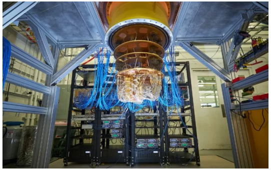
Introduction to Quantum Computing
Semester Project Presentation

Dimas Christos

Department of Electrical and Computer Engineering
Technical University of Crete

July 1, 2025



Conclusion

Introduction

Introduction

About this presentation

This presentation is a semester project for the course *Introduction to Quantum Computing* and aims to explain the ***Quantum Generative Adversarial Networks for learning and loading random distributions*** paper, a novel approach into using hybrid Quantum-Classical algorithms in order to avoid the data loading bottleneck and use the real quantum speedup. This paper was written by Christa Zoufal, Aurélien Lucchi and Stefan Woerner. In addition of the paper's presentation I will try to reproduce or if possible improve, some of their results and confirm the successfull results seen in the paper.

The Core Challenge: The Data Loading Bottleneck

Why do we need a new way to load data?

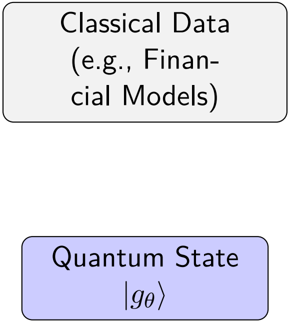


Figure: The problem of loading classical data into a quantum state.

Definition

Problem Statement

Why do we need a new way to load data?

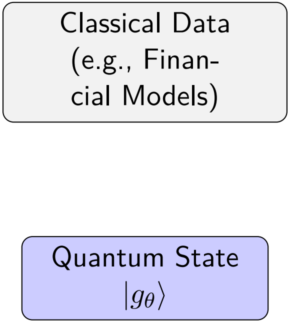


Figure: The problem of loading classical data into a quantum state.

Definition

Problem Statement

- Quantum Algorithms often need classical data as input.

Why do we need a new way to load data?

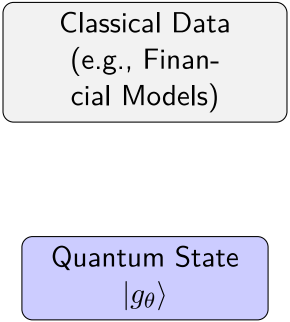


Figure: The problem of loading classical data into a quantum state.

Definition

Problem Statement

- Quantum Algorithms often need classical data as input.
- Known methods for exact loading require an exponential number of gates $O(2^n)$.

Why do we need a new way to load data?

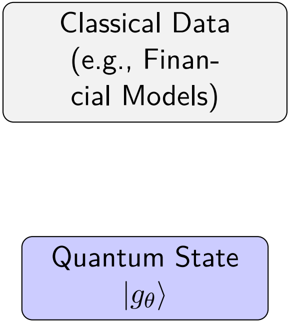


Figure: The problem of loading classical data into a quantum state.

Definition

Problem Statement

- Quantum Algorithms often need classical data as input.
- Known methods for exact loading require an exponential number of gates $O(2^n)$.
- This classical bottleneck can completely erase any potential quantum speedup.

Why do we need a new way to load data?

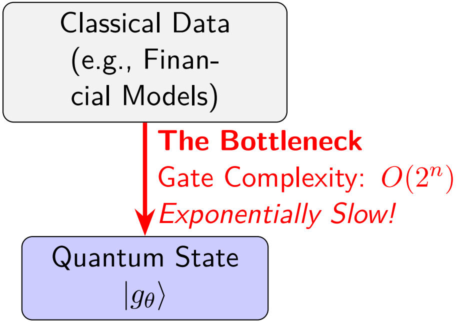


Figure: The problem of loading classical data into a quantum state.

Definition

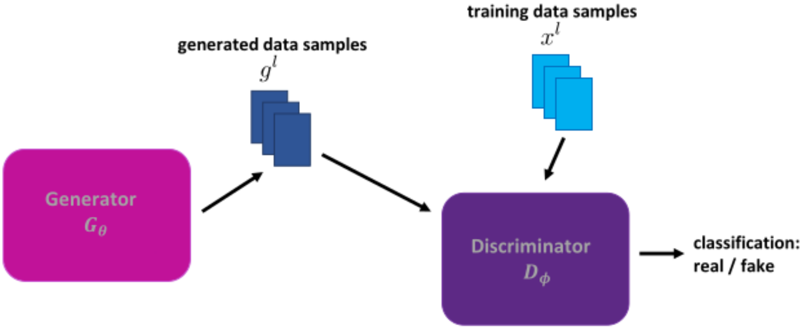
Problem Statement

- Quantum Algorithms often need classical data as input.
- Known methods for exact loading require an exponential number of gates $O(2^n)$.
- This classical bottleneck can completely erase any potential quantum speedup.

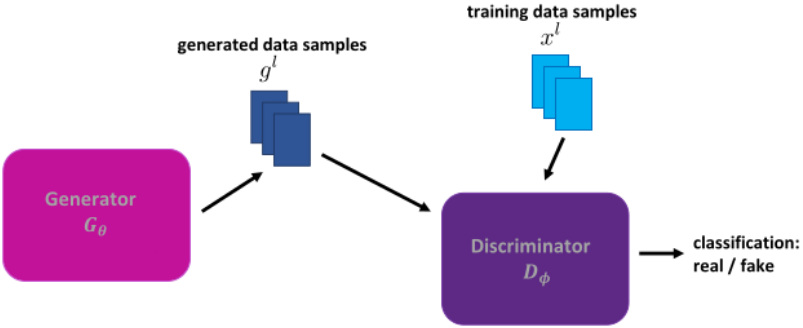
Background: Generative Adversarial Networks (GANs)

The Adversarial Game: Generator VS Discriminator

The Adversarial Game: Generator VS Discriminator



The Adversarial Game: Generator VS Discriminator



Key Idea: They are trained together. The Generator gets better at making realistic fakes, and the Discriminator gets better at spotting them. The goal is to train the Generator so well that it can fool the Discriminator.

Loss functions

In order to train the model effectivelly we need to define the loss functions.

