Understanding the ELBO in VAEs and β-VAEs

Nikita Baklazhenko, Miguel Conner, David Vallmanya, Dominik Wielath

March 2023

In this project, we sought to understand how VAEs leverage sampling to outperform autoencoders purely built with neural networks. By plotting the latent space of a vanilla autoencoder and a VAE for the MNIST data set and comparing images created, we visualize how the VAE changes the distribution of the latent space and improves image generation. As an extension, we further observe how different weighting of the KL divergence in the loss function of our model affects our results. We do so by following the architecture of β-VAEs introduced by Higgins et al. 2017. This implementation allows us to analyze the tension between image recreation and approximating a tractable distribution in the latent space.

1 Introduction

The Variational Autoencoder was first proposed by Kingma and Welling 2013, and later expanded in Kingma and Welling 2019. It is an autoencoder that uses Variational Inference methods to perform inference and generative functions. Specifically, the model takes an input and runs it through a stochastic encoder, drawing a latent variable from a distribution. Then, this latent variable is run through a decoder to generate an image. The model is optimized through maximizing the ELBO using stochastic gradient descent. The ELBO is an objective function that is composed of a KL divergence term (pushing the stochastic encoder to produce a distribution similar to a $\mathcal{N}(0, 1)$) and a reconstruction loss term (pushing the generated image to look as much as possible like the original image). Part of the reason the model achieves both a nice image and a nicely distributed latent space is because it balances both of these terms. The main contribution of Kingma and Welling 2013 is to propose an estimator that allows us to accurately compute the gradient, which is discussed in more detail below.

The performance of a VAE was shown to be improved by simply changing how the two terms in the ELBO are balanced, as was first proposed in Higgins et al. 2017. By controlling the importance of having a nicely distributed latent space and having generated images that are very accurate, the VAE is better able to disentangle complex latent variables.

In this paper, we review some of the key derivations that define VAEs as discussed by Kingma and Welling 2013 and Kingma and Welling 2019. Then we visualize some of these differences using the MNIST dataset by dissecting and building on top of a basic VAE built in Keras (Dobilas 2022). We finally show how β-VAEs affect the latent space and discuss some of the improvements this model provides.

2 Methods

2.1 Scenario and model assumptions

We consider a dataset $X = \{x^{(i)}\}_{i=1}^{N}$ consisting of N i.i.d. samples from a continuous or discrete variable X. We also consider that an unobserved continuous random variable z generates the data, defining the following directed latent-variable model,

$$p(x, z) = p(x|z)p(z) \qquad x \in X \qquad z \in R^k$$

where we can imagine x being an image of a face (made up of a vector of pixel values), and z being a latent feature like how happy or sad the person is, or whether the person has blond hair or dark hair.

In this way we can describe the following generative process for the latent state $z^{(i)}$ and its corresponding observation $x^{(i)}$:

1. $z^{(i)}$ is generated from a prior distribution $p_{\theta^*}(z)$
2. $x^{(i)}$ is generated from a conditional distribution, the posterior $p_{\theta^*}(x|z)$

Where the prior $p_{\theta^*}(z)$ and the posterior $p_{\theta^*}(x|z)$ are defined by parametric families of distributions $p_\theta(z)$ and $p_\theta(x|z)$ respectively and we can consider them differentiable w.r.t. both θ and z. Additionally we assume that in our scenario,

- The posterior probability $p_{\theta^*}(x|z)$ is intractable, we cannot compute it.
- There is a large amount of data, and batch optimization is too expensive to perform. We want to update the model parameters using smaller mini-batches or individual data points because the whole dataset cannot fit in memory. For sampling we cannot afford to loop for each data point, therefore traditional sampling-based methods such as Monte Carlo EM are not suitable.

2.2 The ELBO (Evidence Lower Bound)

In a variational autoencoder the objective is to maximize the evidence lower bound, we can obtain an expression for the ELBO by developing the following expression for the log-likelihood. For any choice of the approximation distribution $q_\phi(z|x^{(i)})$ (i.e. the encoder) and variational parameters ϕ, we have:

$$\begin{aligned}
\log p_\theta(x^{(i)}) &= E_{q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] \\
&= E_{q_\phi(z|x^{(i)})}\left[\log \frac{p_\theta(x^{(i)}, z)}{p_\theta(z|x^{(i)})}\right] \\
&= E_{q_\phi(z|x^{(i)})}\left[\log \frac{p_\theta(x^{(i)}, z)}{p_\theta(z|x^{(i)})}\frac{q_\phi(z|x^{(i)})}{q_\phi(z|x^{(i)})}\right] \\
&= E_{q_\phi(z|x^{(i)})}\left[\log \frac{p_\theta(x^{(i)}, z)}{q_\phi(z|x^{(i)})}\right] + E_{q_\phi(z|x^{(i)})}\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z|x^{(i)})}\right]
\end{aligned} \tag{1}$$

where we can identify the ELBO and the Kullback-Leibler (KL) Divergence,

$$\mathcal{L}(\theta, \phi; x^{(i)}) = E_{q_\phi(z|x^{(i)})}\left[\log \frac{p_\theta(x^{(i)}, z)}{q_\phi(z|x^{(i)})}\right]$$

$$\mathrm{D}_{KL}(q_\phi(z|x^{(i)})||p_\theta(z|x^{(i)})) = E_{q_\phi(z|x^{(i)})}\left[\log \frac{q_\phi(z|x^{(i)})}{p_\theta(z|x^{(i)})}\right]$$

By rearranging the three terms we obtain the final expression for the ELBO,

$$\boxed{\mathcal{L}(\theta, \phi; x^{(i)}) = -\mathrm{D}_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + E_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)]} \tag{2}$$

We want to differentiate and optimize the lower bound $\mathcal{L}(\theta, \phi; x^{(i)})$ with respect both the variational parameters of the approximation distribution ϕ and the generative parameters θ. However, the gradient of the lower bound w.r.t. ϕ presents some issues as the usual Monte Carlo gradient estimator for this type of problem is:

$$\nabla_\phi E_{q_\phi(z)}[f(z)] = E_{q_\phi(z)}\left[f(z)\nabla_\phi \log q_\phi(z)\right] \approx \frac{1}{L}\sum_{l=1}^{L} f(z^{(l)})\nabla_\phi \log q_\phi(z^{(l)}) \qquad z^{(l)} \sim q_\phi(z|x^{(i)})$$

But this gradient estimator is impractical for our purposes as it exhibits high variance (Blei 2012). To solve this issue in a VAE setting we can use the reparametrization trick, discussed further below.

