Variational Autoencoders (VAEs): An overview

Yashi Nilesh Suba
Dwarkadas J.Sanghvi College of Engineering
Mumbai University
Mumbai, Maharashtra
yashisuba16ys@gmail.com

Dhvani Jatin Kansara
Dwarkadas J.Sanghvi College of Engineering
Mumbai University
Mumbai, Maharashtra
dhvani.djk@gmail.com

Neepa Shah
Head Of Department (Information Technology), Dwarkadas J.Sanghvi College of Engineering
Mumbai University
Mumbai, Maharashtra
neepa.shah@djsce.ac.in

Abstract— **Recent advancements in the technologies of generative modelling bolstered the development of many new models. Also, due to increasing need for variations in the generated output data there was an urge for forming a model which incorporates these requirements. Variational Autoencoders proved to be the most apt solution and thus is one of the most trending approach in generative modelling. This model leverages the power of unsupervised learning for feature modelling. It performs nonlinear factor analysis by sampling values from a probabilistic distribution to form an exclusive output. This paper presents an overview of Variational Autoencoders. In this paper we have tried to explain the basic functioning and the applications of VAEs.**

Keywords—Variational Autoencoders, latent attributes, Gaussian distribution.

I. INTRODUCTION

A colossal amount of data is already available and easily accessible to everyone, the important part is to learn models and algorithms to extract knowledge from the data and create new data which is beyond reach. Generative models are the best fit for this approach where we provide large amount of data (e.g., thousands of images, text sequences or sounds) to the system and train it to be capable enough to generate data similar to its own input data [1]. Recently autoencoders have been widely used for learning generative models of data. Autoencoders are a type of neural networks that can be deployed to generate efficient data in an unsupervised manner- method of inferring knowledge from unlabelled data [2]. The main aim of autoencoders is to compress the data into a short codes for the purpose of dimensionality reduction and denoising. Basically an autoencoder tries to learn an approximate function for the input function so that the output data is similar to the input data. So an autoencoder learns a compressed representation of the input with the help of an encoder and then decompressing it back to appear similar to the original data with the help of a decoder. Thus having a superior hand for the purpose of dimensionality reduction.

Although autoencoders provide great functionalities, when using any generative model, we can simply generate a random, new output, that looks similar to the training data but variations cannot be developed. Forming similar output images as that of the input can certainly be done with the help of VAE also. But more often, you'd like to alter, or explore variations on data you already have, and not just in a random way, but in a desired and a particular direction. In such cases, VAEs work better than any other method currently available. VAEs also work on the concept of encoders and decoders but instead of just learning a function for representing the data (like autoencoders) variational autoencoders learn the parameters of the probability distribution that represents the input data. So VAE learns to model the data into a distribution, and so later we can sample a value from the distribution to form a new output data. Thus, VAE can also function as a generative model similar to Generative Adversarial Networks (GANs).

II. RELATED WORKS

Variational autoencoders being a highly advanced model, it is developed by taking the basic idea and then refining the concept of various other related models. VAEs gave better accuracy and convenience over many other trivial generative models and is therefore one of the most advanced technique available. Many Generative models are available today, they can be summarized as given in figure 1.

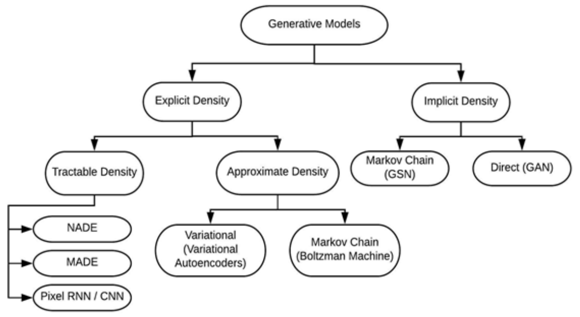


Figure 1: Taxonomy of Generative models [3]

The basic difference between Implicit Density Models and Explicit Density Models is that direct probability distribution of a variable is not available in the implicit one and so training such models becomes very difficult. Implicit density estimation has to learn a model that can sample the probability distribution without explicitly defining it. Of all the types of Generative models available the most popular ones are: GANs, VAEs and Pixel Recurrent Neural Network (RNN) and Pixel Convolutional Neural Network (CNN). Besides these models autoencoders have also proven to be an important base for the creation of VAEs.

