Variational Autoencoders

Joyce Lu

December 14, 2019

1 Introduction

First defined by Kingma and Welling (2013) and Rezende, Mohamed, and Wierstra (2014), *variational autoencoders* (VAEs) belong to a broad class of machine learning methods called *generative models* (as opposed to *discriminative models*). The distinction between the two is summarized below [4].

Definition 1.1. A *discriminative model* aims to learn the conditional probability of the target variable given the observed variables, modeled as $p(z|x)$.

Definition 1.2. A *generative model* aims to learn the joint probability distribution over all variables, observed and target, modeled as $p(z, x)$.

While discriminative models may perform better on narrow classification tasks, they fall short in more flexible situations that require learning a set of generalizable rules which can then be used to, for instance, reconstruct data [4]. The tension between discriminative models and generative models is a useful analogy for the relationship between standard autoencoders and variational autoencoders. Although it shares the same name and basic architecture as the standard autoencoder, the variational autoencoder is much more powerful and suited for a different range of tasks. First, I will provide a brief overiew of a standard autoencoder, and then elaborate on how variational autoencoders build on their structure.

A standard autoencoder is a neural network architecture comprising an *encoder*, which receives high-dimensional data $\mathbf{x}$ and projects it into a lower-dimensional representation $\mathbf{z}$ ("encoding"), and a *decoder*, which takes the encoding $\mathbf{z}$ and attempts to generate as close a reconstruction $\tilde{\mathbf{x}}$ ("decoding") to the original as possible. The goal of encoding is to capture *latent variables* or *latent factors*, which are hidden, unobserved features of the data that exist in a lower-dimensional *latent space* [1].

Autoencoders are trained to minimize information loss, or reconstruction error, during the encoding-decoding process via gradient descent. However, using reconstruction error alone as the loss function incentivizes distinct clustering, which may lead to an irregular or discontinuous latent space. As a result, it is possible that observations which lie close together as lower-dimensional encodings in the latent space may not have similar decodings when projected back, and decoding from regions between clusters may result in meaningless re-

constructions [7].

VAEs avoid this problem by assigning probabilistic distributions to the data, which are deterministic in the standard autoencoder framework. In the encoding process, a VAE encodes probability *distributions* rather than singular points, and optimizes a loss function that incentivizes simple, orderly latent spaces and reconstructions that are similar to the original data points. The objective function is optimized using techniques from variational inference, hence the name [7].

The fundamental workings of a VAE can be understood as follows:

1. encode data as a probabilistic distribution over the latent space
2. sample a point from the latent space
3. optimize the objective function
4. decode the sampled point

2 Method

2.1 Notation and Assumptions

Let $\mathbf{x}$ be i.i.d. observations in $\mathbb{R}^n$ and $\mathbf{z}$ be the latent variables in $\mathbb{R}^m$, with $m \ll n$. In the mathematical formulation of VAEs, both $\mathbf{x}$ and $\mathbf{z}$ are drawn from a probability distribution. Then, their joint probability is

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}). \tag{1}$$

The process by which the data are generated can be described as follows: for each observation i, (1) a latent variable $\mathbf{z}_i$ is sampled from the prior distribution $p(\mathbf{z})$ and (2) an observation $\mathbf{x}_i$ is sampled from the distribution $p(\mathbf{x}|\mathbf{z})$. Each draw i is i.i.d. On the RHS of the equation, we recognize the conditional probability $p(\mathbf{x}|\mathbf{z})$ as the likelihood of the data $\mathbf{x}$ and $p(\mathbf{z})$ as the prior probability of the latent variables $\mathbf{z}$.

Recall that the first goal of VAEs is to predict the latent variable via encoding; thus, we would like to obtain the posterior distribution $p(\mathbf{z}|\mathbf{x})$. Rewriting the LHS of Eq 1 to make the connection between prior, posterior, and likelihood more explicit, we have:

$$p(\mathbf{z}|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{2}$$

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} \tag{3}$$

Now, we can clearly see that $p(\mathbf{z}|\mathbf{x})$ is the posterior, $p(\mathbf{z})$ is the prior, and $p(\mathbf{x}|\mathbf{z})$ is the likelihood. The denominator, $p(\mathbf{x})$, also has a name: it is called evidence.

