On Variational Autoencoders: Generative Models for Dimension Reduction

Abdul Saleh

December 14, 2019

1 Introduction

Variational Autoencoders (VAEs) are flexible, deep, generative models that have successfully been applied to modelling complex distributions. Over the past few years, VAEs have been applied to generating text, human faces, anime characters, handwritten digits, mesh models, and graphs. VAEs were first proposed as generative models [5]. This project motivates and analyzes VAEs as dimension reduction algorithms instead. VAEs can capture complex manifolds and produce flexible, interpretable representations making them an attractive option for dimension reduction despite being originally designed for generative modelling. We release the code to reproduce our experiments at https://github.com/AbdulSaleh/dimension-reduction-vae.

2 Method

2.1 Setting

Let $\mathbf{X}$ be a dataset of N observations $\mathbf{x}_1, \ldots, \mathbf{x}_N$ which are samples of some random variable $\mathbf{x} \in \mathbb{R}^D$. We assume that this dataset lies close to some manifold defined by the continuous latent variable $\mathbf{z} \in \mathbb{R}^K$. The data generating process acts in two steps:

1. First, sample a value $\mathbf{z}_i$ from the prior distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$
2. Then sample an observation $\mathbf{x}_i$ from the conditional distribution $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$

The distribution of $\mathbf{x}|\mathbf{z}$ comes from a family of distributions parametrized by $\boldsymbol{\theta}$ and the distribution of $\mathbf{z}$ is arbitrary. However, we do not get to observe much of this process. The parameters $\boldsymbol{\theta}$ and distributions are unknown, and we are not given the latent variables $\mathbf{z}_i$.

2.2 Goals

Our goal in this setting is to find efficient estimates of the parameter $\boldsymbol{\theta}$. This can be divided into three subgoals:

1. Approximate inference of the marginal distribution of $\mathbf{x}$, given by $p_{\boldsymbol{\theta}}(\mathbf{x})$. The data distribution can be used for many real-world applications such as image denoising or inpainting in computer vision.
2. Approximate inference of the posterior distribution of $\mathbf{z}|\mathbf{x}$, given by $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, which provides low-dimensional latent representations of the data and learns the manifold the data lies on.
3. Approximate inference of the generative distribution of $\mathbf{x}|\mathbf{z}$, given by $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$. Along with the latent variable distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$, this allows us to understand and simulate the data generating process (maybe we are modelling some natural process).

2.3 Inference as Optimization

We begin by considering our 1st goal of estimating $p_{\boldsymbol{\theta}}(\mathbf{x})$. We can attempt to marginalize out $\mathbf{z}$:

$$p(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\, p_{\boldsymbol{\theta}}(\mathbf{z})\, d\mathbf{z} \tag{1}$$

However, since this integral is intractable, we consider a different approach and focus on our 2nd goal of estimating $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ instead. Let $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ be some arbitrary distribution parametrized by $\boldsymbol{\phi}$ that we use to estimate $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. The optimal $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is the one that is as close as possible to $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ so we want to minimize the Kullback-Leibler divergence between the two distributions:

$$\begin{aligned}
D_{KL}\big[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,\big|\, p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\big] &= \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})} d\mathbf{z} && (2)\\
&= E_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\Big[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\Big] && (3)\\
&= E_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\Big[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\Big] && (4)
\end{aligned}$$

where equation 3 follows from the law of the unconscious statistician. Moving forward we assume the expectation is with respect to $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ so we use E instead of $E_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}$. We then use Bayes' rule to decompose the term on the right:

$$\begin{aligned}
D_{KL}\big[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,\big|\, p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\big] &= E\Big[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\Big] && (5)\\
&= E\Big[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}\Big] && (6)\\
&= E\Big[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - \log p_{\boldsymbol{\theta}}(\mathbf{z})\Big] + \log p_{\boldsymbol{\theta}}(\mathbf{x}) && (7)
\end{aligned}$$

where $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ does not depend on $\mathbf{z}$ so it can be pushed out of the expectation.
Now rearrange to get:

$$\begin{aligned}
\overbrace{\log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{KL}\big[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,\big|\, p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\big]}^{\text{ELBO}} &= E\Big[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - \big(\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{z})\big)\Big] && (8)\\
&= E\Big[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\Big] - E\Big[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) - \log p_{\boldsymbol{\theta}}(\mathbf{z})\Big] && (9)\\
&= \underbrace{E\Big[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\Big] - D_{KL}\big[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,\big|\, p_{\boldsymbol{\theta}}(\mathbf{z})\big]}_{\text{SGD objective}} && (10)
\end{aligned}$$

We have arrived at the Evidence Lower BOund (ELBO) which describes the VAE objective function:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x}) = \log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{KL}\big[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,\big|\, p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\big] \tag{11}$$

The ELBO has multiple intuitive and theoretically appealing interpretations that make it a natural objective function for VAEs.

