Understanding Variational Autoencoders and Their Applications

STA5077Z: Unsupervised Learning

Adam Mosam[1] and *Ndakondja* Shilenga[1,]

[1]Financial Innovation Hub, Department of Economics, University of Cape Town,South Africa

1 Introduction

In the realms of digital information, we are constantly flooded with images, texts, and audio. Diving deeper into the structure of this data we are exposed to; the underlying factor, or underlying process of the data can be much simpler and in much smaller dimensional spaces than the actual data we are looking at in comparison to their original size [1]. Many techniques in machine learning try to compress the dimensionality of data into a smaller space, one of the popular techniques used is Variational Autoencoders [2]. Variational Autoencoders (VAEs) are a class of generative models, designed to compress high dimensional data into low dimensional data which is reconstructed into the original data from the compressed representation [1, 2].

Autoencoders typically take some kind of input data (image, or a vector) anything with very high dimensionality. They run the data through a neural network and compress the data into a smaller representation using two components. The first component is known as the encoder and they could be fully connected layers or convolutional that take the input and compress it down into a smaller representation in a condensed latent space which has fewer dimensions than the input, the data is then reconstructed using fully connected layers [3]. The second component is the decoder and it computes the reconstruction loss with respect to the input, comparing differences in the output and input to create a loss function which can be used to train the network to compress data [3].

Variational autoencoders use a similar approach and process as general autoencoders [4] but with a twist. Instead of mapping an input to a fixed vector, in VAE, the input is mapped onto a distribution, with a sample of this distribution being fed to the decoder, allowing for data generation from the latent subspace.

VAEs are versatile, with the capability to be trained on diverse data such as images, text, and melodies [1]. According to an experiment conducted by An and Cho [3] for their study; Variational Autoencoder based Anomaly Detection using Reconstruction Probability, VAEs can discern the underlying structure or latent representation of data, facilitating numerous applications like pinpointing anomalies and reducing data dimensions. Furthermore, by generating fresh samples from the latent space distribution, VAEs are instrumental in tasks like creating images [2], crafting text [4], or composing music.

*e-mail: msmada002@myuct.ac.za

In this report, we discuss two uses and application areas of VAE, explain how VAEs work as well as an example of a VAE implementation in Python.

2 Uses and application areas of VAE

In this section, we discuss three uses of VAEs and provide application examples for each use.

2.1 Anomaly detection

Anomaly detection aims to identify unusual data points or abnormal patterns in datasets. VAEs, with their architecture, are usually used in this domain. Their ability to reconstruct data based on latent features allows them to detect anomalies through reconstruction probabilities and errors. That is, VAEs can identify the distance between data points and the latent space distribution, outlier datapoints scoring lower probabilities from the distribution, or use the reconstruction errors for each dataset [3]. In their experiments, An and Cho [3] compared the use of VAE based anomaly detection with reconstruction probability with other reconstruction-based methods, autoencoder, and PCA based methods of which VAEs outperform the other methods (please refer to the paper for more details on this).

2.1.1 Application to fraud detection

In the area of financial services and e-commerce, fraud detection is important. As mentioned above, VAEs offer methods to detect anomalies in transactional data which can be used to determine fraudulent activities [3].
The application of VAEs for credit card detection has been explored in academic literature. A noteworthy study by Sweers, Heskes, and Krijthe [5] investigated the efficacy of autoencoders and VAEs for this purpose. They found that while basic autoencoders were generally more effective in fraud detection, the stacked versions of VAEs surpassed regular autoencoders in performance. However, it's worth noting that their insights were largely based on a specific dataset, calling into question the broader applicability of their findings.
Another study by Ding et al., [6] introduced a fresh perspective by integrating a refined Variational Autoencoder Generative Adversarial Network (VAEGAN) oversampling technique with the XGBoost classifier. XGBoost is a classification model used as the baseline method for fraud detection in this paper. VAEGAN is a generative model that combines the advantages of VAE and Generative Adversarial Network (GAN) to learn the latent space representation and the distribution of the generated samples through two stage training [6]. While their primary focus was on credit card fraud, they posited that their model could be effective in other areas with class imbalances too. Notably, the XGBoost method showcased superiority over several traditional models, implying the potential strength of ensemble techniques in such contexts. Yet, it's important to mention that while their VAEGAN approach had commendable precision and F1 scores, other metrics like recall and AUC saw limited improvement.
When considering the broader use of VAEs in the realm of credit card fraud detection, both pieces of research emphasize the need for meticulous model adjustments. The distinct performances between traditional VAEs and their more advanced iterations like VAEGANs offer rich areas for future academic exploration, especially regarding metric optimization.