Evidence can be rewritten using the Law of Total Probability. That is,

$$p(\mathbf{x}) = \int_z p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{4}$$

Unfortunately, this integral does not have an analytical solution, nor can it be directly estimated in a computationally efficient way—it is intractable. Instead of calculating the evidence to obtain the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$, we must approximate it using a distribution $q_\phi(\mathbf{z}|\mathbf{x})$, where θ and ϕ represent the parameters of their respective distributions. In the next section, I will introduce the objective function that we must optimize in order to achieve the best approximation.

2.2 Objective Function

Kullback-Leibler divergence is a measure of how different a probability distribution is from a reference distribution. Thus, we can use it to evaluate how well $q_\phi(\mathbf{z}|\mathbf{x})$ does in approximating the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. In our setting, K-L divergence is defined as follows:

$$\begin{aligned}
\mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \;||\; p_\theta(\mathbf{z}|\mathbf{x})) &= E_q[\log q_\phi(\mathbf{z}|\mathbf{x})] - E_q[\log p_\theta(\mathbf{z}|\mathbf{x})] && (5)\\
&= E_q[\log q_\phi(\mathbf{z}|\mathbf{x})] - E_q\left[\log \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{x})}{p_\theta(\mathbf{Z})}\right] && (6)\\
&= E_q[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x},\mathbf{z}))] + \log p_\theta(\mathbf{x}) && (7)
\end{aligned}$$

Our goal is to minimize this quantity over the parameters ϕ and θ; in other words,

$$q^*_\phi(\mathbf{z}|\mathbf{x}) = \text{argmin}_{\phi,\theta}\mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \;||\; p_\theta(\mathbf{z}|\mathbf{x})) \tag{8}$$

Notice, however, that the evidence term, $p_\theta(\mathbf{x})$, is present in the expression that we wish to optimize. Attempting to approximate the posterior distribution might then appear circular, but luckily we can eliminate the need to calculate the evidence term by introducing the ELBO function:

$$ELBO(\phi,\theta) = E_q[\log(p_\theta(\mathbf{x},\mathbf{z}))] - E_q[\log(q_\phi(\mathbf{z}|\mathbf{x}))] \tag{9}$$

Combining with K-L divergence and rearranging, we have

$$\log p_\theta(\mathbf{x}) = \mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \;||\; p_\theta(\mathbf{z}|\mathbf{x})) + ELBO(\phi,\theta) \tag{10}$$

Equation 10 is central to the theory of VAEs because it provides a lower bound on the evidence term, as I will now show. First, as the log function is concave (and the negative of the log function is convex), we can use Jensen's inequality to prove that K-L divergence is always nonnegative:

$$\begin{aligned}
\mathbb{KL}(f(x) \;||\; g(x)) &= E_f[\log f(x)] - E_f[\log g(x)] && (11)\\
&= E_f[-\log(g(x)/f(x))] && (12)\\
&\geq -\log E_f[g(x)/f(x)] && (13)\\
&\geq -\log \int g(x)dx && (14)\\
&\geq 0 && (15)
\end{aligned}$$

Then, if K-L divergence is always greater than or equal to 0, we have that

$$\log p_\theta(\mathbf{x}) \geq ELBO(\phi, \theta) \tag{16}$$

That is, $ELBO(\phi, \theta)$, short for **E**vidence **L**ower **BO**und, is the lower bound on $\log p_\theta(\mathbf{x})$! Because of the inverse relationship between the two expressions, minimizing K-L divergence is akin to maximizing ELBO. The next section will walk through two methods, borrowed from variational inference, that can be used to optimize ELBO.

2.3 Maximizing the ELBO Function

Before we jump into the optimization, let us broadly recap what we have covered so far. Recall that an autoencoder comprises an encoder and decoder. In our probabilistic model, the encoder is $q_\phi(\mathbf{z}|\mathbf{x})$ and the decoder is $p_\theta(\mathbf{x}|\mathbf{z})$; note that in the literature, the encoder is also called the *inference model* or the *recognition model*, while the decoder is called the *generative model*. Optimizing the ELBO function over ϕ and θ achieves two goals simultaneously:

1. by definition, minimizing the K-L divergence between our approximation to the posterior and the true posterior, and
2. maximizing the evidence $p_\theta(\mathbf{x})$, which gives a better generative model by increasing the marginal likelihood of reconstructing output similar to the original data.