Recall our 3rd goal, which involved estimating $p_{\boldsymbol{\theta}}(\mathbf{x})$. Now, note that the ELBO is a lower bound on the probability distribution $\log p_{\boldsymbol{\theta}}(\mathbf{x})$, since the KL divergence is non-negative:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \;\geq\; \underbrace{\log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{KL}\big[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,\big|\, p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})\big]}_{\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x})} \tag{12}$$

This relationship allows us to formulate the inference of $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ as an optimization problem. Since it is difficult to estimate $\log p_{\boldsymbol{\theta}}(\mathbf{x})$ (as we saw in equation 1), we can instead estimate a lower bound, $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x})$, on $\log p_{\boldsymbol{\theta}}(\mathbf{x})$. If we let $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ be some arbitrary distribution over $\mathbf{z}$, the ELBO is equal to the desired distribution (i.e. $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x}) = \log p_{\boldsymbol{\theta}}(\mathbf{x})$) when $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ has the same distribution as $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, which is when the KL divergence between the two distributions is 0. In other words, we hope to find a distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ which is as close as possible to $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$. This is equivalent to maximizing $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x})$, which leads to a good estimate of $\log p_{\boldsymbol{\theta}}(\mathbf{x})$.

2.4 SGD Variational Estimator

Unfortunately, it is still unclear how we would calculate the ELBO objective in equation 11 since the distributions of $p_{\boldsymbol{\theta}}(\mathbf{x})$, $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ remain unknown. To get around this issue, we rewrite the ELBO in an alternative, equivalent form (refer to equation 10):

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x}) = \underbrace{E\Big[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\Big]}_{-1\times \text{ decoder loss}} - \underbrace{D_{KL}\big[q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\,\big|\,p_{\boldsymbol{\theta}}(\mathbf{z})\big]}_{\text{encoder loss}} \tag{13}$$

We make some modeling assumptions to optimize this objective with stochastic gradient descent (SGD). We assume that $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z}\,|\,\vec{0}, \mathbf{I})$, and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}\,|\,\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\sigma}^2_{\mathbf{x}}\,\mathbf{I})$. More specifically, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ can be viewed as an encoder which maps each $\mathbf{x}$ to a mean $\boldsymbol{\mu}_{\mathbf{x}}$ and diagonal covariance matrix $\boldsymbol{\sigma}^2_{\mathbf{x}}\,\mathbf{I}$. The latent embedding $\mathbf{z}$ is sampled from the resulting distribution.

Similarly, $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ can be viewed as a decoder that uses $\mathbf{z}$ to reconstruct $\mathbf{x}$. For example, if we are interested in decoding $[0,1]$ images with D pixels we can assume $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})_j = Bernoulli(\mathbf{x}_j\,|\,p_{\mathbf{z},j})$ models the probability of the j^{th} pixel, $\mathbf{x}_j$, being black as opposed to white. Here, $p_{\mathbf{z}} \in \mathbb{R}^D$ are probability estimates returned by the decoder / neural network. Alternatively, if $\mathbf{x}$ is outside of the range $[0,1]$, we can assume $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}\,|\,\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}^2_{\mathbf{z}}\mathbf{I})$, where $\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}} \in \mathbb{R}^D$ are again the estimates returned by the decoder / neural network.

