A HYBRID QUANTUM WASSERSTEIN GAN WITH APPLICATIONS TO OPTION PRICING

FELIX FUCHS AND BLANKA HORVATH

Abstract. This paper conducts an in-depth exploration of harnessing Machine Learning techniques in the context of Quantum methods for Option Pricing. We develop and implement a novel hybrid Quantum Wasserstein GAN for loading arbitrary distributions from a classical state into a quantum state, which is of interest beyond its financial applications. In particular, our hybrid methodology eliminates several potential sources of instability and has superior performance compared to fully quantum generative models if the target distribution is classical. Our qWGAN can be used to capture the probability distribution of an asset at maturity, and transmuting it into a quantum state as demonstrated on synthetic as well as real data experiments. In an Option Pricing context we present a full pipeline using this methodology and leveraing the Iterative Quantum Amplitude Estimation algorithm to derive the expected option payoff, ensuring a quadratic enhancement in error scaling compared to traditional methods.

Contents

Date: July 18, 2023.
The authors would like to thank Oleksiy Kondratyev for helpful and inspiring remarks to our results.

1. Introduction

1.1. The Landscape of Quantum Machine Learning for Option Pricing

In recent years, there has been a growing interest in exploring quantum computing (QC) in financial research, see (see eg Nature, July 2023). Portfolio optimization and credit risk analysis were the first areas of application to be addressed in this field by quantum computing – in particular quantum annealing–more recently followed by option pricing. Quantum technologies promise a great potential to significantly speed up computational processes, and these technologies, combined with methods from the field of machine learning can yield additional synergies. In this section we recall existing research on option pricing with quantum computers and place our work within this research landscape, motivate the advantages of our approach and highlight its contributions to the literature.

The foundation for the recent rapid growth of research activity for applications of quantum algorithms was laid by the research of Ashley Montanaro in 2015. In the seminal work **Montanaro2015**, Montario demonstrates a quadratic speed up for Monte Carlo methods by utilizing quantum algorithms. Montarios results have contributed to a growing interest in exploring the potential of quantum computers to speed up computational processes and deliver more accurate results. This acceleration had an impact in multiple fields where Monte Carlo methods are extensively used, such as Statistical Physics, Microelectronics as well as Mathematical Finance. Building on these foundations, **Rebentrost_2018** demonstrate in their 2018 paper the prowess of these techniques in the context of option pricing: They present a procedure for pricing vanilla options within the Black-Scholes framework that can bee seen as a quantum-based Monte Carlo method. The paper demonstrates in particular, that for a given accuracy, the number of samples required grows at a slower rate than for classical Monte Carlo methods: For a desired accuracy of ϵ, the quantum algorithm has a $1/\epsilon$ dependence on the number of samples, while classical methods exhibit $1/\epsilon^2$ dependence. Since the results in **Rebentrost_2018** are a cornerstone with considerable impact on later research (including our paper), we summarize a few of its main points here: Their procedure can be broken down into three steps:
(I) Loading a distribution function into a quantum state (here, an iterative method for loading integrable distribution functions is employed).
(II) Linking the payoff function of the option to an additional qubit (ancilla qubit), and
(III) concatenating the ancilla qubit with the loaded distribution function and measuring the expected value of the qubit using Quantum Phase Estimation (QPE), where the use of QPE contributes the quadratic advantage over classical Monte Carlo methods. In this paper we also adhere to this three-step procedure, but replace the first and third steps with more efficient methods, which we motivate below.

The above three-step methodology of **Rebentrost_2018** has since been widely adopted in quantum option pricing research, but various improvements were made in the individual steps in subsequent years and adapted to individual applications: For instance, **Pracht2022** employed the procedure shown in **Rebentrost_2018** to price Asian options, replacing step (I) of **Rebentrost_2018** (loading the distribution) by a method proposed by **Isometry_Iten** for enhanced efficiency[1]. Similarly, **Bermuda_Miyamoto_2021** determine the price of Bermuda options under the assumption that the distribution to be loaded can be efficiently transformed into a quantum state.
On the matter of loading random distributions into a quantum state (step (I) of **Rebentrost_2018**), the work of **Zoufal_2019** has brought considerable improvements to the previous state of the art. Their work presents a technique that significantly enhances the efficiency of this

[1]This approach can also become computationally costly as the number of qubits increases.

procedure and demonstrates considerable improvement compared to previous techniques requiring only $\mathcal{O}(poly(n))$ quantum gates for n qubits onto which the distribution is to be loaded, as opposed to prior results which require $\mathcal{O}(2^n)$ gates. The proposed procedure employs quantum generative adversarial networks (introduced in **Lloyd_2018**, henceforth qGANs) to construct a quantum state distribution loading channel. This channel is not only more efficient but also more flexible than classical loading channels, as it imposes no restrictions on the distribution that is to be loaded.
In 2021, a novel variant of Quantum Amplitude Estimation (step (III) of **Rebentrost_2018**) called Iterative QAE (IQAE) has been introduced by **grinko2021iterative**, which does not rely on QPE[2]. This innovation offers significant advantages, in particular the considerable reduction in the number of quantum gates required compared to QPE-based algorithms. This innovation also enhances the implementability under the current quantum mechanical constraints, as currently a higher number of gates still involves an inherent instability of the quantum computer. In their empirical study, **grinko2021iterative** demonstrate that their algorithm significantlyoutperforms other known QAE variants without QPE. A direct consequence of this is that IQAE requires significantly fewer samples to achieve the same estimation accuracy and confidence level. It is therefore this type of Quantum Amplitude Extimation that we will be using in our current work as well.

In summary, our work–as many others–also follows the general three-step procedure proposed in **Rebentrost_2018** with some key modifications in the individual steps: In this paper, we replace their method for the step (I) by the more promising technique of using qGANs as outlined in **Zoufal_2019**. This idea of using qGANs is further significantly enhanced and improved upon in our proposed algorithm, compared to the original innovations suggested in **Zoufal_2019**. The main contributions of our paper lie in the innovations presented in this part.
In step (II), we replicate the the procedure in **Rebentrost_2018**, where the payoff profile of a European Call is loaded onto an ancilla qubit. Furthermore, we outline how other payoff structures can be handled and also demonstrate this step on other payoff profiles (digital options) which yield equally good results.
For the third step (combining the payoff with the output of step (I) and measuring the expected value of the resulting qubit) the original method uses Quantum Phase Estimation. QPE remains unstable on current noisy intermediate-scale quantum (NISQ) hardware and is not well scalable with the number of qubits. For this reason (and for the several reasons argued in the previous paragraph) we employ instead the IQAE introduced by **grinko2021iterative** in this third and final step of **Rebentrost_2018**.

As mentioned above, the main contributions of our work lie in the improvement achieved in step (I), which not only replaces the step of loading the distribution into a quantum state by a method using qGANs, as proposed in **Zoufal_2019**, but further improves on the latter in numerous ways: For a start, we increase the number of qubits compared to the experiments presented in **Zoufal_2019**. While **Zoufal_2019** has established a proof of concept for this methodology, it is not directly obvious how a similar qGAN setup would perform with higher numbers of qubits[3]. For our setup we (successfully) incorporate a higher number of in order to accomodate the applications we are interested in: A larger number of qubits enables us to consider a more nuanced grid of possible values of the stock price distribution at maturity. A more major difference—and improvement—compared to **Zoufal_2019** in our paper, is the update to its version as a (hybrid) Wasserstein

[2]Instead, this method is based solely on Grover's Algorithm, see Section 2.1.2.

[3]See eg. the *"Barren plateau problem"* **mcclean2018barren** for training Quantum Neural Networks as the number of qubits grows (cf. Section 2.2).

GAN (WGAN). In (standard) Machine Learning, Wasserstein GANs have become dominant in the field of image generation **arjovsky2017wasserstein** due to a number of desirable properties in their training. Their quantum versions have similarly promising properties (as recognised in **qWGAN_Chakrabarti**) but have not yet been studied extensively, to the best of our knowledge. In **qWGAN_Chakrabarti** it is shown that (similarly to their standard counterparts) *quantum* Wasserstein GANs (qWGANs) have the potential of improving the adversarial training as well as robustness and scalability of quantum generative models even on noisy quantum hardware. With this in mind, the central contribution of our work is to harness this potential by designing and implementing a concrete example of a quantum of a Wasserstein GAN, in a way that we believe to combine stability with efficiency: For a number of reasons–motivated in Sections 1 and 4.1 below–we opted for a *hybrid* qWGAN, with a standard discriminator and a quantum generator equipped with Wasserstein loss. In recent literature surrounding quantum machine learning (see **Herman2023** for a review) and related applications in finance there is an increasing number of contributions (eg. **Zoufal_2019**) recognising the advantages of modular *hybrid*-quantum-ML models: The current state-of-the-art is moving in a direction where not the entire work-flow, but only one (carefully chosen) module of it is infused with a quantum network. Our results presented in this work reinforce this recognition arguing that a fully quantum generator is not ideal if the target distribution is classical (cf. Section 2.2.2 for details), further advocating a *hybrid* and modular approach to quantum ML.
Overall, our newest additions to the procedures of **Rebentrost_2018** lead to more stable and accurate results that outperform not only standard qGAN implementations for the purpose of option pricing, but also the existing qWGANs we are aware of in this space, culminating in our novel model - the hybrid quantum Wasserstein GANs (qWGAN). We demonstrate the prowess of our model for the crucial first step, as well as the efficiency of the entire improved three-step pipeline for option pricing in a number of numerical experiments, both on synthetic and on real data.

The rest of the paper is organised as follows: In Section 1 we proceed to motivate our proposed generative model. For this, we start by recalling properties of (standard) GANs and the advantages of WGANs in Section 1.2. In Section 1.3 we proceed to discuss their (fully) quantum counterparts, elaborate on their advantages and challenges and give a brief overview of the literature landscape in this area. In Section 1.4, we present the reasons for opting for a hybrid classical-quantum version of the Wasserstein GAN for our applications and discuss the contributions of this paper in this respect. In Sections 2, 3 and 4 we then lay out the details and implementation of our methods. We aim to make our results accessible to wide audiences who are not necessarily experts in Quantum Algorithms. Therefore, Section 2 contains a brief reminder of the most necessary background, notations and quantum algorithms used in this paper, before describing the setup of our hybrid qWGAN in detail in Section 2.2. Section 3 then proceeds to lay out the remainder of **Rebentrost_2018** approach with our proposed improvements, in particular Section 3.1 explains how step (I) uses the outputs of the hybrid qWGAN from 2.2, Section 3.2 details how the payoff function of an option is computed using an ancilla qubit and Section 3.3 explains how the IQAE described above is used to estimate the resulting probabilities of the qubits. Finally, Section 4 presents the implementation of the above in numerical experiments, Section 4.2 applies them to (i) verifiable synthetic data and (ii) real market data, and Section 5 concludes.

1.2. Classical GANs and Wasserstein GANs (WGANs)

Generative Adversarial Networks (GANs) were first proposed in 2014 by Goodfellow and his collaborators **Goodfellow_GAN**. In just a few years, GAN structures have gained immense popularity in the machine learning community due to their superior performance compared to other methods (such as variational autoencoders) in the task of generating realistic synthetic images. GANs consist of two neural networks, (the generator and the discriminator) set up in a game-theoretic framework, competing with each other: During training, the discriminator learns to judge the authenticity (real or synthetic) of a given datum, while the generator learns to produce "fake" (synthetic) data that the discriminator cannot distingush from real data samples. The generator and discriminator are trained alternately. Overall, the process of GAN training can be summarized as a two-player minimax game, where the following objective function is minimized:

$$\min_G \max_D \left(L_D - L_G\right) \tag{1}$$

where the generator minimizes the loss function L_G, while the discriminator minimizes the loss function L_D. The loss function L_G and L_D can be given as follows:

$$L_G := -\mathbb{E}_{z\sim p_z(z)}[\log D\left(G(z)\right)] \quad \text{(Generator loss)}$$

$$L_D := -\mathbb{E}_{x\sim p_{real}(x)}[\log D(x))] - \mathbb{E}_{z\sim p_z(z)}\left[\log\left(D\left(G(z)\right)\right)\right] \quad \text{(Discriminator loss)}.$$

This alternating adversarial training process allows the generator to produce increasingly realistic synthetic data, as it learns to generate samples that are more likely to be classified as real by the discriminator. The discriminator, in turn, adapts and improves its ability to distinguish between real and generated data, driving the generator to create even more convincing outputs. For any given dataset $X \in \mathbb{R}^{g_{out}}$ with (unknown) distribution $p_{real}(x)$ the generator G_θ network maps $G_\theta : \mathbb{R}^{g_{in}} \to \mathbb{R}^{g_{out}}$ and the discriminator D network maps $D : \mathbb{R}^{g_{out}} \to [0, 1]$, i.e. it outputs an estimate of the probability that the pattern is a genuine one from $p_{real}(x)$. The training is considered complete (and successful), if the discriminator's score for any given sample (fake/real) is 0.5. From this summary it already becomes clear that the training of GANs is a delicate balance act.

1.2.1. Wasserstein GANs (WGANs)

Due to the challenge of training the generator and the discriminator during the adversarial training, GANs are well known to be somewhat unstable and difficult to train[4]. To tackle the training issue of GANs, much attention has been devoted to research on the convergence of training GANs in classical machine learning: It is known from the seminal work of **arjovsky2017wasserstein** that the choice of the metric is critical for the stability of the performance in the training. Some of the standard metrics (loss functions) such as the Kullback-Leibler (KL) divergence, the Jensen-Shannon (JS) divergence, and the total variation (statistical) distance have been understood to yield unstable results for learning distributions supported by low dimensional generative models. Finally **arjovsky2017wasserstein** used the Wasserstein distance **Villani_2008** as a metric for measuring the distance between real and synthetic distributions and achieved a significantly improved performance. The Wasserstein loss[5] is—unlike the KL-divergence or the JS- divergence— continuous and differentiable almost everywhere, which makes it

[4]Though this statement is primarily known for classical GANs it was observed in **situ2020quantum** that this property is carried over to quantum counterparts, quantum GANs are prone to the same diffuculties in training as their classical counterparts. This is especially true for models with more than one qubit.

[5]In the version we use it in this paper, it also known as the Earth-Mover (EM) distance.

superior from an optimization perspective in terms of training stability and better convergence behaviour. Indeed, it was demonstrated in **Mescheder_2018** that WGAN and its variants[6] such as **Gulrajani_2017** have an improved training stability compared to original GAN formulations. Consequently, Wasserstein GANs (and its related variants) are the most superior algorithms among GANs available today.

1.3. (Fully) Quantum GANs and Wasserstein qGANs (or qWGANs)

When we speak of (fully) quantum GANs, we refer to a Generative Adverserial Network, where both generator and discriminator are constructed by quantum circuits, connected directly with the other. They together apply to a system of qubits as considered in several articles such as **DallaireDemers2018**, **Stein2021**, **Huang_2021**, **Shrivastava_2019**, **Hu_2019**, **Benedetti_2019**, **Du_2019**. The target distributions are hence (or can be) quantum, which can be fed directly into the network, or classical, which must be encoded to some quantum states before being input into the network. The workflow of fully quantum GANs is depicted in Figure 1. The noise from latent space puts the quantum system in the state $|z\rangle$. After being applied by the generator, the system is in state $|\psi\rangle$. At this time, in the case the real data are used, the real quantum data source outputs state $|x\rangle$ from the quantum system. The discriminator is then applied, and it changes the state to $|y\rangle$. The states $|y\rangle$, in the cases where the discriminator's input is real or generated, are used for the cost function and updating the parameters in the circuits.

