Recreating "Auto Encoding Variational Bayes"

Patrick Astorga
pastorga3@gatech.edu

Gregor Avgustin
gavgustin3@gatech.edu

Trinayaan Hariharan
thariharan6@gatech.edu

Lauren Nolton
lnolton6@gatech.edu

Bodan Pittman
bpittman30@gatech.edu

Abstract

Can VAE models reconstruct high dimensional datasets, such as facial images, with minimal degradation? If so, what number of latent space dimensions provide the strongest level of image generation? In the following paper, our group builds upon the findings of 'Auto-Encoding Variational Bayes' (Kingma and Welling [2022]) by seeking to find optimal levels of latent dimensional layers in image creation. Through experimentation with MNIST and CelebA datasets, we find that VAE models can reconstruct high dimensional datasets with minimal degradation along with the idea that the optimal size of latent dimensions varies based upon both the complexity of image.

1 Introduction

In recent years, unsupervised learning has become more prevalent within the field of machine learning, as its applications have multiplied to an enormous scale. Certain models like the Variational Autoencoder (VAE) are extremely important to this domain, as it allows for a scalable training of latent variable models using neural networks. As VAEs combine both an encoder and a decoder, they play a pivotal role both in the unsupervised learning fields of generative modeling and representation learning. Unfortunately, VAEs face a huge challenge when it comes to balancing efficient inference with more expressive and detailed latent representations. (Insert example here).

The goal of this paper is to reproduce the paper 'Auto-encoding variational Bayes' (Kingma and Welling [2022]), which introduced a framework that uses variational inference and backpropagation to train probabilistic generative models. Our goal is to adapt the methods that were presented in the original paper and to investigate how the VAE will perform across multiple datasets, and experiment with how the latent space captures individual generative features. Therefore, this will allow us to understand our problem statement: trade-offs between efficient inference and expressive latent representations in autoencoder-based probabilistic models.

We are motivated to investigate the findings in this paper as it builds upon the latent variable model that we discussed in class, in a scalable deep learning approach that is very relevant today. In class, we touched on the limitations of the Expectation-Maximization algorithm, such as the intractable posterior. Furthermore, we research other latent variable models prior to this paper, such as Monte Carlo Expectation-Maximization (Ruth [2024]), Monte Carlo Markov-Chain (Speagle [2020]), and Wake-Sleep (Hinton et al. [1995]), which suffer from high computational cost and/or instability. We believe that by leveraging certain insights, such as how the Evidence Lower Bound, reparameterization trick, and KL divergence interact, a scalable and stable algorithm for learning latent variable models can be established. These insights remain conceptually challenging and underexplored in many implementations. Through reproduction, we aim to not only validate the results, but also gain deeper insight into how model architecture and latent space dimensionality affect overall performance, and how generative features are learnt.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

2 Background

2.1 Latent variable model

As stated in the introduction, the findings in 'Auto-encoding variational Bayes' (Kingma and Welling [2022]) pertain to latent variable models. Latent variable models are a type of generative model, which aims to model $p(x)$: the distribution of X, the observed data. We assume that the generative features of X are captured by a set of unobserved 'latent' variables Z, such that if z is known, then $p(x \mid z)$ is relatively deterministic. In this paper, we model $p(x \mid z)$ with a neural network. Finally, to sample from $p(x)$, we can sample from $p(z)$ and then sample from $p(x \mid z)$.

The challenge of unsupervised learning of latent variable models is rooted in this: we are tasked with simultaneously learning the latent variables Z and the likelihood $p(x \mid z)$. For even the simplest likelihood functions, an exact solution is infeasible. Common iterative solutions such as Expectation-Maximization (EM) have proven effective, however they require computing the posterior distribution on the observed data

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{\int_z p(x \mid z)p(z)\mathrm{d}z}$$

In most cases, computing the posterior distribution is intractable due to the difficulty of integrating over the latent space in the denominator of the expression above. There are few prior solutions to the intractable posterior problem. The Monte Carlo Expectation-Maximization (Ruth [2024]) algorithm uses sampling to estimate expectations over the latent space, but is difficult to scale in high dimensions. The Markov Chain Monte Carlo method (Speagle [2020]) alters the previous approach offering accuracy at the cost of computational efficiency, requiring a large number of samples to be able to approximate the posterior well. The Wake-Sleep algorithm (Hinton et al. [1995]) uses separate networks to approximate the generative and inference processes however is prone to instability because of the lack of alignment between the two networks.

