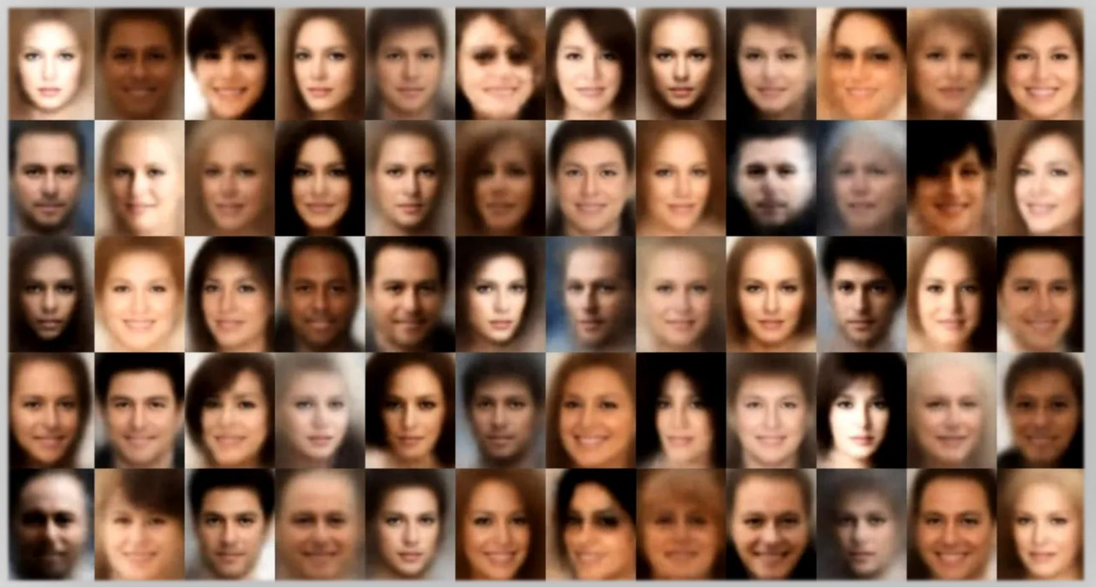
ES 204 – Digital Systems

Variational Autoencoders

Use Cases: Generative AI



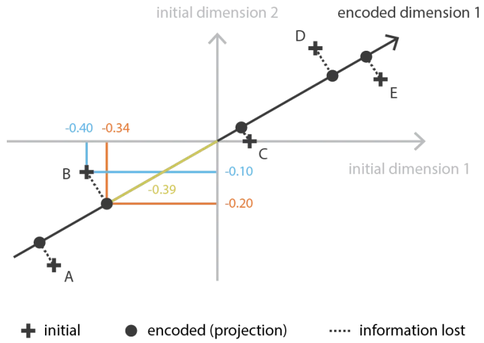
Why VAE?

- capture the underlying probability distribution of the data
- enable generation of new samples closely resembling training data
- offer control over the generation process, allowing manipulation of the latent space to produce data with specific characteristics
- useful for generative AI applications and diffusion models

PCA: A Lower Dimension

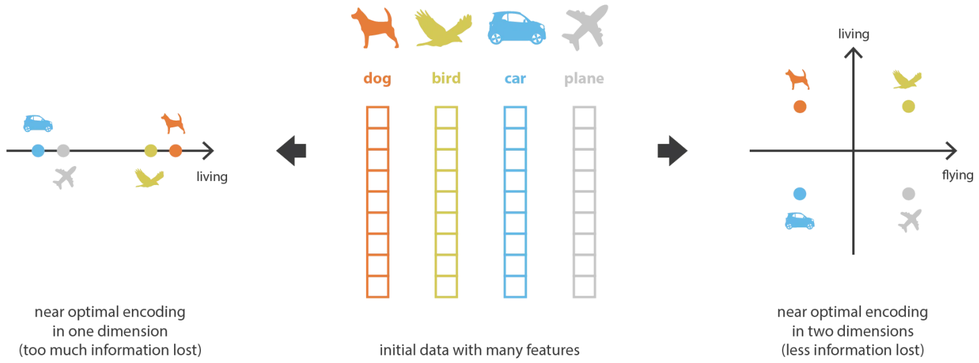
What is it?