In this variant of quantum GANs, both the generator and the discriminator are boosted by *"quantum supremacy"*, and they can benefit from two facts: First, the computational power is leveraged at an exponential rate in comparison with classical neural networks. Second, they can manipulate, learn from, and generate certain types of data that classical GANs cannot **Lloyd_2018**. For example, **Huang_2021** achieve the same performance with the quantum batch GAN using totally 21 parameters, where comparable classical GANs must have at least 106 parameters. The works which are most closely related to ours are **cortes1995support**, **lussange2021modelling**, **dong2008quantum** where specific GANs dealing with quantum data are considered. Among these, **lussange2021modelling**, **dong2008quantum** only handles the 1-qubit case (both pure and mixed) and **cortes1995support** discussed the pure state case (up to 6 qubits) but with the loss function being the quantum counterpart of the total variation distance.

As indicated in the first section, classical GANs are well known for being delicate and somewhat unstable in training, which can be (according to the results presented in **mitchell1990machine**) traced back to the choice of the metric between real and fake distributions. The shortcomings described in the previous section of standard metrics and loss functions for the training of GANs can be expected to carry through to their quantum counterparts and hence any quantum GANs based on the aforementioned standard metrics (see section 1.2.1) will likely suffer from the same weaknesses in training.
This (inheritance of) training issues was not directly observable or significant in numerical studies of quantum GANs in the 1-qubit case **lussange2021modelling**, **dong2008quantum**, however (as observed by **cortes1995support** and and also confirmed in our experiments) the training issue becomes much more significant when the quantum system scales up:

[6]There are several subsequent studies on various modifications of the WGAN, such as GAN with regularized Wasserstein distance **Sanjabi_2018**, WGAN with entropic regularizers **Cuturi_2013**, **Seguy_2017**, WGAN with gradient penalty **Gulrajani_2017**, **Petzka_2017**, relaxed WGAN **Guo_2017**, etc.

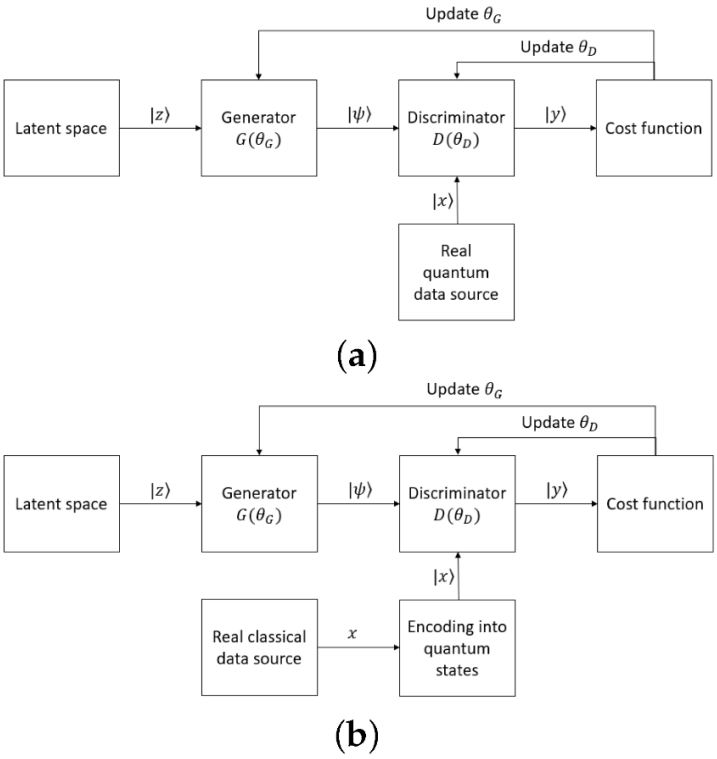


FIGURE 1. The workflow of a fully quantum network in the cases where the target data are (a) quantum and (b) classical.

It becomes significant already in the case of a few qubits. To tackle the training issue of classical GANs, much research attention has been directed towards the convergence of training GANs in classical machine learning, which established Wasserstein GANs (and its subsequent modifications, see Section 1.2.1) among the most popular and powerful generative models to date.

1.3.1. Wasserstein qGANs (or qWGANs)

The advantages of using Wasserstein loss in the training of a GAN were expected to carry over to its' quantum counterparts as well. Quantum Wasserstein GANs were only introduced to the research landscape very recently: Before 2021, all existing quantum GANs (for example **cortes1995support**, **lussange2021modelling**, **dong2008quantum**, **khoshaman2018 arjovsky2017wasserstein**, **karras2021alias**, **wang2018esrgan**, **zhu2017unpaired**,) have used loss functions other than the Wasserstein distance. Indeed, the first quantum Wasserstein GAN, proposed by Chakrabarti et al. **Chakrabarti_2021** confirms the conjecture of improved training performance and stability.

Quantum Wasserstein GANs are the quantum GANs that use Wasserstein distance, or Earth mover's distance (EMD), as their cost functions. Differently from other quantum GANs variants, the mission of the discriminator in Wasserstein GANs (WGANs) is not to distinguish between the real and the generated data. Due to the fact that Wasserstein distance contains a function that satisfies the Lipschitz condition, the discriminator acts as an estimator that seeks out the optimal function for the distance. After the distance is computed, it is used to update the parameters in the generator. The workflow of quantum WGANs is illustrated in Figure 2.

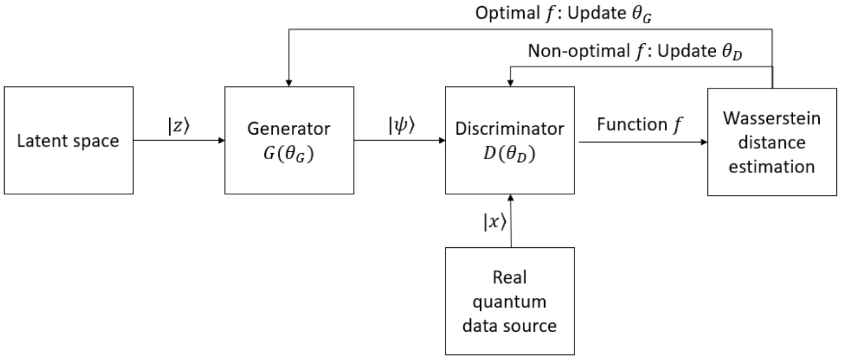


FIGURE 2. The general workflow of a quantum WGAN. For notations please see Section 2.

In the Wasserstein GAN proposed by Chakrabarti et al. **Chakrabarti_2021**, the generator consists of ensembles of unitaries $(p_1, U_1), \ldots, (p_r, U_r)$, parameterized by the set of parameters θ. Each tuple (p_i, U_i) means applying the unitary U_i, a 1-qubit or 2-qubit Pauli rotation quantum gate, with probability p_i. The discriminator contains linear combinations of tensor products of Pauli matrices that are parameterized by two sets of parameters, α and β, corresponding to ϕ and ψ, respectively, in the formula in Section 4.1.2. The gradients of the cost function with respect to p_i, α and β are computed directly, since the cost function is a linear function of those parameters. The gradients of the cost function with respect to θ are estimated by the parameter-shift method **schuld2019evaluating**. In a recent study Kiani et al. **kiani2022learning** have shown that this formula of quantum Wasserstein distance is unitarily invariant. In the same work, they developed another formula and applies it to their quantum WGAN **kiani2022learning**. The Wasserstein distance between two states, ρ and σ, is

$$D_{EM}(\rho, \sigma) = \max_{H \in O_n \text{ and } ||H||_{Lip} \leq 1} |\mathrm{Tr}((\rho - \sigma)H)|,$$

where O_n is the set of n-qubit observables. The quantum Lipschitz constant of the observable H must be at most one. The discriminator contains the parameterized sum of strings of Pauli operators and is supposed to estimate the quantum EMD. The generator in this network has the same architecture as that in Chakrabarti et al.'s work **Chakrabarti_2021**. The weights of the discriminator are updated by executing a linear program, the probabilities of the unitaries in the generator are updated by directly computing the gradients, and gradients of the cost function with respect to the unitary parameters are calculated via parameter-shift rules **schuld2019evaluating**. Quantum Wasserstein GANs can help overcome the training difficulties of quantum GANs with other metrics, such as discontinuous distances between real and generated distributions, mode collapse, and vanishing gradients. They can successfully estimate GHZ state **kiani2022learning**, up to 8 qubit pure states, up to 3 qubit mixed states, and even 4 qubit pure states with noise **Chakrabarti_2021**. They can also be used to approximate quantum circuits such as the 1D 3-qubit Heisenberg model circuit **childs2018toward** with an average output fidelity of 0.9999 **Chakrabarti_2021** and 8 qubit teacher circuits using student circuits of depth 4 with the fidelity of approximately 1 **kiani2022learning**.

1.4. Hybrid (quantum - classical) GANs and Our Contribution

Many quantum technologies are currently moving in this (hybrid) direction, where one selected part of a workflow is replaced by a quantum algorithm or similarly where an isolated quantum module is integrated into a more complex (classical) system. This has,

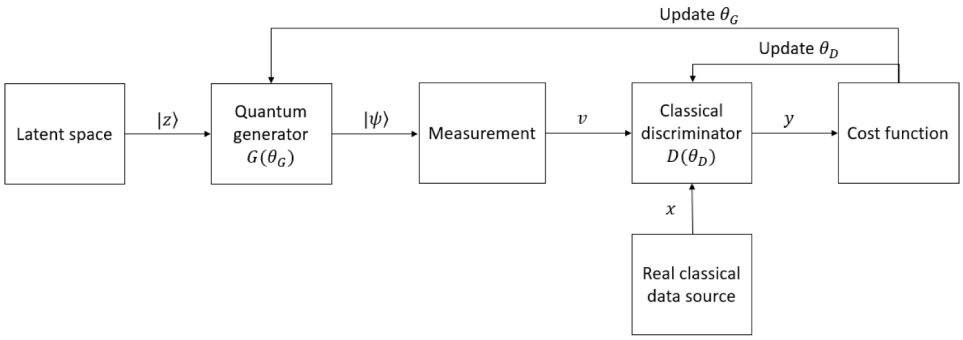


FIGURE 3. The workflow of a hybrid quantum–classical GAN. For notations please see Section 2.

in general, the advantage of better controllability and stability of the system as a whole, in the case of hybrid qGANs it has further advantages in addition (see below). A hybrid GAN could be any combination of a quantum and a classical module. In this section we lay out our rationale for the choice of our hybrid (quantum-classical) set-up. In practice, a GAN with a classical generator and a quantum discriminator is never used: As stated in **Lloyd_2018**, if the target distribution is quantum, it is impossible for the classical generator to estimate and learn such a distribution. On the other hand, if the target data are classical, the generator is able to generate such data, but it can always be beaten by the quantum discriminator. In this way, the training is easily thrown off balance and the generator never achieves convergence. On the other hand, a hybrid quantum–classical GAN with a quantum generator and a classical discriminator has good potential to be successful if the target distribution is classical. The architecture of such a hybrid quantum–classical GAN is illustrated in Figure 3. This variant of quantum GANs can generate classical data with extraordinary performance in comparison with traditional GANs: Even though "only" one module of this system (only the generator) has quantum power, it is enough for hybrid GANs to outperform traditional ones as shown in **situ2020quantum**. An additional advantage compared to fully quantum GANs is that (since the discriminator is classical) the target data do not need encoding, which eliminates the potential for errors in the encoding/decoding phase, each of which is often an unstable process[7]. However, since the generator is quantum (while the target distribution is classical) the states of the quantum system after generation must be measured (i.e. transformed into classical statistics, which are readable for the discriminator). In several preceding works this classical discriminator is simply a fully connected neural network **Huang_2021**, **Zoufal_2019**, **situ2020quantum**, with a binary output $\{0, 1\}$, corresponding to real and fake examples, respectively. This reduces the problem to a two-class classification. In **situ2020quantum**, a GAN with a quantum generator and a classical discriminator was able to generate discrete distributions, a difficult task for classical GANs to deal with. Most notably, the proposed hybrid GAN could easily converge after only about 1000 epochs.

1.4.1. Our Contribution

Our work combines the principles of Quantum Wasserstein GANs and the hybrid architecture, leading to a significant improvement in the efficiency and precision of quantum

[7]Since this encoding/decoding is prone to considerable errors in each of the steps, the overall outcome can offset one of the primary desirable properties of GANs (generative models): That they can reflect fine properties of complex distributions far more faithfully than traditional numerical methods.

computing applications. The centrepiece of our contribution is the development and implementation of a hybrid Quantum Wasserstein GAN (hybrid qWGAN). By employing the Wasserstein metric as the loss function for the quantum generator within this hybrid framework, we have substantially enhanced the accuracy and efficiency of the model in learning and loading arbitrary distributions into quantum states. This methodology outperforms traditional qGAN models (amongst other this by eliminating the need to convert the target distribution into quantum state) and thus can play a central role across a variety of quantum computing applications. Moreover, our research extends beyond the theoretical development of the hybrid qWGAN model, offering practical applications that underscore its potential. We demonstrate this by incorporating the hybrid qWGAN into an option pricing procedure on a quantum computer. This integration not only showcases the model's remarkable potential and superior precision but also highlights its versatility. We demonstrate this flexibility by loading beyond the lognormal distribution (following previous works in this direction) but also the Heston model and even transforming the distribution induced by the implied probability distribution of real Call Options on Apple stock into a quantum state using this methodology. This further illustrates the extensive applicability of our hybrid qWGAN approach. A possible python implementation of our work can be found *here*.

2. Background and Notations in Quantum Computing

This section is intended to make the this paper as self-contained as possible and easily accessible to a broad audience in quantitative finance. The contents of the section are mainly inspired by IBM Quantum Services **IBM_Q** and the textbook **Qiskit**.Furthermore, notations and definitions are based on the book "Quantum Computation and Quantum Information: 10th Anniversary Edition" **nielsen_chuang_2010**. The necessary notations and main principles can be found in this section as a reminder. Readers, who are confident in their familiarity with notations and basics of quantum algorithms can bona fide skip this section.

2.1. Notations, Concepts and Quantum Algorithms Used in this Work

In this section, we recall basic tools and notation of quantum computing and present a brief overview of the circuits used in subsequent chapters to make our results accessible to readers unfamiliar with quantum computing.

Quantum bits, or qubits, are the basis of quantum computing and are fundamentally different from classical bits. Unlike classical bits, which can only be in the states 0 or 1, qubits have a unique property called "superposition," which allows them to be in the basis states 0 and 1, but additionally, they can be in any linear combination of these two states and assume all of these possible states simultaneously in a probabilistic fashion. For the notation of the states of one or more qubits, the "Bra-Ket" notation is used as the standard in quantum mechanics. The quantum basis states (0 and 1) are denoted as "Kets" representing a column vector, e.g.:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

In terms of completeness, "Bras" are the respective row vectors $\langle 0| = \begin{pmatrix} 1 & 0 \end{pmatrix}$, giving rise to the notation of the scalar product $\langle 0|0\rangle = 1$. An arbitrary qubit state $|\psi\rangle$ can be expressed in terms of the basis states as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad \alpha, \beta \in \mathbb{C}. \tag{2}$$

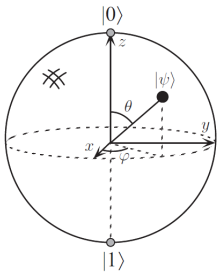


FIGURE 4. Bloch Sphere representation of a qubit.

with α and β being called amplitudes. Born's rule is a fundamental postulate in quantum mechanics that connects the amplitudes of quantum states to classical probabilities.

