Introduction to Variational AutoEncoder

Mallikarjun S
Electrical Communication
Indian Institute of Science
mallikarjun1@iisc.ac.in

Rajendra P Singh
Electronics Systems
Indian Institute of Science
rajendraps@iisc.ac.in

Ronak D Dedhiya
Computational
and Data Science(CDS)
Indian Institute of Science
ronakdedhiya@iisc.ac.in

Abstract

Variational Autoencoders (VAEs) have emerged as one of the most popular approaches to unsupervised learning of complicated distributions. We introduce intuitions behind VAEs, explains the mathematics behind them and attempts to do qualitative assessment of learned latent factors and generated samples using basic VAE, conditional VAE in conjugation with β - VAE.

1 Introduction

Variational Autoencoders (VAEs) are powerful generative models that merge elements from statistics and information theory with the flexibility offered by deep neural networks to efficiently solve the generation problem for high-dimensional data. The key insight of VAEs is to learn the latent distribution of data in such a way that new meaningful samples can be generated from it. This approach led to tremendous research and variations in the architectural design of VAEs, nourishing the recent field of research known as unsupervised representation learning.

Having a representation that is well suited to the particular task and data domain, can significantly improve the learning success and robustness of the chosen model [1]. VAE find applications in learning a disentangled representation of the generative factors in the data can be useful for a large variety of tasks and domains [2].A modified version of VAE provides a generative way to learn encoded state; where a change in one dimension corresponds to a change in one factor of variation, keeping almost no change in other factors [3].The influential factors in VAE are also analyzed in the paper [4].

Also there are interesting extensions to VAE in conjugation with other techniques. Generative adversarial networks [5] usually generate more clear images; therefore, some works have combined variational and adversarial inferences [6, 7] for using the advantages of both models. Variational discriminant analysis [8] has also been proposed for classification and discrimination of classes. For image data, modelling the images with caption, a fusion of VAE and convolutional neural network is also proposed [9]. RNN in combination with VAE exploits the learning of distributed latent representation of entire sentence explicitly modelling style, topic and high level synaptic features [10].DRAW[11] attempts to generate images step by step correlated to natural form of painting, sketching which involves sequential and iterative steps. It utilizes Deep recurrent attention network with VAE for iterative construction of complex images.

Although VAEs were principally designed as generative models for image and text generation, [12] have leveraged some of the qualities of VAE in the anomaly detection domain. In this paper, we will perform the qualitative assessment of standard VAE and conditional VAE in conjugation with β VAE while understanding mathematics intuition behind the framework.

2 The Variational Autoencoder

Generative modeling is a broad area of machine learning which deals with models of distributions $P(X)$, defined over datapoints X in some potentially high-dimensional space χ. Usually X is distributed according to some unknown distribution $P_{gt}(X)$, and our goal is to learn a model P which we can sample from, such that P is as similar as possible to P_{gt}. Currently, one of the most popular frameworks is the Variational Autoencoder. The assumptions of this model are weak, and training is fast via backpropagation. VAEs do make an approximation, but the error introduced by this approximation is arguably small given high-capacity models. These characteristics have contributed to a quick rise in their popularity.

2.1 Latent variable model

Formally, say we have a vector of latent variables z in a high-dimensional space Z which we can easily sample according to some probability density function (PDF) $P(z)$ defined over Z. Then, say we have a family of deterministic functions $f(z;\theta)$, parameterized by a vector θ in some space θ, where $f : Z \times \theta \longrightarrow X$. We wish to optimize θ such that we can sample z from $P(z)$ and, with high probability, $f(z;\theta)$ will be like the X's in our dataset. To make this notion mathematically precise, we are aiming maximize the probability of each X in the training set under the entire generative process, according to

