Quantum Wasserstein Generative Adversarial Networks

An overview and implementation

Santiago Acevedo - Angel Martínez - November 2023

Content

Quantum Wasserstein Generative Adversarial Networks

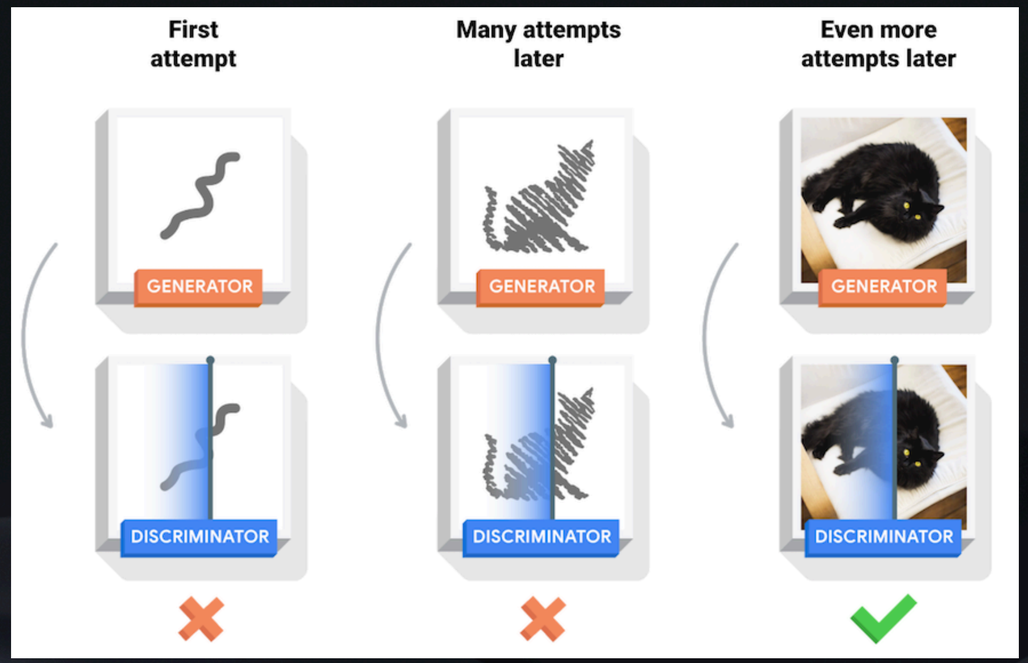
1. Generative Adversarial Networks
    1. Planets Images Generation Example
2. Wasserstein Generative Adversarial Networks
    1. WGANs vs. DCGANs
3. Quantum Computing
    1. Quantum Machine Learning
4. Quantum Generative Adversarial Networks
    1. Quantum Finance Example
5. References

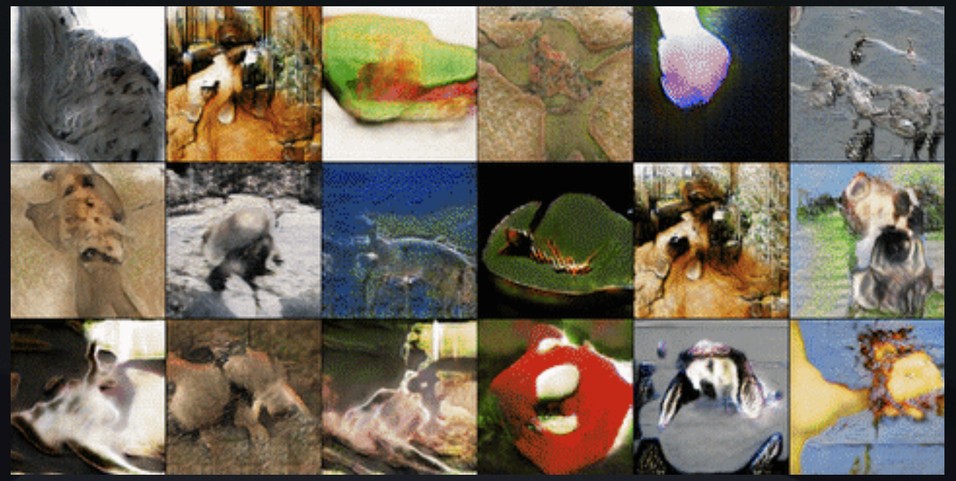
Generative Adversarial Networks

GANs

Introduction

Framework for estimating generative models via adversarial process:



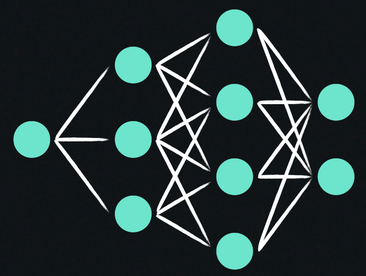


A generative model G generates samples that captures the data distribution

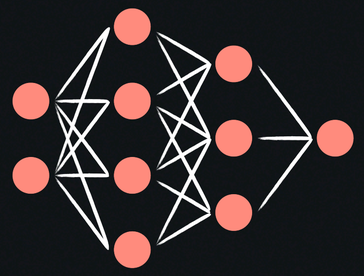
A discriminative model D estimates the probability that a sample came from the training data rather than G