2.2 Data generation

VAEs have gained attention for their ability to generate data. By training on datasets, VAEs can create samples that resemble the training data [2]. Models can be trained on datasets such as CIFAR 10, MNIST [2] or complex ones like CelebA to create images of objects, handwritten digits, or even faces respectively [7].

2.2.1 Application to medical imaging

A noteworthy application of image generation is in the domain of medical imaging (Ibrahim, 2022). VAEs have the potential to undergo training using images such as X-rays or MRIs in order to produce medical images, as highlighted by recent studies (Ibrahim, 2022; Li et al., 2021). This application is particularly beneficial for research and training objectives, especially when real medical data accessibility is limited due to privacy concerns.
Li et al. [8] investigated the use of a invasive MRI technique called Arterial Spin Labeling (ASL) for diagnosing dementia diseases by measuring cerebral blood flow. They faced challenges due, to the lack of ASL data in existing dementia disease datasets. To overcome this the study introduced a VAE GAN architecture that effectively generated ASL images. This innovative approach showed improvements in diagnosing dementia highlighting the significant impact of VAEs in complementing missing data and enhancing diagnostic tools.
Similarly, Ibrahim [9] discussed the importance of Chest X rays (CXR) in detecting lung diseases. To overcome issues, with image artifacts; the study proposed a two-stage method; enhancing images using a combination of VAE and U Net followed by segmenting the lungs with ResUNet++. Combining VAE and UNet features enabled lung segmentation when working with limited image datasets. The performance metrics obtained through this approach demonstrated its superiority further emphasizing the role played by VAEs in imaging. Both studies underscore the capabilities of VAEs in medical imaging, from synthesizing missing data to enhancing image quality. While the methodologies used are different, the theme is evident: VAEs hold immense promise in advancing medical diagnostics, offering tools that are both versatile and effective in addressing the unique challenges of the medical domain.

3 Mathematical treatment

One of the fundamental advantages of variational autoencoders (VAEs) lies in their capability as a generative model. Specifically, they are designed to produce new, previously unseen samples drawn from a learned distribution. But how do VAEs differentiate themselves from their predecessor, the general autoencoder (GAE)? And why is the GAE not proficient in functioning as a generative model? To delve into this, we first examine the structure and operation of a GAE. [10]

Given a dataset $\mathbf{x}$ that exists in a p-dimensional space, denoted as $\mathbf{x} \in \mathbb{R}^p$, the objective is to compress this data into a latent representation $Z = e(\mathbf{x})$. This representation exists in a lower dimensional subspace d, where the function e signifies an encoder, which may be implemented using a neural network.

The decoder, denoted as d, attempts to reconstruct the original data from its compressed latent representation $\mathbf{z}$. The reconstructed data is represented as $\hat{\mathbf{x}} = d(\mathbf{z})$. The aim during training is to minimize the difference between the original data $\mathbf{x}$ and its reconstruction $\hat{\mathbf{x}}$. This difference is commonly measured using the mean squared error (MSE) loss function:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \tag{1}$$

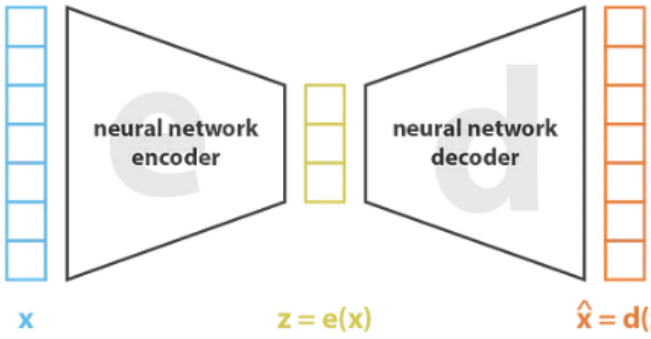

Figure 1. Illustration of a General Autoencoder (GAE). [11]

The model may be optimized using gradient descent- and backpropagation to minimize the error. This process is deterministic; for a given input **x**, the encoder will always produce the same latent representation **z**, and given **z**, the decoder will always produce the same reconstruction $\hat{\mathbf{x}}$. For this reason, this method excels at reconstructing data from a compressed representation, but struggles to generate new samples from the latent space. This limitation stems from the GAE's deterministic mapping, which translates points from the original space to specific points in the latent subspace. Consequently, interpolations between points in the latent subspace might not always produce consistent reconstructions in the data space, given that some aspects of the original data, such as the spatial positioning between points, could be lost during the compression into the latent subspace.