$$P(X) = \int P(X/z;\theta)P(z)dz \tag{1}$$

To solve Equation (1), how to define the latent variables z (i.e., decide what information they represent), VAE assumes that there is no simple interpretation of the dimensions of z, and instead assert that samples of z can be drawn from a simple distribution, i.e., $N(0,I)$, where I is the identity matrix. How is this possible? The key is to notice that any distribution in d dimensions can be generated by taking a set of d variables that are normally distributed and mapping them through a sufficiently complicated function. For example, say we wanted to construct a 2D random variable whose values lie on a ring. If z is 2D and normally distributed, $g(z) = z/10 + z/ \parallel z \parallel$ is roughly ring-shaped. Hence, provided powerful function approximators, we can simply learn a function which maps our independent, normally-distributed z values to whatever latent variables might be needed for the model, and then map those latent variables to X.

2.2 Setting up the objective

The key idea behind the variational autoencoder is to attempt to sample values of z that are likely to have produced X, and compute $P(X)$ just from those. This means that we need a new function $Q(z|X)$ which can take a value of X and give us a distribution over z values that are likely to produce X. Hopefully the space of z values that are likely under Q will be much smaller than the space of all z's that are likely under the prior $P(z)$. This lets us, for example, compute $E_{z\sim Q}P(X|z)$ relatively easily. However, if z is sampled from an arbitrary distribution with PDF $Q(z)$, which is not $N(0,I)$, then how does that help us optimize $P(X)$? The relationship between $E_{z\sim Q}P(X|z)$ and $P(X)$ is one of the cornerstones of variational Bayesian methods. We begin with the definition of Kullback-Leibler divergence (KL divergence or D) between $P(z|X)$ and $Q(z)$, for some arbitrary Q (which may or may not depend on X):

$$D[Q(z) \parallel P(z|X)] = E_{z\sim Q}[logQ(z) - logP(z|X)] \tag{2}$$

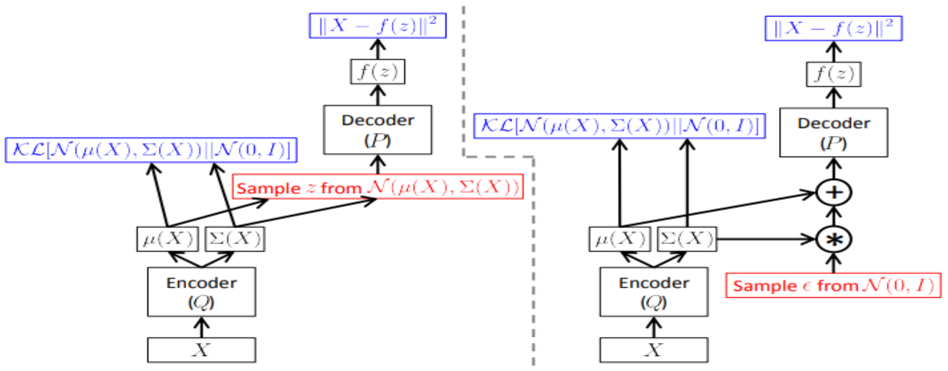
We can get both $P(X)$ and $P(X|z)$ into this equation by applying Bayes rule to $P(z|X)$:

$$D[Q(z) \parallel P(z|X)] = E_{z\sim Q}[logQ(z) - logP(X|z) - logP(z)] + logP(X). \tag{3}$$

Here, $\log P(X)$ comes out of the expectation because it does not depend on z. Negating both sides, rearranging, and contracting part of $E_{z\sim Q}$ into a KL-divergence terms yields:

$$logP(X) - D[Q(z|X) \parallel P(z|X)] = E_{z\sim Q}[logP(X|z)] - D[Q(z|X) \parallel P(z)] \tag{4}$$

This equation serves as the core of the variational autoencoder. In two sentences, the left hand side has the quantity we want to maximize: $logP(X)$ (plus an error term, which makes Q produce z's that can reproduce a given X; this term will become small if Q is high-capacity). The right hand side is something we can optimize via stochastic gradient descent given the right choice of Q. Note that the framework—in particular, the right hand side of Equation (4) has suddenly taken a form



