POLITECNICO DI TORINO

Department of Mathematical Sciences

Master's Degree in Mathematical Engineering

Master's Degree Thesis

Hybrid Quantum-Classical Generative Adversarial Networks: a Study on Image Analysis and Probability Distribution Loading

Supervisor:
Prof. Bartolomeo MONTRUCCHIO

Candidate:
Mirko MATTESI

Company Supervisor
Data Reply S.R.L.
Dr. Davide CAPUTO

Academic Year 2022/2023

Acknowledgements

My academic journey has been quite challenging, considering my high school studies and the change of course of study between bachelor's and master's degrees.
This significant achievement is primarily dedicated to my family who have always supported my ambitions and provided me with the opportunity to reach this point. To Valerio and Francesco, for bearing with me during stressful times (and beyond); to my aunt Tiziana, for her constant presence despite the distance. I owe special thanks to two people. To my grandmother Ada, without whom I probably couldn't have achieved this goal. And to my mother Mariateresa, for never burdening me with anything and for always believing in me.
To my lifelong friends: Alessandro, Luca, Gianluca, and Giovanni, who have always been there for me during the most difficult times and have never made me feel the pressure of my absences due to exam studies. To Luigi, a reliable shoulder I can always lean on, who made the years we spent together in our house in Turin special. And to Antonio, Marco, Giuseppe, Alessia, Carolina, and Ludovica, who along with Luigi form my second family in Turin.
Lastly, I would like to express my gratitude to all the colleagues at Data Reply with whom I had the opportunity to work during my internship and thesis months, for welcoming me and making me feel comfortable, and most importantly, for teaching me so much. A special thanks goes to Blanca and Davide, for granting me this opportunity, and to Luca, for his support throughout this journey and his patience.

Summary

This work explores the intersection of quantum computing and machine learning, with a particular focus on investigating the potential of hybrid quantum-classical algorithms in generative modeling tasks. The work begins by providing an overview of the current state of Noisy Intermediate-Scale Quantum (NISQ) devices, considering the challenges posed by inherent noise and the limited number of available qubits. It then establishes a solid mathematical foundation of quantum computing principles, covering fundamental concepts such as qubits, superposition, entanglement and quantum gates. Based on this understanding, the study investigates the structure and training methods of Parametrized Quantum Circuits (PQCs), highlighting their versatility in machine learning and optimization tasks. Two significant PQC-based algorithms, the Variational Quantum Eigensolver (VQE) and the Quantum Approximate Optimization Algorithm (QAOA), are discussed in detail. We also explore different data encoding schemes, which play a crucial role in effectively representing information within quantum systems.
The core focus of this work is the exploration of quantum Generative Adversarial Networks (qGANs), which extend classical GANs using principles from quantum computing. The performance and potential benefits of qGANs are assessed through two experiments. The first experiment examines the qGAN's ability to generate realistic images using the MNIST dataset, a standard benchmark in image generation. The second experiment is focused on efficiently learning and encoding probability distributions into quantum states. More specifically, through extensive experimentation, this work investigates the effectiveness of qGANs in capturing the characteristics of both univariate and multivariate Student's t-distributions, models that find frequent application in the fields of finance and risk management.

Sommario

Questo lavoro esplora l'intersezione tra il quantum computing e il machine learning, con particolare enfasi sull'indagine delle potenzialità degli algoritmi ibridi quantistico-classici in problemi di modellazione generativa. Il lavoro fornisce in prima battuta una panoramica dello stato attuale dei dispositivi quantistici Noisy Intermediate-Scale Quantum (NISQ), considerando le difficoltà ingegneristiche relative al rumore degli attuali dispositivi quantistici e al numero limitato di qubit disponibili. Successivamente, viene stabilita una solida base matematica dei principi del calcolo quantistico, coprendo concetti fondamentali come qubit, sovrapposizione, entanglement e quantum gates. Sulla base di questa comprensione, lo studio indaga la struttura e i metodi di training dei Circuiti Quantistici Parametrizzati (PQCs), evidenziando la loro versatilità nei campi del machine learning e dell' ottimizzazione. Due importanti algoritmi basati su PQC, ovvero il Variational Quantum Eigensolver (VQE) e il Quantum Approximate Optimization Algorithm (QAOA), vengono discussi in dettaglio. Inoltre, vengono presentati diversi schemi di codifica dei dati, i quali svolgono un ruolo cruciale nel rappresentare efficacemente le informazioni all'interno dei sistemi quantistici.
L'obiettivo principale di questo lavoro è l'esplorazione delle quantum Generative Adversarial Networks (qGANs), che estendono le GAN classiche utilizzando i principi del calcolo quantistico. Le capacità e i potenziali vantaggi delle qGANs sono valutati attraverso due esperimenti. Il primo esperimento esamina l'abilità delle qGANs di generare immagini realistiche utilizzando il dataset MNIST, un riferimento standard nella generazione di immagini. Il secondo esperimento è incentrato sull'apprendimento e la codifica efficiente di distribuzioni di probabilità in stati quantistici. Più precisamente, attraverso un'ampia sperimentazione, questo lavoro indaga l'efficacia delle qGANs nel catturare le caratteristiche di distribuzioni t di Student sia univariate che multivariate, le quali trovano frequente applicazione nei campi della finanza e della gestione del rischio.

Table of Contents

Chapter 1

Introduction

Quantum Computing is a relatively new and rapidly evolving field at the intersection of computer science, mathematics and physics that investigates how to leverage some of the peculiar properties of quantum physics for use in computer science.
Modern computers are physical devices that utilize electronic circuits to process information through algorithms or software programs, which provide instructions on how to manipulate electrical signals to execute computations. While these processes involve microscopic particles, such as electrons, atoms, and molecules, they can be described using a macroscopic, classical theory for electronic circuits. However, when microscopic systems (like photons, electrons, and atoms) are directly used to process the same information, a set of specific mathematical rules is required in order to describe them, as nature behaves very differently on small scales than our intuition suggests. Such mathematical framework is known as *quantum theory*, and a computer whose computations are exclusively described by the laws of quantum theory is referred to as a *quantum computer*.
Starting from the 1990s, quantum physicists and computer scientists have been examining the possibilities of building quantum computers and their potential applications. They devised a variety of languages to describe the calculations carried out by a quantum system, enabling us to examine these devices theoretically. The *circuit model* is the most well-known language for formulating quantum algorithms and its key ingredients are represented by the concepts of *qubits*, which replace conventional bits, and *quantum gates*, which enable qubits to be processed.
During the early 1990s, two of the most important algorithms for quantum computing were devised. One of these, introduced by Peter Shor in 1994, was a revelation that caused a stir in both the physics and computer science fields. Shor's algorithm theoretically demonstrated that quantum computers can provide an exponential speedup when it comes to factoring large integers [1]. This was a notable revelation because the current modern cryptography methods rely on the belief that factoring

large numbers is an unsolvable computational task. The potential of quantum computers to efficiently solve this problem drew a lot of attention to the field. The second algorithm was developed by Lov Grover in 1996, and provided a quadratic speedup over classical counterparts for searching through unsorted data sets [2].
Although there were highly encouraging initial outcomes in algorithmic research, quantum computing primarily remained a theoretical discipline due to the difficulties encountered in hardware advancement. Creating a quantum computer in a laboratory setting is actually a challenging task, requiring precise control over extremely small systems while simultaneously avoiding any interference that could disrupt the fragile quantum coherence necessary for the desired quantum effects. To maintain this coherence throughout numerous computational operations, error correction plays a crucial role. However, error correction for quantum systems is considerably more complex than for classical systems, presenting a significant engineering hurdle for the development of a noiseless quantum computer.
The first physical realizations of quantum computers were announced by two distinct research groups at the end of 1990s, employing nuclear magnetic resonance technology and a computational volume of two qubits [3, 4]. This kind of quantum computer architecture was also used for a first implementation of a simplified version of Shor's factorization algorithm [5] and a first physical realization of Grover's search algorithm [6]. However, progress in experiments using this architecture slowed down afterwards, as nuclear magnetic resonance proved to be difficult to scale.
The hardware development in the quantum computing sector did not take up quickly again until recent years. Indeed, the attempt to commercialize quantum technologies has given rise to the "near-term" era of quantum computing, which has markedly impacted research in this field. There are already a number of physical implementations of quantum hardware, with trapped ions and superconducting tunnel junctions appearing to be among the most promising. Both private investors and government agencies are investing substantial resources into quantum technologies, expanding the domain of quantum hardware beyond the confines of academia. Moreover, the development of cloud computing has revolutionized the way quantum hardware is utilized, making it more accessible for individuals and companies interested in building quantum software. A watershed event happened in 2016 when IBM launched the "IBM Q Experience" cloud platform for quantum computing, which started with a single 5-qubit machine but at the time of writing has over 20 devices and has played a crucial role to the development and expansion of the quantum computing industry. Other major software companies such as Amazon and Microsoft are also offering cloud computing quantum services on their cloud platforms, AWS and Azure, respectively.
Significant advancements have been made in the creation of the so-called Noisy Intermediate-Scale Quantum (NISQ) devices, which are the first models of what

could eventually become a full scale, noise-free quantum computer. These devices, which currently consist of up to a few hundreds qubits (depending on the technology employed) that may not all interact directly with each other, suffer from noise and consequently, for a high number of operations on qubits, may lead to inadequate results. While quantum devices in the NISQ era have the potential to demonstrate the benefits of quantum computing, the requirement to limit algorithms to only a few qubits and gates significantly affects the design of quantum algorithms themselves. The ideal objective would be to identify practical computational problems that intermediate-scale quantum devices can solve while exhibiting (preferably) exponential and verifiable improvements in run-time compared to the best-known classical algorithms. *Quantum supremacy* is the term used in this context to describe the theoretical point at which quantum computers can complete a task that the most powerful classical supercomputers cannot accomplish in a reasonable amount of time. In 2021, a team associated with the University of Science and Technology of China exhibited their remarkable 62-qubit programmable quantum computer called " Zuchongzhi" based on a superconducting processor [7]. Later, the same device was upgraded and used to attain quantum supremacy.[1] The demonstration of quantum supremacy is an exciting development that highlights the vast potential of quantum computing, despite the remaining challenges. Hence, it can be concluded that the use of quantum computers is unlikely to completely displace classical computers, but rather their potential lies in tackling challenging scientific computing tasks that demand considerable computational resources.
There are several reasons why quantum computers are thought to be able to swiftly and effectively solve a variety of scientific problems, with machine learning and optimization frequently identified as promising fields. Firstly, they can perform numerous calculations simultaneously using qubits and *superposition*, which is the ability of a quantum system to exist in multiple states (or states of uncertainty) at once. This means that a qubit can have a probability of being in one state or another, unlike classical bits which can only be deterministically in one state at a time. This is similar to the act of spinning a coin, where the coin is in a state of "heads" and "tails" simultaneously until it comes to rest and collapses into one of the two possible states. In the same way, a qubit can exist in a superposition of "0" and "1" states until it is measured, at which point it collapses into a definite classical state. To describe the state of a qubit in a superposition, a wave function can be used to determine the probability of finding the qubit in each of its possible states. Secondly, they can use *quantum entanglement* to correlate two or more qubits so

[1]The original quantum supremacy barrier was surpassed by Google's Sycamore processor in 2019 [8], though the challenge posed by Zuchongzhi's problem is approximately six orders of magnitude more complex. Additionally, doubts have been raised regarding the intricacy of the problem that Sycamore solved [9].

that the state of one qubit can be dependent on the state of the other. When two qubits are entangled, no matter how far away they are, measuring the state of one qubit instantaneously reveals the information about the other qubit. Due to its non-local nature, this association cannot be accounted for by conventional physics. Finally, quantum computers can be utilized for simulations of quantum systems, such as chemical reactions and materials properties, which are computationally expensive for classical computers but may be possible on quantum computers. For instance, to simulate a penicillin molecule with 42 atoms, the exponentially huge parameter space of electron configurations would require 10^{86} states — more states than the number of atoms in the universe; the quantum systems can achieve this with only 286 qubits.
In general, a quantum computer with n qubits can thus represent 2^n states at the same time due to the ability of qubits to exist in a superposition of states. However, it is important to note that using n qubits for computation is not equivalent to having 2^n classical bits. This is because qubits exist in a quantum state as long as they are not measured, but once they are measured, they collapse into a single classical state and can no longer be used for quantum computations.
This property, along with the fundamental concepts and principles of quantum mechanics used in quantum computing, will be presented in the initial part of this work. This includes a detailed discussion on key elements such as qubits, superposition, quantum gates, and other essential components that underpin quantum computation. Based on this understanding, we will explore the concept of Parametrized Quantum Circuits (PQCs), which offer a versatile framework for machine learning and optimization tasks. We will see that the structure of a PQC consists of an initial state preparation, an encoder component to embed input data, and a quantum component formed by gates with trainable parameters. The circuit's output is obtained through measurement, yielding a classical quantity that can be used to update the trainable parameters through classical optimization techniques, aiming to minimize a specific objective function. Then, we will describe two significant PQC-based algorithms, namely the Variational Quantum Eigensolver (VQE) and the Quantum Approximate Optimization Algorithm (QAOA). VQE is a hybrid quantum-classical algorithm designed to determine the ground state energy of a specific Hamiltonian. It employs a PQC to prepare trial states and classically optimizes the circuit parameters to minimize the expectation value of the Hamiltonian with respect to the trial states. QAOA is another hybrid quantum-classical algorithm developed for solving combinatorial optimization problems. It involves encoding the optimization problem into a quantum state and evolving the state under a time-dependent Hamiltonian. The Hamiltonian is constructed from a series of unitary operators, and the circuit parameters are optimized classically to find the best approximate solution to the optimization problem. Additionally, we will explore various data encoding schemes that are crucial for efficient information

representation within quantum systems.
The core focus of this work lies in the exploration and analysis of quantum Generative Adversarial Networks (qGANs). The classical GANs model has garnered significant interest in the machine learning community due to its ability to generate realistic data samples. Quantum Generative Adversarial Networks are extensions of classical GANs that leverage quantum computing principles to enhance the capabilities of generative modeling. To assess the performance and potential benefits of these models, we will conduct two main experiments. The first experiment will involve evaluating the qGAN ability to generate realistic images using the MNIST dataset, which consists of a large collection of handwritten digits that have been labeled with their corresponding numeric values. This dataset is commonly used for tasks such as digit recognition, image classification, and deep learning model evaluation. Finally, we will focus on the specific application of efficiently learning and loading probability distributions into quantum states. This will include investigating the effectiveness of the qGAN in capturing the characteristics of both univariate and multivariate Student's t-distributions, which are frequently utilized models in finance and risk management.

Chapter 2

Foundations of Quantum Computing

In this chapter, we will cover the fundamentals of quantum computing, laying the theoretical groundwork for exploring its applications in the algorithms presented in subsequent chapters.