Definition 2.1 (Born's Rule)**.** For a given qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the probability of measuring the qubit in state $|i\rangle$ is given by the square of the absolute value of the inner product between $|i\rangle$ and $|\psi\rangle$, i.e., $|\langle i|\psi\rangle|^2$. Hence, the probability of measuring $|\psi\rangle$ to be in the state $|0\rangle$ is $|\langle 0|\psi\rangle|^2 = |\alpha|^2$, and the probability of measuring it to be in the state $|1\rangle$ is $|\langle 1|\psi\rangle|^2 = |\beta|^2$. It follows that

$$|\alpha|^2 + |\beta|^2 = 1. \tag{3}$$

Unfortunately, the amplitudes of qubits are not directly observable. Instead, it is possible to measure a qubit, resulting in the qubit collapsing its superposition and decaying into either $|0\rangle$ or $|1\rangle$ with the respective probability, but losing all other information about its initial state.

Due to (3) we can rewrite (2) as

$$|\psi\rangle = \exp^{i\gamma}\left(\cos\frac{\theta}{2}|0\rangle + \exp^{i\varphi}\sin\frac{\theta}{2}|1\rangle\right) \tag{4}$$

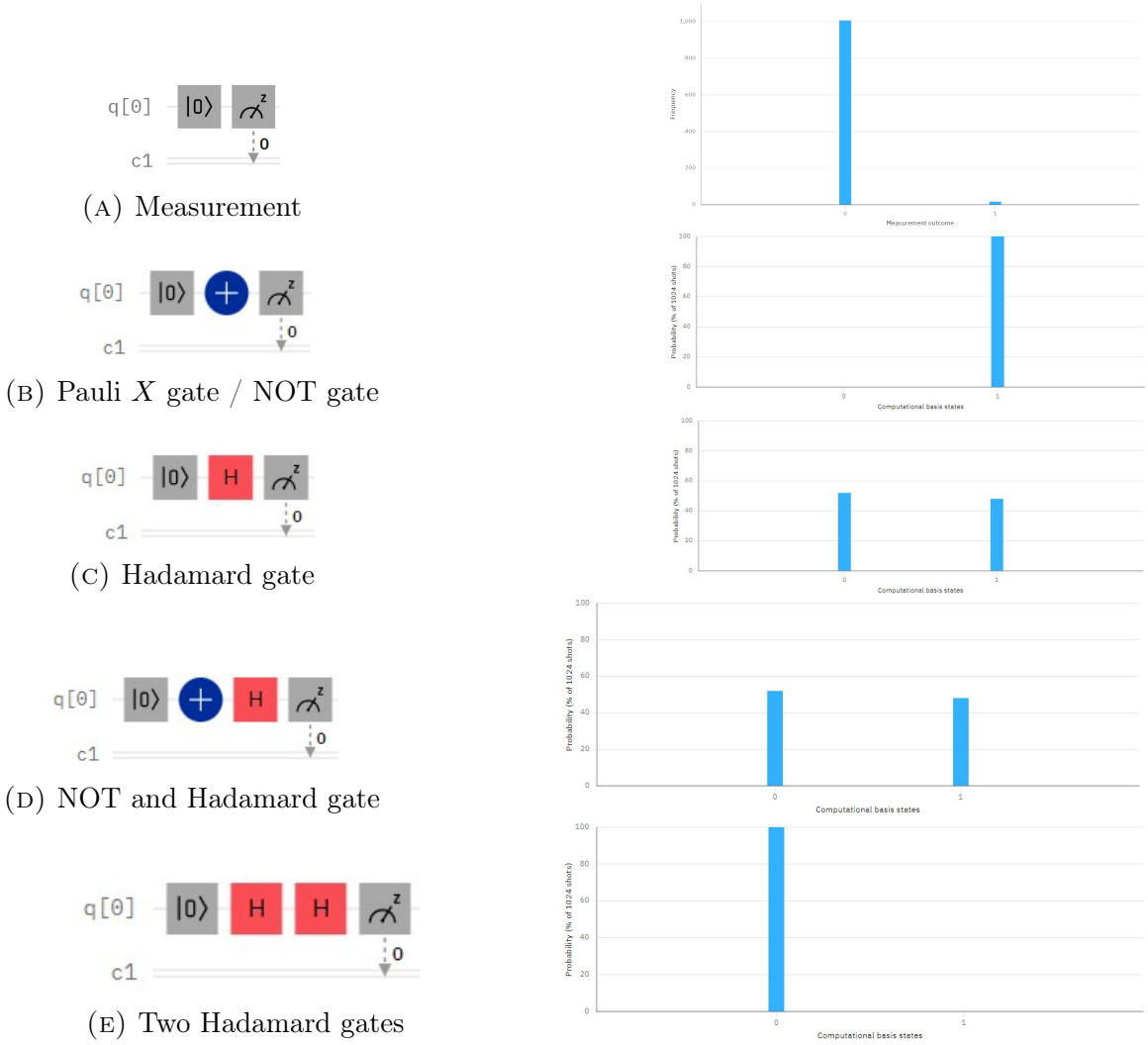
with $\theta, \varphi, \gamma \in \mathbb{R}$. The factor $\exp^{i\gamma}$ is often left out, and one can simply write

$$|\psi\rangle = \cos\frac{\theta}{2}|0\rangle + \exp^{i\varphi}\sin\frac{\theta}{2}|1\rangle \tag{5}$$

giving θ and φ a precise interpretation as dimensions in the Bloch Sphere, as shown in Figure 4. It should be noted that the Bloch sphere representation serves as a good tool for intuitively understanding a single qubit and will be helpful in many more operations. However, it has its limitations in representing the full repertoire of a qubit, especially when dealing with multi-qubit systems, and should thus be used with caution.

2.1.1. Quantum Gates

There are two key operations that can be performed on qubits: computational measurement and quantum logic gates. Quantum gates are the building blocks of quantum algorithms and are essential for manipulating the states of qubits. By stringing multiple gates together, circuits are created that can solve larger tasks. Here, fore demonstrative purposes we explore the basic gates, recreate them using the IBM quantum composer, and run them on quantum hardware "ibm_nairobi" with 7 qubits. In the common notation for circuits, there is one horizontal line for each qubit we are working on. Each operation to be executed on that qubit is placed on that line in the respective order from left to right. At the beginning of a circuit, a qubit is always in its ground state $|0\rangle$, and it is common to indicate this by placing it as the first operation. Additionally, we have a control line at the very bottom of all lines that returns binary classical information when



(A) Measurement

(B) Pauli X gate / NOT gate

(C) Hadamard gate

(D) NOT and Hadamard gate

(E) Two Hadamard gates

FIGURE 5. On the left, we see the circuit implementation of the gate, while on the right, the corresponding outcome is displayed. All images were sourced directly from the IBM Quantum Composer **IBM_Q**.

we measure a qubit. In Figure 5a, we can see that when measuring a qubit in the ground state, we get the expected result of 0. When executing tasks on a real quantum device, one can enter the number of *"shots"* to execute. The default value is 1024, which simply means how many times the circuit is set up and executed. Since quantum hardware is very sensitive and prone to noise and errors due to imperfect measurements and/or residual heating of the qubit, it makes sense to repeat the same experiment multiple times and take the average of the results to mitigate these imperfections [8].

Single Qubit Gates

A single qubit gate U transforms an arbitrary initial state $|\psi\rangle$ into a final state $|\psi'\rangle = U|\psi\rangle$, where U can be represented by a 2×2 unitary matrix, making the operation a simple matrix-vector multiplication. The simplest gate is the so-called Pauli X gate, which is the

[8]This also means that in our first experiment, in which only the ground state is measured, we can get a 1 as a result, contrary to all expectations. To be precise, this occurred 17 times out of 1024 in our case. This demonstrates how error-prone quantum hardware still is at the moment.

equivalent of the classical NOT gate. It can be represented by the permutation matrix

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

and simply flips the initial state, e.g.

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle$$

The Quantum Composer application of a Pauli X gate is presented in Figure 5b. For completeness, we also show Pauli Y and Pauli Z gates. The three Pauli gates are quite useful and are very commonly used single qubits gates.

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

The Hadamard gate H is fundamental in quantum computing. Often referred to as the superposition gate, it takes a basis state and transforms it into an equal superposition of both basis states. The Hadamard gate is defined as follows:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

and operates as

$$H|0\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$

$$H|1\rangle = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$$

In Figure 5c, we can see the behavior of the Hadamard gate on a qubit in $|0\rangle$, putting it in a superposition of state $|0\rangle$ and $|1\rangle$. Again, we can see that due to external noise on the hardware, we don't get a perfect 50:50 outcome. Figure 5d shows that the Hadamard gate produces the same result regardless of whether we start from state $|0\rangle$ or $|1\rangle$. At first glance, the Hadamard gate and thus the regular superposition may appear to work just like a classical coin flip; however, Figure 5e shows that quantum randomness is different from classical randomness. Applying a second Hadamard gate to the qubit reverses the effect of the first one and returns the qubit to its initial state. This behavior contradicts the intuition that a second coin flip would still produce an equally distributed result of 0 and 1 and gives reason to why our quantum gates need to be unitary. It also leads to another important property of qubits: Interference.

Interference is a key concept in quantum mechanics, and to introduce it, it is worth leaving the representation of the Bloch sphere. Qubits are an abstraction of a two-dimensional quantum system, and there is no correct representation that can always transmit a full understanding of all properties. Qubits are often described as wave-like systems because of their ability to exist in multiple states simultaneously. The wave can exist in different states at different points along the string, and its amplitude and frequency can change depending on the conditions. Similarly, a qubit can exist in different superposition states, and its probability of being in a particular state can change depending on the operations performed on it. When two waves interact, their amplitudes can either add up (constructive interference) or cancel each other out (destructive interference), depending on the

phases of the waves. We can also manipulate the phase of a qubit with a P-gate, shifting the phase of the qubit by $\phi \in \mathbb{R}$, that is,

$$P(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{j\phi} \end{pmatrix}.$$

However, the qubit must first be in superposition for the P-gate to have any effect. This is because the P-gate acts on the relative phase between the two basis states, and this phase is only defined when the qubit is in a superposition. The global phase of a qubit cannot be measured, which means that we can add a phase factor to the state without affecting the outcome of any measurements. This is because the global phase only affects the overall probability of measuring the qubit in either state and not the relative probabilities. This property allows us to write a qubit state as $|\psi\rangle = e^{j\varphi}|\psi\rangle$, where j is the imaginary unit, $0 \leq \varphi \leq 2\pi$, and the state $|\psi\rangle$ is a linear combination of the basis states.

Multiple Qubit Gates

Just as in classical computing, in quantum computing it is essential to work on multiple computing units simultaneously to solve more complex tasks. In multiple qubit systems, the vector notation for qubit states is defined using the tensor product (also known as the Kronecker product) of the individual qubit states. For a 2-qubit system, the basis states are formed by taking the tensor product of the basis states of the individual qubits. Here is a general definition for the states of a 2-qubit system: Given two qubits φ and ψ with states $|\varphi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ and $|\psi\rangle = \begin{pmatrix} \gamma \\ \delta \end{pmatrix}$, their combined state in a 2-qubit system can be represented as

$$|\varphi\psi\rangle = |\varphi\rangle \otimes |\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \otimes \begin{pmatrix} \gamma \\ \delta \end{pmatrix} = \begin{pmatrix} \alpha \begin{pmatrix} \gamma \\ \delta \end{pmatrix} \\ \beta \begin{pmatrix} \gamma \\ \delta \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \alpha\gamma \\ \alpha\delta \\ \beta\gamma \\ \beta\delta \end{pmatrix}$$

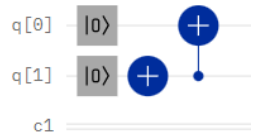
This notation allows us to define the basis states for a 2-qubit system as follows:

$$|00\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad |01\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \quad |10\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad |11\rangle = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

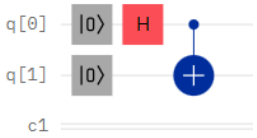
Naturally, this notation can also be extended to larger qubit systems. A fundamental 2-qubit gate is the controlled NOT gate (CNOT). Basically, it operates on two qubits, a control qubit and a target qubit, and applies a NOT (or X) operation on the target if the control qubit is in state $|1\rangle$ but leaves it unchanged otherwise. This extension of the NOT gate can be further extended to more control qubits. In the case of only one control qubit, we can write its matrix representation as:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Any multiple-qubit gate can be decomposed into a sequence of single-qubit gates and CNOT gates (the universality of the CNOT gate). Hence, with the application of a CNOT gate, one can create entanglement between the control and the target qubit. Entanglement is another key property of quantum mechanics and is a type of correlation between qubits where the state of one qubit depends on the others' state. When two or



(A) CNOT$|01\rangle = |11\rangle$



(B) Bell circuit

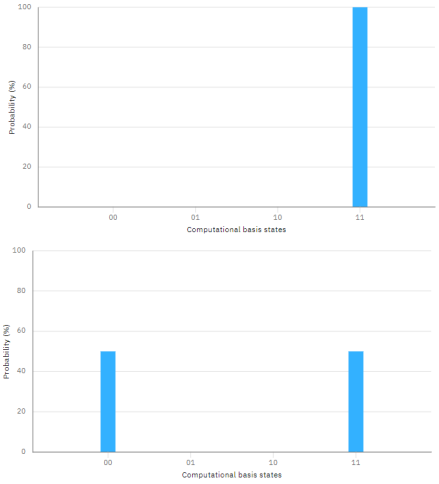


FIGURE 6. On the left we have the circuit implementation of the gate and on the right the respective outcome. All images are retrieved directly from the IBM Quantum Composer **IBM_Q**.

more qubits are entangled, the state of each qubit cannot be described independently anymore. By exploiting this property, a quantum computer can solve tasks faster than classical computers. The CNOT gate is just one of many controlled gates that can create entanglement, it is however the one most commonly used. The state evolution of the pairs of basis states applied by the CNOT is straightforward.

$$|00\rangle \to |00\rangle, \quad |01\rangle \to |01\rangle, \quad |10\rangle \to |11\rangle, \quad |11\rangle \to |10\rangle,$$

where the third is shown in 6a and holds since,

$$\text{CNOT}|10\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

A case worth highlighting is when setting the control qubit into superposition right before the CNOT gate. The result is a maximally entangled state known as a Bell state, which is important in various areas of quantum information processing.Even though the Hadamard gate is only applied to one qubit, both qubits are in superposition in the Bell state.

Up to this point, we have discussed numerous fixed gates. However, gates can also be parameterized. The most notable parameterized gates are the Pauli rotations, which generalize the Pauli gates. These rotations are continuous transformations around the respective axes in the Bloch sphere and serve as crucial building blocks for constructing quantum circuits. The Pauli rotations are given by:

$$R_X(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) X = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$R_Y(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Y = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$$

$$R_Z(\theta) = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Z = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}$$

Here, I, X, Y, and Z are the Pauli gates, and θ is the rotation angle. Note that $R_Z(\theta)$ has a slightly different form than the other two because Z is diagonal, so the exponentiation of Z just changes the phase of the state. Furthermore, for $\theta = \pi$ the rotations equal their respective Pauli gate. While an $R_Z(\theta)$ rotation only changes the phase of a qubit, the $R_X(\theta)$ and $R_Y(\theta)$ rotations change both, phase and amplitude of the qubit. These parameterised gates allow a high degree of flexibility in the design of quantum circuits, as they enable the implementation of arbitrary one-qubit rotations. The combination with other gates, such as the CNOT gate, opens up the possibility of constructing more complex quantum circuits for a wide range of applications, including quantum algorithms and error correction methods. The ability to perform arbitrary rotations around the different axes of the Bloch sphere is essential for the optimisation of quantum circuits and the precise control of qubits in quantum computing.