1. Generator loss:
   - $L_G(\phi, \theta) = -\frac{1}{m} \sum_{l=1}^{m} [\log(D_\phi(G_\theta(z^l)))]$
2. Discriminator loss:
   - $L_D(D_\phi, G_\theta) = \frac{1}{m} \sum_{l=1}^{m} [\log D_\phi(x^l) + \log(1 - D_\phi(G_\theta(z^l)))]$

They are mathematicly related as expected and they are based on the classical binary-cross entropy methods.

The Proposed Solution: Quantum GAN (qGAN)

A Hybrid Quantum-Classical Approach

A Hybrid Quantum-Classical Approach

- **Quantum Generator (G_θ):**
  - A Parametrized Quantum Circuit (PQC).
  - Learns to produce a quantum state $|g_\theta\rangle$ that encodes the target data distribution.

A Hybrid Quantum-Classical Approach

- **Quantum Generator (G_θ):**
  - A Parametrized Quantum Circuit (PQC).
  - Learns to produce a quantum state $|g_\theta\rangle$ that encodes the target data distribution.
- **Classical Discriminator (D_ϕ):**
  - A standard feed-forward Neural Network.
  - Tries to distinguish real data from the "fake" data generated by the quantum circuit.

A Hybrid Quantum-Classical Approach

- **Quantum Generator (G_θ):**
  - A Parametrized Quantum Circuit (PQC).
  - Learns to produce a quantum state $|g_\theta\rangle$ that encodes the target data distribution.
- **Classical Discriminator (D_ϕ):**
  - A standard feed-forward Neural Network.
  - Tries to distinguish real data from the "fake" data generated by the quantum circuit.

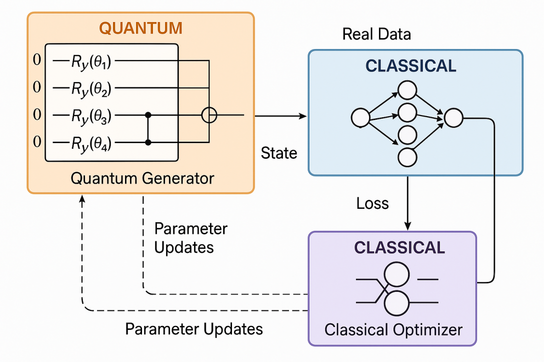


Figure: The qGAN training loop.

Parametrized Quantum Circuits (PQC)

Parametrized Quantum Circuits

Parametrized Quantum Circuits (PQC) are quantum circuits that consist of both fixed and parametrized gates. Fixed gates are typically multi-qubit entangling gates, such as C-NOT and C-Z. Parametrized gates are single-qubit rotation gates (R_Y and R_Z) and the operations of these gates depend on a set of adjustable classical parameters, θ, that define the rotation angles. They are often used inside Variatonal Quantum Circuits (VQAs) and form a Machine Learning model.

The Quantum Generator

Quantum circuit

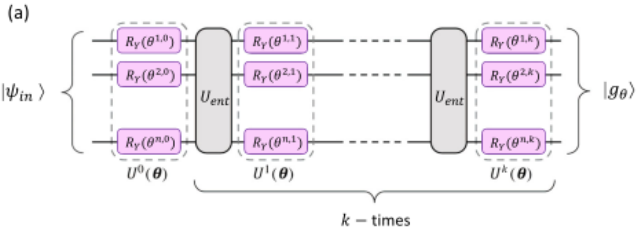
Quantum circuit

Circuit parts

An initial $R_Y(\theta)$ rotation gate layer. Alternating k layers of:

- Rotation gates $R_Y(\theta)$, the trainable parameters.
- Entangling gates CZ, in order to create complex correlations.

Quantum circuit



Circuit parts

An initial $R_Y(\theta)$ rotation gate layer. Alternating k layers of:

- Rotation gates $R_Y(\theta)$, the trainable parameters.
- Entangling gates CZ, in order to create complex correlations.

Generator Creation

Generator Creation

The application of the above circuit onto the initial state, $|\psi_{in}\rangle = |0\rangle^{\otimes n}$ is:

$$|g_\theta\rangle = G_\theta|\psi_{in}\rangle = \prod_{p=1}^{k}\left(\otimes_{q=1}^{n}(R_Y(\theta^{q,p}))U_{ent}\right)\otimes_{q=1}^{n}(R_Y(\theta^{q,0}))|\psi_{in}\rangle$$

Generator Creation

The application of the above circuit onto the initial state, $|\psi_{in}\rangle = |0\rangle^{\otimes n}$ is:

$$|g_\theta\rangle = G_\theta|\psi_{in}\rangle = \prod_{p=1}^{k}\left(\otimes_{q=1}^{n}(R_Y(\theta^{q,p}))U_{ent}\right)\otimes_{q=1}^{n}(R_Y(\theta^{q,0}))|\psi_{in}\rangle$$

We can make some assumptions easier though, by observing that when G_θ acts on $|0\rangle$ the result will always be the first column of G_θ.

Generator Creation

The application of the above circuit onto the initial state, $|\psi_{in}\rangle = |0\rangle^{\otimes n}$ is:

$$|g_\theta\rangle = G_\theta|\psi_{in}\rangle = \prod_{p=1}^{k}\left(\otimes_{q=1}^{n}(R_Y(\theta^{q,p}))U_{ent}\right)\otimes_{q=1}^{n}(R_Y(\theta^{q,0}))|\psi_{in}\rangle$$

We can make some assumptions easier though, by observing that when G_θ acts on $|0\rangle$ the result will always be the first column of G_θ.

$$G_\theta|0\rangle^{\otimes n} = \otimes_{i=1}^{n}\begin{bmatrix} a & b \\ c & d \end{bmatrix}\cdot\begin{bmatrix} 1 \\ 0 \end{bmatrix} = \otimes_{i=1}^{n}\underbrace{\begin{bmatrix} a \\ c \end{bmatrix}}_{a|0\rangle + c|1\rangle}$$

Generator Creation

The application of the above circuit onto the initial state, $|\psi_{in}\rangle = |0\rangle^{\otimes n}$ is:

$$|g_\theta\rangle = G_\theta|\psi_{in}\rangle = \prod_{p=1}^{k} \left(\otimes_{q=1}^{n} (R_Y(\theta^{q,p})) U_{ent} \right) \otimes_{q=1}^{n} (R_Y(\theta^{q,0})) |\psi_{in}\rangle$$

We can make some assumptions easier though, by observing that when G_θ acts on $|0\rangle$ the result will always be the first column of G_θ.

$$G_\theta|0\rangle^{\otimes n} = \otimes_{i=1}^{n} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \otimes_{i=1}^{n} \underbrace{\begin{bmatrix} a \\ c \end{bmatrix}}_{a|0\rangle + c|1\rangle}$$

Hence,

$$G_\theta|0\rangle^{\otimes n} = \sum_{j=0}^{2^n - 1} [G_\theta]_{j,0} |j\rangle$$

where the elements $[G_\theta]_{j,0}$ are complex functions of all the parameters θ in the circuit.