2.1 Qubit and Quantum State

Quantum computers rely on qubits, or quantum bits, as the basic building block for information processing. A *qubit* is a two-level quantum mechanical system whose state can be represented by a unit vector in a complex two-dimensional Hilbert space $\mathbb{H}$, which serves as the mathematical framework for characterizing its probabilistic nature. The term "Hilbert space" generally refers to an infinite-dimensional inner product space that is complete. However, it is also commonly used in the case of finite-dimensional spaces, which inherently fulfill the completeness requirement due to their finite nature. We denote such complex vector space as $\mathbb{C}^n$, where n indicates the number of dimensions.
When working with vectors in quantum mechanics, the widely used notation is the Dirac one, also known as *braket* notation. A *ket* is simply a column vector labeled with a specific symbol, such as $|\psi\rangle$, which represents a vector denoting a state labeled as ψ. A *bra*, on the other hand, is a row vector denoted as $\langle\psi|$ representing the Hermitian conjugate of a ket; therefore, we have that $\langle\psi| = |\psi\rangle^\dagger$.
With this notation, an *inner product* between two states $|\psi\rangle$ and $|\phi\rangle$ can be expressed as $\langle\psi|\phi\rangle$; consequently, the (Euclidean) norm of a generic vector $|\psi\rangle \in \mathbb{H}$ can be expressed as

$$||\psi|| = \sqrt{\langle\psi|\psi\rangle}$$

When it comes to quantum computing, the labels $|0\rangle$ and $|1\rangle$ are the most common used due to their association with classical bit values. Notably, such special vectors $|0\rangle$ and $|1\rangle$, known as *computational basis states*, form an orthonormal basis of $\mathbb{C}^2$ and are given by:

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

By utilizing the basis vectors, it is possible to express any two-dimensional vector into a linear combination of them. This also means that a one-qubit system can exist in a *superposition* of basis states. Therefore, any general quantum state $|\psi\rangle$ may be expressed as:

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

where $\alpha, \beta \in \mathbb{C}$ are called *probability amplitudes*. The requirement that the state is represented by a unit vector implies that

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.2}$$

which is known as the *normalization constraint*, and is essential for the consistency of quantum measurements, as we will see later in the chapter. Although a qubit may take infinitely many different states, when it undergoes measurement, its state collapses to a single classical state, either 0 or 1. The values of $|\alpha|^2$ and $|\beta|^2$ indicate the probability of finding the qubit in either the 0 or 1 state after measurement, respectively. By way of illustration, a qubit may exist in the state

$$\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

which, upon measurement, yields the outcome 0 with a probability of fifty percent ($|1/\sqrt{2}|^2$), and the outcome 1 with an equal probability.

An electron orbiting around a nucleus serves as an example of a physical system that can be represented by a vector in a two-dimensional Hilbert space [10]. When examining energy as the primary variable, the electron may occupy a nearly infinite number of energy levels, resulting in an infinite-dimensional Hilbert space. Quantum mechanics principles state that these energy levels are discrete, indicating that the electron can only adopt specific energy values instead of a continuous range. In certain situations, an electron is most likely to be found in the ground state (minimum energy level) or the first excited state, while higher energy levels necessitate a significantly large amount of energy, making them nearly unattainable. In these cases, we can disregard the subspace spanned by energy levels beyond the

first excited state. This simplifies the system to a two-level model, which can be represented by a two-dimensional vector in the space consisting of the two lowest energy levels.
Another instance of a two-level quantum system involves the spin state of specific particles [10]. In quantum physics, particles possess a degree of freedom known as *spin*, which is absent in classical descriptions. Many particles belong to the spin-$\frac{1}{2}$ category, and their spin states are represented by vectors in a two-dimensional Hilbert space. A suitable basis for this space comprises a unit vector for the particle's 'spin-up' state and an orthogonal unit vector for the 'spin-down' state. Therefore, the overall spin state of a spin-$\frac{1}{2}$ particle is a combination of both spin-up and spin-down states.
One crucial aspect of state vectors is that a state represented by the vector $e^{i\theta}|\psi\rangle$ is equivalent to the state represented by the vector $|\psi\rangle$, where $e^{i\theta}$ is any complex number with unit norm. For instance, the state $|0\rangle + |1\rangle$ is equivalent to the state represented by the vector $e^{i\theta}|0\rangle + e^{i\theta}|1\rangle$. However, *relative phase factors* between two orthogonal states in a superposition hold physical importance. Consequently, the state represented by the vector $|0\rangle + |1\rangle$ is physically distinct from the state represented by the vector $|0\rangle + e^{i\theta}|1\rangle$. In theory, we could describe quantum states using equivalence classes of unit vectors, but we will simply use a unit vector, acknowledging that any two vectors related by a *global phase* are equivalent.
The state of a qubit can be conveniently represented as a point on a unit sphere, commonly referred to as the *Bloch sphere*, as shown in Figure 2.1. Usually, the north and south poles of the Bloch sphere are selected to correspond to the standard basis vectors $|0\rangle$ and $|1\rangle$, respectively. Since state vectors must have unit norm and are equivalent up to global phase, two real parameters θ and ϕ are adequate to describe them. Notably, the mapping

$$\alpha = \cos\left(\frac{\theta}{2}\right), \quad \beta = e^{i\phi}\sin\left(\frac{\theta}{2}\right)$$

allows for representing the state $|\psi\rangle$ of a generic qubit in the computational basis as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle \tag{2.3}$$

where $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$. The interpretation of the angles θ and ϕ as spherical coordinates is clear, which means that the vector $|\psi\rangle$ in the Hilbert space can be visualized as a vector $\vec{a} \in \mathbb{R}^3$, pointing from the origin to the point of the sphere with coordinates $\vec{a} = (\sin\theta\cos\phi,\, \sin\theta\sin\phi,\, \cos\phi)$. Vector $\vec{a} \in \mathbb{R}^3$ is commonly referred to as *Bloch vector*. Several operations on individual qubits, which will be explained later in this chapter, can be neatly described using the Bloch sphere representation. Nonetheless, it is important to remember that this

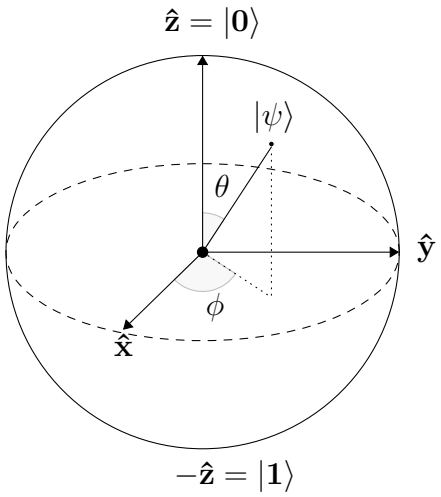


Figure 2.1: State $|\psi\rangle$ of a qubit on the Bloch Sphere.

visualization has its limitations since there is no straightforward way to extend the Bloch sphere to multiple qubits.
The standard orthonormal basis $\{|0\rangle, |1\rangle\}$ is not the only option for choosing basis vectors. Indeed, any pair of linearly independent unit vectors $|u\rangle$ and $|v\rangle$ from the two-dimensional complex vector space can be used as a basis. This means that

$$\alpha|0\rangle + \beta|1\rangle = \alpha'|u\rangle + \beta'|v\rangle$$

for some $\alpha', \beta' \in \mathbb{C}$ such that $|\alpha'|^2 + |\beta'|^2 = 1$. An example of an alternative basis is the *Hadamard basis* $\{|+\rangle, |-\rangle\}$, defined by the vectors

$$|+\rangle \doteq \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}, \quad |-\rangle \doteq \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{pmatrix}$$

It is important to specify which basis is being used, as different bases can give different measurement outcomes for the same vector. For instance, a state vector with components

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix}$$

measured in the standard orthonormal basis will give outcomes $|0\rangle$ and $|1\rangle$ with an equal probability of $\frac{1}{2}$. However, when measured in the Hadamard basis, it will give the outcome $|+\rangle$ with probability 1. In general, the choice of the basis is determined by the measurement process or the physical implementation of the quantum computer.

2.2 Multiple Qubits

So far we have only examined simple systems with two possible states, such as a single electron's spin. However, when we begin to describe more complex composite systems, such as two particles, the situation becomes even more interesting. Quantum mechanics employs the algebraic concept of *tensor product* of vector spaces to offer an abstract description of such systems. This is a consequence of the formalism that relies on Hilbert spaces, whereby the Hilbert space $\mathbb{H}^{\otimes n}$ of an $n-$composite quantum system is the tensor product of the Hilbert spaces of its individual subsystems [11], e.g.,

$$\mathbb{H}^{\otimes n} = \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes ... \otimes \mathbb{C}^2 \tag{2.4}$$

As an example, let us consider a two-qubit system; as we have already learned, we can describe the quantum state of each of them separately using a two-dimensional complex vector:

$$|\zeta\rangle = \begin{pmatrix} \zeta^1 \\ \zeta^2 \end{pmatrix}, \quad |\varphi\rangle = \begin{pmatrix} \varphi^1 \\ \varphi^2 \end{pmatrix}$$

By taking the tensor product of such two states, we obtain a single vector $|\psi\rangle \in \mathbb{C}^4$ describing the state of our composite system:

$$|\psi\rangle = |\zeta\rangle \otimes |\varphi\rangle = \begin{pmatrix} \zeta^1 \\ \zeta^2 \end{pmatrix} \otimes \begin{pmatrix} \varphi^1 \\ \varphi^2 \end{pmatrix} = \begin{pmatrix} \zeta^1 \cdot \begin{pmatrix} \varphi^1 \\ \varphi^2 \end{pmatrix} \\ \zeta^2 \cdot \begin{pmatrix} \varphi^1 \\ \varphi^2 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \zeta^1\varphi^1 \\ \zeta^1\varphi^2 \\ \zeta^2\varphi^1 \\ \zeta^2\varphi^2 \end{pmatrix} \tag{2.5}$$

In Dirac notation, it is common practice to eliminate the $\otimes$ symbol between the kets that undergo the tensor product and merge them into a single ket, i.e.,

$$|\zeta\rangle \otimes |\varphi\rangle \doteq |\zeta\varphi\rangle = |\psi\rangle \tag{2.6}$$

Similarly to how we can use a two-state orthonormal basis to represent a two-dimensional state vector, a four-state orthonormal basis is suitable for a composite system that comprises two two-level systems. Hence, the standard orthonormal basis for the tensor product $|\zeta\rangle \otimes |\varphi\rangle$ will consist of the following four orthonormal unit vectors

$$|00\rangle \doteq \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |01\rangle \doteq \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \quad |10\rangle \doteq \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad |11\rangle \doteq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{2.7}$$

and the system state will be described by four probability amplitudes

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \tag{2.8}$$

with $\alpha, \beta, \gamma, \delta \in \mathbb{C}$ such that $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$.
It is important to note that not all states of a combined system can be decomposed into the tensor product form $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_n\rangle$. Notably, if the qubits are prepared independently and kept isolated, then each qubit can be considered a closed system, and the state of the composite system can be expressed as the tensor product of the individual states. In this case, two probability amplitudes are required to fully describe the state of each individual qubit before measurement, thus $2n$ probability amplitudes are required to describe the state $|\psi\rangle$ of the composite system.
However, if the qubits are allowed to interact, then they form a closed system that includes all qubits together and the tensor product representation of the system state $|\psi\rangle$ does not exist; in this case, we need to specify 2^n probability amplitudes to describe the state of the composite system, which is an effective measure of useful information that can be stored in the system when qubits are *entangled*. The concept of *entanglement* will be explored in detail in section 2.7.

2.3 Pure and Mixed States

Up to now, we have assumed that the state of a quantum system has a definite state vector. However, there are cases where the qubit can only be described by one of a particular set of state vectors, with corresponding probabilities (which must add to 1). For instance, let us assume that a qubit has a probability of $\frac{1}{3}$ of being in the state $|\psi_1\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and a probability of $\frac{2}{3}$ of being in the state $|\psi_2\rangle = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. This probability distribution describes a state known as a *mixture* or *ensemble* of the states $|\psi_1\rangle$ and $|\psi_2\rangle$. The state of a system in such a situation is referred to as a *mixed state*, while a system with a uniquely specified state vector is said to be in a *pure state*.
One way to describe a generic mixed state on n qubits is by using an ensemble representation

$$\{(|\psi_1\rangle, p_1)\,,\ (|\psi_2\rangle, p_2)\,,\ ...,\ (|\psi_k\rangle, p_k)\} \tag{2.9}$$

which implies that the system is in the pure (n-qubit) state $|\psi_i\rangle$ with probability p_i, for $i = 1, 2, ..., k$. It is worth noting that a pure state can be viewed as a special instance of a mixed state, where all p_i except one equal zero.
However, using the representation (2.9) consistently in all our computations would be challenging. Fortunately, there exists an alternative way to express mixed states in terms of operators on the Hilbert space $\mathbb{H}$. These operators are known as *density operators*, and their matrix representation is referred to as a *density matrix*. A density matrix can be constructed by taking the *outer product* of a state vector $|\psi\rangle$:

$$\rho = |\psi\rangle\langle\psi| \tag{2.10}$$

A state $|\psi\rangle$ is considered a *pure state* if it can be expressed in such a form. Even if the state vector is in a superposition, the corresponding density matrix will still describe a pure state. In quantum physics, which is inherently probabilistic, it is advantageous to conceive of a pure state as a pure ensemble, i.e., a set of identical particles with the same physical configuration. For pure states, the following properties hold:

- A density matrix is *idempotent*, i.e., $\rho^2 = (|\psi\rangle\langle\psi|)\,(|\psi\rangle\langle\psi|) = |\psi\rangle\langle\psi|\psi\rangle\langle\psi| = |\psi\rangle\langle\psi| = \rho$.
- Given any orthonormal basis $\{|e_k\rangle\}$ of $\mathbb{H}$, the trace of a density matrix is 1: $tr(\rho) = \sum_k \langle e_k|\rho|e_k\rangle = \sum_k \langle e_k|\psi\rangle\langle\psi|e_k\rangle = \sum_k \langle\psi|e_k\rangle\langle e_k|\psi\rangle = \langle\psi|\psi\rangle = 1$.
- Similarly, we have that $tr(\rho^2) = 1$.
- A density matrix is *Hermitian*, or *self-adjoint*: $\rho^\dagger = (|\psi\rangle\langle\psi|)^\dagger = |\psi\rangle\langle\psi| = \rho$.
- A density matrix is *positive semi-definite*: $\langle\phi|\rho|\phi\rangle = \langle\phi|\psi\rangle\langle\psi|\phi\rangle = |\langle\phi|\psi\rangle|^2 \geq 0$, where $|\phi\rangle$ is an arbitrary vector.

For a mixed state such as (2.9), the density matrix is given by:

$$\rho_{mixed} = \sum_{i=1}^{k} p_i |\psi_i\rangle\langle\psi_i| \tag{2.11}$$

In other words, the density matrix ρ_{mixed} for a mixed state is a probabilistic combination of matrices ρ_i for pure states, i.e., $\rho_{mixed} = \sum_{i=1}^{k} p_i\rho_i$, where $0 \leq p_i \leq 1$ and $\sum_{i=1}^{k} p_i = 1$. Considering a statistical perspective again, a mixed state is comprised of indistinguishable particles that are distributed among different physical configurations. This is why the term "density matrix" is appropriate, as a mixed state is essentially a distribution of probabilities over pure states. A mixed state is characterized by the following properties:

- Idempotency is violated: $\rho_{mixed}^2 = \left(\sum_{i=1}^{k} p_i|\psi_i\rangle\langle\psi_i|\right)\left(\sum_{i=1}^{k} p_i|\psi_i\rangle\langle\psi_i|\right) = \sum_{i=1}^{k}\sum_{j=1}^{k} p_i p_j |\psi_i\rangle\langle\psi_i|\psi_j\rangle\langle\psi_j| \neq \rho_{mixed}$
- $tr(\rho_{mixed}) = 1$.
- $tr(\rho_{mixed}^2) < 1$.
- Hermicity.
- Positive semidefiniteness.

We typically use the same notation, ρ, to denote both mixed and pure states, without a lower index to distinguish between them. To emphasize the difference between superposition and mixed states, let us consider the computational basis $\{|0\rangle, |1\rangle\}$. A superposition in this two-dimensional space is a linear combination of two vectors:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $|\alpha|^2 + |\beta|^2 = 1$. The corresponding density matrix has non-zero interference terms, or off-diagonal elements, and is given by

$$\rho = |\psi\rangle\langle\psi| = \begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix} \tag{2.12}$$

where the symbol * stands for complex conjugation. On the other hand, a mixed state is represented by a density matrix with no interference terms:

$$\rho = |\alpha|^2|0\rangle\langle 0| + |\beta|^2|1\rangle\langle 1| = \begin{pmatrix} |\alpha|^2 & 0 \\ 0 & |\beta|^2 \end{pmatrix} \tag{2.13}$$

It is important to note that a density matrix depends on the chosen basis; nonetheless, its trace is invariant under a change of basis.
Pure and mixed states have a nice geometrical interpretation in terms of the Bloch sphere, introduced in section 2.1. Specifically, pure states correspond to points on the surface of the sphere, while mixed states correspond to points within the sphere. In order to demonstrate this, we shall focus on the simple two-dimensional case. One can show that any density operator ρ for a single qubit can be expressed as

$$\rho = \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{a_x}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \frac{a_y}{2}\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} + \frac{a_z}{2}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.14}$$

where $\vec{a} = (a_x, a_y, a_z) \in \mathbb{R}^3$ is the corresponding Bloch vector. The eigenvalues of ρ are $\lambda_{1,2} = \frac{1}{2}\left(1 \pm ||\vec{a}||^2\right)$ and, since density matrices are positive semi-definite, it is required that $||\vec{a}||^2 \leq 1$. For pure states, the relation $tr(\rho^2) = tr(\rho)^2 = 1$ must be satisfied, while for a mixed state, it holds that $tr(\rho^2) < tr(\rho)^2 = 1$, thus we get

$$tr(\rho^2) = \frac{1}{2}\left(1 + ||\vec{a}||^2\right) = 1 \iff ||\vec{a}||^2 = 1 \tag{2.15}$$

In simpler terms, the Bloch vector $\vec{a}$, which corresponds to the point in $\mathbb{R}^3$ representing the state of our system, lies on the surface of the unit sphere if the state is pure. On the contrary, if the state is mixed, the vector $\vec{a}$ represents an interior point within the sphere.