2.1.2. Quantum Algorithms Used in this Work: Grover's Algorithm

Quantum algorithms can be broadly classified into two categories. The first category is based on Shor's quantum Fourier transform and is primarily used for factoring, which has the potential to cause a dramatic impact on encryption. The second category is based on Grover's algorithm and has shown significant progress in searching unstructured databases. In the following sections, we provide a detailed discussion of the algorithms that will be used for the option pricing portion.

Grover's algorithm (named after Lov Grover) promises a quadratic speed-up compared to classical search algorithms. Superposition and interference play a major role in this algorithm: Suppose we have a database with N entries and want to find a certain entry in it. Previous classical methods need on average $\frac{N}{2}$ and in the worst-case N checks for such a problem. The Grover algorithm only needs $\mathcal{O}(\sqrt{N})$ queries. Essentially, the algorithm consists of an initialisation and the "Grovers iteration". During initialisation, we simply apply a superposition to all possible states of the database. The right graphic of Figure **??** gives a bar graph of the amplitudes.

$$|s\rangle = |\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \tag{6}$$

with $|i\rangle$ being the i-th element of the database. In the Grover iteration, two iterative transorfmations are applied to the qubits, the "oracle" and the "diffusion operator". The structure of the oracle is always problem-specific, because one needs it in the construction in a way such that the marked element can be identified in the database. It should reflect the phase of the object to be sought and is defined as follows:

$$U_\omega |i\rangle = \begin{cases} -|i\rangle & \text{if } i = \omega \\ |i\rangle & \text{otherwise} \end{cases} \tag{7}$$

where ω is the index of the target object. The application of the oracle can be seen in the bar plot of Figure **??**. The diffusion operator D amplifies the amplitude of the marked state and suppresses the amplitudes of the other states. It is defined as follows:

$$D = 2 |\psi_0\rangle \langle\psi_0| - I \tag{8}$$

where I is the identity operator and thus affects the amplitude as in **??**. Now the oracle and diffusion operator is iterated k times to amplify the amplitude of the marked state, where k is chosen such that $k = \mathcal{O}(\sqrt{N})$. After k iterations we get

$$|\psi_k\rangle = U_\omega D^k |\psi_0\rangle \tag{9}$$

and the probability of measuring the marked state is at maximum. We can use a measurement to find the index of the target object with high probability.

2.1.3. Quantum Fourier Transform (QFT)

The Quantum Fourier Transform (QFT) is a key subroutine for many quantum algorithms, including Shor's algorithm for integer factorization and various quantum phase estimation algorithms. It is essentially the quantum counterpart of the classical discrete Fourier transform. The classical fourier transformation converts a complex vector with N elements into another complex vector by decomposing it into a sum of trigonometric functions with different frequencies. Similarly, the quantum Fourier transform works with a quantum state with N basis states and transforms it into another quantum state by decomposing it into a sum of orthogonal basis states with different phases. Given an input state $|X\rangle = \sum_{j=0}^{N-1} x_j |j\rangle$, the QFT maps this state to the output state $|Y\rangle = \sum_{j=0}^{N-1} y_k |k\rangle$ as follows:

$$|y_k\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j \omega_N^{jk} \tag{10}$$

with $\omega_N^{jk} e^{\frac{2\pi i j k}{N}}$. Here, y_k represents the amplitude of the k-th basis state in the output state $|Y\rangle$. The transformation can also be represented as:

$$|Y\rangle = QFT(|X\rangle) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \sum_{j=0}^{N-1} x_j \omega_N^{jk} |k\rangle \tag{11}$$

QFT can be efficiently implemented on a quantum computer using a combination of Hadamard gates and controlled phase rotation gates. The quantum circuit for the QFT consists of several layers, each of which contains a Hadamard gate followed by a series of controlled phase rotation gates with decreasing rotation angles. The final step in the QFT circuit is a reverse of the order of the qubits.

A commonly used version of the QFT is its inverse. This operation, denoted as QFT^{-1} transforms a quantum state from the frequency domain back to the original domain. It is implemented using a quantum circuit similar to the QFT circuit, but with controlled phase rotation gates employing negative rotation angles. The inverse QFT is a crucial component in several quantum algorithms, including quantum phase estimation, where it assists in retrieving the estimated phase of an eigenvector after applying controlled unitary operations. By applying the inverse QFT, the quantum state is transformed back into a form that enables the extraction of relevant information with high probability upon measurement.

2.1.4. Quantum Algorithms Used in this Work: Quantum Phase Estimation

We can now introduce the Quantum Phase Estimation (QPE) algorithm that builds on the QFT and was used for the option pricing procedure by **Rebentrost_2018**. In general, QPE is a very important building block for many promising algorithms and serves as a major reason why many algorithms may achieve quantum advantage. We have pointed out that in theory it would be possible to save large amounts of information in a single qubit due to their property that their state can be in any continuous direction, however to retrieve this information one would have to measure the qubit infinite times. QPE addresses this problem by estimating the aforementioned information, thus making it accessible to some degree without measuring infinite times. Consider a unitary operator U with eigenvector $|\psi\rangle$ and eigenvalue $e^{2\pi i\theta}$, where $\theta \in \mathbb{R}$. The QPE algorithm provides an estimate of θ in binary representation with high probability. The algorithm consists

of two registers, a control register of n qubits, initialised in the state $|0\rangle^{\otimes n}$ and a target register of m qubits, initialised in the state $|\psi\rangle$. The algorithm uses the controlled-U gate defined as follows: $CU^{2^j}|\psi\rangle|\varphi\rangle = |\psi\rangle U^{2^j}|\varphi\rangle$ where $|\varphi\rangle$ is an arbitrary state and j is the j-th qubit of the control register. The algorithm can be broken down into the following steps:

(1) Apply a Hadamard gate to each qubit in the control register.

$$|0\rangle^{\otimes n}|\psi\rangle \rightarrow \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle|\psi\rangle \tag{12}$$

(2) Apply the controlled-U^{2^j} gate j times to the target register, with $j = \{0, \ldots, n-1\}$

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle|\psi\rangle \rightarrow \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle U^{2^k}|\psi\rangle \tag{13}$$

(3) Apply the inverse Quantum Fourier Transform $(\text{QFT})^{-1}$ to the control register.

$$\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle U^{2^k}|\psi\rangle \rightarrow \frac{1}{2^n}\sum_{k=0}^{2^n-1}\sum_{j=0}^{2^n-1}e^{2\pi ijk/2^n}|j\rangle U^{2^k}|\psi\rangle \tag{14}$$

(4) Measure the control register to obtain an estimation of θ in binary representation with high probability.

The probability of obtaining an estimation $\tilde{\theta}$ that is ϵ-close to the true θ in binary representation is given by: $P(|\tilde{\theta} - \theta| \leq \epsilon) \geq 1 - \frac{4}{\pi^2}\frac{1}{\epsilon^2}$. We can use QPE to estimate the eigenvalues of a unitary matrix U. First, we need to decompose U as a sum of powers of two unitary matrices, such that $U = \sum_{j=0}^{2^n-1} e^{2\pi i\theta_j}|j\rangle\langle j|$, where n is the number of qubits in the quantum register, $|j\rangle$ is a computational basis state, and θ_j is the eigenphase corresponding to the eigenvector $|j\rangle$. To perform QPE, we prepare a state $|\psi\rangle$, which is a tensor product of two quantum registers: a control register of n qubits and a target register of m qubits. The control register is initialized to $|0\rangle^{\otimes n}$, and the target register is initialized to an eigenvector $|u\rangle$ of U corresponding to the eigenvalue $e^{2\pi i\theta_j}$. The state $|\psi\rangle$ is then given by $|\psi\rangle = \frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1}|k\rangle|u\rangle$. Next, we apply a sequence of controlled-U^{2^j} gates to the state $|\psi\rangle$, where the j-th gate acts on the j-th qubit of the control register. This results in the state $\frac{1}{\sqrt{2^n}}\sum_{k=0}^{2^n-1} e^{2\pi i\theta_k}|k\rangle|u\rangle$. Finally, we perform an inverse quantum Fourier transform (QFT) on the control register, which gives us an estimate of the eigenphase θ_j with high probability. The number of qubits required for the control register depends on the desired precision of the estimation, with a trade-off between the precision and the circuit depth.

One drawback of QPE is that it is particularly prone to error and noise on NISQ devices. QPE requires deep quantum circuits consisting of a large number of gates. This increases the probability of errors, and these errors can accumulate, leading to incorrect results. In addition, NISQ devices have a limited coherence time, which means that qubits lose their quantum properties over time due to interactions with the environment. QPE relies on performing longer and coherent operations accurately and thus the accuracy of the algorithm can be negatively impacted.

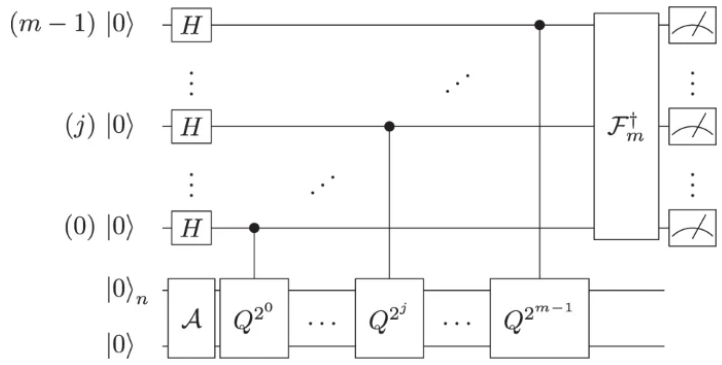


FIGURE 7. The canonical QAE with m ancialla qubits and n+1 state qubits. Source: **grinko2021iterative**

2.1.5. Quantum Algorithms Used in this Work: Quantum Amplitude Estimation
Quantum Amplitude Estimation (QAE) was developed by Brassard et al. **Brassard_2002** and is a fundamental quantum algorithm with the potential to achieve a quadratic speedup over classical Monte Carlo (MC) simulations for various applications. This algorithm is a combination of QPE and Grover's Algorithm, but instead of estimating the phase of an eigenvalue as QPE does, QAE is designed to estimate the amplitude of a specific state in a quantum superposition with high probability. Compared to QPE, QAE may be better suited for NISQ devices because it can be implemented with fewer quantum gates, resulting in circuits of lower depth. This smaller number of gates can lead to a lower error rate, making the algorithm more resistant to noise and other sources of errors. However, QAE still faces similar challenges on NISQ devices and is dependent on error mitigation techniques. Suppose we have a quantum channel $\mathcal{A}$ that can be an arbitrary operation on n qubits and one additional (auxiliary) qubit. That has the following mapping for ground state $|0\rangle^{\otimes n}|0\rangle$:

$$\mathcal{A}|0\rangle_n|0\rangle = \sqrt{1-a}\,|\psi_0\rangle_n|0\rangle + \sqrt{a}\,|\psi_1\rangle_n|1\rangle, \tag{15}$$

where $a \in [0,1]$ is the unknown amplitude we are interested in. And $|\psi_0\rangle_n$ and $|\psi_1\rangle_n$ are two normalized states, not necessarily orthogonal. QAE allows us to estimate a with high probability such that the estimation error scales as $\mathcal{O}(1/M)$, where M corresponds to the number of applications of $\mathcal{A}$ **Brassard_2002 grinko2021iterative**. To achieve this, a Grover type operator $\mathcal{Q} = \mathcal{A}\mathcal{S}_0\mathcal{A}^\dagger\mathcal{S}_{\psi_0}$ is defined. To point out the similarities to the Grover Algorithm, the diffusion operator D in (8) is now given by the two reflection operators $\mathcal{S}_0$ (reflecting around the initial state) and $\mathcal{S}_{\psi_0}$ (reflecting around the state $|0\rangle$), with $\mathcal{S}_{\psi_0} = \mathbb{I} - 2\,|\psi_0\rangle_n\langle\psi_0|_n \otimes |\,0\rangle\langle 0|$, and $\mathcal{S}_0 = \mathbb{I} - 2|0\rangle_{n+1}\langle 0|_{n+1}$ Furthermore, the Grover oracle U_ω in 7 is now given by the operator $\mathcal{A}$.

The QAE algorithm follows a structure similar to the Quantum Phase Estimation (QPE) algorithm. It uses m ancilla qubits, initialized in an equal superposition state, to represent the final result. The number of quantum samples is defined as $M = 2^m$, and the QAE operator $\mathcal{Q}$ is applied in a controlled manner with geometrically increasing powers, steered by the ancilla qubits. After this, an inverse Quantum Fourier Transform (QFT) is performed on the ancilla qubits before they are measured. The whole structure of the QAE is given in Figure 7.

The measured integer $y \in 0, \ldots, M-1$ is mapped to an angle $\theta_a = y\pi/M$. The resulting estimate of a is defined as $\tilde{a} = \sin^2(\theta_a)$. The error of the estimate can be bounded with a probability of at least $8/\pi^2 \approx 81\%$ to satisfy

$$|a - \tilde{a}| \leq \frac{2\pi\sqrt{a(1-a)}}{M} + \frac{\pi^2}{M^2}, \tag{16}$$

which shows the quadratic speedup[9] over classical Monte Carlo simulation **grinko2021iterative**. By repeating the experiment multiple times the probability of success quickly rises very close to 100%, however, the estimates $\tilde{a}$ are naturally restricted to the discrete grid $\left\{\sin^2(y\pi/M) : y = 0, \ldots, M/2\right\}$ by the possible outcomes of y. The number of grid points, on the other hand, increases exponentially with the number of ancilla qubits used.

2.1.6. Iterative Quantum Amplitude Estimation

To address the concerns regarding the QPE-dependency and its impact on the resource requirements, a modified version of the Quantum Amplitude Estimation algorithm has been proposed by **grinko2021iterative**, called Iterative Quantum Amplitude Estimation (IQAE) This approach aims to simplify QAE by eliminating the need for QPE and relying only on Grover iterations, thereby reducing the number of qubits and circuit depth required for practical applications and thus improving overall stability on NISQ devices. IQAE focuses on efficiently estimating the probability of obtaining a specific outcome from a quantum circuit by approximating the probability of the last qubit being in state $|1\rangle$ for different powers of the operator $\mathcal{Q}$. It achieves this by combining the ideas of previous works, optimizing efficiency, and maintaining rigor in error estimation and computational complexity. The algorithm operates by iteratively narrowing a confidence interval for the parameter θ_a. The key steps in the algorithm are as follows:

(1) Given a confidence interval $[\theta_1, \theta_u]$ for θ_a and a power k of operator $\mathcal{Q}$, the algorithm approximates $\sin^2((2k+1)\theta_a)$.

(2) Using the FINDNEXTK subroutine, the largest k is found such that the scaled interval $[(4k+2)\theta_1, (4k+2)\theta_u] \mod 2\pi$ is fully contained either in $[0, \pi]$ or $[\pi, 2\pi]$.

(3) The algorithm improves the estimate for θ_a and converges with a quadratic speedup.

The core of the algorithm is the FINDNEXTK procedure, which optimizes Fisher Information on a given iteration in a greedy manner. The FINDNEXTK procedure ensures a quadratic speedup compared to classical techniques.