VAE's, introduced by Kingman [10], address the aforementioned issue by encoding the input **x** not as a single point, but as a distribution over the latent subspace. This distribution is typically chosen to be Gaussian, characterised by a mean, μ and standard deviation, σ. A sample **z**) may then be drawn from this Gaussian distribution and fed into the decoder to produce the reconstructed output, $\hat{\mathbf{x}}$. This probabilistic approach results in a continuous latent space which allows for generative capabilities [12].

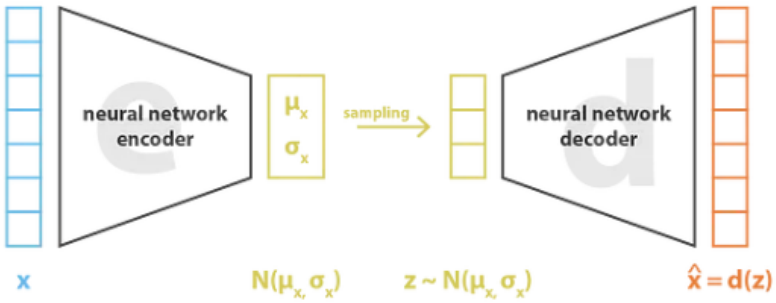

Figure 2. Illustration of a Variational Autoencoder (VAE). [11]

3.1 Variational inference

In Fig.(2), we define the encoder as $p(\mathbf{z}|\mathbf{x})$, which represents the latent variable given an input $\mathbf{x}$ and is commonly referred to as the true posterior. The decoder is determined by calculating $\hat{\mathbf{x}}$ given the sample from the latent variable $\mathbf{z}$, $p(\hat{\mathbf{x}}|\mathbf{z})$). Although the decoder is deterministic, the encoder is probabilistic, as discussed earlier. This poses challenges when trying to compute the posterior directly. To understand the difficulty in computing the posterior, we first employ Bayes Rule as outlined in Eq.(2)

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

While the numerator in Eq.(2) can be easily determined, computing the denominator, $p(\mathbf{x})$, involves integrating over the multidimensional latent subspace:

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z}) \cdot p(\mathbf{z})\, d\mathbf{z} \tag{3}$$

For a large dimensional subspace, the integral above becomes intractable, meaning we need an alternative method to solve for the posterior. The technique of variational inference is employed to address this problem, and is the origin behind the naming of Variational Autoencoders (VAEs). [10]

Variational inference is firstly applied by assuming a surrogate approximation of the true posterior, $q(\mathbf{z})$, which follows a Gaussian distribution. In order to constrain $q(\mathbf{z})$ to $p(\mathbf{z}|\mathbf{x})$, we attempt to minimize the Kullback-Leibler divergence, which computes the statistical distance between the distributions.

$$D_{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z} \tag{4}$$

Over a discrete space, this may be represented as

$$D_{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})) = \sum q(\mathbf{z}) \log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} \tag{5}$$

After rearranging the equation above, applying Eq.(2), and making $p(\mathbf{x})$ the subject, results in:

$$\log(p(\mathbf{x})) = D_{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}|\mathbf{x})) + \underbrace{\sum q(\mathbf{z}) \log\left(\frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})}\right)}_{\mathcal{L}\,(\text{ELBO})} \tag{6}$$

With the second term being defined as the *evidence lower bound* or ELBO. As $\log(p(\mathbf{x}))$ is a constant defined by the data, which is also always less than zero, and with the KL term, shown by the first term, being greater than zero, it results in the following constraint:

$$\mathcal{L} < \log(p(\mathbf{x})) \tag{7}$$

As we aim to minimize the KL divergence between the true posterior and the surrogate posterior in Eq.(6), this requires the maximization of $\mathcal{L}$. Such an approach offers a means to approximate the true posterior by prioritizing the maximization of $\mathcal{L}$. [10]

We can expand $\mathcal{L}$ further as shown in Eq.(8), which comprises a typical reconstruction loss term on the left (maximization of the expected log likelihood of the data, $\mathbf{x}$ given the latent variable $\mathbf{z}$) and an additional regularization loss term on the right.