2.4 Measurement of Quantum States

In classical physics, a measurement of a system state has no significant impact on the system itself, and any external disturbance resulting from the measurement is usually negligible and does not require consideration. Furthermore, classical physics imposes no limit on measurement accuracy other than the quality of the measurement technique, and there are no restrictions on measuring different physical quantities simultaneously, except for the complexity of the experimental setup.
In contrast, quantum physics operates differently. Measurement activities are destructive to the quantum state, and measurements of non-commuting observables cannot be made simultaneously. The measurement process in quantum physics is controlled by the concepts of *observable* and *measurement operators*, which we shall explore in more detail in this section.
Specifically, each measurable property of a physical system, known as *observable*, corresponds to a unique *Hermitian operator*. The values of the physical observables are represented by the *expectation values* of their corresponding Hermitian operators. In general, the expectation value $\langle \mathcal{A} \rangle$ of a Hermitian operator $\mathcal{A}$ in the normalized state $|\psi\rangle$ is calculated as:

$$\langle \mathcal{A} \rangle \doteq \langle \psi | \mathcal{A} | \psi \rangle = \mathrm{Tr}\left(\rho \mathcal{A}\right) \tag{2.16}$$

where ρ denotes the density matrix associated to state $|\psi\rangle$ and $\mathrm{Tr}\left(A\right)$ is the trace of a generic square matrix A.
It turns out that the eigenstates of a Hermitian operator and the concept of superposition are closely related in quantum mechanics. Indeed, the spectral theorem [12] establishes that for any Hermitian operator $\mathcal{A}$, the state $|\psi\rangle$ of a system can be expressed as a superposition of the eigenstates $\left(|\psi_i\rangle\right)_{i=1,\dots,n}$ of $\mathcal{A}$:

$$|\psi\rangle = \sum_{i=1}^{n} \alpha_i |\psi_i\rangle \tag{2.17}$$

where the coefficients $\left(\alpha_i\right)_{i=1,\dots,n}$ are complex probability amplitudes assumed to be normalized, i.e., such that $\sum_{i=1}^{n} |\alpha_i|^2 = 1$. The measurement postulate of quantum mechanics states that measuring the Hermitian operator $\mathcal{A}$ in the state $|\psi\rangle$ described in equation (2.17) yields possible outcomes corresponding to the eigenvalues $\left(\lambda_i\right)_{i=1,\dots,n}$ of $\mathcal{A}$, and the probability of measuring a given eigenvalue λ_i is given by $p_i = |\alpha_i|^2$. After the measurement outcome λ_i, the state of the system will collapse to the corresponding eigenstate $|\psi_i\rangle$. If we perform a second measurement immediately afterward in the same computational basis, we will obtain the same outcome without any uncertainty.
The process of quantum measurement is described using *measurement operators*

$(\mathcal{P}_i)_{i=1,\ldots,n}$ that act on the state space of the system with n possible outcomes. If the state of the system before the measurement is $|\psi\rangle$, then the probability of obtaining outcome i is given by

$$\mathbb{P}(i) = \langle\psi|\mathcal{P}_i^\dagger\mathcal{P}_i|\psi\rangle \tag{2.18}$$

In order to ensure that the sum of the probabilities of all possible outcomes adds up to 1, the measurement operators must also fulfill the *completeness condition*

$$\sum_{i=1}^{n}\mathcal{P}_i^\dagger\mathcal{P}_i = \mathcal{I} \tag{2.19}$$

where $\mathcal{I}$ denotes the identity operator.
From the standpoint of quantum computing, we are interested in measurement operators that are *projections* (i.e., such that $\mathcal{P}^2 = \mathcal{P}$) onto the *computational basis*, such as the standard orthonormal basis. For instance, in the case of a single qubit, the measurement operators can be defined as follows:

$$\mathcal{P}_0 \doteq |0\rangle\langle 0| = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix} \quad \text{and} \quad \mathcal{P}_1 \doteq |1\rangle\langle 1| = \begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix} \tag{2.20}$$

We can easily check that $\mathcal{P}_0^2 = \mathcal{P}_0$ and $\mathcal{P}_1^2 = \mathcal{P}_1$, as expected for projection operators, and that the completeness condition (2.19) holds. Moreover, if the qubit is in state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, then the measurement operator $\mathcal{P}_0$ will yield $|0\rangle$ with a probability of $|\alpha|^2$, and the measurement operator $\mathcal{P}_1$ will yield $|1\rangle$ with a probability of $|\beta|^2$:

$$\mathcal{P}_0|\psi\rangle = |0\rangle\langle 0|\,(\alpha|0\rangle + \beta|1\rangle) = \alpha|0\rangle\langle 0||0\rangle + \beta|0\rangle\langle 0||1\rangle = \alpha|0\rangle$$
$$\mathcal{P}_1|\psi\rangle = |1\rangle\langle 1|\,(\alpha|0\rangle + \beta|1\rangle) = \alpha|1\rangle\langle 1||0\rangle + \beta|1\rangle\langle 1||1\rangle = \beta|1\rangle$$

The measurement postulate in quantum mechanics asserts that when an immediate measurement is performed in the same computational basis, the result will be the same without any uncertainty. The crucial phrase here is *"the same computational basis"*. What would happen if the subsequent measurement is carried out in a different basis, specified by another set of linearly independent unit vectors from the state space? To answer the question, let us assume that the qubit is in state

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle = |+\rangle$$

Measuring $|\psi\rangle$ in the $\{|0\rangle, |1\rangle\}$ computational basis will yield states $|0\rangle$ and $|1\rangle$ with an equal probability of $\frac{1}{2}$. Let us now suppose that our measurement was 0; the qubit state then becomes

$$|\psi'\rangle = 1\cdot|0\rangle + 0\cdot|1\rangle$$

When the measurement is performed again in the same $\{|0\rangle, |1\rangle\}$ computational basis, the state $|0\rangle$ is obtained with probability 1, consistently with the measurement postulate. On the other hand, if we had measured the state $|\psi'\rangle$ in the Hadamard basis $\{|+\rangle, |-\rangle\}$, we would have observed equal probabilities for outcomes $|+\rangle$ and $|-\rangle$. Assuming that we measured $|-\rangle$, the state of the qubit now becomes

$$|\psi''\rangle = 0 \cdot |+\rangle + 1 \cdot |-\rangle$$

If we perform another measurement of the state $|\psi''\rangle$ in the Hadamard basis $\{|+\rangle, |-\rangle\}$, we get state $|-\rangle$ with probability 1. However, from the perspective of the $\{|0\rangle, |1\rangle\}$ computational basis, the state of the qubit is an equal superposition of states $|0\rangle$ and $|1\rangle$, meaning that we have an equal chance of measuring either $|0\rangle$ or $|1\rangle$ in this basis.
Measurement is critical in quantum computing, as it involves collapsing a quantum state and extracting classical information from it. When qubits encoding a quantum state are measured, a classical bit string is produced. Nevertheless, since the measurement process yields probabilistic outcomes, it is essential to perform multiple measurements on the same quantum state. This process generates an adequately large set of classical bit strings to produce reliable statistics.

2.5 Time-Evolution of a Closed System

In general a physical system evolves over time, which means the state vector $|\psi\rangle$ of a system is actually a function of time, represented as $|\psi(t)\rangle$. According to quantum theory, the evolution of the state vector of a closed quantum system is governed by the *Schrödinger equation*

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H|\psi(t)\rangle \tag{2.21}$$

where $\hbar$ is Planck's constant and H is a time-independent Hermitian operator called the *Hamiltonian* of the system. The Hamiltonian of a quantum system is an operator that describes the total energy of the system, with its eigenvalues representing the possible energy levels of the system. Having knowledge of the Hamiltonian offers all the essential information about the system dynamics.
In the Schrödinger equation (2.21), the state $|\psi(t_1)\rangle$ of a closed quantum system at time t_1 is linked to the state $|\psi(t_2)\rangle$ at time t_2 by a *unitary operator* $\mathcal{U}(t_1, t_2)$ that depends solely on t_1 and t_2. This relationship can be expressed as:

$$|\psi(t_2)\rangle = \mathcal{U}(t_1, t_2)|\psi(t_1)\rangle \tag{2.22}$$

where $\mathcal{U}(t_1, t_2)$ can be derived from the Hamiltonian H as

$$\mathcal{U}(t_1, t_2) = \exp\left(-\frac{iH(t_2 - t_1)}{\hbar}\right) \tag{2.23}$$

We recall for the sake of convenience that an operator $\mathcal{U}$ on a Hilbert space $\mathbb{H}$ is unitary if $\mathcal{U}^\dagger\mathcal{U} = \mathcal{U}\mathcal{U}^\dagger = \mathcal{I}$, or equivalently, if $\mathcal{U}^\dagger = \mathcal{U}^{-1}$. This indicates that all operations are inherently *reversible* by definition, as applying the adjoint after the transformation brings the vector back to its initial state, i.e.,

$$\mathcal{U}|\psi(t_1)\rangle = |\psi(t_2)\rangle \tag{2.24}$$

$$\mathcal{U}^\dagger|\psi(t_2)\rangle = |\psi(t_1)\rangle \tag{2.25}$$

Unitary operators preserve the inner product (and consequently norms, lengths, and distances), which implies that for any two vectors $|u\rangle$ and $|v\rangle$, the inner product between $|u\rangle$ and $|v\rangle$ is the same as the inner product between $\mathcal{U}|u\rangle$ and $\mathcal{U}|v\rangle$:

$$\langle u|\mathcal{U}^\dagger\mathcal{U}|v\rangle = \langle u|v\rangle$$

This property is crucial, as it ensures that two orthogonal vectors will remain orthogonal after undergoing a unitary transformation.
In quantum mechanics, physical transformations such as rotations, translations, and time evolution correspond to unitary operators that map quantum states to other quantum states, thus implementing the computation. This perspective enables us to view unitary operators as the *quantum logic gates* responsible for carrying out quantum computation processes. In the next section, we will delve deeper into the various types of quantum gates, their properties, and how they are used to perform operations on qubits.

2.6 Quantum Gates

2.6.1 Single-Qubit Gates

In contrast to classical computing, which allows for only two logic gates operating on a single bit (the identity gate and the NOT gate), quantum computing boasts an infinite number of single-qubit logic gates. Indeed, any unitary 2×2 matrix can be considered a quantum logic gate. Among these gates, however, some are more significant or easier to implement compared to others. In the following discussion, we shall focus on a few of these noteworthy gates, beginning with the *identity gate (I)* and the *Pauli matrices (X, Y,* and *Z)*.
A transformation of the qubit state can be visualized as a transition from one point on the Bloch sphere to another. As a result, the unitary matrix responsible for this transformation can be regarded as a rotation operator. In this context, we can refer to gate operations as rotations and consider rotation angles as gate parameters. The effect of the I gate is straightforward – it does not alter the qubit state; the Pauli matrices X, Y, and Z, on the other hand, rotate the qubit state by π radians

around the x, y, and z axes, respectively:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.26}$$

By carrying out basic algebraic operations, we can readily confirm that the X gate performs a bit flip while the Z gate induces a phase flip, leaving $|0\rangle$ invariant, and changing the sign of $|1\rangle$:

$$\begin{aligned} X \text{ Gate:} \quad & \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ Z \text{ Gate:} \quad & \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \end{aligned}$$

For this reason, the *X gate* is also referred to as the *NOT gate*, while the *Z gate* is commonly called the *PHASE gate*. The visualization of these operations can be facilitated by utilizing graphical representations of the quantum gates:

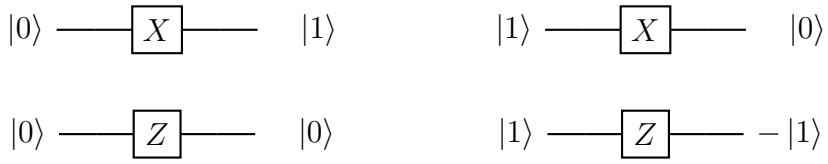


Figure 2.2: Graphical representation of the X and Z gates.

In this representation, horizontal lines represent *quantum registers*, while boxes symbolize quantum gates. Quantum registers and quantum gates collectively form the graphical representation of *quantum circuits*, which are sequences of quantum gates that transform quantum states to execute quantum computation. Quantum circuits are interpreted from left to right, with the initial quantum state displayed on the left side and the final state on the right side of the circuit.
We know that one of the primary reasons behind the potency of quantum computing is the capability of a qubit to exist in a superposition of basis states. But how can we achieve a superposition of states $|0\rangle$ and $|1\rangle$ for a qubit that was initialized as $|0\rangle$ (or $|1\rangle$)? The solution lies in the *Hadamard gate, H,* which generates an equal superposition of states $|0\rangle$ and $|1\rangle$ when applied to either state $|0\rangle$ or state $|1\rangle$:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.27}$$

$$H \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad \text{and} \quad H \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 0 \end{pmatrix} - \frac{1}{\sqrt{2}} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Figure 2.3: Graphical representation of the Hadamard gate H.

It turns out that the Hadamard gate acts as its own inverse, meaning that applying it twice reverses the effect of the first application; mathematically, $H^2 = I$, or equivalently, $H = H^{-1}$.

Figure 2.4: Hadamard gate H applied twice.

Other useful one-qubit gates include *phase shift gates,* S and T, which shift the phase by $\frac{\pi}{2}$ and $\frac{\pi}{4}$, as opposed to the Z *gate* that shifts the phase by π:

$$S = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\frac{\pi}{2}} \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 \\ 0 & e^{-i\frac{\pi}{4}} \end{pmatrix} \tag{2.28}$$

Lastly, it is essential to mention the *adjustable* one-qubit gates that execute a rotation of the qubit state around a particular axis by an arbitrary angle θ. These gates are represented as $R_x(\theta)$, $R_y(\theta)$, and $R_z(\theta)$, and their expressions are as follows:

$$\begin{aligned} R_x(\theta) &= \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \\ R_y(\theta) &= \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \\ R_z(\theta) &= \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \end{aligned} \tag{2.29}$$

The adjustable gates are crucial in Parameterized Quantum Circuits (PQC), which we will explore in much detail in chapter 3.
The final operator applied to the quantum register is the *measurement operator*, shown in Figure 2.5. After measurement (in the computational basis), a qubit is converted into a classical bit, and its value becomes a known binary number.

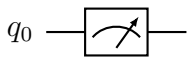


Figure 2.5: Application of the measurement operator to qubit q_0.