Furthermore, up until now, we have not defined a family of distributions for the prior distribution from which the latent variables are drawn. Typically, simple priors such as Gaussians are taken to be reasonable models of latent attributes; in cases where a Gaussian is not a sufficient representation for the latent space, it may still be easily mapped to an arbitrarily complex function. For the purposes of providing a concrete example in Method 2 (Reparametrization Trick), we will assume a Gaussian prior.

2.3.1 Method 1: Score Function Gradient

The first method is called *score function gradient* or, in reinforcement learning circles, REINFORCE. It stems from a very simple idea: that the gradient of a function is the gradient of the log of the function multiplied by the function itself. In our setting, we wish to obtain the gradient

$$\nabla ELBO(\phi, \theta) = \nabla E_q[\log(p_\theta(\mathbf{x}, \mathbf{z})) - \log(q_\phi(\mathbf{z}|\mathbf{x}))] \tag{17}$$

Using the score function gradient or *score function estimator*, we can compute the gradient with respect to ϕ and θ, and then use Monte Carlo estimation to achieve an unbiased estimate, assuming i.i.d. data.

First, to take the gradient with respect to θ, we can simply move the gradient inside the expectation since we are taking the expectation with respect to ϕ, not θ. Thus, we have

$$\nabla_\theta ELBO(\phi, \theta) = E_q[\nabla_\theta(\log(p_\theta(\mathbf{x}, \mathbf{z})) - \log(q_\phi(\mathbf{z}|\mathbf{x})))] \tag{18}$$

$$= E_q[\nabla_\theta \log(p_\theta(\mathbf{x}, \mathbf{z}))] \tag{19}$$

For the gradient with respect to ϕ, we use the log-derivative trick:

$$\nabla_\phi ELBO(\phi, \theta) = E_q[(\log(p_\theta(\mathbf{x}, \mathbf{z})) - \log(q_\phi(\mathbf{z}|\mathbf{x})))\nabla_\phi \log q_\phi(\mathbf{z}|\mathbf{x})] \tag{20}$$

Now we have the gradients of the ELBO function in forms that we can estimate using Monte Carlo methods. The steps for Monte Carlo sampling are as follows: (1) draw S samples from the distribution, (2) evaluate the expression inside the expectation for each sample, and (3) compute the mean of the expressions.

2.3.2 Method 2: Reparametrization Trick

In their paper, Kingma and Welling propose using the reparametrization trick instead. Recall that we assumed that our latent variables are sampled from a Multivariate Normal distribution $q_\phi(\mathbf{z}|\mathbf{x}) \sim N(\mu, \Sigma)$. However, once we sample the $\mathbf{z}$, they are fixed, and it is impossible to take a gradient of a function of fixed values $\nabla E_q[f(\mathbf{z})]$, where $f(\mathbf{z})$ is the ELBO function that we wish to maximize.

In this case, we can reparametrize $\mathbf{z}$ such that the draw is a function of the variational parameters ϕ (in this case μ and Σ since $\mathbf{z}$ is Gaussian), and the randomness is no longer dependent on those parameters. Compare the original representation

$$\mathbf{z}|\mathbf{x} \sim N(\mu, \Sigma), \tag{21}$$

where it is not possible to take a gradient with respect to the variational parameters, to the sampling

$$\mathbf{z}|\mathbf{x} \sim \mu + L\epsilon, \tag{22}$$

where $\Sigma = LL^T$ and $\epsilon \sim N(0, I)$. In the second representation, $\mathbf{z}$ can be rewritten deterministically as a function of the original parameters and the random noise variable ϵ:

$$\mathbf{z} = \mathbf{g}_\phi(\epsilon; \mathbf{x}) \tag{23}$$

Then, we can perform a change of variables and rewrite our original expectation in terms of the distribution of ϵ, $r(\epsilon)$:

$$\nabla_\phi E_q[f(\mathbf{z})] = \nabla_\phi \int q_\phi(\mathbf{z}) f(\mathbf{z}) d\mathbf{z} \tag{24}$$

$$= \nabla_\phi \int r(\epsilon) f(\mathbf{z}) d\epsilon \tag{25}$$

$$= \nabla_\phi \int r(\epsilon) f(g_\phi(\epsilon; \mathbf{x})) d\epsilon \tag{26}$$

$$= \nabla_\phi E_r[r(\epsilon) f(g_\phi(\epsilon; \mathbf{x}))] \tag{27}$$

Now that the argument inside the expectation is a deterministic function, it is possible to use backpropagation to reach a local maximum of the ELBO function.

