconstruction loss over a new batch of data. The procedure involves inputting a new batch of 128 samples into the trained model. The VAE then

Changes in ELBO loss during my training process.



Changes in ELBO loss in the original paper.

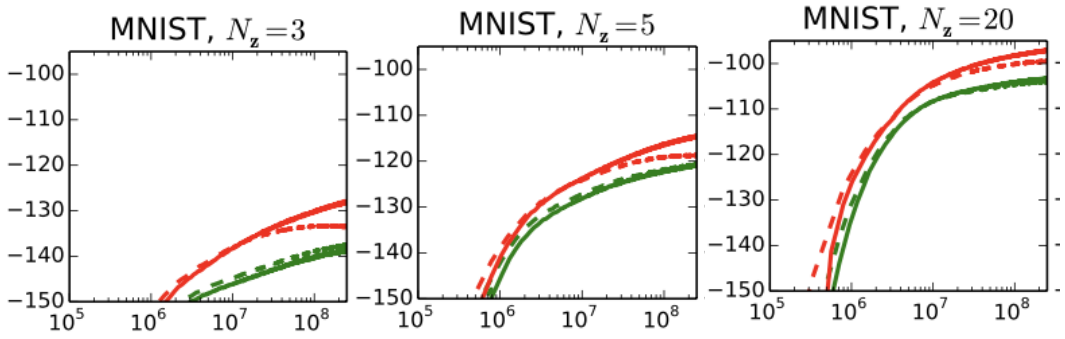


Figure 3: Side by side comparison of ELBO loss vs. # training samples between my result and Kingma/Welling's. In the second row, red curves denote the AEVB algorithm and green curves denote Wake-Sleep algorithm.

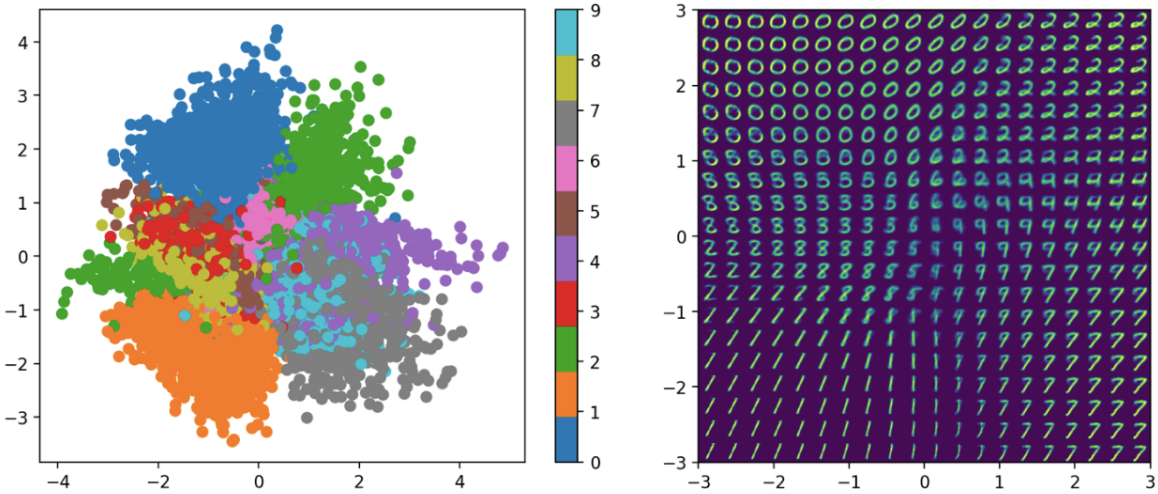


Figure 4: Visualization of the latent space. Different colors in the left plot represents different digits. The plot on the right consists of digits reconstructed form points on the latent space.

Figure 5: Random samples from the learned generative model

generates reconstructed data based on its learned latent representations. The average reconstruction loss is computed by comparing each reconstructed sample with its corresponding original input and taking average of all the MSE. This procedure provides a quantitative measure of the VAE's ability to accurately reconstruct diverse patterns within the new data batch, offering insights into the model's proficiency in capturing essential features and generating high-quality reconstructions. The VAE model achieved a test reconstruction loss similar to the training loss, indicating that the model was not overfitting and can be generalized to new data.