2.2 Variational bound

Instead of computing the posterior $p(z \mid x)$ directly, we approximate it by $q(z \mid x)$, modeled by another neural network. To optimize the $q(z \mid x)$ (encoder) and $p(x \mid z)$ (decoder) networks simultaneously, a new objective function is introduced. The log likelihood of a data sample $x^{(i)}$ is decomposed into two terms

$$\log p(x^{(i)}) = \mathbb{E}_{z\sim q(z|x^{(i)})}\left[\log \frac{p(x^{(i)}, z)}{q(z \mid x^{(i)})}\right] + KL\left(q(z \mid x^{(i)}) \mid\mid p(z \mid x^{(i)})\right)$$

The first term is known as the Evidence Lower Bound (ELBO), and is the objective function we are to maximize. Note that by the equation above, we see that maximizing the ELBO w.r.t. the q network is equivalent to minimizing the KL-divergence between our approximate posterior $q(z \mid x)$ and the true posterior $p(z \mid x)$. We will demonstrate through our experiments that maximizing this objective yields both an meaningful encoder and accurate decoder networks. For explicitness, let θ and ϕ be the parameters of the p and q networks, respectively:

$$\mathcal{L}\left(x^{(i)}; \theta, \phi\right) = \mathbb{E}_{z\sim q(z|x^{(i)})}\left[\log \frac{p(x^{(i)}, z)}{q(z \mid x^{(i)})}\right]$$

It is both insightful and will be helpful later to decompose the ELBO into two terms

$$\mathcal{L}\left(x^{(i)}; \theta, \phi\right) = \mathbb{E}_{z\sim q(z|x^{(i)})}\left[\log p(x^{(i)} \mid z)\right] - KL\left(q(z \mid x^{(i)}) \mid\mid p(z)\right)$$

Given samples from the output of the encoder $q(z \mid x^{(i)})$, the first term measures how accurately the decoder reconstructs the original data sample $x^{(i)}$. Therefore it is commonly known as the 'reconstruction accuracy.' The second term penalizes the encoder for outputting a distribution too different from the prior $p(z)$. This prior is chosen before training. You can see explicitly how maximizing the ELBO balances reconstruction fidelity with a term which regularizes the posterior, encouraging meaningful latent representations. The Auto-Encoding Variational Bayes (AEVB) framework combines the encoder/decoder model with the ELBO objective for efficient learning (Kingma and Welling [2022]).

3 Methods

Our goal is to apply the Auto-Encoding Variational Bayes framework to two image datasets, using neural networks for the encoder/decoder.

3.1 Adaptation of variational bound

The main contribution of Auto-Encoding Variational Bayes (Kingma and Welling [2022]) is the application of the Evidence Lower Bound as a single differentiable objective function for training model via backpropagation. In this section, suppose that the images X are N-dimensional real numbers in the range $[0, 1]$, representing pixel activations. Furthermore, suppose that the latent variables Z are K-dimensional continuous, real numbers. Assume that the prior $p(z) \sim \mathcal{N}(0, I)$.

The decoder takes an N-dimensional image $x^{(i)} \in X$ and outputs two K-dimensional vectors: $q_\mu(x^{(i)})$, $q_\sigma(x^{(i)})$, which are the mean and variance of the posterior distribution

$$q(z \mid x^{(i)}) \sim \mathcal{N}\left(q_\mu(x^{(i)}), q_\sigma(x^{(i)})\right)$$

The encoder takes a K-dimensional latent code $z \in Z$, and outputs a N-dimensional vector $p_{\hat{x}}(z)$ such that the ith entry in $p_{\hat{x}}(z)$ corresponds to the probability that the ith pixel is set to 1 (we assume each pixel is independently distributed)

$$p(x_i = 1|z) = p_{\hat{x}}(z)_i$$

In this section we will express the ELBO in terms of the outputs of the p and q networks in a way that is differentiable w.r.t. θ and ϕ, the parameters of the networks.