- to come up with n_e new independent features, that are linear combinations of the n_d old features such that projection of the data on the subspace (defined by the new features) is as close as possible to the initial data
- for our framework, we are looking for an encoder in the family E of the $n_e \times n_d$ matrices whose rows are orthonormal, and for the associated decoder among the family D of $n_e \times n_d$ matrices



Point	Initial	Encoded	Decoded
A	(-0.50, -0.40)	-0.63	(-0.54, -0.33)
B	(-0.40, -0.10)	-0.39	(-0.34, -0.20)
C	(0.10, 0.00)	0.09	(0.07 0.04)
D	(0.30, 0.30)	0.41	(0.35, 0.21)
E	(0.50, 0.20)	0.53	(0.46, 0.27)

Principal Component Analysis (PCA) is looking for the best linear subspace using linear algebra.

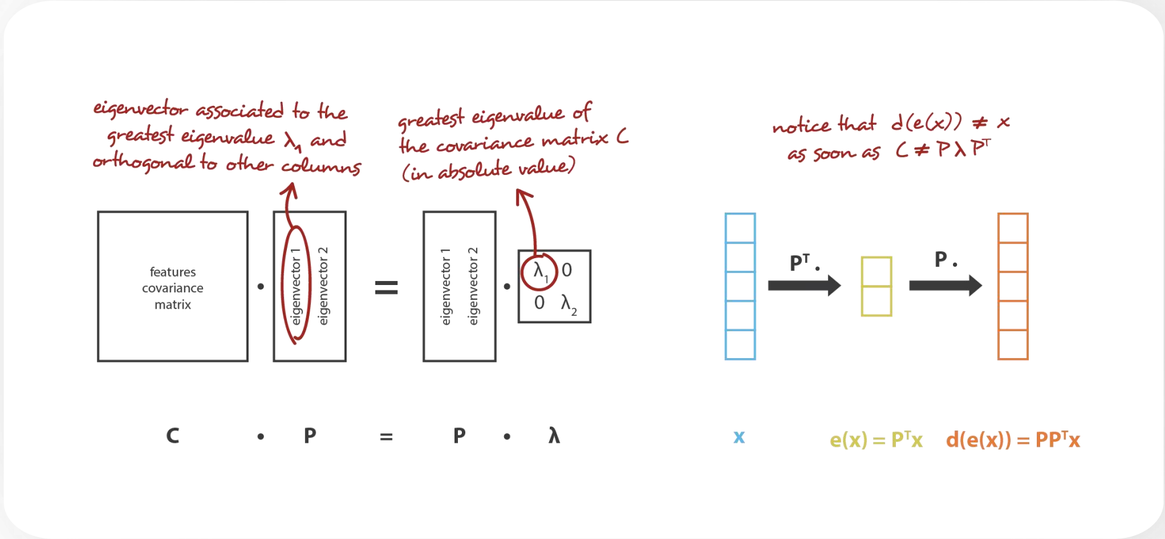


When reducing dimensionality, we want to keep the main structure there exists among the data.

PCA: A Lower Dimension

How it works?

- the unitary eigenvectors corresponding to the n_e greatest eigenvalues of the covariance matrix are orthogonal and define the best subspace of dimension n_e to project data on with minimal error of approximation
- the **decoder matrix**, in such cases, is the **transpose of the encoder**
- dimension reduction basically becomes an **eigen-value/vector problem**

