* Supervised Learning
* Unsupervised Learning
* Semi supervised Learning
* Self supervised Learning(Representation Learning, Contrastive Learning)
  + Exploit Unlabelled data to yield labels
  + Initially design supervised tasks(called pretext/auxiliary tasks) that can learn meaningful representations for downstream tasks
  + Analogous to fill in the blanks: predict certain part of the input(may be masked input) from any other part of the input
    - Context AutoEncoder(Image InPainting)
    - Unsupervised Learning of Visual Representation by solving Jigsaw puzzles
    - Unsupervised Representation Learning by predicting Image Rotations
    - Colorful Image Colorization
* Inductive Learning (Supervised Learning format)
  + Learns the representation using the observed examples and generalize it to unseen examples
* Transductive Learning
  + Learns using labelled and unlabelled data during training itself.(Labelling the unlabelled data using the labelled data representation and again train to fit the entire data) For, new unseen instance, the whole training has to be done to include this new unseen instance and to predict the label for it. Tough to use
* Representation Learning
  + Learn a distribution by pretraining on an auxiliary task(latent space representation)(supervised//self supervised fashion) that implicitly reveals the data representation in higher abstract level that helps a downstream task(like object detection using YOLO, ViT, Mae). Transfer Leaning is also the example of representation learning The model learns the features(data representations) in some low dimensional space and use those learned feature extractors for downstream tasks
* Contrastive Learning (Siamese Networks, MOCO, SIMCLR)

Contrastive self-supervised training objectives enabled recent successes in image representation pretraining by learning to contrast input-input pairs of augmented images as either similar or dissimilar.

* Zero shot, One shot, Few shot Learning
  + Use the representation from pretext task to predict our domain examples.
    - Zero shot - no training examples per class to train on. The unseen instance as directly predicted using the pretrained model representations
    - One shot - one example per class and use the model representations to fientune for these examples to predict them
    - Few shot - few examples per class
* Transfer Learning
* Regression
* Classification
* Decision Trees
* Deep Neural Networks (MultiLayer Perceptron)(Feed Forward Networks)
  + Computer Vision
    - Object Classification or Object Recognition
      * CNNs
        + LeNet5
        + ALexNet
        + VGG-16
        + ResNet (Residual Networks)
        + Inception(GoogLeNet)
        + DenseNet
        + MobileNet
        + Exception Net
        + EfficientNet
    - Object Localization, Landmark detection, Object Detection(Using the data representation for further downstream tasks)
      * R-CNN(Region Proposals and classification, object detector)(2 stage detectors)
      * OverFeat (Unified convolutional detection, classification and localization)(1 stage detector)
      * Fast RCNN, Faster RCNN
      * YOLO, YOLOv2 with anchorboxes, YOLOv3
    - Object Segmentation(Semantic Segmentation)
      * U-Net
* Sequence 2 Sequence Models’(NLP-Text)
  + Word Embeddings
    - CBOW
    - SkipGram
      * Word2Vec
      * GloVe
  + Recurrent Neural Networks
  + LSTM(Long Short Term Memory)
  + GRU(Gated Recurrent Units)
  + Bidirectional LSTM(ELMO)
  + Encoder Decoders
    - AutoEncoders (encoder, decoder)
    - Transformers(Encoder-decoder with position encoding, self attention, cross attention)
      * BERT(Bidirectional Encoder only Representations from Transformer)
      * AutoRegressive Models(Decoder only transformers)
        + GPT, GPT2, GPT3(Generative PreTraining)
      * Generative Models
        + GAN(Generative Adverserial Networks)

VQGAN

* + - * + Variational Auto Encoder VAE

Denoising VAE, Disentangled VAE , VQ-VAE

* + - * + Diffusion Models
      * Transformers in computer Vision
        + Vit

BeIT

Masked Auto Encoder (MAE)

VitDet

DiNO

MOCO

* + - * + Context Encoder(Image InPainting)
        + MaskGIT
        + VQGAN

VAE:

—----

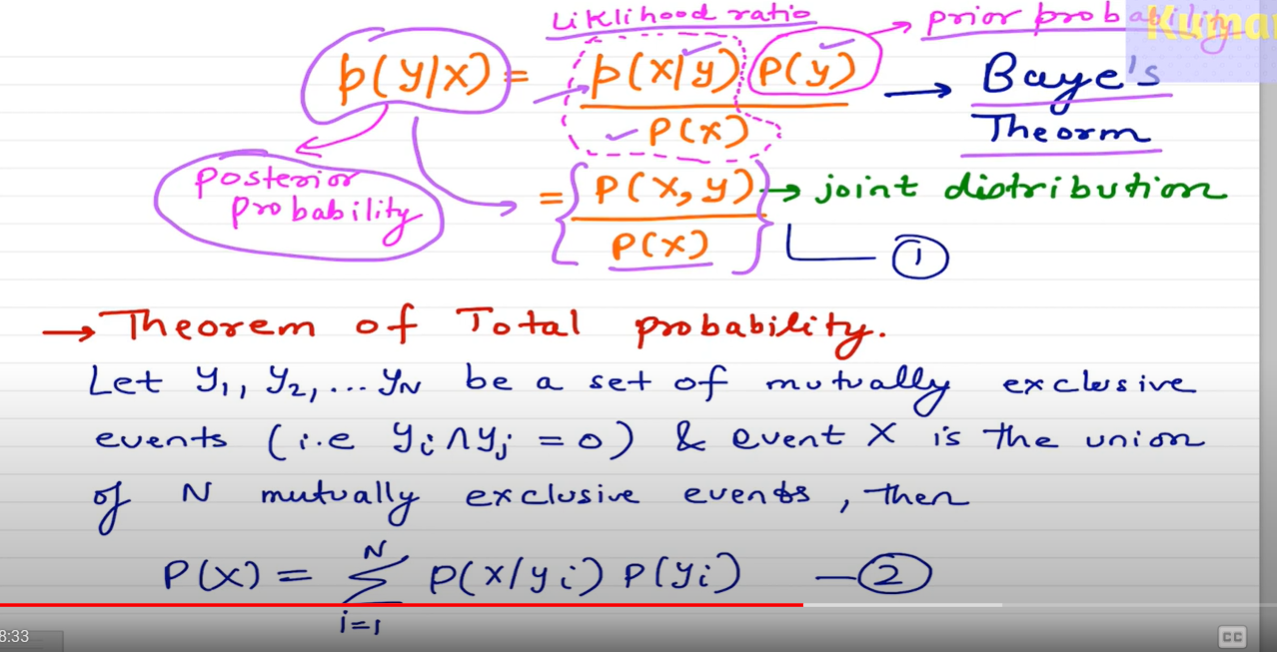
In the context of autoencoders and variational autoencoders (VAEs), deterministic and probabilistic refer to the nature of the encoding process.

