Indian Institute of Technology, Delhi

Autoencoders

Gauri Gupta

Supervisor: Prof. B.S. Panda

A report submitted in fulfillment of the requirements of

MTL782 - Data Mining *in the*

Department of Mathematics

August 29, 2020

Abstract

Autoencoders are unsupervised learning methods in machine learning which can automatically learn and extract patterns and features from a large set of unlabeled data. Due to its simple structure and easy functionality, autoencoders form a backbone in the field of generative modelling and have received a lot of recent attention in research community. Many different variants of autoencoders have been proposed and studied by various researchers which have found immense applications in many fields, such as image classification, pattern recognition, anomaly detection, and data generation. In this report we present our comprehensive study on autoencoders, its various variants and future research prospects that can be explored. We first introduce the fundamental theory and framework of auto-encoder methods, along with some of its applications. Next, we discuss some of the variants of the autoencoders that find great applications in deep learning methods and have deeply impacted the research on generative modelling. In the last section of the report, we discuss state-of-the-art literature where autoencoders can be combined with more recent flow-based models to achieve better density estimation and image generation.

1 Introduction

Generative models are unsupervised machine learning algorithms that automatically discover and learn the patterns and regularities in input data in such a way that the model can be used to generate similar images. Suppose, given a set of input data X and a corresponding set of labels Y, the generative model tries to capture the full joint probability distribution P(X, Y) or at only times P(X) when the labels are not provided. Images are the most common datasets that are fed to these models to generate similar looking pictures. Generative modelling thus have an extremely challenging task of capturing the dependencies in the input data within a very high-dimension space. In the case of images, these models might capture information like pixel correlation, nearby pixels having same colour or features etc. Due to their advantage of capturing the underlying distribution and patterns in the data, deep generative models have attracted much recent attention in the machine learning literature.

Generative models can be roughly classified into two categories of implicit and explicit density estimation models. The first category of implicit models does not perform specific parametrisation of the underlying distribution and are dominated by Generative Adversarial Networks (GAN) while the second category of explicit models directly model the P(X), the distribution of the data. Some of these explicit density models are auto-regressive models, normalising flow models and autoencoders-based models.

- **GANs** are based on adversarial training between a generator and discriminator which play a mini-max game. GANs are famous for generating high quality images but training GANs is very unstable making it challenging at times. They also suffer from issues such as mode "collapse". This renders them less than useful in a variety of applications where different models are preferred instead.

- **Autoregressive models** use information from the previous time steps to predict the value at

the next time step. They have the advantage of simplicity, but the synthesis has limited parallelizability as the computational length of synthesis is proportional to the dimensionality of the data. This serves as a disadvantage for large images or video.

- **Normalising flow models** are reversible transformations from a simple latent space distribution to complex distribution of the image space and extract the exact likelihood of the data distribution. However, these models require a lot of memory and are computationally very heavy for the system.

- Another type of generative models exploit the information stored in a low dimensional latent code representation from a high dimensional input data obtained from autoencoders and generate samples by decoding the latent code sampled from a pre-defined prior distribution. These models are called **Autoencoder (AE) based models**. AE based models are easy to train and provide low dimensional representation of the high-res input data.

Autoencoders are a special type of feed forward neural networks that perform representation learning. They compress the given input data into a lower dimensional vector with a narrow bottleneck layer in the middle and reconstruct the output from this encoded representation. Not only it generates a reconstructed image but also helps in dimensionality reduction. The bottleneck layer stores most of the information of the input data in a compressed latent representation. Autoencoders which are mainly used for dimensionality reduction have the following properties:

Data-specific: Autoencoders are only able to successfully compress data similar to what they have been trained on but they don't give desired results if tested on out-of-distribution data. Since they are trained to learn features specific to a given data distribution, they perform differently than a standard data compression algorithm. An autoencoder trained on pictures of faces (CelebA dataset) would do a poor job of compressing handwritten digits (MNIST dataset), because it learns the features specific to faces dataset. Hence out-of-distribution compression makes no sense.

Lossy: Autoencoders are mostly used for compression algorithms but the reconstructed output of the autoencoder is not exactly same as the input but close enough. Autoencoders are lossy that means some information loss happens while storing the information in the encoded vector of the bottleneck layer.

Unsupervised: Training an autoencoder is an unsupervised learning task that train automatically and do not require specific labels to train on making it very easy to train such models.