A. Generative Adverserial Networks (GANs)

Generative Adversarial Networks (GANs) are implicit density models. They have to learn a probability distribution for the latent variables in their inputs. The architecture of GANs consists of two main components: the generator and the discriminator. The generator develops photorealistic images and the discriminator tries to find the difference between real images provided and the images formed by the generator. Based on these differences, the model learns the intermediate features and tries to reduce the difference in each iteration. This is done until the image formed resembles the real image and the discriminator is no longer able to identify any difference between the two images.

GANs are used to produce photorealistic images for hypothetically viewing the interiors of houses or industries or for creating simulations of some gaming items or environment. They can also be used for visualizing new designs and patterns in shoes, bags, clothes etc.

B. Markov Chain

Markov chain is a stochastic model. A stochastic model is a model having random probability distribution that can be studied statistically but cannot be predicted accurately. It follows a process in which sequence of possible events are predicted based on the output of the previous process. It is also considered to be a memoryless model as it does not keep a track of all the occurring events. Since there is no record of the previous data and only one event is taken into account to predict the successor event, the prediction cannot be considered to be plausible and precise.

C. Neural Autoregressive Distribution Estimator (NADE)

A Neural Autoregressive Distribution Estimator can be considered to have been inspired from the concept of Boltzman machine. A Boltzman machine is a powerful approach for modelling distribution of high dimensional vectors but the problem is that it approximates the values and so the result is not tractable [6]. NADE provides a more tractable approach by leveraging the probability product rule and weight sharing. Thus NADE has better and generalized performance [5].

D. Pixel RNNs and Pixel CNNs

Pixel RNNs and CNNs are fully visible belief networks that model the density explicitly. Consider, if we have an image data X and the probability or the likelihood of the data is supposed to be obtained. Here, the chain rule is used to decompose likelihood of the data X into a product of one dimensional distributions in contrast to other approached that use more dimensions. Here probability of any pixel 'x_i' is conditioned on all previous pixels (x_1 to x_{i-1}). Then once we have the probability of each pixel we can maximize the knowledge from training data. But the process of obtaining the likelihoods of each pixel and then learning a complex distribution is very tedious so neural networks are used. Since we have to learn data about each pixel and this is dependent on the values of all the other pixels, the data has to be calculated recursively. So, RNNs are used for this purpose.

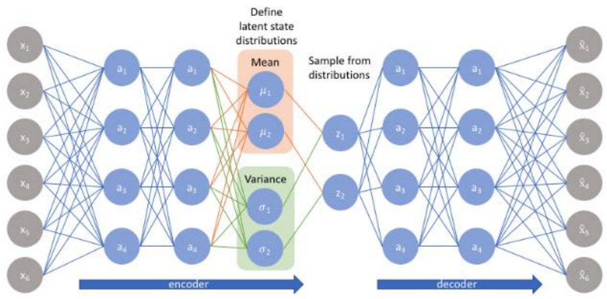
RNNs used in these models use a sequential approach for calculating the probabilities. But since, the dependencies are very high this type of sequential generation becomes very slow. To overcome this problem a new model called pixel CNNs was introduced using a similar kind of formulation but the difference is that a CNN is used over a context region around a particular pixel. So using the part of the image from the area that is already generated, the values of the next pixel can be calculated [3]. Thus, improving the efficiency by reducing the dependencies.

III. Definition of VAE

A Variational Autoencoder can be defined as a combination of primitive autoencoders with the concepts of Variational Approximation and Reparameterization trick. Here variational approximation refers to the maximization of the variation lower bound and reparameterization refers to the adjustments made in the parameters (mean and the standard deviation) to adjust the weights of the intermediate layers.