2.6.2 Two-Qubit Gates

Just like one-qubit gates, which are represented by 2×2 unitary matrices, we can also construct multi-qubit gates. Specifically, n-qubit gates are described by $2^n \times 2^n$ unitary matrices. As multi-qubit gates can operate on several qubits simultaneously, they can be utilized for *entangling* qubits, causing their states to become interdependent. Moreover, it is possible to create conditional operators that act on a target qubit only if a control qubit is in the state $|1\rangle$. These types of gates are referred to as *controlled gates*, and below we shall provide the details of some of them.
Controlled gates are represented in a quantum circuit by a straight line connecting two quantum registers. The control qubit is represented by one quantum register, marked by a dot at the end of the connecting line. The other quantum register represents the target qubit, on which the intended conditional operator is placed. Figure 2.6 demonstrates this with a *Controlled Y (CY) gate* example. In this case, q_1 is the quantum register for the control qubit, q_2 is the quantum register for the target qubit, and the Y operator is applied to the target qubit q_2.

$$CY = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{pmatrix} \tag{2.30}$$

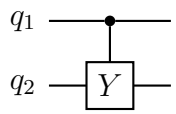


Figure 2.6: Controlled Y (CY) gate.

The *Controlled NOT gate*, commonly denoted as *CNOT* or *CX*, serves as another example of a two-qubit controlled gate. In this case, the Pauli X gate is applied to the target qubit when the control qubit is in state $|1\rangle$. The corresponding unitary matrix for the *CNOT* gate is:

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.31}$$

This gate is frequently depicted in quantum circuits using an *XOR* logic symbol (a circled plus) placed on the target qubit quantum register. This is because the truth table for the target qubit aligns with that of the *XOR* logic gate.

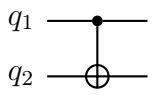


Figure 2.7: Controlled X ($CNOT$) gate.

The *CZ (CPHASE) gate* applies a Pauli Z (phase flip) operation to the target qubit, provided that the control qubit is in state $|1\rangle$ and is represented by the following unitary matrix:

$$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{2.32}$$

Interestingly, the CZ gate does not distinguish between the target and control qubits – the outcome remains the same regardless of which qubit is designated as the control or target qubit.

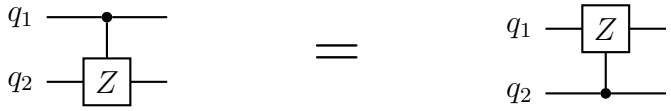


Figure 2.8: Controlled Z (CZ) gate.

The *SWAP gate* is a two-qubit gate that interchanges the quantum states of two qubits. In other words, it swaps the information stored in the two qubits. The $\sqrt{SWAP}$ gate is a *universal gate*, which means that any multi-qubit gate can be constructed using just the $\sqrt{SWAP}$ gate and single-qubit gates. Their representations in terms of unitary matrices are:

$$SWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \sqrt{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1+i}{2} & \frac{1-i}{2} & 0 \\ 0 & \frac{1-i}{2} & \frac{1+i}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.33}$$

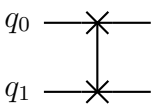


Figure 2.9: $SWAP$ gate.

Typically, the choice of the set of universal gates from which all other gates may be built is governed by the characteristics of the physical system used to perform quantum computation. Related gates like $iSWAP$ and $\sqrt{iSWAP}$ are inherent gates in systems that exhibit Ising-type interactions [13]:

$$iSWAP = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \sqrt{iSWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} & 0 \\ 0 & \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.34}$$

Finally, the XY *gate* is an example of adjustable two-qubit gate that performs a rotation by a specific angle θ between the $|01\rangle$ and $|10\rangle$ states:

$$XY(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\left(\frac{\theta}{2}\right) & i\sin\left(\frac{\theta}{2}\right) & 0 \\ 0 & i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{2.35}$$

We may observe that $XY(\pi) = iSWAP$ and $XY(\frac{\pi}{2}) = \sqrt{iSWAP}$. The $iSWAP$ gate, along with the CZ gate, holds significant importance in the formation of quantum circuits, as any two-qubit gate can be represented using a maximum of three CZ or three $iSWAP$ gates [14].

2.6.3 Quantum Gate Decompositions

At the time of writing, the most commonly utilized NISQ computing technologies include trapped ions and superconducting qubits. In both instances, one-qubit gates are significantly faster and have higher fidelity than two-qubit gates [15]. As a result, one-qubit gates can be considered computationally efficient, and their quantity is less of a concern. On the other hand, it is crucial to optimize the use of two-qubit gates since circuits with fewer of them can perform better in terms of less amount of noise introduced. Consequently, we should be familiar with native two-qubit gates specific to a system—gates that can be naturally implemented using standard hardware control methods. While more complex gates can be broken down into subcircuits consisting of native gates, a better approach is to create algorithms that exploit the native gates, eliminating the need for non-native two-qubit gates. For instance, Rigetti's Aspen system [16], based on superconducting qubits, utilizes two native two-qubit gates, CZ and XY. Building circuits with these gates instead of, for example, $SWAP$ gates, would result in improved performance.
Nevertheless, designing hardware-dependent algorithms may not always be feasible or desirable. As the selection of native gates is inherently limited, it is helpful to

be acquainted with a few fundamental decompositions. The following relationships can be confirmed through direct calculations and serve an essential function in quantum circuit construction:

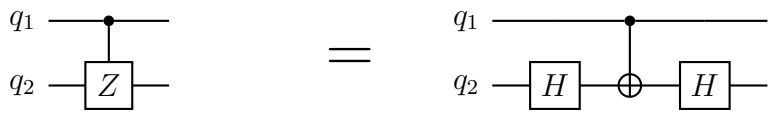


Figure 2.10: *CZ* gate decomposition into *CX* and Hadamard gates.

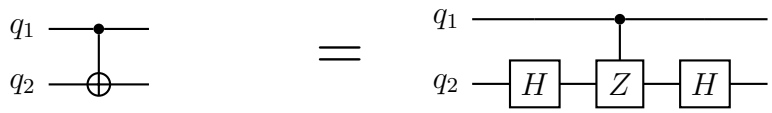


Figure 2.11: *CX* gate decomposition into *CZ* and Hadamard gates.

Finally, we mention the *CNOT* representation of the *SWAP* gate. Notably, due to the limited connectivity of NISQ devices (nearest neighbours for most qubits), the *SWAP* gate is particularly valuable, and its efficient implementation using available native gates is crucial. The *SWAP* gate can be depicted as a sub-circuit composed of three *CX* gates:

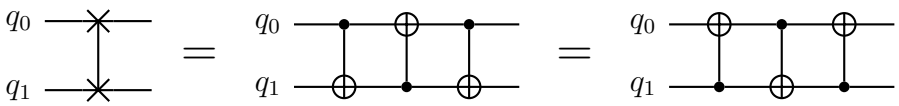


Figure 2.12: *SWAP* gate decomposition using three *CX* gates.

In conclusion, the ability to decompose non-native two-qubit gates into subcircuits composed of native two-qubit gates and high-fidelity one-qubit gates enables the development of hardware-independent quantum algorithms. This approach ensures that the designed algorithms can be adapted to various quantum computing platforms, making them more versatile and broadly applicable across different systems.

2.7 Entanglement

In section 2.2, we mentioned that entanglement is a crucial feature of quantum computing. Here, we explain this in detail and present examples for two-qubit

systems.
We know that an $n-$qubit system can be in any superposition of the 2^n basis states:

$$\sum_{i=0}^{2^n-1} c_i|i\rangle = c_0|00...00\rangle + c_1|00...01\rangle + ... + c_{2^n-1}|11...11\rangle \tag{2.36}$$

with

$$\sum_{i=0}^{2^n-1} |c_i|^2 = 1 \tag{2.37}$$

If such a state can be expressed as tensor product of the individual qubit states, then the qubit states are *not entangled*. For instance, it can be readily verified that

$$\begin{aligned}&\frac{\sqrt{3}}{4\sqrt{2}}\Big(|000\rangle + \frac{1}{\sqrt{3}}|001\rangle + \frac{3}{\sqrt{3}}|010\rangle + |011\rangle + |100\rangle + \frac{1}{\sqrt{3}}|101\rangle + \frac{3}{\sqrt{3}}|110\rangle + |111\rangle\Big)\\ &= \left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle\right) \otimes \left(\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle\right),\end{aligned} \tag{2.38}$$

so that the quantum state is not entangled, but only in superposition. On the contrary, an *entagled* state cannot be expressed as a tensor product of individual qubit states. For instance, the two-qubit state

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{2.39}$$

cannot be decomposed into a tensor product. In other words, for any complex numbers $c_1, c_2, c_3, c_4 \in \mathbb{C}$ such that $|c_1|^2 + |c_2|^2 = |c_3|^2 + |c_4|^2 = 1$, we have

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \neq (c_1|0\rangle + c_2|1\rangle) \otimes (c_3|0\rangle + c_4|1\rangle)$$

We observe that 2^n probability amplitudes are required to describe the state on the left-hand side of (2.38), while only $2n$ probability amplitudes are needed to describe the state on the right-hand side of (2.38). The number of probability amplitudes necessary to fully describe the state of a system is directly connected to the amount of information it can encode. Entanglement enables us to encode a substantially larger amount of information compared to individual independent qubits.
Let us now ask what happens when we measure entangled qubits. In (2.39), both qubits are in a state of equal superposition, meaning that if we measure the first qubit, we will obtain both 0 and 1 with a probability of $\frac{1}{2}$. If we measure the second qubit instead, we will also get 0 and 1 with equal probability. However, the situation drastically changes if we tried to measure the second qubit after the first one has already been measured. In this scenario, the state of the second

qubit is entirely determined by the act of measuring the first qubit, and there is no uncertainty about its value anymore: if the first qubit was measured as 0, the second qubit is also in state 0, and if the first qubit was measured as 1, the second qubit is also in state 1. In other words, measuring one qubit causes the superposition to collapse and immediately affects the other qubit.
Qubits can be entangled using two-qubit gates. The two-qubit state described by (2.39) is known as one of the four maximally entangled *Bell states*. This state can be derived from the unentangled state $|00\rangle$:

$$|00\rangle = (1 \cdot |0\rangle + 0 \cdot |1\rangle) \otimes (1 \cdot |0\rangle + 0 \cdot |1\rangle)$$

by employing the *Bell circuit*, which is composed of Hadamard and Controlled *NOT* gates:

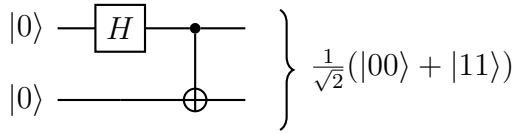


Figure 2.13: Bell circuit.

The other three Bell states can be derived by applying this circuit to the unentagled states $|01\rangle, |10\rangle,$ and $|11\rangle$:

$$|01\rangle \rightarrow \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$$
$$|10\rangle \rightarrow \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$$
$$|11\rangle \rightarrow \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$$

Entanglement can also be achieved using various other two-qubit gates. The specific gate employed may depend on the hardware implementation, and could include a *SWAP*, *CPHASE*, another fixed two-qubit gate, or an adjustable two-qubit gate like $XY(\theta)$.

2.8 Quantum Hardware and Challenges

After establishing the theoretical framework for quantum bits and quantum gates, it becomes crucial to comprehend how these can be implemented from a hardware perspective. One of the most critical challenges in building a quantum computer is the phenomenon of *decoherence*, which represents the loss of information from a

quantum system to its environment due to entanglement and coupling with the surroundings. The system under consideration is no longer a closed system when interaction with the environement is in action. To describe the evolution of an open quantum system in the presence of decoherence, density matrices are commonly used. Notably, the diagonal elements of the density matrix correspond to the probabilities of finding the system in the corresponding state, while the off-diagonal elements describe the coherence between the states of the system. Nevertheless, due to interactions with the environment, the off-diagonal elements of the density matrix decay over time, leading to a loss of coherence and a mixed state that cannot be described by a pure wavefunction.
To see this with an example, let us consider the pure state $|+\rangle$. The corresponding density matrix with respect to the standard computational basis $\{|0\rangle, |1\rangle\}$ is given by

$$\rho = |+\rangle\langle+| = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

Assuming now that the time evolution of $|+\rangle$ is

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \rightarrow \frac{|0\rangle + e^{iwt}|1\rangle}{\sqrt{2}}$$

the corresponding density matrix becomes

$$\rho(t) = \frac{1}{2}\begin{pmatrix} 1 & e^{iwt} \\ e^{-iwt} & 1 \end{pmatrix}$$

The "population" in state $|+\rangle$ can be obtained by computing the expectation value of the operator $\rho(t)$ with respect to the state $|+\rangle$:

$$\begin{aligned} \langle+|\rho(t)|+\rangle &= \mathrm{Tr}\left(|+\rangle\langle+|\rho(t)\right) \\ &= \mathrm{Tr}\left(\frac{1}{4}\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & e^{iwt} \\ e^{-iwt} & 1 \end{pmatrix}\right) \\ &= \frac{1}{4}\left(2 + e^{iwt} + e^{-iwt}\right) \\ &= \frac{1}{2} + \frac{1}{2}\cos\left(wt\right) \end{aligned} \tag{2.40}$$

where we have used the well-known relation $e^{iwt} + e^{-iwt} = 2\cos\left(wt\right)$. The oscillation in (2.40) is due to the off-diagonal elements of $\rho(t)$, and it is called the *coherence* of the system. The state is pure at any time t. However, in a real physical system, the coherence often decays exponentially over time due to interactions with the surroundings. In such a situation, the density matrix can be expressed as

$$\rho(t) = \frac{1}{2}\begin{pmatrix} 1 & e^{iwt-\gamma t} \\ e^{-iwt-\gamma t} & 1 \end{pmatrix}$$

and the population in the state $|+\rangle$ decays in accordance with

$$\langle +|\rho(t)|+\rangle = \frac{1}{2} + \frac{1}{2}e^{-\gamma t}\cos(wt)$$

This is referred to as *decoherence* of the system, and the value of γ relies on the physical mechanism that leads to the decoherence.
To address the issue of decoherence, *quantum error correction* is a crucial concept that must be incorporated. Quantum error correction relates to a set of techniques that protects information in quantum computing by encoding it across several physical qubits to create a *logical qubit*. The concept of a logical qubit refers to a theoretical unit of quantum information that a quantum processor can manipulate. The number of logical qubits that a quantum processor has determines its computing capacity. In other words, the more logical qubits a quantum processor has, the more computational power it has. This method is considered essential for developing a large-scale quantum computer that can perform useful calculations with low error rates [17]. Current research efforts are aimed at enhancing the quality of qubits and increasing the isolation of quantum systems, with the objective of utilizing fewer physical qubits to construct a logical qubit.
Given the challenges posed by decoherence and the need for quantum error correction, it is essential to consider the requirements for a physical system to be suitable for use as a quantum computer. In 2000, David DiVincenzo proposed a set of necessary conditions for such a system, which are now commonly referred to as the "DiVincenzo criteria" [18]. These requirements include:

1. *A scalable physical system with well-characterized qubits:* the physical system must be capable of adding more qubits in a way that does not compromise the performance of the system. Additionally, the qubits must be well-characterized, meaning their physical parameters should be accurately known, including their internal Hamiltonian (defining the qubit energy eigenstates), the presence of and couplings to other states, and couplings to external fields (necessary to manipulate the qubit state) and to other qubits in the system (necessary for implementing multi-qubit gates).

2. *The ability to initialize the state of the system to a simple fiducial state:* the system must be able to be initialized to a known state, typically an all-zero state, before the start of the computation.

3. *Long relevant decoherence times:* the physical system must be able to maintain coherence for a sufficiently long time, much longer than the gate operation time, to ensure that the quantum features of computation can be utilized.

4. *Capability to implement a universal set of quantum gates:* the system must be able to implement a universal set of quantum gates, which can be used

to construct any quantum algorithm. This typically includes single-qubit gates and a single type of two-qubit gate, which is native to a particular implementation (e.g., *CNOT, CPHASE, or XY).*

5. *A qubit-specific measurement capability:* the system must be able to measure specific qubits, which is necessary for the efficient execution of quantum algorithms.

Despite the impressive advances of quantum computing technology over the last several years, achieving sufficiently fault-tolerant quantum computers is still a few years away. This is why it is useful (and necessary) to utilize *quantum simulators*, which are classical computers that operate based on the principles of quantum computing. The only hindrance for classical computers to operate in this way is the memory requirements, as the state of an $n-$qubit quantum system can be stored in classical memory as 2^n probability amplitudes. Due to this limitation, most quantum simulators are typically constrained to operate on a limited number of qubits, usually ranging from a few to a dozen. Nonetheless, these simulators play a vital role in providing an ideal computing environment, free from quantum hardware imperfections. As a result, they are crucial for testing fundamental principles and small-scale versions of quantum algorithms.
In the following, we will discuss three primary physical realizations of quantum computers that satisfy the DiVincenzo criteria, namely superconducting qubits, photonic qubits, and trapped ions qubits. We will provide a brief overview of their fundamental principles, advantages, and challenges, as well as mention some notable companies working on these technologies.