From eq.(2) we can make two important observations about the maximization of the ELBO w.r.t. the parameters θ and ϕ,

- It will approximately maximize the marginal likelihood $p_\theta(x)$. This means that the generative model will create more accurate images.
- It will minimize the KL divergence of the approximation $q_\phi(z|x)$ from the true posterior $p_\theta(z|x)$, so $q_\phi(z|x)$ becomes closer to the compared distribution.

2.3 The SG optimization of the ELBO

In a VAE setting we are interested in minimizing the loss of our neural network model by back-propagating the loss gradients through the encoder and decoder parameters which correspond to our ϕ and θ parameters respectively. This process is equivalent to maximizing the ELBO, which is the sum of individual-datapoint ELBO's for i.i.d. data:

$$\mathcal{L}_{\theta,\phi}(D) = \sum_{x \in D} \mathcal{L}_{\theta,\phi}(x) \tag{3}$$

By using a simple Monte Carlo estimator of the gradient and obtaining a random sample from $q_\phi(z|x)$ we can compute unbiased gradients of the ELBO w.r.t. the generative model parameters θ,

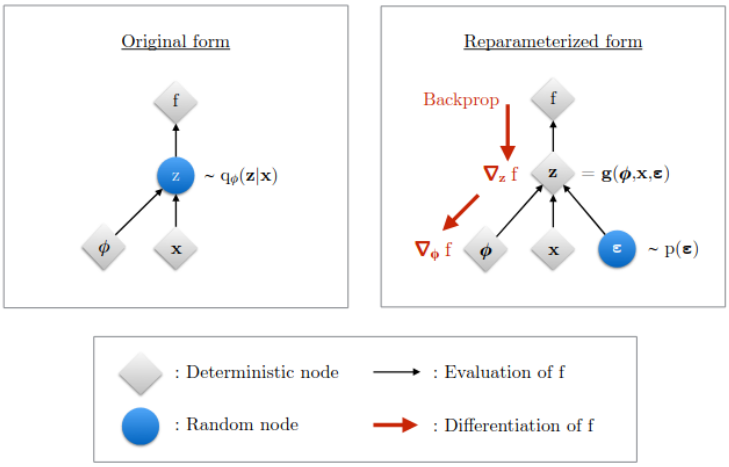

Figure 1: Illustration of the reparameterization trick. By using it we can back-propagate through z and compute the gradients $\nabla_\phi f$. Figure from Kingma and Welling 2019

$$\begin{aligned}\nabla_\theta \mathcal{L}_{\theta,\phi}(x) &= \nabla_\theta E_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] \\ &= E_{q_\phi(z|x)}\left[\nabla_\theta\left(\log p_\theta(x,z) - \log q_\phi(z|x)\right)\right] \\ &= E_{q_\phi(z|x)}\left[\nabla_\theta \log p_\theta(x,z)\right]\end{aligned} \tag{4}$$

The gradient w.r.t. the ϕ parameters of the approximation distribution is problematic as the distribution itself depends on ϕ, so we cannot apply the same methods as before,

$$\begin{aligned}\nabla_\phi \mathcal{L}_{\theta,\phi}(x) &= \nabla_\phi E_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] \\ &\neq E_{q_\phi(z|x)}\left[\nabla_\phi \log q_\phi(z|x)\left(\log p_\theta(x,z) - \log q_\phi(z|x)\right)\right]\end{aligned} \tag{5}$$

To overcome this issue we use the reparametrization trick.

2.4 The Reparameterization trick

By reparameterizing the node z, we do not sample z from $\mathcal{N}(\mu, \sigma)$ but instead we define $g(\phi, x, \epsilon) = z = \mu + \sigma * \epsilon$, where now ϵ is drawn from a $\mathcal{N}(0, 1)$. In this way we express the expectation as,

$$E_{q_\phi(z|x)}[f(z)] = E_{p(\epsilon)}[f(z)] \tag{6}$$

and compute the gradient,

$$\begin{aligned}\nabla_\phi E_{q_\phi(z|x)}[f(z)] &= \nabla_\phi E_{p(\epsilon)}[f(z)] \\ &= E_{p(\epsilon)}[\nabla_\phi f(z)] \\ &= \nabla_\phi f(z)\end{aligned} \tag{7}$$

With the reparameterization, we can replace an expectation with respect to $q_\phi(z|x)$ with one with respect to $p(\epsilon)$ and the ELBO can be rewritten in the following form,

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(x) &= E_{q\phi(z|x)}[\log p_\theta(x,z) - \log q_\phi(z|x)] \\ &= E_{p(\epsilon)}[\log p_\theta(x, g_\phi(\epsilon)) - \log q_\phi(g_\phi(\epsilon)|x)]\end{aligned} \tag{8}$$

Finally we can build a simple Monte Carlo estimator $\mathcal{L}_{\theta,\phi}(x)$ of the individual-datapoint ELBO where we use a single noise sample ϵ from $p(\epsilon)$,

$$\epsilon \sim p(\epsilon) \qquad z = g_\phi(x,\epsilon) \qquad \mathcal{L}_{\theta,\phi}(x) = \log p_\theta(x,z) - \log q_\phi(z|x) \tag{9}$$

For every datapoint we will do this series of operations that can be easily implented in TensorFlow models, and we can differentiate with respect to the parameters θ and ϕ now without issues. With the resulting gradient $\nabla_\phi \mathcal{L}_{\theta,\phi}(x)$ we can optimize the ELBO using minibatch Stochastic Gradient Descent.