In addition, Eq.(8), may be viewed as representing the decoder, capturing the reconstruction loss, and by the encoder shown by the regularization term. This presents a tradeoff between trying to maximize the first term while minimizing the second KL term.

$$\mathcal{L} = \underbrace{\mathbb{E}_{q(\mathbf{z})}[\log p(\mathbf{x}|\mathbf{z})]}_{\text{Reconstruction Error}} - \underbrace{\text{KL}(q(\mathbf{z}) \parallel p(\mathbf{z}))}_{\text{Regularization Term}} \tag{8}$$

Assuming $q(\mathbf{z})$ and $p(\mathbf{z})$ follow normal distributions $\mathcal{N}(\mu, \sigma^2)$, and $\mathcal{N}(0, 1)$, the regularization term may be simplified as:

$$KL(q(z)||p(z)) = \frac{1}{2}\sum_j(\sigma_j^2 + \mu_j^2 - 1 - \log(\sigma_j^2)) \tag{9}$$

3.2 Reparametrization trick

One challenge in maximizing Eq.(8) during the gradient descent backpropagation is that the latent variable $\mathbf{z}$, defined by μ and σ, causes the function to be non-differentiable. A solution is the Reparametrization Trick. Rather than sampling $\mathbf{z}$ directly from the distribution $q(\mathbf{z})$, this trick lets us represent $\mathbf{z}$ as a function of deterministic variables [11].

The Reparametrization Trick redefines $\mathbf{z}$ as:

$$\mathbf{z} = \mu + \sigma \odot \epsilon \tag{10}$$

Here, $\odot$ indicates element-wise multiplication and ϵ is a sample from a standard normal distribution: $\epsilon \sim \mathcal{N}(0, I)$. This adjustment allows the computation of the gradient of Eq.(8) during backpropagation.

4 Numerical example

We utilize a Variational Autoencoder (VAE) to process the MNIST dataset, containing grayscale images of handwritten digits. For consistent processing, pixel values are normalized between 0 and 1.

The neural network of both the encoder and decoder segments comprises a two hidden layers with 256 and 128 nodes, respectively. A latent layer is set to 2.

The VAE is optimized using the Adam optimizer. The combined loss function incorporates reconstruction and regularization loss, as described in the previous section. The former is quantified using cross-entropy, while the latter is described by Eq.(9).

Training is conducted over 10 epochs with a batch size of 64 images. As displayed in Fig.(3), the lower two-dimensional latent subspace offers a condensed representation of our data. This visualisation allows us to observe how the various MNIST digits are clustered. The distinct clusters suggest that the latent space has effectively captured the unique features of each digit.

For instance, sampling from the position (-3,1) in the latent space, we anticipate an image resembling the number 1. This is verified in Fig.(4), which indeed showcases this digit.

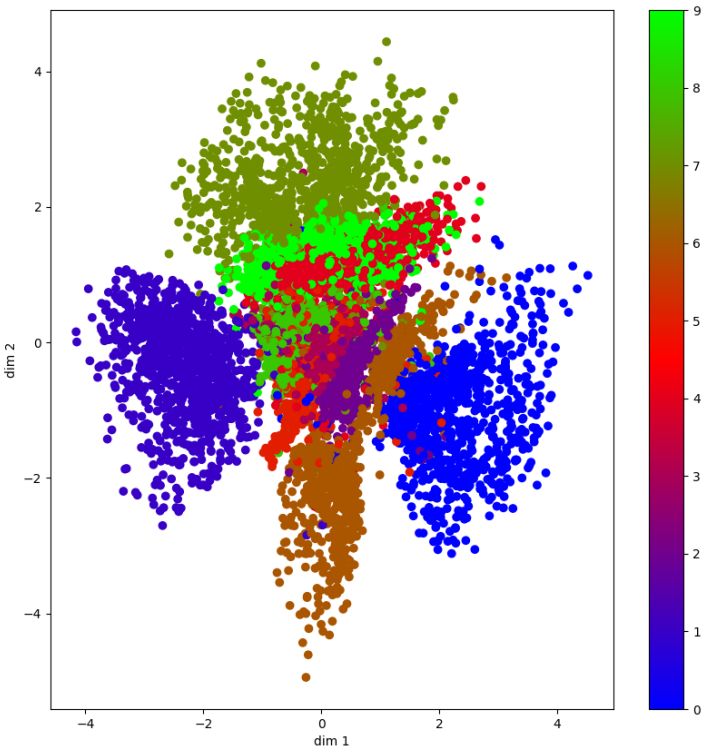

Figure 3. Data representation in the two-dimensional latent space for MNIST digits.