We use neural networks for the encoder and decoder, and use stochastic gradient descent (SGD) to maximize $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x})$ and learn parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$. We can estimate the decoder loss based on minibatches of size M:

$$E\Big[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\Big] \approx \frac{1}{M}\sum_{m=1}^{M} \log p_{\boldsymbol{\theta}}(\mathbf{x}_m|\mathbf{z}_m) \tag{14}$$

Returning to our images example, in the case where $\mathbf{x} \in [0,1]$, we define $\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ as the log likelihood of the data under a Bernoulli distribution:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = \sum_{j=1}^{D} \mathbf{x}_j \log p_{\mathbf{z},j} + (1 - \mathbf{x}_j)\log(1 - p_{\mathbf{z},j}) \tag{15}$$

Similarly, in the case where $\mathbf{x}$ is outside of $[0,1]$, we define $\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ to be the log likelihood under a multivariate normal distribution:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) = -\frac{D}{2}\log(2\pi) - \frac{1}{2}\log|\boldsymbol{\sigma}^2_{\mathbf{z}}\mathbf{I}| - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{z}})^T(\boldsymbol{\sigma}^2_{\mathbf{z}}\,\mathbf{I})^{-1}(\mathbf{x} - \boldsymbol{\mu}_{\mathbf{z}}) \tag{16}$$

We have described how to estimate the decoder loss. Now we describe how to estimate the encoder loss. The Gaussian assumptions we made on $p_{\boldsymbol{\theta}}(\mathbf{z}), q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ allow us to compute the KL divergance in closed form, where $\mathbf{z} \in \mathbb{R}^K$:

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

$$= \frac{1}{2}\sum_{k=1}^{K}(1 + \log\boldsymbol{\sigma}^2_{\mathbf{x},k} - \boldsymbol{\mu}^2_{\mathbf{x},k} - \boldsymbol{\sigma}^2_{\mathbf{x},k}) \tag{18}$$

We can combine equations 18 and 14 to estimate the ELBO in 13. However, we still cannot maximize the ELBO using gradient descent. One crucial issue here is that the decoder outputs (for example $p_{\mathbf{z},j}$), depend on $\mathbf{z}$ which is sampled from $\mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\sigma}^2_{\mathbf{x}}\,\mathbf{I})$. The sampling procedure is not differentiable, and thus breaks the gradients trying to flow back from the decoder to the encoder. The result is that the neural network cannot be optimized with gradient descent. The next section presents the reparameterization trick introduced by [5] to resolve this issue.

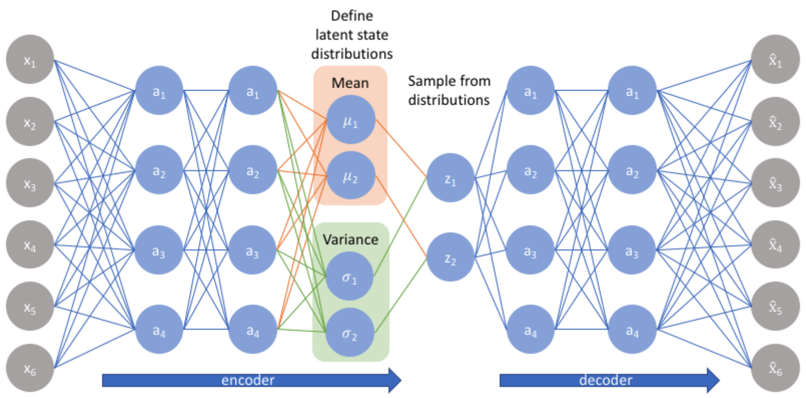


Figure 1: A variational autoencoder. The encoder, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, maps an input $\mathbf{x}$ to a distribution parametrized by $\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\sigma}^2_{\mathbf{x}}$. The decoder, $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ maps $\mathbf{z}$ back to reconstruct $\mathbf{x}$.

2.5 The Reparameterization Trick

The sampling issue arises because the gradients cannot flow back through the sampling procedure to reach the encoder network since sampling is not differentiable. This issue can thus be avoided by diverting the non-differentiable sampling procedure outside of the network. Instead of sampling $\mathbf{z}|\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\sigma}^2_{\mathbf{x}}\mathbf{I})$, we have:

$$\mathbf{z}|\mathbf{x} = \boldsymbol{\mu}_{\mathbf{x}} + \boldsymbol{\sigma}_{\mathbf{x}} \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{19}$$

Here $\mathbf{z}|\mathbf{x}$ still follows the same distribution as before, but the gradients can now flow through the linear function used to derive $\mathbf{z}|\mathbf{x}$.

This completes the description of variational autoencoders. Refer to algorithm 1 for the full training procedure in the case where we are interested in encoding images with pixels $\in [0, 1]$. This algorithm can be modified to work with larger ranges of $\mathbf{x}$ by using the appropriate form of $\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ in 16.