2 Architecture of Autoencoders

Autoencoders are an unsupervised learning process that let you take advantage of the unlabelled data and learn things about the structure of the data for various different kinds of applications. Autoencoders can be used for classification, dimensionality reduction, feature extraction, anomaly detection, and to fill missing and generate new values.

Autoencoders are type of a feed forward neural network that attempt to do 2 things:

- Compresses input data into a lower dimension (known as latent code representation)

- Use this lower dimensional representation of the data to recreate the original input

The reconstruction error is defined as the measure of difference between the attempted recreation and the original input:

$$\boxed{\text{Reconstruction Error} = \text{Reconstructed} - \text{Original}}$$

Autoencoders are designed to learn to exploit the natural structure of the input data to find an efficient and an almost-information preserving lower dimensional representation. They are designed to typically optimize their working to minimize this reconstruction error. This basic principle of their working mechanism is then exploited in a myriad of ways (see next section).

Autoencoders have 3 main components which can be visualized as sequential layers:

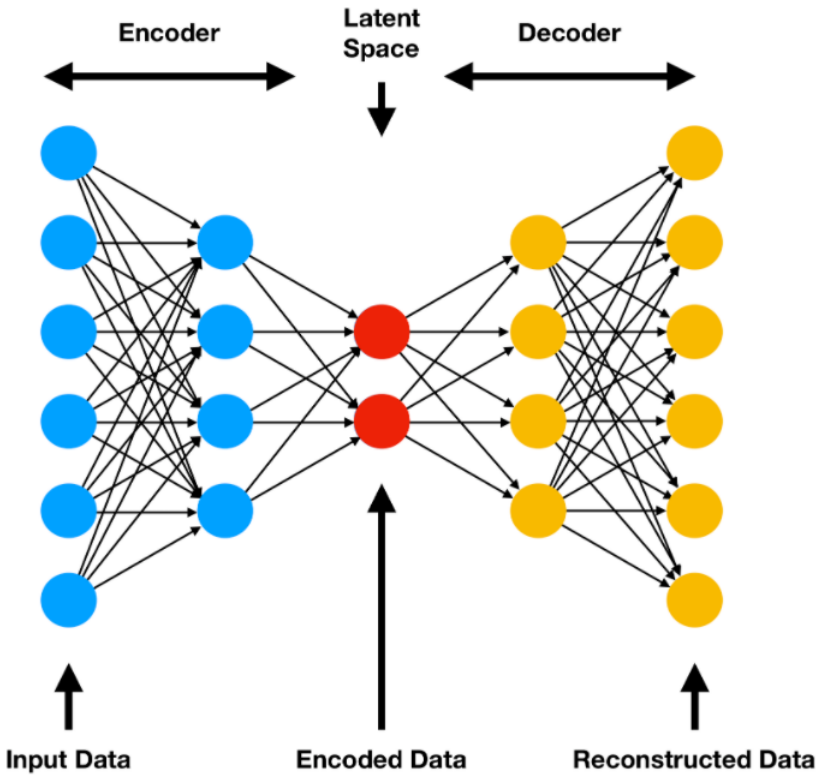

Figure 1: Overview of the VAE architecture

- **Encoder:** The job of the encoder part is to transform from the original input into a meaningful lower dimensional representation (which we named latent space representation). The encoder approximates the function that maps the data from the input space into a lower dimensional coordinate system that takes advantage of the structure of our data.
- **Decoder:** The decoder attempts to recreate the original input using the output of the previous layers. That is, it tries to reverse the encoding process. The decoder tries to convert a lower-dimensional data distribution into higher-dimensional data.

- **Bootleneck/Code:** The job of this middle layer is to make data even more compressed and smaller than what we obtained from the Encoding layer.

The bottleneck layer forces Information Loss. This layer thus feeds "imperfect" information into the decoder. By making the whole network minimize the reconstruction error, we force the encoder and decoder to work in tandem with each other to find the most efficient way to condense the data into a lower dimension while still ensuring reconstruction by the decoder (with some qualifications) at an acceptable level. Thus we try to tweak the network to optimize this hidden layer according to the specific function we need the network to perform,

3 Applications of Autoencoders

- **Feature Learning & Dimensionality Reduction**: Hinton et al. in 2006 [5], used a Restricted Boltzmann Machine (which is like a 2-layered autoencoder) and autoencoders to reduce dimensionality of data. He showed that using AutoEncoders,it was possible to reduce dimensionality of non-linear type of data. The patterns he obtained were much better than that obtained with PCA.
  If the activation function is linear, the autoencoder finds the direction of the maximum energy in the input (finds the variance of the input if the input is a zero mean random variable). Looking at just the decoder portion of an AutoEncoder, the decoder takes as input some variable (say z) which is an output of a neuron. Varying this z over its domain, this reconstructed output always lies on the major axis (the direction of maximum energy) which is a single line. If such an AutoEncoder only has a single later, it behaves exactly like PCA.
  Using an autoencoder with multiple layers and non-linear activation functions, we can capture the non linear behavior of the data, which isn't possible using techniques like PCA.
  But this improved functionality comes at the cost of more time and consequently more resources spent in training such an AutoEncoder.