GANs

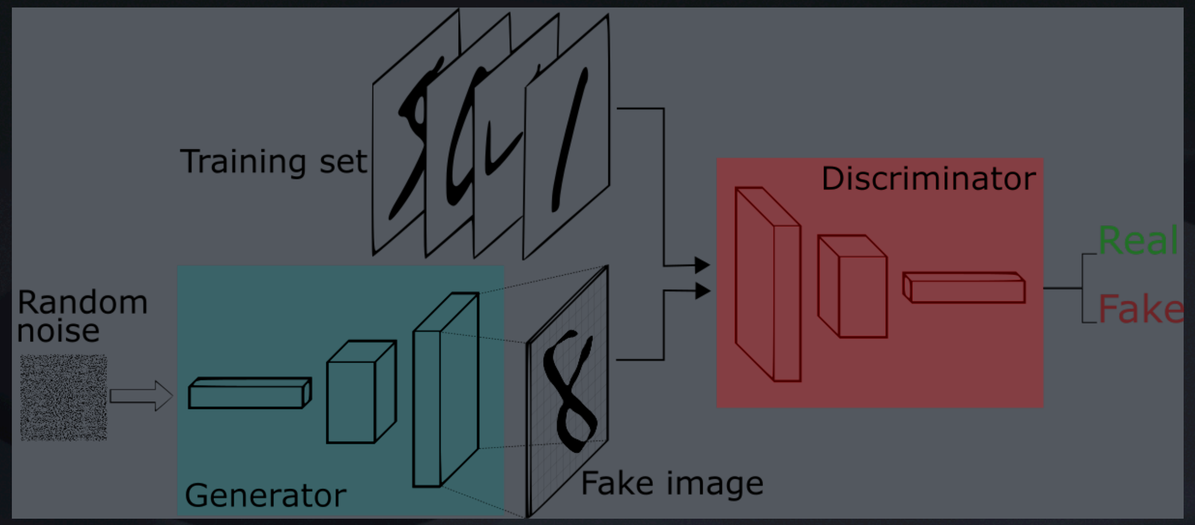
Generative Adversarial Networks (2014)



The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons



Generator $G(z, \theta_g)$ $D(x, \theta_d)$ Discriminator



The training procedure for G is to maximize the probability of D making a mistake

GANs

Algorithm

for numbers of training iterations **do**

for k steps **do**

> Sample $\{z^{(1)}, \ldots, z^{(m)}\} \sim p_g(z)$

> Sample $\{x^{(1)}, \ldots, x^{(m)}\} \sim p_{data}(x)$

> Update the discriminator by ascending stochastic gradient

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))\right]$$

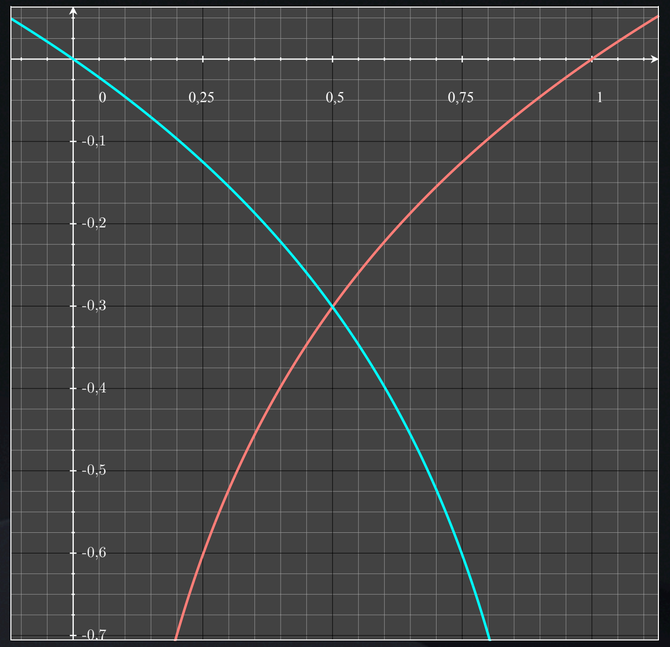
end for

> Sample $\{z^{(1)}, \ldots, z^{(m)}\} \sim p_g(z)$

> Update the generator by descending stochastic gradient

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z^{(i)})))$$

end for



We alternate k steps optimizing D and one step optimizing G

This results in D maintaining near its optimal solution, as G changes slowly enough

Lets see an implementation in astrophysics context

GANs

Planets Images Generation Specifications

Dataset size: 1834 Images of 9 planets
Images size: 80px X 80 px X 3 Channels
Training time: 25 minutes approx.
Optimizer: Adam
Epochs: 10000

Generator Architecture

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 100)]	0
dense (Dense)	(None, 102400)	10240000
batch_normalization (BatchNormalization)	(None, 102400)	409600
leaky_re_lu (LeakyReLU)	(None, 102400)	0
reshape (Reshape)	(None, 20, 20, 256)	0
conv2d_transpose (Conv2DTranspose)	(None, 20, 20, 128)	819200
batch_normalization_1 (BatchNormalization)	(None, 20, 20, 128)	512
leaky_re_lu_1 (LeakyReLU)	(None, 20, 20, 128)	0
conv2d_transpose_1 (Conv2DTranspose)	(None, 40, 40, 64)	204800
batch_normalization_2 (BatchNormalization)	(None, 40, 40, 64)	256
leaky_re_lu_2 (LeakyReLU)	(None, 40, 40, 64)	0
conv2d_transpose_2 (Conv2DTranspose)	(None, 80, 80, 3)	4800

Total params: 11,679,168
Trainable params: 11,473,984
Non-trainable params: 205,184