2.9 Physical Realizations of Quantum Computers

2.9.1 Superconducting Qubits

Superconducting qubits are based on the manipulation of quantized energy levels in superconducting electrical circuits. These circuits are made of materials that exhibit zero electrical resistance below a certain critical temperature. The basic building block of a superconducting qubit is the Josephson junction, which consists of two superconductors separated by a thin insulating barrier. The non-linear inductive behavior of the Josephson junction allows for the creation of quantized energy levels that can be used to represent the quantum states. Some popular superconducting qubit designs include:

- Transmon: a variant of the Cooper-pair box qubit, the transmon qubit features a large shunting capacitance to suppress the charge noise, while still maintaining sufficient anharmonicity to allow for selective qubit state manipulation.

- Flux Qubit: this qubit design is based on a superconducting loop interrupted by one or more Josephson junctions. The qubit states are defined by the quantized magnetic flux in the loop, and the qubit can be controlled by applying an external magnetic field.

- Xmon: the Xmon qubit is a modification of the transmon design, with the addition of an extra coupling capacitor that allows for stronger qubit-qubit interactions in a scalable architecture.

Superconducting qubits represent a mature and promising technology for addressing the scalability challenge. These qubits are kept in a cryogenic dilution fridge operating at millikelvin temperatures, providing isolation from thermal and electrical noise. Control of these qubits requires a multitude of complex wiring and cables, which are connected to a rack-full of external electronics. While it is possible to fabricate thousands of qubits on a single chip and cool them down to extremely low temperatures, the major hurdle lies in scaling up the control and readout electronics [19]. This difficulty arises primarily from the limited cooling power available in cryogenic systems, which imposes constraints on the wiring capacity and management of cabling heat load. Another challenge associated with superconducting qubits is their limited connectivity, which poses difficulties in implementing complex quantum algorithms and performing multi-qubit operations efficiently.
IBM and Google are two prominent companies that have significantly contributed to the development and application of quantum computing technologies, particularly in the field of superconducting qubits. IBM has been at the forefront of this research, unveiling the IBM Quantum System Two in 2022, which offers improved performance and scalability with its advanced superconducting qubits [20]. This system features a sophisticated design that combines advanced cryogenics, electronics, and shielding to minimize the effects of noise and decoherence. IBM's continuous improvement of their quantum systems is evidenced by their roadmap, which aims to achieve a 1,121-qubit system, named "Condor," by 2023 [21]. Google, another key player in quantum computing, has continued to make strides with their superconducting qubit technology. By demonstrating a quantum advantage with error-correction techniques such as the surface code, Google has further established the potential of superconducting qubits in quantum computing [22]. Moreover, Google unveiled their ambitious plan to build a practical, error-corrected quantum computer, the "Quantum Era", by 2029 [23]. This plan includes a series of engineering milestones aimed at improving error rates, qubit counts, and connectivity.

2.9.2 Photonic Qubits

Photonic qubits represent another promising avenue for the realization of quantum computers, utilizing the quantum properties of photons as the basis for encoding and manipulating quantum information. The main advantage of photonic quantum computing lies in the inherent robustness of photons against decoherence, making them less susceptible to environmental noise compared to other physical implementations of qubits. One of the primary approaches to photonic quantum computing is the linear optical quantum computing (LOQC) model, which relies on linear optical elements such as beam splitters, phase shifters, and single-photon sources to manipulate and process photonic qubits [24]. In recent years, integrated quantum photonics has emerged as a scalable approach to implement LOQC, leveraging advances in the fabrication of photonic integrated circuits to miniaturize and increase the complexity of quantum photonic devices [25].
Xanadu [26], a Canadian company, is a notable pioneer in the field of photonic quantum computing. They have developed a unique approach to quantum computing based on continuous-variable (CV) quantum information processing, which employs the quantum states of light, such as quadrature amplitudes, to encode and manipulate quantum information. Moreover, PsiQuantum, a US-based company, is working on building a large-scale, fault-tolerant photonic quantum computer using a silicon photonics platform [27]. They aim to leverage the mature silicon photonics fabrication technology to manufacture photonic qubits and the required quantum gates, which could enable large-scale, error-corrected quantum computing.

2.9.3 Trapped Ion Qubits

Trapped ion technology has emerged as another intriguing alternative for the physical realization of qubits and quantum gates. In this approach, individual ions are trapped using electromagnetic fields and manipulated using laser beams to perform quantum operations. The advantages of trapped ion qubits include their long coherence times, high-fidelity quantum gates, and potential for scalability. However, the technology also faces challenges such as the complexity of the required control systems and the need for efficient ion transport and reordering.
One of the leading companies working on trapped ion quantum computing is IonQ. The company has developed several generations of trapped ion quantum computers, demonstrating consistent improvements in qubit count and overall system performance. In 2023, they announced the release of their fifth-generation quantum computer, IonQ Forte, featuring 32 fully connected qubits and a quantum volume of over 4 million [28].

2.10 Conclusions

In this chapter, we laid the groundwork for understanding quantum computing by discussing its essential concepts, components, and techniques. We began with an introduction to the fundamental building block of quantum computing, the qubit, and its canonical mathematical representation. We then discussed multiple qubits and their interactions. The density operator was also introduced, allowing us to describe both pure and mixed quantum states, in contrast to the state vector that can only represent pure quantum states. We continued with the exploration of the measurement of quantum states and the time-evolution of closed systems, highlighting the importance of unitary operators. Next, we delved into quantum gates, starting with single-qubit gates and their representation as rotation operators on the Bloch sphere. We proceeded to examine two-qubit gates and their matrix representations, as well as quantum gate decompositions. We also covered the topic of entanglement, a distinctive feature of quantum systems that enables powerful computational capabilities. Subsequently, we outlined the primary physical realizations of quantum computers, focusing on superconducting qubits, photonic qubits, and trapped ion qubits, along with the rapid advancements in each of these areas.
In the next chapter, we will present a specific kind of quantum circuit, known as Parameterized Quantum Circuits (PQCs), which offer considerable versatility for a range of applications. Furthermore, we will investigate different data encoding methods – the process of transforming samples from classical datasets into their respective quantum states.

Chapter 3

Parametrized Quantum Circuits and Data Encoding

In recent years, Parameterized Quantum Circuits (PQC), also known as Variational Quantum Circuits, have emerged as a versatile and powerful tool in the development of quantum algorithms. These circuits are characterized by their ability to adapt and evolve through the adjustment of tunable parameters, making them particularly suitable for optimization and machine learning tasks. In this chapter, we will explore the fundamental concept of PQC and provide a comprehensive description of two prominent PQC-based algorithms, namely the Variational Quantum Eigensolver (VQE) [29] and the Quantum Approximate Optimization Algorithm (QAOA) [30]. These algorithms have garnered significant attention due to their ability to leverage NISQ devices for solving complex optimization problems. Additionally, this chapter will introduce data encoding schemes, which play a crucial role in translating classical data into quantum states. This allows for an integration of the quantum and classical computing paradigms, further expanding the potential applications of PQC in various domains.

3.1 Parametrized Quantum Circuits

As we discussed in the previous chapter, quantum gates can be combined to create quantum circuits of varying depth. Notably, these circuits apply a sequence of unitary operators $U_i(\theta_i)$ that transform an initial $n-$qubit state $|\psi_{in}\rangle$ into a final quantum state $|\psi\rangle$:

$$|\psi\rangle = U_m(\theta_m) \dots U_1(\theta_1)|\psi_{in}\rangle \tag{3.1}$$

The unitary operators U_i correspond to individual gates, each with an associated vector of gate parameters θ_i, for $i = 1, ..., m$. Certain gates in a quantum circuit may

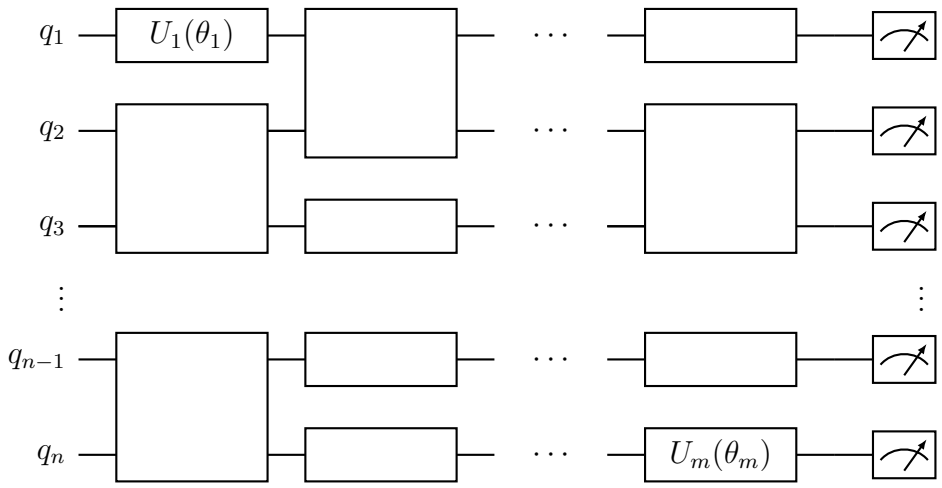


Figure 3.1: Graphical representation of a generic Parametrized Quantum Circuit (PQC) acting on n qubits.

have *fixed* parameters, meaning that the operation performed by the gate cannot be changed. For instance, a two-qubit CZ gate can be considered a controlled rotation of the target qubit state around the $z-$axis by a fixed angle of $\theta = \pi$. Other gates, however, may have *adjustable* parameters that can be tuned to change the operation performed by the gate. For example, a one-qubit $R_y(\theta)$ gate can rotate the qubit state around the $y-$axis by an arbitrary angle $\theta \in [-\pi, \pi]$.
After constructing the final quantum state $|\psi\rangle$, the qubits can be measured, yielding a classical bitstring. We do not always need information about the entire quantum system. In discriminative models, only a single qubit or a few qubits can provide sufficient information about the binary representation of the class label for a given sample. Conversely, in generative models, the information relative to all the qubits is necessary to obtain a bitstring representing a generated sample from the probability distribution encoded in the final quantum state $|\psi\rangle$.
This is a general description of a PQC, which is illustrated schematically in Figure 3.1. One of the most important aspects of a PQC is that it can be *trained* to find an optimal set of adjustable parameters (such as the vectors $\theta_1, ..., \theta_m$ above) based on the overall PQC architecture, known as *ansatz*. The meaning of "optimal" may vary depending on the problem being solved, but generally refers to a configuration of the parameters that leads to the closest possible match between the final quantum state $|\psi\rangle$ and a desired target quantum state corresponding to a specific probability distribution that needs to be encoded. When dealing with quantum machine learning use cases, the process of determining the optimal configuration of PQC parameters is commonly referred to as *learning*. This can be achieved through differentiable or non-differentiable methods and always involves minimizing some

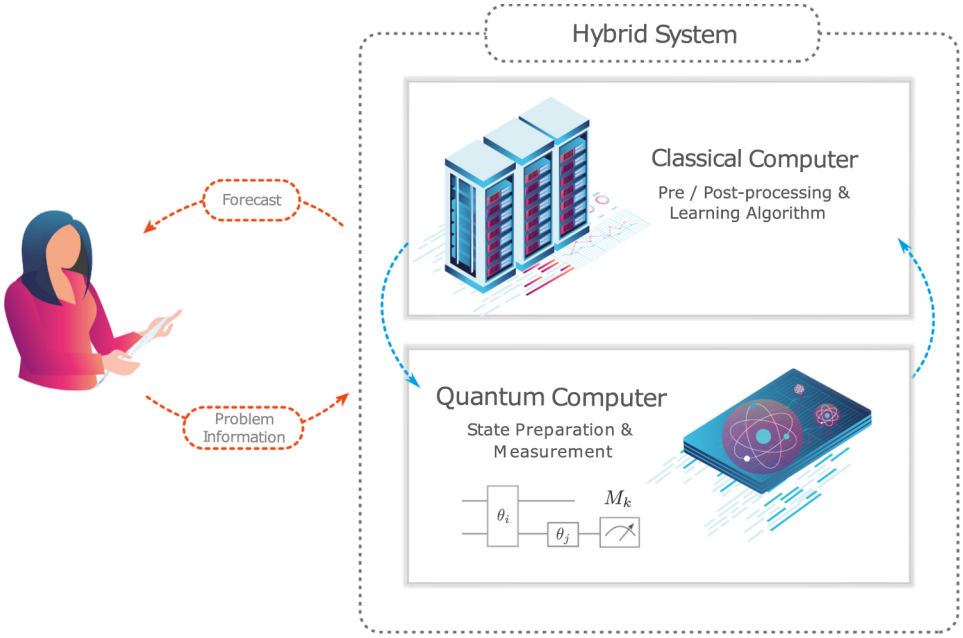


Figure 3.2: Illustration of the hybrid quantum-classical algorithm used to train PQCs, reproduced from [31].

cost function by adjusting the circuit parameters.
Parametrized Quantum Circuits are typically trained through a *hybrid quantum-classical paradigm*, which is schematically represented in Figure 3.2. This approach involves three components: the user, a classical computer, and a quantum computer [31]. The user provides the model for the problem to be solved along with the input training data, while the classical computer pre-processes the initial data and generates the initial set of parameters for the PQC. The quantum computer executes the PQC by setting up the quantum state in accordance with the PQC instructions and by performing measurements. The measurement outcomes are then post-processed by the classical computer to evaluate the cost function. To improve the cost, the classical computer updates the model parameters according to the chosen learning algorithm (e.g., backpropagation of error with gradient descent [32]). The overall algorithm is therefore run in a closed loop between the classical and quantum hardware.

In conclusion, the PQC framework is widely considered to be a promising approach for utilizing NISQ devices to solve complex real-world problems. It employs a hybrid quantum-classical computational protocol and can be applied to experiment with various quantum machine learning models.