3.1.1 Reconstruction accuracy

The reconstruction accuracy as presented in section 2.2 is as follows

$$\mathbb{E}_{z\sim q(z|x^{(i)})}\left[\log p(x^{(i)} \mid z)\right]$$

The log likelihood $\log p(x^{(i)} \mid z)$ can be broken down per pixel and expressed in terms of the output of the decoder network

$$\log p(x^{(i)} \mid z) = \sum_{j=1}^{N} \log p(x_j = x_j^{(i)} \mid z) = \sum_{j=1}^{N} \left[x_j^{(i)} \log(p_{\hat{x}}(z)_j) + (1 - x_j^{(i)}) \log(1 - p_{\hat{x}}(z)_j)\right]$$

The expectation $\mathbb{E}_{z\sim q(z|x^{(i)})}$ can be estimated by taking samples. Although many samples would yield a more accurate estimate, one sample will suffice (Kingma and Welling [2022]). Given an input image $x^{(i)}$, a latent vector $z^{(i)}$ is sampled from $\mathcal{N}(q_\mu(x^{(i)}), q_\sigma(x^{(i)}))$ and then the reconstruction accuracy is estimated as

$$\mathbb{E}_{z\sim q(z|x^{(i)})}\left[\log p(x^{(i)} \mid z)\right] \approx \sum_{j=1}^{N} \left[x_j^{(i)} \log(p_{\hat{x}}(z^{(i)})_j) + (1 - x_j^{(i)}) \log(1 - p_{\hat{x}}(z^{(i)})_j)\right]$$

Reparameterization trick Our goal is to calculate the gradient of the reconstruction accuracy w.r.t. the parameters of the encoder network, but sampling from $\mathcal{N}(q_\mu(x^{(i)}), q_\sigma(x^{(i)}))$ does not yield a well defined gradient w.r.t. q. To get around this, we sample an auxiliary $\epsilon^{(i)} \sim \mathcal{N}(0, I)$ and define

$$z^{(i)} = q_\mu(x^{(i)}) + \epsilon^{(i)} \odot q_\sigma(x^{(i)})$$

where $\odot$ represents element-wise multiplication. This is the same as sampling from $\mathcal{N}(q_\mu(x^{(i)}), q_\sigma(x^{(i)}))$, but the gradient is well defined.

3.1.2 KL-divergence

The KL-divergence term as presented in section 2.2 is as follows

$$KL\left(q(z \mid x^{(i)}) \mid\mid p(z)\right)$$

In our case, we can compute the KL-divergence between two gaussian distributions analytically

$$KL\left(\mathcal{N}\left(q_\mu(x^{(i)}), q_\sigma(x^{(i)})\right) \mid\mid \mathcal{N}(0, I)\right) = -\frac{1}{2}\sum_{j=1}^{K}\left[1 + \log(q_\sigma(x^{(i)})_j)^2 - (q_\mu(x^{(i)})_j)^2 - (q_\sigma(x^{(i)})_j)^2\right]$$

3.2 Advantages

The approach outlined in this section yields a single differentiable objective function for the ELBO. In the age of highly efficient GPU matrix operations, deep neural networks are king. If the ELBO is calculated like in this section and its gradient is used to update a neural network, then a large scale model can be trained using previously existing neural network architecture with software/infrastructure specialized in efficient training of neural networks. This contrast from previous approaches which require special training algorithms, not at suitable for today's infrastructure.

4 Experiments: MNIST

The MNIST dataset is a collection of 60,000 28×28 grayscale images of handwritten digits. Within the original experiments by Kingma and Welling, they demonstrate the efficiency of the AEVB framework on the MNIST dataset in terms of the time it takes to reach a marginal likelihood limit along with achieving a higher variational lower bound. In those experiments, they noted that increasing the dimensionality did not result in overfitting in terms of the variational lower bound. Alternatively, we wanted to focus on qualitative metrics of the algorithm's ability to reconstruct images given different dimensionalities of the latent spaces.

This experiment is designed to answer the following:

1. How does changing the number of dimensions of the latent space affect the generation of images?
2. What implications does this have on the number of important factors to consider in digit recognition?

4.1 Model architecture

The images in MNIST consist of a single channel of $28 \times 28 = 764$ pixel activations. The images are flattened into a 784-dimensional vector during training. Both the encoder and decoder are fully connected multi-layer perceptrons with two hidden layers of 512 neurons each, with ReLU activations. Suppose that the dimension of the latent space is K.