IV. Architecture And Working

For any data to be accurately generated, the model should first decide what it wants to generate before mapping any pixel value of the output. This decision can be taken on the basis of the features in an image represented in the form of vectors. Each attribute in this vector can be considered to be a latent variable, which is randomly sampled from a Gaussian probability distribution function in the form of floating point numbers which is generated based on prior understanding of the input data the model is trained upon.



Define latent state distributions
Sample from distributions
x_1 x_2 x_3 x_4 x_5 x_6
a_1 a_2 a_3 a_4
Mean
μ_1 μ_2
Variance
σ_1 σ_2
z_1 z_2
$\hat{x}_1$ $\hat{x}_2$ $\hat{x}_3$ $\hat{x}_4$ $\hat{x}_5$ $\hat{x}_6$
encoder
decoder

Figure 2: Architecture of VAE model [8]

Consider an example, every image of a flower can be assumed to consist of attributes like: shape, petal_color, texture, thorns, center_color, bunch, no_of_petals etc. These are called as latent attributes. Every attribute in this list can be represented with the help of latent variables collectively forming a latent vector for each image.

Accordingly the picture of such a flower is represented in a mathematical format. These mathematical units are then mapped into probability function by the encoder of the variational autoencoder (VAE). Thus, it provides a probabilistic manner for describing an observation in latent space [8]. So the features in the form of discrete values are extracted which are translated into a probability distribution developed by the encoder of the VAE model, as shown in figure 4. This is the work of the encoder of VAE.

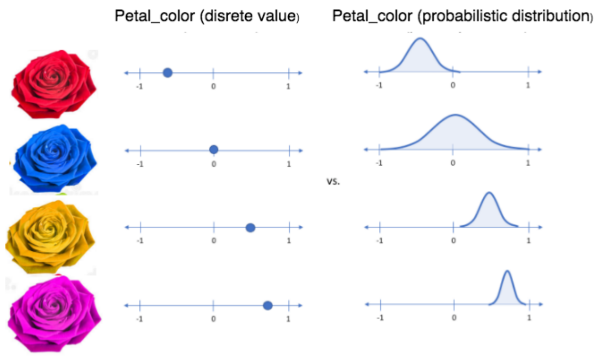
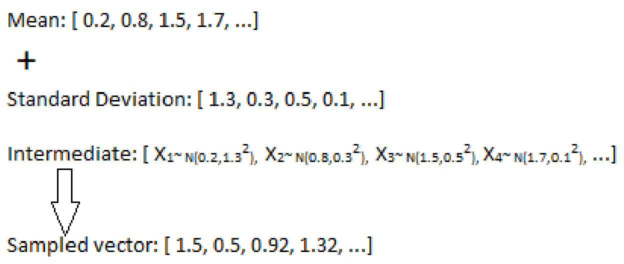


Figure 3: conversion from discrete to probability distribution [8]

While the encoder develops a probability distribution, the decoder samples units of data from the function by using the mean (μ) and standard deviation (σ) of the probability distribution obtained from the encoder and then an intermediate formula is used to form a new sample vector according to the required output, as seen in figure 4.



Mean: [0.2, 0.8, 1.5, 1.7, ...]

+

Standard Deviation: [1.3, 0.3, 0.5, 0.1, ...]

Intermediate: [$X_1 \sim N(0.2, 1.3^2)$, $X_2 \sim N(0.8, 0.3^2)$, $X_3 \sim N(1.5, 0.5^2)$, $X_4 \sim N(1.7, 0.1^2)$, ...]

Sampled vector: [1.5, 0.5, 0.92, 1.32, ...]

Figure 4: Sample intermediate formulation

Now consider the case where two different sample values are extracted from the latent distribution, we can generate two different outputs based on the sampled values as exemplified in figure 5.