2.2. Quantum Neural Networks and the Setup of Our (Hybrid) qWGAN

Merging the latter two sections on GANs and quantum computing leads directly to the heart topic of generative quantum machine learning, with our goal being to learn a random distribution underlying a predefined training data set by employing quantum techniques. In this section, we introduce the concept of generative quantum machine learning using the example of Quantum Generative Adversarial Networks (qGANs), a proposal from quantum machine learning researchers inspired by the inherent probabilistic nature of quantum systems. Their backbone are Quantum Neural Networks. The main reference and guidline in notation in this chapter are the books "Machine Learning with Quantum

[9]While other QPE-based algorithms, such as Shor's Algorithm for factoring, achieve exponential speedup, it has been speculated whether QAE can be simplified by only using Grover iterations without a QPE-dependency. Removing the QPE-dependency would help to reduce the resource requirements of QAE in terms of qubits and circuit depth, making it more accessible for practical applications.

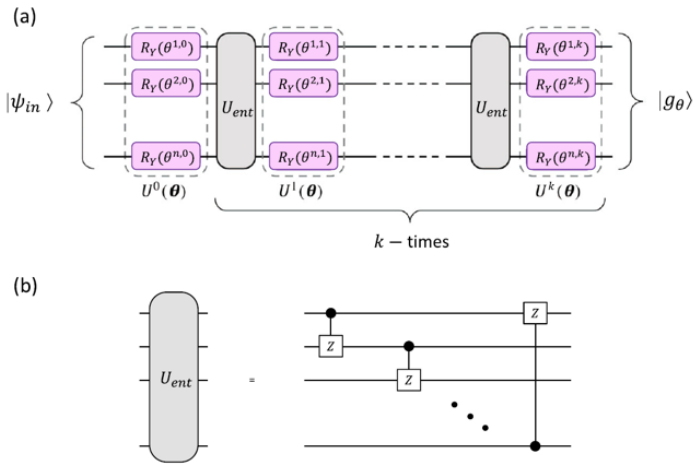


FIGURE 8. A General Version of a Variational Form (Quantum Neural Network) with depth k acting on n qubits. Source: **Zoufal_2019**

Computers" **schuld2021machine** and "Quantum Computation and Quantum Information: 10th Anniversary Edition" **nielsen_chuang_2010**.

2.2.1. Quantum Neural Networks (QNN) or Variational Quantum Models

Quantum Neural Networks (QNNs) are also referred to as *Variational Quantum Models*. They represent a class of parametrized quantum circuits specifically designed for modeling and processing data, serving as quantum analogues to classical neural networks. The term "quantum neural network" can be somewhat misleading as the structure of variational circuits does not naturally exhibit the multi-layer perceptron structure commonly found in classical neural networks. Nonetheless, QNNs, like their classical counterparts, are structured in layers and consist of basic computational units. These circuits feature layers of adjustable parameters that can be optimized during the training process, similar to how weights and biases are adjusted in classical neural networks.

Figure 8 illustrates a variational form, which is a parametrized quantum circuit composed of alternating layers of single-qubit Pauli-Y rotations and entangling blocks U_{ent} of two-qubit controlled-Z gates. The depth of the variational circuit, denoted by k, determines the number of alternating repetitions, with a total of $(k+1)n$ parametrized single-qubit gates and kn two-qubit gates acting on n qubits. The parametrized R_Y rotations in QNNs should not be considered as direct analogues to perceptrons or nodes in classical neural networks. Instead, being unitary matrix operations, they resemble the affine linear transformations found in classical networks. This n-qubit QNN can be writen as the mapping

$$\begin{aligned} |g_\theta\rangle &= G_\theta\,|\varphi_{in}\rangle \\ &= |\varphi_{out}\rangle \\ &= \prod_{p=1}^{k}\left(\bigotimes_{q=1}^{n}\left(R_Y(\theta^{q,p})U_{ent}\right)\right)\bigotimes_{q=1}^{n}\left(R_Y(\theta^{q,0})\right)|\varphi_{in}\rangle \\ &= \sum_{j=0}^{2^n-1}\sqrt{p_\theta^j}\,|j\rangle \end{aligned} \tag{17}$$

Variational Quantum Models as Functional Approximators: In the context of quantum machine learning, the expressivity of quantum models is crucial for their ability to approximate functions. Similar to classical neural networks, where (some variant of) the *Universal Approximation Theorem* can guarantee that networks can approximate functionals and distributions on a given space **Horvath_2021**, quantum models also exhibit the potential to serve as universal function approximators. However, current research has not yet established universal approximation theorems for quantum models as precisely as in the classical cases. In **schuld2021machine**, it is demonstrated that circuits must be sufficiently wide and deep to achieve universality, and the target function must possess an integer-valued frequency spectrum. The requirement of an integer-valued frequency spectrum means that the Fourier series representation of the target function must have coefficients corresponding to integer multiples of some fundamental frequency. The study of the expressivity of QNNs is still an active area of research, with ongoing work aiming to establish the foundation for the expressive power of quantum neural networks **Wu_2021**.

Potential Challenges in Training QNNs with Increasing Number of Qubits: Depending on the structure, classical optimization techniques can be utilized to train QNNs by adjusting the parameters in the variational circuit to minimize a cost function. In other cases, it is necessary to incorporate quantum gradient-based optimization techniques. For both scenarios, the optimization process necessitates calculating gradients of the cost function with respect to the circuit parameters. Automatic differentiation, a technique borrowed from deep learning, is commonly employed to compute gradients efficiently. This approach allows for more efficient use of near-term quantum computing resources while leveraging classical co-processors for training. In the context of training QNNs, gradients are calculated using techniques such as the parameter-shift rule, which involves applying the chain rule of differentiation to propagate the error from the output layer through the variational form to the input layer, as in (17). Gradient-based optimization of QNNs can face a significant challenge called the "Barren plateau" problem **mcclean2018barren**. This refers to the observation that the gradient of a QNN becomes exponentially small as the number of qubits in the circuit grows, which significantly hinders or even prevents successful optimization. This issue is primarily caused by the prevalence of vanishing gradients in circuits containing a large number of qubits.

2.2.2. Setup and Training of the Hybrid Quantum GAN

Here we introduce a hybrid qGAN, where the generator is a QNN and the discriminator is a classical neural network, with the generated data being quantum states, as shown in 9. Similar to classical GANs, the objective of qGANs is to train a generator to produce samples that resemble the specific distribution of a training set p_{real}. Meanwhile, the discriminator aims to distinguish between generated samples and real data. The training process alternates between updating the generator and discriminator parameters using gradients of the loss function with respect to the parameters. This hybrid model is particularly suitable for loading classical distribution into quantum states. It is important to note that a neural network, be it quantum or classical, can only process data in the same state as the network itself. Consequently, the discriminator must be a classical neural network. Although other techniques exist for transforming data into quantum states **Grover_2000 Grover_2002**, which could potentially allow for classical data and a quantum discriminator, these approaches remain costly, requiring $\mathcal{O}(2^n)$ rotation gates to load a distribution into n qubits. Moreover, this constraint limits distributions to those that are integrable. Therefore, for our use case, the discriminator must be a classical neural network. The work by **Zoufal_2019** has demonstrated that such a hybrid form of

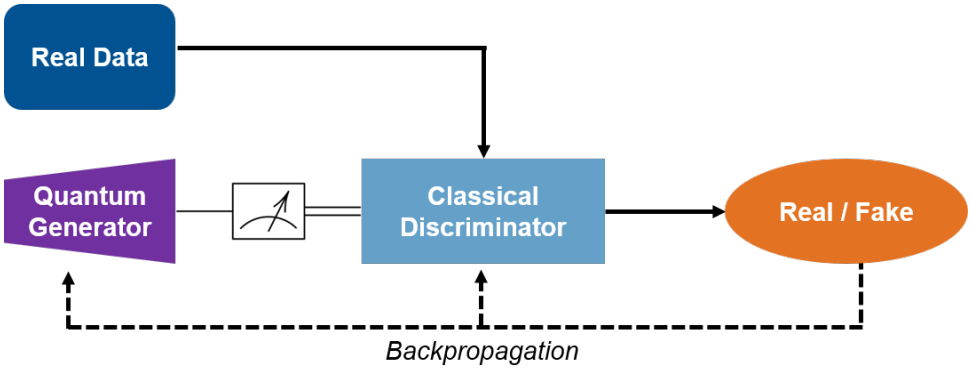


FIGURE 9. Hybrid QGAN with Quantum generator and classical discriminator

qGANs can reduce the complexity of loading data into quantum states to $\mathcal{O}(poly(n))$. With a quantum generator producing quantum output, the two networks essentially speak different languages, necessitating the translation of quantum output into classical data through measurement. A single measurement consists of m shots, which by default is often set to 1024 shots, providing a batch of classical data that can be mapped to the same grid as the real data and subsequently classified by the discriminator.

Setup

We now discuss the construction of a hybrid qGAN to create a quantum generator that, after sufficient training, acts like a quantum loading channel to generate distributions in quantum states. In general, the procedures do not deviate significantly from the classical ones. We have a training set $X = x_0, x_1, \ldots, x_{m-1} \subseteq \mathbb{R}^{d_{out}}$ underlying a distribution p_{real} that is not necessarily known. For n qubits, we can represent a quantum state of 2^n discrete output states. Thus, it may be necessary to discretize the training set to a grid of 2^n values, such as $0, 1, \ldots, 2^n - 1$. The classical discriminator is denoted by $D^\phi : \mathbb{R}^{d_{out}} \to 0, 1$ with $\phi \in \mathbb{R}^{k_d}$ representing the network parameters. The architecture can be described by the hyperparameters, with l_d linear layers and $d^{(j)}$ the number of nodes in layer j. Each hidden layer is equipped with a nonlinear activation function. The output layer l_{out} has only one node and processes its input with a sigmoid activation function. This assignment to a value between 0 and 1 can be interpreted as the probability with which the input received is classified as real or fake (generated). The input layer has 2^n nodes, as the input will always be a generated distribution or a batch of the discretized training set. The quantum generator G is a variational form transforming a given n-qubit input state $|\varphi_{in}\rangle$ to an n-qubit output state

$$G|\varphi_{in}\rangle = |g^\omega\rangle = \sum_{i=0}^{2^n-1} \sqrt{p_i(\omega)}\,|i\rangle \tag{18}$$

due to the probabilistic nature of qubits, input and output state can be perfectly interpreted as a probability distribution function, with $p_i(\omega)$ being the probability of observing the basis state $|i\rangle$ and $\omega \in \mathbb{R}^{k_g}$ being its parameters. For l_g being the number of layers in the variational form this results in $k_g = (l_g + 1)n$ R_y-rotations and thus parameters. Furthermore, we have k_g entanglement blocks, with each having n or $n-1$ controlled-gates, depending on the type of entanglement (linear, circular, full, shifted-circular-alternating (sca), ...). We assume the domain of X consists of values from $0, 1, \ldots, 2^n - 1$, enabling us to map the output of the generator to the sample space of the training data, other mappings suiting the underlying problem better may be easily implemented. Thus, the

(A) Quantum Generator Implementation on 5 Qubits, 1 layer and full entanglement

(B) Quantum Generator Implementation on 5 Qubits, 2 layers and shifted-circular-alternating (sca) entanglement

(C) Quantum Generator Implementation on 5 Qubits, 3 layers and linear entanglement

FIGURE 10. Quantum Generator Implementations on 5 Qubits

number of qubits directly affects the resolution of the generated output, with 2^n discrete values being represented. Figure 10 shows three concrete examples of a generator. All having $n = 5$ qubits and $l_g = 1, 2, 3$ layers, consequently also $k_g = 5, 10, 15$ parameterised R_y rotations and different entanglement types with, in these cases cx-gates. In addition, one can clearly see here that the $|\varphi_{in}\rangle$ chosen is the uniform superposition, which is particularly easily generated by a Hadamard gate on each qubit. Depending on the training process requirements, other $|\varphi_{in}\rangle$ options, such as normal or multimodal distributions, can be advantageous, albeit more complex to generate.

Training

The fully trained quantum generator can be easily incorporated into another quantum algorithm by inserting the circuit in the appropriate place. However, for the training process, the discriminator and generator effectively "speak" different languages. To bridge the gap between the quantum state output of the generator and the classical state required by the discriminator, it is necessary to measure $|g^\omega\rangle$.

When measuring the generator output m times or with m shots, a second data set $\mathcal{G}$ is obtained, where the computational basis is $|i\rangle, i \in 0, \ldots, 2^n - 1$. It is important to differentiate between classical Monte Carlo sampling and the sampling obtained here because no additional stochastic input is needed—the sampling is based solely on the inherent stochasticity of quantum measurements.

For higher training efficiency, we specify a batch size b, with $m \mod b$, so that we run through $\frac{m}{b}$ batch iterations for one epoch in the training process. The data samples $g_j \in \mathcal{G}$ with $g_j \sim p_j(\omega)$ and b randomly chosen $x_j \in \mathcal{X}$ for $j \in \{1, \ldots, b\}$ of one batch are to be used for each batch iteration. The generator and the discriminator are trained alternately according to their respective lossfunctions $L_G(\phi, \omega)$ and $L_D(\phi, \omega)$ and some analytic quantum gradient technique.

The loss functions in **Zoufal_2019** for the generator and discriminator are defined as follows:

$$L_G(\phi, \omega) = -\sum_{j=1}^{b} p_j(\omega) \log\left(D^\phi(g_j)\right) \tag{19}$$

$$L_D(\phi, \omega) = \sum_{j=1}^{b} \frac{1}{m} \log\left(D^\phi(x_j)\right) + p_j(\omega) \log\left(1 - D^\phi(g_j)\right). \tag{20}$$

Their corresponding gradients are:

$$\nabla_\omega L_G(\phi,\omega) = -\sum_{j=1}^{b} \nabla_\omega p_j(\omega) \log\left(D^\phi(g_j)\right) \tag{21}$$

$$\nabla_\phi L_D(\phi,\omega) = \sum_{j=1}^{b} \nabla_\phi \left[\frac{1}{m}\log\left(D^\phi(x_j)\right) + p_j(\omega)\log\left(1-D^\phi(g_j)\right)\right]. \tag{22}$$

The optimization can then be performed using a stochastic gradient descent method, such as AMSGRAD or ADAM optimization. A learning rate η is defined as the step size for each optimization step. Following **lehtokangas1998weight**, it is helpful to break symmetries in the generator's parameters by randomly choosing small values close to 0 as initial parameters. This approach should improve optimization in general and lead to better results. To achieve this, we sample k_g uniformly from $[-0.1, 0.1]$ and input these values into the generator before the first training. In addition to efficiency, qGANs have another practical advantage in that they have the possibility of incremental learning or online learning. This means that if there are small changes in the training data set, it is not necessary to start the entire training process from scratch. Instead, the previously trained parameters can be used as starting points, and optimization can be performed from there on. This approach can save a significant amount of time, and in sectors like finance, it allows for rapid incorporation of small changes in market conditions.

2.2.3. Extension to Our Hybrid Quantum Wasserstein GAN

The Quantum Wasserstein GAN (qWGAN) is a variation of the qGAN model that employs the Wasserstein distance as a metric to compare probability distributions during the optimization process. We remain in the hybrid setting of the generator being quantum variational form and the discriminator a classical NN. The Wasserstein loss, also known as the Earth-Mover (EM) distance, measures the distance between two probability distributions. The EM distance is a part of the optimal transport distances family and is defined as follows:

Definition 2.2 (Earth Movers distance or Wasserstein-1)**.** Given two probability measures μ and ν on a compact metric space (Ω, d), the Earth Movers distance, or Wasserstein-1 distance, is defined as:

$$W_1(\mu,\nu) = \inf_{\gamma\in\Gamma(\mu,\nu)} \int_{\Omega\times\Omega} d(x,y)d\gamma(x,y) \tag{23}$$

where $\Gamma(\mu,\nu)$ is the set of all joint probability measures on $\Omega\times\Omega$ with marginals μ and ν on the first and second factors, respectively.