- **Anamoly Detection**: Anamoly detection is relevant in detecting fraudulent cases with applications in fields such as the banking sector (credit card fraud etc.), telecom industry (SIM Swap), retail sector etc. The AutoEncoder is trained on a "normal" data set, so that during real-world application, when the same pattern doesn't work, then the model flags that as an "anamoly". This framework of learning the normal instead of the the outlier definition can also help in detecting in new kinds of anamolies etc.
  In an AutoEncoder model, reconstruction errors are evaluated. Setting a threshold for the reconstruction errors, one can locate the anamolies (which are chosen based on training set) and the points exceeding this reconstruction error are flagged as outliers or anamolous points.

- **Denoising**: Denoising has overwhelming applications in "image denoising" which is relevant in the medical field and restoring digital movies and images etc. In the medical field, experiments are being done to detect breast cancer using AutoEncoder models etc.

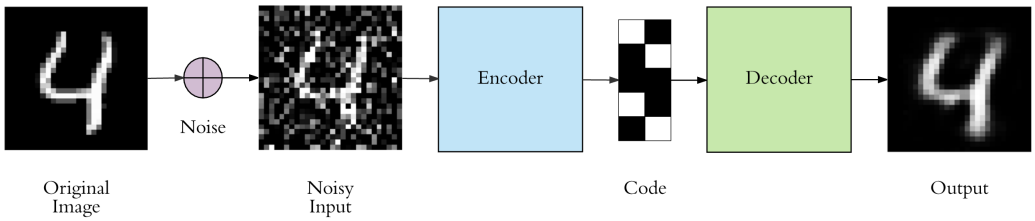

Figure 2: Denoising Images using VAE

Denoising AutoEncoders achieve denoising by changing the reconstruction criterion. A noisy variant of the actual input is fed into the AutoEncoder (like random assignment, or Guassian additive noise etc.). The AutoEncoder is fed this noisy variant as it's training data. But the reconstructed data is compared with the noiseless data (that is the loss function is compares the output to the noiseless variant). Thus the AutoEncoder is made to learn about the relationship between the noiseless input and the noisy input. Thus by adding noise, we force the AutoEncoder to learn about the noisy distribution and at the same time, the AutoEncoder learns more meaningful features (the ones subjected to less noise).

- **Image Generation**: Applications such as generating new images which have not yet been modeled based on given images, image colorization and inpainting all involve image generation. Image Inpainting can be understood as image generation with constraints. The AutoEncoder is supposed to take the image with a "hole" as input and output an image that fills this missing hole. Variational AutoEncoder which is a Generative Model that we will discuss below is often used to construct and generate images that don't yet exist.

- **Recommender Systems**: AutoEncoders have shown better performance and adaptability than traditional models for recommender systems. A variety of variants of AutoEncoders have shown potential to being applied widely in Recommendre Systems. They make the recommender to be more adaptable spatially and temporally and due to the nature of AutoEncoders (as explained in applications above) makes it more resistant to noise compared to many other available mechanisms.

4 VAE and its variants

In this section we discuss a variety of deep generative models, (which as the name suggests, are variants or improvements of AutoEncoder) the **Variational Auto-Encoder** (VAE) and its sub-variants such as the Beta-VAE, Factor-VAE and the TD-VAE. The theory and working of the VAE forms the basis of discussion of all its sub-variants. Therefore, we will first discuss the general Variational Autoencoder. Subsequently, we will investigate the above mentioned variants of VAE which have evolved over time and have greatly impacted the continuous ongoing research on Autoencoders.

4.1 VAE: Variational Autoencoder

Variational Autoencoders derives its design and conception from the methods of variational Bayesian method and graphical models. The intuition behind the design of a Variational Encoder is this:

Map the input data into a probability distribution

Traditionally, the input data is mapped into fixed vectors but mapping the input into a probability distribution makes the data more flexible and thus the distribution can be adapted to our use. This expands the usability of the data and what we can extract from it. This idea looks deceptively simple at the first glance but as is the case with most of new ideas, it's impact lends itself to the design of the VAE here making it a redoubtable model in data science.