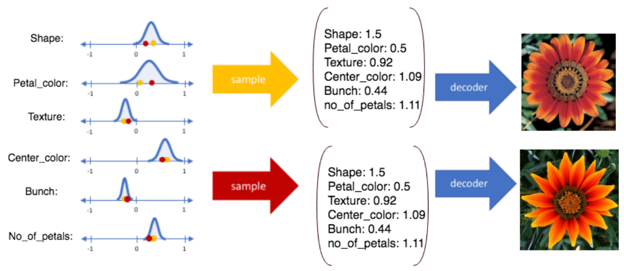


Figure 5: Sampling of data [8]

The sampling part mentioned in figure 4 requires some extra attention. In normal neural networks, weights in the network are adjusted by the process called backpropagation, which analyses the output loss to find the relationship among each parameter and adjust intermediate weights accordingly. However, we cannot use this method for random sampling. Thus to adjust weights in a VAE we use a reparametrization trick. In this, we first sample a value ε randomly from a unit Gaussian distribution, and then adjust its value to the mean (μ) and scale it to standard deviation (σ) required by the distribution by shifting ε in latent space. Thus the reparametrized distribution will look eq. 1.

$$z = \mu + \sigma \odot \varepsilon \tag{1}$$

Thus with this trick, we can assign weights to the intermediate layers and optimize parameters of the distribution while maintaining the ability to randomly sample from the distribution as seen in figure 6.

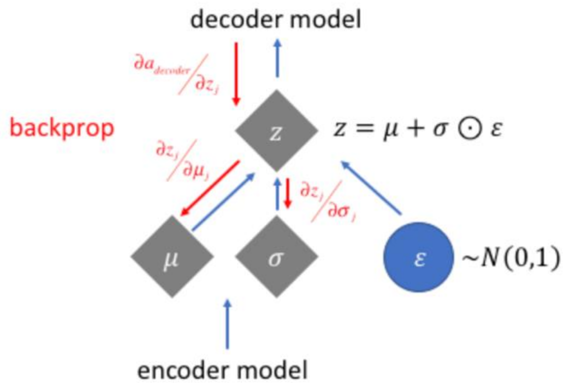

decoder model
$\partial a_{decoder} / \partial z_i$
backprop
z
$z = \mu + \sigma \odot \varepsilon$
$\partial z_i / \partial \mu_i$
$\partial z_i / \partial \sigma_i$
μ
σ
ε
$\sim N(0,1)$
encoder model

Figure 6: Reparameterization trick [8]

The output image is also represented in the form of a vector which is a combination of latent attributes. The values of the latent attributes are chosen in the form that they produce the most desirable feature. These features are calculated using vector arithmetic. For example, from a vector of image of a flower and a vector of the image of dew drops we can produce an image with features of a flower with dew drops on it.

Vector arithmetics

So how can we generate these complex features? This is certainly done by the decoder of the VAE using simple vector arithmetic in the latent space.

- ***Interpolating between samples***

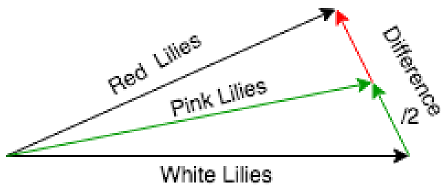


Figure 7: Vector interpolation [21]

One way is, if we wish to engender a unique sample halfway between two samples, just find the distinction between their mean (μ) vectors, and add half the difference to the pristine, and then simply decode it. This can be used in applications where we want to develop an image which has the features of two other images. For example, to form a pink lily we can extract features from that of red and white lilies and produce intermediate vector by calculating the vector difference and then dividing it into equal halves, as shown in figure 7.