Now, having learned estimates for $p_{\boldsymbol{\theta}}(\mathbf{z}), p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, we can return to our original goal of estimating $p_{\boldsymbol{\theta}}(\mathbf{x})$. This can be achieved using an MCMC estimator but the details are not interesting from a dimension reduction perspective so we refer the reader to [5] for more information.

Algorithm 1: Example VAE training algorithm for $[0, 1]$ images

```
Initialize neural networks $f_{enc}, f_{dec}$;
for $iter = 1, 2, \dots I$ do
    Sample batch $\mathbf{X}'$ of $M$ observations from $\mathbf{X}$;
    Set $ELBO = 0$;
    for $\mathbf{x} \in \mathbf{X}'$ do
        Compute $\boldsymbol{\mu}_{\mathbf{x}}, \boldsymbol{\sigma}_{\mathbf{x}} = f_{enc}(\mathbf{x})$;
        Sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$;
        Set $\mathbf{z} = \boldsymbol{\mu}_{\mathbf{x}} + \boldsymbol{\sigma}_{\mathbf{x}} \odot \epsilon$;
        Compute $p_{\mathbf{z}} = f_{dec}(\mathbf{z})$;
        Calculate the KL loss $\mathcal{L}_{KL}$ using 18;
        Calculate the reconstruction likelihood $\mathcal{L}_R$ using 15;
        Update $ELBO = ELBO + (\mathcal{L}_{KL} + \mathcal{L}_R)$
    Gradient descent step, maximize $ELBO$ wrt $\boldsymbol{\phi}, \boldsymbol{\theta}$
```

2.6 Variants

We presented the standard formulation for VAEs first proposed by [5]. However, over the past few years multiple variants have been introduced and VAEs remain an active are of research. VAEs also have close connection to traditional, deterministic autoencoders.

2.6.1 Autoencoders

Traditional, deterministic autoencoders map an input $\mathbf{x}$ to an encoding $\mathbf{z}$, and attempt to reconstruct $\mathbf{x}$ using that encoding. The encoding $\mathbf{z}$ is deterministic so there is no sampling involved. These non-variational autoencoders are a special case of VAEs with $\boldsymbol{\sigma}_{\mathbf{x}}^2 = \vec{0}$. The first likelihood term in equation 13 is the reconstruction error while the second KL term regularizes the norm of the learned encoding. Autoencoders have been part of the neural network landscape for decades while VAEs are a recent development [1, 5].

From a dimension reduction perspective, VAEs have many attractive features that are not present in autoencoders. Some of these features are discussed in the Strengths section (3.1) below.

2.6.2 Priors

In the above discussion we assumed the prior distribution $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z} \,|\, \vec{0}, \mathbf{I})$. This prior can encourage the representations learned by $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to have desirable properties (see section 3.1). This can be attributed to the KL term in the ELBO objective pushing $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to be as close as possible to $p_{\boldsymbol{\theta}}(\mathbf{z})$.

In the same way, we can make different assumptions on $p_{\boldsymbol{\theta}}(\mathbf{z})$ to encourage learning representations with specific properties. For example, [6] introduce priors that result in sparse representations (i.e. $\mathbf{z}$ is mostly zeros) or clustered representations (i.e. $\mathbf{z}$ embeddings are clustered around a small number of points in space). This flexibility on prior opens the door for many variations of VAEs depending on the setting.

3 Strengths and Weaknesses

So far we have followed the original motivation behind VAEs as generative models. Recent studies have shown that VAEs have many attractive features as dimensionality reduction algorithms [3]. In this section, we describe the strengths and weaknesses of VAEs for both dimension reduction and data generation.

3.1 Strengths

Generation: A major strength of VAEs that many dimension reduction algorithms lack is that VAEs learn a generative model of the data which can help us understand and simulate the data generating process.

Interpretability: Another advantage of VAEs is that they learn disentangled representations of the data. The isotropic Gaussian prior $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{z} \,|\, \vec{0}, \mathbf{I})$ pushes the encoder distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ to learn embeddings with independent dimensions (due to the KL term between these two distributions in the ELBO objective). This requirement, although weakly enforced through one term in the objective, results in representations where each dimension corresponds to an independent latent generative factor in simple cases. Representations satisfying this property are called disentangled representations since we can adjust a specific dimension separately to control a feature of the generated data. For example, if we are encoding images of circles, a specific dimension of $\mathbf{z}$ could correspond to the size of the circle and another could correspond to the color of the circle [2].