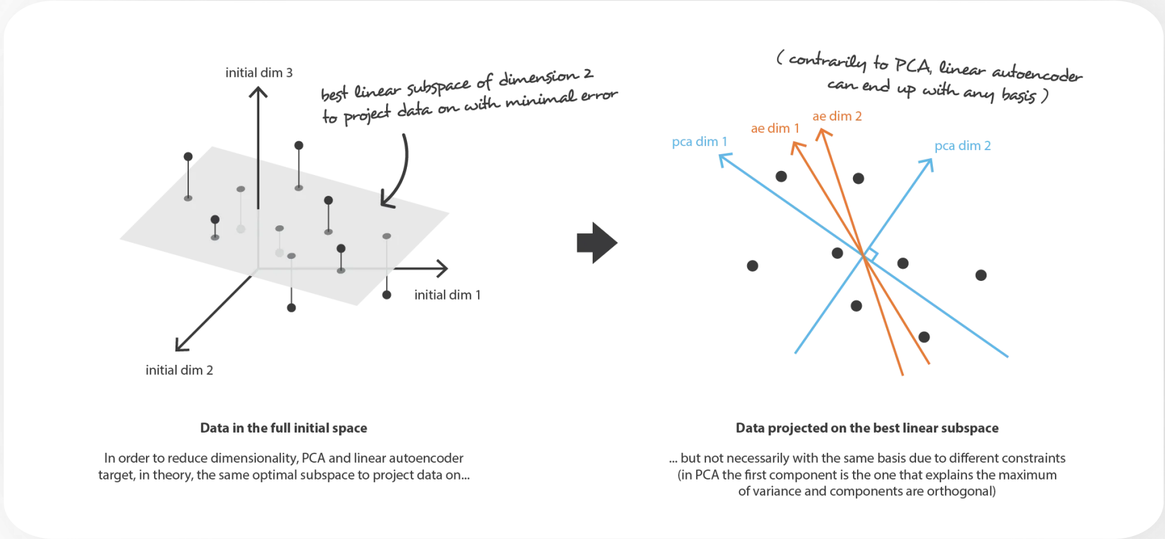


PCA matches the encoder-decoder framework we described.

Encoder-Decoder ⇒ Autoencoders

General Idea

- the general idea of autoencoders consists in **setting an encoder-decoder pair as neural networks** and to **learn the best encoding-decoding scheme using iterative optimization**
- encoding and decoding matrices obtained via PCA define one of the satisfiable solutions
- however, **several basis can be chosen to describe the same optimal subspace**, and so, several encoder/decoder pairs can give the same optimal reconstruction error

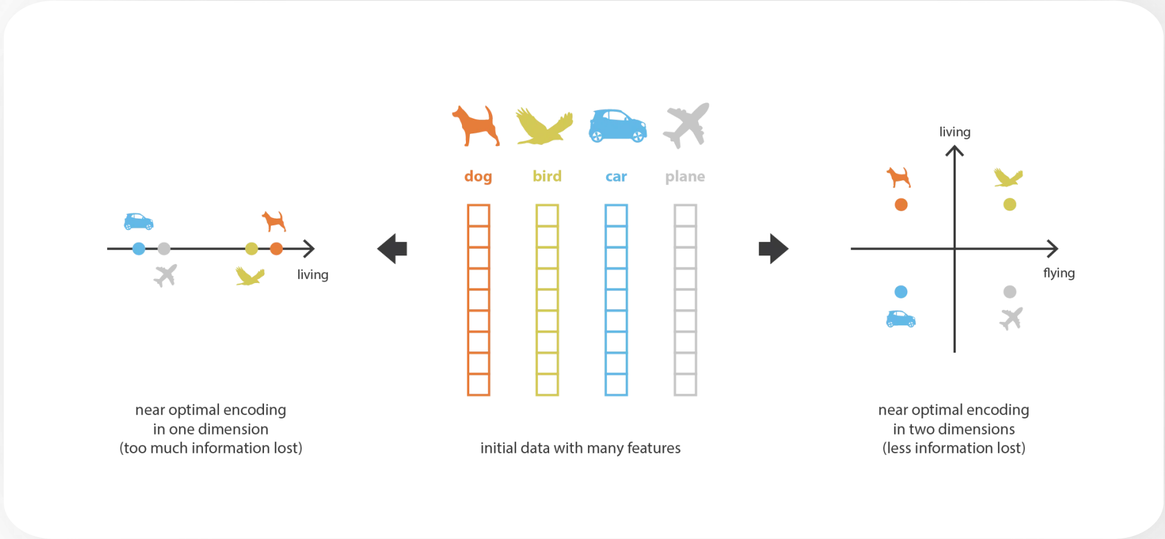


Link between linear autoencoder and PCA.

Encoder-Decoder ⇒ Autoencoders

General Idea

- both encoder and decoder are deep and non-linear
- **more complex** the architecture is, more the autoencoder can proceed to a **high dimensionality reduction** whilst keeping reconstruction loss low
- the final purpose of dimensionality reduction is to reduce this number of dimensions while **keeping the major part of the data structure information in the reduced representations**
- dimension of latent space and depth of autoencoders have to be controlled