Generator Creation

So we conclude that:

$$|g_\theta\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_\theta^j}|j\rangle$$

Generator Creation

So we conclude that:

$$|g_\theta\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_\theta^j}|j\rangle$$

Also we can confirm that

$$\underbrace{O(n)}_{\substack{\text{first layer of} \\ R_Y \text{ rotations}}} \quad + \quad \underbrace{k \; \cdot O(n^p)}_{\substack{\text{entangler } U_{\text{ent}} \\ + \text{ subsequent } R_Y, \\ \text{repeated } k \text{ times}}} \quad = \; O\big(n+k\,n^p\big) \; = \; O\big(n+n^q\,n^p\big) \; = \; O\big(n^{\,p+q}\big) \; = \; O\big(\mathrm{poly}(n)\big).$$

Generator Creation

So we conclude that:

$$|g_\theta\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_\theta^j}|j\rangle$$

Also we can confirm that

$$\underbrace{O(n)}_{\substack{\text{first layer of} \\ R_Y \text{ rotations}}} + \underbrace{k \ \cdot O(n^p)}_{\substack{\text{entangler } U_{\text{ent}} \\ + \text{ subsequent } R_Y, \\ \text{repeated } k \text{ times}}} = O\big(n+k\,n^p\big) = O\big(n+n^q\,n^p\big) = O\big(n^{\,p+q}\big) = O\big(\text{poly}(n)\big).$$

Key takeaway: The gate complexity of the circuit now is $O(poly(n))$. Sounds like a massive improvement right?

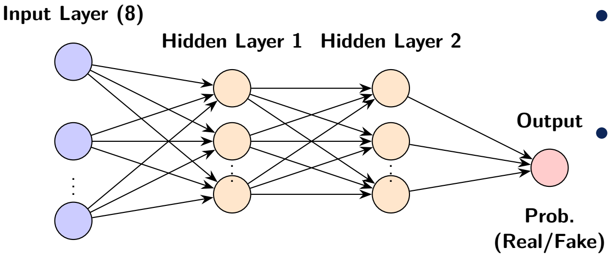
Classical Discriminator

Discriminator Creation

Discriminator Creation

- **Input Layer**: 8-dimensional, receives the flattened output from the quantum generator.
- **Hidden Layers**: 2 fully connected layers with the nonlinear activation function LeakyReLU, to capture complex patterns.
- **Output Layer**: A single neuron using sigmoid activation function, to express the output in probability terms.

Discriminator Creation



- **Input Layer**: 8-dimensional, receives the flattened output from the quantum generator.
- **Hidden Layers**: 2 fully connected layers with the nonlinear activation function LeakyReLU, to capture complex patterns.
- **Output Layer**: A single neuron using sigmoid activation function, to express the output in probability terms.

Dataset and Preperation

Let's "meet" the distributions

Let's "meet" the distributions

There are generated 3 target distributions for our generator to recreate:

Let's "meet" the distributions

There are generated 3 target distributions for our generator to recreate:

1. A Log-Normal, with $\mu = 1$ and $\sigma = 1$.

Let's "meet" the distributions

There are generated 3 target distributions for our generator to recreate:

1. A Log-Normal, with $\mu = 1$ and $\sigma = 1$.
2. A triangular, with l (lower limit) = 0, u (upper limit) = 7 and m = 2.

Let's "meet" the distributions

There are generated 3 target distributions for our generator to recreate:

1. A Log-Normal, with $\mu = 1$ and $\sigma = 1$.
2. A triangular, with l (lower limit) = 0, u (upper limit) = 7 and m = 2.
3. A bimodal, containing two Gaussian distributions with $\mu_1 = 0.5$, $\sigma_1 = 1$ and $\mu_2 = 3.5$, $\sigma_2 = 0.5$.

Let's "meet" the distributions

There are generated 3 target distributions for our generator to recreate:

1. A Log-Normal, with $\mu = 1$ and $\sigma = 1$.
2. A triangular, with l (lower limit) = 0, u (upper limit) = 7 and m = 2.
3. A bimodal, containing two Gaussian distributions with $\mu_1 = 0.5$, $\sigma_1 = 1$ and $\mu_2 = 3.5$, $\sigma_2 = 0.5$.

From each of the above mentioned distributions, 20,000 samples were generated and then truncated to the range $[0, 7]$, so we can have a natural mapping between training data and generator states.

Preperation

Preperation

A good habit we should all have, when working with such models is to always initialize the generator parameters and prepare the quantum state. In our case there were 3 ways to prepare the state:

Preperation

A good habit we should all have, when working with such models is to always initialize the generator parameters and prepare the quantum state. In our case there were 3 ways to prepare the state:

- A discrete uniform distribution, using a Hadamard gate in each qubit.

Preperation

A good habit we should all have, when working with such models is to always initialize the generator parameters and prepare the quantum state. In our case there were 3 ways to prepare the state:

- A discrete uniform distribution, using a Hadamard gate in each qubit.
- A discrete normal distribution, achieved by fitting the parameters of a 3-qubit variational quantum circuit with depth 1 with a least squares loss function.