4 Experiment: Finding a Minimal Complexity Architecture

4.1 Experiment design

In this experiment, four trials were executed to investigate the impact of varying neural network architectures on the performance of VAE models on MNIST. In each trial, the dimensions of the latent space were controlled, and the exploration of model complexity was accompanied by the experimentation with different architectures for both encoder and decoder networks similar to the method used by Hasanpour et al., 2016. I started with a simple network featuring a single hidden layer of 32 nodes and progressively increased the model complexity. Given that encoding involves a dimension reduction process for input data of length 28*28=784 in MNIST, layers with a number of units less than 784 were exclusively considered. These layers were arranged in decreasing order based on the number of units, and a symmetrical configuration was assumed for both the encoder and decoder networks to maintain consistency across the trials. To manage training time and computational resources, hidden layers with a number of nodes that are powers of 2, greater than 32 and less than 512, were exclusively considered. This provided a range of model complexities from 5×10^4 to 1.2×10^6 parameters. An example architecture would be an encoder that has three hidden layers with 512, 128, and 32 nodes and a decoder that has hidden layers with 32, 128, and 512 nodes.

4.2 Procedure

Within each trial, the dimensions of the latent space were fixed to 3, 5, 10, and 20, respectively. Then, all 31 candidate architectures following the format described above was constructed, trained, and evaluated. Each model was trained for 20 epochs with 1.2×10^6 samples and evaluated by computing the average reconstruction loss over a batch of testing data for each model. Both the model performance, quantified by the reconstruction loss, and the corresponding number of parameters were recorded throughout the experiment. To maintain fidelity to the original VAE paper, binary-entropy loss was utilized instead of MSE loss in the process. This approach aimed to provide insights into the interplay between latent space dimensionality, network architecture, and VAE performance, aiming to find a minimal complexicity model that strikes a balance between simplicity and reconstruction

performance.

In order to ensure convergence across all models, a common stopping condition was implemented during training. Specifically, training was halted for each model if the change in the loss function fell below 0.01, thereby ensuring consistency in convergence criteria.

4.3 Results

Following the training of all candidate models, plots depicting the test reconstruction loss versus the number of parameters were generated, with each curve representing models with differing numbers of hidden layers, distinguished by varying colors. As illustrated in Figure 6, a distinct descending and converging trend is evident in the reconstruction loss as model complexity increases while maintaining the same number of hidden layers. When the latent dimension was set to 3, the architecture that yielded the best performance consisted of encoder and decoder networks with hidden layers of 512, 256, and 128 nodes, achieving a reconstruction cross-entropy loss of 115. Models equipped with 2, 3, and 4 hidden layers demonstrate comparable convergence speed and achieve similar final loss values.

The converging trend within same number of hidden layers is shared across different latent dimensions. In the case of VAEs with five latent dimensions, see Figure 7a, models with 3 hidden layers exhibit the smallest final loss value, with the optimal reconstruction loss attained by a model featuring hidden layers of widths 512, 256, and 64 nodes. Similarly, for VAEs with ten latent dimensions, see Figure 7b, models with 2 hidden layers converge to the smallest final loss value, with the best reconstruction loss achieved by a model comprising hidden layers of widths 512 and 256 nodes. Lastly, in instances where VAEs have twenty latent dimensions, see Figure 7c, models with 1 hidden layer demonstrate the most efficient convergence to the smallest final loss value, with the optimal reconstruction loss achieved by a model featuring a hidden layer width of 512 nodes.

An intriguing observation arises as the number of latent dimensions in VAEs increases: models with fewer latent layers emerge as increasingly favorable choices, exhibiting faster convergence to smaller reconstruction losses. For example, when the number of latent dimensions is 3 or 5, the best performing models have 3 hidden layers, while a model with only 1 hidden layer performs optimally when there are 20 latent dimensions. The phenomenon where models with fewer hidden layers perform better as the number of latent dimensions increases can be attributed to the balance between model capacity and latent space complexity. When the latent space is small (3 or 5 dimensions), additional hidden layers help the model capture complex patterns and variations, leading to better performance and lower reconstruction loss. Conversely, with a larger latent space (20 dimensions), the model already has ample capacity to represent the data, making a simpler architecture with fewer hidden layers sufficient to avoid overfitting and ensure faster convergence.

Additionally, an interesting trend is noted when a hidden layer of 32 nodes precedes and follows the latent layer: despite controlling the number of hidden layers and increasing model complexity, there is a slight uptick in the reconstruction loss. This phenomenon suggests that the 32-node-layer may lack the breadth to adequately encapsulate the inherent complexity within the dataset, resulting in the loss of vital information during the encoding process. These observations underscores the importance of thoroughly designing the size of hidden layers.