1. **\*\*Deterministic Autoencoder:\*\***

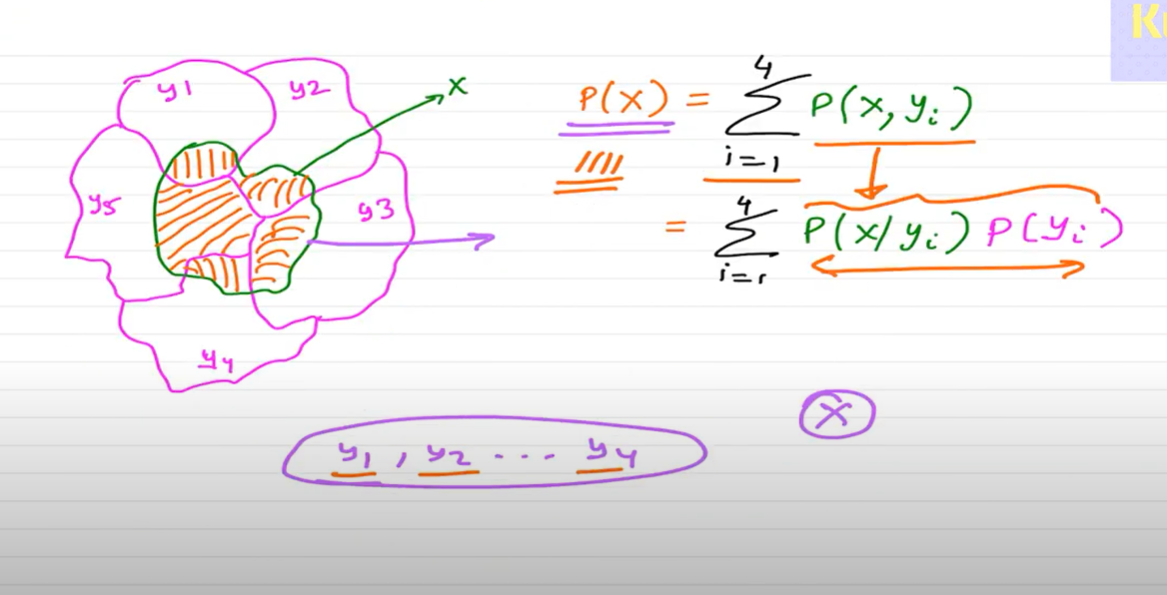
- \*\*Encoder:\*\* In a deterministic autoencoder, the encoder network directly maps an input data point to a fixed latent representation. There is no randomness involved in the encoding process.

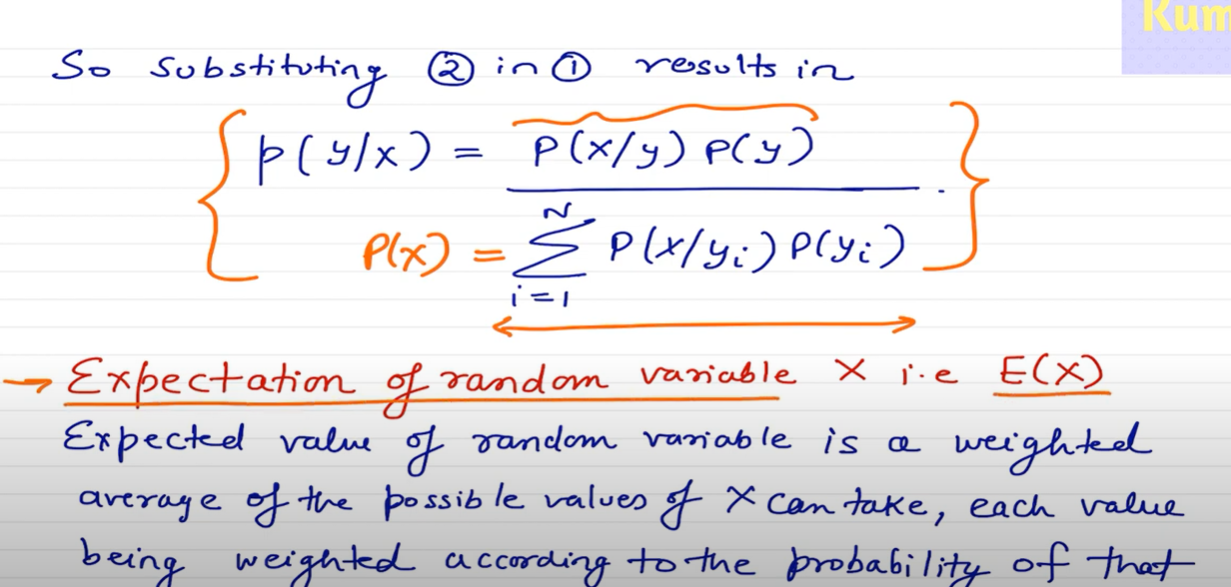
- \*\*Decoder:\*\* The decoder takes this fixed latent representation and reconstructs the input. The reconstruction is deterministic and does not involve any probabilistic sampling.