Preperation

A good habit we should all have, when working with such models is to always initialize the generator parameters and prepare the quantum state. In our case there were 3 ways to prepare the state:

- A discrete uniform distribution, using a Hadamard gate in each qubit.
- A discrete normal distribution, achieved by fitting the parameters of a 3-qubit variational quantum circuit with depth 1 with a least squares loss function.
- A random distribution, by initializing the state to $|0\rangle^{\otimes n}$ and initializing circuit's parameters in order to follow a uniform distribution on $[-\pi, \pi]$

Preperation

A good habit we should all have, when working with such models is to always initialize the generator parameters and prepare the quantum state. In our case there were 3 ways to prepare the state:

- A discrete uniform distribution, using a Hadamard gate in each qubit.
- A discrete normal distribution, achieved by fitting the parameters of a 3-qubit variational quantum circuit with depth 1 with a least squares loss function.
- A random distribution, by initializing the state to $|0\rangle^{\otimes n}$ and initializing circuit's parameters in order to follow a uniform distribution on $[-\pi, \pi]$

For the first two ways the circuit parameters are set by a uniform distribution on $[-0.1, 0.1]$

Training Process

Training Process

Training Process

Training Steps (epochs)	2000
Batches	2000

Table: Training configuration

Training Process

Training Steps (epochs)	2000
Batches	2000

Table: Training configuration

Samples are fed, shuffled, into the discriminator in batches of 2000 samples per training step (epoch). There are totally 2000 training steps, which means that the discriminator is trained on the entire dataset multiple times, allowing it to learn the underlying patterns and characteristics of the data effectively.

Training Process

Training Steps (epochs)	2000
Batches	2000

Table: Training configuration

Samples are fed, shuffled, into the discriminator in batches of 2000 samples per training step (epoch). There are totally 2000 training steps, which means that the discriminator is trained on the entire dataset multiple times, allowing it to learn the underlying patterns and characteristics of the data effectively.

In the context of training a model to learn a random distribution, the Kolmogorov-Smirnov (KS) statistic as well as the relative entropy represent suitable measures to evaluate the training performance.

Application in Finance: Pricing a European Call Option

A Real-World Use Case

A Real-World Use Case

European call option

European call option is the right to to buy an underlying asset for a given strike price K at a predefined future maturity date T, where the asset's spot price at maturity S_T is assumed to be uncertain.

A Real-World Use Case

European call option

European call option is the right to to buy an underlying asset for a given strike price K at a predefined future maturity date T, where the asset's spot price at maturity S_T is assumed to be uncertain.

Payoff Function: $\max\{0, S_T - K\}$

A Real-World Use Case

European call option

European call option is the right to to buy an underlying asset for a given strike price K at a predefined future maturity date T, where the asset's spot price at maturity S_T is assumed to be uncertain.

Payoff Function: $\max\{0, S_T - K\}$

The Challenge

Calculate the expected payoff, for a future spot price S_T that is uncertain and follows a probability distribution.

A Real-World Use Case

European call option

European call option is the right to to buy an underlying asset for a given strike price K at a predefined future maturity date T, where the asset's spot price at maturity S_T is assumed to be uncertain.

Payoff Function: $\max\{0, S_T - K\}$

The Challenge

Calculate the expected payoff, for a future spot price S_T that is uncertain and follows a probability distribution.

Remember

The qGAN can be used to efficiently load a probability distribution into a quantum computer.

Evaluation model

Evaluation model

In order to compare the metrics of different methods, we need to introduce a refernece model. This is going to be the Black-Scholes model.

Evaluation model

In order to compare the metrics of different methods, we need to introduce a refernece model. This is going to be the Black-Scholes model.

Black-Scholes model

Black-Scholes is a model used in finance to price option contracts. It gives a theoritical estimate of the price of European-style options only, because American could be exercised before expiration date.

Evaluation model

In order to compare the metrics of different methods, we need to introduce a refernece model. This is going to be the Black-Scholes model.

Black-Scholes model

Black-Scholes is a model used in finance to price option contracts. It gives a theoritical estimate of the price of European-style options only, because American could be exercised before expiration date.

According to the Black-Scholes model, the spot price at maturity S_T is assumed to follow a Log-Normal distribution like the ones created before, so the qGAN can be trained to learn this distribution successfully, after a carefull initialization though, with normal and uniform distributions to constitute the best options based on the previous experiments.

A Note

A Note

In more realistic and complex cases, where the spot price follows a more generic stochastic process or where the payoff function has a more complicated structure, options are usually evaluated with Monte Carlo simulations.

Quantum Advantage with QAE

What is QAE?

What is QAE?

Quantum Amplitude Estimation algorithm

Quantum Amplitude Estimation algorithm (QAE) is an algorithm used to estimate the unknown amplitudes of a certain state.

What is QAE?

Quantum Amplitude Estimation algorithm

Quantum Amplitude Estimation algorithm (QAE) is an algorithm used to estimate the unknown amplitudes of a certain state.

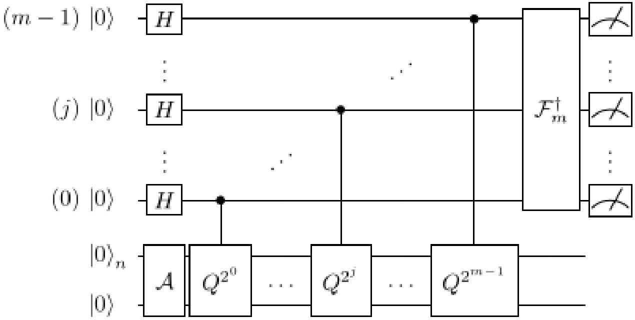


Figure: QAE circuit

Circuit Analysis

Circuit Analysis

1. Hadamard on all evaluation qubits, to create superposition.

Circuit Analysis

1. Hadamard on all evaluation qubits, to create superposition.
2. Unitary operator A, acts on the initial state plus one ancilla qubit. This gives us

$$A|0\rangle^{\otimes(n+1)} = \sqrt{1-\alpha}|\psi_0\rangle^{\otimes n}|0\rangle + \sqrt{\alpha}|\psi_1\rangle^{\otimes n}|1\rangle$$

letting us use an other operator to estimate the probability a.