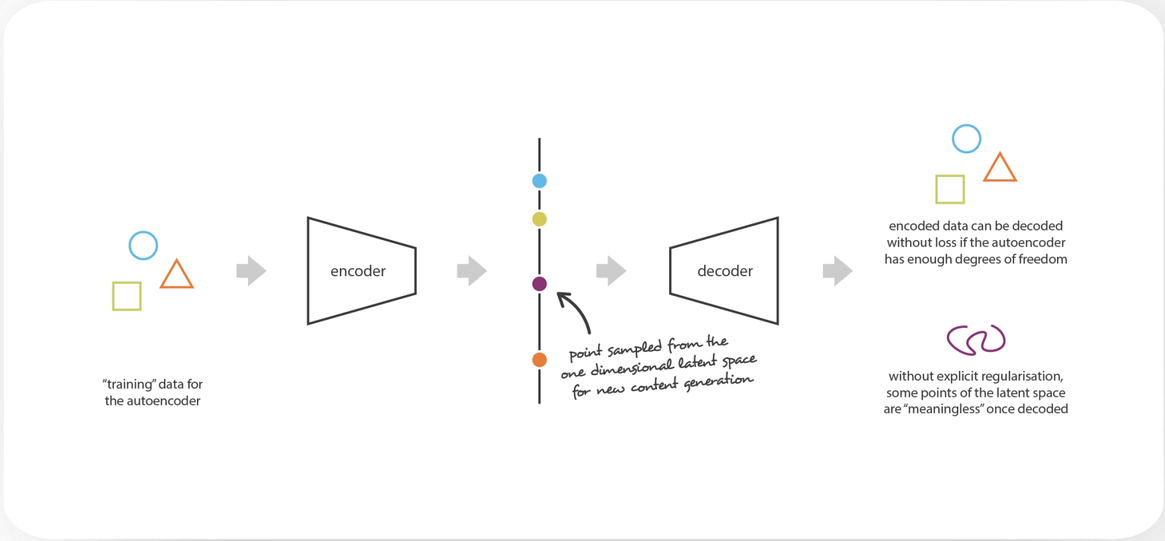


When reducing dimensionality, we want to keep the main structure there exists among the data.

Where a Regular Autoencoder fails

Limitations

- the **latent space** of regular autoencoders are **irregular**
- say we have a powerful enough encoder-decoder pair to reduce N dimensions onto the real axis
- in such a case, it is possible to encode and decode with no information loss but it **leads to severe overfitting** implying some points of the latent space will give meaningless content once decoded

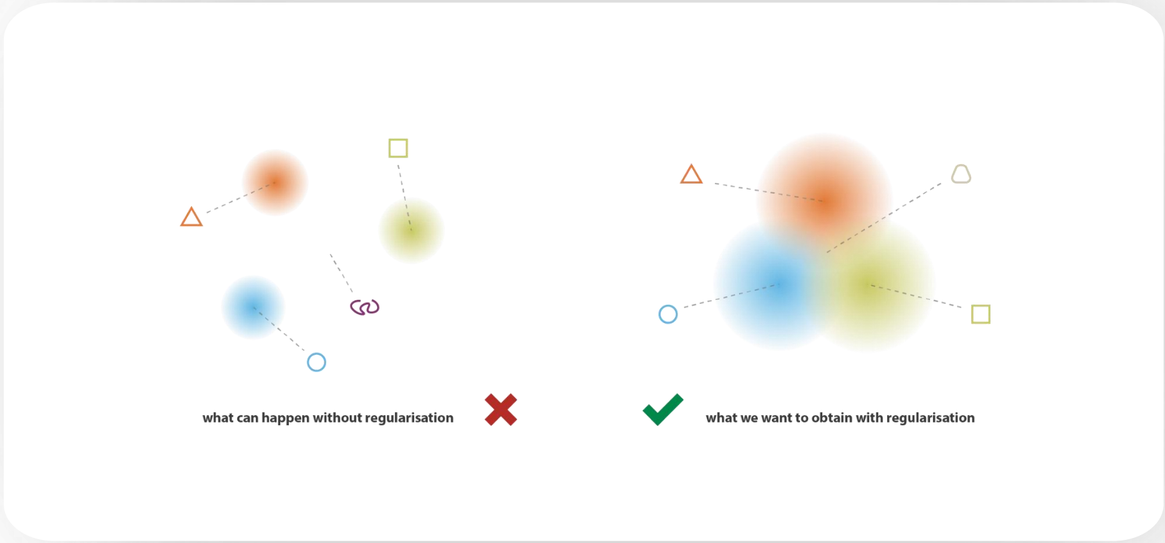


Irregular latent space prevent us from using autoencoder for new content generation.