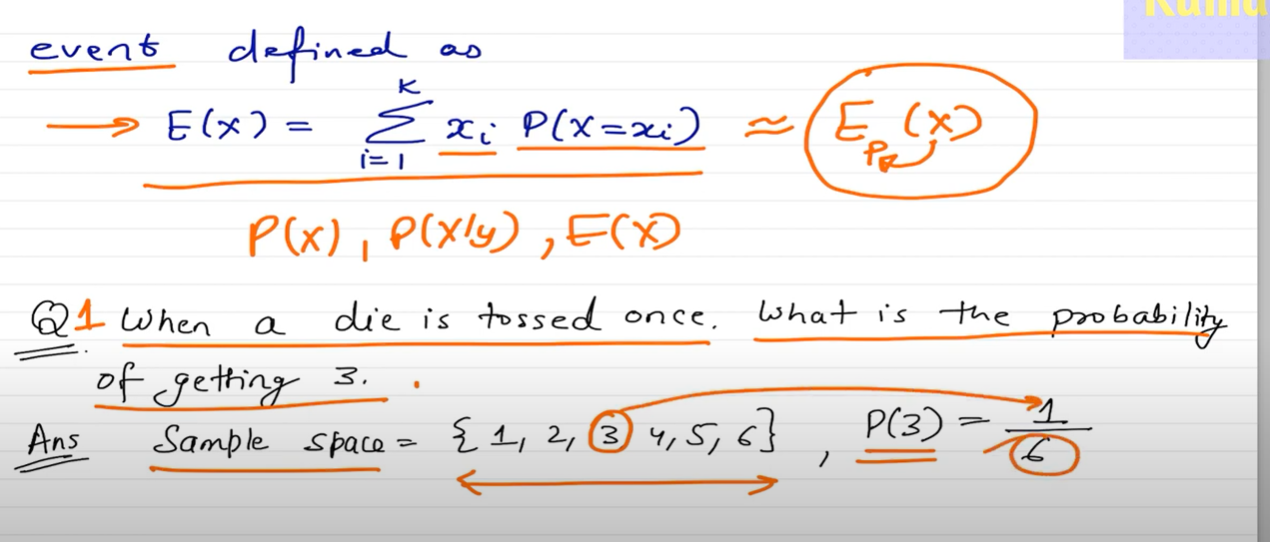
2. \*\*Probabilistic Autoencoder (Variational Autoencoder - VAE):\*\*