3.2 Variational Quantum Eigensolver

The Variational Quantum Eigensolver (VQE) is a PQC-based algorithm that seeks to determine the smallest eigenvalue (i.e., the lowest energy) of a Hamiltonian. The Hamiltonian of a quantum system entirely describes its dynamics, as we discussed in section 2.5. Therefore, if we are able to encode the objective function we want to minimize in the Hamiltonian of a quantum system, then finding the ground state of the Hamiltonian is equivalent to finding the set of decision variables that minimizes the objective function. The VQE was first developed in 2014 by [29] and has become a popular tool for solving optimization problems on NISQ devices. The algorithm uses a *variational approach* to systematically search for the best possible approximation of the ground state by exploring various PQC ansatzes and configurations of adjustable PQC parameters.
Let us now take a closer look at how the VQE algorithm works. Given a Hamiltonian H, its characteristic equation is given by

$$H\left|\psi_i\right\rangle = \mathcal{E}_i\left|\psi_i\right\rangle \tag{3.2}$$

where $\mathcal{E}_i$ is an eigenvalue corresponding to the eigenstate $|\psi_i\rangle$. The goal is to determine the smallest eigenvalue $\mathcal{E}_0$ of H that corresponds to the ground state $|\psi_0\rangle$. If the latter were known, such a task would be straightforward, as the eigenvalue of H in a given eigenstate is nothing but the expectation value of H in that eigenstate:

$$\langle\psi_i|H|\psi_i\rangle = \langle\psi_i|\,\mathcal{E}_i\,|\psi_i\rangle = \mathcal{E}_i\,\langle\psi_i|\psi_i\rangle = \mathcal{E}_i \tag{3.3}$$

However, in practice, the ground state $|\psi_0\rangle$ is often not known, and the aim is to find the ground state that encodes the solution of the optimization problem by minimizing the expectation value of H. In this case, we can construct a series of increasingly better approximations to the ground state, resulting in a increasingly accurate upper bound for the ground state energy $\mathcal{E}_0$.
The variational approach is based on the spectral theorem, which allows the expansion of the Hermitian Hamiltonian H in terms of its eigenstates and eigenvalues as:

$$H = \sum_i \mathcal{E}_i\,|\psi_i\rangle\langle\psi_i| \tag{3.4}$$

Let us suppose we constructed a state $|\psi\rangle$ that approximates the actual ground state $|\psi_0\rangle$. By considering the expression of H given by (3.4), we can express the expectation value of H in state $|\psi\rangle$ as

$$\begin{aligned}\langle\psi|H|\psi\rangle &= \langle\psi|\left(\sum_i \mathcal{E}_i\,|\psi_i\rangle\langle\psi_i|\right)|\psi\rangle \\ &= \sum_i \mathcal{E}_i\,\langle\psi|\psi_i\rangle\,\langle\psi_i|\psi\rangle = \sum_i \mathcal{E}_i\,|\langle\psi|\psi_i\rangle|^2\end{aligned} \tag{3.5}$$

Equation (3.5) demonstrates that the expectation value of H in any state $|\psi\rangle$ can be written as a linear combination of the eigenvalues of H, with all weights being non-negative since $|\langle\psi|\psi_i\rangle|^2 \geq 0$ for every i. Consequently, we can conclude that

$$\langle\psi|H|\psi\rangle \geq \mathcal{E}_0 \tag{3.6}$$

since $\mathcal{E}_0$ is the smallest eigenvalue of H and all the weights in (3.5) are non-negative. The PQC is responsible for generating the candidate state $|\psi\rangle$, while the variational aspect of the algorithm involves incrementally improving this candidate state through iterative updates of the adjustable parameters. This process is carried out as a classical part of the hybrid quantum-classical protocol. The quantum component of the algorithm involves executing the PQC and measuring H on the resulting quantum state to determine the expectation value of H.
In the context of variational quantum eigensolvers, a crucial issue pertains to the implementation of H as an observable, since this may require an infeasible amount of measurements for Hamiltonians that are not simple to handle. Fortunately, for many practical scenarios, H can be expressed as a linear combination of weighted local (i.e., 1- or 2- qubit) observables H_j

$$H = \sum_{j=1}^{J} h_j H_j \tag{3.7}$$

with $h_j \in \mathbb{R}\ \forall j = 1, \ldots, J$. Actually, it is always possible to express the Hamiltonian as a sum of Pauli operators [29]

$$H = \sum_{i\alpha} h^i_\alpha \sigma^i_\alpha + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \sigma^i_\alpha \sigma^j_\beta + \ldots \tag{3.8}$$

for real h, where the superscripts $i, j, \ldots$ runs over the qubits that the Pauli operator acts on, while the subscripts $\alpha, \beta, \ldots$ identify the Pauli operator (with σ_1 being the identity operator). For instance, $i = 2$, $\alpha = y$, and $\sigma^2_y = Y$ acting on qubit 2. The validity of this expansion does not rely on any assumptions regarding the dimension or structure of the Hermitian Hamiltonian. We can utilize the property of linearity of quantum observables to compute the expectation value of the Hamiltonian by summing up the expectations of the individual terms:

$$\langle H\rangle = \sum_{i\alpha} h^i_\alpha \langle\sigma^i_\alpha\rangle + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \langle\sigma^i_\alpha \sigma^j_\beta\rangle + \ldots \tag{3.9}$$

This representation allows for efficient estimation of the energy expectation if the Hamiltonian can be expressed as a sum of only a few terms, each consisting of only a few Pauli operators. This is often the case in quantum chemistry where Hamiltonians describe electronic systems under the Born-Oppenheimer approximation, as well as

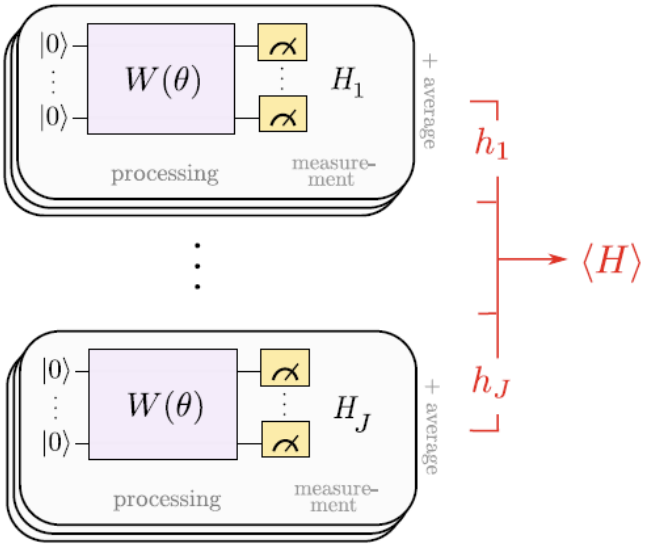


Figure 3.3: Principle of VQE, reproduced from [34]. The variational quantum eigensolver operates by using the Hamiltonian of the system as the observable. In many cases, the Hamiltonian H can be expressed as a linear combination of local Hamiltonian terms H_j that can be measured individually and then summed up classically to obtain the overall expectation value $\langle H \rangle$. The VQE uses this approach to iteratively improve the candidate state $|\psi(\theta)\rangle$ prepared by the parametrized ansatz $W(\theta)$ and obtain the ground state energy of the system.

in the well-known Ising and Heisenberg models [33].
Using this decomposition, the overall cost $C(\theta)$ is given by a sum of estimates of local expectation values, i.e.

$$C(\theta) = \sum_{j=1}^{J} h_j \langle \psi(\theta) | H_j | \psi(\theta) \rangle \tag{3.10}$$

where θ denotes the set of all PQC parameters. The local estimates $\langle \psi(\theta) | H_j | \psi(\theta) \rangle$ are multiplied by the coefficients h_j and summed up on the classical device. This is schematically shown in Figure 3.3. If the number of local terms in the objective function is relatively small and scales polynomially with the number of qubits, then the process of estimating the energy expectation via quantum measurements can be done in a qubit-efficient manner.
Finally, it is worth mentioning that the task of finding the lowest energy state of an Ising-type Hamiltonian can also be efficiently tackled on a *quantum annealer* [35], which is based on *adiabatic quantum computing* (AQC). Notably, the gate model of quantum computing involves applying a series of unitary gates to a set of qubits, which are then measured at the end of the computation. On the other hand,

AQC encodes the solution of the optimization problem in the ground state of a Hamiltonian that defines the dynamics of a $n-$qubit system. The process involves starting with a quantum system in the ground state of a simpler Hamiltonian that can be easily realized experimentally, and then gradually adjusting the system so that it is controlled by the desired Hamiltonian, in such a way that the system likely ends up in its ground state.
Adiabatic quantum computing has been shown to be polynomially equivalent to gate-based universal quantum computing, as any quantum circuit can be represented as a time-dependent Hamiltonian with at most polynomial overhead [36]. Both approaches have their strengths and weaknesses, and a careful analysis is required to determine which approach is better suited for a particular problem.

3.3 Quantum Approximate Optimization Algorithm

The Quantum Approximate Optimization Algorithm (QAOA) is another instance of a hybrid quantum-classical approach for tackling combinatorial optimization problems, and was introduced in 2014 by [30]. It is inspired by and incorporates the ideas of two previously mentioned optimization algorithms, namely AQC and VQE. Specifically, QAOA uses a parametrized quantum circuit to encode the objective function of an optimization problem into a Hamiltonian, which is then evolved adiabatically to find the ground state. Similar to VQE, a classical optimization algorithm is used to iteratively update the parameters of the quantum circuit to minimize the expectation value (i.e., the energy) of the evolved Hamiltonian. However, unlike VQE, the QAOA algorithm is not designed to find the exact ground state of the Hamiltonian but rather a good approximation of it.
To see this, let us consider again the Schrödinger equation (2.21) introduced in Chapter 2, which describes the time evolution of a quantum state $|\psi(t)\rangle$ under a *time-independent* Hamiltonian H:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = H|\psi(t)\rangle$$

Given some initial condition $|\psi(0)\rangle$, its solution is given by

$$|\psi(t)\rangle = \mathcal{U}(0,t)|\psi(0)\rangle$$

where $\mathcal{U}(0,t)$ is the time evolution operator obtained from the Hamiltonian H as

$$\mathcal{U}(0,t) = \exp\left(-\frac{iHt}{\hbar}\right)$$

For the sake of simplicity, we use units where the value of $\hbar$ is set to 1; consequently, the system dynamics can be expressed as

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle \tag{3.11}$$

If we know the initial state $|\psi(0)\rangle$ of the system, then the state at any time t can be determined by applying the Hamiltonian H over the time period t.
In contrast to the time-independent Hamiltonian used in VQE, the dynamics of AQC involves *time-dependent* Hamiltonians of the form

$$H(t) = \left(1 - \frac{t}{T}\right) H_0 + \frac{t}{T} H_F \tag{3.12}$$

where H_0 is the *initial* Hamiltonian and H_F is the *final* or *problem* Hamiltonian that encodes the optimization problem. The solution to this mismatch is to approximate the time-dependent Hamiltonian $H(t)$ over the time interval $[0, T]$ using a sequence of time-independent Hamiltonians $H_1, H_2, ..., H_m$, each transforming the state over the shorter time intervals $[t_0 = 0, t_1], [t_1, t_2], \ldots, [t_{m-1}, t_m = T]$.
In the same way, we can approximate the operator $\mathcal{U}(0, T)$ as

$$\mathcal{U}(0, T) \approx \mathcal{U}(t_{m-1}, t_m)\mathcal{U}(t_{m-2}, t_{m-1}) \ldots \mathcal{U}(t_2, t_1)\mathcal{U}(t_0, t_1) \tag{3.13}$$

Clearly, the quality of the approximation improves as the time intervals $[t_{i-1}, t_i]$ become more fine-grained.
The Suzuki-Trotter expansion [37] provides a useful way to approximate the operator $\mathcal{U}(0, T)$. If we have operators $\mathcal{A}_1, \mathcal{A}_2, ..., \mathcal{A}_k$ (that do not necessarily commute), then we can use the expansion:

$$\exp\left(\mathcal{A}_1 + \mathcal{A}_2 + ... + \mathcal{A}_k\right) = \lim_{m\to\infty} \left[\exp\left(\frac{\mathcal{A}_1}{m}\right) \exp\left(\frac{\mathcal{A}_2}{m}\right) \ldots \exp\left(\frac{\mathcal{A}_k}{m}\right)\right]^m \tag{3.14}$$

Hence, if $\mathcal{U}(0, T)$ is expressed as $\exp\left([\mathcal{B} + \mathcal{C}]\, T\right)$, it is possible to use the Suzuki-Trotter expansion to get

$$\exp\left([\mathcal{B} + \mathcal{C}]\, T\right) = \lim_{m\to\infty} \left[\exp\left(\frac{\mathcal{B}T}{m}\right) \exp\left(\frac{\mathcal{C}T}{m}\right)\right]^m$$

This means that instead of evolving the system with the Hamiltonian $[\mathcal{B} + \mathcal{C}]$ over a time interval T, we can break the time interval into m smaller subintervals of length T/m and apply the operators $\mathcal{B}$ and $\mathcal{C}$ alternately over these subintervals to approximately evolve the system over the whole time interval.
In the context of AQC, the general form of Hamiltonians H_0 and H_F is given by

$$H_0 = \sum_{i=1}^{n} \sigma_x^i, \quad H_F = \sum_{i=1}^{n} a_i \sigma_z^i + \sum_{i=1}^{n} \sum_{j=1+1}^{n} b_{ij} \sigma_z^i \sigma_z^j \tag{3.15}$$

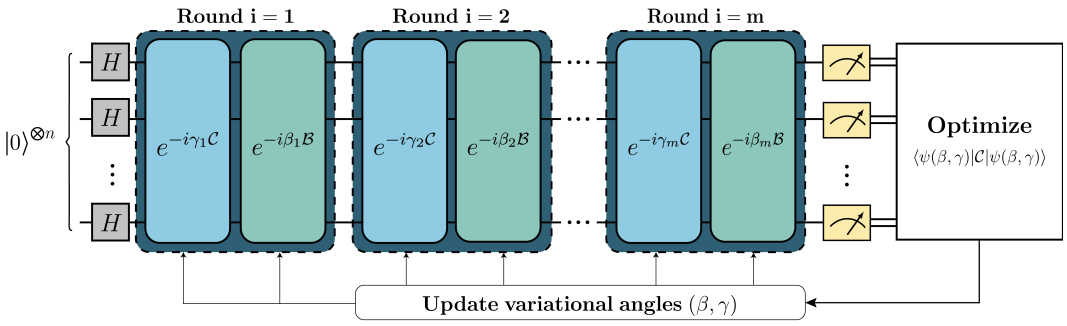


Figure 3.4: Principle of QAOA. In QAOA, the observable is the phase Hamiltonian $\mathcal{C}$ of a combinatorial optimization problem. The algorithm involves alternately applying a sequence of two Hamiltonians ($\mathcal{B}$ and $\mathcal{C}$) to an initial state that is an equal superposition of n qubits. The qubit states are then measured and used to calculate a cost, which is then minimized using a classical optimization algorithm by varying the angles β and γ.

for some coefficients a_i and b_{ij}, $i, j = 1, \ldots, n$. The operator $\mathcal{B}$, also known as the *mixing Hamiltonian*, corresponds to the initial Hamiltonian H_0, whereas the operator $\mathcal{C}$, also called the *phase Hamiltonian*, corresponds to the final Hamiltonian H_F.
The initial state $|\psi(0)\rangle$ is prepared in an equal superposition of all possible solutions, and can thus be expressed as:

$$|\psi(0)\rangle = \frac{1}{\sqrt{2^n}}\left(|0\ldots00\rangle + |0\ldots01\rangle + \ldots + |1\ldots11\rangle\right) = \frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}|i\rangle$$

Such a state corresponds to the ground state of the operator $\mathcal{B}$ and can be easily constructed from the all-zero state by applying the Hadamard gate H to each qubit, i.e.

$$|\psi(0)\rangle = H^{\otimes n}|0\rangle^{\otimes n} \tag{3.16}$$

Figure 3.4 shows a schematic illustration of the QAOA procedure, which we briefly summarize below:

1. Given the initial state $|\psi(0)\rangle = H^{\otimes n}|0\rangle^{\otimes n}$, a parametrized quantum state $|\psi(\beta,\gamma)\rangle$ is constructed by alternately applying the operators $\mathcal{B}$ and $\mathcal{C}$, where $\beta \doteq (\beta_1, \ldots, \beta_m)$ and $\gamma \doteq (\gamma_1, \ldots, \gamma_m)$ are vectors of m variational parameters that determine the duration of each round:

$$|\psi(\beta,\gamma)\rangle = e^{-i\beta_m\mathcal{B}}e^{-i\gamma_m\mathcal{C}}\ldots e^{-i\beta_1\mathcal{B}}e^{-i\gamma_1\mathcal{C}}\left(H^{\otimes n}|0\rangle^{\otimes n}\right)$$

2. The final quantum state $|\psi(\beta,\gamma)\rangle$ is then measured in the computational basis (z-basis) to obtain a candidate solution that is a binary string. The above

state preparation and measurement are repeated multiple times to obtain a set of binary strings. The expectation value $\langle\psi(\beta,\gamma)|\mathcal{C}|\psi(\beta,\gamma)\rangle$ of the operator $\mathcal{C}$, which corresponds to the cost function to be minimized, is then evaluated using the obtained set of candidate solutions.

3. The variational parameters β and γ are updated based on the outcome of the previous step, using a classical optimization algorithm. The procedure is repeated for multiple iterations until convergence is achieved, resulting in the optimal set of parameters that provide the best approximate solution to the optimization problem.