$\|X - f(z)\|^2$
$f(z)$
Decoder (P)
$\mathcal{KL}[\mathcal{N}(\mu(X),\Sigma(X))\|\mathcal{N}(0,I)]$
Sample z from $\mathcal{N}(\mu(X),\Sigma(X))$
$\mu(X)$
$\Sigma(X)$
Encoder (Q)
X
$\|X - f(z)\|^2$
$f(z)$
Decoder (P)
$\mathcal{KL}[\mathcal{N}(\mu(X),\Sigma(X))\|\mathcal{N}(0,I)]$
+
*
$\mu(X)$
$\Sigma(X)$
Sample ϵ from $\mathcal{N}(0,I)$
Encoder (Q)
X

Figure 1: A training-time variational autoencoder implemented as a feedforward neural network, where P(X|z) is Gaussian. Left is without the "reparameterization trick", and right is with it. Red shows sampling operations that are non-differentiable. Blue shows loss layers. The feedforward behavior of these networks is identical, but backpropagation can be applied only to the right network.

which looks like an autoencoder, since Q is "encoding" X into z, and P is "decoding" it to reconstruct X. Starting with the left hand side, we are maximizing $logP(X)$ while simultaneously minimizing $D[Q(z|X) \parallel P(z|X)]$

2.3 Optimizing the objective

First we need to be a bit more specific about the form that $Q(z|X)$ will take. The usual choice is to say that $Q(z|X) = N(z|\mu(X;\theta),\Sigma(X;\theta))$, where μ and Σ are arbitrary deterministic functions with parameters θ that can be learned from data (we will omit θ in later equations). In practice, μ and Σ are again implemented via neural networks, and Σ is constrained to be a diagonal matrix. The advantages of this choice are computational, as they make it clear how to compute the right hand side. The last term $D[Q(z|X) \parallel P(z)]$ is now a KL-divergence between two multivariate Gaussian distributions, which can be computed in closed form. After all, we are already doing stochastic gradient descent over different values of X sampled from a dataset D. The full equation we want to optimize is:

$$E_{X\sim D}[logP(X) - D[Q(z|X) \parallel P(z|X)]] = E_{X\sim D}[E_{z\sim Q}[logP(X|z)] - D[Q(z|X) \parallel P(z)]] \tag{5}$$

If we take the gradient of this equation, the gradient symbol can be moved into the expectations. Therefore, we can sample a single value of X and a single value of z from the distribution Q(z|X), and compute the gradient of:

$$logP(X|z) - D[Q(z|X) \parallel P(z)] \tag{6}$$

We can then average the gradient of this function over arbitrarily many samples of X and z, and the result converges to the gradient of Equation (5).

2.4 Reparametrization trick

To see the problem a different way, the network described in Equation (6) is much like the network shown in Figure 1(left). The forward pass of this network works fine and, if the output is averaged over many samples of $X$and z, produces the correct expected value. However, we need to back-propagate the error through a layer that samples z from $Q(z|X)$, which is a non-continuous operation and has no gradient. Stochastic gradient descent via backpropagation can handle stochastic inputs, but not stochastic units within the network! The solution, called the "reparameterization trick" in [13], is to move the sampling to an input layer. Given $\mu(X)$ and $\Sigma(X)$—the mean and covariance of $Q(z|X)$. we can sample from $N(\mu(X),\Sigma(X))$ by first sampling $e \sim N(0,I)$, then computing $z = \mu(X) + \Sigma^{1/2}(X) \cdot e$. This is shown schematically in Figure 1(right).