- ***Adding new features to samples***

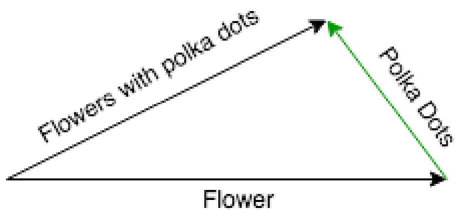


Figure 8: Vector Addition [21]

For situations where we need to produce specific features such as flower with polka dots this method is used. Here we need to use two samples, one with a plain flower and other with polka dots, obtain their encoded vectors and find the difference. Then add the new polka dot feature vector to the original flower image, as shown in figure 8. These are the ways by which the decoder forms the desired features of the output image.

V. FUNCTIONING AND EQUATIONS

Any data to be represented accurately needs two important parameters: Information (I) and entropy (H).

Knowledge obtained from any probability distribution can be indicated by the equation:

$$I = -\log p(x) \tag{2}$$

Here, if the probability is very low the negative log value becomes very high so the value of 'I' becomes high. Similarly if the probability is equal to 1 (i.e. the event is most certain) then the value of information is zero as log1=0. So the information obtained from the most certain event is zero.

Entropy is elucidated as the expectation or the average of information. It can be represented by eq. 3.

$$H = -\sum p(x) \log p(x) \tag{3}$$

Now, consider two probability distributions 'p' and 'q' then the difference between them is called as KL divergence. Thus, KL(p||q) is said to be a measure of dissimilarity between the two distributions. So, using the concept of entropy, KL divergence is calculated for a VAE model and can be formulated as:

$$KL(p||q) = \sum p(x) \log \frac{p(x)}{q(x)} \tag{4}$$

Here $KL \geq 0$ and $KL(p||q) \neq KL(q||p)$

This is one of the methods of calculating divergence but it is certainly not the only one. This method is closely related to information but for variational model a more graphical approach is preferable.

Consider two variables 'z' and 'x' such that z is hidden variable (feature of input image) and x is observation (output image). We can only see the output image x but for this the intermediate feature z has to be computed in the form of posterior probability. This can be done by the eq. 5.

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x,z)}{p(x)} \tag{5}$$

Where, $p(x) = \int p(x|z)p(z)dz$ (6)

But computing the value of p(x) is very complicated because it is a marginal function that can be represented in the form of an integral, and solution to the integral is intractable. The number of integrals increases with increase in dimensions. This makes it more intractable. A solution to estimate this value, is by applying variational inference. For this, we define q(z) such that it has a tractable distribution and it lies in close approximation or proximity to the intractable function p(x). The distance between p(x) and q(x) should be as small as possible. To get q(x) close to the value of p(x), KL divergence is used. Thus, it can be implied that the KL divergence has to be minimized. i.e.

$$\min KL(q||p) = -\sum q(x) \log \frac{p(x)}{q(x)} \tag{7}$$

But here q=q(z) and p = p(z|x),

$$\min KL(q(z)||p(z|x)) = -\sum q(z) \log \frac{p(z|x)}{q(z)} \tag{8}$$

From equation (4) we have,

$$KL(q(z)||p(z|x)) = -\sum q(z) \log \frac{\frac{p(x,z)}{p(x)}}{q(z)} \tag{9}$$

Solving further and simplifying the above equation we get,

$$KL(q(z)||p(z|x)) = -\sum q(z) \log \frac{p(x,z)}{q(z)} + \sum q(z) \log p(x) \tag{10}$$

But $\log p(x)$ is constant and can be brought outside the summation sign and summation of q(z) for all 'z' is '1' thus the equation becomes:

$$KL(q(z)||p(z|x)) = -\sum q(z) \log \frac{p(x,z)}{q(z)} + \log p(x) \tag{11}$$

Rearranging:

$$\log p(x) = KL(q(z)||p(z|x)) + \sum q(z) \log \frac{p(x,z)}{q(z)} \tag{12}$$

Here, $\log p(x)$ is a constant and our goal is to minimize the KL divergence ($KL(q(z)||p(z|x))$). For this, we have to maximize $\sum q(z) \log \frac{p(x,z)}{q(z)}$. This quantity is denoted by 'L' which indicates the variational Lower bound value.
The value of 'L' can also be indicated in the form,

$$L = E_{q(z)} \log p(x|z) - KL(q(z)||p(z)) \tag{13}$$

Where,
$$E_{q(z)} \log p(x|z) = \sum q(z) \log p(x|z) \tag{14}$$

The term $E_{q(z)} \log p(x|z)$ is called the expectation or the likelihood of pixel 'x' given 'z'.
So to make 'q' similar to 'p' we have to minimize the KL divergence which is identical to maximizing 'L' (lower bound), and 'L' means make q similar to p and at the same time maximize the expectation.