A computationally more efficient representation is given with the Kantorovich-Rubenstein duality[10]. In the context of QGANs, we have a discriminator D^ϕ, the distribution of the

[10]Reminder of the Kantorovich-Rubenstein duality statement

Problem 2 (Kantorovich-Rubenstein duality)**.** For probability measures μ and ν on a compact metric space (Ω, d), the Wasserstein-1 distance can be represented as:

$$W_1(\mu,\nu) = \sup_{f\in \mathrm{L}_1(\Omega)} \mathbb{E}_{x\sim\mu}[f(x)] - \mathbb{E}_{y\sim\nu}[f(y)] \tag{24}$$

where $\mathrm{L}_1(\Omega)$ denotes the set of all 1-Lipschitz functions $f:\Omega\to\mathbb{R}$, i.e., functions that satisfy $|f|_L \le 1$ with $|f|_L = \sup_{x\neq y}\frac{|f(x)-f(y)|}{d(x,y)}$ being the Lipschitz norm of the function f.

For the proof, we refer to **Kantorovich_EDWARDS**.

real training data p_{real}, and the distribution of the output generated by the generator $p_{\text{fake}}(G^\omega)$. The general objective function can be formulated as follows:

$$L_{\text{QWGAN}}(\phi,\omega) = \mathbb{E}_{x\sim p_{\text{real}}}[D^\phi(x)] - \mathbb{E}_{x\sim p_{\text{fake}}(G^\omega)}[D^\phi(x)]. \tag{25}$$

The generator G^ω aims to minimize the objective function, while the discriminator D^ϕ aims to maximize it: $\min_\omega \max_\phi L_{\text{QWGAN}}(D^\phi, G^\omega)$. For a fixed D^ϕ, the generator's objective changes in the qWGAN case compared to the QGAN case as follows:

$$\begin{aligned} &\text{QGAN:} && \mathbb{E}_{x\sim p_{\text{fake}}(G^\omega)}[\ln(1 - D^\phi(x))] \\ &\text{QWGAN:} && \mathbb{E}_{x\sim p_{\text{fake}}(G^\omega)}[D^\phi(x)] \end{aligned} \tag{26}$$

The gradient of the qWGAN loss with respect to the discriminator's parameters ϕ is:

$$\nabla_\phi L_{\text{QWGAN}}(\phi,\omega) = \mathbb{E}_{x\sim p_{\text{real}}}[\nabla_\phi D^\phi(x)] - \mathbb{E}_{x\sim p_{\text{fake}}(G^\omega)}[\nabla_\phi D^\phi(x)]. \tag{27}$$

Similarly, the gradient of the loss with respect to the generator's parameters ω is:

$$\nabla_\omega L_{\text{QWGAN}}(\phi,\omega) = -\mathbb{E}_{x\sim p_{\text{fake}}(G^\omega)}[\nabla_\omega D^\phi(x)]. \tag{28}$$

These gradients can be used to update the parameters of the discriminator and generator using an optimizer such as ADAM. The discriminator is trained to maximize the Wasserstein distance between the distribution of the generated samples and the true distribution of the training data, while the generator is trained to minimize the same distance. This contrasts with the traditional GAN, where the discriminator is trained to maximize the probability of correctly classifying samples as real or fake, and the generator is trained to minimize the probability of the discriminator correctly classifying fake samples as fake. In qWGAN training, it is a common to modify the architecture of the discriminator, replacing the sigmoid output with a linear output layer. Additionally, during the training process, the discriminator is updated more frequently than the generator. This is because the Wasserstein distance provides a more meaningful and informative gradient for the discriminator, enabling it to better distinguish between real and fake samples. This also helps prevent the generator from exploiting weaknesses in the discriminator, which can lead to mode collapse and other issues.

3. Option Pricing with qGANs

In this section, we combine the concepts from the previous sections to a workflow implementing a concrete financial application on a quantum computer. The first task will be to price a European call option. The goal is to evaluate the expected payoff $E[\max\{S_T - K, 0\}]$, where S_T is assumed to follow a given random distribution. We demonstrate below, how qGANs can be used to estimate the expected payoff of a European call option by training a quantum generator to sample from the underlying probability distribution of S_T. A first proposition of pricing options via a Quantum Computer was made by Patrick Rebentrost, Brajesh Gupt and Thomas R. Bromley **Rebentrost_2018** in 2018. The authors built upon existing fundamentals provided by **Montanaro2015** to leverage Quantum Phase Estimation to achieve quadratic speed up for Monte Carlo Simulations. In their model they assume the Black-Scholes assumptions, which essentially means that the prices of a stock at maturity of an option follow a log-normal distribution and that its fluctuations over time can be modeled as a random walk with constant drift and volatility.

The pricing approach of **Rebentrost_2018** can be broken down into three steps:

(I) Load a (in **Rebentrost_2018** log-normal) distribution into quantum state.

(II) Compute the payoff function of the call option.

(III) Estimate the probabilities of the qubits.

Their procedure has a quadratic advantage in terms of speed compared to classical Monte Carlo methods. That is for a desired accuracy ϵ, the classical methods show a $1/\epsilon^2$ dependence in the number of samples, while the quantum algorithm shows a $1/\epsilon$ dependence. We also follow these three steps with appropriate modifications and improvements in each of the steps, to achieve optimal results.

3.1. Loading a Sample Distribution into Quantum State

In this section we demonstrate how we load a samole distribution into a quantum state on the example of a log-normal distribution. The log-normal distribution can represent the stock price of the underlying at maturity of the call option. It is not the most realistic distribution ine can opt for in this context, but it is a well-known standard choice and hence it is straighforward to verify outcomes to demonstrate a proof of concept. Bringing a given distribution into quantum state is thus essential for the algorithm and we follow the algorithm from **Grover_2002** for this, that requires $\mathcal{O}(2^n)$ gates. As already shown, it is precisely here that the application of qGANs can help to make a decisive advance. Therefore, in this step, unlike the authors, we will follow the procedure of **Zoufal_2019** and use the qGANs to create a lognormally distributed superposition of qubit states.

We want to price a European call option on a stock with reasonable parameters of an average return of 10% and a volatility of 25%. For a lognormal distribution it holds $E[X] = e^{\mu+\frac{\sigma^2}{2}}$, since we want to model an average return of 10% ($E[X] = 1.1$) and a volatility of 25% ($\sigma = 0.25$), we get for $\mu = \ln(1.1) - \frac{0.25^2}{2}$.

To achieve this, we will use five qubits, which allows us to create $2^5 = 32$ discrete initial states. Since $\text{Quantile}_{0.99999}(\text{Lognormal}(\mu, \sigma)) \approx 3.09$, we choose the output values $0.0, 0.1, ..., 3.1$ for the discrete states. This results in the canonical mapping from qubit output states to values such that $|00000\rangle \to 0.0$, $|00001\rangle \to 0.1$, ..., and $|11111\rangle \to 3.1$. While other values could be chosen, scaled and truncated, this choice is straightforward given the distribution and number of qubits used. Moreover, using more qubits would result in an exponentially finer grid, and the current most advanced quantum computer, IBM Osprey, has 433 qubits (as of March 2023) **Castelvecchi2023**. However, accessible [11] quantum computers like the ibm_nairobi and the ibm_oslo have seven qubits available and since we need two ancilla qubits for the option evaluation step, we are left with using only five qubits for this step. It is worth noting that the average queue time to execute a code on a quantum computer can be between 20 and 40 minutes. This and the gap between the IBM Osprey and publicly available quantum computers demonstrates that the access and usage of quantum computers are still in their early stages.
Training Set: To begin, we will generate a training dataset $\mathcal{X}$ with a sample size of $N = 10000$ by sampling from a lognormal distribution with parameters $\mu = \ln(1.1) - \frac{0.25^2}{2}$ and $\sigma = 0.25$, such that $x_i \sim \text{Lognormal}(\mu, \sigma)$ for all $x_i \in \mathcal{X}$, where $i \in 1, \ldots, N$. After rounding the training set to one decimal point, we obtain the training set shown in 11, with a maximum value of 3.1, which means that no scaling is necessary in this case. However, linear scaling could be easily applied if needed, which then has to be taken in the calculation of the final result.

Generator Setup:
We implement the quantum generator $\mathcal{G}$ using Qiskit **Qiskit** in Python. We aim to build

[11] Accessible here means that you still need an account at IBM, as well as access via a scientific institution such as the Technical University of Munich.

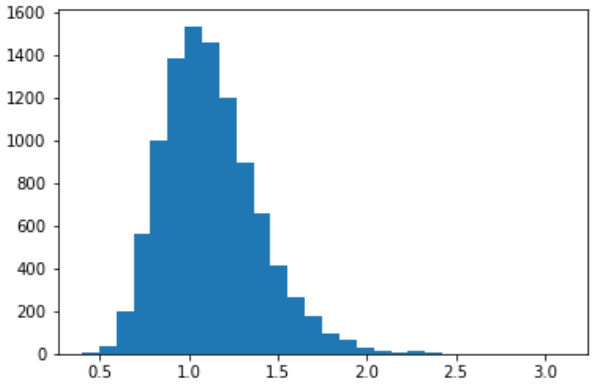


FIGURE 11. Trainingset with Lognormal distribution ($\mu = 1.1$, $\sigma = 0.25$), with $N = 10000$.

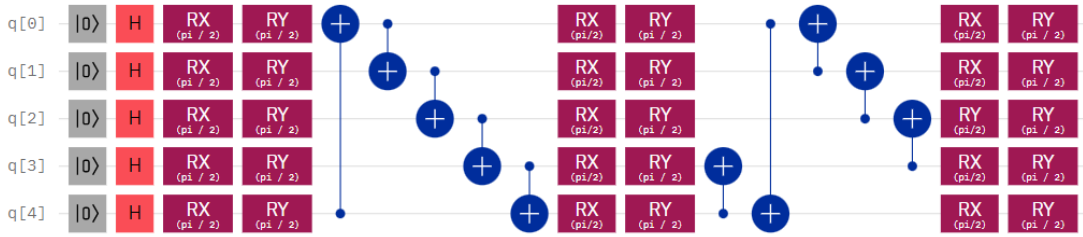


FIGURE 12. Quantum Generator Implementation on 5 Qubits with Two Layers

a variational form on five qubits, using a Two_Local circuit that corresponds to a parameterized circuit of several alternating rotation layers and entanglement layers. We specify a variational form with a depth of $l_{Generator} = 2$. It is first initialized with 5 Hadamard gates to obtain a uniform distribution of the qubit superposition. This is followed by a layer of rotation gates. Our initial experiments have shown that with simple layers of R_Y gates we can get already get good results, however, to further improve accuracy it was helpful to build a rotation layer through a layer of R_X and R_Y gates. R_X and R_Y gates both affect the amplitude of the qubits, while R_Z gates only manipulate the phase, at this point it is again recommended to have a look at the Bloch sphere representation 4. The values we now have to adjust for the qubits are the amplitudes, i.e. the z-axis intercepts in the Bloch sphere, since these are directly coupled with the probabilities according to Born's rule 2.1. Following the rotation layer, a block of entanglement gates is installed. For this issue we also empirically tested multiple possiblities and chose the best perfoming one, that is pairwise c_x entanglements arranged with the shifted-circular-alternating (sca). Sca simply is a circular entanglement where the 'long' entanglement connecting the first with the last qubit is shifted by one each block. The complete architecture is shown in 12.

As shown earlier we break similarities by sampling the 10 parameters k_g of the R_Y rotations of $\mathcal{G}$ uniformly from $[-0.1, 0.1]$ for initialisation.
Discriminator Setup: The classical discriminator is a 3-layer deep neural network implemented using PyTorch **PyTorch**. Let z_k denote the k-th grid point in the set $0.0, 0.1, \ldots, 3.1$, and let $p(z_k)$ denote the probability of observing the value z_k. The discriminator receives a 32-dimensional vector $\mathbf{p} = (p(z_1), p(z_2), \ldots, p(z_{32}))^T$ as input, where each element corresponds to the probability of observing the respective grid point, for both batches coming

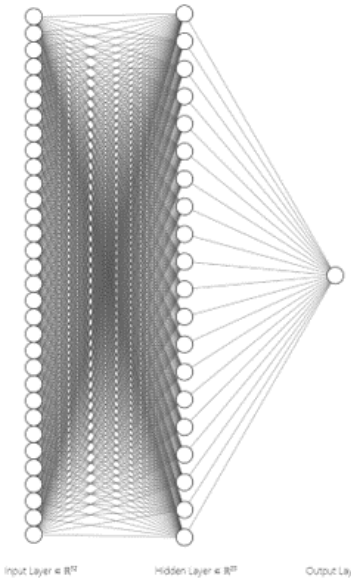


FIGURE 13. The discriminator is in both cases a classical neural network with $I = d_0 = 32, d_1 = 20, d_2 = 1$.

from the generator or from the training set.

The Discriminator contains a single hidden layer with 20 nodes. An affine transformation $AX + b$ is applied to the input vector, where $A \in \mathbb{R}^{32\times 20}$, $X = p = \begin{pmatrix} p(z_1) \\ \vdots \\ p(z_{32}) \end{pmatrix}$, and $b \in \mathbb{R}^{20}$. In the hidden layer, a Leaky ReLU function with a factor of 0.2 is applied:

$$\text{Leaky ReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.2x, & \text{else.} \end{cases} \tag{29}$$

This activation function helps mitigate the problem of inactive neurons (dying ReLU problem) and allows the network to better model complex relationships due to its nonlinearity. Subsequently, another affine transformation $AX + b$ is applied, where $A \in \mathbb{R}^{20\times 1}$, $X = \begin{pmatrix} x_1 \\ \vdots \\ x_{20} \end{pmatrix}$, and $b \in \mathbb{R}$. The output layer consists of a single node with a sigmoid activation function $\sigma(x) = \frac{1}{1+e^{-x}}$. This function maps any input x to the range $[0, 1]$, making it suitable for binary classification problems. The output of the discriminator can be interpreted as the probability that the input batch is classified as fake (i.e., generated by the generator). For example, an output of 0.7 means that the discriminator believes the input batch is generated artificially by the generator with a probability of 70%. The exact structure of the network can be seen in Figure 13.

Further Considerations: In qGANs, it is important that the network is neither too weak nor too powerful to ensure that neither the generator nor the discriminator dominates the other during training. The choice of the topology of the discriminator is based on empirical tests where different configurations of the discriminator were tested for their

ability to detect fakes. As with many neural networks, there is a large range of possible architectures, and it is difficult to find an optimal one.
Loss Functions: It is crucial to choose an appropriate loss function that matches the problem at hand and optimizes the prediction accuracy of the model. In this study, we employ two different approaches and compare them, namely:

(1) Binary Cross Entropy Setup

(2) Wasserstein Setup

In the first one, we choose the more classical approach using the binary cross entropy (BCE) loss function for both the generator G^ω and the discriminator D^ϕ, which was also used in the same composition in **Zoufal_2019**. The function measures the distance between the model's prediction and the actual result and provides a measure of how well the model predicts the correct class. The BCE loss function is a common loss measure for neural networks and in general for classification tasks. The general formula for the BCE loss function is:

$$BCE(\hat{y}_n, y_n) = -\frac{1}{N} \sum n = 1^N \left[y_n \log \hat{y}_n + (1 - y_n) \log(1 - \hat{y}_n) \right] \tag{30}$$

where y_n is the true label for the n-th sample, $\hat{y}_n$ is the predicted label for the n-th sample, and N is the number of samples. In our case, the discriminator loss is:

$$\begin{aligned} L_D(\phi, \omega) &= \frac{1}{2} \left(L_{\text{fake}} + L_{\text{real}} \right) \\ &= \frac{1}{2} \left([BCE_{x \sim p_{\text{fake}}(G^\omega)}(D^\phi(x), 0)] + [BCE_{x \sim p_{\text{real}}}(D^\phi(x), 1)] \right) \end{aligned} \tag{31}$$

Here,0 and 1 are the labels the discriminator should output upon receiving fake and real input, respectively. The generator has the loss function:

$$L_G(\phi, \omega) = -BCE_{x \sim p_{fake}(G)}(D^\phi(x), 0) \tag{32}$$

which is the negative of the first part of the discriminators loss.