The Variational in VAEs

Basic Definition

- ensure regularity of the latent space by introducing **explicit regularization during the training process**
- a **variational autoencoder** can be defined as being an autoencoder whose training is regularised to **avoid overfitting and ensure that the latent space has good properties** that **enable generative process**
- foundational reliance on probabilistic graphical models and variational inference



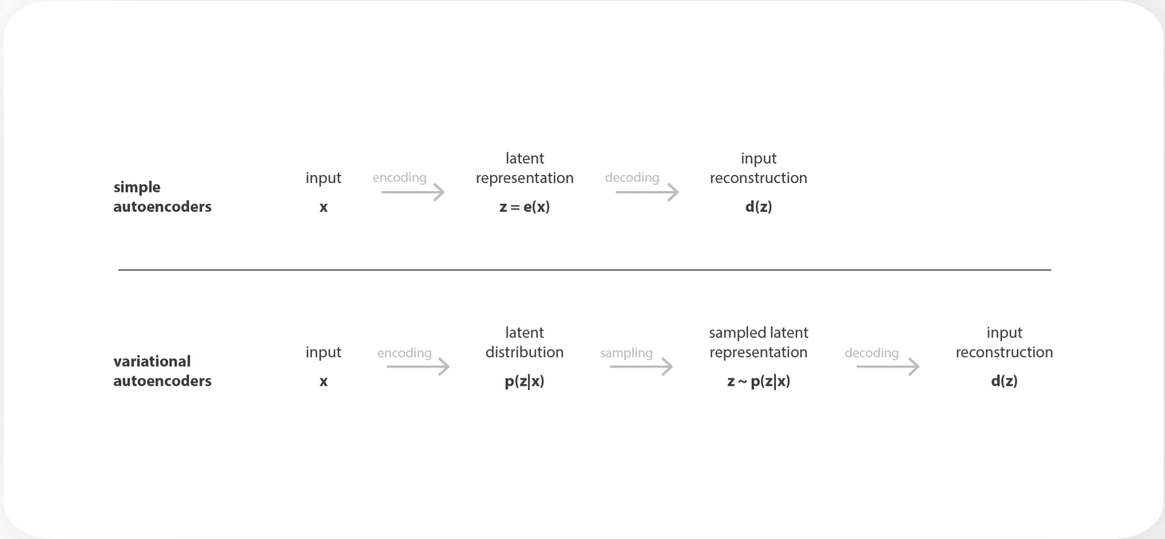
The returned distributions of VAEs have to be regularised to obtain a latent space with good properties.

The Variational in VAEs

Training the Model

1. input is encoded as distribution over latent space
2. a point from the latent space is sampled from the distribution
3. sampled point is decoded and reconstruction error calculated
4. reconstruction error is backpropagated through the network

- instead of encoding an input as a single point, we encode it as a **distribution over the latent space**



Difference between autoencoder (deterministic) and variational autoencoder (probabilistic).

The trick of Reparameterization

Regularizing Latent Space

- probability distribution of the latent space is **parametrized by a mean and standard deviation**
- a differentiable transformation that samples from a unit Gaussian $\mathcal{N}(0, 1)$ and transforms it as follows:

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$$

instead of directly from $\mathcal{N}(\mu, \sigma^2)$

- sampling operation becomes differentiable with respect to the parameters μ and σ **enabling backpropagation through the sampling step** during training

