UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Divija L
Teaching Assistant

PES UNIVERSITY

VAE- Variational Autoencoders

"Why generative models" take-aways:

1. **Training data:**

- **Unsupervised/Semi-supervised learning: More training data available**
- **Any energy-based unsupervised learning method can be seen as a probabilistic model by estimating the partition function**

- **E.g. all of the videos on YouTube**

1. **Many right answers**

- **Caption -> Image**

https://openreview.net/pdf?id=Hyvw0L9el

A man in an orange jacket with sunglasses and a hat skis down a hill

"Why generative models" take-aways:

PES UNIVERSITY

3. Outline -> Image

https://arxiv.org/abs/1611.07004

"Why generative models" take-aways:

PES UNIVERSITY

4. Intrinsic to task

https://arxiv.org/abs/1609.04802

Example: Super resolution

VAEs

- Just as autoencoders can be viewed as performing a non-linear PCA,variational autoencoders can be viewed as performing a non-linear Factor Analysis (FA)

- Variational autoencoders (VAEs) get their name from variational inference, a technique that can be used for parameter estimation.

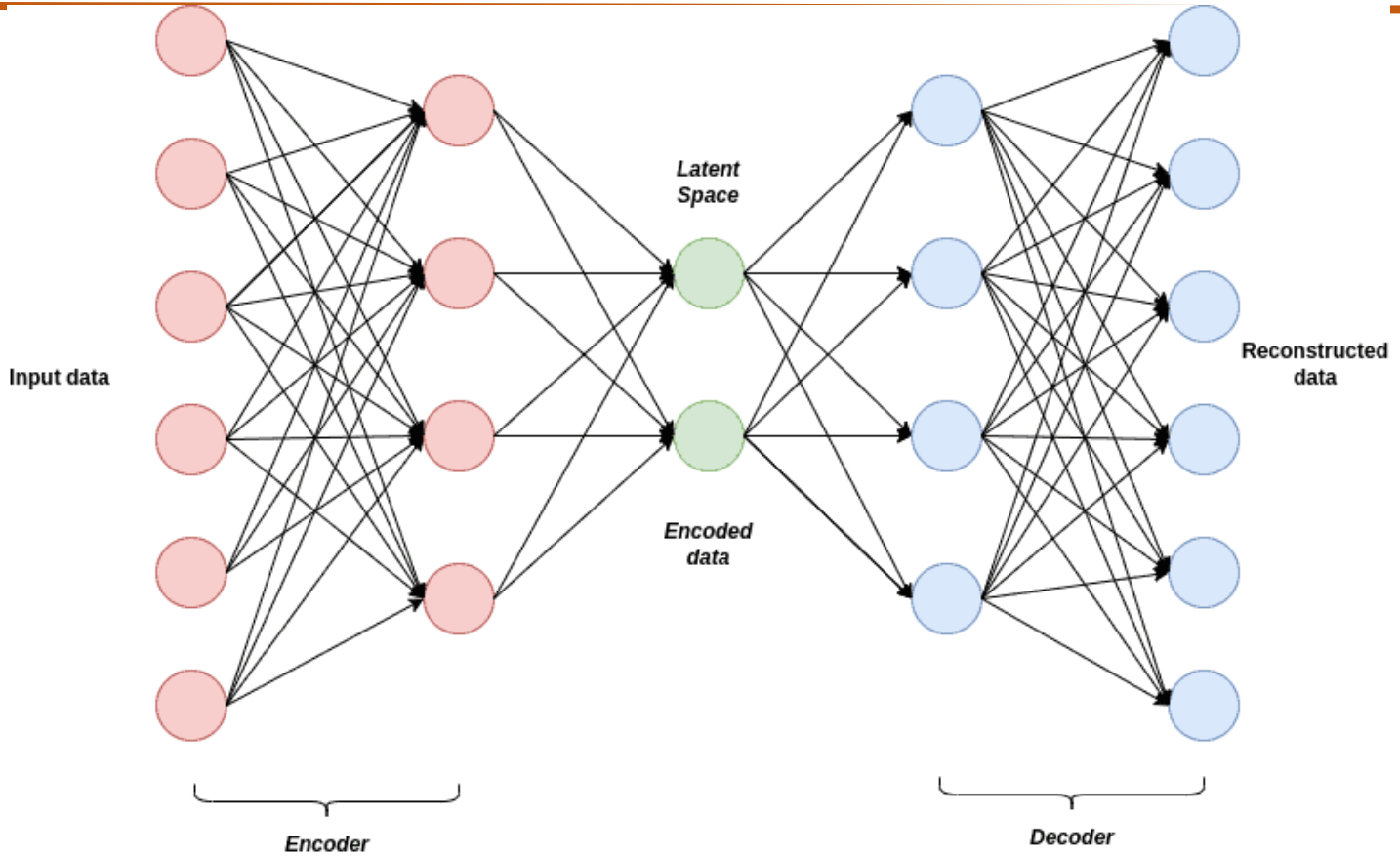
- We will introduce Factor Analysis, variational inference and expectation maximization, and finally VAEs