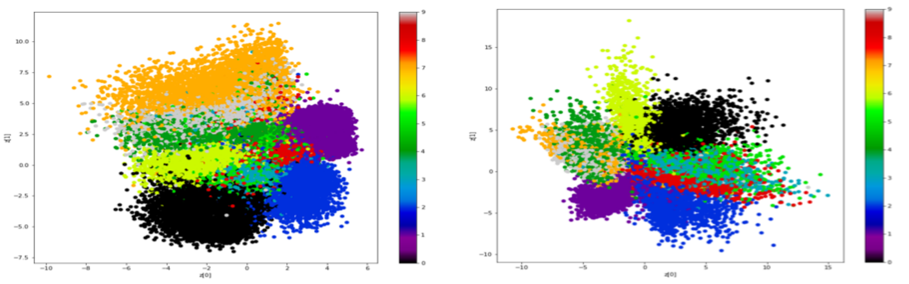


Figure 2: VAE Model with 2 convolutional layers (1 and 32 filter of 3x3 kernel size) having batch normalization at each stage followed by Relu activations at different *epochs = 10 (left) and epochs = 100 (right)*.

2.5 Testing the learned model

At test time, when we want to generate new samples, we simply input values of $z \sim N(0, I)$ into the decoder. That is, we remove the "encoder," including the multiplication and addition operations that would change the distribution of z.

3 Extended VAE - Beta VAE

If each variable in the inferred latent representation is only sensitive to one single generative factor and relatively invariant to other factors, we will say this representation is disentangled or factorized. One benefit that often comes with disentangled representation is good interpretability and easy generalization to a variety of tasks.

β-VAE[2] is a modification of Variational Autoencoder with a special emphasis to discover disentangled latent factors. Following the same incentive in VAE, we want to maximize the probability of generating real data, while keeping the distance between the real and estimated posterior distributions small (say, under a small constant The loss function of β-VAE is defined as:

$$L_{\beta}(\phi, \beta) = -E_{z \sim Q} logP(X/z) + \beta D[Q(z|X) \parallel P(z)] \tag{7}$$

When $\beta = 1$ it is same as VAE. When $\beta > 1$ it applies a stronger constraint on the latent bottleneck and limits the representation capacity of z . For some conditionally independent generative factors, keeping them disentangled is the most efficient representation. Therefore a higher β encourages more efficient latent encoding and further encourages the disentanglement. Meanwhile, a higher may create a trade-off between reconstruction quality and the extent of disentanglement.

4 Extended VAE - Conditional VAE

Conditional Variational Autoencoder (CVAE) is an extension of Variational Autoencoder (VAE). We have no control on the data generation process on VAE. This could be problematic if we want to generate some specific data. As an example, suppose the user inputted character '2', how do we generate handwriting image that is a character '2'? We couldn't. Hence, CVAE [14] was developed. Whereas VAE essentially models latent variables and data directly, CVAE models latent variables and data, both conditioned to some random variables. To condition on additional inputs, we do not need do any derivation, We just need to condition all the distribution with another variable c. Hence our objective is modified as

$$E_{X \sim D}[logP(X|c) - D[Q(z|X,c) \parallel P(z|X,c)]] = E_{X \sim D}[E_{z \sim Q}[logP(X|z,c)] - D[Q(z|X,c) \parallel P(z,c)]] \tag{8}$$

5 Experiment and Results

We implemented vanilla VAE using 2 convolution layer networtk and observed the change in latent distribution during the training. As shown in 2, As Epoch increases, we see that model inherently find similarity among data points belonging to similar class. We observe that latent distribution is approaching towards zero mean and unit variance from epochs = 100 (right) We also experimented with the latent capacity of the model by trying latent vector of length = 2 and 23. As shown in 3, we see as latent capacity of model is increased, the latent vectors shows fewer overlap.