2.3.3 Auto-Encoding Variational Bayes (AEVB)

The method for computing the maximum based on reparametrization is called Auto-Encoding Variational Bayes (AEVB), and is a form of doubly stochastic optimization, since there is randomness in both the selection of a subset of data as well as the generation of the random noise ϵ. Given a set of data $\mathbf{X}$ with the goal to learn the parameters θ and ϕ, the algorithm is as follows:

1. initialize parameters (θ, ϕ)
2. while SGD has not converged

   sample a random minibatch of data, $\mathbf{X_s} \sim \mathbf{X}$

   generate random noise for each data point in $\mathbf{X_s}, \epsilon \sim N(0, I)$

   compute ELBO and gradients using reparametrization trick

   update θ and ϕ

3 Discussion

Let us take another look at the ELBO as the lower bound on $\log p_\phi(\mathbf{x})$, with the expression slightly rearranged using an identity for joint probabilities, properties of logarithms, and the linearity of expectation:

$$\log p_\theta(\mathbf{x}) \geq E_q[\log(p_\theta(\mathbf{x}, \mathbf{z})) - \log(q_\phi(\mathbf{z}|\mathbf{x}))] \tag{28}$$

$$\geq E_q\big[\log\left(\frac{p_\theta(\mathbf{x}|\mathbf{z})}{p_\theta(\mathbf{z})}\right) - \log(q_\phi(\mathbf{z}|\mathbf{x}))] \tag{29}$$

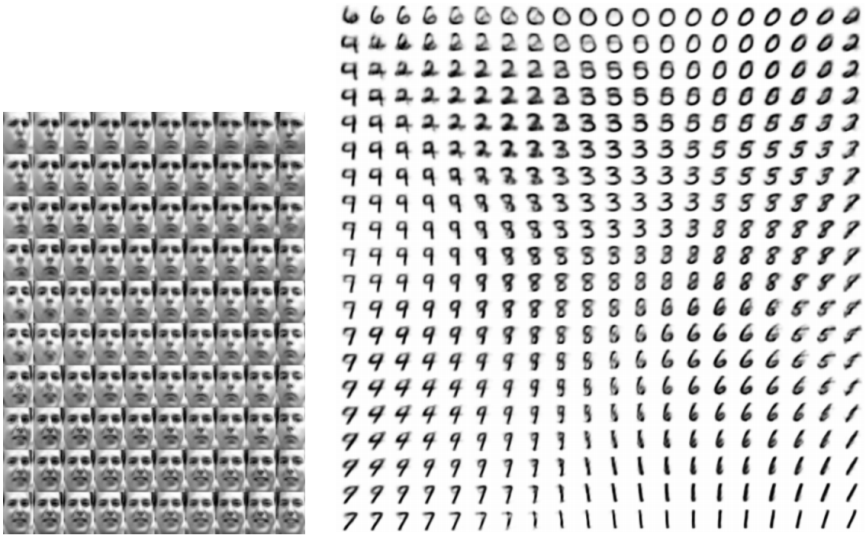
$$\geq E_q(\log(p_\theta(\mathbf{x}|\mathbf{z})) - \mathbb{KL}(q_\phi(\mathbf{z}|\mathbf{x}) \;||\; p_\theta(\mathbf{z})) \tag{30}$$

This equation offers an intuitive explanation of what a VAE does. Examining the RHS of the inequality, we see it is composed of two terms. The first is the likelihood of the observed $\mathbf{x}$ given the sampled latent variable $\mathbf{z}$, and is known as the *reconstruction error*. Because we are maximizing ELBO, this term encourages a reconstruction close to the original.