VAEs

- Latent variable models form a rich class of probabilistic models that can infer hidden structure in the underlying data.
- Variational autoencoders (VAEs) were defined in 2013 by Kingma et al. and Rezende et al..
- Unlike traditional autoencoders, VAEs learn a probabilistic latent space representation of input data, enabling them to generate new data samples.
- These models also yield state-of-the-art machine learning results in image generation and reinforcement learning.

Let's think about them first using neural networks, then using variational inference in probability models.

VAEs

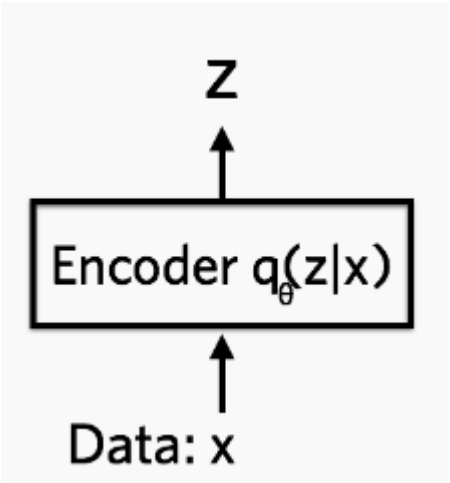

Input data
Latent Space
Encoded data
Reconstructed data
Encoder
Decoder

- The prior on the latent space:
    - Allows you to inject domain knowledge
    - Can make the latent space more interpretable

- The VAE also makes it possible to estimate the variance/uncertainty in the predictions

VAEs Architecture: Neural Perspective

In neural net language, a variational autoencoder consists of an encoder, a decoder, and a loss function.



The *encoder* is a neural network. Its input is a datapoint x, its output is a hidden representation z, and it has weights and biases θ.

VAEs Architecture: ENCODER

To be concrete, let's say x is a 28 by 28-pixel photo of a handwritten number. The encoder 'encodes' the data which is 784-dimensional into a latent (hidden) representation space z, which is much less than 784 dimensions.

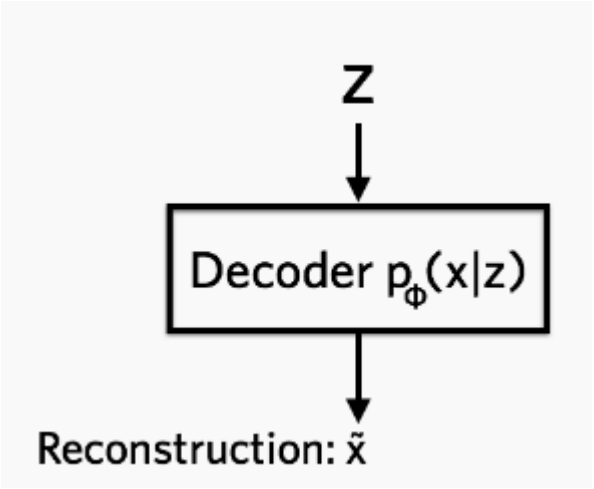
This is typically referred to as a **'bottleneck'** because the encoder must learn an efficient compression of the data into this lower-dimensional space.

Let's denote the encoder $q_\theta(z|x)$. which is a Gaussian probability density.

VAEs Architecture: DECODER

The *decoder* is another neural net. Its input is the representation z, it outputs the parameters to the probability distribution of the data, and has weights and biases ϕ.