Circuit Analysis

1. Hadamard on all evaluation qubits, to create superposition.
2. Unitary operator A, acts on the initial state plus one ancilla qubit. This gives us

$$A|0\rangle^{\otimes(n+1)} = \sqrt{1-\alpha}|\psi_0\rangle^{\otimes n}|0\rangle + \sqrt{\alpha}|\psi_1\rangle^{\otimes n}|1\rangle$$

   letting us use an other operator to estimate the probability a.
3. The following unitary operator, Q, is called Grover operator. This operator is used because its eigenvalues are directly related to amplitude: $\lambda_{1,2} = \exp^{\pm 2i\theta_\alpha}$, where $\alpha = \sin^2(\theta_\alpha)$. The m additional evaluation qubits in the algorithm are required to control the applications of $Q = -AS_0A^\dagger S_{\psi_0}$, where $S_0 = I^{\otimes(n+1)} - 2|0\rangle\langle 0|^{\otimes(n+1)}$ and $S_{\psi_0} = I^{\otimes(n+1)} - 2|\psi_0\rangle\langle\psi_0| \otimes |0\rangle\langle 0|$

Circuit Analysis

1. Hadamard on all evaluation qubits, to create superposition.
2. Unitary operator A, acts on the initial state plus one ancilla qubit. This gives us

$$A|0\rangle^{\otimes(n+1)} = \sqrt{1-\alpha}|\psi_0\rangle^{\otimes n}|0\rangle + \sqrt{\alpha}|\psi_1\rangle^{\otimes n}|1\rangle$$

   letting us use an other operator to estimate the probability a.
3. The following unitary operator, Q, is called Grover operator. This operator is used because its eigenvalues are directly related to amplitude: $\lambda_{1,2} = \exp^{\pm 2i\theta_\alpha}$, where $\alpha = \sin^2(\theta_\alpha)$. The m additional evaluation qubits in the algorithm are required to control the applications of $Q = -AS_0A^\dagger S_{\psi_0}$, where $S_0 = I^{\otimes(n+1)} - 2|0\rangle\langle 0|^{\otimes(n+1)}$ and $S_{\psi_0} = I^{\otimes(n+1)} - 2|\psi_0\rangle\langle\psi_0| \otimes |0\rangle\langle 0|$
4. Finally, an Inverse Quantum Fourier Transform (IQFT) is applied to the evaluation register, to transform the phase information into a computational basis state and the measurements.

Problem Solved: European Option Price

Estimation Process

Estimation Process

Now, qGAN model was trained on real quantum hardware (IBM Q Boeblingen) with 20 qubits, using a batch size of 2000 samples and a circuit with the architecture mentioned above with 1 layer depth for 200 training steps (epochs).

Estimation Process

Now, qGAN model was trained on real quantum hardware (IBM Q Boeblingen) with 20 qubits, using a batch size of 2000 samples and a circuit with the architecture mentioned above with 1 layer depth for 200 training steps (epochs).
The resulting quantum generator loads a random distribution that approximates the spot price at maturity S_T, in order to evaluate the expected payoff $\mathbb{E}[\max\{ST - K, 0\}]$, for $K = 2$ approximate data loading.

Estimation Process

Now, qGAN model was trained on real quantum hardware (IBM Q Boeblingen) with 20 qubits, using a batch size of 2000 samples and a circuit with the architecture mentioned above with 1 layer depth for 200 training steps (epochs).
The resulting quantum generator loads a random distribution that approximates the spot price at maturity S_T, in order to evaluate the expected payoff $\mathbb{E}[\max\{ST - K, 0\}]$, for $K = 2$ approximate data loading.

Figure: Payoff function European Call option. Probability distribution of the spot price at maturity S_T and the corresponding payoff function for a European Call option.

Estimation Comparisson

Estimation Comparisson

More specifically, they integrated the distribution loading quantum channel into a quantum algorithm based on QAE, using m $= 8$ evaluation qubits, i.e., $2^8 = 256$ quantum samples and into a Monte Carlo simulation, working with 1024 random samples. Estimations of $\mathbb{E}[\max\{ST - K, 0\}]$ are:

Estimation Comparisson

More specifically, they integrated the distribution loading quantum channel into a quantum algorithm based on QAE, using m $= 8$ evaluation qubits, i.e., $2^8 = 256$ quantum samples and into a Monte Carlo simulation, working with 1024 random samples. Estimations of $\mathbb{E}[\max\{ST - K, 0\}]$ are:

Method	Distribution	Samples	CI
1. Analytic (Exact)	$X \sim$ Log-N(1, 1)	-	-
2. Monte Carlo (Quantum HW)	$\lvert g_\theta \rangle$	1024	± 0.0848
3. QAE (Simulated)	$\lvert g_\theta \rangle$	256	± 0.0710

Table: Comparisson of option price estimation methods, their samples and their confidence intervals.

Estimation Comparisson

At this point it is worth noting that the estimates and CIs of Monte Carlo and QAE evaluation are not subject to the same level of noise effects. Monte Carlo simulation solely run on actual quantum hardware, while QAE on a quantum simulator. In order to run QAE on a quantum computer, further improvements are required, e.g., longer coherence times and higher gate fidelities.

Method	Distribution	Samples	CI
1. Analytic (Exact)	$X \sim$ Log-N(1, 1)	-	-
2. Monte Carlo (Quantum HW)	$\lvert g_\theta \rangle$	1024	± 0.0848
3. QAE (Simulated)	$\lvert g_\theta \rangle$	256	± 0.0710

Table: Comparisson of option price estimation methods, their samples and their confidence intervals.

