Small Project in DD2434 Advanced Machine Learning

Re-Auto-Encoding Variational Bayes

Abstract

Since presented in 2014, Variational Auto-Encoders have become one of the most popular approaches to regression, classification and even showing great performance as generative models. In this report, we present our re-implementation of Variational Auto-Encoders based on the original paper. The implementation of the evidence lower bound, the marginal likelihood are discussed, and Variational Auto-Encoders as generative model are also considered. For our experiments, we use the MNIST and Frey Face data sets. While our implementation follows that presented in the original paper, in a few aspect it slightly deviates, for instance, in the realization of the marginal likelihood and the choice of the optimizer. All in all, we achieve and present almost identical results to that of the original paper.

Diogo Pinheiro **Jakob Lindén** **Márk Csizmadia** **Patrick Jonsson**

KTH Royal Institute of Technology

January 15, 2021

Introduction

Auto-Encoders are artificial neural network-based models consisting of two separate networks called the encoder and the decoder. The encoder network is trained to learn an encoding of the data in an unobserved, fixed, latent space. The decoder is trained to reconstruct the data from the fixed latent representation. An efficient encoding of the data in latent space has smaller dimensionality with virtually no loss of information when compared to the original data.

In [1], the Auto-Encoding VB (AEVB) algorithm is introduced and demonstrated for efficient inference and learning in approximate inference models using the Stochastic Gradient VB (SGVB) estimator. When the AEVB algorithm is used to jointly optimize the parameters of neural network-based encoders and decoders, the encoder network becomes the probabilistic approximation to the intractable posterior over the latent space. Analogously, the probabilistic decoder produces a distribution over the possible values of the input data given the encoded representation. The aforementioned model is termed Variational Auto-Encoder (VAE) in which the posterior of the latent variable is Gaussian. The encoder generates two vectors, one for the mean μ and one for the log of the variance $\log \sigma^2$ of the approximate posterior distribution. The latent representation is then constructed by sampling from the approximate distribution, after which the probabilistic decoder reconstructs the input data. Unlike in regular auto-encoders, there is stochastic element in VAEs as the encoder and the decoder are probabilistic and the latent variable is sampled from the posterior rather than being fixed. This helped VAEs prove to be useful algorithms in dimensionality reduction, and de-noising among other applications.

The aim of this report is to replicate the original paper written on VAEs titled *Auto-encoding Variational Bayes* [1] by Diederik P. Kingma and Max Welling. After re-implementing the VAE, first, the MNIST dataset and the Frey Face dataset is be used to train the model to reconstruct images. Then, the variational lower bound and the marginal likelihood of the trained model are visualized, and the trained model is used to reconstruct images.

Method

In implementing the VAE, we followed the explanation of the AEVB algorithm provided in the original paper [1] and used the Tensorflow machine learning framework and the Keras deep learning API.

The encoder and the decoder of a VAE were implemented as neural networks (NN) of type Multilayer Perceptron (MLP) with a single hidden layer. In the "Future Work" section the authors mention the use of convolutional neural networks (CNNs) for the encoder and the decoder architecture that could result in potentially better performance due to the capability of CNNs to learn hierarchical representations of visual features. Nevertheless, we implemented the approach presented in the paper, that is, realizing the encoder and the decoder as MLPs. The encoder and the decoder are visualized in Figure 1. The goal of the encoder is to transform the input data into a vector of means and a vector of standard deviation (log of variance) that characterize the approximate variational posterior distribution. To paraphrase, the probabilistic encoder is the approximation $q_\phi(\mathbf{z}|\mathbf{x})$ to the intractable, true posterior $p_\phi(\mathbf{z}|\mathbf{x})$ over the latent space. The decoder is the generative model $p_\theta(\mathbf{x}|\mathbf{z})$ that given a sampled latent vector, produces a probability distribution over the possible value of the input data. The encoder is parameterized by the set of variational parameters ϕ and the decoder network is parameterized by the set of generative parameters θ. All NN layers were implemented as densely-connected layers, via the *Keras* functional API. While the number of units in the input and the output layers depend on the length of the flattened image data, the number of nodes in the hidden layer vary depending on the data being modelled. For example, the hidden layer has 500 units in the case of the MNIST data set and 200 units in the case of the Frey Face data set. The non-linearities of the fully-connected layers were implemented with the hyperbolic tangent (tanh) activation function. While the last layer of the encoder does not utilize any non-linearity to generate the parameters of the approximate variational posterior, the activation of the last layer of the decoder is defined

by the probability distribution of the decoder, which in turn, is defined by the data being modelled.