In the second setup we use the Wasserstein distance for the generators loss as described in 2.2.3. For the discriminator we keep the BCE loss function and consequently do not adapt the architecture as it is done in other research on the WGAN **qWGAN_Chakrabarti**. Therefore, we have more regularization on the discriminator.
Training: We will train our qGANs for 2000 epochs using the AMSGrad ("Adaptive Moment Estimation with Rectified Adagrad") optimization method, an extension of the ADAM ("Adaptive Moment Estimation") algorithm. ADAM estimates the moments of the gradient to accelerate the learning process, using an adaptive learning rate that can vary for each parameter. AMSGrad was developed to address a specific issue with ADAM, as it tends to over- or underestimate the moments of the gradients, leading to slower learning. AMSGrad has been shown to be a more effective optimization method, particularly for complex neural networks.

For our BCE setup, we choose an initial learning rate of $\delta BCE = 0.0001$, and for our Wasserstein setup, we choose an initial learning rate of $\delta_W = 0.0001$. These steps are the maximum step the optimizer can improve in the gradient direction for each parameter per epoch. Furthermore, we define the momentum parameters for the AMSgrad optimizer

to be $\beta_1 = 0.9$ and $\beta_2 = 0.999$. These parameters determine the exponential decay rate of the learning rate based on the first and second moment of the gradients. These values are saying that the optimizer weights the previous estimate of the first moment of the gradient by a factor of 0.9 and the second moment of the gradient by a factor of 0.999, and are commonly used in other papers. Overall they directly affect the convergence of the optimization algorithm. As already mentioned, we set a batch size of 1000. This means that we carry out 10 batch iterations in one epoch, which has empirically proven to be the best way for us to optimise the process.

3.2. Computing the Payoff Function of the Option via an Ancilla Qubit

After bringing the distribution into the quantum state, we need to incorporate the payoff structure of the call option into our algorithm. In this slightly smaller step, we will follow the approach of **Rebentrost_2018**. To do so, we take an ancilla qubit $|0\rangle$ and build a comparator circuit that applies an X gate to the ancilla qubit when the call option is exercised. This can be expressed as follows:

$$|i\rangle|0\rangle \rightarrow \begin{cases} |i\rangle|0\rangle, & \text{if } i \leq K \\ |i\rangle|0\rangle, & \text{if } i > K \end{cases} \tag{33}$$

with K being the strike price of the option. Next, the qubits are in state

$$\sum_{i=0}^{K} \sqrt{p_i}\,|i\rangle|0\rangle + \sum_{i=K+1}^{2^n-1} \sqrt{p_i}\,|i\rangle|1\rangle. \tag{34}$$

We now need to map the payout function onto the amplitude of another ancilla qubit $|0\rangle$, including the comparison ancilla. This construction implements a channel $\mathcal{A}$ that approximates the following quantum state:

$$\mathcal{A}|0\rangle^{\otimes n+1} = \sum_{i=0}^{K} \sqrt{p_i}\,|i\rangle|0\rangle|0\rangle + \sum_{i=K+1}^{2^n-1} \sqrt{p_i}\,|i\rangle|1\rangle\left(\sqrt{1-f(i)}\,|0\rangle + \sqrt{f(i)}\,|1\rangle\right), \tag{35}$$

where $f(i) = \frac{i-K}{2^n-K-1}$. At this point one could apply the classic QAE algorithm as defined in 2.1.5 for the linear objective rotation given in 35. However, for practical purposes on current NISQ hardware it is recommended to use the IQAE from **woerner2019quantum** to approximate the state $|1\rangle$ in the newly implemented ancilla qubit with a probability of

$$\begin{aligned} \mathbb{P}[|1\rangle] &= \frac{1}{2^n-K-1} \sum_{i=K+1}^{2^n-1} p_i(i-K) \\ &= \frac{1}{2^n-K-1}\mathbb{E}[\max\{0, S_T - K\}]. \end{aligned} \tag{36}$$

3.3. Estimating the Probabilities of the Qubits

Comparing (35) with (15) in 2.1.5, we see that $\mathbb{P}[|1\rangle] = a$ and with this setup we only need to determine the amplitude, and thus we can finally evaluate $\mathbb{E}[\max\{0, S_T - K\}] = \mathbb{P}[|1\rangle]$. We now use the IQAE outlined in 2.1.6 to determine a. The evaluation on the "ibm_nairobi" device takes about 20 minutes[12]. With the $m = 2$ ancilla qubits used in our

[12]The extended duration is primarily due to a queue of usually around 500 pending jobs. Once the job starts, the actual execution is quite fast.

procedure, the error can be bounded by $\frac{\pi}{2^m}$ **Zoufal_2019**, assuming that 2^m is the number of quantum samples for the estimate evaluation. Thus, we achieve a quadratically better error convergence compared to a Monte Carlo simulation **Zoufal_2019Rebentrost_2018**.

4. Numerical Results

4.1. Numerical Results for the Performance Evaluation of the qWGAN

After completing the necessary preparations, we can now discuss the results and compare the outputs obtained. This section shows and discusses the improvements in performance of the qGAN due to the incorporation of the Wasserstein metric. To demonstrate the improvement in performance, we benchmark against the Binary Cross Entropy (BCE) version of a qGAN. The following plots were generated using Spyder with Python 3.9. The neural network of the discriminator was implemented in PyTorch, and the generator's variational form training was conducted using Qiskit's "aer_simulator." The final result of the option pricing was executed on the real quantum computer "ibm_oslo."

In this section we compare the performance of the two setups:

(1) BCE Setup

(2) Wasserstein Setup

The Binary Cross Entropy Setup employs BCE Loss for both the generator and the discriminator, as described in **Zoufal_2019**, the Wasserstein Setup has the discriminator still equipped with the BCE, but the generator will optimize the Wasserstein distance as we defined in Equation (26).

Figure 14 provides a direct comparison between the Binary Cross Entropy Setup (on the left) and the Wasserstein Setup (on the right). In both cases, we ran the qGANs for 2000 epochs and then selected the result with the best relative entropy. This selection process would likely be employed when working with the optimal loaded parameters.

Starting with the loss function plots at the top, we can see that the BCE loss appears to osscilate around a value. The actual goal should be for the discriminator loss function to reach a value where it can no longer distinguish between real and fake data, thereby outputting 0.5 for both cases (i.e., 50% certainty). Based on eq. (31), the optimal loss value for the discriminator in this case should be approximately $-\frac{1}{2}(\log(0.5)+\log(0.5)) \approx 0.693147$, and the generator should converge to around -0.693147. Similar results were observed for the BCE setup's loss function plots when varying the loss function. We see similar oscillations in the second setup (on the right), with both the discriminator and generator converging to suitable values. The discriminator settles at 0.693147, as it continues to use the BCE loss function. The generator converges to -0.5, as its results are debunked at 50%, which is optimal since the discriminator no longer has any insight.

The second row displays the generated distributions (in blue) compared to the log-normal training data (in grey). Here we can now see significant difference in terms of accuracy. Although the BCE distribution produces reasonably good results that largely overlap with the training data, the Wasserstein setup distribution generates almost perfect results, with only minimal deviations.

The third row shows the relative entropies, with the Wasserstein distance (in yellow) plotted for the Wasserstein setup as well. It is evident that both cases begin with an entropy of around 4.5, which corresponds to the input distribution introduced as a uniform superposition. Both setups optimize effectively at the start, but the BCE qGAN loses focus after about 100 epochs, which is also visible in the loss plot. After approximately 500

epochs, it finds a better result, but not as consistently and determined as the Wasserstein setup on the right. At the optimal point in the training, relative entropy values and Wasserstein distance values were provided for both setups.

Setup	Relative Entropy	Wasserstein Distance
1. BCE Setup	0.1858	0.017387
2. Wasserstein Setup	0.0043	0.001394

TABLE 1. Comparison of the relative entropy and Wasserstein distance for the two setups.

The results in table 1 clearly illustrate that the Wasserstein setup outperforms the Binary Cross Entropy setup in terms of both relative entropy and Wasserstein distance. This outcome suggests that the Wasserstein setup may be more robust and reliable when working with quantum generative adversarial networks. For both cases, we would like to add that we have extensively tried out different set-ups and finally decided on the one that best suits our needs. In Figure 15, the payoff of the European call option with a strike price of K = 1.0 is illustrated. The option evaluation results in Table 2 show that the Wasserstein setup provides a more accurate estimate of the expected payoff with respect to the target distribution when compared to the BCE setup.

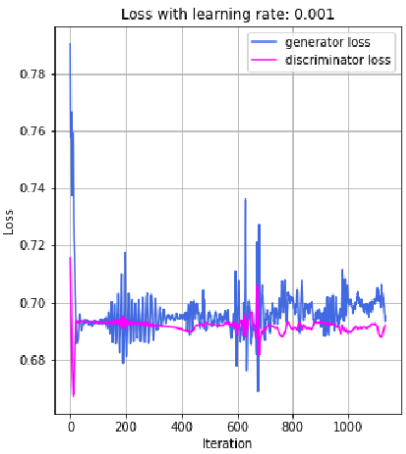
The trained payoff distribution is the analytical payoff based on the generated distribution, without the IQAE algorithm. It can be seen that the distribution from the BCE setup slightly underestimates the value, this also becomes clear when looking again at the distribution in Figure 14 (c), since the values greater than 1 are underestimated more often and the spike on the value 1.0 is not found in the payoff. The estimated values are now the results of the IQAE, which have been calculated on a real quantum computer and are therefore more error-prone. Both results are close to the value of the analytical trained distribution payoff. The IBM quantum hardware has also provided a 99% confidence interval, which again reflects the uncertainty of the quantum computer, and has less to do with the distributions. Of course, with the BCE setup, there are also a few more outliers in the distribution, which should additionally expand the Confidence Interval. In the images of Figure 14 one can clearly see that the distribution of the Wasserstein setup yields again very close fits to the actual value, significantly outperforming the BCE results.

In summary, the results obtained are quite satisfactory. Both confidence intervals encompass the actual value, and the estimates are convincingly close to the target. The Wasserstein setup particularly stands out, as it is only 0.3% above the target value, while the BCE setup is 2.1% below.

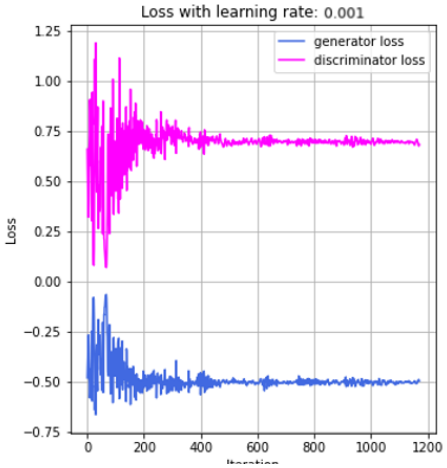
Metric	**BCE Setup**	**Wasserstein Setup**
Target distribution payoff	0.1618	0.1618
Trained distribution payoff	0.1584	0.1623
Estimated value	0.1574	0.1661
Confidence interval (lower)	0.1315	0.1512
Confidence interval (upper)	0.1833	0.1810

TABLE 2. Option evaluation results for BCE and Wasserstein setups with a strike price of K = 1.0

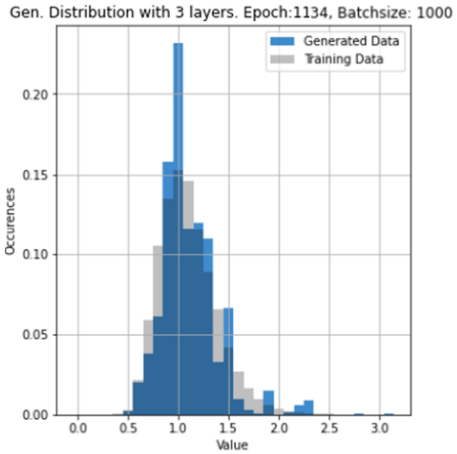
In **Zoufal_2019** the efficiency of qGANs in loading a distribution on 3 qubits is already demonstrated. In this study, we have shown that qGANs can achieve convincing results



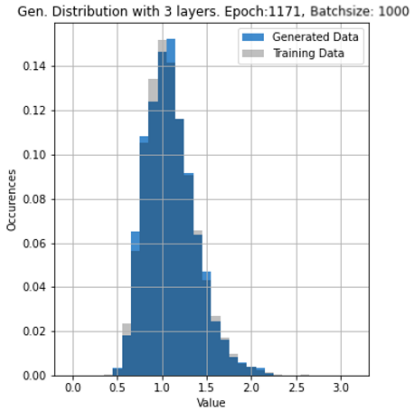
(A) Loss BCE Setup



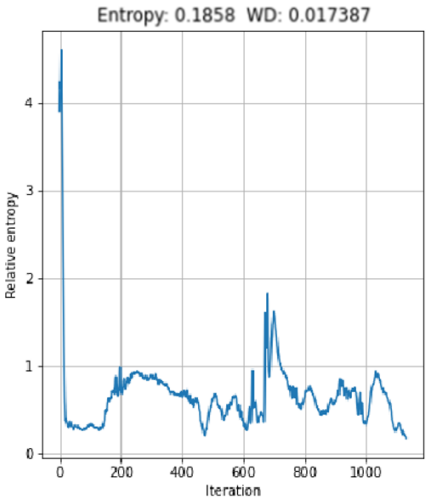
(B) Loss Wasserstein Setup



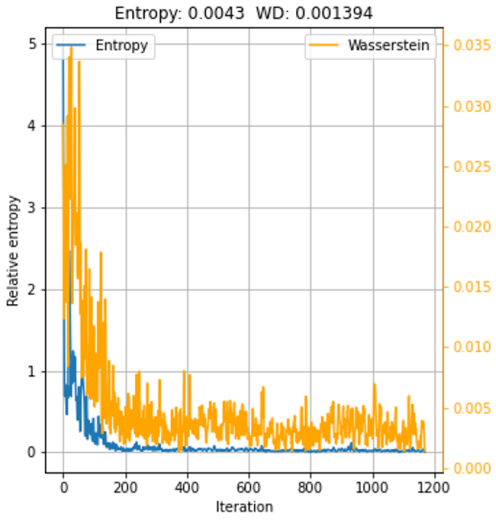
(C) Generated Distribution BCE Setup



(D) Generated Distribution Wasserstein Setup



(E) Relative Entropy BCE Setup



(F) Relative Entropy / Wasserstein

FIGURE 14. Loaded Distributions with BCE Loss and Wasserstein Distance.