Flexibility: The neural networks used to estimate $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}), q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ can capture complex and non-linear manifolds. Large neural networks are highly flexible and can approximate any continuous function on a closed, bounded subset of $\mathbb{R}^n$ (see the universal approximation theorem [1, 4]). The powerful modelling abilities of neural networks extend to VAEs and the manifolds they learn.

Prior Knowledge: VAEs are highly flexible since different priors allow us to encourage learning representations $\mathbf{z}$ with different properties such as sparsity and disentanglement (see section 2.6.2). This is another strength of VAEs since it allows for incorporating prior domain knowledge and tuning VAE modelling assumptions to fit the setting in question. Another feature of these priors is that they provide a crucial regularizing effect that set VAEs apart from deterministic autoencoders.

Robustness and Manifold Learning: VAEs have been shown to be more robust to outliers relative to deterministic autoencoders due to the regularization applied by the KL term [3]. It has also been shown that in simple cases when the asummed dimension of $\mathbf{z}$ (defined by the neural network architecture) is larger than necessary, the extra dimensions are pruned and the true dimension of $\mathbf{z}$ is reflected in the model [3].

3.2 Weaknesses

Noise: Recall that the ELBO objective was derived by minimizing the KL divergence between $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}), p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ in equation 2. This objective can be maximized by producing blurry and smooth images that ignore small details. Thus the generated images are often too blurry to be realistic.

Complexity: VAEs do not have simple closed form solutions and require optimization with iterative methods. This can lead to getting stuck in local minima and sub-optimal solutions. Large VAEs also require a long time to train.

Flexibility: Large VAEs can learn complex non-linear manifolds which could be both a strength and a weakness. An over-parameterized encoder can learn to memorize each example $\mathbf{x}$ and place it into a specific region of the space of $\mathbf{z}$. An over-parameterized decoder can then learn to perfectly reconstruct this memorized sample from $\mathbf{z}$ to $\mathbf{x}$. Thus, a large model can have small training error without capturing useful manifolds or generalizing to test data. This overfitting can be somewhat controlled by the regularizing effect of the KL term in the ELBO objective but this is not guaranteed [3].

4 Examples

Here we cover a few brief example illustrating the strengths and weaknesses of variational autoencoders.

4.1 MNIST

We implement a variational autoencoder for MNIST. Our model is composed of 8 hidden layers, half of which act as the encoder with the other half acting as the decoder. The MNIST images are $28 * 28 = 784$ pixels so our model takes a 784 dimensional input. We assume that $\mathbf{z} \in \mathbb{R}^2$, so we compute 2-dimensional means $\boldsymbol{\mu}_{\mathbf{x}}$ and covariances $\boldsymbol{\sigma}^2_{\mathbf{x}}$. Since the image pixels are $\in [0,1]$ we use the Bernoulli $\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ form in 15. The hidden layer sizes are:

$$\begin{gathered}\overbrace{784 \rightarrow 400 \rightarrow 256 \rightarrow 128}^{\text{Encoder}} \rightarrow \\ 2, 2 \\ \underbrace{\rightarrow 128 \rightarrow 256 \rightarrow 400 \rightarrow 784}_{\text{Decoder}}\end{gathered}$$

At each layer of the network to map an s dimensional input, $\mathbf{x} \in \mathbb{R}^s$, to a t dimensional output, $\mathbf{y} \in \mathbb{R}^t$, we apply a non-linear transformation:

$$\mathbf{y} = f(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{20}$$

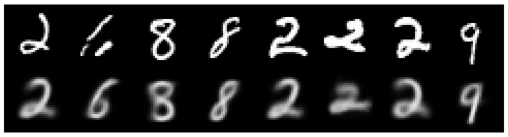
where our goal is to learn the optimal $\mathbf{W} \in \mathbb{R}^{t \times s}$, and $\mathbf{b} \in \mathbb{R}^t$. Here f is a non-linear activation function such as a ReLU or sigmoid.

4.1.1 Dimension Reduction

VAEs are effective at dimension reduction, encoding input images into the latent variable $\mathbf{z}$ through the encoder, and then reconstructing them back through the decoder to retrieve the original image. We can see input images and their reconstructions in figure 2. These images are from a held-out test set and were never seen during training.

We find that the VAE produces somewhat blurry images, but it is also effective at filling in missing or noisy details such as completing the 6 and writing smoother 2's.

We also plot the sampled $\mathbf{z}$ vectors used to encode the test data in figure 3. We find that our VAE learns to separate the images into different clusters as expected.