Now, consider an encoder, whose input is x passes through a neural net (encoder) to produce intermediate distribution z, which passes through another neural net (decoder) to output a reconstructed image x', as shown in figure 9.

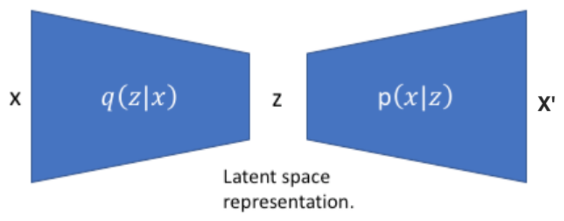


Figure 9: latent space representation [8]

As mentioned, eq. 13 represents two terms, the Expectation term and KL divergence term. The Expectation term is called as the reconstruction likelihood, because, if we assume z to have a Gaussian distribution then $P(x|z) = e^{-(|x-x'|)^2}$, inserting this value in Expectation

term (eq. 14) we can see that we are left with $\Sigma q(z)(x - x')^2$. Since we are to maximize 'E', it can be done by minimizing (x-x') which is the difference between the input and the reconstructed image, thus E is called as reconstruction likelihood, which is to be maximized. And the second term in eq. 13 ensures that our learnt distribution 'q' is similar to true prior distribution 'p', which is to be minimized. We require these two terms to ensure that the latent space is continuous, thus, making interpolation of sampled values is easy. Thus, from the intermediate Gaussian distribution 'z', mean and variance are calculated from which values are sampled to generate the output image.

VI. Applications

The latent space created by a VAE can be exploited to generate novel, creative data given sufficient training data. VAEs can be used not just to obtain a dense representation of the input, but also to generate new samples, reconstruct clean data from noisy data (called denoising), morphing subtleties from one images to other, sophisticated data manipulations, smooth transition of voices, and much more. VAEs have many practical applications, and many more are being discovered constantly, some of the domains are briefly described here.

1) Computer Vision

Face Generation:
Methods like attribute selection can be applied on given input sentence to map attributes mentioned in the sentence to vectors and generate images corresponding to those vectors [9]. For example, given sentence "An old man with white hair is frowning", the model extracts words to map onto attributes like: (gender: man), (age: old), (hair colour: white), (expression: frowning) to obtain disentangled latent representations, which is passed through the decoder to give an appropriate output.

Facial Expression detection and editing:
Facial features can be mapped using latent variables, for example a "smile" is encoded a value in the latent space, thus for given input, closer latent vector values will help detect the relevant feature. Editing can also be performed by generating new images by using latent vector arithmetic [10]. For example, man vector + smile vector = smiling man vector, later decoded by the decoder to generate a smiling man image.

Handwritten Digit Generation:
Popularly used MNIST dataset [11] for handwritten digit generation, works in the similar manner as explained above, by plotting digit features in latent space.

Medical Image Analysis:
Manifold learning perspective of VAEs can be leveraged for medical applications whereby local structure of input data can be preserved even after applying dimensionality reduction. This mechanism can be used for generation and classification of many clinical parameters [12].

2) Sound

Speech emotion recognition
Speech signals is a distribution of various features like emotional state, speaking style, linguistic content, age, gender, environmental and channel effects [13]. Thus, learning is done by giving a latent representation of speech emotion, and detection can be done by other supported models like Long Short Term Memory (LSTMs) to create an overall classification framework.