The input layer of the encoder takes in the 764-dimensional flattened image vector. Both output layers of the encoder have K neurons. The μ-output layer of the encoder has linear activations, and the σ-output layer of the encoder is passed through the $\exp$ function to map it to $\mathbb{R}^+$.

The input layer of the decoder takes in the K-dimensional latent code. The output layer of the decoder has 784 neurons with the sigmoid activation function, to map them to $(0, 1)$, since the output is Bernoulli.

The code containing the model architecture can be found in `mnist_vae.py`.

4.2 Results

4.2.1 Effects of dimensionality

In changing the number of dimensions of the latent space, the image generation had an unexpected effect on the given MNIST data. The MNIST data noted that even with 1 dimension, image replication

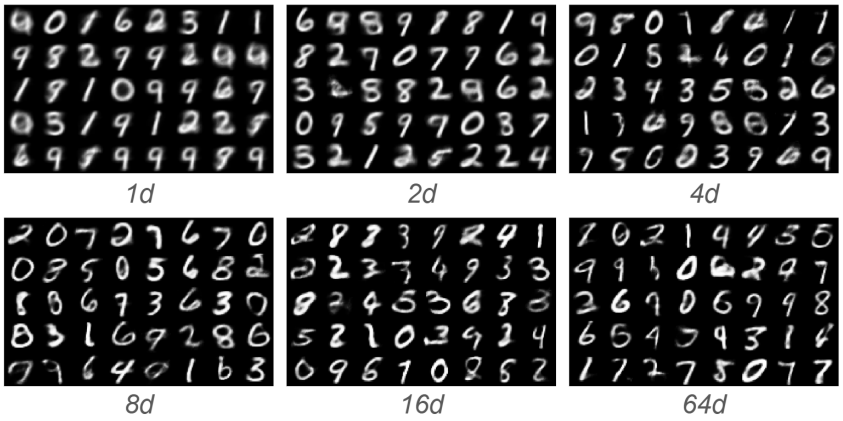

Figure 1: Effects of dimension size of latent space. From left to right, we have: 1, 2, 4, 8, 16, and 64 dimensions respective to each image.

had a somewhat strong quality and built recognizable digits. At two dimensions, the blurriness of the digits decreased and we saw improved recognizable digits. Upon further dimension additions, the improvements were relatively unnoticeable. This lack of improvement shows the simplicity of the replication and digits. This suggests that underlying factors in digit variation are relatively small as compared to something complex such as a face.

5 Experiments: CelebA

The CelebA dataset is a collection of approximately 200,000 178×218 color images of celebrity faces. The images were center cropped and rescaled to 64×48 before applying the VAE. The images in the CelebA dataset are labeled with binary annotations of important features to help distinguish them, such as expression, hair type, face-wear, face shape, etc. This dataset is significantly larger and higher dimensional than those in the paper by Kingma and Welling, since the computational resources available for larger neural networks were limited.

This experiment is designed to answer the following:

1. Can the VAE model scale to high-dimensional datasets, such as facial images, with minimal degradation?
2. How does changing the number of dimensions of the latent space affect the generation of images?
3. How well does latent space encode specific features and can we utilize this to append new features to already existing image?

5.1 Model architecture

The modified images in CelebA consist of three channels of 64×48 pixels, for a total of $9,212$ pixel activations. Unlike in the MNIST model, these images are not flattened before training.

The output layers of the encoder and decoder are the exact same as in section 4.1, with the decoder output scaled up to match the $9,212$ pixel activations. Unlike in the MNIST model, the encoder and decoder are not fully connected and instead utilize convolutional layers.

The encoder utilizes four stacked convolutional blocks. Each block consists of a convolution of kernel size 4 and stride 2, which cuts the spacial dimensions in half. This is followed by a batch normalization layer and a ReLU activation. The four convolutional blocks reduce the spacial dimensions from $64 \times 48 \to 32 \times 24 \to 16 \times 12 \to 8 \times 6 \to 4 \times 3$ while increasing the channels from $3 \to 32 \to 64 \to 128 \to 256$. Finally the activation map is flattened before the final two linear output layers.