Reproducing the Results

Validating the Paper's Findings

Validating the Paper's Findings

Findings that were reproduced:

Validating the Paper's Findings

Findings that were reproduced:

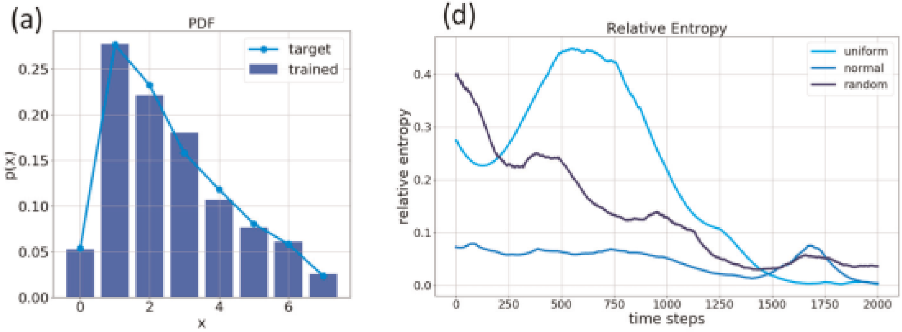


Figure: Comparisson of the real and the generated distribution, with the generator initialized with a uniform distribution (left) and relative entropy of the generator (right).

Challenges Faced

Challenges Faced

The main challenges were on how to process and manage the dataset effectivelly. This was needed because the IN/OUT values for Discriminator were 8/1 and for Generator 8/8.

Challenges Faced

The main challenges were on how to process and manage the dataset effectivelly. This was needed because the IN/OUT values for Discriminator were 8/1 and for Generator 8/8.

Challenges

Solutions

Challenges Faced

The main challenges were on how to process and manage the dataset effectivelly. This was needed because the IN/OUT values for Discriminator were 8/1 and for Generator 8/8.

Challenges

1. Trying to compute loss functions and gradients with casual methods didn't work.

Solutions

Challenges Faced

The main challenges were on how to process and manage the dataset effectivelly. This was needed because the IN/OUT values for Discriminator were 8/1 and for Generator 8/8.

Challenges

1. Trying to compute loss functions and gradients with casual methods didn't work.

Solutions

1. Use seperate loss functions. Generator computes and sums up the loss, on a probability distribution $\sum_{i=0}^{7} L_{G_i}$, where L_{G_i} the generator loss value for each discrete value.

Challenges Faced

The main challenges were on how to process and manage the dataset effectivelly. This was needed because the IN/OUT values for Discriminator were 8/1 and for Generator 8/8.

Challenges

1. Trying to compute loss functions and gradients with casual methods didn't work.
2. Even when the solution was found, we should intergate effective batch functionallity to reduce computational load.

Solutions

1. Use seperate loss functions. Generator computes and sums up the loss, on a probability distribution $\sum_{i=0}^{7} L_{G_i}$, where L_{G_i} the generator loss value for each discrete value.

Challenges Faced

The main challenges were on how to process and manage the dataset effectivelly. This was needed because the IN/OUT values for Discriminator were 8/1 and for Generator 8/8.

Challenges

1. Trying to compute loss functions and gradients with casual methods didn't work.
2. Even when the solution was found, we should intergate effective batch functionallity to reduce computational load.

Solutions

1. Use seperate loss functions. Generator computes and sums up the loss, on a probability distribution $\sum_{i=0}^{7} L_{G_i}$, where L_{G_i} the generator loss value for each discrete value.
2. Use one-hot encoding for batches, to ensure dimension compatibility between the two models and effective classical processing.

Final Modifications

Final Modifications

The line followed was the same as mentioned above in the paper's methods, except some minor changes.

Final Modifications

The line followed was the same as mentioned above in the paper's methods, except some minor changes.

- Circuit parameters were initialized at $[-\delta, \delta]$ with 10^{-1}, that was now changed to 10^{-2}, giving us less divergence in the state probabilties.

Final Modifications

The line followed was the same as mentioned above in the paper's methods, except some minor changes.

- Circuit parameters were initialized at $[-\delta, \delta]$ with 10^{-1}, that was now changed to 10^{-2}, giving us less divergence in the state probabilties.
- Learning rate was set to $2 \cdot 10^{-4}$ instead of 10^{-4}, that helped to match the change above.

Final Modifications

The line followed was the same as mentioned above in the paper's methods, except some minor changes.

- Circuit parameters were initialized at $[-\delta, \delta]$ with 10^{-1}, that was now changed to 10^{-2}, giving us less divergence in the state probabilties.
- Learning rate was set to $2 \cdot 10^{-4}$ instead of 10^{-4}, that helped to match the change above.

These two changes should lead us into later convergence of the loss functions but, slightly better results regarding the relative entropy and KS statistic, based on empirical knowledge from previous runs.

The Results

The Results

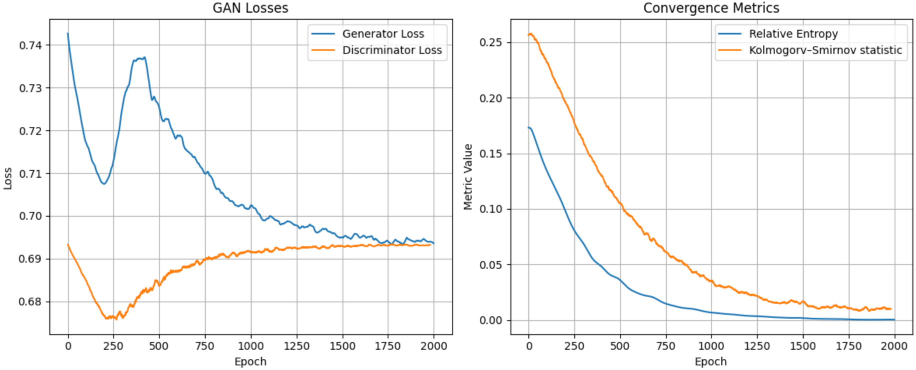
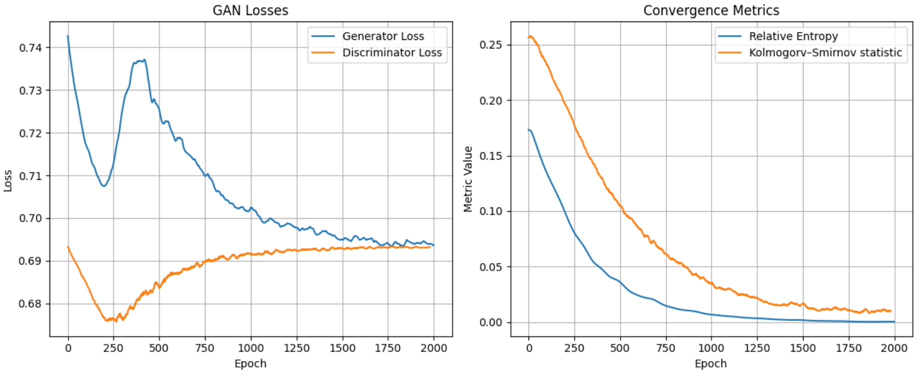