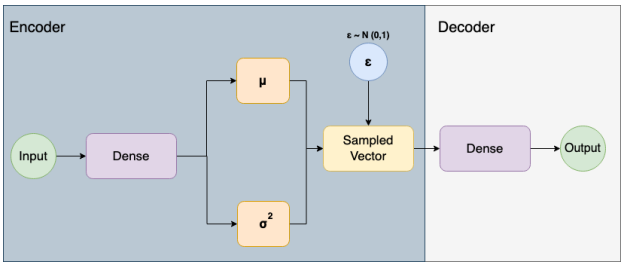


Figure 1: Representation of the auto-encoder neural network.

The overall VAE loss function comprises two terms - the log-likelihood of the generative decoder's output distribution, which is akin to the reconstruction loss (RLO), and the Kullback-Leibler (KL) Divergence. While the reconstruction loss is a measure of the pixel to pixel difference in the input image and its reconstruction, the KL divergence is a measure of the dissimilarity between the encoder's variational posterior approximation and the true posterior. The KL divergence ensures that the approximate posterior distribution learned by the encoder is sufficiently close to the true posterior. The aforementioned general loss function, which we refer to as variational *lower bound* (ELBO), is presented in [1] as

$$\mathcal{L}\left(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}\right) = -D_{KL}\left(q_{\phi}\left(\mathbf{z} \mid \mathbf{x}^{(i)}\right) \| p_{\boldsymbol{\theta}}(\mathbf{z})\right) + E_{q_{\phi}\left(\mathbf{z}|\mathbf{x}^{(i)}\right)}\left[\log p_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)} \mid \mathbf{z}\right)\right] \tag{1}$$

To be able to differentiate the expectation w.r.t. ϕ, the second right-hand side term is approximated by its Monte-Carlo estimate. In turn, the backpropagation algorithm can be employed in jointly optimizing the variational and the generative weights via the differentiable transformation $g_{\phi}(\boldsymbol{\epsilon}|\mathbf{x})$ that is used to reparameterize the latent random variable $\mathbf{z}$ with the use of an (auxiliary) noise variable $\boldsymbol{\epsilon}$ such that $z = \mu + \sigma^2 * \epsilon$, where $\sigma^2 = e^{log(\sigma^2)} \in \mathcal{R}^{+}$, as described in equation 10 of [1]. This is called the *reparametrization trick* and the result of this operation is usually referred as the "sampled vector". The sampled latent vector serves as the input to the decoder. The *Keras* deep learning framework provides a custom layer called *Lambda* that comes with the differentiability property of the reparameterization transformation and can be passed an arbitrary function. In our implementation of the reparameterization trick, the `get_latent` function is passed into the *Lambda* layer, and it receives the parameters of the approximate variational posterior and outputs the sampled vector $\mathbf{z}$.

The general ELBO in the AEVB algorithm shown in Equation 1 is used to derive the ELBO specific to VAEs with Gaussian encoder. The expectation of the marginal log-likelihood $E_{q_{\phi}\left(\mathbf{z}|\mathbf{x}^{(i)}\right)}\left[\log p_{\boldsymbol{\theta}}\left(\mathbf{x}^{(i)} \mid \mathbf{z}\right)\right)$, a measure of the RLO, may be approximated as $\int log(p_{\theta}(x_i|z))q_{\theta}(z|x_i).dz$. Applying Monte-Carlo sampling to this equation yields $\frac{1}{L}\Sigma_{z^{i,l}} log(p_{\theta}((x_i|z^{i,l}))$. Then, letting the variational approximate posterior be a multivariate Gaussian with diagonal covariance, that is, $\log\, q_{\phi}(\mathbf{z}|\mathbf{x}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\sigma}\mathbf{I})$, the ELBO becomes:

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

In practice, by setting $L = 1$, that is, sampling one value from the approximate latent posterior, we have the ELBO estimator of the VAE and the RLO becomes $\log\left(p_{\theta}\left(x_i \mid z^{i}\right)\right)$ [2].