Music Generation
New musical notes can be generated from learnt, latently represented notes, giving artists an intuitive palette to work from [14]. They can combine multiple musical scores for exploring and mixing music ideas and thereby creating innovative melodies.

3) Text Generation

Given two input sentences, intermediate sentences can be generated using VAEs, with the help of latent variable values in the latent space. There is a transition from one sentence to another by replacing certain words with other similar words calculated using vector algebra, thus giving rise to multiple intermediate sentence formations [15].

4) Forecasting

Video forecasting
Generating future frames in a video can be done using VAEs. The underlying stochasticity can be modelled using latent variables. Training can be done in the form of reinforcement learning for model-based system in which next frame prediction is made conditioned on current frame and the current action performed by the agent [16] or by using latent space on top of ground-truth frames, or previously predicted frames [17].

Human motion synthesis from static images
Using realistic videos as training data, human motion can be interpolated on a trajectory from a given static image to predict the motion or future position of an object in the same image. The trajectory of upcoming motion is learnt by representing them in the form of latent variables. The model is thereby encouraged to extract as much information as possible from the input image before relying on encoding the trajectories themselves [18].

5) Anomaly Detection

Since VAEs model distributions, anomalies can be seen as coming from a different process altogether, thus their vectors will be very far from the normal position vectors in the latent space. Thus, results having high reconstruction errors or low reconstruction probabilities can be regarded as anomalies[19].

6) Reinforcement Learning

VAEs allow latent planning and information to be is easily sampled from, thus an agent can learn to predict multimodal transition dynamics in model-based reinforcement learning effectively [20]. Complex, high dimensional outcome distributions can be captured using VAEs to learn about previously generated model using function approximations.

Apart from these application domains, VAEs are also used for denoising - generating clear images out of noisy images, the input images are inserted into the encoder along with a noise factor, and in the output we obtain a clear image without noise (as explained in introduction section). It is also used for dimensionality reduction applications for data visualization purposes.

VII. Advantages

One advantage that variational autoencoders have over vanilla autoencoders is that, the latent space of VAE is continuous. The direct encoding coordinates will not be a single sampled point, rather a continuous probability distribution, where the encoding of the input should be centred around and is an area around the mean. Because of this continuity, the decoder is not restricted over limited samples rather it is exposed to a range of values from which it can pick different variations for the similar input. Also, care is taken that there is overlap between two slightly different classes as well, so as to obtain a smooth interpolation. Because of this, the random sampling is accurate and easy, thus making it useful as a generative model as construction of new samples can be done easily on the smooth continuous space [21]. As shown in fig. 8, to pick a random point from latent space there are more chances for a sampled point to belong to a certain class in fig 10(b) than in fig 10(a). Thus interpolation is easy.

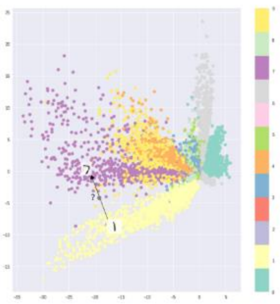


Figure 10(a): Discontinuous Latent space [21]

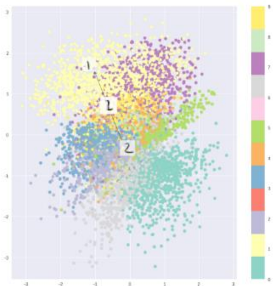


Figure 10(b): Continuous Latent space [21]

Other advantages of VAEs include:

- The log-likelihood can be estimated very efficiently via importance sampling. The expected log-likelihood measures how well samples from q(z|x) (represents probability distributed value z given x) are able to explain the data x. Thus, saying that one can optimize a bound based on the likelihood of the data and not the reconstruction error is a property which makes VAEs more appealing [22].
- It is suitable for Manifold learning, which is a non-linear method of dimensionality reduction. Many datasets have artificially high dimensionality, which is reduced non-linearly using this algorithm. It ultimately works by discovering minimal factors of variation present in input data which alone can be used to generate the output.
- The VAE framework is straightforward, thus, it can be extended to a large range of model architectures and therefore can be leveraged for various generative applications. This is advantageous over Boltzman Machine which requires careful design to assert traceability.
- It is faster compared to GANs
- It is more flexible as a generative model
- It has a measurable and clear objective function Enables end-to-end gradient training.