The decoder is simply the mirror image of the encoder. Instead of convolutions the decoder uses transposed convolutions of kernel size 4 and stride 2, which effectively preforms the inverse operation, multiplying the spacial dimensions by two. This is followed by the ReLU activation function. The latent code passes through an initial linear layer and is reshaped to a 3D feature map. The four transposed convolutional blocks increase the spacial dimensions from $4 \times 3 \rightarrow 8 \times 6 \rightarrow 16 \times 12 \rightarrow 32 \times 24 \rightarrow 64 \times 48$ while decreasing the channels from $256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 3$. The output layer is the final activation map passed through the sigmoid activation function, as described in section 4.1.

The code containing the model architecture can be found in `celeba_vae_bernoulli.py`.

5.2 Results

5.2.1 Reconstruction Quality

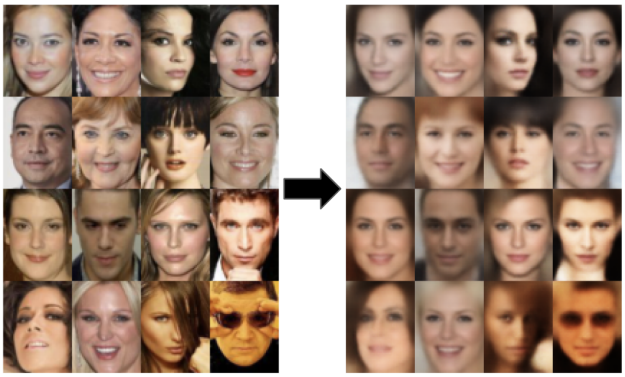

Figure 2: The reconstruction of a dataset after encoding and decoding into the learnt latent space.

Although novel generation is the main purpose of the VAE model, examining qualitatively the reconstruction accuracy can give insights on what features the model is able to effectively capture. A set of images is passed into the encoder, a latent code is sampled from their posterior distribution, and the images are regenerated from the latent codes by the decoder. As seen in **Figure 2**, qualitatively, we can see that the model maintains the key facial features of each celebrity. For example, for each celebrity, their skin tone, hair color, and face-wear were all preserved. This indicates that the model is properly learning how to encode and decode the general features of each of its inputs.

However, we can see that there are some critical flaws in the quality of these reconstructions. The largest flaw in this reconstruction is the loss of the more subtle, nuanced features of the images that is replaced with blurriness and an excessive smoothing of the images.

5.2.2 Novel Generation

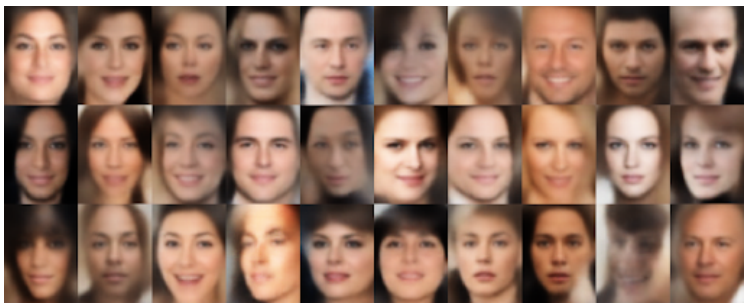

Figure 3: Randomly generated images sampled from latent space.

To evaluate the generative capabilities of our model, we sample latent vectors from our normal prior $\mathcal{N}(0, I)$ and input them into our decoder in order to generate images. As seen in **Figure 3**, the resulting generated images are diverse and show distinct facial features. While these generations are realistic, there still are concerns with the results such as the mild blurriness found in many of the images.

5.2.3 Effects of Dimensionality

Figure 4: Effects of dimension size of latent space. From left to right, we have: 2, 8, 16, 32, 64, and 96 dimensions respective to each image.

We can determine the impact of latent space dimensionality of the VAE's efficacy by training models with varying dimensions: $z \in \{2, 8, 16, 32, 64, 96\}$. As expected, due to the high-dimensional facial images, the larger dimension sized models lead to, qualitatively, better results than the smallest sized models. This trend suggests that larger latent sizes can more aptly capture nuanced details.

With an extremely low dimensional latent space, most of the generative features are lost and the decoder learns to output an extremely average face. Instead, the latent space encode only the most significant feature (in terms of raw pixel values), which is the lighting direction.

If the latent space dimension is too large, then the model experiences a sparse latent space. As seen in **Figure 4**, the model with the largest number of dimensions, the right-most image with 96 dimensions, has distinctly worse results than the previous four models. This is most likely due to the random sample of the prior distribution occurring in a "dead-spot" of the latent space: an area of the model's latent space that does not map to any encoded facial feature.