Discriminator Architecture

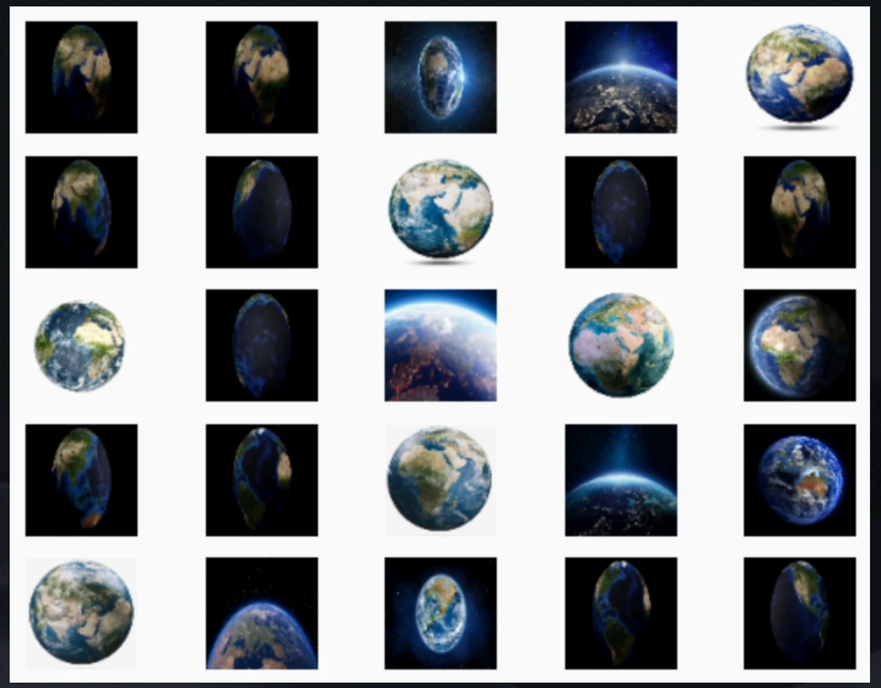
Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 80, 80, 3)]	0
conv2d (Conv2D)	(None, 40, 40, 64)	4864
dropout (Dropout)	(None, 40, 40, 64)	0
conv2d_1 (Conv2D)	(None, 20, 20, 128)	204928
dropout_1 (Dropout)	(None, 20, 20, 128)	0
flatten (Flatten)	(None, 51200)	0
dense_1 (Dense)	(None, 1)	51201

Total params: 260,993
Trainable params: 260,993
Non-trainable params: 0

GANs

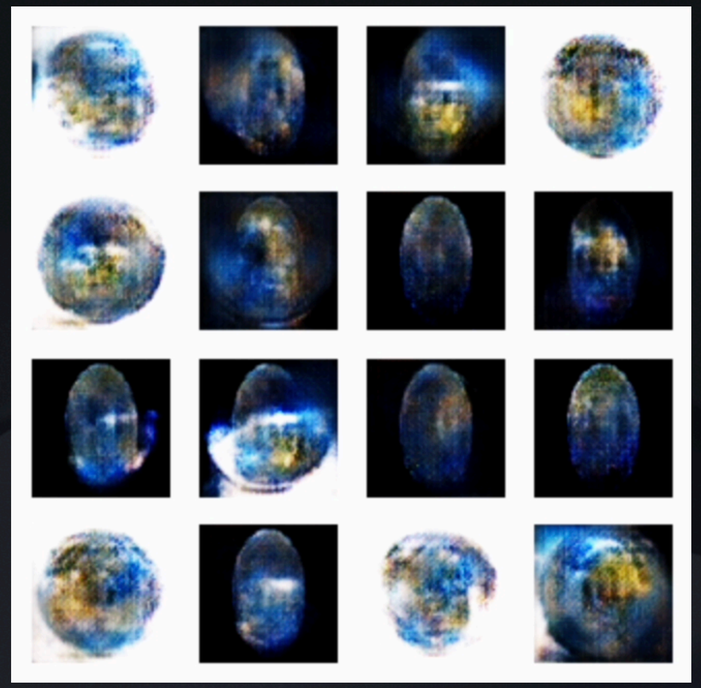
Planets Images Generation Results

219 images of the Earth



Original Images

Epoch 10000



Generated Images

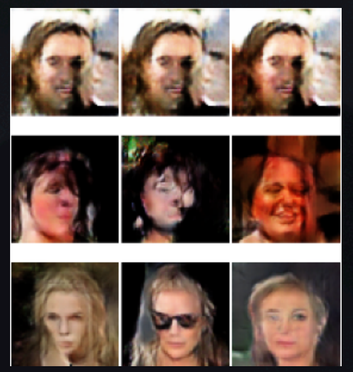
GANs

Difficulties

GANs architecture possess an **immense potential** but fail trying to perform some difficult computations

Two of the most common reasons are:

Convergence failure



The model failed to produce optimal or good quality results

Mode Collapse

The model failed to produce unique images repeating a similar pattern or quality

Wasserstein Generative Adversarial Networks

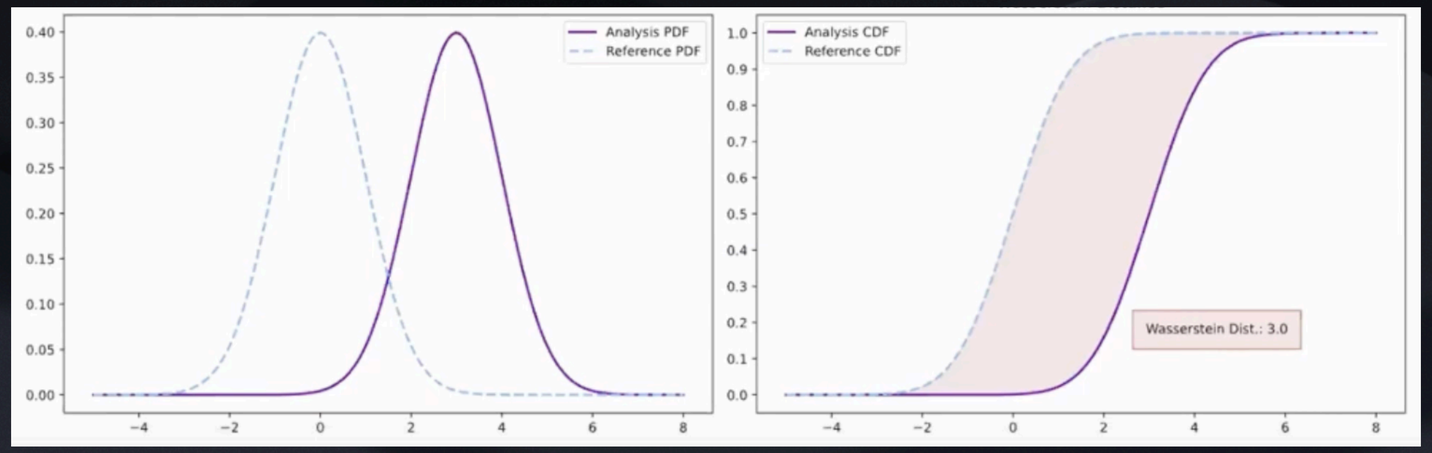
WGANs

Probability Distributions Distance

The objective of the Generative Adversarial Networks is to ensure that $\mathbb{P}_g(z)$ and $\mathbb{P}_r(x)$ probability distributions are **close** to each other so that the output generated is highly realistic and high-quality

How can we calculate the **distance** between two probability distributions?

The Wasserstein distance



WGANs

WGANs Efficience

The Generator network in a WGANs is the function $g_{\theta'}$
While the Critic (discriminator) network corresponds to f_w

We can apply gradient descent on the Wasserstein distance (due to its continuity) in order to learn a probability distributions

Meaningful learning curves useful for debugging and hyperparameters searches

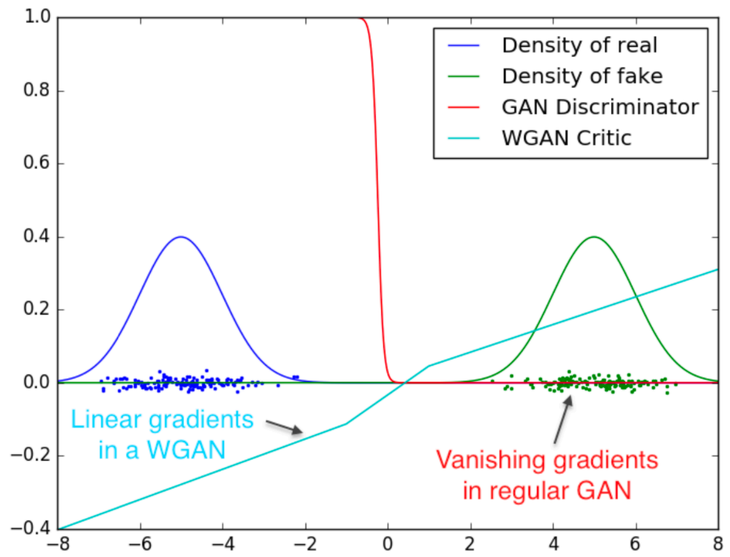


Figure 2: Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the discriminator of a minimax GAN saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.

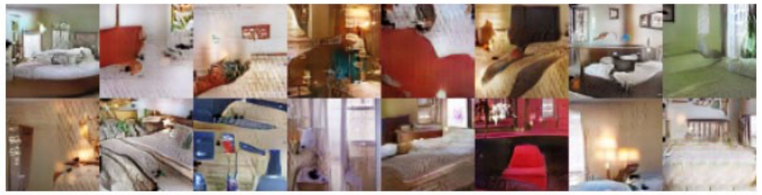
WGAN offers higher **stability** to the training model in comparison to simple GAN architectures