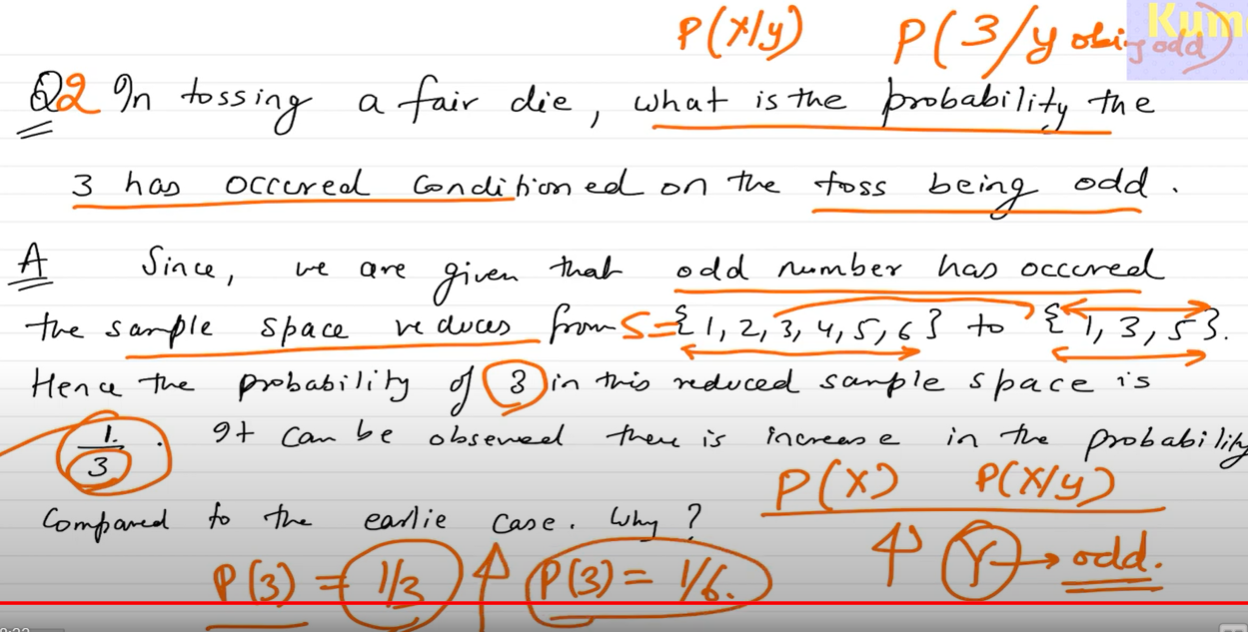


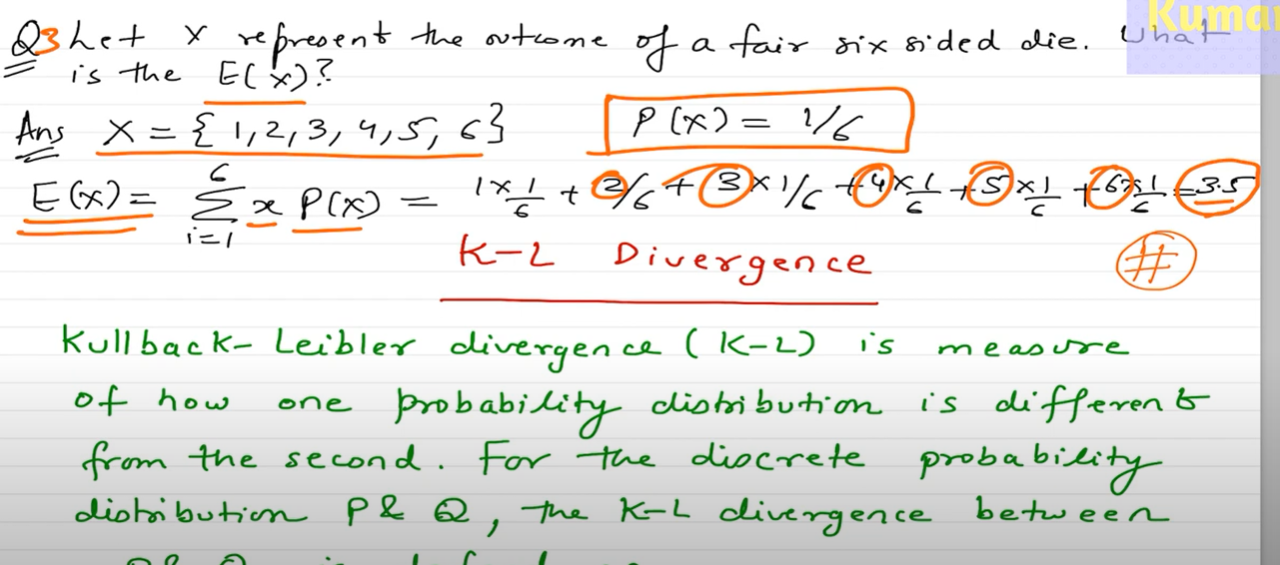
[Deep Learning 19: (1) Variational AutoEncoder : Introduction and Probability Refresher](https://youtu.be/w8F7_rQZxXk?si=OBoivIKO2z_N4qpn)