2.5 Gaussian KL divergence

The KL divergence term can be solved analytically, as is shown in the appendix of Kingma and Welling 2013. Here, we reproduce the derivation. We define our $q_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x}^{(i)})$ as a Gaussian, as well as our prior $p_{\boldsymbol{\theta}}(\mathbf{z})$ where J is the dimensionality of $\boldsymbol{z}$. We note that if we use a different distribution, we would solve these integrals differently, and so our analytical solution is valid only for Gaussians. The key step that allows us to compute the integral is to convert our normal distributions to exponentials, thereby allowing us to combine terms and integrate. Then,

$$\begin{aligned}
-D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z})||p_{\boldsymbol{\theta}}(\mathbf{z})) &= \int q_{\boldsymbol{\theta}}(\mathbf{z})\Big(\log p(\mathbf{z}) - \log q_{\boldsymbol{\theta}}(\mathbf{z})\Big)\, d\mathbf{z} \\
&= \int q_{\boldsymbol{\theta}} \log p(\mathbf{z})\, d\mathbf{z} - \int q_{\boldsymbol{\theta}}(\mathbf{z}) \log q_{\boldsymbol{\theta}}(\mathbf{z})\, d\mathbf{z} \\
&= \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)\, \log \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{1})\, d\mathbf{z} - \int \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)\, \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)\, d\mathbf{z} \\
&= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(\mu_j^2 + \sigma_j^2) - \Big(-\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}(1 + \log\sigma_j^2)\Big) \\
&= \frac{1}{2}\sum_{j=1}^{J}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)
\end{aligned} \tag{10}$$

In our case, the $\boldsymbol{\mu}$ and the $\boldsymbol{\sigma}$ are simply functions of $\boldsymbol{x}$ and our parameters $\boldsymbol{\phi}$. Now, we can finally combine this with our other approximation to write:

$$\mathcal{L}(\theta, \phi; x^{(i)}) \simeq \frac{1}{2}\sum_{j=1}^{J}\Big(1 + \log\Big((\sigma_j^{(i)})^2\Big) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2\Big) + \frac{1}{L}\sum_{l=1}^{L}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}|\boldsymbol{z}^{(i,l)}) \tag{11}$$

where $\boldsymbol{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)}$ and $\boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(0, \mathbf{1})$. The RHS term comes from an estimator of the marginal likelihood lower bound of the full dataset based on minibatches. We can now quickly compute the steps in our stochastic gradient descent, and actually run our algorithm.

2.6 The Auto-encoding Variational Bayes algorithm

Having understood the components of the ELBO and the reparametrization trick we can take a look at the practical estimator of the lower bound and its derivatives with respect of the parameters ϕ and θ that was introduced by Kingma and Welling 2013. This algorithm is also very illustrative to understand the process TensorFlow or other neural network libraries do in order to train their models.

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

During this process the two components of the ELBO are computed:

- The KL Divergence $D_{\mathrm{KL}}(q_\phi(z|x^{(i)})||p_\theta(z))$ can be integrated analytically for the Gaussian case as done before, as well as for other distributions.
- The expected reconstruction error $E_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)]$ requires estimation by sampling. In our VAE setting it can be computed as the difference of the squared generated and original values from the VAE's outputs and inputs respectively.

2.7 β-VAE

We can see in eq. (2) that our ELBO term balances two terms: the KL divergence and the reconstruction loss. What if we can change the how these two terms are balanced? To do so, we follow the framework introduced by Higgins et al. 2017.

$$max_\theta E_{p\theta(z)}[p_\theta(x|z)]$$

$$max_{\phi,\theta} E_{x\sim D}[E_{q_\phi(z|x)}[logp_\theta(x|z)]] \quad subject \quad to \quad D_{KL}(q_\phi(z|x)||p(z)) < \epsilon$$

Rewritten as a Langragian under the KKT conditions, we get:

$$F(\theta, \phi, \beta; x, z) = E_{q_\phi(z|x)}[logp_\theta(x|z)] - \beta(D_{KL}(q_\phi(z|x)||p(z)) - \epsilon)$$

In other words, for a parameter β we can rewrite our ELBO as:

$$\boxed{F(\theta, \phi, \beta; x, z) \geq \mathcal{L}(\theta, \phi; x^{(i)}) = -\beta \mathrm{D}_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + E_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}|z)]} \tag{12}$$

Intuitively, we imagine that as β increases, our KL divergence term becomes penalized more harshly, meaning that our algorithm tries harder to make our posterior $q_\phi(z|x^{(i)})$ track a $\mathcal{N}(0, 1)$. In the other case, the reconstruction loss is prioritized. The balance between these two terms using the β parameter is investigated in Higgins et al. 2017 and shown to outperform a traditional VAE, which is the same as a $\beta - VAE$ where $\beta = 1$.

3 Data

The dataset used for exploration is the widely used Modified National Institute of Standards and Technology (MNIST) dataset containing 28x28 pixel images of handwritten numbers from "0" to "9". The dataset contains 60,000 training images and 10,000 test images.

4 Results and Discussion

4.1 Visualizing the Latent Space

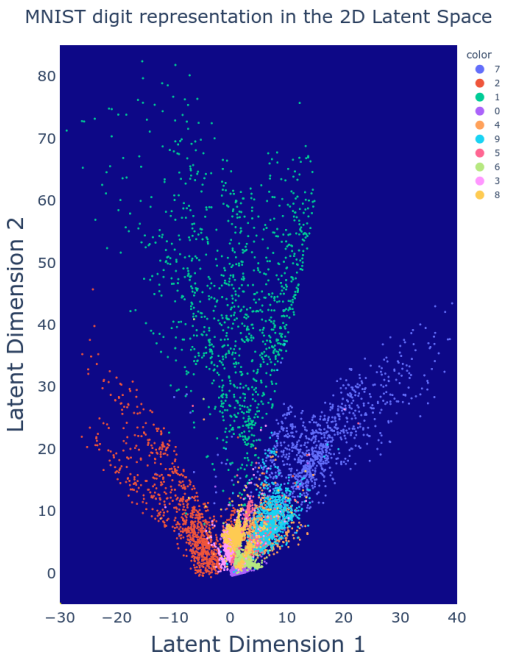

Figure 2: Latent Space - Neural Network

To better understand the latent space and the variational autoencoder, we first built a neural network encoding, thereby reducing the image dimensions from 784 to two and then decoding and reproducing an image of the original size. Decreasing the dimensions to two allows us to plot points on the latent space by allocating them using one node as a value for the x-axis and the other as a value for the y-axis. Note that since no sampling is involved, we train the network with the loss function solely defined by the reconstruction error. Therefore, as soon as the network is trained, all values in the latent space are entirely determined by a combination of input values and the weights within the network. We created figure 2 using the test set of the MNIST data containing 10,000 images and plotting their corresponding values in the latent space. As we can see, there is some level of clustering in the sense that similar digit values are located closer to each other. It is notable that points are not densely distributed but rather spread out with a high variance. Mainly the latent space corresponding to values one, five, and seven is very large compared to the other digits, for which the latent values are closer to the origin. One may assume that this distribution follows some general pattern. But, it is crucial to note that training the network again may result in an entirely different distribution of z, as it also depends on the random initialization of the neural network's weights. Using this observation as a baseline, we produced similar plots for the variational autoencoder. The main difference is that now, the x and y values of the latent variables are not entirely determined by the neural network's weights. Instead, it is a combination of weights of the network parameterizing a normal distribution and a random draw from this distribution using the parameters and leveraging the reparametrization trick. Hence, even with a fully trained neural network, it is impossible to pinpoint the exact latent value for a given input in a variational autoencoder. This is the case for all inputs to the autoencoder and is particularly important in the training phase of our model as it hinders the neural network from overfitting to individual data points. Even for a fully trained VAE, inputting multiple times the same picture results in slightly different latent space