The second term is the K-L divergence between the posterior distribution of $\mathbf{z}$ and the prior distribution of $\mathbf{z}$, and is called the *regularization term*. It serves to penalize latent variables that are not close to the assumed prior distribution (which, again, in our case is Gaussian).

This equation, then, explains why VAEs are so suited to generative tasks, such as producing and editing digital images that are similar to but not exactly the same as input images. Because both of the reconstruction error and the regularization term occur in the objective function, it forces the VAE to encode the data as a distribution closely tied to the Gaussian while generating meaningful decodings that do not stray too far from the input data. Its major strength lies in the fact that it is a principled approach wherein it learns a distribution for the latent variables, rather than treating them deterministically.

Another important strength of VAEs is that because we assume independent or very weakly correlated draws from the prior distribution, the latent factors lend themselves to being easily interpretable. In the images below, taken from [3], we can see that as the percentiles of the draws from the prior vary, there is a gradual, smooth shift in the effects of the latent variables along each axis. This interpretability is present in both the latent variables and the reproduced images; due to the regularity of the latent space, we can be much less wary of nonsensical or meaningless representations.



(a) Learned Frey Face manifold

(b) Learned MNIST manifold

The reparametrization trick is also a strength of VAEs. In fact, this method as applied to VAEs is a key contribution of the original papers by Kingma et. al and Rezendes et. al. The major advantage of reparametrization over other Monte Carlo sampling methods, such as the score function gradient, is that it has a much smaller variance. While the variance of the score function estimator scales linearly with the number of independent random variables in the target function, it can be shown that the variance of reparametrization is that the estimator resulting from the reparametrization method is bounded by a constant. In fact, among competing estimators, the estimator resulting from reparametrization often has the lowest variance due to implicit modeling of dependencies via the chain rule [2].

On the flip side, the reparametrization trick comes with assumptions that may not necessarily be kosher. First, it assumes that the prior comes from a "nice" family of continuous distributions, such as Exponential, Gaussian, and Log-Normal, rather than a more complex distribution or one that is discrete. Thus, it limits our choice of model for the latent space to a continuous distribution [10]. In addition, VAEs depend heavily on the assumption that data points are i.i.d. samples from a static underlying distribution. In cases where data points may not be independent or the underlying system changes, the estimators may be unbiased, leading to inaccurate learned parameters [4].

Also, keep in mind that the optimization in VAEs is merely maximizing the lower bound of the likelihood, rather than the likelihood itself. Therefore, it is not as direct an optimization

as other methods have.

There is no single way to evaluate performance of VAEs, but one metric used for Gaussian priors is the Frechet Inception Distance (FID) [9], defined as followed, with $\mathbf{x_r}$ being the ground truth (real) and $\mathbf{x_g}$ being the generated reconstructions (both sampled from Gaussians):

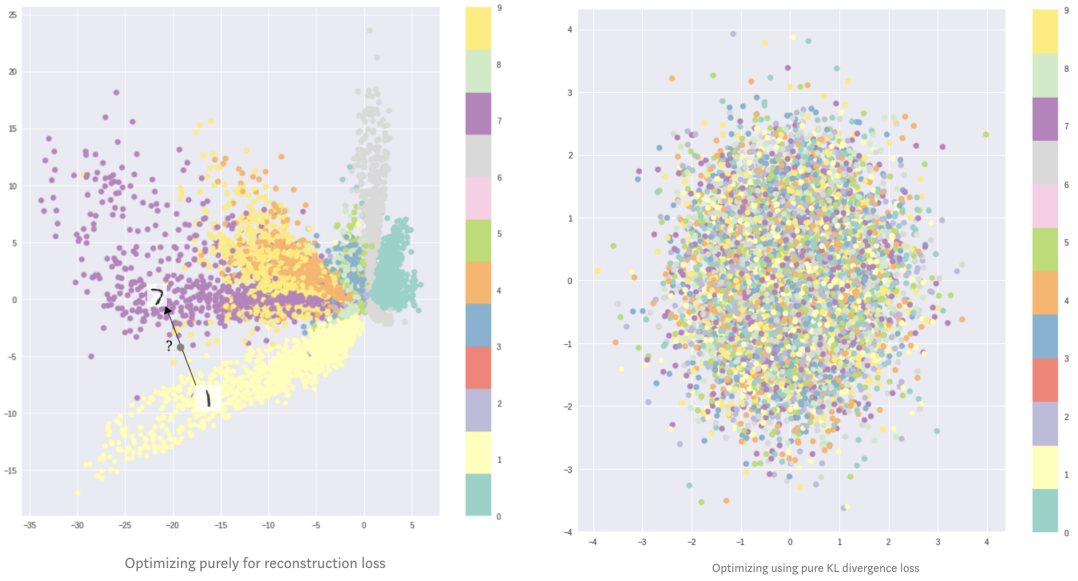
$$FID = ||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{\frac{1}{2}} \tag{31}$$