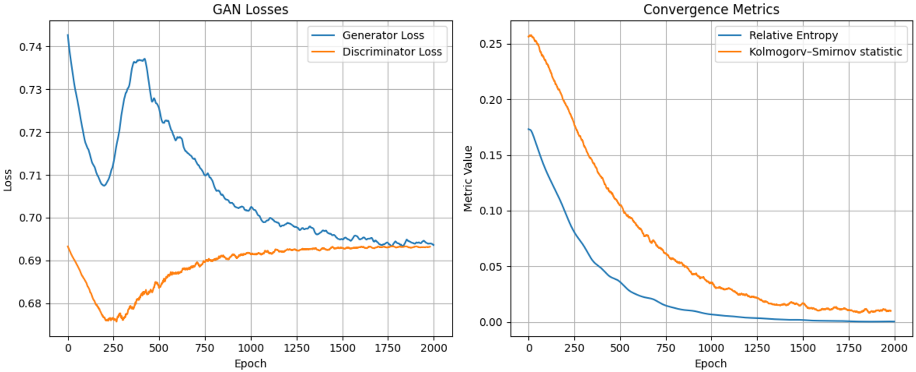
Figure: Training loss functions (left) and convergence metrics (right)

The Results



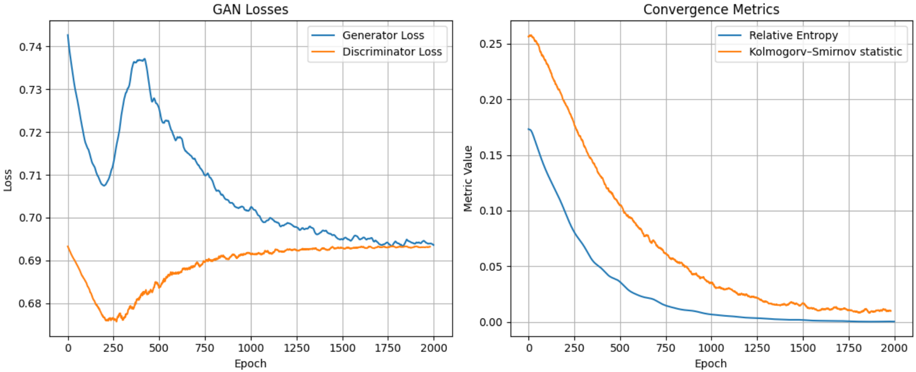
Confidence level of KS statistic was set to 95% from paper authors. Here 99.19% was achieved.

The Results



Relative entropy reaches 0.0004, when the training is finished.

The Results



Loss functions converge later during the training process.

The Results

The Results

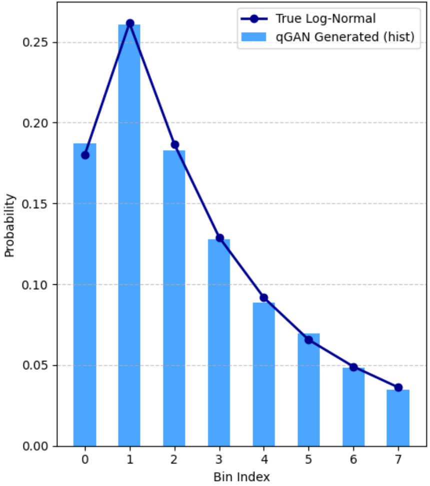


Figure: True Log-Normal versus qGAN generated

Comparisson (1/4)

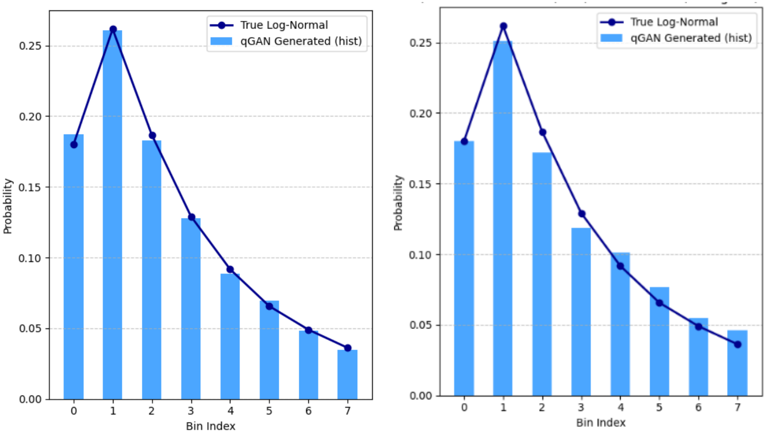


Figure: My setup results (left) in comparisson with paper's setup results (right)

Comparisson (2/4)

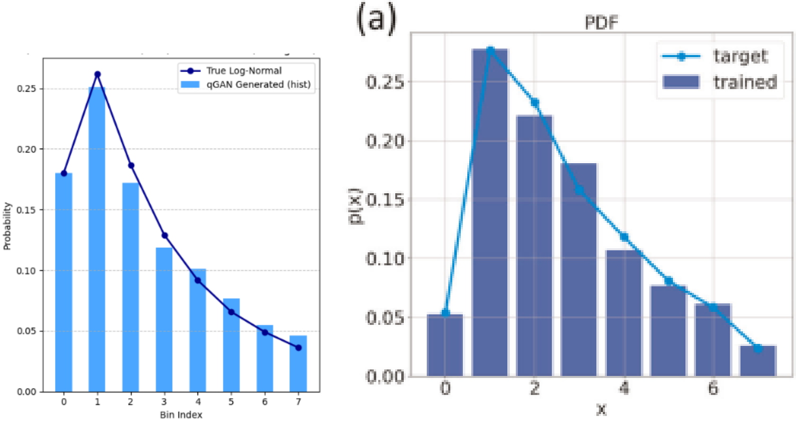
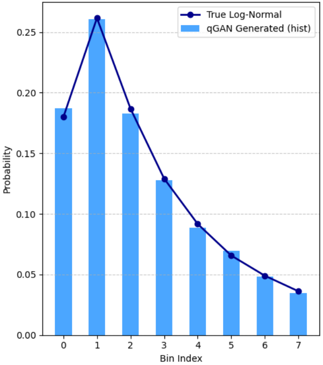