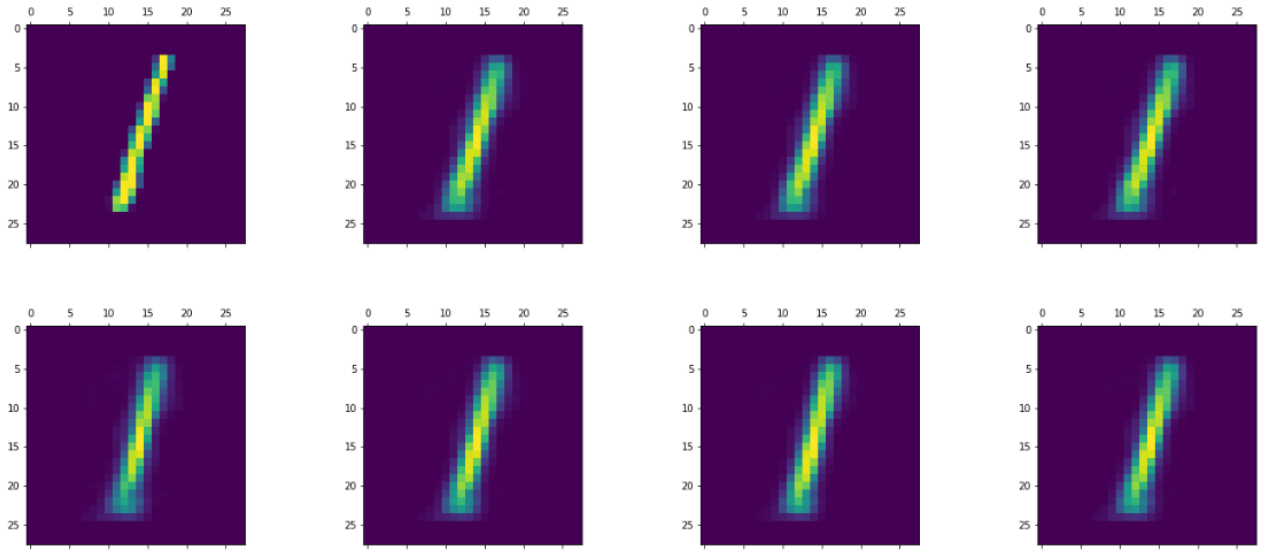

Figure 3: Digits created with a trained Variational Autoencoder using the same input digit (top left) for each output to compare variation in the outputs induced by the sampling

values and outputs, as shown in figure 3. The digit on the top left is the original input image, while all other images are encoded and then decoded versions of this same image using a fully trained variational autoencoder. At first sight, all output images created by the VAE appear to be the same, but if we look closer, we can observe some variation induced by the sampling in the latent space. This helps to prevent overfitting as to reduce reconstruction loss, the VAE needs to recreate the input well, not just for one specific value of z but for all values within the distribution from which we draw z. Note further that the distributions of points with similar characteristics overlap, creating a smooth distribution of z.

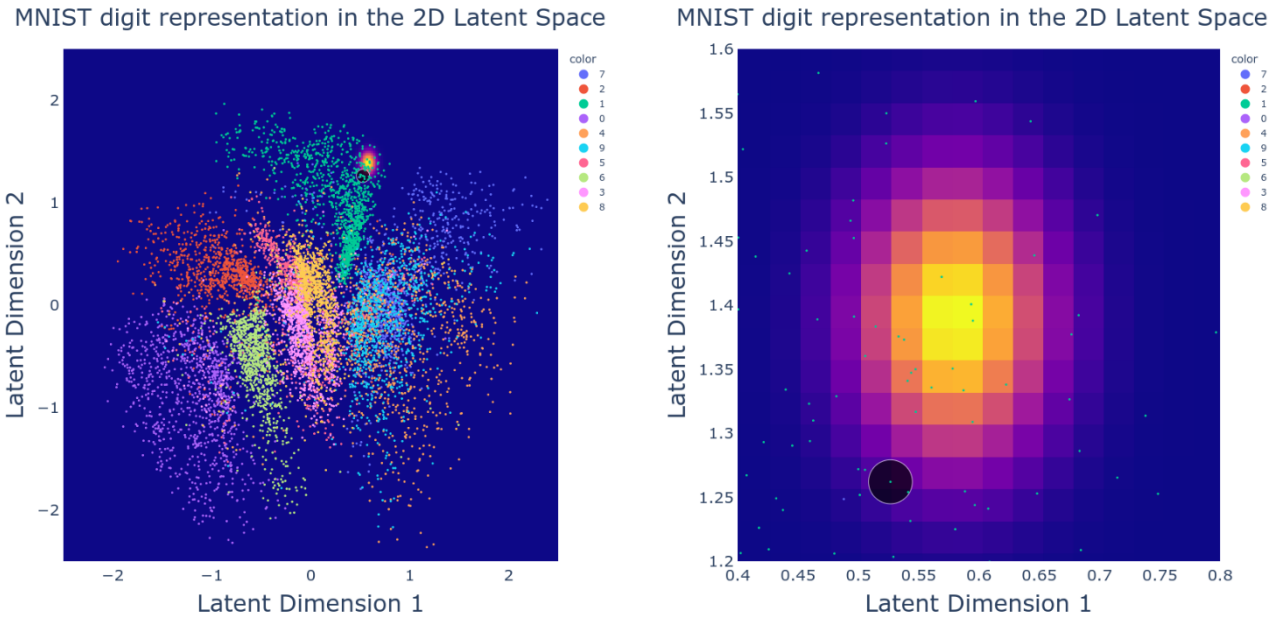

Figure 4: Latent space - Variational Autoencoder. Left: the distribution of test data on the latent space. Right: the distribution of z given one input image and the according draw from this distribution