Lower FID is better.

4 Examples

4.1 Investigating Latent Spaces

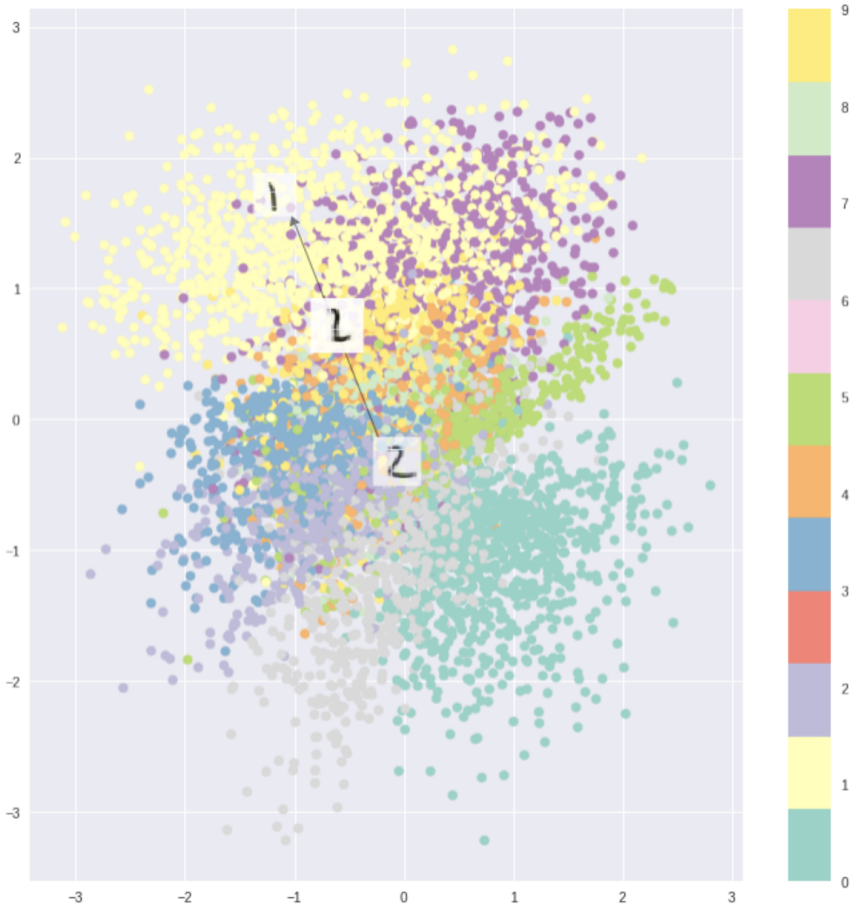
Below are examples taken from [5] to illustrate the importance of including both reconstruction error and the regularization term in the objective function.



The data for this example is the canonical MNIST digits. The first image shows the latent space—that is, the lower-dimensional projections of the data—using only reconstruction loss as the loss function, while the second image shows the latent space using only K-L divergence as the loss function. As you can see, in the first image, the latent space is extremely irregular; while there is good separation and distinct clusters, the density of the projections are concentrated at the origin and then splay out with great variance in the leftward direction to various degrees. By contrast, in the second image, there is no separation at all between the different digits, and the points form a giant cluster in the center.

The third image shows a happy medium between the two; by incorporating both terms, the new loss function has formed distinct clusters while minimizing the gaps between the

clusters. The new latent space is thus essentially continuous, and can handle interpolating reconstructions of data based on encodings that were not seen previously.