No evidence of **mode collapse** in the experiments

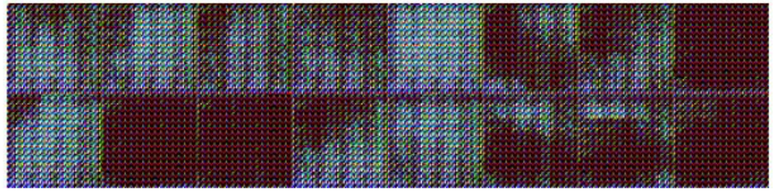
WGANs

Comparison between GAN and WGAN

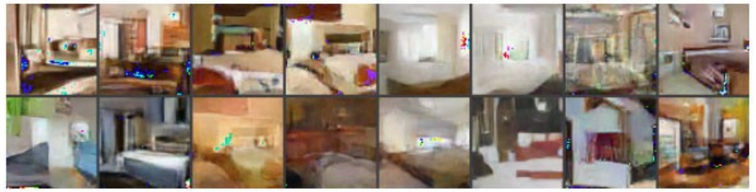
WGAN with DCGAN generator



GAN with DCGAN generator



Without batch normalization &
constant number of filters at each layer



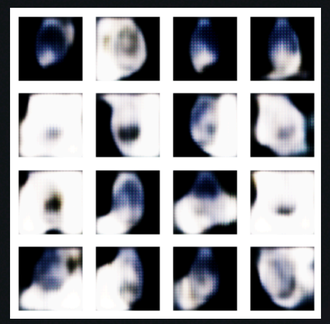
Using a MLP as the generator

All critics and discriminators follow the same discriminator design in DCGAN

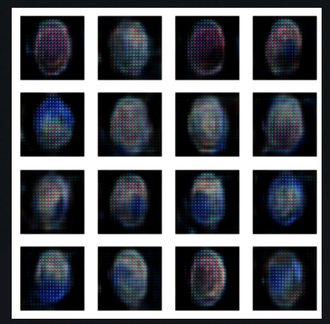
WGANs

Generated Planets Images

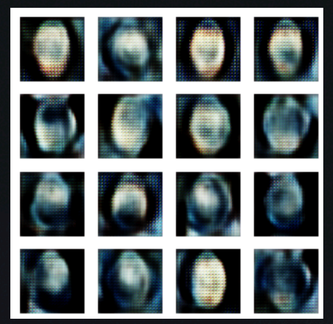
Epoch 1000



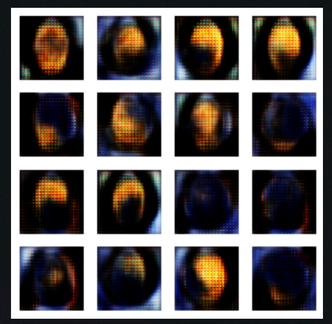
Epoch 2200



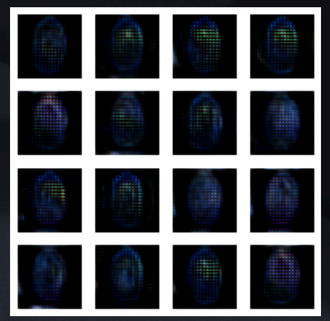
Epoch 3100



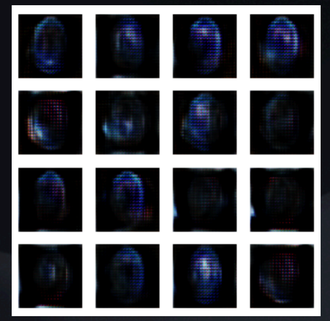
Epoch 4400



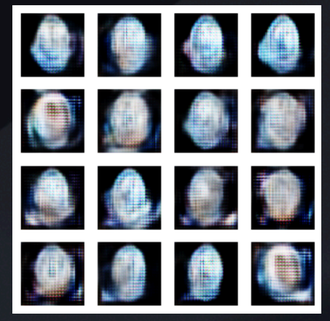
Epoch 5000



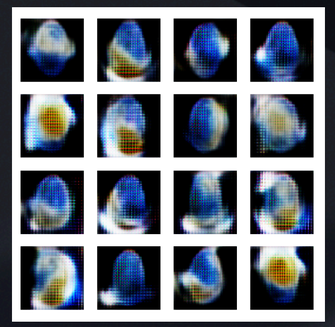
Epoch 5800



Epoch 6800



Epoch 9700

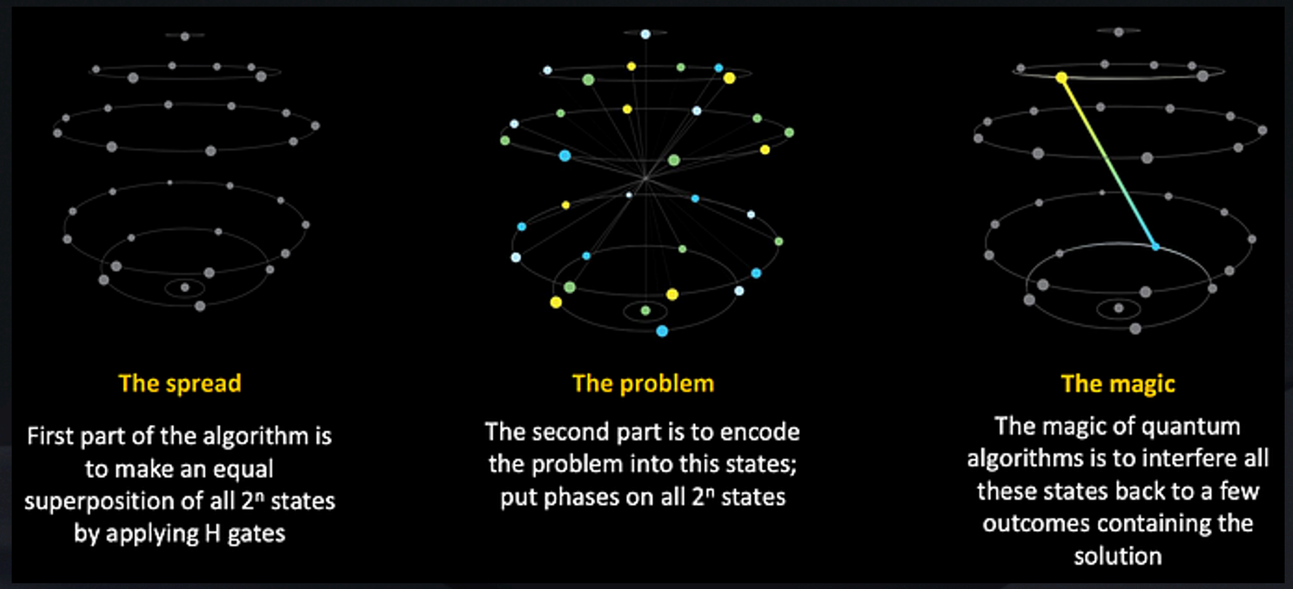


Quantum Computing

Quantum Computing

Introduction

Qubits are the core of quantum computing, they are in superposition until they are measured where collapse into one of two posible states



$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle$$

When qubits are measured, the capacity is no different from bits. It's extremely tricky to design algorithms under this constraint

In a system of n qubits:

$$|\psi\rangle = |q_1 q_2 q_3 q_4\rangle = |1010\rangle$$

Quantum computing take advantage of two properties of quantum nature: entanglement and interference.