To get a better understanding, the raster in the background of the right side of figure 4 shows the density of the normal distribution parameterized by the neural network part of our encoder for some input. The black point is a draw from this distribution, giving us two values for z associated with the input. Note that we get two values for z as we defined our latent space as two-dimensional and drew from a bivariate normal distribution. Besides including random noise to make the neural network more robust to overfitting and the latent space smoother, the sampling serves another purpose. It allows us to complement our model's recreation

loss term with the KL divergence. As described in the methodology part, the KL divergence allows penalizing the distribution of each image if it differs from the prior distribution of z. As distributions are parameterized by the neural network, including the KL divergence in the loss function, trains the weights so that the encoder outputs mean and variance terms closer to zero and one (or other parameters, depending on our specification for the prior distribution of z). On the left side of figure 4 we can observe the entire latent space of the VAE visualized using the 10,000 test examples again. Comparing this plot to the latent space defined by the neural network in figure 2, we see that now the points are distributed closer to zero, almost assembling a standard normal distribution with a minor variance. Having the latent space similarly distributed as the defined prior (here: standard normal) allows us to sample from an according bivariate distribution and plug the sampled values as input in the decoder part of our VAE. Two terms aiming at different outcomes jointly in the loss function create tension. On one side, the VAE has to achieve good recreation of the input images, on the other, its latent space has to approximate a predefined distribution. We can imagine these parts of the loss function competing during the training.

4.2 Effects of β in β-VAEs

As previously discussed, the architecture of variational autoencoders offers various possibilities for modification, including the adjustment of the loss function and the choice of different probability distributions. The efficiency and results of these modifications depends on the specific data and desired outcomes.

To demonstrate the potential impact of modifying the loss function, we conducted an experiment where we adjusted the weight of the KL divergence term by multiplying it with an arbitrary beta value. Theoretically, increasing the beta value places greater emphasis on ensuring that the probability distributions resemble a chosen reference distribution (e.g., $\mathcal{N}(0, 1)$), while also maximizing the utilization of available space.

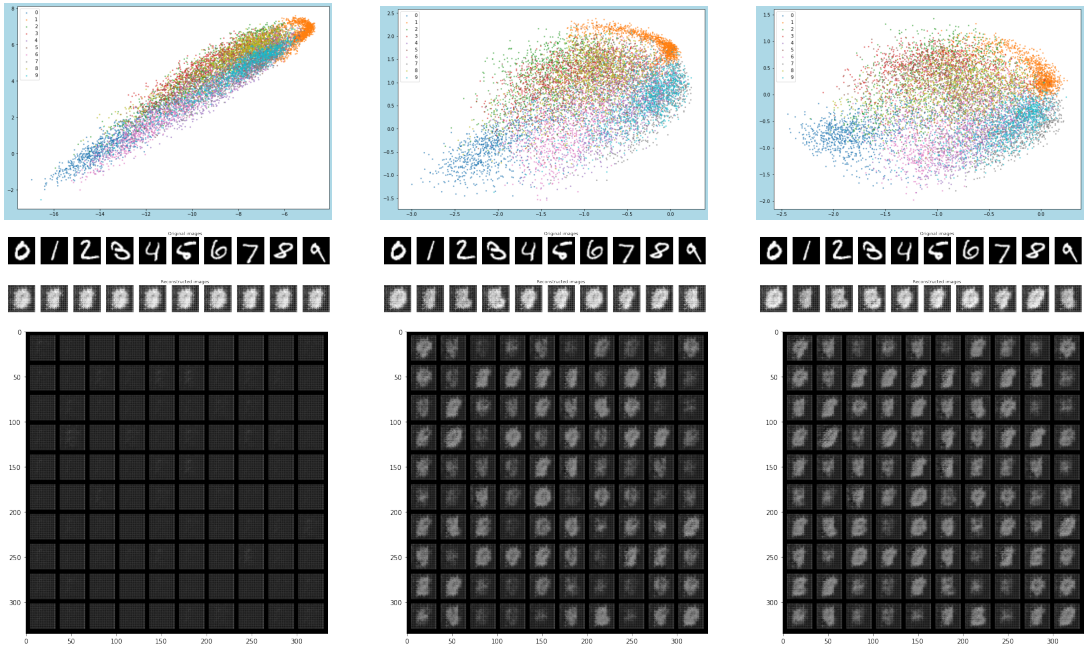

Figure 5: VAE's representations with different β values of 0, 1, and 5 respectively for epoch 1. 1st row: Latent space: 2 row reconstructed images, 3 row: sampling from latent space.

There are several interesting observations that can be inferred from figure 5. Firstly, the scatter plots of latent dimension distributions with higher β values appear to look more "normal-like", exhibiting a rounder structure and more efficient allocation of space provided. While the reconstructed images for all β values are not yet entirely clear and no essential differences are observed between them, significant differences can be observed in the last row of images. In these images, random sampling was performed from the latent spaces defined by VAE's, with latent vectors generated from $\mathcal{N}(0, 1)$ distributions. For a β value of 0 (no KL term was used), the images sampled from the latent space had little meaning and bore little resemblance to the images in our data set. However, images sampled from the latent space with higher β values began to look more like our initial data set images.

As we start to iterate over multiple epoch some other interesting observations can be made.