(a) MNIST

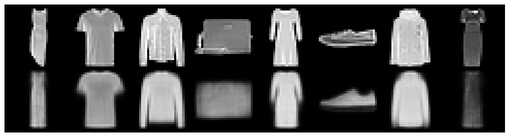

(b) Fashion-MNIST

Figure 2: Input images (top) and their reconstructions (bottom) generated by our trained VAEs.

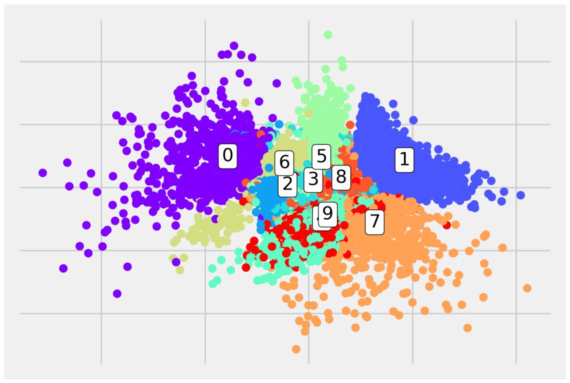
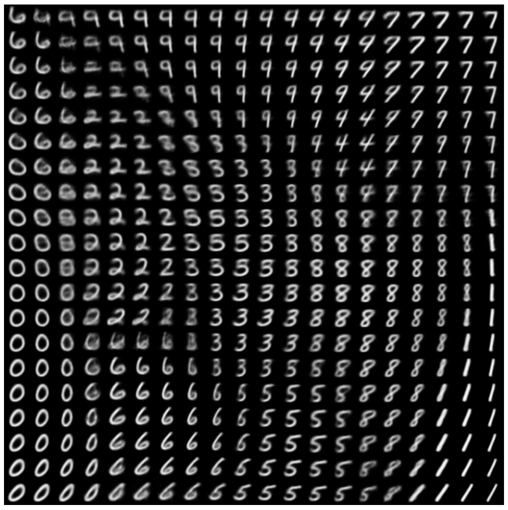


Figure 3: A scatter plot of the **z** encodings of images in the test set. Our VAE learns to classify different digits into separate clusters.

4.1.2 Manifold Learning

Since we assume that $\mathbf{z} \in \mathbb{R}^2$, we can sample images along the 2-dimensional manifold learned by the VAE. We apply the inverse CDF of the standard normal distribution (the quantile function, Φ^{-1}) on a 2D grid $\in [0.05, 0.95]$. We then feed our decoder the returned values of **z** to generate the images on the manifold in figure 4. The returned manifold is smooth, showing how numbers morph into each other as we traverse the embedding space. This manifold has connections with the scatter plot in figure 3. We see that distinct digits on the edge of the manifold such as 1,7,0 are placed in distinct clusters on the edges of the scatter plot, while similar digits that morph into each other such as 2,3,8 are clustered closely in the center.


(a) MNIST

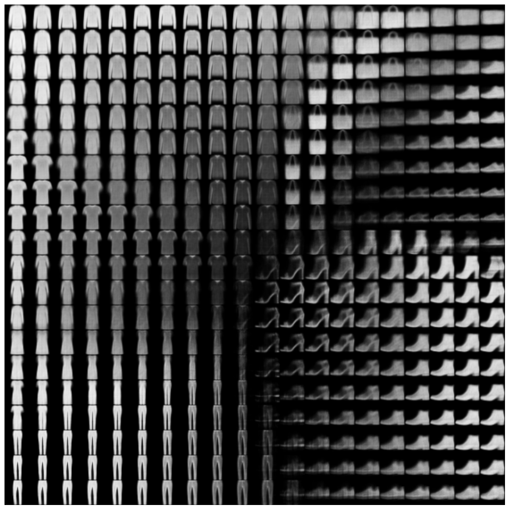
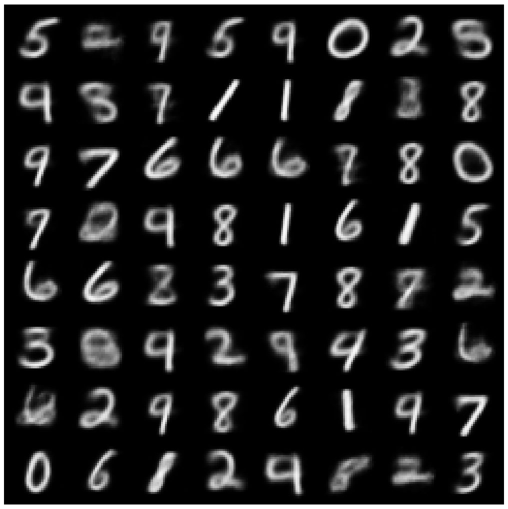