Finally, it is worth noting that applying operators $e^{-i\beta\mathcal{B}}$ and $e^{-i\gamma\mathcal{C}}$ alternatively is essential to avoid getting trapped in a local minimum. It is also crucial to ensure that $\mathcal{B}$ and $\mathcal{C}$ do not commute [38]. Notably, the application of only $e^{-i\gamma\mathcal{C}}$ can result in an eigenstate of the phase Hamiltonian, leading to being stuck there. This is because any further application of a linear operator to its eigenvector could only change its magnitude but not its direction. By the same token, if two *commuting* operators are applied alternately, we could get stuck in an eigenstate of both operators. However, by using non-commuting operators such as σ_x and σ_z, there is always a chance to escape from the local minimum and find the global optimal solution.
QAOA has been applied to various optimization problems, such as MaxCut [39] and Graph Coloring [40], and has shown promising results on near-term quantum devices.

3.4 Data Encoding Strategies

In the context of quantum machine learning, data encoding strategies represent a fundamental component for a variety of quantum algorithms. As a matter of fact, in order to utilize quantum computers to learn from classical data, it is necessary to represent such data using quantum states. In classical quantum computing, preparing the initial state of the quantum computer, which encodes the input to the algorithm, is known as *state preparation*. In quantum machine learning, however, data encoding is an essential step that goes beyond just preparing the algorithm. In fact, it plays a critical role in the runtime of the algorithm and can often be the bottleneck. Therefore, it is crucial to have a good understanding of different encoding strategies and their advantages and drawbacks. In this section, we will introduce some relevant schemes for encoding data into an $n-$qubit system, including *basis encoding, amplitude encoding, angle encoding,* and *Hamiltonian encoding.*

3.4.1 Basis Encoding

Basis encoding is commonly used in quantum algorithms where arithmetic manipulation of real numbers is required [41]. Essentially, it links a classical $n-$bit string (for example, 0100) with a computational basis state of an $n-$qubit system ($|4\rangle = |0100\rangle$). This encoding strategy is considered the simplest since each bit is directly substituted by a qubit, and computation operates in parallel on all bit sequences in a superposition.
Let us consider a real number $x \in \mathbb{R}$ approximated by a $\tau-$bit string according to

$$\begin{aligned} x &\approx (-1)^{b_s} \left(\sum_{j=0}^{\tau_i - 1} b_j 2^j + \sum_{j=1}^{\tau_f} b_{-j} 2^{-j} \right) \\ &\mapsto |b_s b_{\tau_i - 1} \dots b_0 b_{-1} \dots b_{-\tau_f}\rangle \doteq |x\rangle \end{aligned}$$

where $\tau = (1 + \tau_i + \tau_f)$ and $b_s, b_j, b_{-j} \in \{0, 1\}$. The first bit b_s accounts for the sign of x, while the remaining bits are divided into two parts: τ_i bits to the left of the decimal point (*integer bits*) and τ_f bits to the right of the decimal point (*fractional bits*). Let us now consider a vector $\mathbf{x} = (x^1, \dots, x^N) \in \mathbb{R}^N$, where N denotes the number of features. We can concatenate the binary approximations of each feature x^i into a vector

$$\left(b_s^1, b_{\tau_i - 1}^1, \dots, b_{-\tau_f}^1, \dots, b_s^N, b_{\tau_i - 1}^N, \dots, b_{-\tau_f}^N\right) \in \{0,1\}^{\tau N}$$

which can then be used to derive a quantum state representation of $\mathbf{x}$ with τN qubits of the form

$$|b_s^1 b_{\tau_i - 1}^1 \dots b_{-\tau_f}^1 \dots b_s^N b_{\tau_i - 1}^N \dots b_{-\tau_f}^N\rangle$$

For instance, if we use a binary representation with a precision of $\tau_i = 2$ and $\tau_f = 3$, we can represent the input vector $\mathbf{x} = (-2.625, 0.375)$ as $\mathbf{x} = (110.101, 000.110)$. Then, we concatenate these bit strings to obtain the binary sequence 110101000110 of length τN that can be represented by the $n-$qubit quantum state $|\mathbf{x}\rangle = |110101000110\rangle$. The process of preparing such a state is relatively straightforward, as it only involves flipping the qubits that correspond to non-zero bits. Hence, the encoding circuit is created by applying the NOT gate X^{b_i} on qubit i:

$$|0\rangle^{\otimes \tau N} \mapsto \bigotimes_{i=1}^{\tau N} X^{b_i} |0\rangle^{\otimes \tau N}$$

Although the algorithm is simple and involves only the use of the single-qubit gate X, it demands a substantial amount of qubits, particularly when encoding data with high accuracy, and is not generally efficient in real-world applications.
Let us now consider a dataset $\mathcal{D} = (\mathbf{x}^1, \dots, \mathbf{x}^M)$, where each data point $\mathbf{x}^m \in \mathcal{D}$ is expressed using a binary string of the form $b^m = (b_1^m, ..., b_n^m) \in \{0,1\}^n$, $m =$

$1, \ldots, M$. The binary encoding of $\mathcal{D}$ can be obtained by preparing a uniform superposition of basis states $|\mathbf{x}^m\rangle$, where each basis state corresponds qubit-wise to the binary representation of the corresponding input data point:

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |\mathbf{x}^m\rangle$$

To illustrate with an example, let us consider two binary inputs $\mathbf{x}^1 = (01,\, 11)^T$ and $\mathbf{x}^2 = (10,\, 01)^T$, where each feature is represented using a binary precision of $\tau = 2$. We can express them as binary strings $b^1 = (0111)$ and $b^2 = (1001)$, respectively. These strings can then be associated with the corresponding basis states, namely $\mathbf{x}^1 = |0111\rangle$ and $\mathbf{x}^2 = |1001\rangle$, and the complete data superposition can be obtained as

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{2}}|0111\rangle + \frac{1}{\sqrt{2}}|1001\rangle = \frac{1}{\sqrt{2}}|7\rangle + \frac{1}{\sqrt{2}}|9\rangle$$

The vector of probability amplitudes related to the state $|\mathcal{D}\rangle$ have components equal to $\frac{1}{\sqrt{M}}$ for the basis states that correspond to a binary instance in $\mathcal{D}$, while all other components are zero. In our example, since we have a $4-$qubit system and only two basis states corresponding to binary instances in $\mathcal{D}$, the amplitude vector (in the computational basis) will be given by

$$\Big(0,\, 0,\, 0,\, 0,\, 0,\, 0,\, 0,\, \frac{1}{\sqrt{2}},\, 0,\, \frac{1}{\sqrt{2}},\, 0, 0,\, 0,\, 0,\, 0,\, 0\Big)^T$$

From this, we can notice that basis encoding of datasets typically results in sparse amplitude vectors, with a much smaller number of non-zero amplitudes (M) compared to the total number of amplitudes (2^n).

3.4.2 Amplitude Encoding

When quantum algorithms do not require arithmetic manipulation of data, more compact data representations can be used to take advantage of the large Hilbert space of a quantum device. One such technique is amplitude encoding, which relates classical information, represented by a real or complex vector, with the probability amplitudes of a quantum system. Given an $N-$dimensional data point $x = (x_1, \ldots, x_N) \in \mathbb{C}^N$, with $N = 2^n$, we can associate quantum amplitudes to the components of x according to

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} \mapsto |\psi_x\rangle = \frac{1}{||x||} \sum_{i=1}^{N} x_i \, |i\rangle$$

where $||x|| = \sum_{i=1}^{N} |x_i|^2$ is the normalization constant. For example, to apply amplitude encoding to the vector $x = (0.3, -0.8, 1.0)$, we first normalize it to unit norm and then pad it with zeros to reach a dimension of integer logarithm. The resulting vector (rounding to three digits) is therefore

$$x = (0.228, -0.608, 0.760, 0.000)$$

This normalized vector can now be encoded in the probability amplitudes of a 2-qubit quantum state as:

$$|\psi_x\rangle = 0.228\,|00\rangle - 0.608\,|01\rangle + 0.760\,|10\rangle + 0\,|11\rangle$$

It is also possible to use amplitude encoding to encode an entire dataset $\mathcal{D} = (x^1, \ldots, x^M)$ consisting of M patterns in $\mathbb{R}^N$ in superposition as

$$|\psi_{\mathcal{D}}\rangle = \frac{1}{C_{\mathcal{D}}} \sum_{i=1}^{2^p} \bar{x}_i |i\rangle$$

for some integer p, where

$$\bar{x} = (\bar{x}_i)_{i=1,\ldots,2^p} = (x_1^1, \ldots, x_N^1, x_1^2, \ldots, x_N^2, \ldots, x_1^M, \ldots, x_N^M)^T \in \mathbb{R}^{MN}$$

is the concatenation of all the patterns in $\mathcal{D}$ and $C_{\mathcal{D}}$ is a normalization factor. The integer p is chosen such that $2^p \geq MN$, or equivalently, $p \geq \log_2(MN)$. However, there may be some sparsity in the amplitude vector if $2^p \geq MN$.
Amplitude encoding allows n quantum registers to hold up to 2^n continuous features. Nonetheless, achieving this requires the creation of deep quantum circuits, which may not be feasible for NISQ devices when working with a large number of features.

3.4.3 Angle Encoding

Angle encoding is a method that utilizes rotation gates to encode classical information into a quantum state. As discussed in section 2.1, the state of a qubit can be represented as a point on the Bloch sphere using the following unit vector in $\mathbb{C}^2$:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle = \begin{pmatrix} \cos(\frac{\theta}{2}) \\ e^{i\phi}\sin(\frac{\theta}{2}) \end{pmatrix}$$

As a qubit state can be completely identified by specifying two continuous variables, $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$, a single qubit can encode two real-valued features.
Let us assume we are given a dataset with M samples and K features, where all the features $X_1, \ldots, X_K$ are real-valued. Then, for each data pattern $j = 1, \ldots, M$,

we can consider a one-to-one correspondence between the feature values X_i^j and the corresponding rotation angles θ_i^j according to:

$$\theta_i^j = \frac{X_i^j - X_i^{min}}{X_i^{max} - X_i^{min}} \pi \tag{3.17}$$

where $X_i^{min} \doteq \min_j X_i^j$ and $X_i^{max} \doteq \max_j X_i^j$ are the minimum and maximum values of the $i-$th feature across all samples, respectively. The rotation angles θ_i^j generalize the angles θ and ϕ in Figure 2.1.
Given the data point $X^j = (X_1^j, \ldots, X_K^j) \in \mathbb{R}^K$, $j = 1, \ldots, M$, angle encoding can thus be applied to X^j in accordance with the mapping

$$X^j \mapsto \bigotimes_{i=1}^{K} \left(\cos\left(\frac{\theta_i^j}{2}\right) |0\rangle + \sin\left(\frac{\theta_i^j}{2}\right) |1\rangle \right)$$

This approach utilizes a single rotation gate per qubit and is thus capable of encoding the same number of features as the number of qubits used. However, it should be noted that a single quantum register has the ability to encode two real variables. To account for this, the authors in [42] proposed *dense angle encoding*, an alternative encoding scheme that utilizes an additional property of qubits, namely relative phase, to encode two features per qubit. The corresponding mapping for the classical data point X^j reads

$$X^j \mapsto \bigotimes_{i=1}^{K} \left(\cos\left(\frac{\theta_{2i-1}^j}{2}\right) |0\rangle + \exp\left(i\theta_{2i}\right) \sin\left(\frac{\theta_{2i-1}^j}{2}\right) |1\rangle \right)$$

This strategy allows for the encoding of n data points using only $\frac{n}{2}$ qubits.
The primary benefit of angle encoding is its high operational efficiency, as it requires only a constant number of parallel operations, regardless of the number of data values that need to be encoded. However, the number of qubits required is influenced by the number of data values, as one qubit is needed to encode each component of the input vector. Therefore, although the state preparation is efficient, the number of qubits needed for this encoding is not optimal.

3.4.4 Hamiltonian Encoding

While the methods explored so far involve encoding features directly into quantum states, Hamiltonian encoding is an alternative approach that utilizes the features to determine the evolution of the quantum system. It is inspired by the Schrödinger equation (2.21), whose solution can be written (ignoring the term $1/\hbar$ for simplicity) as

$$|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$$

The idea of this method is to associate the Hamiltonian H with a square matrix $\mathbf{X} \in \mathbb{C}^{n \times n}$ that represents the initial data. When $\mathbf{X}$ is Hermitian, we can create the Hamiltonian matrix $H_{\mathbf{X}}$ by setting $H_{\mathbf{X}} \doteq \mathbf{X}$. On the other hand, when $\mathbf{X}$ is not Hermitian, we can construct a Hermitian version of $H_{\mathbf{X}}$ using the augmented matrix

$$H_{\mathbf{X}} \doteq \begin{pmatrix} \mathbf{0} & \mathbf{X} \\ \mathbf{X}^\dagger & \mathbf{0} \end{pmatrix}$$

To apply Hamiltonian encoding, it is therefore necessary to execute the evolution

$$|\psi(t)\rangle = e^{-iH_{\mathbf{X}}t}|\psi(0)\rangle \tag{3.18}$$

on a quantum computer, which is also known as *Hamiltonian simulation*. The initial state $|\psi(0)\rangle$ represents an $n-$qubit system, and the resulting state $|\psi(t)\rangle$ can be regarded as a quantum state that "encodes" the information of the Hamiltonian, such as its eigenvalues in the phase of the amplitudes.
Given a precision level ϵ, the goal of Hamiltonian simulation is to determine a state $|\tilde{\psi}\rangle$, or an algorithm that produces this state, that satisfies the condition

$$\left\| |\tilde{\psi}\rangle - |\psi(t)\rangle \right\| \leq \epsilon$$

where $\|\cdot\|$ is a suitable norm used to quantify the distance between quantum states, and $|\psi(t)\rangle$ is the solution of the Schrödinger equation (3.18).
A significant bottleneck of this technique is related to the calculation of the exponential matrix $e^{-iH_{\mathbf{X}}t}$. Let us consider a Hamiltonian $H_{\mathbf{X}}$ that can be expressed as sum of several simpler Hamiltonians, i.e.

$$H_{\mathbf{X}} = \sum_{j=1}^{J} h_j H_j$$

where each H_j is simple to simulate. However, when the terms H_j do not commute, the factorization rule for exponentials does not hold, i.e.,

$$\exp\left(-i \sum_{j=1}^{J} H_j t\right) \neq \prod_{j=1}^{J} e^{-iH_j t}$$

To tackle the issue of non-commuting H_j, [43] suggested using the first-order Suzuki-Trotter formula

$$\exp\left(-i \sum_{j=1}^{J} H_j t\right) = \prod_{j=1}^{J} e^{-iH_j t} + \mathcal{O}(t^2) \tag{3.19}$$

This formula implies that the factorization rule is approximately valid for small values of t. This can be applied when expressing the evolution of $H_{\mathbf{X}}$ over time t as a sequence of small time steps of length Δt, so that the factorization reads

$$\exp\left(-i\sum_{j=1}^{J} H_j t\right) = \left[\exp\left(-i\sum_{j=1}^{J} H_j\,\Delta t\right)\right]^{\frac{t}{\Delta t}} = \left[\prod_{j=1}^{J} e^{-iH_j t} + \mathcal{O}(\Delta t^2)\right]^{\frac{t}{\Delta t}}$$

which has a small error. Clearly, there is a trade-off in choosing Δt, as smaller values result in a greater number of repetitions of the sequence. Despite this, the approach enables us to simulate $H_{\mathbf{X}}$ by simulating the H_j terms. In most cases, an $n-$qubit Hamiltonian H can be expressed as a combination of up to 4^n elementary Hamiltonians in the Pauli form as

$$H = \frac{1}{2^n} \sum_{i_1,\ldots,i_n \in \{X,Y,Z,I\}} \mathrm{Tr}\left(\bigotimes_{k=1}^{n} \sigma_{i_k} H\right) \bigotimes_{k=1}^{n} \sigma_{i_k}$$

Here, σ_{i_k} denotes a Pauli operator. However, using 4^n Pauli operators may be too extensive in general. Fortunately, local features of the Hamiltonian, such as sparsity [44], can help reduce the complexity of the problem.