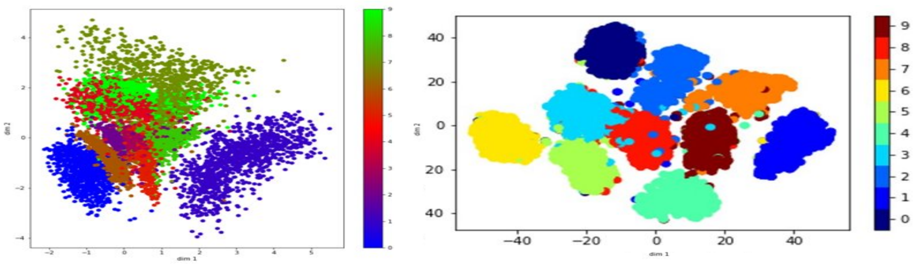


Figure 3: (Left) shows the latent distribution with 2 latent vector. (Right) shows latent distrbution with 23 dimension vector

We combined MNIST and A-Z Handwritten Alphabets Dataset to create Alpha-numeric dataset and used it to train a Conditional Variational Autoencoder network. we constructed CVAE as convolutional encoder-decoder architecture with 2 layers of Convolutional, Pooling Layer followed by Dense layer. Architecture is simialar to vanilla VAE, 2 dense output are taken from encoder as mean and variance latent vector which are then sampled using reparametrization trick before feeding to decoder. Decoder has additional label information as an input to allow conditioning during data generation inference.

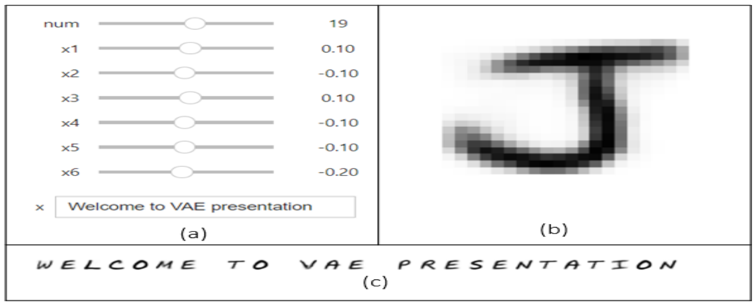


Figure 4: [a] x1 to x6 are the latent vectors, num is slider for label [0-9 -> numeric, 10-35 -> Alphabets (A-Z)], x is the text input to generate sentences. [b] It is the generated alphabet when num is set to 19, varying latent slider, different version of J are generated. [c] It is the generated text output using the text editor input and latent vector

As shown in 4, we have more control on data generation process with inclusion of label information. We provide label information either with num slider or as text box input, along with x1-x6 latent vector the decoder generate new samples which vary based on the change in latent vector in style, boldness, angle etc.

Similarly, we extended Conditional VAE to Beta VAE by varying the beta parameter in loss function. We experimented with differnt values of $\beta = 1e-3, 1e-2, 1e-1, 1, 10, 100, 1024$ as shown in 5. We observed that as beta decreases, reconstruction loss become more dominant and better reconstruction output is obtained whereas higher value of beta encourages learning a disentangled representation of latent vector. The extra pressures coming from high values, however, may create a trade-off between reconstruction fidelity and the quality of disentanglement within the learnt latent representations.

Also as shown in 6, with increase in the beta, the variance of the latent vector shrinkages as also evident from 5.