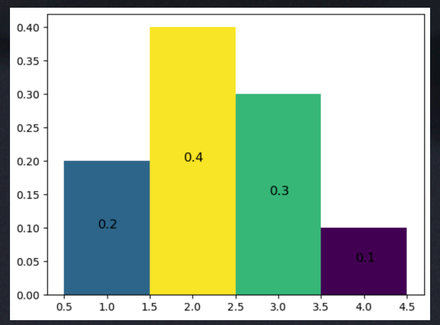
Quantum Machine Learning

Variational Quantum Circuits

In order to facilitate the realization of QML,
it is important to enable efficient **representation** of the given training data

Quantum data loading:
Mapping of classical data into a quantum state or quantum channel
$\mathscr{D} : X \to \mathscr{H}$ where $\mathscr{H}$ represents a Hilbert space of dimension 2^n
using a type of quantum data encoding (Quantum Feature Maps)

1D Normalized Discrete Probability distribution



Amplitude Encoding:

$$\mathscr{E}(X) = \sum_{j=0}^{k-1} \sqrt{p_j}\,|j\rangle$$

$$p = \begin{cases} 0.2 & \text{si} \quad x = 1 \\ 0.4 & \text{si} \quad x = 2 \\ 0.3 & \text{si} \quad x = 3 \\ 0.1 & \text{si} \quad x = 4 \end{cases}$$

Basis Encoding:

$$\mathscr{E}(X) = \frac{1}{\sqrt{k}} \sum_{j=0}^{k-1} |x_j\rangle$$

$$x = 5 = 101$$

$$|\psi\rangle = |101\rangle$$

$$|\psi\rangle = \sqrt{p_1}\,|1\rangle + \sqrt{p_2}\,|2\rangle + \sqrt{p_3}\,|3\rangle + \sqrt{p_4}\,|4\rangle$$

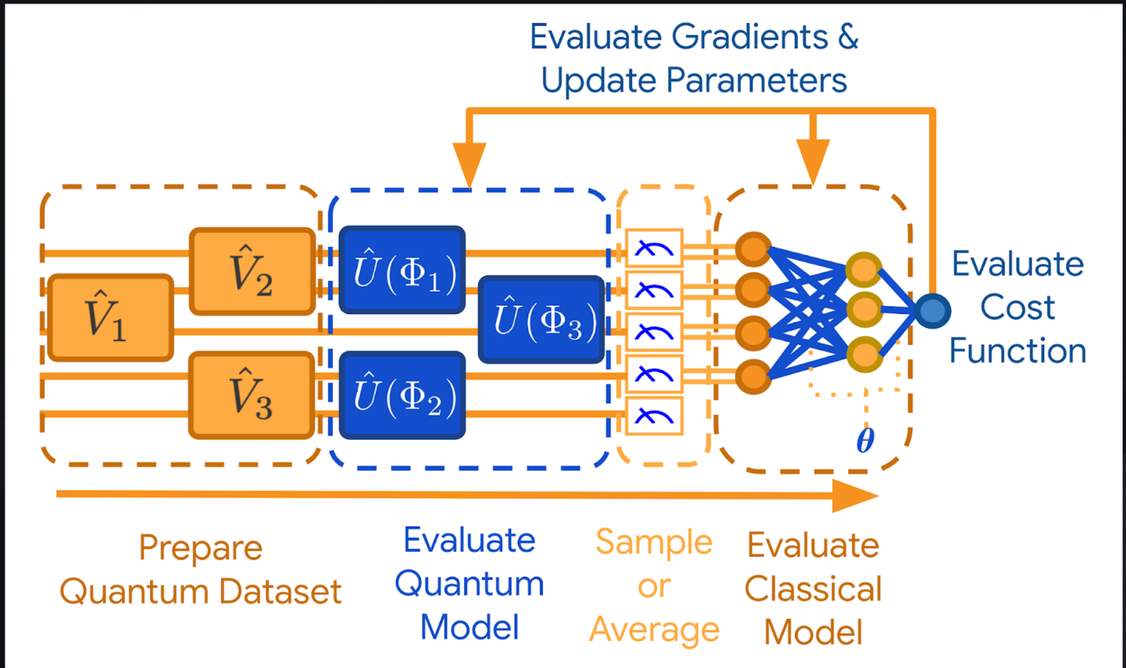
$$|\psi\rangle = \sqrt{0.2}\,|0001\rangle + \sqrt{0.4}\,|0010\rangle + \sqrt{0.3}\,|0100\rangle + \sqrt{0.1}\,|1000\rangle$$

Quantum Machine Learning

Variational Quantum Circuits

Parameterized quantum state:

$$|\psi(\omega)\rangle = U(\omega)|0\rangle^{\otimes n}$$



Reach the ground state

Cost function:

$$L(|\psi(\omega)\rangle) = \langle\psi(\omega)|\hat{O}|\psi(\omega)\rangle$$

Optimization problem:

$$\min_{\omega} L(|\psi(\omega)\rangle$$

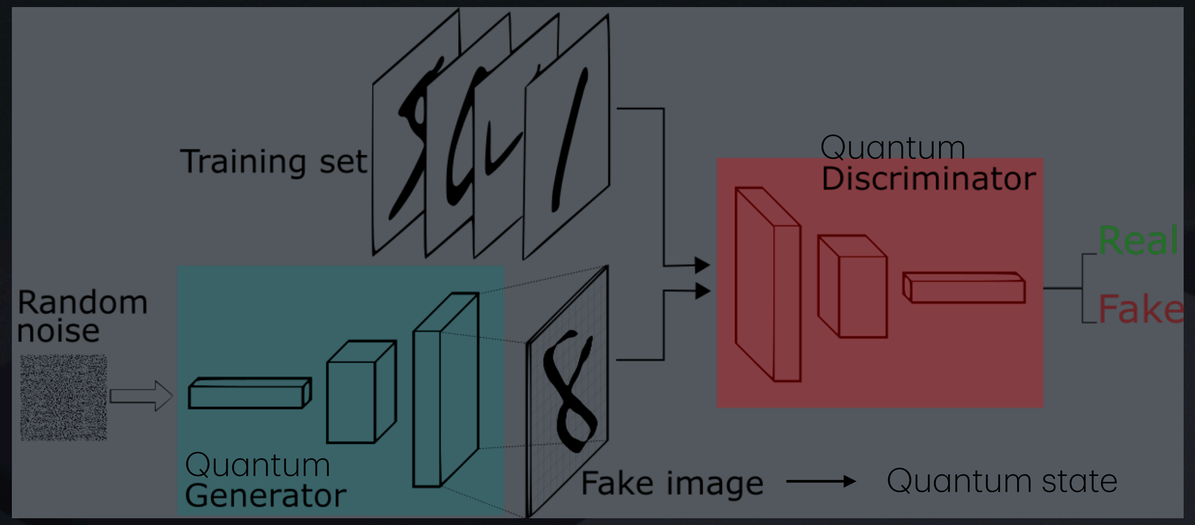
Quantum Generative Adversarial Networks

Quantum GANs

Introduction

Quantum GANs employ an interplay of a generator and a discriminator to learn an implicit approximate representation of a **probability distribution** underling given data samples into a variational quantum circuits

Quantum generator
generates quantum states
↓
Variational Quantum Circuit



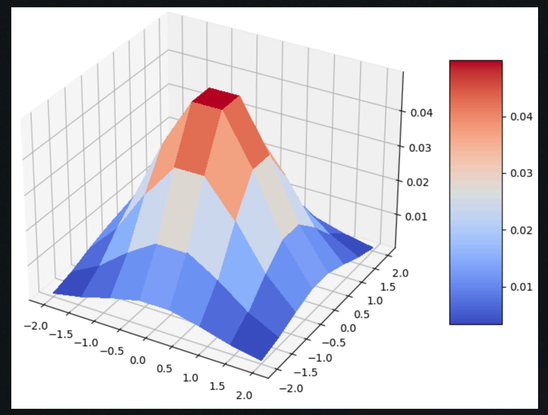
Quantum discriminator
measures quantum states
↓
Classical Neural Network

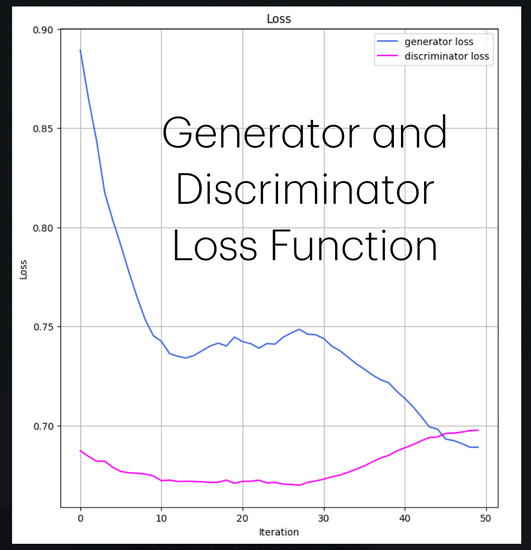
The generator goal is to minimize the distance between the generated and target quantum states
The discriminator goal is to discriminate correctly between these states