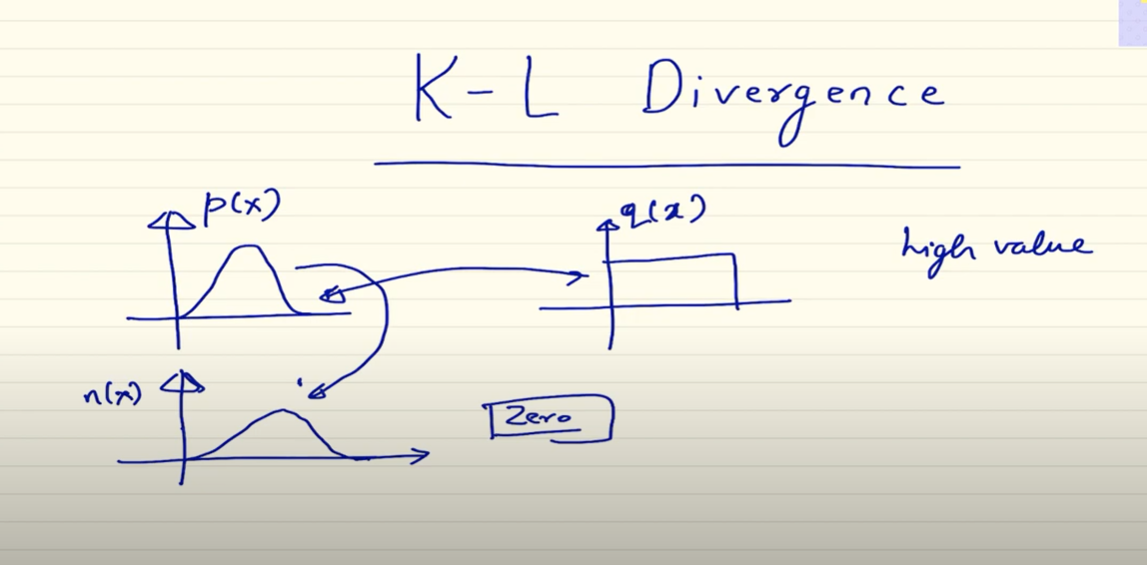


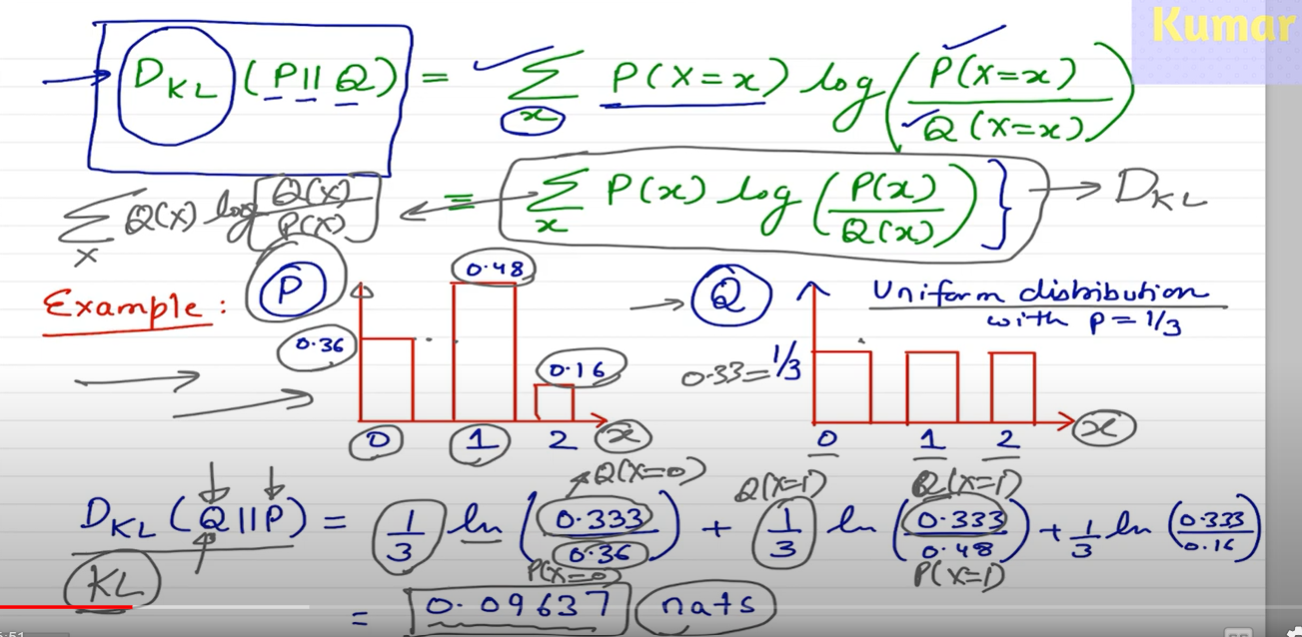


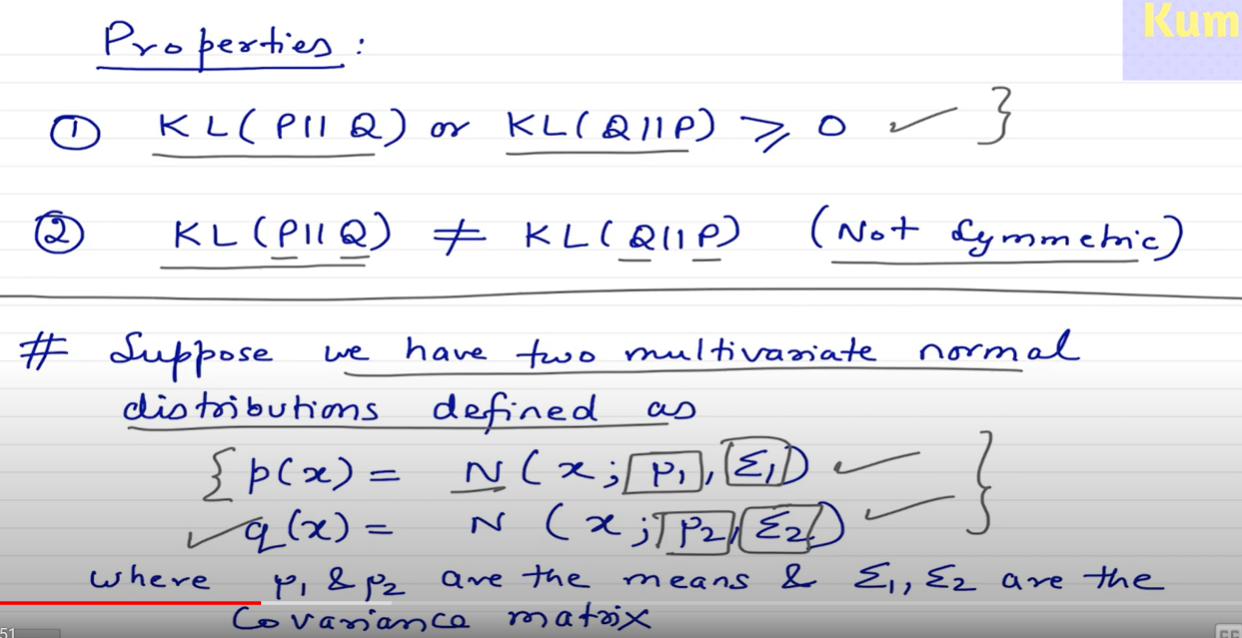


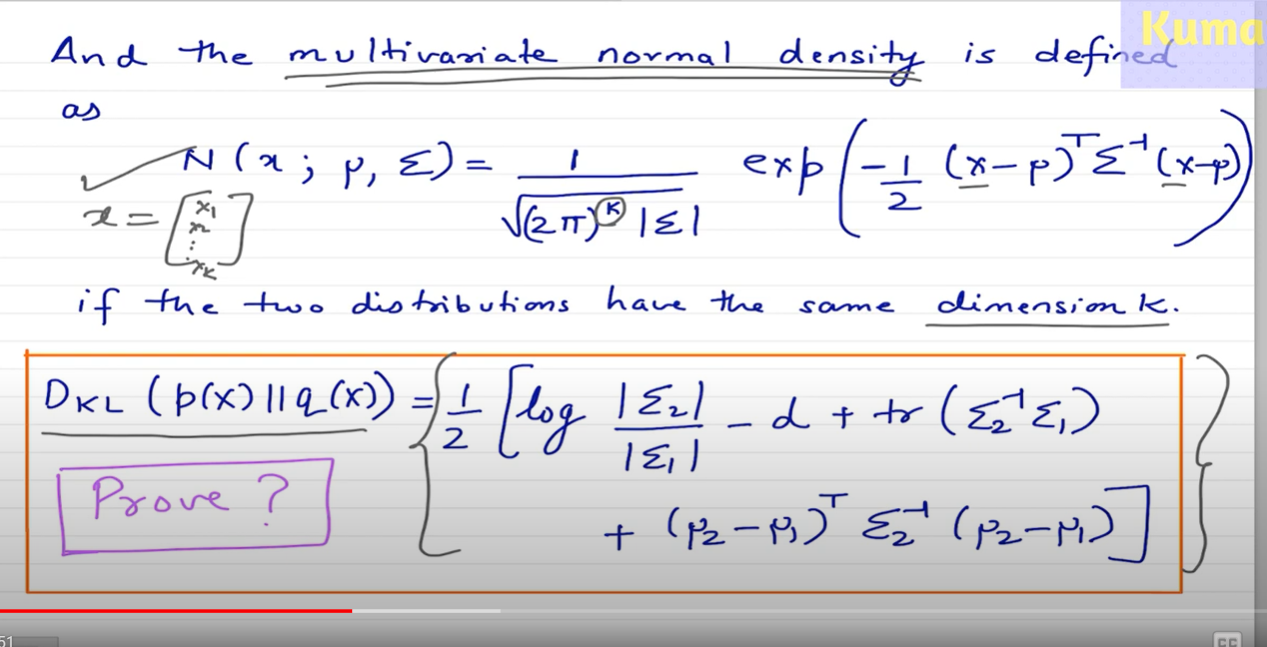


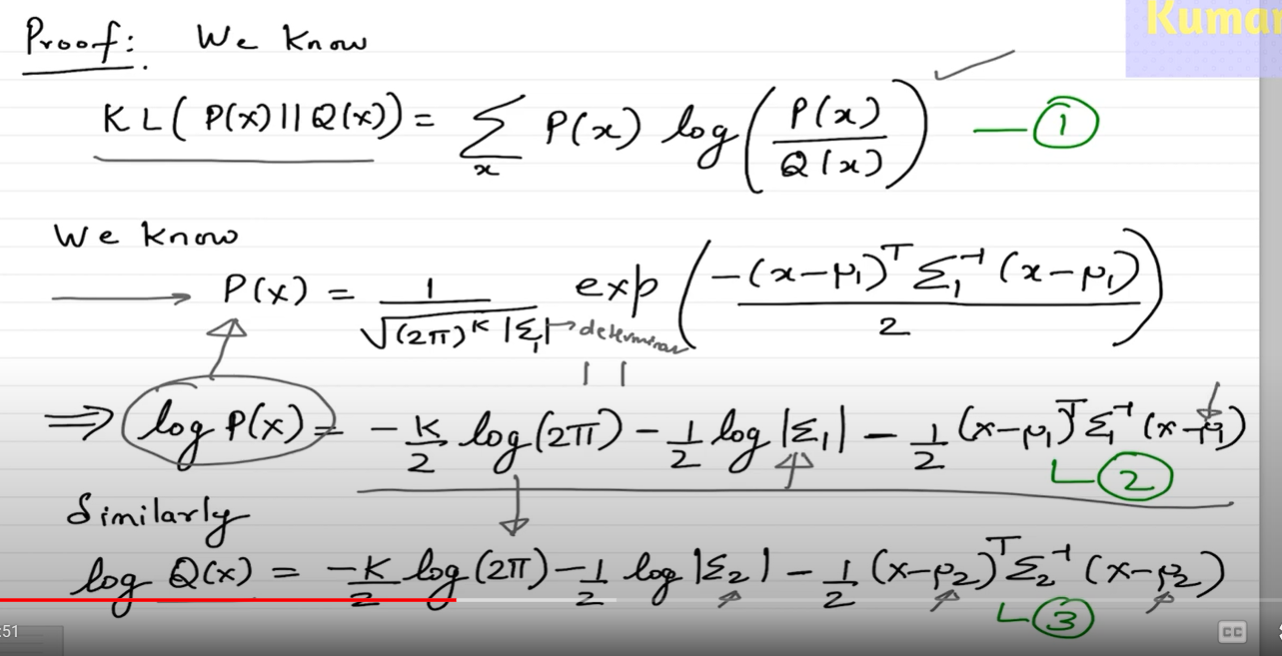


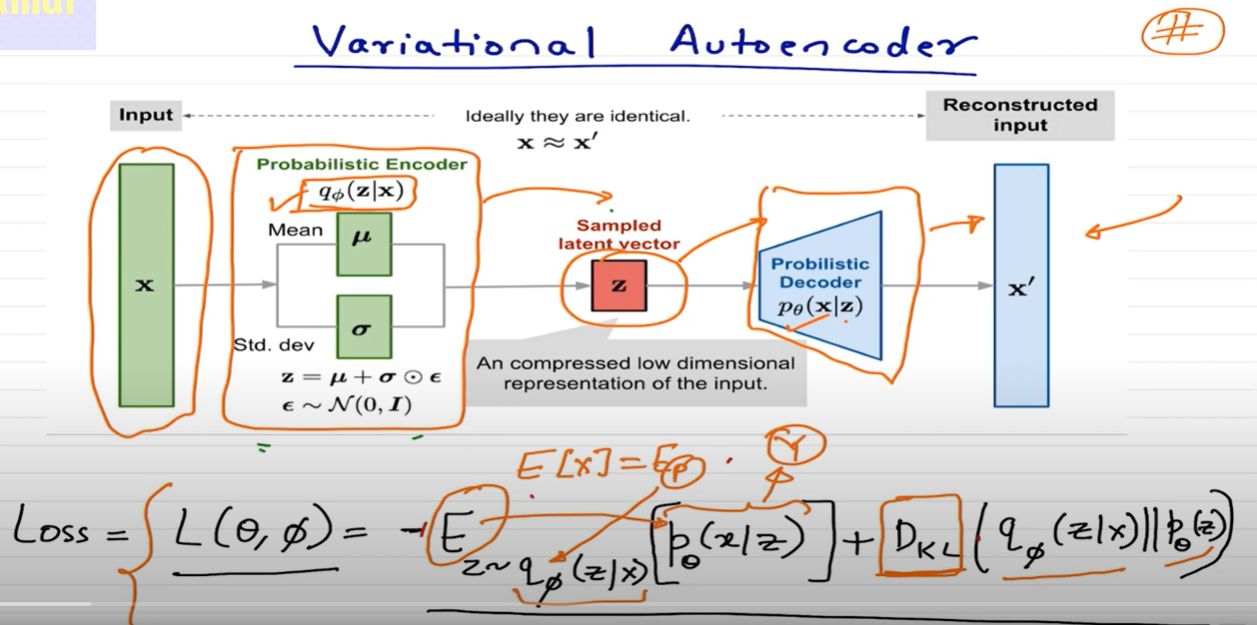


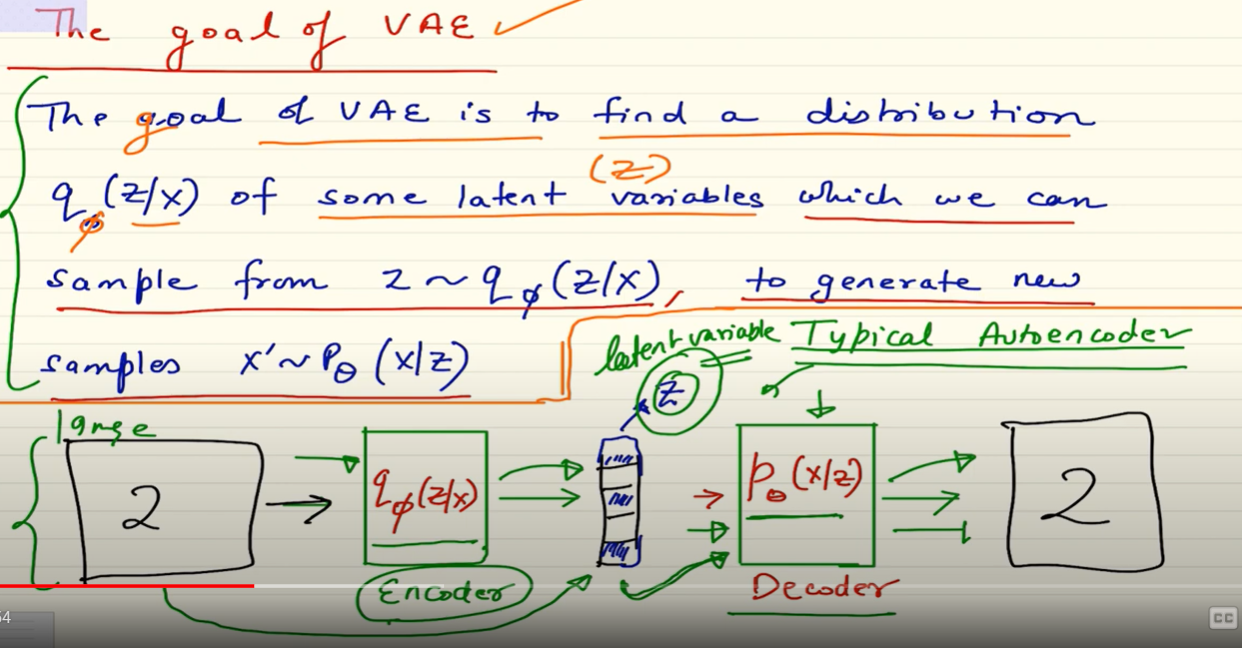












- **\*\*Encoder:\*\*** In a VAE, the encoder does not output a fixed latent representation. Instead, it outputs parameters of a probability distribution (commonly Gaussian) from which we sample the latent representation.(this makes the latent variables stochastic). It basically maps input to a distribution rather than fixed latent representation and we sample the latent variable from this distribution to send it to the decoder for reconstruction

- **\*\*Sampling:\*\*** The sampling step introduces a probabilistic element. Instead of having a single deterministic latent representation, we sample from the distribution defined by the encoder's parameters. This sampling introduces a level of uncertainty or randomness.

- **\*\*Decoder:\*\*** The decoder takes the sampled latent representation and reconstructs the input in a probabilistic manner.

Now, let's delve into some key terms related to VAEs:

- **\*\*Prior:\*\*** In a VAE, the prior is the assumed distribution of latent variables before observing any data. It represents our belief about the latent space before seeing the actual samples. Commonly, a simple distribution like a standard Gaussian is chosen as the prior.

- **\*\*Posterior:\*\*** The posterior represents the distribution of latent variables after observing the data. It is inferred during the training process and is a key component in Bayesian inference. The goal is to approximate the true posterior, which is often intractable.