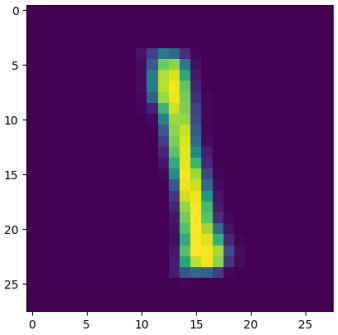

Figure 4. Sampled digit image from the point (-3,1) in the latent space.

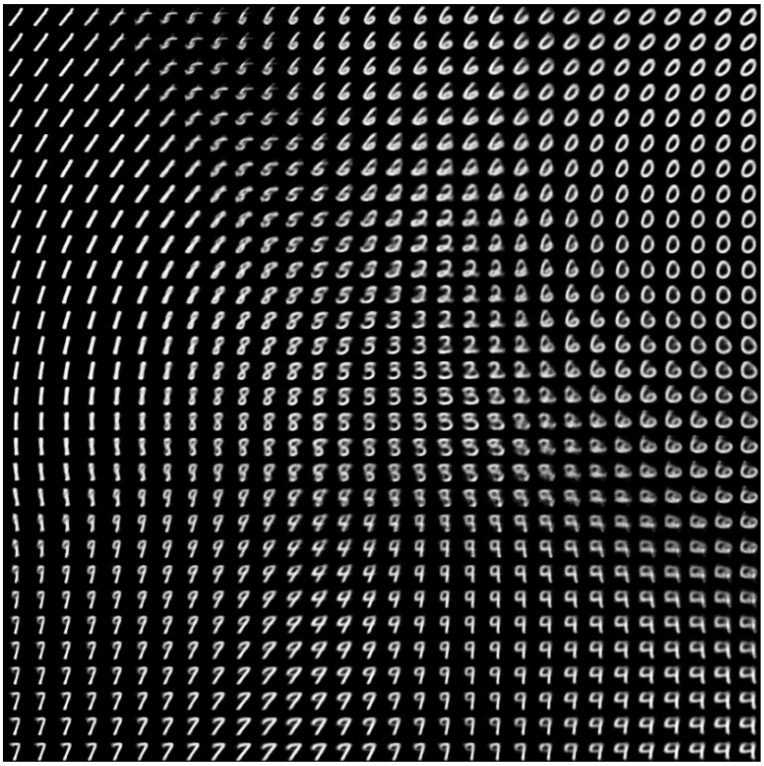

Figure 5. Visualisation of MNIST digits distributed over a 2D grid in the latent space.

Furthermore, by plotting a comprehensive grid over the latent space, we can visualize the entire spread of the MNIST digits. In Fig.(5), each position corresponds to a specific digit, allowing us to discern the transition between different numbers.

5 Conclusions

Variational Autoencoders (VAEs) offer a significant advancement over General Autoencoders (GAEs) by effectively encoding data distributions in latent space, enabling both superior data representation and advanced generative capabilities. In our example using the MNIST dataset, the latent space distinctively clustered handwritten digits, visually illustrating the efficiency of VAEs. By sampling from specific positions within this space, we successfully generated individual, recognizable digits. This illustrative example reinforces the prowess of VAEs not just in data compression, but also in their generative potential.

References

[1] Z. Liu, J. Smith, P. Tan, Journal of Machine Learning **15**, 129–158 (2020)

[2] C. Doersch, ArXiv preprint ArXiv:1606.05908 (2016)

[3] J. An, S. Cho, SNU Biointelligence & Bioinformatics Lab (2015)

[4] S.R. Bowman, L. Vilnis, O. Vinyals, A.M. Dai, R. Jozefowicz, S. Bengio, Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (2015)

[5] K. Sweers, T. Heskes, J. Krijthe, Journal of Financial Services Analytics **3**, 15 (2018)

[6] Z. Ding, J. Park, M. Kwon, Journal of Financial Security **22**, 315 (2023)

[7] A. Chauhan, Journal of AI Advances **18**, 42 (2022)

[8] W. Li, Y. Zhang, C. Tao, Journal of Medical Research **29**, 568 (2021)

[9] F. Ibrahim, Journal of Medical Imaging **25**, 123 (2022)

[10] D.P. Kingma, M. Welling, *Auto-encoding variational bayes*, in *International Conference on Learning Representations (ICLR)* (2013)

[11] J. Rocca, *Understanding variational autoencoders (vaes)* (2019), accessed: 10-10-2023, `https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73`

[12] I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, 2016)