Optimizing using both reconstruction loss and KL divergence loss

4.2 Reconstructing MNIST Digits

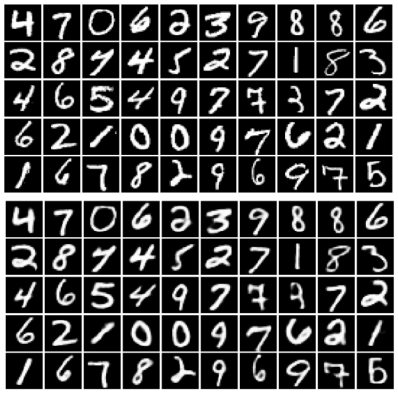


Figure 2: Top: Digits from the validation set, Bottom: After projecting these digits, the reconstruction using the decoder.

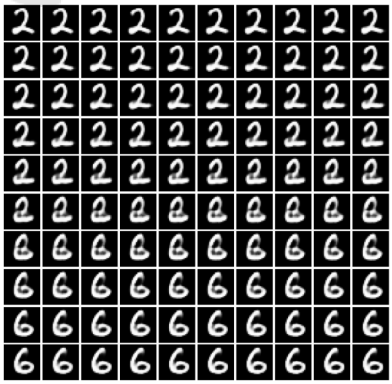


Figure 3: Example of linear interpolation between two points in the latent space, corresponding to the numbers 2 and 6, and the reconstruction of the computed points in between.

The above two images again are reconstructions of digits from the classic MNIST data set. As you can see, the bottom set of digits in the first picture are very similar to the top set. However, close examination reveals that the generated images are a little fuzzier than their sharper counterparts on top; this is a weakness of VAEs in image reconstruction as compared to their counterparts in the machine learning realm.

The figure on the right shows examples of interpolation between two points in the latent space. We can see very clearly that the 2s morph steadily into 6s, with the digits in between them incorporating features of both. Whereas a normal autoencoder would likely produce random noise in the middle digits, VAE produces a smooth, continuous spectrum. Below, we see a similar continuous progression between pairs of the same characters, this time from the UJIPEN handwritten characters data set.

The images are from [6].

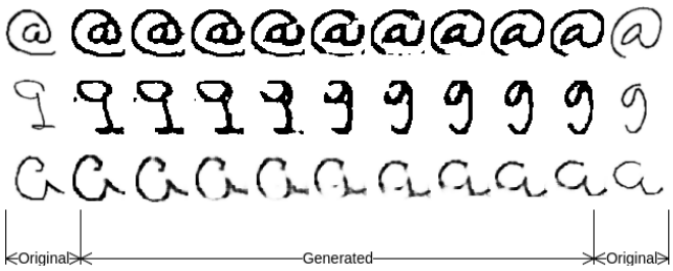


Figure 5: Example of linear interpolation between pairs of characters: “@”, “9” and “a”.

References

[1] Carl Doesch. *Tutorial on Variational Autoencoders.* Carnegie Mellon/UC Berkeley, 2016.

[2] Danilo J. Rezende, Shakir Mohamed, Daan Wierstra. *Stochastic Backpropagation and Approximate Inference in Deep Generative Models.* Proceedings of the 31 st International Conference on Machine Learning, 2014.

[3] Diederik P. Kingma and Max Welling. *Auto-Encoding Variational Bayes.* 2014.

[4] Diederik P. Kingma and Max Welling. *An Introduction to Variational Autoencoders.* Foundations and Trends in Machine Learning, Vol. 12 (2019): No. 4, pp 307-392, Dec 2017.

[5] Irhum Shafkat. *Intuitively Understanding Variational Autoencoders.* Feb 2018.

[6] Javier Jorge et. al. *Empirical Evaluation of Variational Autoencoders for Data Augmentation. VISIGRAPP Vol. 5*, 2018.

[7] Joseph Rocca. *Understanding Variational Autoencoders (VAEs).* 2019.

[8] Jaan Altosaar. *Tutorial - What is a variational autoencoder?*

[9] Neal Jean. *Fréchet Inception Distance.* Jun 2018.

[10] Sayed Ashaar Javed. *REINFORCE vs Reparametrization.* Aug 2018.