We will the distribution that the input is being mapped to as p_θ , (parameter θ) and we define:

- "Prior" to be $p_\theta(z)$
- "Likelihood" to be $p_\theta(x|z)$
- "Posterior" to be $p_\theta(z|x)$

We suppose that the real parameter θ^* is known for the real distribution. Our objective is to get a sample that looks like x(i) (that is, it has the same distribution as an actual data point):

1. Take a prior distribution $p_{\theta^*}(z)$, to sample $z\,(i)$
2. After this, from a conditional distribution $p_{\theta^*}\,(x|z = z\,(i))$, a value $x(i)$ is generated

We define θ^* as the parameter that generates real data samples with maximum probability

$$\boxed{\theta^* = arg\max_\theta \prod_{i=1}^{n} p_\theta\,(x\,(i))}$$

For convenience, as is it is usually done, we convert the above log product into a summation. Thus redefining the expression for our optimal parameter using the log probabilities we have:

$$\boxed{\theta^* = arg\max_\theta \sum_{i=1}^{n} \log p_\theta\,(x\,(i))}$$

The following equation to demonstrates the data generation process so that the encoding vector can be taken in account :

$$\boxed{p_\theta\,(x\,(i)) = \int p_\theta\,(x\,(i)\,|z)\,p_\theta\,(z)\,dz}$$

As computing $p_\theta(x(i))$ iterating over all values of z and then summing all of them is very costly and infeasible, we need to find a faster and a more efficient search mechanism to find the optimal value. For this purpose, the search space needs to be narrowed down. To fulfill this purpose, we introduce an approximation function $q_\varnothing(z|x)$, parameterized by $\varnothing$.

Now one can look at it as an autoencoder:

- The likelihood $p_\theta(x|z)$ is a generative model, which is the *probabilistic decoder*.
- The probabilistic encoder is the approximation function $q_\varnothing(z|x)$.

Loss Function: ELBO

The estimated posterior $q_\varnothing(z|x)$ needs to be as similar to the real one $p_\theta(z|x)$ as possible.The Kulback-Leilber Divergence, often used in Machine Learning and Information Theory is a measure of distance between the two distributions that is to find out how much one probability distribution is different from another. As we know from the theory of Information Theory, the KL divergence $D_{KL}(X||Y)$ quantifies the loss in information if one uses Y to denote X.

In this case $D_{KL}(q_\varnothing(z|x)||p_\theta(z|x))$ is to be minimized with respect to $\varnothing$.

$D_{KL}(q_\varnothing||p_\theta)$ (reversed KL) is used instead of $D_{KL}(p_\theta||q_\varnothing)$ (forward KL). This is because:

- Forward KL divergence: $D_{KL}(P||Q) = E_{z\sim P(z)} \log \frac{P(z)}{Q(z)}$; such that $Q(z) > 0$ wherever $P(z) > 0$. So that the whole of $P(z)$ is covered by the optimized variational distribution $Q(z)$.
- Reversed KL divergence: $D_{KL}(Q||P) = E_{z\sim Q(z)} \log \frac{Q(z)}{P(z)}$; squeezes the $Q(z)$ under $P(z)$ which is due to the minimization of KL.

Arranging the equation as follows

$$= \log p_\theta(x) - D_{KL}(q_\varnothing(z|x)||p_\theta(z|x)) = E_{z\sim q_\varnothing(z|x)} \log p_\theta(x|z) - D_{KL}(q_\varnothing(z|x)||p_\theta(z))$$

The objective here is to maximize the left side of the equation while simultaneously learning the true distributions: i.e. to bring the real and estimated posterior distributions as close as possible (D_{KL} is a regularizer here) and, increase the (log-)likelihood of generating real data. The distribution $p_\theta(x)$ is kept fixed w.r.t. $q_\varnothing$.

The negation of the above defines the loss function:

$$\begin{aligned} L_{VAE} &= -\log p_\theta(x) + D_{KL}(q_\varnothing(z|x)||p_\theta(z|x)) \\ &= -E_{z\sim q_\varnothing(z|x)} \log p_\theta(x|z) + D_{KL}(q_\varnothing(z|x)||p_\theta(z)) \end{aligned}$$

$$\theta^*, \varnothing^* = arg\min_{\theta,\varnothing} L_{VAE}$$

In Variational Bayesian methods, what we generally refer to as the *evidence lower bound* is nothing but just the loss function we defined above:

$$\begin{aligned} -L_{VAE} &= log p_\theta(x) - D_{KL}(q_\varnothing(z|x)||p_\theta(z|x)) \\ &\leq \log p_\theta(x) \end{aligned}$$

Therefore the observation is that lower bound of the real data generated is maximized as the loss gets minimized.