4.4 Limitations

The experiment has certain limitations imposed by hardware constraints, which led to considerations for managing training time. Specifically, only hidden layers with a number of units that are powers of two were taken as candidates, and a restricted set of latent dimensions were explored. The choice of these architectures was somewhat arbitrary due to these constraints. Future experiments could benefit from exploring a more diverse range of architectures to provide a more comprehensive understanding of model behavior.

Additionally, expanding the procedure to encompass different datasets, such as Frey Faces, could offer insights into the generalizability of the method for finding minimal complexity models across varied data structures. This approach would enable a comparative analysis of minimal models tailored to

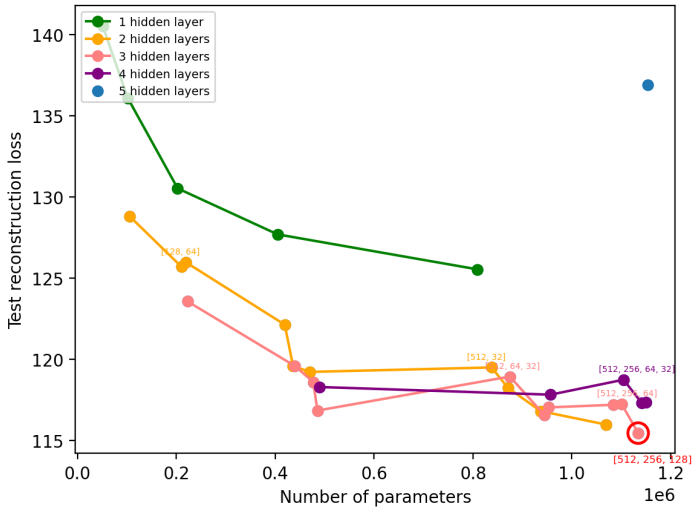
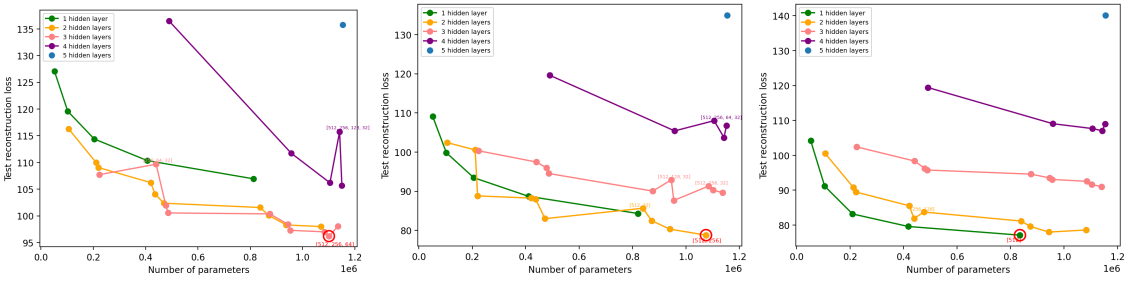


Figure 6: Number of Parameter vs. Test Reconstruction Loss plot when training set is from MNIST and latent dim = 3. Different colors stand for different numbers of hidden layers in the encoders and decoders. Numbers in brackets represents number of nodes in the encoder and decoder of each VAE. For example, the model denoted by [256, 128] has encoder and decoder with hidden layers with 256 and 128 nodes. Deeper networks with wider layers consist of larger number of parameters, leading to more complex models.



(a) Latent dimension = 5 (b) Latent dimension = 10 (c) Latent dimension = 20

Figure 7: Number of parameters vs. Test reconstruction loss plot when training set is from MNIST, latent dim = 5, 10, and 20.

different datasets, shedding light on the adaptability and robustness of the identified architectures.

Moreover, considering different evaluation metrics beyond reconstruction loss, such as sample diversity or generation quality, could offer a more comprehensive assessment of model performance. Despite these limitations, the findings from this experiment provide a valuable foundation for future research endeavors that aim to unravel the intricacies of minimal complexity models in the context of VAEs.

5 Application: Attribute Manipulation in Facial Images

Variational Autoencoders (VAEs) are known for their ability to learn meaningful latent representations. This means that the positions within the latent space of a VAE correspond to interpretable features of the input data. As a result, VAEs offer a powerful tool for generating new data and modifying existing data in a controlled manner. This characteristic enables VAEs to be combined with other models and techniques to create a meaningful latent space that reflects specific attributes of the data (Preechakul et al., 2022). In this section, we will demonstrate an application of the meaningful latent space of VAEs by manipulating attributes in facial images.