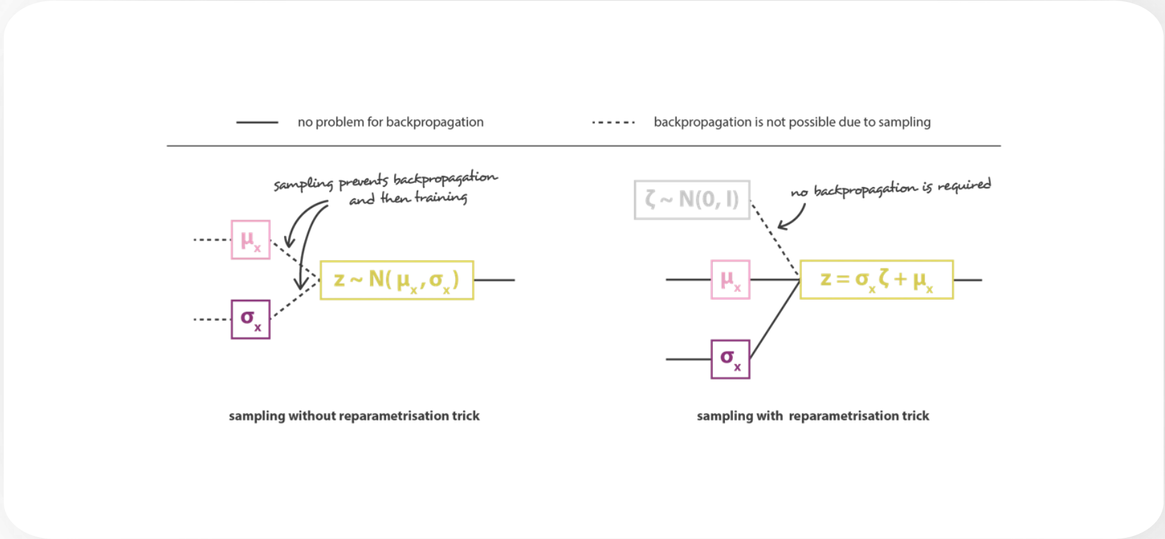
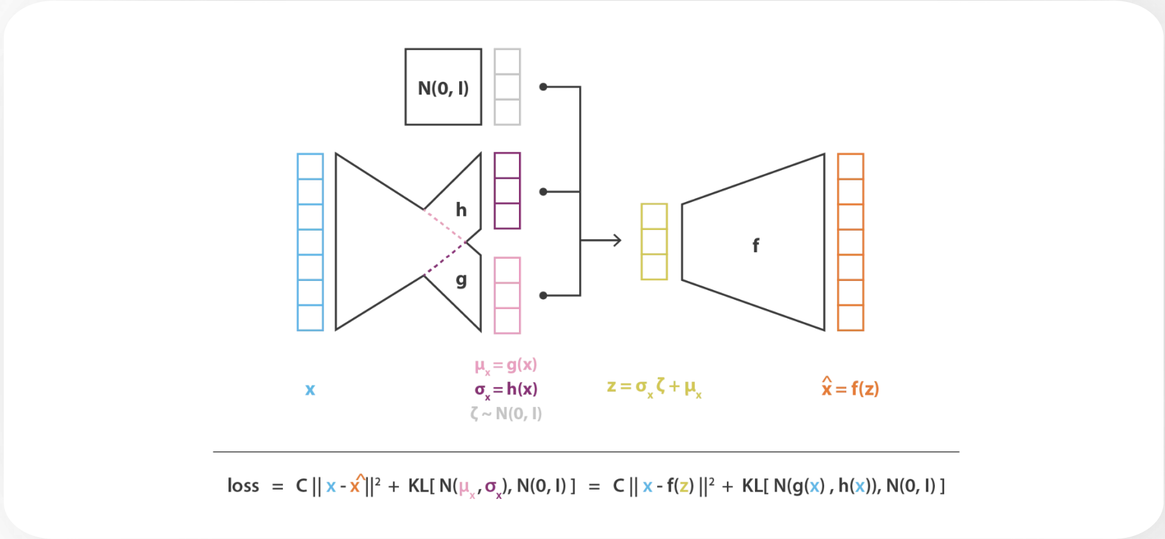


Illustration of the reparameterization trick.

Why does it work?

Everything comes together

- probability distribution of the latent space is **parametrized by a mean and standard deviation**
- separates sampling operation from the parameters of the distribution
- a differentiable transformation that samples from a unit Gaussian $\mathcal{N}(0, 1)$ and transforms it as follows: $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}$
- instead of directly from $\mathcal{N}(\mu, \sigma^2)$



Variational Autoencoders representation.

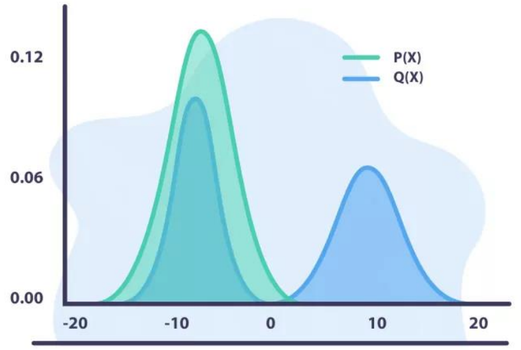
Mathematical Fundamentals

Kullback-Leibler Divergence

- VAE optimization involves minimizing the K-L divergence between the variational posterior $\boldsymbol{q_\theta(z|x)}$, where θ are learnable parameters and the true posterior $\boldsymbol{p(z|x)}$, integrated over the likelihood of the data.