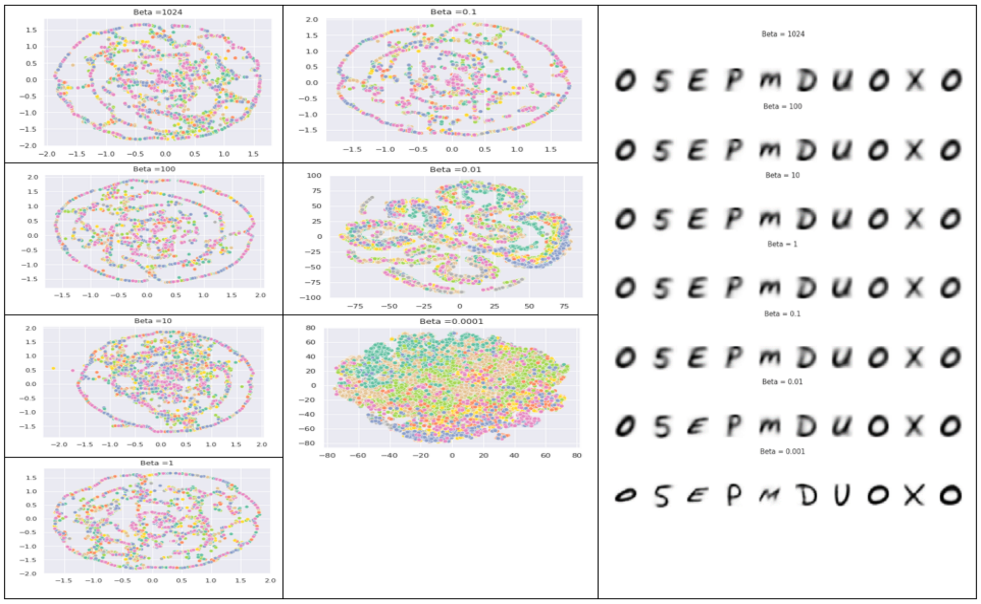


Figure 5: (Left Side) shows tsne plot of latent vector for various beta. (Right Side) is the image generated for corresponding beta

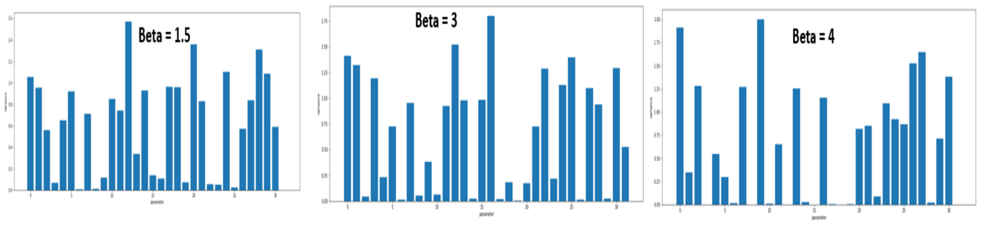


Figure 6: Comparision of variance in parameters for different beta to select paratmeters with high variance.

6 Conclusion

We experimented with vanilla VAE and observed how latent distribution changes as training progress. we also observed the how latent capacity of model can improve the models. To overcome the limitation of generating data based on choice, We experimented with conditional VAE and generated sentences and number by sampling from the distribution and providing label information. Post we observed the effect of Beta VAE on conditional VAE where with increase in beta, model encourages towards disentangled representation of latent vectors. Overall, we infer from our study that VAE are very powerful generative networks and also aids in modelling the distribution through latent varible model.

References

[1] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[2] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," 2016.

[3] H. Kim and A. Mnih, "Disentangling by factorising," *arXiv preprint arXiv:1802.05983v3*, 2019.

[4] S. Liu, J. Liu, Q. Zhao, X. Cao, H. Li, D. Meng, H. Meng, and S. Liu, "Discovering influential factors in variational autoencoders," *Pattern Recognition*, vol. 100, p. 107166, 2020.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[6] L. Mescheder, S. Nowozin, and A. Geiger, "Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks," in *International Conference on Machine Learning*, pp. 2391–2400, PMLR, 2017.

[7] H. Huang, Z. Li, R. He, Z. Sun, and T. Tan, "Intro vae: Introspective variational autoencoders for photographic image synthesis," *Neural Information Processing Systems*, 2018.

[8] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Factor analysis, probabilistic principal component analysis, variational inference, and variational autoencoder: Tutorial and survey," *arXiv preprint arXiv:2101.00734*, 2021.

[9] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," *Advances in neural information processing systems*, vol. 29, 2016.

[10] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[11] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," in *International Conference on Machine Learning*, pp. 1462–1471, PMLR, 2015.

[12] X. Wang, Y. Du, S. Lin, P. Cui, Y. Shen, and Y. Yang, "advae: A self-adversarial variational autoencoder with gaussian anomaly prior knowledge for anomaly detection," *Knowledge-Based Systems*, vol. 190, p. 105187, 2020.

[13] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[14] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," *Advances in neural information processing systems*, vol. 28, 2015.