While the Frey Face data set may be interpreted as continuous data, meaning that the grey-scale pixel values take on the full range of 0-255, the MNIST data set may be interpreted as binary data in a sense that the grey-scale pixels take on the extremes of the range. With this in mind, the log-likelihood term in the ELBO becomes different depending on which data set is being modelled. In the case of the MNIST data set, the decoder distribution is Bernoulli and the last layer of the decoder therefore has sigmoid non-linearity. In this case, the log-likelihood, which is a measure of the reconstruction loss, becomes the negative

binary cross-entropy[1]:

$$\begin{aligned}\log\left(p_\theta\left(x_i \mid z^i\right)\right) &= \log\prod_{j=1}^{D} p_\theta\left(x_{i,j} \mid z^i\right) = \sum_{j=1}^{D}\log p_\theta\left(x_{i,j} \mid z^i\right) \quad \text{, where } p_\theta\left(x_{i,j} \mid z^i\right) \sim \text{Bernoulli}(p_i) \\ &= \sum_{j=1}^{D}\log p_{i,j}^{x_{i,j}}\left(1-p_{i,j}\right)^{1-x_{i,j}} \quad \text{, where } p_{i,j} \text{ is the network output} \\ &= \sum_{j=1}^{D} x_{i,j}\log p_{i,j} + (1-x_{i,j})\log\left(1-p_{i,j}\right) \quad \text{, which represents the cross-entropy}\end{aligned} \tag{3}$$

On the contrary, in the case of the Frey Face data set, the decoder distribution is Gaussian, so the log-likelihood becomes:

$$\begin{aligned}\log\left(p_\theta\left(x_i \mid z^i\right)\right) &= \log\prod_{j=1}^{D} p_\theta\left(x_{i,j} \mid z^i\right) = \sum_{j=1}^{D}\log p_\theta\left(x_{i,j} \mid z^i\right) \quad \text{, where } p_\theta\left(x_{i,j} \mid z^i\right) \sim \mathcal{N}(x;\mu,\sigma) \\ &= -\frac{D}{2}\log(\sigma^2) - \frac{D}{2}\log(2\pi) - \frac{1}{2\sigma^2}\sum_{j=1}^{D}(x_{i,j}-\mu)^2\end{aligned} \tag{4}$$

This difference between the decoder distributions gives rise to two types of VAE that we implemented. One of them is the VAE with the Gaussian encoder and Bernoulli decoder and the other one is the fully Gaussian VAE, which were respectively used to model the MNIST and the Frey Face data set.

In the paper, the ELBO is maximized using a gradient ascent technique in the joint optimization of the learnable parameters of the encoder and the decoder. Moreover, originally the optimization is implemented using the *AdaGrad* algorithm with a weight decay function but, due to new developments, our more computationally efficient approach uses the *Adam* optimizer instead, which incorporates weight decays. As we utilized the built-in *Keras* optimization scheme that comes with gradient descent techniques, we actually minimized the negative ELBO, $-ELBO = KL - RLO$, that is equivalent to maximizing the positive ELBO. In the case of the MNIST data set (Gaussian encoder and Bernoulli decoder), it is $-ELBO = KL - RLO = KL + BCE$ where BCE refers to the binary cross-entropy (see derivations shown in Equation 3 [3]. In the case of the Frey Face data set, it is $-ELBO = KL - RLO$ where RLO is the log-likelihood of the normal distribution of the generative decoder.

This relation between RLO and BCE for discrete data sets, allows us to also assume a connection between the RLO and the marginal likelihood, which is supported by Equation 16 in [4]. According to [2], a Monte-Carlo estimator for the marginal likelihood is approximately the second term on the right-hand side of Equation 2, where $L \to \infty$ leads to the actual marginal. The most efficient way to compute this is by implementing the importance sampling algorithm discussed in [5], that essentially follows the following steps: a) Sample latent vectors for all datapoints of the batch; b) Estimate $p(x|z)$ for each sample; c) Compute the average $p(x|z)$ for the entire batch using the Monte-Carlo estimator. Using a custom *Keras* loss function, this whole process ends up being significantly more efficient because it automatically iterates through all samples and returns the mean value of the loss. Moreover, with the previously explained relation between RLO and BCE we end up simply applying the latter's formula as follows:

$$BCE = log\frac{1}{N}\Sigma_{i=1}^{N} latent\_vector * log(output) + (1 - latent\_vector) * log(1 - output) \tag{5}$$

This is quite similar to the Monte-Carlo estimator as we sum all estimations of $p(x|z)$ for each latent vector and then return the log of the mean value. Therefore, in subsequent respective experiments, we assume the marginal likelihood as the BCE.