- **\*\*Marginal Likelihood:\*\*** Also known as the evidence, it represents the likelihood of observing the given data under the model *p*(*X*) or *p*(*X*∣*θ*). In VAEs, calculating the marginal likelihood involves integrating over all possible values of the latent variable. This integral is often intractable, and that's where the variational inference comes in. *p*(*X*)=∫*p*(*X*, *Z* ∣ *θ*)*dZ*

- **\*\*Reparameterization Trick:\*\*** Since backpropagation relies on gradients, sampling from a distribution in a non-differentiable manner (like direct sampling from a Gaussian) can pose challenges. The reparameterization trick is a technique used to make the sampling operation differentiable. Instead of sampling directly, it involves introducing a differentiable transformation that includes the sampled noise.

In summary, VAEs introduce a probabilistic framework to the encoding process, allowing for more expressive latent spaces and capturing uncertainty in the data generation process. The reparameterization trick is a crucial element in training VAEs by making the sampling operation differentiable.

Let's dive deeper into the concept of marginal likelihood and how it relates to the challenge of intractability, leading to the introduction of variational inference in Variational Autoencoders (VAEs).

**\*\*Marginal Likelihood (Evidence):\*\***

The marginal likelihood, often denoted as *p*(*X*) or *p*(*X*∣*θ*), represents the likelihood of observing the given data *X* under the probabilistic model with parameters *θ*. It is essentially the integral of the joint distribution of the data and latent variables over all possible values of the latent variables. In mathematical terms:

*p*(*X*)=∫*p*(*X*, *Z* ∣ *θ*)*dZ*

where *Z* represents the latent variables. In practice, computing this integral can be analytically intractable for many models, including VAEs.

**\*\*Intractability Challenge:\*\***

For VAEs, the joint distribution involves the product of the likelihood of the data given the latent variables (*p*(*X*∣*Z*,*θ*)) and the prior distribution of the latent variables (*p*(*Z*)). However, the true posterior distribution *p*(*Z* ∣ *X*, *θ*) is often intractable to compute directly.

*p*(*X*, *Z* ∣ *θ*)=*p*(*X* ∣ *Z*,*θ*)⋅*p*(*Z*)

The intractability arises because the integral involves summing or integrating over all possible configurations of latent variables, which can be computationally prohibitive or impossible to do analytically.

**\*\*Variational Inference in VAEs:\*\***

To address the intractability of the marginal likelihood, VAEs introduce variational inference. The key idea is to approximate the true posterior distribution *p*(*Z* ∣ *X*,*θ*) with a more tractable distribution *q(Z* ∣ *X,ϕ)*, where *ϕ* represents the parameters of the approximate distribution.

The variational lower bound, also known as the Evidence Lower Bound (ELBO), is derived using Jensen's inequality. The ELBO is a lower bound on the log marginal likelihood:

log *p*(*X*)≥ELBO=E*q*(*Z*∣*X*,*ϕ*)[log *p*(*X*, *Z* ∣ *θ*)−log *q*(*Z* ∣ *X*,*ϕ*)]

Here Expectation E is using bcuz we are sampling the latent vraibles.

The objective during training is to maximize this ELBO, which encourages the approximate posterior to be close to the true posterior while remaining computationally feasible.

In summary, the marginal likelihood represents the overall probability of observing the given data under the model, but calculating it directly is often intractable. Variational inference, through the ELBO, provides a method to approximate the true posterior and make the computation more manageable.

This probabilistic approach helps to generate new sample/images instead of the same image generation like auto encoders.

—---------------------------------------

Positional Embeddings (learned and unlearned fixed position embedding) <https://machinelearningmastery.com/a-gentle-introduction-to-positional-encoding-in-transformer-models-part-1/>

<https://www.inovex.de/de/blog/positional-encoding-everything-you-need-to-know/>

<https://medium.com/@hunter-j-phillips/positional-encoding-7a93db4109e6>

[Rotary Positional Embeddings: Combining Absolute and Relative](https://www.youtube.com/watch?v=o29P0Kpobz0)

[Positional embeddings in transformers EXPLAINED | Demystifying positional encodings.](https://youtu.be/1biZfFLPRSY?si=KKyemnF6Qclw6adn)

[Adding vs. concatenating positional embeddings & Learned positional encodings](https://youtu.be/M2ToEXF6Olw?si=SU6WUIspAGjq0eZU)

[Self-Attention with Relative Position Representations – Paper explained](https://youtu.be/DwaBQbqh5aE?si=dpud_BcP9P1bc11t)

[Stanford XCS224U: NLU I Contextual Word Representations, Part 3: Positional Encoding I Spring 2023](https://www.youtube.com/watch?v=JERXX2Byr90&t=469s)

* Absolute Position Embedding
  + 2D absolute Position Embedding
* Relative Position Embedding
* Rotary Position Embedding
* Interpolation to fill the positional embeddings of scaled image using the existing low res image positional emebeddings

[Interpolation in 5 minutes](https://www.youtube.com/watch?v=Xj129kA3Ci0)

[Linear interpolation and resampling | Image processing](https://www.youtube.com/watch?v=rLMznzIslVA)

[Resizing Images - Computerphile](https://www.youtube.com/watch?v=AqscP7rc8_M&t=15s)

[Bicubic Interpolation - Computerphile](https://youtu.be/poY_nGzEEWM?si=gkUk3lFCfRCmoHkx)

[Image Interpolation Examples (Introduction)](https://youtu.be/8bTDssnJyZc?si=9dWuioHXr7rphf1W)

* + Linear
  + Bilinear(linear in 2D)
  + Spline
  + NearestNeighbour
  + Cubic
  + Bicubic Interpolation