5.1 Dataset

The CelebA dataset was utilized to implement a VAE for the purpose of attribute manipulation in facial images. CelebA is a large-scale dataset containing over 200,000 celebrity images, each annotated with 40 different attribute labels. These attributes cover a wide range of facial features and characteristics, including gender, age, hair color, and the presence of accessories like glasses or hats. The dataset also includes various annotations for facial landmarks, which aid in precise attribute manipulation and image analysis.

CelebA is particularly valuable for research involving facial image generation and manipulation due to its diverse and richly labeled dataset. In the context of this project, CelebA provides an ideal platform for implementing a VAE to manipulate specific attributes in facial images. By exploring the meaningful latent representations learned by the VAE, we can identify attributes that significantly contribute to perceived attractiveness, demonstrating the practical applications of VAEs in image processing and attribute analysis.

5.2 Model Structure

The VAE model implemented in this section consists of custom down-sampling and up-sampling convolutional blocks within the encoder and decoder components.

The encoder extracts features from input images and projects them into the 200-dimensional latent space. It comprises four convolutional layers: two layers with 32 filters and two layers with 64 filters, each using a kernel size of 3x3 and a stride of 2. The output is then flattened, and two dense layers compute the mean and log variance of the latent variables. The latent variables are sampled using a Gaussian sampling layer.

The decoder reconstructs images from the latent space representation. It starts with a dense layer that projects the latent variables into a high-dimensional space. This is followed by four up-sampling convolutional blocks, each designed to progressively increase the spatial dimensions. A final convolutional layer with a sigmoid activation function produces the output image.

After training this model, we can reconstruct images, as shown in Figure 8. Additionally, by sampling random values from a Gaussian distribution in the latent space, we can generate new facial images, as illustrated in Figure 9.

Figure 8: Reconstructed faces.

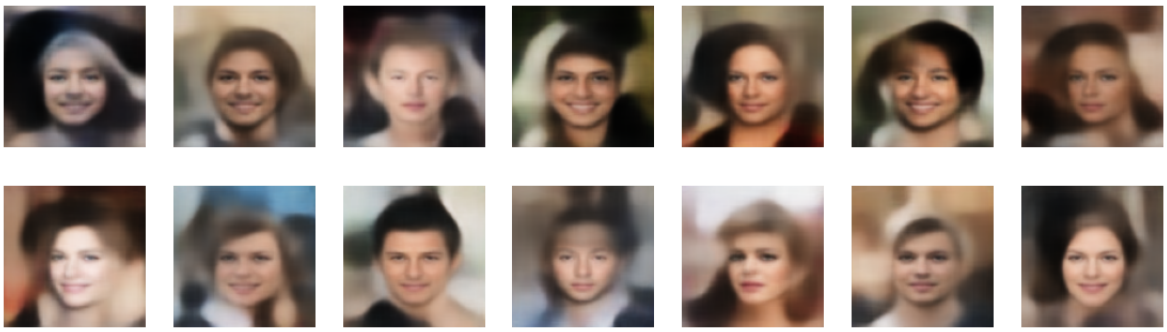


Figure 9: Decoded random samples from the latent space.

5.3 Finding Direction Vectors for Specific Attributes

The CelebA dataset includes facial attribute annotations for each image, with binary labels indicating the presence or absence of specific attributes. To identify direction vectors for specific attributes using these annotations and the encoded latent variables, the following procedure is employed:

1. **Generate Latent Vectors**: Samples from the test or training dataset are used, and their corresponding latent vectors are generated using the VAE encoder.

2. **Group Latent Vectors**: The latent vectors are divided into two groups based on the presence (positive vectors) or absence (negative vectors) of the attribute of interest.

3. **Compute Averages**: The average latent vector is calculated for each group (positive and negative).

4. **Determine Attribute Direction Vector**: The direction vector for the attribute is obtained by subtracting the average negative vector from the average positive vector.

This approach allows us to determine the specific direction in the latent space that corresponds to the presence of a particular attribute, enabling controlled manipulation of facial attributes in generated images. After obtaining the direction vector for an attribute, we can enhance the presence of that attribute in an image by adding the direction vector to its latent representation. Conversely, subtracting the direction vector from the latent representation reduces the presence of the attribute in the image. Figure 10 shows how an image move along the axis of the attribute "Attractive".

In this way, attribute manipulation can be achieved by adding weighted attribute vectors to the latent

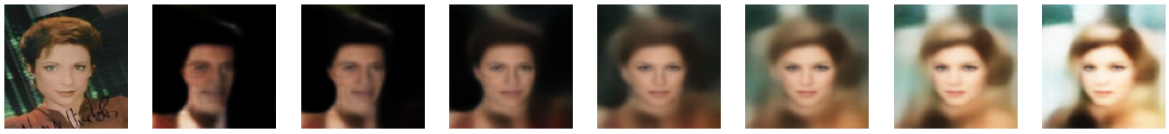


Figure 10: We can make a face more or less attractive by adding or subtracting the direction vector.