(b) Fashion-MNIST

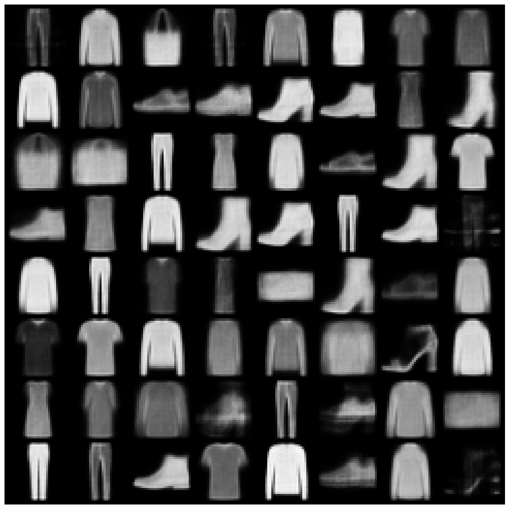
Figure 4: Examples of 2D manifolds learned by our variational autoencoder.

4.1.3 Generative Modelling

VAEs learn generative models of the data. Here we show the effectiveness of VAEs at generating new examples. We sample 64 $\mathbf{z}$ embeddings from a 2D standard multivariate normal $\mathcal{N}(0, \mathbf{I})$ and pass them through the decoder. The generated images are displayed in figure 5.



(a) MNIST



(b) Fashion-MNIST

Figure 5: New examples generated by passing random samples of $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ through the decoder.

4.1.4 Robustness

To explore the effects of noisy data, we train a VAE and a traditional autoencoder on the MNIST data corrupted with different amounts of noise. We then measure the reconstruction error achieved by each of the models. We find that for small dimensions of $\mathbf{z}$ the two models achieve comparable performance, while for large dimensions of $\mathbf{z}$ the traditional autoencoder outperforms the VAE.

[3] show that VAEs are more robust than autoencoders when training on synthetic data (refer to figure 6) but we were unable to verify that these results extend to MNIST.

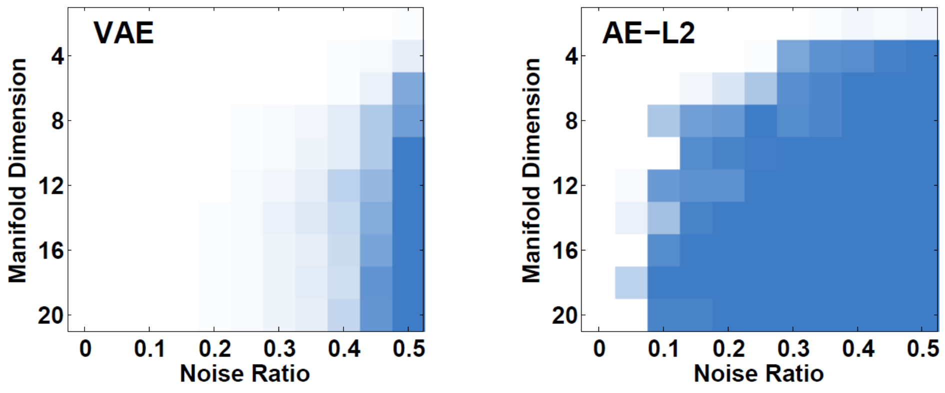


Figure 6: A scatter plot showing the robustness of VAEs to noisy synthetic data. The intensity of the blue color corresponds to higher MSE. We were unable to verify similar effects on MNIST.

4.2 Fashion-MNIST

We repeat the experiments above using the Fashion-MNIST dataset. We make the same assumptions and use the same neural network architecture. Figure 2 shows image reconstructions. Figure 4 shows the discovered manifold. Figure 5 shows some generated examples.

4.2.1 Disentanglement

We train our VAE only on the boots class from Fashion-MNIST seperately to show that the generated manifold learns disentangled/independent generative features as discussed in section 3.1. We can see from figure 7 that the x-axis corresponds to the height of the heel, while the y-axis corresponds to the height of the actual boot (from ankle height to over-the-knee) excluding the heel.