Reparameterization Trick

The expectation term in above equation takes in generating samples from $z \sim q_{\varnothing}(z|x)$. The gradient cannot be backpropagated due to the process of sampling being stochastic . Reparameterization is done so as to make it trainable: Sometimes the random variable z is represented as $z = T_{\varnothing}(x, \epsilon)$, which is a deterministic variable and ϵ being an auxiliary independent random variable. The conversion from ϵ to z is done by the transformation function $T_{\varnothing}$ parameterized by $\varnothing$.

Mutivariate Gaussian is a common choice for $q_{\varnothing}(z|x)$ with a diagonal covariance structure:

$$\begin{aligned} z &\sim q_{\varnothing}(z|x(i)) \\ &= N\left(z; \mu(i), \sigma^2(i) I\right) \end{aligned}$$

Therefore, we can write z as

$$\boxed{z = \mu + \sigma \odot \epsilon} \quad \text{where } \epsilon \sim N(0,1) \text{ and } \odot \text{ is the element wise product}$$

In the multivariate Gaussian case, the reparameterization trick helps the model learn parameters of distribution, μ and σ , while the stochasticity is in the random variable $\epsilon \sim N(0,1)$.

4.2 Beta-VAE

β -VAE (Higgins et. al. 2017) talks about a variant of VAE which in particular focuses on disentanglement. The motivation comes from the fact that suppose given a set of human faces on which our model is trained it may be possible that it picks up certain traits such as the skin color, particular expressions and other factors which maybe completely independent in separate dimensions. So β -VAE , what it does is induces a hyper-parameter in the form of β . This in turn puts a weight on the KL in the ELBO term.

-VAE loss function is given as:

$$\boxed{L_{\beta VAE}(\varnothing, \beta) = E_{z \sim q_{\varnothing}(z|x)} \log p_{\theta}(x|z) - \beta D_{KL}(q_{\varnothing}(z|x) \| p_{\theta}(z))}$$

For different values of β , we see :

- $\beta = 1$, will give us the VAE itself.
- $\beta > 1$, limits the representational capacity of z . Keeping it disentangled helps it to capture the conditionally independent generative factors. Therefore a higher value of β is preferred. Also a higher β may lead to find a balance among reconstruction quality and the extent of disentanglement.

The training objective therefore enhances the learning of the distribution $q_{\varnothing}(z|x)$ about the data point x by minimizing β -KL term and maximizing the data log likelihood. Although the images generated maybe blurry due to the fact that weighting the KL-term would minimize the flexibility of posterior distribution.

4.3 FACTOR-VAE

One of the main problems of β -VAE is that it has poor reconstruction quality. The main motivation of Factor-VAE is to cope up with the reconstruction quality. This is done by introducing a penalty term to the VAE objective which helps to learn the representations as factorial. Therefore it is independent across dimensions and hence not degrading the reconstruction quality. A factoral distribution is basically a joint distribution of random variables which are independent of each other. Therefore a factorized distribution provides a stronger disentanglement among the learned features in the latent representation.

Therefore, given a data point x the posterior over z for all of X is $q(z) = \frac{1}{N}\sum_{i=1}^{n} q\left(z|x^i\right)$

So, the Factor-VAE objective turns out to be an augmentation of the original VAE :

$$\boxed{E_{z\sim q_{\varnothing}(z|x)}\left[\log p_\theta(x|z) - D_{KL}\left(q_{\varnothing}(z|x)\,\|p_\theta(z)\right) - \gamma D_{KL}\left(q(z)\,\|q^*(z)\right)\right]}$$

$$\text{where } q^*(z) = \prod_{j=1}^{d} q\left(z_j\right)$$

The extra term indicates the correlation as it is intractable one needs to approximate it. This is done using the density ratio trick. Which uses a classifier D to classify between samples drawn from each so as to get the density ratio of the divergence.

So as in theory we see that the Factor-VAE is in improvement to the above β -VAE as in incorporating full independence among features it helps reconstruction quality as in compared to the blurry images from the earlier variant as $q^*(z)$ is the factors distribution as it takes in each of the factors independently.