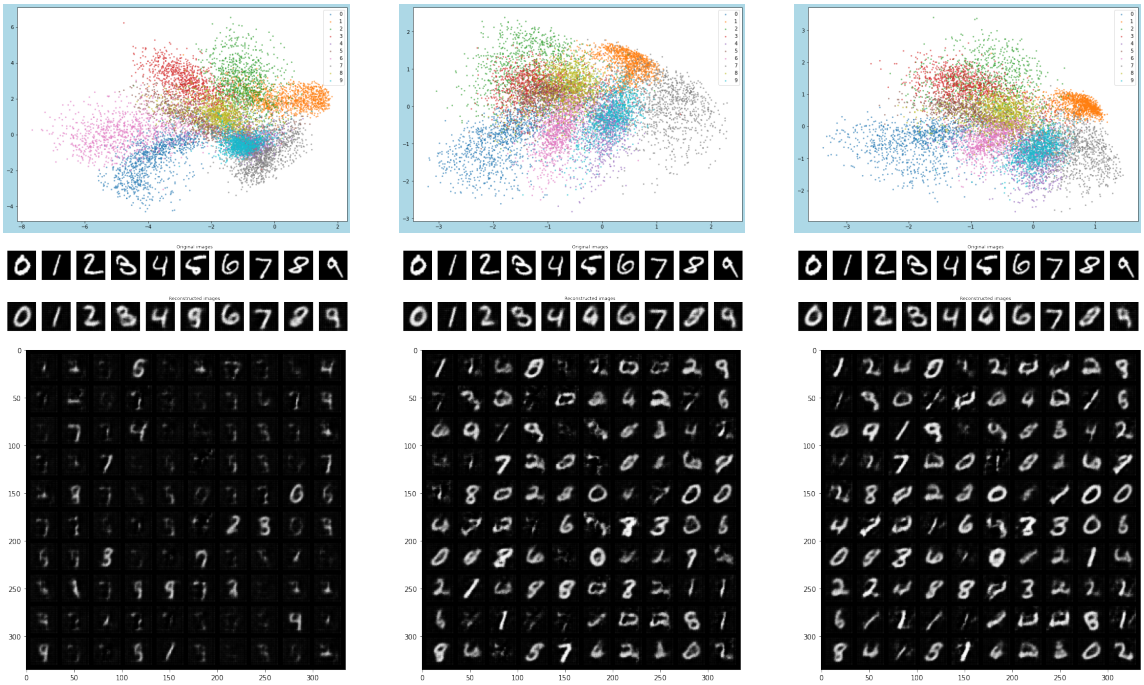

Figure 6: VAE's representations with different β values of 0, 1, and 5 respectively for epoch 5.

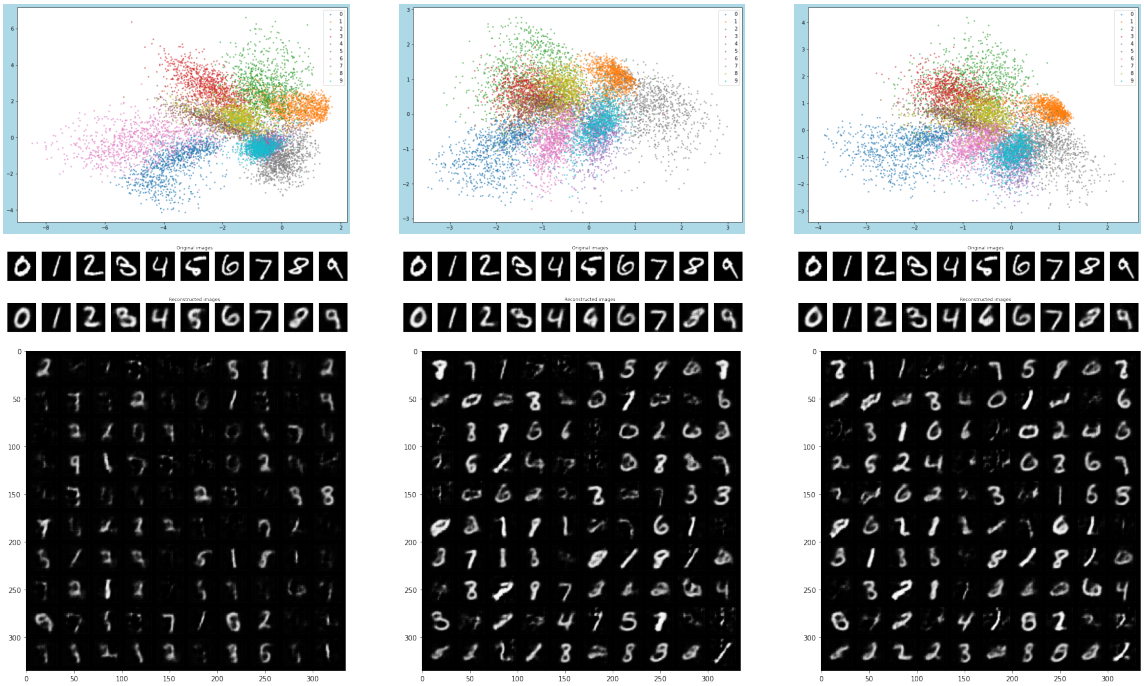

Figure 7: VAE's representations with different β values of 0, 1, and 5 respectively for epoch 10

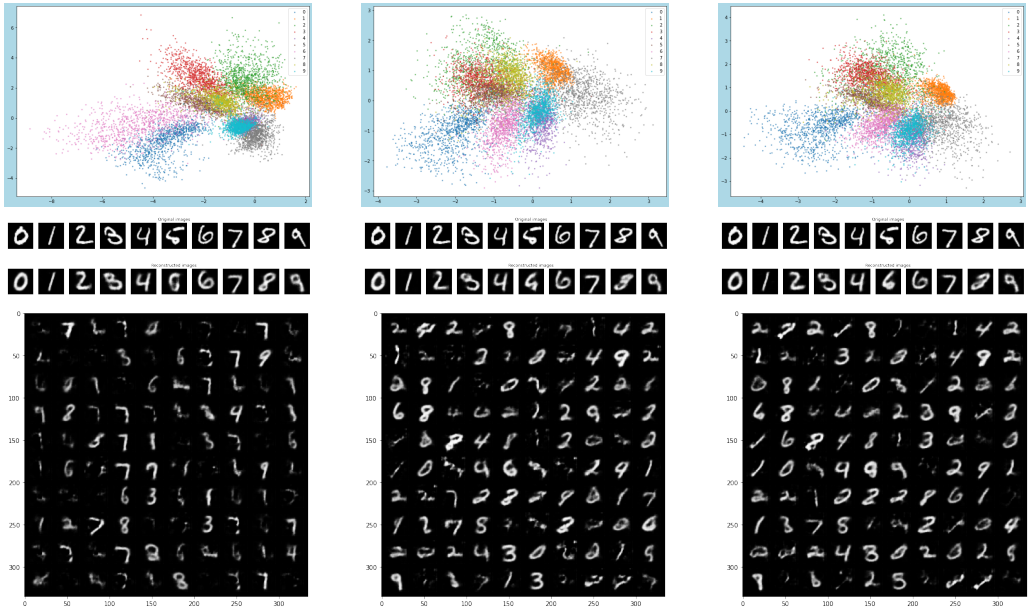

Figure 8: VAE's representations with different β values of 0, 1, and 5 respectively for epoch 20.