- The resulting objective function is:

$$\mathcal{L}(\theta, \phi; x) = -\mathbb{E}_{q_\theta(z|x)}\left[\log p_\phi(x|z)\right] + \mathrm{KL}(q_\theta(z|x)||p(z))$$

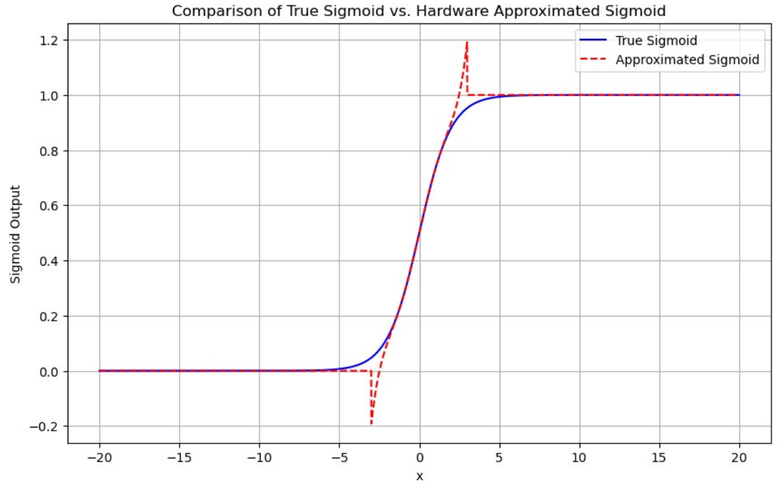


Representative Image for Kullback-Leibler Divergence

Mathematical Fundamentals

Approximation of Sigmoid

- the sigmoid function was approximated using a 5^{th} order Taylor series expansion around its midpoint

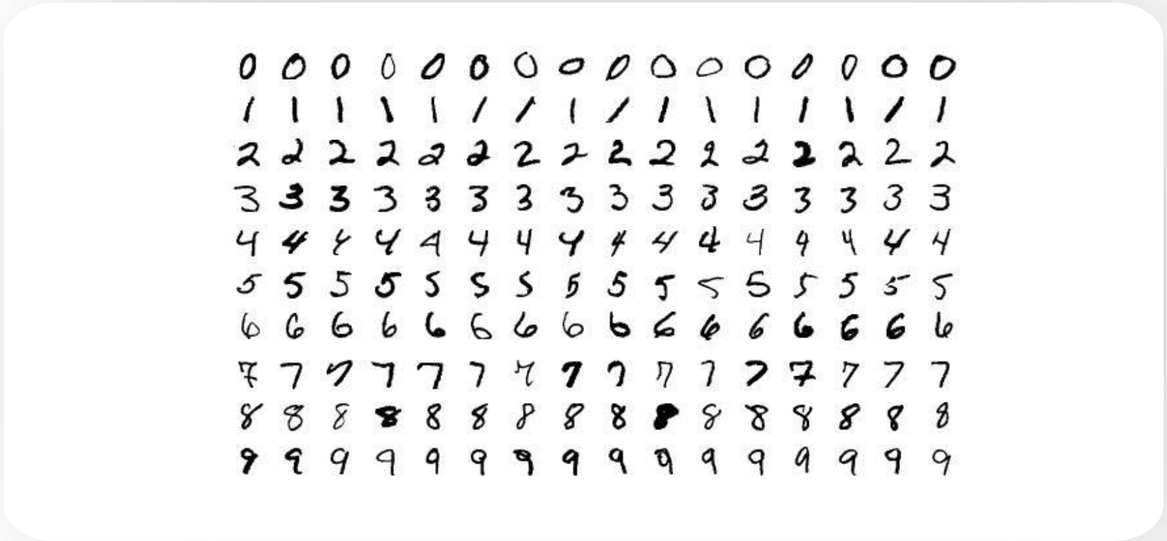


True vs. Approximated Sigmoid

Datasets Used

MNIST

- database of handwritten digits has a training set of 60,000 examples, and a test set of 10,000 examples