4.4 TD-VAE

Temporal Difference VAE(Gregor et al., 2019) arises from the fact that in order to make predictions about the future one must take the past into account and act optimally.So therefore a model needs to have a belief state. While a model also needs to predict distant future without taking into account each and every time step. For this the authors introduce something known as jumpy predictions. This means that the model is able to predict several states beforehand while learning from separated time steps without going back through the entire time interval. Therefore, we mention the three main modifications which TD-VAE brings in comparison to the original VAE.

1. **State-Space Model**
   In these models, the unobserved sequential hidden states $z = (z_1, \ldots.., z_T)$ determine the

observation states $x = (x_1, \ldots.., x_T)$. So each time step in the Markov chain model can be trained similarly, to approximate the intractable posterior $p(z|x)$ by a function $q(z|x)$.

2. **Belief State**
   $b_t = belief(x_1, \ldots.., x_t) = belief(b_{t-1}, x_t)$, is that the model should encode the past states so as to be able to predict the future. Therefore, the distribution of future states can be written conditined on past as $p(x_{t+1}, \ldots.., x_T | x_1, \ldots.., x_t) \approx p(x_{t+1}, \ldots.., x_T | b_t)$. So the hidden states in a recurrent way are taken as the belief state in TD-VAE. Giving $b_t = RNN(b_{t-1}, x_t)$.

3. **Jumpy Prediction**
   A model should foresee separate futures from all the information gathered, giving the possibility of jumpy predictions, i.e., predicting states several steps ahead.

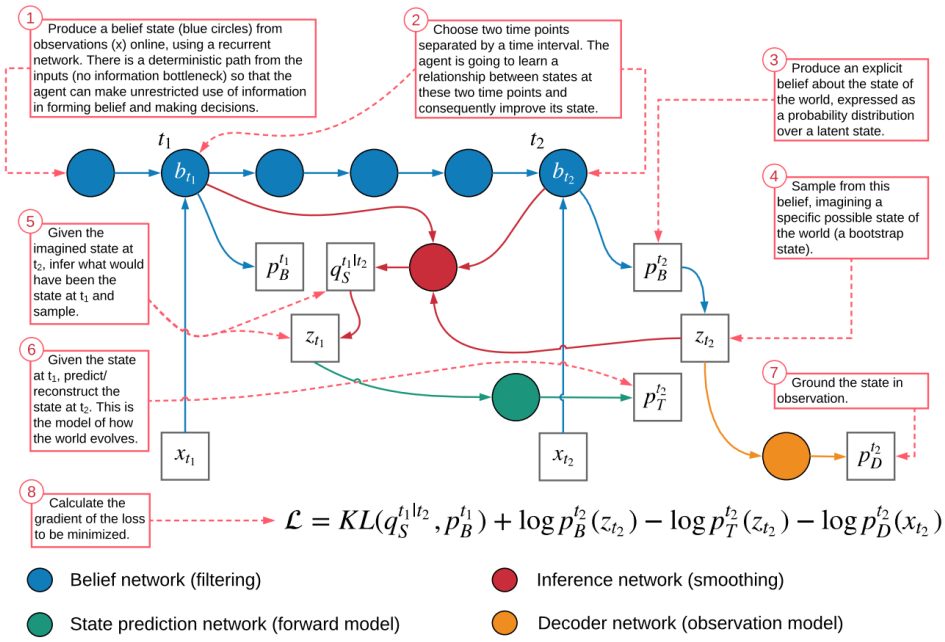

Figure 3: An overview of the TD-VAE architecture

To bring in the idea of jumpy prediction, the sequential ELBO has to function on $t, t+1$, as well as two separated timestamp $t_1 < t_2$. The final TD-VAE function which needs to be maximized is given as :

$$L_{t_1,t_2} = E\left[log\ p_D(x_{t_2}|z_{t_2}) + log\ p_B(z_{t_1}|b_{t_1}) + log\ p_T(z_{t_2}|z_{t_1}) - log\ p_B(z_{t_2}|b_{t_2}) - log\ q_S(z_{t_1}|z_{t_2}, b_{t_1}, b_{t_2})\right]$$

The intuition to this comes fro the fact that suppose till time t_1 the entire belief of the world is take in b_{t1} . And suppose x_t is the observed state at time step t given z_t which captures the whole information at t . So at time t_2 the current state of the world can be guessed by sampling from

$p_D(x_{t_2}|z_{t_2})$. Therefore the agent would maximize the first term but then it also wouldn't't like too much information from x_{t_2} to be encoded in z_{t_2} . So as to balance this the negative term which is the fourth term is introduced.

