Introduction

Generative modeling

It 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 X.

For instance, images are a popular kind of data for which we might create generative models

In generative modeling, we get examples X distributed according to some unknown distribution Pgt(X), and our goal is to learn a model P which we can sample from, such that P is as similar as possible to Pgt.

Limitations:

- they might require strong assumptions about the structure in the data.
- they might make severe approximations, leading to suboptimal models.
- they might rely on computationally expensive inference procedures like Markov Chain Monte Carlo

Due to these limitations, VAEs (Variation autoencoders) came they train via fast backpropagation and approximations but that give very small errors.

Latent Variable Models

In this our model has to decide first surely what to output, this decision is called latent variable and what it has outputted is called latent.

Intuition:

X is an image, z is the latent factor used to generate x: attributes, orientation etc,

Before we can say that our model is representative of our dataset, we need to make sure that for every datapoint X in the dataset, there is one (or many) settings of the latent variables which causes the model to generate something very similar to X.

To make this notion precise mathematically, we are aiming maximize the probability of each X in the training set under the entire generative process, according to:

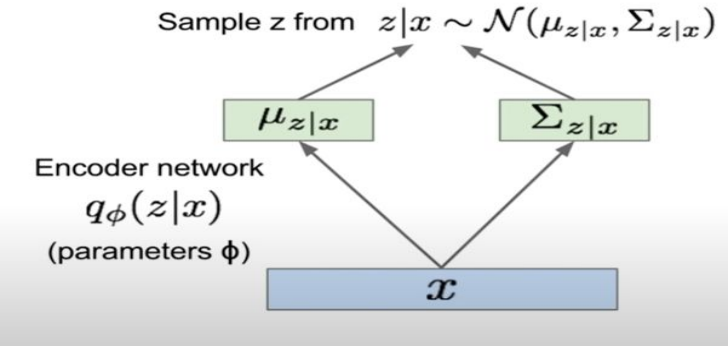
$$P(X) = \int P(X|z;\theta)P(z)dz. \quad (1)$$

Variational Autoencoders(probabilistic generation of data)

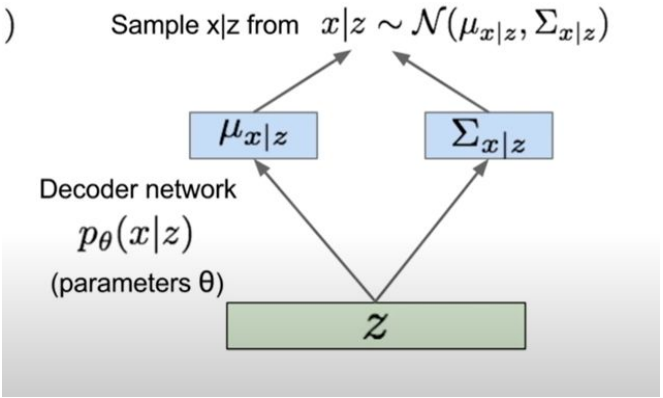
But there are some limitations of equation (1)

- How to decide what latent variables z represent
- How to integrate ie limits

To solve the first problem we first have z vector to be a normal distribution in every dimension as afterward, we can map it to any distribution we want.
Here P(X|z;theta) is a neural network function.
As the integral is intractable we need a new encoder distribution q that encodes X datasets to and outputs a normal distribution with some mean and standard deviation.



Now, we can sample some z from this distribution and feed it to our decoder.



Now, we have to maximize the P(X) ie log(P(X)) we can approximate this by multiplying it by expectation and doing some calculation as shown:

$$
\begin{aligned}
\log p_\theta(x^{(i)}) &= \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta(x^{(i)})\right] && (p_\theta(x^{(i)}) \text{ Does not depend on } z) \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\right] && \text{(Bayes' Rule)} \\
&= \mathbf{E}_z\left[\log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})}\frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})}\right] && \text{(Multiply by constant)} \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})}\right] && \text{(Logarithms)} \\
&= \mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z)) + D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z \mid x^{(i)}))
\end{aligned}
$$

First-term: This is provided by our decoder network. We can approximate this term by sampling.
Second-term: This has two Gaussian distribution, so we can easily take out its solution.
Third-term: This is the error term that is still intractable, but this term is always greater than 0.
So we can maximize our probability by maximizing our lower bound(first term+second term) and then finding theta and phi.