VIII. Disadvantages

The main drawback of VAE compared to other generative models is that, VAE produces blurred results, sharp images cannot be formed using VAE like in case of GANs. This is because, it tries to maximize log likelihood by minimizing KL Divergence. It ignores features of inputs that occupy less pixels or pixels that cause less change in the brightness in attempt to maximize log likelihood. It will assign high probability to points not only in the training set but also to other points which include blurry data this is due to the fact that VAE is spread across a Gaussian distribution. Thus, the inference model used during training are not expressive enough to capture true posterior distribution rather performs only limited approximation.

Another drawback of the model is that, its inference revolves around ‘z’ and ‘x’ only, i.e. it learns to generate ‘z’ given ‘x’, thus, it focuses on only one problem, on the other hand, the other generative models can give results over any subset of variables given any other subset of variables. So the scope is not restricted to one feature at a time.

References

[1] blog.openai.com/generative-models/[last accessed on 5th August 2018]

[2] en.wikipedia.org/wiki/Autoencoder [last accessed on 5th August 2018]

[3] ww.cc.gatech.edu/classes/AY2018/cs7643_fall/slides/L18_generative_annotated.pdf [last accessed on 8th August 2018]

[4] en.wikipedia.org/wiki/Generative_adversarial_network

[5] en.wikipedia.org/wiki/Markov_chain

[6] proceedings.mlr.press/v15/larochelle11a/larochelle11a.pdf [last accessed on 8th August 2018]

[7] Benigno Uria, Marc-Alexandre Cˆot´, Karol Gregor, Iain Murray, Hugo Larochelle Neural, “Autoregressive Distribution Estimation”, Journal of Machine Learning Research 1 (2000) 1-48

[8] www.jeremyjordan.me/variational-autoencoders/ [last accessed on 6th August 2018]

[9] Xinchen Yan , Jimei Yang, Kihyuk Sohn and Honglak Lee, “Attribute2Image: Conditional Image Generation from Visual Attributes”

[10] Raymond Yeh, Ziwei Liu, Dan B Goldman, Aseem Agarwala, “Semantic Facial Expression Editing using Autoencoded Flow”

[11] yann.lecun.com/exdb/mnist/ [last accessed 10th August 2018]

[12] Eunbyung Park, “Manifold Learning with Variational Auto-encoder for Medical Image Analysis”

[13] Siddique Latif, Rajib Rana, Junaid Qadir, Julien Epps, “Variational Autoencoders for Learning Latent Representations of Speech Emotion: A Preliminary Study”

[14] Adam Roberts, Jesse Engel, Douglas Eck, “Hierarchical Variational Autoencoders for Music”

[15] nicgian.github.io/text-generation-vae/ [last accessed on 7th August 2018]

[16] Rui Shu, “Stochastic Video Prediction with Deep Conditional Generative Models”

[17] Alex X. Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine, “Stochastic Adversarial Video Prediction”

[18] Jacob Walker, Carl Doersch, Abhinav Gupta, and Martial Hebert, “An Uncertain Future: Forecasting from Static Images using Variational Autoencoders”

[19] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, Jie Chen, Zhaogang Wang, Honglin Qiao, “Unsupervised Anomaly Detection via Variational Auto-Encoder for Seasonal KPIs in Web Applications”

[20] Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker, “Learning Multimodal Transition Dynamics for Model-Based Reinforcement Learning”

[21] towardsdatascience.com/intuitively-understanding-variational-autoencoders-1bfe67eb5daf [last accessed on 6th August 2018]

[22] www.deeplearningbook.org/contents/generative_models.html [last accessed on 9th August 2018]