The next thing which comes up is given the state of the world at t_1 can it be used to predict the state at time t_2 . To this the authors have suggested that as the agent can aggregate the information till time t_1 combined with the current guess that is z_{t_2} . This is done by the smoothing distribution $q_S(z_{t_1}|z_{t_2}, b_{t_1}, b_{t_2})$. The third term does nothing but optimizing the jumpy prediction between the states z_{t_2} and z_{t_1} . Also it may be seen that if the smoothing distribution could only be predicted using information only at t_1 . To check this simply one can compute the KL between the smoothing term and the belief distribution which accounts for the second and last term.

Summing all of them gives us the loss term of TD-VAE.

As mentioned above the advantage of this is rather than taking a step by step time model which wouldn't't be suitable both cognitively and computationally. Rather an interval based time step model would be more advantageous incorporating all the above features.

5 Combining VAE with other generative models like Normalising flows: state-of-the-art research

Normalizing flows are reversible and invertible transformations used to learn complex image space distribution from simple latent space. These flow-based models have gained a lot of focus due to their exact log-likelihood inference and tractability. They not only provide an efficient inference of the latent space variables but also generate images of high quality.

Decoupling Local and Global Representations for Image Generation
One of the disadvantages of generative flows is their lack of ability of expressiveness as they usually rely on local dependencies for generation of data. Most of the information captured in these models is local pixel co-relation and the model fails to learn the global dependencies thus they don't generate very realistic images in comparison to GANs. But VAE, which belong to the class of latent representation models, have the capability of automatically learning low-dimensional meaningful representations and local level dependencies that usually captures the global features of the input data. This encoded vector is then passed as an input to the invertible decoder of flow based model which then finally generates the reconstructed image.

In this research paper[6], they propose a simple and efficient generative model that tries to simultaneously tackle the problems encountered in VAE and flow based models by leveraging their advantages to compliment each other. Using flow based invertible decoder in the VAE framework, the model is able to learn both local and global information of images simultaneously and separately in a completely unsupervised way, that requires no external supervision.

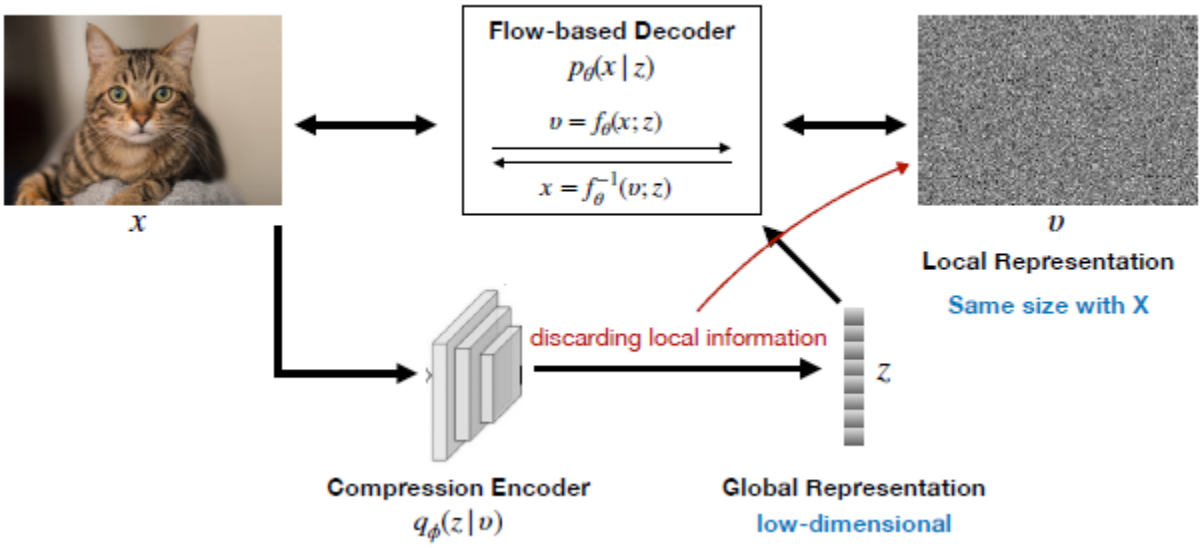

Figure 4: Decoupling local and global representations from an image x

The input image x is fed to the compression encoder of VAE to compress the raw input data x to meaningful low dimensional latent code z. The flow-based decoder captures the local pixel co-relation information in the image and outputs a local representation v, having same dimension as the input image x.
Hence, effectiveness and usage of this model can be understood in terms of disentanglement of local and global information of an image through decoupled representation learning which has many advantages in the field of generative modelling.