Quantum GANs

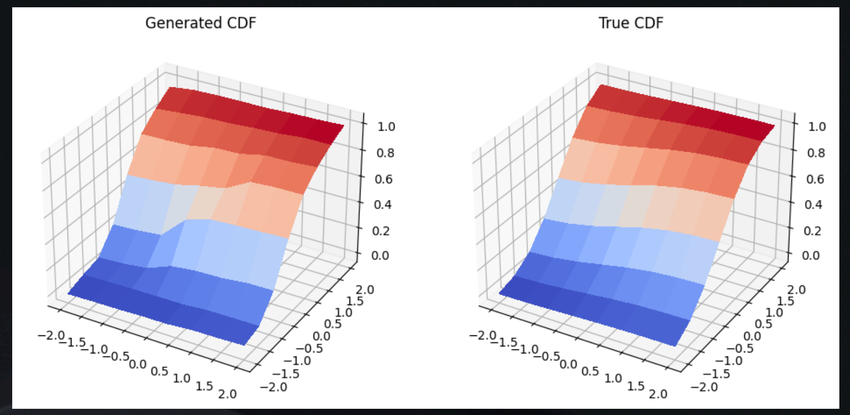
Statistical Example Arquitecture

Discrete Bivariate Normal Distribution

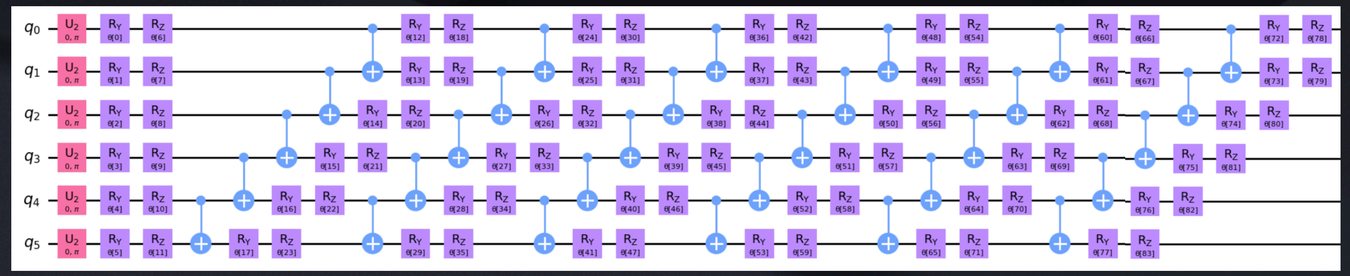




Cumulative Density Function



Generator as a Quantum Neural Network



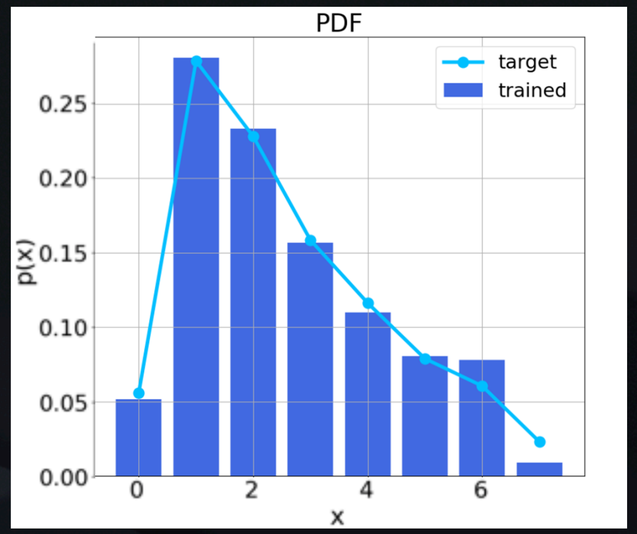
Discriminator as a Classical Neural Network

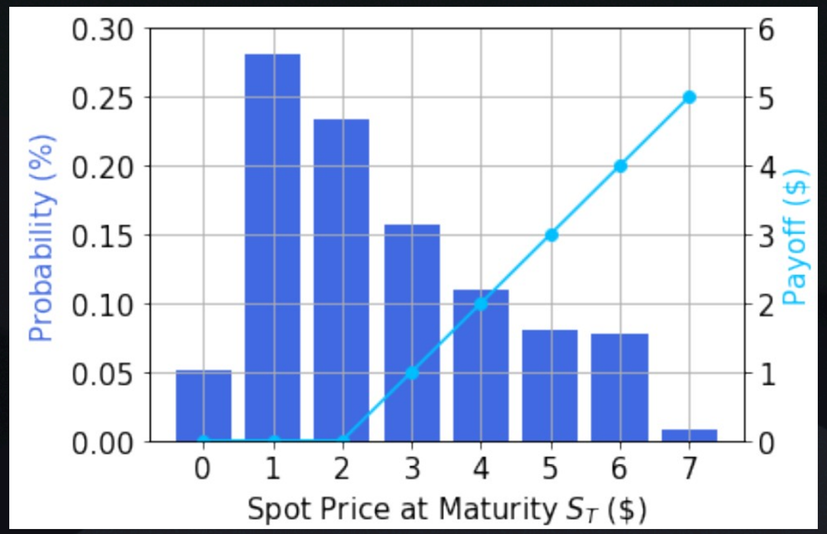
```
Discriminator(
  (linear_input): Linear(in_features=2, out_features=20, bias=True)
  (leaky_relu): LeakyReLU(negative_slope=0.2)
  (linear20): Linear(in_features=20, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)
```

Quantum GANs

Quantum Finance Example

Example of application of a hybrid qGAN in the loading of a log-normal distribution associated with the price of an European call option





The use of this method allows the potential of quantum computing to be exploited without being affected by the large number of quantum gates used in the loading of a distribution in a variational circuit

Thanks for your attention

References

Papers and websites with code implementations

- Ian J. Goodfellow, et al. (2014). Generative Adversarial Nets.
- TensorFlow. Deep Convolutional Generative Adversarial Network. Retrieved from: https://www.tensorflow.org/tutorials/generative/dcgan?hl=es-419
- Generative Adversarial Networks implementation. Retrieved from: https://nn.labml.ai/gan/original/index.html
- Martin Arjovsky, et al. (2017). Wasserstein GAN.
- Ishaan Gulrajani, et al. (2017). Improved Training of Wasserstein GANs.
- Nanny ML. Wasserstein Distance Explained. Retrieved from: https://www.youtube.com/watch?v=9rqmkDsUK2g&t=118s
- WGAN: Wasserstein Generative Adversarial Networks. Retrieved from: https://blog.paperspace.com/wgans/
- WGANs: Explained. Retrieved from: https://medium.com/@mladenkorunoski/wgans-explained-fab61bda2576
- Wasserstein GAN: TensorFlow Implementation using MNIST. Retrieved from: https://github.com/Mohammad-Rahmdel/WassersteinGAN-Tensorflow
- QC — What are Qubits in Quantum computing? Retrieved from: https://jonathan-hui.medium.com/qc-what-are-qubits-in-quantum-computing-cdb3cb566595
- QC — Programming with Quantum Gates (Single Qubits) Retrieved from: https://jonathan-hui.medium.com/qc-programming-with-quantum-gates-8996b667d256
- Christa Zoufal. (2021). Generative Quantum Machine Learning.
- Christa Zoufal, et al. (2019). Quantum Generative Adversarial Networks for Learning and Loading Random Distributions.
- Qiskit. PyTorch qGAN implementation. Retrieved from: https://qiskit.org/ecosystem/machine-learning/tutorials/04_torch_qgan.html