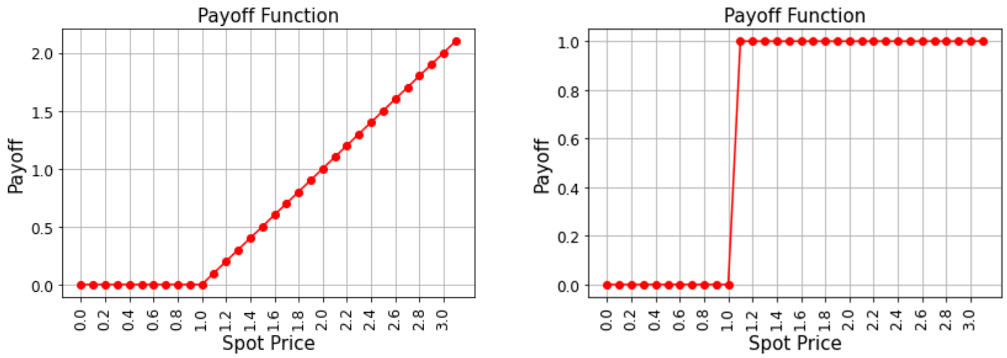


FIGURE 15. The payoff approximation of a European Call with K = 1.0 (left) and the approximation of a Digital Call with Strike K = 1.0 (right).

and scale well, even on 5 qubits, which results in quadrupling the grid from 8 to 32 values. Our choice of 5 qubits was constrained by the requirement for 2 ancilla qubits for option pricing and the availability of the best quantum computers, "ibm_oslo" and "ibm_nairobi," each with 7 qubits. But larger trials would certainly be very interesting in further research.

4.2. Experiments for Pricing with qWGANs on Synthetic- and Real Datasets

In this section, we will now focus on demonstrating the flexibility of qWGANs and the overall readiness of the approach for options pricing through further experiments.

4.2.1. Pricing of Other Payoff Structures (Digital Payoff)

To demonstrate versatility in the second step, we change the payoff structure to a digital payoff and remark that other options that do not exhibit explicit path-dependence can be handled in a similar way. The real difficulty lies in the loading of the distributions and the further procedure can still be adapted very easily, although no less impressive, since this is also where the superiority over MC sampling lies.

A Digital call option pays a binary output dependend on a predefined strike price. The payoff can be written as

$$\textbf{Payoff} = \begin{cases} 1 & \text{if } S_T > K \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

where S_T is the underlying asset price at maturity, and K is the strike price, this can again be seen in Figure 15. We again use K=1 and evaluate it on our Wasserstein setup from above. distribution. The estimation of the IQAE algorithm slightly underestimates the result this time. However, it is still only 1% or 0.1 cent below the actual value. And the Confidence interval still contains the true value, even if it is close.

Metric	**Wasserstein Setup**
Target distribution payoff	0.5258
Trained distribution payoff	0.5264
Estimated value	0.5203
Confidence interval (lower)	0.5111
Confidence interval (upper)	0.5271

TABLE 3. Digital Option Pricing using the Wasserstein setup with a strike price of K = 1.0

4.2.2. Heston Model

Now we want to take a more complex approach and modify the distributions or use a different model. We simulate the Heston model to generate a distribution at maturity, load it into quantum state, and then proceed as before to the steps involving option pricing. First, we simulate 100 instances of the variance process and the corresponding 100 asset price processes using the Heston model. We choose parameters similar to those used previously, i.e., a mean return of 10%, an initial and long-term average volatility of 25%. The results can be seen in Figure 16a.

To estimate the probability density function (PDF) of the asset prices at maturity, we use a Gaussian kernel density estimator (KDE). This provides us with a continuous distribution, as illustrated in fig. 16 (b). We then discretize this continuous distribution to obtain the discrete version shown in fig. 16(c), which serves as our new training distribution. From now on we can proceed as before with the training, that will again be done with the Wasserstein setup. The trained distribution and the progression of relative entropy can be seen in Figure 16 (d). With a relative entropy value of 0.0079 and a Wasserstein distance of 0.001526, the outcome closely approximates the target distribution. In addition, we can also mention that in the case of our Wasserstein setup, in both cases we achieved a good result after about 300 epochs already.

In Table 4 the evaluation of the IQAE is been presented, where we can observe a slight underestimation of the price. However, we still achieve a result that is very close to the target, and the confidence interval successfully captures the target payoff.

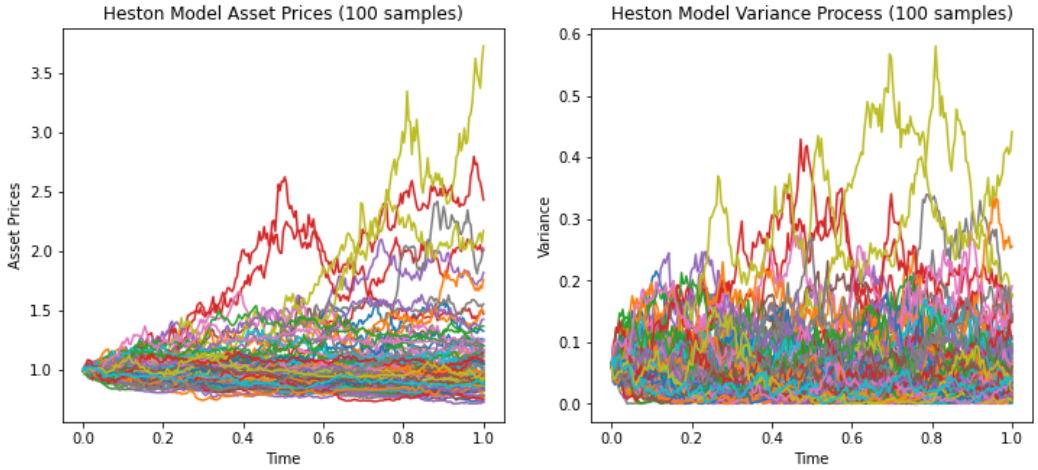
Metric	**Wasserstein Setup**
Target distribution payoff	0.1583
Trained distribution payoff	0.1528
Estimated value	0.1478
Confidence interval (lower)	0.1339
Confidence interval (upper)	0.1621

TABLE 4. Option Pricing for the Heston model using the Wasserstein setup with a strike price of K = 1.0

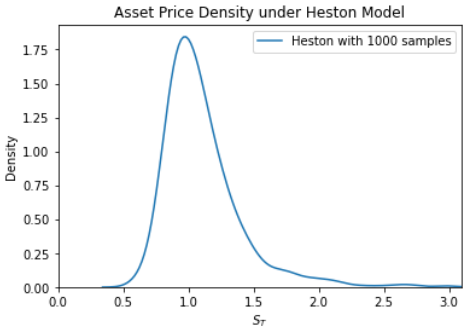
4.2.3. Real Data Experiments

After successfully modifying the underlying model for generating the distributions and loading it, we aim to further demonstrate the flexibility of the qGANs by using actual option data. This data can be freely downloaded from Yahoo Finance **yahoo_finance_2023**. The distributions of prices at maturity are not directly observable, but they are approximated from the implied volatility data and the option prices for a fixed maturity and a range of strike prices using the Breeden-Litzenberger formula[13]. The results are shown in Figure 17, where it is noticeable that the implied distribution for smaller strike prices, i.e., deep in the money, does not converge to 0, but is rather truncated. This truncation could be attributed to the limited availability of strike prices or other market factors that affect the option pricing. Since our primary objective is to challenge the qGAN with difficult tasks our primary goal is not to optimize this estimation step but to showcase

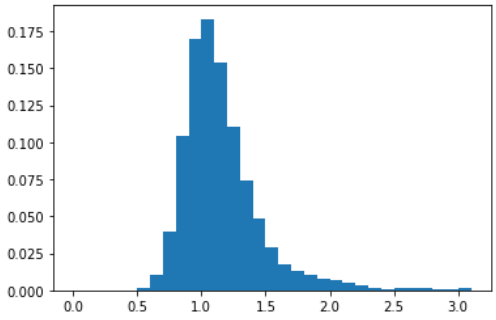
[13]As a reminder, for the density of a stock at maturity we have: $f(S_T) = e^{rT} \frac{\partial^2 C(K)}{\partial K^2}\Big|_{K=S_T}$ where C refers to the call price and K to the strike. The Breeden-Litzenberger approach approximates this density as: $g(K) \approx \frac{\exp(rT)}{\pi \delta K} \sum_{i=1}^{n} \left(C(K_{i+1}) - C(K_{i-1})\right) \gamma_i$.



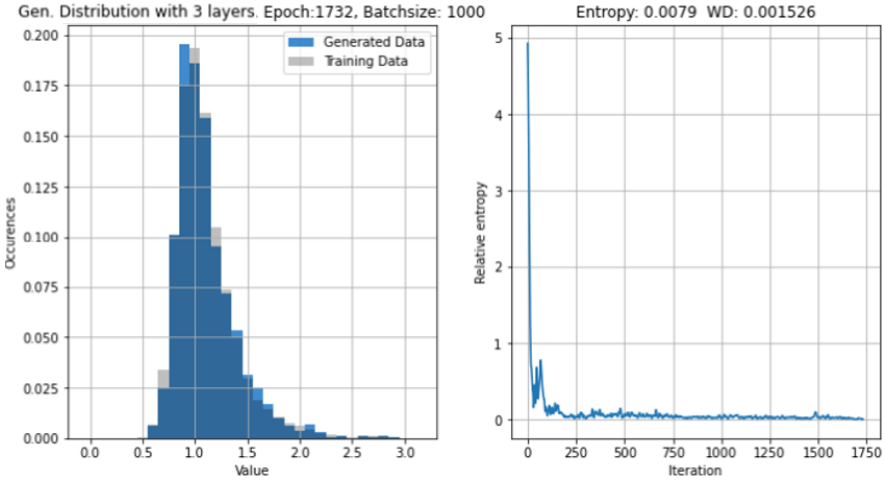
(A) 100 Heston Model Simulations of Asset and Variance Processes



(B) PDF of S_T in the Heston Model



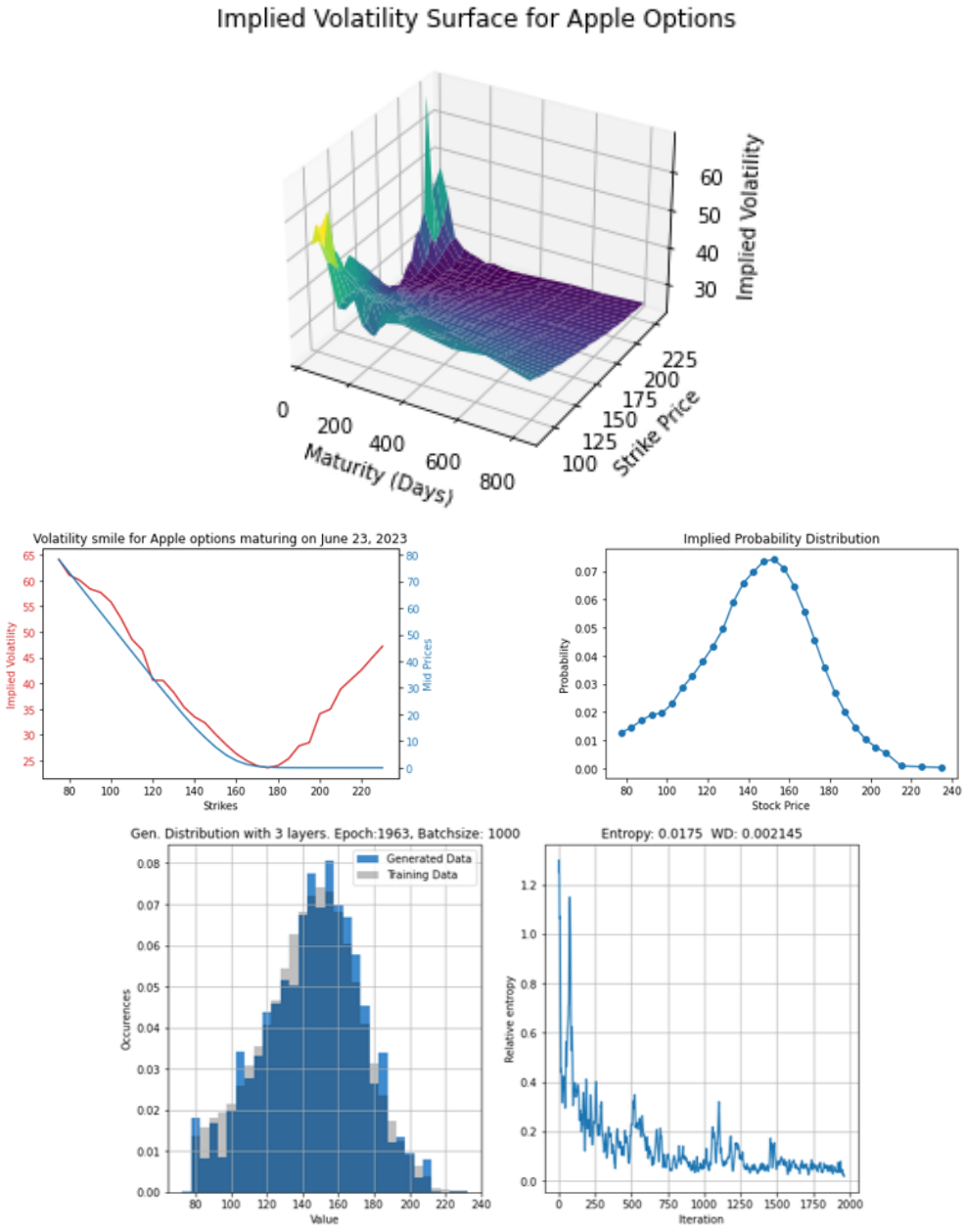
(C) Discretized Version of the PDF



(D) Trained Distribution and Evolution of Relative Entropy

FIGURE 16. Heston Model

that our method can be applied to arbitrary distribution. The distribution was again discretized and normalized. With our qGAN, we loaded the distribution into the quantum state and shows that our method yields a close fit with a relative entropy of 0.0175 and a Wasserstein distance of 0.002145. The distribution is more spread out this time, which may be more challenging to load. Despite some small level of approximation error that we allowed for in the estimation step, for completeness we also calculate the second and third steps of the programme including a calculation of the option price with IQAE. We



(A) Trained distribution

FIGURE 17. Option data: Apple stock, source: **yahoo_finance_2023**

choose a strike price of 150$, and at the time of data collection, the Apple share was around 152$. The results are convincing, as summarized in Figure 17.

Metric	**Wasserstein Setup**
Real Option Price	7.83$
Trained distribution payoff	9.30$
Estimated value	9.24$
Confidence interval (lower)	8.8131$
Confidence interval (upper)	9.7624$

TABLE 5. Option Pricing for Apple Options with 3 months to maturity using the Wasserstein setup with a strike price of K = 150 $.

5. Conclusion

In conclusion, our work represents a significant advancement in the field of quantum computing applications, with a particular focus on option pricing. By developing and applying a hybrid Quantum Wasserstein Generative Adversarial Network (qWGAN), we have substantially improved the efficiency and accuracy of loading arbitrary distributions into quantum states. This methodology, which uses the Wasserstein metric as loss function within the quantum generator, outperforms conventional qGAN models, demonstrating its considerable potential for quantum computing in general.

Many promising quantum algorithms necessitate efficient mechanisms for loading arbitrary distributions into quantum states. Our work with qWGANs provides a reliable tool in this aspect and hence a step forward in this field. We highlight its practical application by integrating our model into an option pricing procedure on a quantum computer. In particular, we have underlined its flexibility and versatility by successfully loading a variety of distributions into quantum states, ranging from the lognormal, to the distribution of the Heston model, and the one implied by actual Apple stock options.

Felix D. Fuchs, Technical University of Munich

Email address: felix.d.fuchs@tum.de

Blanka N. Horvath, Mathematical Institute, University of Oxford and Oxford Man Institute

Email address: blanka.horvath@maths.ox.ac.uk