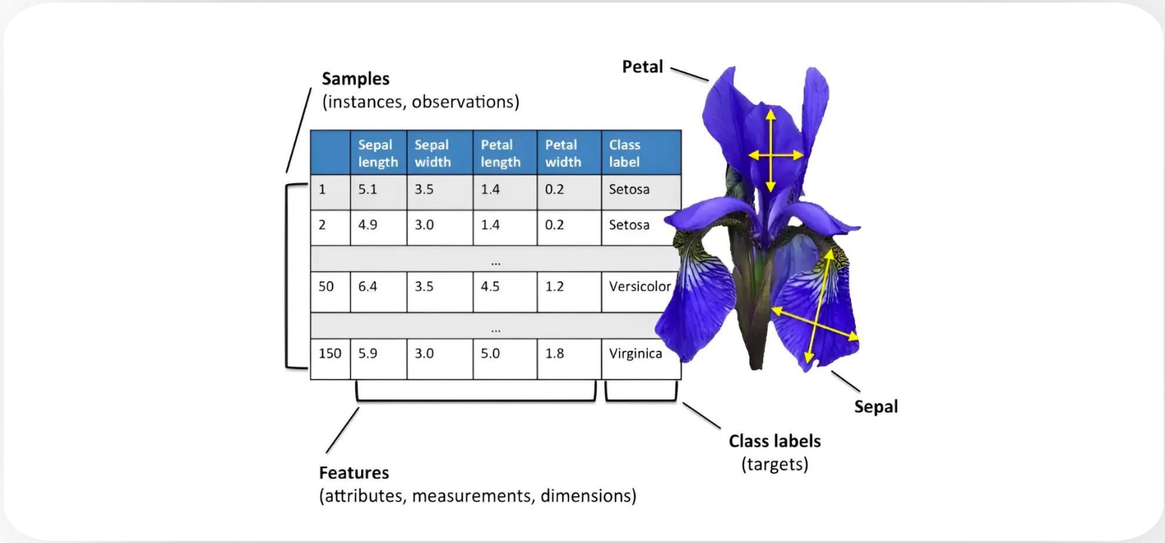


MNIST dataset

Datasets Used

IRIS

- database of 150 observations of iris flowers, including the sepal and petal length and width for each flower, as well as the species of the flower



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

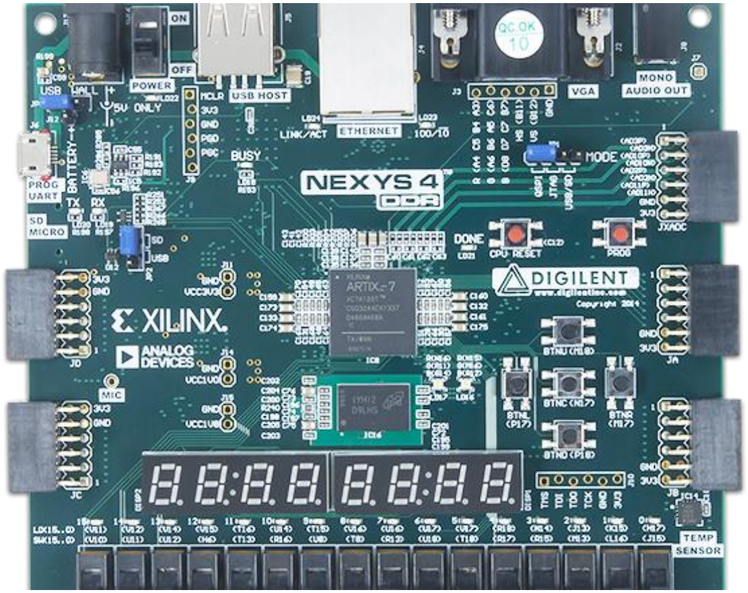
IRIS dataset

FPGA Implementation

on Nexys4 DDR

- the implementation of the VAE was done on the board itself, with all the code for the Neural Networks, and Matrix multiplications written in Verilog

1. **BRAM**
   initially, we planned to store the parameters of the model in the Block RAM of the board
2. **Arrays**
   however, since BRAM didn't work out, we tried and shrank our model enough to fit its parameters in a normal array vector
3. **UART**
   due to hardware issues with the UART module of the board, we decided to shift to the Iris dataset for which we could show the results without having to output an image



Nexys4 DDR FPGA Board

Thank You

Pranav Joshi, Laqshya Ghosh, Shreyans Jain, Akshat Barnwal

References: J. Rocca, “Understanding Variational Autoencoders (VAEs),” Medium, Mar. 15, 2020. https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73