[1] `https://github.com/hwalsuklee/tensorflow-generative-model-collections/issues/13`

Results

First and foremost, we aim to reproduce the variational lower bound (ELBO) experiments presented in the original paper. When applying the VAE to the two different data sets, we considered the differences mentioned in the Method section, that is, MNIST is discrete and Frey Face is continuous data. Therefore, to MNIST we apply a VAE with Gaussian encoder and Bernoulli decoder, and to the Frey Face we apply a fully Gaussian VAE (i.e. Gaussian encoder and Gaussian decoder). Furthermore, in line with the original paper, we use 500 hidden units for MNIST and 200 for Frey Face.

In training the VAE, mini-batches of size $M = 100$ were used for both data sets, and the learning rate of the Adam optimizer was set to 0.001. This learning rate was not specifically stated in [1], however empirically trying the set of proposed learning rates $\{0.01, 0.02, 0.1\}$ did not result in similar results to that of the original paper, while using 0.001 did. The low learning rate used to replicate the results in [1] indicates that the authors may have found that it is more beneficial during the training to make smaller adjustments to the output when the model is wrong, rather than large ones which may result in over-correcting and an unstable learning process. The downside of this would be that since the learning is slower with a low learning rate, it requires more training samples to train the model, which may be why their sample size is large.

In Figure 2, our satisfyingly accurate reproductions of the original paper's variational lower bound figures are shown for both MNIST and Frey Face for different dimensionality of the latent space, N_z. When fitting the models, we observed that our model convergences faster than presented in the original paper. In lack of adequate computing resources and since the algorithm demonstrated comparable performance, we evaluated the model performance for 10^7 samples rather than for 10^8 training samples.

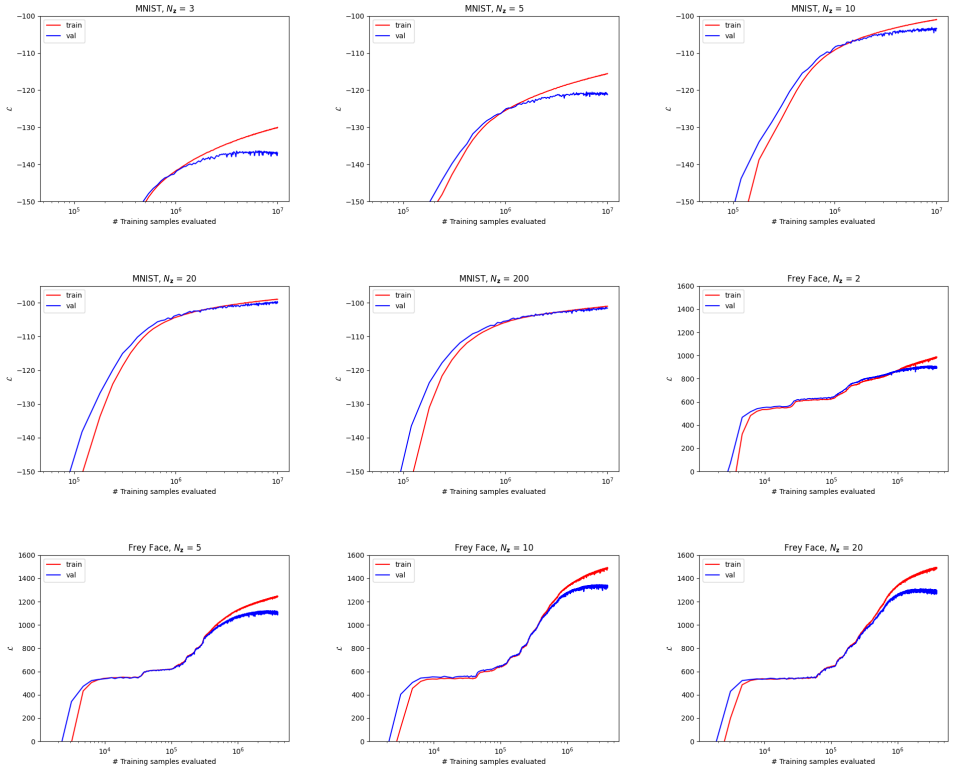
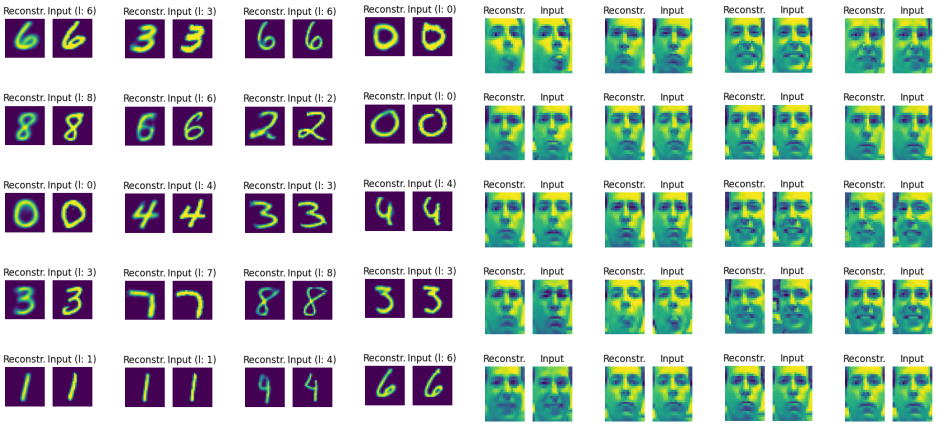