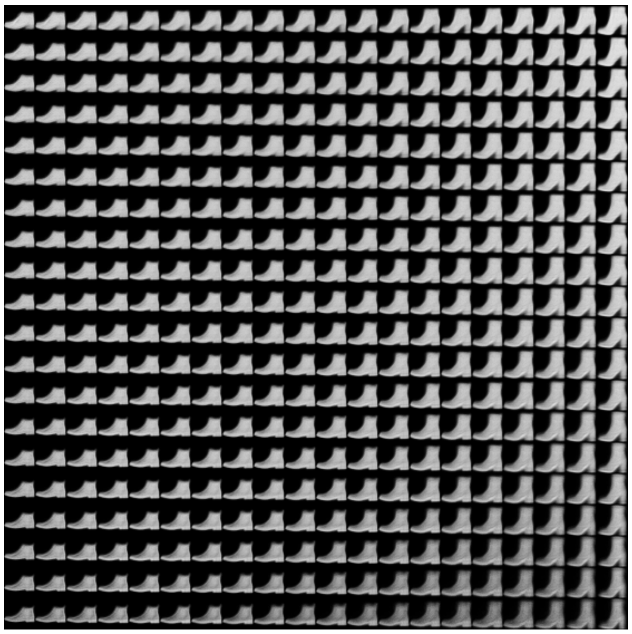


Figure 7: The VAE learns disentangled features with the x-axis being the height of the heel and the y-axis being the height of the boot (excluding the heel).

4.2.2 Manifold Dimension

We use Fashion-MNIST to explore how VAEs learn to prune unnecessary dimensions if $\mathbf{z}$ is chosen to be too large. We train the VAE architecture described in section 4.1 on the Fashion-MNIST boots, but assume $\mathbf{z} \in \mathbb{R}^{20}$ instead of $\mathbb{R}^2$. This corresponds to increasing the size of the hidden layer that generates $\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\sigma}^2_\mathbf{x}$.

VAEs can prune extra dimensions by allocating small values to the columns of the decoder weight matrix $\mathbf{W}$ that follows $\mathbf{z}$. This implies that specific dimensions of $\mathbf{z}$ are pushed to zero through the product $\mathbf{Wz}$ (refer to Eq. 20). To illustrate this effect we take the absolute value of the column sums of $\mathbf{W}$. This returns 20 values, each one corresponding to a dimension of $\mathbf{z}$. We sort these values in decreasing order and plot them in figure 8 to get something that resembles a scree plot.

We can see from this plot that magnitude of the weights quickly falls after 2 dimensions similar to an elbow plot. This confirms our previous hypothesis that the boot images lie on a 2D manifold (one corresponding to the height of the heel and the other corresponding to the height of the boot). Here our VAE was able to retrieve the true dimension of the manifold.

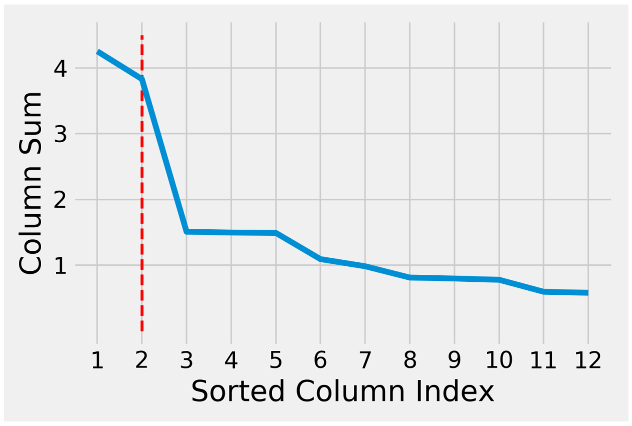


Figure 8: The VAE learns disentangled features with the x-axis being the height of the heel and the y-axis being the height of the boot (excluding the heel).

References

[1] Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. Citeseer, 2017.

[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] Bin Dai, Yu Wang, John Aston, Gang Hua, and David Wipf. Hidden talents of the variational autoencoder. *arXiv preprint arXiv:1706.05148*, 2017.

[4] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

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

[6] Emile Mathieu, Tom Rainforth, Siddharth Narayanaswamy, and Yee Whye Teh. Disentangling disentanglement. *arXiv preprint arXiv:1812.02833*, 2018.