Figure 11: Customizing the generated image by moving along the vector for each attribute.

variables, effectively altering the presence of each attribute in the latent representation. As shown in Figure 11, we can create images with customized attributes by manually adjusting how far we move along the direction vectors.

5.4 Quantifying Attribute Contributions to Attractiveness

In this section, we introduce a method to quantify the contribution of various facial attributes to perceived attractiveness using the latent space representations learned by the VAE. The approach involves calculating the degree to which each attribute aligns with the direction vector of attractiveness.

First, the attractiveness direction vector is obtained from the latent space. For each attribute we compute the alignment between its direction vector and the attractiveness direction vector by calculating the dot product of the two vectors. The dot product provides a quantitative measure of how strongly each attribute correlates with attractiveness in the latent space. Once all contribution scores are calculated, they are sorted in descending order to identify the attributes that have the most significant positive or negative impact on attractiveness.

The result is fairly intuitive. The attribute "Young" has the highest positive impact, with a contribution score of 10.556, indicating that youthful features significantly enhance attractiveness. "Blond Hair" and "Pale Skin" also contribute positively, with scores of 2.479 and 1.858, respectively. On the other hand, "Smiling" has a slight negative impact (-1.408), while "Male" (-3.765) and "Eyeglasses" (-6.977) have more substantial negative contributions. These findings highlight the attributes that most strongly influence attractiveness, with youthfulness and blond hair enhancing it and male features and eyeglasses reducing it.

It is worth noting that these results are specific to the CelebA dataset, which may be subject to severe selection bias. For instance, the negative impact of the "Male" attribute on attractiveness might be because the dataset contains more attractive female celebrities than male ones. Similarly, the negative contribution of eyeglasses could be due to attractive celebrities often removing their glasses when taking photos. These factors highlight the importance of considering dataset-specific biases when interpreting the results.

6 Conclusion

In this project, I embarked on a comprehensive exploration of Variational Autoencoders (VAEs), aiming to elucidate their purpose, evolution from traditional autoencoders, and the underlying mathematical foundations. The exposition covered key concepts such as the Evidence Lower Bound (ELBO) loss function, the Auto-Encoding Variational Bayes (AEVB) algorithm, and the reparameterization trick, providing a solid theoretical background. Following this, the structure of a VAE model was introduced, and a practical demonstration involved building, training, and visualizing the latent space with generated samples.

The paper proceeds with an experiment aimed at identifying the minimal complexity architecture for MNIST, revealing that for VAEs employing three latent dimensions, encoders and decoders featuring 2, 3, and 4 latent layers exhibit comparable performance when matched for complexity, as gauged by the number of parameters. In the case of VAEs with five latent dimensions, models equipped with 3 hidden layers demonstrate superior performance. Meanwhile, for VAEs with ten latent dimensions, those equipped with 2 hidden layers showcase optimal performance. Finally, in instances where VAEs have 20 latent dimensions, models with only 1 hidden layer emerge as the top performers. Across various latent dimensions, the VAE with two hidden layers of 512 and 256 nodes achieve competitive performance while maintaining computational efficiency.

The investigation extended to the CelebA dataset, showcasing the practical application of VAEs in attribute manipulation for facial images. By leveraging the meaningful latent representations learned by the VAE, we identified direction vectors for specific attributes, such as youthfulness and blond hair, which positively impact attractiveness, and attributes like male and eyeglasses, which negatively impact attractiveness. This analysis highlights the interpretability of the VAE's latent space and its utility in understanding and modifying facial features.

To conclude, this project demonstrates the versatility and power of VAEs in both theoretical and practical contexts, providing insights into their architecture, latent structure, and application in real-world datasets.

References

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

Seyyed Hossein Hasanpour, Mohammad Rouhani, Mohsen Fayyaz, and Mohammad Sabokrou. Let's keep it simple, using simple architectures to outperform deeper and more complex architectures. *arXiv preprint arXiv:1608.06037*, 2016.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Konpat Preechakul, Nattanat Chatthee, Suttisak Wizadwongsa, and Supasorn Suwajanakorn. Diffusion autoencoders: Toward a meaningful and decodable representation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1096–1103, New York, NY, USA, 2008. ACM, ACM. doi: 10.1145/1390156.1390294. URL http://doi.acm.org/10.1145/1390156.1390294.