The decoder is denoted by $p_\phi(x|z)$.



VAEs Architecture: DECODER

Running with the handwritten digit example, let's say the photos are black and white and represent each pixel as 0 or 1.

The probability distribution of a single pixel can be then represented using a Bernoulli distribution.

The decoder gets as input the latent representation of a digit z and outputs 784 Bernoulli parameters, one for each of the 784 pixels in the image.

The decoder ‘decodes’ the real-valued numbers in z into 784 real-valued numbers between 0 and 1. Information from the original 784-dimensional vector cannot be perfectly transmitted, because the decoder only has access to a summary of the information (in the form of a less-than 784-dimensional vector z).

PES UNIVERSITY

How much information is lost?

We measure this using the reconstruction log-likelihood $\log p_\phi(x|z)$ whose units are nats.

This measure tells us how effectively the decoder has learned to reconstruct an input image x given its latent representation z.

VAEs Architecture: Loss Function

The *loss function* of the variational autoencoder is the negative log-likelihood with a regularizer.

Because there are no global representations that are shared by all data points, we can decompose the loss function into only terms that depend on a single datapoint l_i.

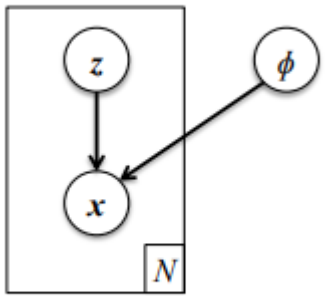
The total loss is then $\sum_{i=1}^{N} l_i$ for N total datapoints.
The loss function l_i for datapoint x_i is:

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] + \mathbb{KL}(q_\theta(z \mid x_i) \,||\, p(z))$$

VAE Training Methods

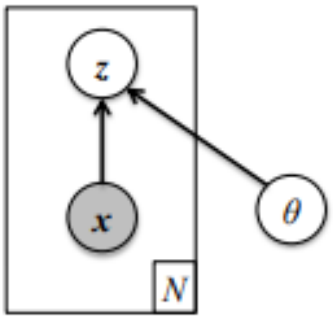
- In this case we have two distributions:
  - **model**: $p_\phi(\mathbf{z} \mid \mathbf{x})$
  - **variational approximation**: $q_{\lambda=f(x;\theta)}(\mathbf{z} \mid \mathbf{x})$
- We have the same model parameters ϕ

$p_\phi(\mathbf{x}, \mathbf{z})$



$z \sim \text{Gaussian}(0, I)$

$q_\theta(\mathbf{z} \mid \mathbf{x})$



Parameters θ and ϕ are neural network parameters (i.e. θ are not the variational parameters)

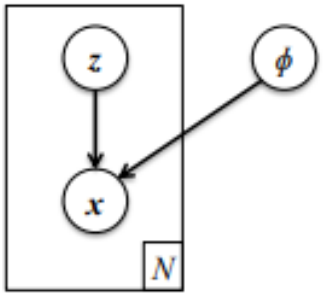
PES UNIVERSITY

We train the variational autoencoder using gradient descent to optimize the loss with respect to the parameters of the encoder and decoder θ and φ.

VAEs consist of three main components: Encoder, Decoder, and Latent Space.

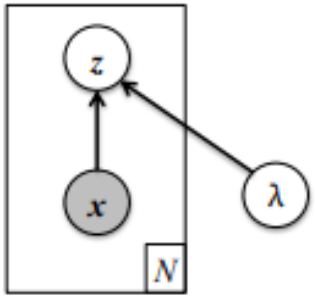
- We can also view the VAE from the perspective of variational inference
- In this case we have two distributions:
  - **model**: $p_\phi(\mathbf{z} \mid \mathbf{x})$
  - **variational approximation**: $q_{\lambda=f(x;\theta)}(\mathbf{z} \mid \mathbf{x})$
- We have the same model parameters ϕ
- The variational parameters λ are a function of NN parameters θ

$p_\phi(\mathbf{x}, \mathbf{z})$



$\mathbf{z} \sim \text{Gaussian}(0, I)$

$q_\lambda(\mathbf{z} \mid \mathbf{x})$



$\lambda = f(x; \theta)$

VAEs Architecture: Probability Model Perspective

VAEs consist of three main components: Encoder, Decoder, and Latent Space.