With each epoch, for figures 6 to 8 we can observe an increasingly better separation and clustering of our classes in the scatter plots of latent space. Even for high β values, the distributions no longer resemble $\mathcal{N}(0, 1)$ distributions entirely. Instead, they exhibit a more condensed and round shape while occupying space more efficiently, resulting in similar class distributions. After epoch 5, the reconstructed images do not change significantly, and we begin to achieve almost perfect reconstruction for all β values.However, setting the β value too high can damage the quality of the reconstructed images as the algorithm will focus too much on minimizing the KL loss and sacrifice the reconstruction term. For the experiment, β values of 200-500 were attempted, but even after 20 epochs, the quality of reconstructed images was inadequate and significantly worse compared to VAEs with small β values.

The most noticeable differences between the images using β values of 0 and higher β values are observed in the images sampled from the latent space. Upon examining the outputs for epoch 20, we can observe that the images sampled from the latent spaces defined by the VAE with β=0 are almost unrecognizable. They are difficult to identify and resemble a collection of common features shared between most of the digits, such as vertical sticks. Images that were sampled from the latent space with positive β values are on the other hand almost as good as the reconstructed ones. Hence, we can see that now we do have an opportunity to sample from the latent space and navigate in it way more comfortable.

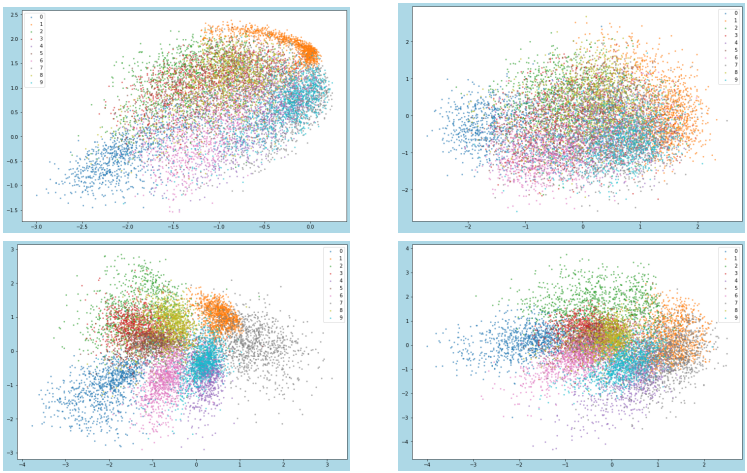

Figure 9: VAE's representations with different β values of 1 and 50 and for epoch 1 and 20 respectively. As we can see, at first distributions look more line N(0,1) with very bad separation between clusters

As we can see from scatter plots on figure 9: once we have high values of β our KL term increases rapidly and our distributions (even after many iterations) become less distinguishable and almost Normal.

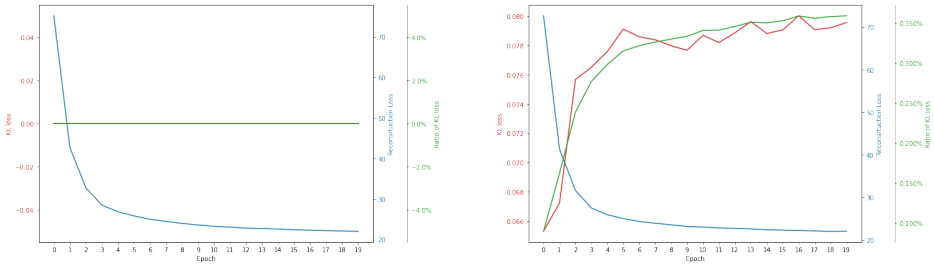

Figure 10: VAE's losses with different β values of 0 and 1 respectively

.

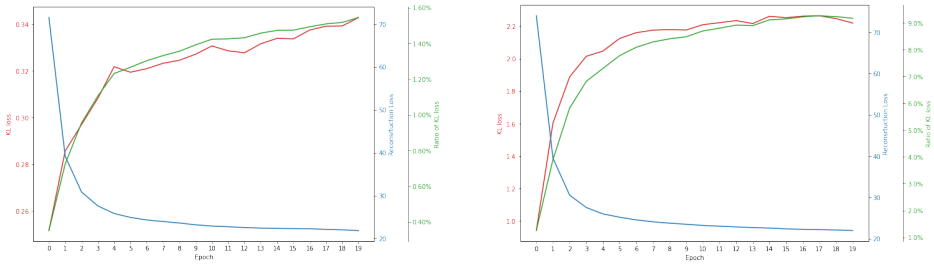

Figure 11: VAE's losses with different β values of 5 and 50 respectively

.

The line charts demonstrated on figures 10 and 11 illustrate the loss over multiple epochs for β values of 0, 1, 5, and 50. It can be observed that even with very high multiplicative β values, the share of KL loss is relatively low at the beginning (below 1 percent). After multiple iterations, the reconstruction loss drops significantly, while the KL loss increases in both absolute and relative terms.

This phenomenon can be explained by the fact that as our model becomes more familiar with the structure of our variables after each epoch, minimizing the KL term becomes less important. Initially, the KL term was easier to minimize (if a Normal distribution is drawn for each class, the KL loss will be 0). However, over time, our model is willing to sacrifice the KL loss by making the distribution less normal-like in order to gain more from having a lower reconstruction loss.

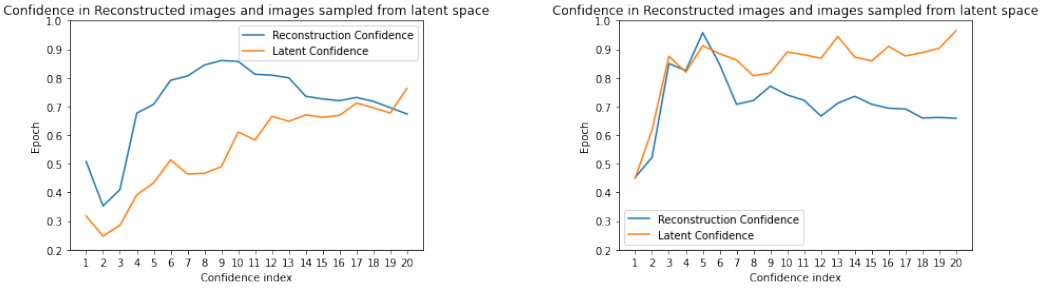

Figure 12: VAE's confidence plots with different β values of 0 and 1 respectively