Figure: **Paper's setup**: My results (left) in comparisson with paper's results (right)

Comparisson (3/4)



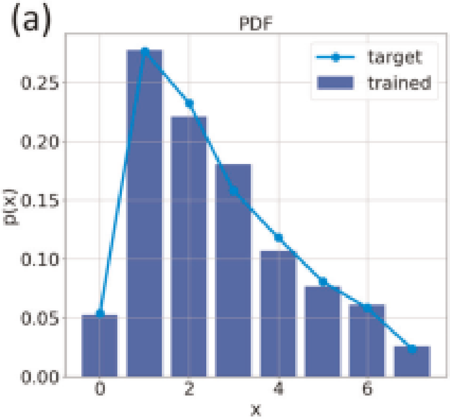


Figure: My results (left) in comparisson with paper's results (right)

Comparisson (4/4)

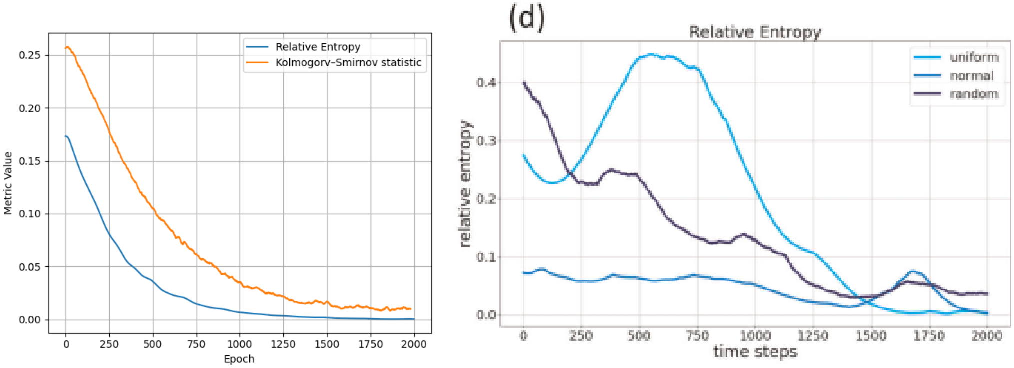


Figure: My results (left) in comparisson with paper's results (right)

Discussion & Key Takeaways

What I Learned from This Project

What I Learned from This Project

- qGANs are really powerfull! Imaging having the huge quantum advantage and in the same time being able to generate such realistic data, that can in a way predicting the future.

What I Learned from This Project

- qGANs are really powerfull! Imaging having the huge quantum advantage and in the same time being able to generate such realistic data, that can in a way predicting the future.
- The "hybrid" aspect is critical. Success depends just as much on classical machine learning techniques (like data representation and loss functions) as it does on the quantum circuit design.

What I Learned from This Project

- qGANs are really powerfull! Imaging having the huge quantum advantage and in the same time being able to generate such realistic data, that can in a way predicting the future.
- The "hybrid" aspect is critical. Success depends just as much on classical machine learning techniques (like data representation and loss functions) as it does on the quantum circuit design.
- **Limitation**: Training is very sensitive to hyperparameters, requiring careful and many times empirical fine-tuning.

Conclusion & Future Work

Conclusion & Future Work

Conclusion & Future Work

Overview

This review and reproduction confirm that qGANs offer a viable, resource-efficient path to loading data on quantum computers, making a wider range of algorithms more practical.

Conclusion & Future Work

Overview

This review and reproduction confirm that qGANs offer a viable, resource-efficient path to loading data on quantum computers, making a wider range of algorithms more practical.

Ideas:

Conclusion & Future Work

Overview

This review and reproduction confirm that qGANs offer a viable, resource-efficient path to loading data on quantum computers, making a wider range of algorithms more practical.

Ideas:

- Scaling the model to more qubits and more complex distributions.

Conclusion & Future Work

Overview

This review and reproduction confirm that qGANs offer a viable, resource-efficient path to loading data on quantum computers, making a wider range of algorithms more practical.

Ideas:

- Scaling the model to more qubits and more complex distributions.
- Running those experiments on real quantum hardware to study the effects of noise.

Conclusion & Future Work

Overview

This review and reproduction confirm that qGANs offer a viable, resource-efficient path to loading data on quantum computers, making a wider range of algorithms more practical.

Ideas:

- Scaling the model to more qubits and more complex distributions.
- Running those experiments on real quantum hardware to study the effects of noise.
- Intergrating this loading technique into other quantum algorithms.

References (1/2)

- Zoufal, C., Lucchi, A., & Woerner, S. (2019). *Quantum generative adversarial networks for learning and loading random distributions*. npj Quantum Information, 5(1), 103. Available: https://www.nature.com/articles/s41534-019-0223-2.
- David McMahon. (2007). *Quantum computing explained. John Wiley & Sons*.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). *Generative Adversarial Networks*. Advances in Neural Information Processing Systems, 27. Available: https://arxiv.org/abs/1406.2661
- Qiskit Development Team. (2023). *Qiskit: An Open-source Framework for Quantum Computing*. [Online]. Available: https://qiskit.org.
- Qiskit Development Team. *Qiskit: PyTorch qGAN Implementation*. Available: https://qiskit.org/ecosystem/machine-learning/tutorials/04_torch_qgan.html.

References (2/2)

- D. Angelakis and team (2025). *Course's lecture notes*
- Medium (2024). *One Hot Encoding Explained*. Available: https://medium.com/@heyamit10/one-hot-encoding-explained-0b0130ccd78e
- Quantum Amplitude Estimation paper (2000). *Quantum Amplitude Amplification and Estimation*. Available: https://arxiv.org/pdf/quant-ph/0005055

Thank you!

cdimas@tuc.gr