Figure 2: Variational lower bound comparison between latent variables for the MNIST and Frey Face data set.

In Figure 3, the reconstructions of randomly selected images are visualized in the case of both data sets for different latent space dimensionalities. From left to right, the latent space dimensionality is $N_z = 2$, $N_z = 5$, $N_z = 10$ and $N_z = 20$, respectively. It is notable that for each data set, the accuracy of the reconstruction improves as we increase the number of latent dimensions.



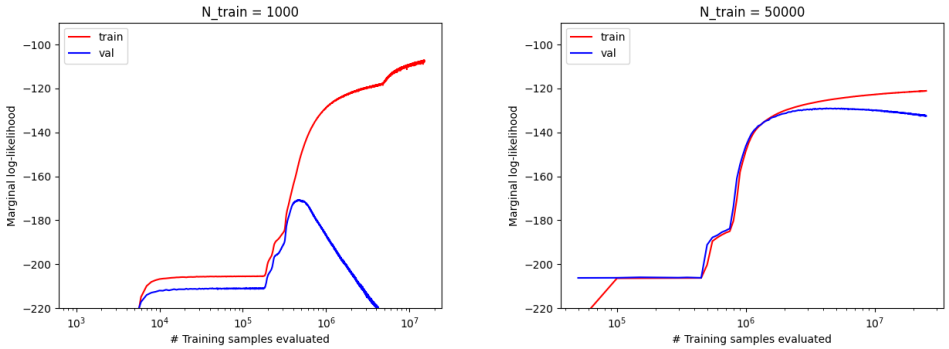
(a) 2 (b) 5 (c) 10 (d) 20 (e) 2 (f) 5 (g) 10 (h) 20

Figure 3: Reconstructed images for different number of latent dimensions

In the original paper, additionally, the marginal log-likelihood is estimated for the MNIST data set, using 100 hidden units and 3 latent variables. Our reproduction is shown in Figure 4 for a random partition of $N_{train} = 1000$ and $N_{train} = 50000$ samples. It is notable that with the larger data set partition, our implementation produced a similar result to that presented in the original paper. However, in the case of the smaller partition, we observe a difference in terms of the trend of the validation loss. In the original paper, the validation and training losses have a constant difference throughout all the training samples evaluated. In our implementation, nevertheless, the validation loss curve takes a deep turn at around -150 marginal log-likelihood, after which the continuing drop implies that the model overfits the smaller data set partition.

Having performed a series of experiments on the MNIST data set, we found that this tendency is demonstrated by both the ELBO and marginal curves, which indicates that for relatively small data sets, our VAE is prone to overfitting. On the contrary, in the case of experiments with the Frey Face data set, we did not observe the aforementioned tendency. In our view, a potential cause of this discrepancy could be the implementation of the BCE, a common factor between both cases. However, when compared to existing implementations, our implementation of the BCE does not appear to have any issue. As a result, we inferred that the issue could potentially lie in loss function optimization, which resulted in unsuccessful attempts to pinpoint the origin. Compared to the original paper, the overfitting was expected due to the use of an aggregated posterior to minimize the prior for KL in the ELBO [6], as it is visible in Figure 2. In our case the algorithm does not appear to fully converge, rather it reaches global minima and then starts to modestly deviate away from it.

In order to deal with the common overfitting, over-regularization and computationally expensive ways to approach priors with regards to the aggregated posterior, there have been some developments that apply a learned acceptance function to the basic technique of rejection sampling [6] or the usage of a mixture of Gaussian distributions, for example, in what has been labelled as VampPrior [7].



(a) (b)

Figure 4: Marginal likelihood comparison for (a) 1000 and (b) 50000 training points, using the MNIST data set.