- **The Encoder network that take input data and transform it into a latent representation.**
- **It outputs two vectors representing the mean and variance of the probability distribution in the latent space.**
- **The encoder's objective is to capture the underlying structure of the input data in the latent space.**

VAEs Architecture: Probability Model Perspective

VAEs consist of three main components: Encoder, Decoder, and Latent Space.

- The Decoder network takes samples from the latent space distribution and generates reconstructions of the input data.
- It maps samples from the latent space to the output space.
- The decoder's objective is to reconstruct the input data from the latent representation.

VAEs Architecture: Probability Model Perspective

VAEs consist of three main components: Encoder, Decoder, and Latent Space.

- **The Latent Space is a lower-dimensional representation of the input data learned by the VAE.**
- **It represents the underlying structure of the input data in a compressed form.**
- **The latent space is probabilistic, with each point representing a probability distribution over possible data points.**

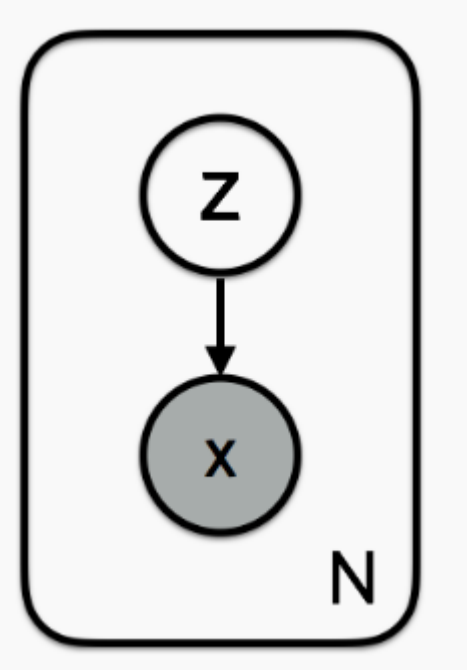
In the probability model framework, a variational autoencoder contains a specific probability model of data x and latent variables z.

We can write the joint probability of the model as $p(x,z)=p(x|z)p(z)$.

The generative process can be written as follows.

For each datapoint i:

- Draw latent variables $z_i \sim p(z)$
- Draw datapoint $x_i \sim p(x \mid z)$



- The Graphical Model Representation Of The Model In The Variational Autoencoder.
- The Latent Variable Z Is A Standard Normal, And The Data Are Drawn From P(X|Z).
- The Shaded Node For X Denotes Observed Data.
- For Black And White Images Of Handwritten Digits, This Data Likelihood Is Bernoulli Distributed.

The latent variables are drawn from a prior $p(z)$.

The data x have a likelihood $p(x|z)$ that is conditioned on latent variables z. The model defines a joint probability distribution over data and latent variables: $p(x,z)$.

We can decompose this into the likelihood and prior: $p(x,z)=p(x|z)p(z)$.

The goal is to infer good values of the latent variables given observed data, or to calculate the posterior $p(z|x)$.

As Bayes says:

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)}.$$

Examine the denominator $p(x)$. This is called the evidence, and we can calculate it by marginalizing out the latent variables: $p(x) = \int p(x \mid z)p(z)dz$. Unfortunately, this integral requires exponential time to compute as it needs to be evaluated over all configurations of latent variables. We therefore need to approximate this posterior distribution.

Variational inference approximates the posterior with a family of distributions $q_\lambda(z|x)$. The variational parameter λ indexes the family of distributions.

How can we know how well our variational posterior $q(z|x)$ approximates the true posterior $p(z|x)$?

We can use the Kullback-Leibler divergence, which measures the information lost when using q to approximate p (in units of nats):

$$\mathbb{KL}(q_\lambda(z \mid x) \,||\, p(z \mid x)) =$$

$$\mathbf{E}_q[\log q_\lambda(z \mid x)] - \mathbf{E}_q[\log p(x, z)] + \log p(x)$$

Our goal is to find the variational parameters λ that minimize this divergence. The optimal approximate posterior is thus

$$q_\lambda^*(z \mid x) = \arg\min_\lambda \mathbb{KL}(q_\lambda(z \mid x) \,||\, p(z \mid x)).$$

Why is this impossible to compute directly?