This implies that the dimensionality of the model is not generic across all models. Rather, it is dependent on nuances that are found in the input datasets the model is trained on. For this experiment, we believe that the optimal dimensionality for the CelebA dataset model is between 32 and 64 dimensions.

5.2.4 Traversal of Facial Feature Encodings

Figure 5: The traversal of a random sampled latent vector across the learned "smile" direction (left) and "male" direction (right)

In order to better understand the organization of the latent space of our model, we can analyze how latent directions affect specific facial features in generated images. If we simply modify a singular

dimension, while maintaining the others fixed, we can observe what changes in the image based on the modification.

To determine what direction in the latent space corresponds to a given feature was an interesting challenge and required an novel solution. Among a test set of 10,000 images, the latent code was generated for each by the encoder. This explains how the decoder classifies each images based on its generative features. Then, a logistic regression model was trained on the latent codes to predict whether or not a given binary annotated feature is true. If the VAE effectively captured a feature along a direction in the latent space, then the positive and negative classes should be linearly separable. Furthermore, the vector composed of the linear coefficients of the separating hyperplane should point in the direction most strongly associated with the feature. The accuracy of the logistic regressor gives a metric for how effectively the model captured the specific feature. The code for determining feature directions can be found in `celeba_feature.py`.

For example, look at **Figure 5**. The first image shows how traversing across the specific dimension shows a traversal of neutral to smiling faces. The second image shows a similar concept but from feminine to masculine featured faces. This would indicate that our model successfully captured meaningful structure in the latent space, where directions of a dimension correspond to human-interpretable changes in faces.

6 Conclusion

In this project, we reproduced and extended the work of "Auto-Encoding Variational Bayes" by Kingma and Welling [2022], with a focus on understanding the critical trade-off between efficient interference and expressive latent representations in VAEs. Our project involved replicating the original framework and exploring modifications aimed to improve its performance and interpretability. In particular, we examined how key components such as the Evidence Lower Bound (ELBO), reparameterization trick, and KL divergence interacted and contributed to the performance of VAEs. We applied our modified VAE across many datasets to test its generalizability and performance across diverse data conditions.

VAEs are compelling because of their ability to encode and decode data, enabling them to understand trends in data and generate new data that reflect those trends. They have a wide range of applications, including generating realistic images, producing coherent sentences in natural language processing (NLP), and detecting anomalies in data based on reconstruction quality. However, VAEs face notable challenges. Compared to Generative Adversarial Networks (GANs), VAEs often produce blurry or low-quality images. They also have limited posterior flexibility due to the use of the simple Gaussian distribution and face issues like posterior collapse during training when the KL terms dominated the reconstruction objective.

To address these issues, future work will focus on employing more flexible posterior models, such as normalizing flows or hierarchal structure, and improving decoder expressiveness by replacing the simple Bernoulli output with a Gaussian or mixed Gaussian based distributions. Techniques like KL annealing, which gradually increases the KL weight during training, offers potential solutions to posterior collapse and improves overall training stability.

In general, this project not only validated prior work, but also deepened our understanding of subtle trade-offs in VAE performance. We gained valuable insight into how model components interact and influence the balance between efficient interference and expressive latent representations. Understanding these trade-offs allows us to move closer to designing generative models capable of learning complex data models while remaining computationally efficient. Ultimately, this progress pushes the boundaries of what VAEs can achieve, further contributing to the advancement of the field of unsupervised learning.

References

Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022. URL https://arxiv.org/abs/1312.6114.

William Ruth. A review of monte carlo-based versions of the em algorithm, 2024. URL https://arxiv.org/abs/2401.00945.

Joshua S. Speagle. A conceptual introduction to markov chain monte carlo methods, 2020. URL https://arxiv.org/abs/1909.12313.

Geoffrey E. Hinton, Peter Dayan, Brendan J. Frey, and Radford M. Neal. The "wake–sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, May 1995. doi: 10.1126/science.7761831.

Appendix

Our implementation is publicly available at https://github.com/patrickmastorga/ML-Project

A more complete derivation of the variational lower bound (and how it is adapted for our use) can be found https://www.overleaf.com/read/snbybqzxwvxkd1565f