6 Research Ideas

While doing a comprehensive analysis of Autoencoders and taking inspiration from the above studied research papers, we came up with some of the following research ideas of our own which have not yet been explored but can potentially help boost the ongoing research in generative modelling.

- The state-of-art flow based generative models like GLOW have gained a lot of recent attention in the research community due to tractability of exact log-likelihood and efficient synthesis. However, one of the main disadvantages of these models is that it learns mainly pixel level correlation and no semantic features are learnt in this model. Given autoencoders have the ability to extract the local-level information of an image in the form an encoded vector, we propose a model where we encourage hierarchical nature in latent space of Glow i.e. enforce that changing lower level of latent spaces brings semantically small changes in the image. This can be achieved using the semantic information stored in the layers of an autoencoder which can be then passed as a sequential input to the latent vectors of GLOW. The architecture of our proposed model would be as follows:

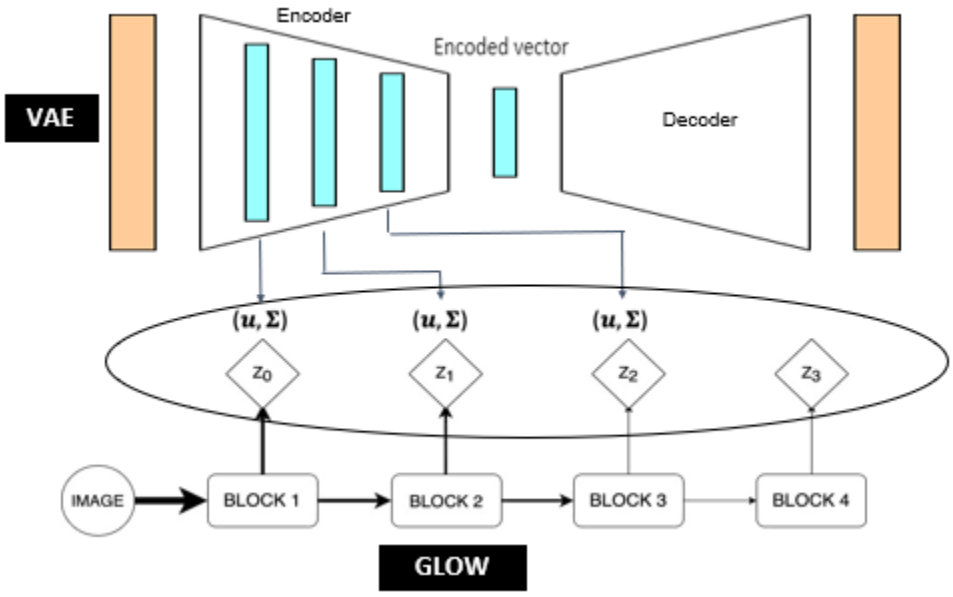

Figure 5: Learning latent space of GLOW latent vector from feature layers of VAE

7 Conclusion

Generative modelling is a continuously evolving domain in machine learning. In today's world where machines cannot learn without being fed with data, generating new and unseen data is the most challenging and important task. Generative models not only try to capture the underlying distribution of the input dataset but also produce new and similar reconstruction of the input data. Autoencoders, a type of generative models, try to compress a high dimensional input data into a low dimensional latent representation. This encoded representation contains most of the information of the input data in a low dimension which is computationally efficient and thus is used for data compression.

References

[1] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes."(2014)

[2] Higgins, Irina, et al. "beta-vae: Learning basic visual concepts with a constrained variational framework." (2017)

[3] Burgess, Christopher P., et al. "Understanding disentangling in beta-VAE."(2018).

[4] Gregor, Karol, et al. "Temporal difference variational auto-encoder."(2018).

[5] Hinton, G.E. Salakhutdinov, R.R.. (2006). Reducing the Dimensionality of Data with Neural Networks. Science (New York, N.Y.). 313. 504-7. 10.1126/science.1127647.

[6] Ma, Xuezhe Zhang, Shanghang Hovy, Eduard. "Decoupling Global and Local Representations from/for Image Generation."(2020)

[7] Diederik P. Kingma, Prafulla Dhariwal. "Glow: Generative Flow with Invertible 1x1 Convolutions."(2020)

[8] Karras, T., Laine, S., Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4401-4410).

[9] https://www.edureka.co/blog/autoencoders-tutorial/

[10] https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798