The pesky evidence $p(x)$ appears in the divergence. This is intractable as discussed above. We need one more ingredient for tractable variational inference.

Consider the following function:

$$ELBO(\lambda) = \mathbf{E}_q[\log p(x, z)] - \mathbf{E}_q[\log q_\lambda(z \mid x)].$$

ELBO stands for **E**vidence **L**ower **BO**und

AIM: training process involving maximizing the evidence lower bound (ELBO) through gradient descent.

Notice that we can combine this with the Kullback-Leibler divergence and rewrite the evidence as

$$\log p(x) = ELBO(\lambda) + \mathbb{KL}(q_\lambda(z \mid x) \,||\, p(z \mid x))$$

By Jensen's inequality, the Kullback-Leibler divergence is always greater than or equal to zero. This means that minimizing the Kullback-Leibler divergence is equivalent to **maximizing the ELBO.**

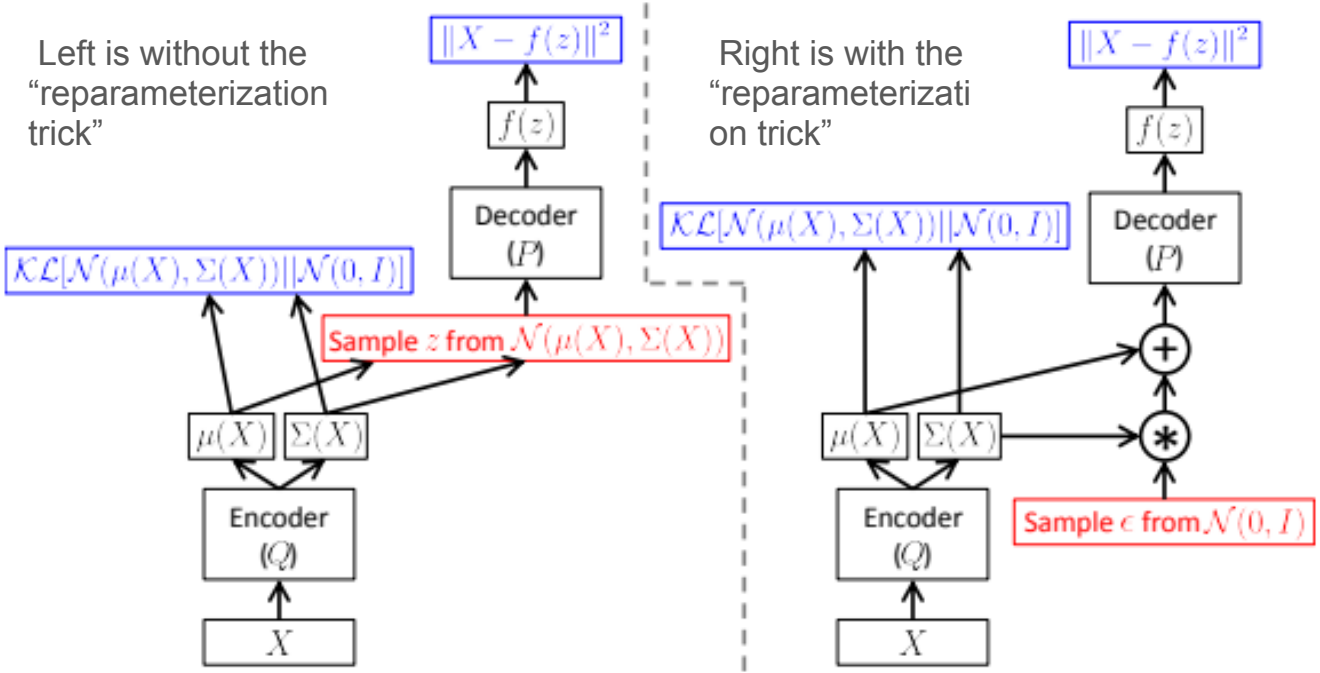
In the variational autoencoder model, there are only local latent variables (no datapoint shares its latent z with the latent variable of another datapoint).

So we can decompose the ELBO into a sum where each term depends on a single datapoint.