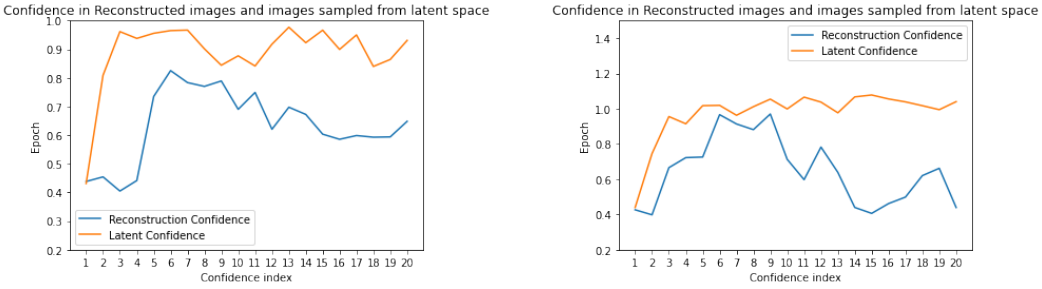

Figure 13: VAE's confidence plots with different β values of 5 and 50 respectively

A given stage neural network with multiple layers was trained on initial data to recognize digits. After 10 epochs, it was able to recognize digits quite well (with an accuracy of over 85 percentage on the testing dataset). Subsequently, the network was run on the testing dataset and the probability (confidence) of the best guess for each prediction was saved. These confidences were then averaged for the testing dataset to determine a baseline guess confidence of 0.71.

The same network was then used to estimate the probability (confidence) of the best guess for each digit drawn from our $\mathcal{N}(0, 1)$ latent spaces at different β values and for images recreated by the VAE. The confidence of the guess for each sampled number was then averaged and divided by the baseline confidence (average confidence of the best guess).

$$Confidence\ of\ Latent\ Sampling = \frac{Average\ confidence\ of\ the\ sample}{Baseline} \tag{13}$$

The figures 12 and 13 show that reconstructed images using β value of 0 are slightly inferior to the original images, and after sufficient iterations, images sampled from the latent space are nearly indistinguishable from the reconstructed ones, but however the are still not as good as the original ones.

Increasing the value of β results in better quality images sampled from the latent space, surpassing the quality of the reconstructed ones. Surprisingly, if we set β to very high values, the sampled images are even more recognizable by the network than the original ones, although they may not necessarily look better in appearance or more realistic, but just simply more structured and recognizable by the network.

Hence, from what we seen the main benefit of using KL-divergence term was sampling from the latent space itself. We do know our distributions look somewhat Normal like and most importantly each class occupies specific partitions of the latent space with some overlays with other classes.

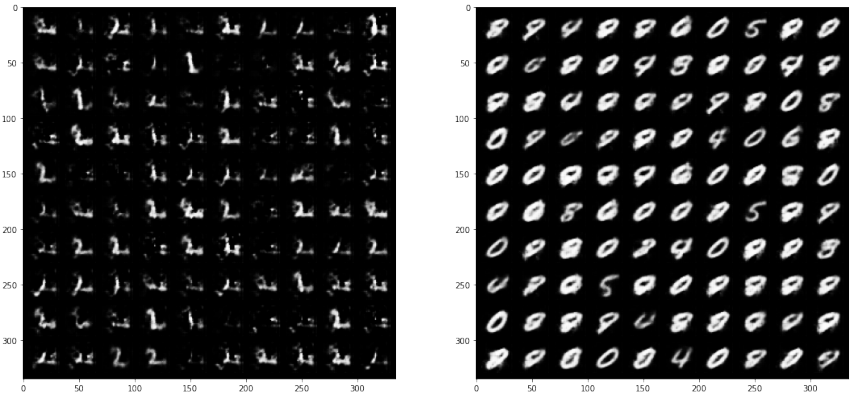

Figure 14: Sample from Latent spaces with β 0 and 5, from Gaussian distribution: $\mathcal{N}(1, 0.5)$

Figure 14 on the other hand clearly indicate that when we travel around the latent space (by creating vectors from a normal distribution with mean=1 and standard deviation=0.5) and sample from it, the samples obtained with $\beta = 0$ are mostly nonsensical, while the samples from the space defined by the KL term with $\beta = 5$ are meaningful and have a lot of similarities with each other. This implies that by zooming in on specific regions of the latent space, we can obtain samples with desired features or extract common features between two classes by drawing samples from the borders of their distributions where they interact.

To summarize, there is much room for further development of VAEs architecture. Techniques for navigating, sampling, and understanding the latent space can be improved, and different distributions can be explored and applied to different tasks. For instance, early attempts were made to create VAEs with uniform distributions where the mean and a-b boundaries move during the sampling process, yielding interesting insights, especially regarding the extraction of common features. Additionally, the Laplace distribution was also explored, but it produced almost the same results as $\mathcal{N}(0, 1)$.

5 Conclusion

In this project, we gained a deeper understanding of variational autoencoders. In particular, we analyzed the distribution of the latent space and compared different autoencoders and their performance for image recreation and generation. As expected, the latent space in a neural network-based autoencoder was distributed in a manner that did not follow a tractable distribution. With the introduction of the sampling and the KL term in the loss function, we showed that the variational autoencoder overcomes this issue by using variational inference to approximate a given prior distribution with the latent space. We showed that the KL term is important because it allows for image generation by sampling from the latent space. Including a scaler beta to increase the weight of the KL divergence in the loss function to values larger than one further amplifies the ability to generate meaningful output. Beta can be treated as a hyperparameter that has to be tuned depending on the application of the VAE, thereby optimizing its performance. Our results show that this increased quality of generated output by increasing beta comes at the cost of worse performance in correctly recreating input data.

References

Blei, David M. (2012). "Probabilistic Topic Models". In: *Commun. ACM* 55.4, 77–84. ISSN: 0001-0782. DOI: 10.1145/2133806.2133826. URL: https://doi.org/10.1145/2133806.2133826.

Kingma, Diederik P and Max Welling (2013). "Auto-Encoding Variational Bayes". In: DOI: 10.48550/ARXIV.1312.6114. URL: https://arxiv.org/abs/1312.6114.

Higgins, Irina et al. (2017). "β-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework". In.

Kingma, Diederik P. and Max Welling (2019). "An Introduction to Variational Autoencoders". In: *CoRR* abs/1906.02691. arXiv: 1906.02691. URL: http://arxiv.org/abs/1906.02691.

Dobilas, Saul (2022). *VAE & Variational Autoencoders: How to Employ Neural Networks to Generate New Images*. https://towardsdatascience.com/vae-variational-autoencoders-how-to-employ-neural-networks-to-generate-new-images-bdeb216ed2c0. [Accessed: March 1, 2023].