When the latent space dimensionality is two, in Figure 5, we visualize the clustering of digits as a scatter-plot between the first and the second latent dimensions in the case of the MNIST data set, with the label of each observation being included. The clusters show that the more similar the digits are, the closer they are in the two dimensional latent space.

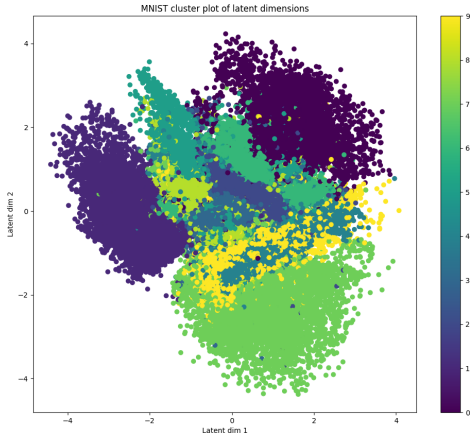
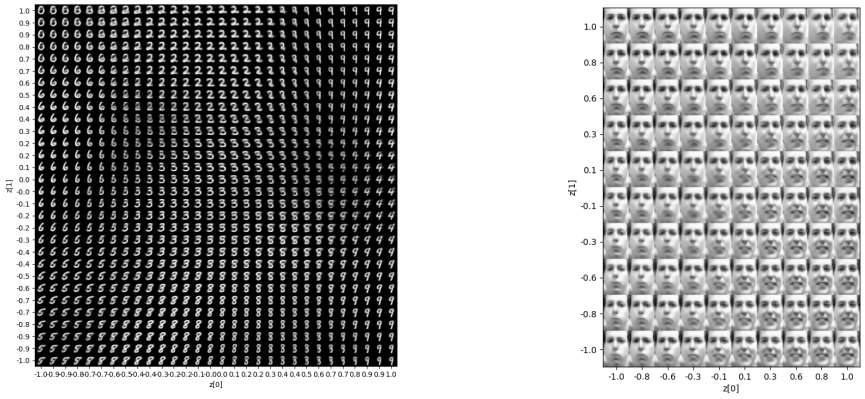


Figure 5: Clustering of the MNIST data set with a latent dimension of 2.

To visualize the reconstructed latent space manifold for our model, we sampled values from the latent space $\mathbf{z}$ and used the generative model, or decoder, $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$ with the parameters θ learned during training the model. In Figure 6, we visualize this two-dimensional manifold for both (a) MNIST and (b) Frey Face.



(a) (b)

Figure 6: Learned manifold of the MNIST (a) and Frey Face data set (b) with a latent dimension of 2.

Conclusion

In this report, our re-implementation of the VAE presented in [1] was discussed. Our implementation slightly differs from the original implementation, nevertheless, we successfully reproduced similar results. Firstly, we used the *Adam* optimizer instead of the *AdaGrad* optimizer, which simplified optimization as *Adam* comes with the weight decay functionality that *AdaGrad* lacks. Using *Keras* allowed us to slightly diverge from the algorithms presented in the original paper, which resulted in faster convergence but also in some problems with overfitting. We derived the RLO and showed its relation to the cross-entropy. With the use of importance sampling, we assumed the RLO to be the marginal likelihood.

Considerations regarding computational complexity may need to be taken into account when using VAEs, as it could become increasingly more complex to train the model depending on the type of data used. For example, RGB image data would result in increased dimensionality of the data input and output, which in turn, would require more computational resources. New different types of VAE have been presented since their first appearance, as well as improvements to the original version, mostly dealing with overfitting, over-regularization of the priors [6].

Bibliography

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

[2] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders, abs/1906.02691, 2019.

[3] Fangda Han.Derivation of elbo in vae. `https://medium.com/@hfdtsinghua/derivation-of-elbo-in-vae-25ad7991fdf7`. Accessed: 2021-01-07.

[4] David M. Blein. Variational inference. `https://www.cs.princeton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf`. Accessed: 2021-01-12.

[5] Brian Keng. Importance sampling and estimating marginal likelihood in variational autoencoders. `https://bjlkeng.github.io/posts/importance-sampling-and-estimating-marginal-likelihood-in-variational-autoencoders/`. Accessed: 2021-01-12.

[6] Matthias Bauer and Andriy Mnih. Resampled Priors for Variational Autoencoders. arXiv:1810.11428, 2019.

[7] Jakub M. Tomczak and Max Welling. VAE with a VampPrior. arXiv :1705.07120, 2017.