The ELBO for a single datapoint in the variational autoencoder is:

$$ELBO_i(\lambda) = \mathbb{E}q_\lambda(z \mid x_i)[\log p(x_i \mid z)] - \mathbb{KL}(q_\lambda(z \mid x_i) \mid\mid p(z)).$$

Reparameterization trick used to enable backpropagation through the latent space.

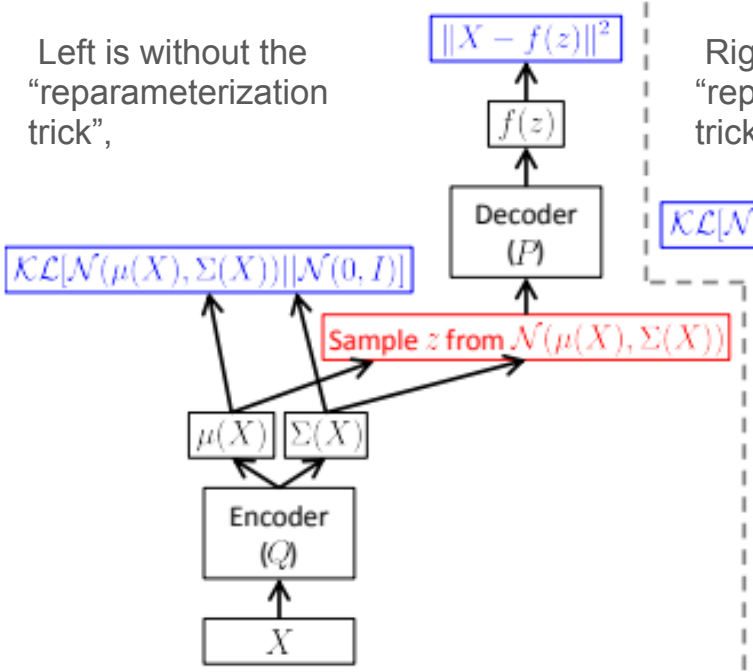


A training-time variational autoencoder implemented as a feedforward neural network, where P(X|z) is Gaussian.

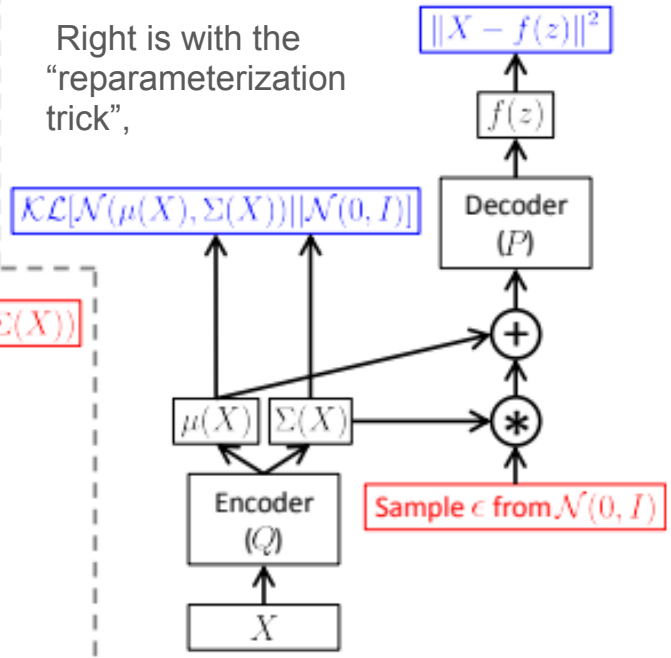
PES UNIVERSITY

Reparameterization trick used to enable backpropagation through the latent space.

Left is without the "reparameterization trick",



Right is with the "reparameterization trick",



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.

UE22CS645BC2-TDL-Lecture - VAE

References:

PES UNIVERSITY

- https://www.cs.toronto.edu/~duvenaud/courses/csc2541/slides/lec2-vae.pdf
- https://www.cs.toronto.edu/~rgrosse/courses/csc421_2019/slides/lec17.pdf
- https://medium.com/analytics-vidhya/generative-modelling-using-variational-autoencoders-vae-and-beta-vaes-81a56ef0bc9f
- https://deepgenerativemodels.github.io/notes/vae/
- https://www.youtube.com/watch?v=FMuvUZXMzKM