3.5 Conclusions

In this chapter, we presented the concept of parametrized quantum circuits (PQCs) as a versatile tool for quantum machine learning and optimization applications. We discussed two significant algorithms associated with PQC, namely the Variational Quantum Eigensolver (VQE) and Quantum Approximate Optimization Algorithm (QAOA). VQE is a quantum-classical hybrid algorithm that aims to find the ground state energy of a given Hamiltonian. This is done by utilizing a PQC to prepare trial states, and then optimizing the circuit parameters classically to minimize the expectation value of the Hamiltonian with respect to the trial states. QAOA is another hybrid quantum-classical algorithm designed to solve combinatorial optimization problems. It involves encoding the optimization problem into a quantum state, and then evolving the state under a time-dependent Hamiltonian. The Hamiltonian is constructed from a series of unitary operators, and the circuit parameters are classically optimized to find the best approximate solution to the optimization problem.

Additionally, we explored several popular data encoding methods. Among them, angle encoding arguably stands out as the simplest and efficient to implement, while other methods have their own strengths and weaknesses, such as requiring more advanced hardware or being better suited for specific use cases.

Moving forward, the next chapter will focus on quantum machine learning and

delve into Generative Adversarial Neural Networks (GANs), providing an in-depth exploration of both classical and quantum versions.

Chapter 4

Quantum Machine Learning

Quantum Machine Learning (QML) is an emerging field that investigates the potential synergies between quantum computing and machine learning. Quantum computing, as we have seen, refers to information processing using devices that operate based on the principles of quantum theory. Machine learning, on the other hand, is a multidisciplinary area that combines statistics, mathematics, and computer science. It involves the analysis of large datasets, which are typically characterized by complex and nonlinear relationships, to enable computers to learn from past examples. This allows them to make predictions or solve problems that have not been encountered before. Machine learning transitioned into an industry-focused research field earlier and on a significantly larger scale compared to quantum computing, primarily due to the remarkable success of *Deep learning*. This machine learning approach involves constructing large architectures by stacking highly modular algorithms − the well-known *neural networks*, which are fed with vast amounts of data and trained using high-performance computing systems.
Proposals to combine quantum computing and machine learning have been around since the 1980s, but the term QML only came into use in 2013, when the authors of [45] mentioned it in their manuscript. Since then, the interest in the subject grew substantially, leading to a rapidly expanding collection of literature that explored various aspects of integrating the two fields. Presently, quantum machine learning is recognized as an active sub-discipline within quantum computing research, encompassing a variety of sub-areas.
There are different definitions of the term QML, but for the context of this chapter, we will use a narrow definition, which refers to *machine learning with quantum computers* or *quantum-assisted machine learning*. In this sense, QML explores the possibilities that the ongoing development of quantum computers presents within the context of intelligent data mining. To provide some generic examples, one might use a quantum computer as part of a model that needs to be trained; alternatively, one might use data generated by a quantum process or use a quantum computer

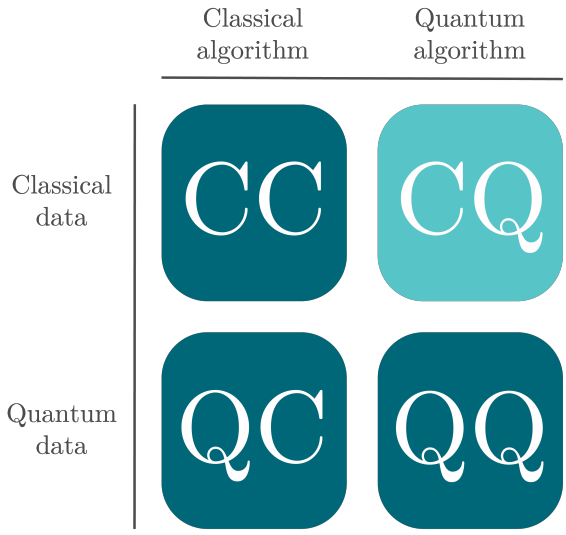


Figure 4.1: The four main categories of quantum machine learning, classified based on the types of models and data they utilize. This work focuses on the CQ category in the chart. Here, the input data is made up of observations from classical systems, such as text, images, or time series. These data are then processed by a quantum computer for analysis. Image adapted from [34].

to process quantum-generated data.
A classification proposed by [46] identifies four methods for integrating quantum computing and machine learning, contingent on whether the data is assumed to be generated by a quantum or classical system, and whether the information processing device is quantum or classical. This is schematically depicted in Figure 4.1.

In the CC scenario, both the data and computers are classical, but a quantum aspect is incorporated into the process. In other words, we can regard *quantum-inspired* classical machine learning techniques as part of quantum machine learning. An instance of this is the utilization of *tensor networks*, originally designed for quantum many-body systems, in the training of neural networks [47]. The QC section in the diagram corresponds to classical machine learning algorithms that depend on quantum data. Essentially, it involves either data generated by quantum processes or the use of classical machine learning techniques to quantum computing. In this chapter, we will concentrate on the type of quantum machine learning denoted by the CQ designation in the chart. This refers to machine learning that depends on classical data and integrates quantum computing into either the model or the training process. Lastly, the QQ category involves working on quantum data using quantum computing in either the models or training processes. Unlike CQ quantum machine learning, the quantum data in this case does not necessarily have to be acquired through measurements, and it can be directly integrated

into a quantum model. Although this area shows great potential, the necessary technologies are still in an early stage of development.
In this work, we will focus on the use of *Generative Adversarial Networks* (GAN) as a machine learning model. We will begin with a brief introduction to Feed-Forward Neural Networks (FFNN), followed by an exploration of the fundamental concepts underlying the classical GAN model. Next, we will shift our attention to examine different types of quantum GAN (qGAN) and their potential advantages.

4.1 Feed-Forward Neural Networks

Deep learning is a paradigm that has revolutionized numerous areas, such as computer vision, natural language processing, and speech recognition. By employing artificial neural networks with multiple layers, deep learning algorithms can learn complex patterns and representations from large datasets.
A *Feed-Forward Neural Network* (FFNN) is a type of artificial neural network where the connections between the units do not form a cycle. The information flows only in one direction, from the input layer, through the hidden layers, and finally to the output layer. This architecture is also known as a *multi-layer perceptron*.
The basic building block of a FFNN is the *artificial neuron*, which is also known as a *perceptron*. A perceptron takes several inputs, combines them using a weighted sum, adds a bias term, and applies an activation function to produce an output. This process is represented in Figure 4.2.

The activation function introduces non-linearity into the network, enabling it to learn complex relationships between inputs and outputs. Frequently used activation functions encompass:

- Sigmoid: $\sigma(x) = \frac{1}{1+\exp(-x)}$. The sigmoid function maps inputs to a range of $(0, 1)$, making it appropriate for binary classification tasks or as an output layer activation function for networks modeling probabilities.

- Rectified Linear Unit (ReLU): $\text{ReLU}(x) = \max(0, x)$. The ReLU function is favored due to its straightforward nature and ability to tackle the vanishing gradient issue [48].

- Hyperbolic Tangent (tanh): $\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$. The tanh function assigns input values to a range between $(-1, 1)$, offering a similar shape to the sigmoid function but with an expanded output range. Additionally, it is centered around zero, providing benefits in specific situations.

- Leaky ReLU: $\text{LeakyReLU}(x) = \max(\alpha x, x)$. This activation function represents a modification of the ReLU function, addressing its drawback of dead

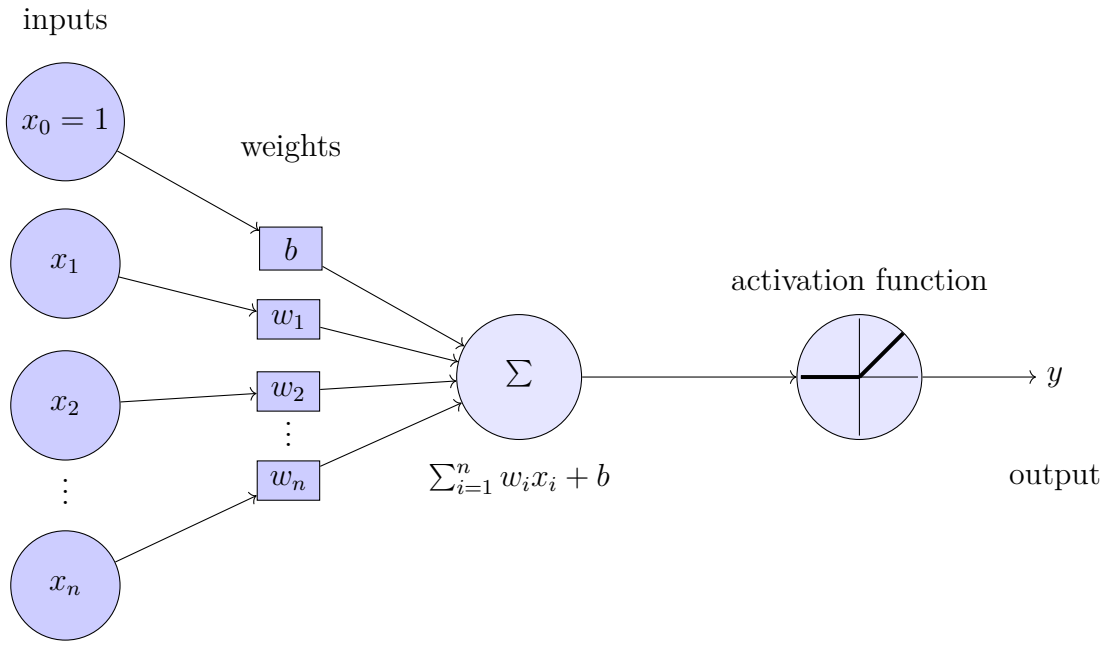


Figure 4.2: Schematic representation of a perceptron. The inputs are multiplied by suitable weights, and the resulting weighted sum is fed into an activation function that generates the output.

neurons by introducing a small slope for negative input values, with α being a small constant (e.g., 0.01).

The output of a perceptron is calculated as:

$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

where x_i is the $i-$th input, w_i is the weight associated with the $i-$th input, b is the bias term, n is the number of inputs, and $\varphi(\cdot)$ is the activation function.
To create a FFNN, several layers of perceptrons are stacked together. The first layer is called the input layer, and it contains the input variables. The last layer is called the output layer, and it produces the final output of the network. The layers in between the input and output layers are called the hidden layers. Figure 4.3 shows an example of FFNN with three hidden layers.

Each layer in a FFNN is fully connected to the next layer. This means that each perceptron in a layer is connected to every perceptron in the next layer. To calculate the output of a FFNN, the inputs are fed forward through the network, layer by layer. Specifically, the $l-$th layer is defined as

$$y_l = \varphi_l(W_l\, y_{l-1} + b_l)$$

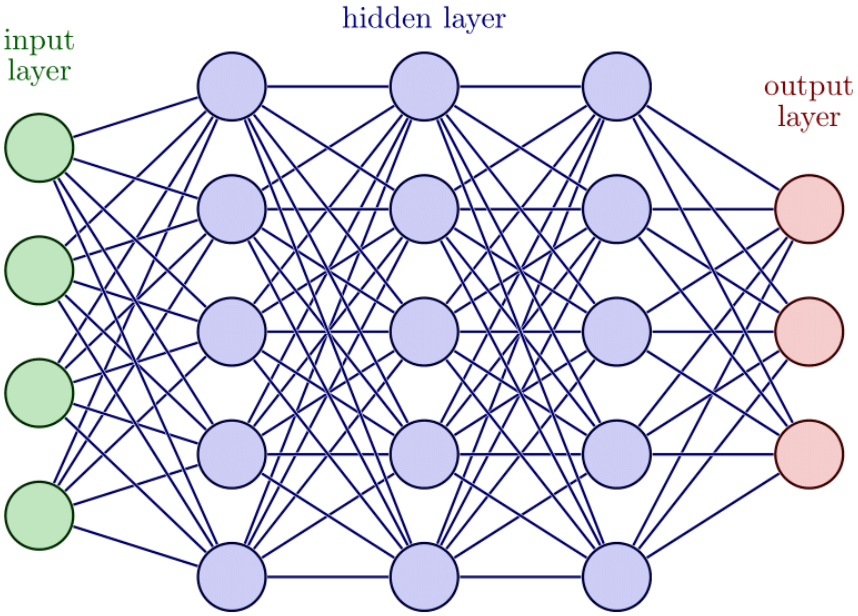


Figure 4.3: A Feed-Forward Neural Network with three hidden layers.

for $l = 1, \ldots, L$, where y_{l-1} is an $n_{l-1}-$dimensional input, y_l is an n_l-dimensional output, W_l is an $n_l \times n_{l-1}$ matrix of weights connecting layer $l-1$ to layer l, b_l is an n_l-dimensional bias vector, and $\varphi_l(\cdot)$ is a non-linear activation function applied element-wise.
The input to the first hidden layer is simply the input data $x = (x_1, \ldots, x_n)^T$ itself. Subsequent layers take the output of the previous layer as their input. Therefore, a multi-layer perceptron with L layers is obtained by setting $y = y_L$ and $y_0 = x$, and can be described by the following input-to-output map

$$y = \varphi_L(W_L\varphi_{L-1}(W_{L-1}\varphi_{L-2}(\ldots\varphi_1(W_1x)\ldots)))$$

parametrized by weight matrices $W_1, \ldots, W_L$ and bias vectors $b_1, \ldots, b_L$. The output of the final layer is the output of the entire network. In a classification task with k classes, the output layer typically has k neurons, each representing the probability of the input belonging to a particular class.
To train the network, a loss function (e.g., mean squared error or cross-entropy) is used to measure the difference between the predicted outputs and the actual labels. During training, the weights and biases of the network are updated using an optimization algorithm (e.g., backpropagation [32]) to minimize the loss.
The universal approximation theorem [49] also shows that Feed-Forward Neural Networks are capable of approximating any continuous function, making them a

versatile and effective tool for solving a variety of problems. However, like all machine learning models, they require careful tuning and selection of hyperparameters to achieve optimal performance.

4.2 Generative Adversarial Networks

Generative models are a class of deep learning techniques that are designed to learn the underlying structure and distribution of data, enabling them to generate new, realistic samples based on the learned patterns. Some popular generative models include Variational Autoencoders [50], Boltzmann Machines [51], and Gaussian Mixture Models [52].
Proposed by Ian Goodfellow et al. in 2014, Generative Adversarial Networks (GANs) [53] are a powerful class of generative models that have demonstrated remarkable results in generating realistic images, texts, and other types of data. The key innovation in GANs lies in the training process, which involves two deep neural networks, the *generator* and the *discriminator*, engaged in a continuous *adversarial game* to improve their respective performances. The goal of the generator is to create synthetic data that closely resembles the true data distribution. It takes random noise as input (usually from a Normal or Uniform distribution) and transforms it into a synthetic sample through a series of layers and nonlinear transformations. The quality of the generated samples improves iteratively as the generator learns to better mimic the true data distribution during the training process. The discriminator, on the other hand, acts as a binary classifier trained to differentiate between real data and synthetic data produced by the generator. It takes samples from both real and generated sources as input and outputs a probability indicating whether the input is real or fake. During training, the discriminator learns to correctly tell apart samples, while the generator attempts to deceive it by generating increasingly realistic synthetic data.
A very popular example to illustrate the GAN model involves comparing the generator to a counterfeiter who produces fake currency, while the discriminator takes on the role of detective, identifying genuine money from fake. The objective of the generator is to create counterfeit samples that are indistinguishable from the genuine samples, while the discriminator must accurately distinguish between the two. This process is depicted in Figure 4.4.
To formalize the GAN model, let $X = \{x^0, \dots, x^{M-1}\} \subset \mathbb{R}^{k_{out}}$ be a classical training dataset drawn from an unknown probability distribution p_{real}. The generator and discriminator networks are represented by two functions, namely $G_\theta : \mathbb{R}^{k_{in}} \to \mathbb{R}^{k_{out}}$ and $D_\phi : \mathbb{R}^{k_{out}} \to \{0,1\}$, respectively. Each of these functions is differentiable with respect to the corresponding network parameters, which are denoted by $\theta \in \mathbb{R}^{k_G}$ and $\phi \in \mathbb{R}^{k_D}$ for the generator and discriminator, respectively. The generator

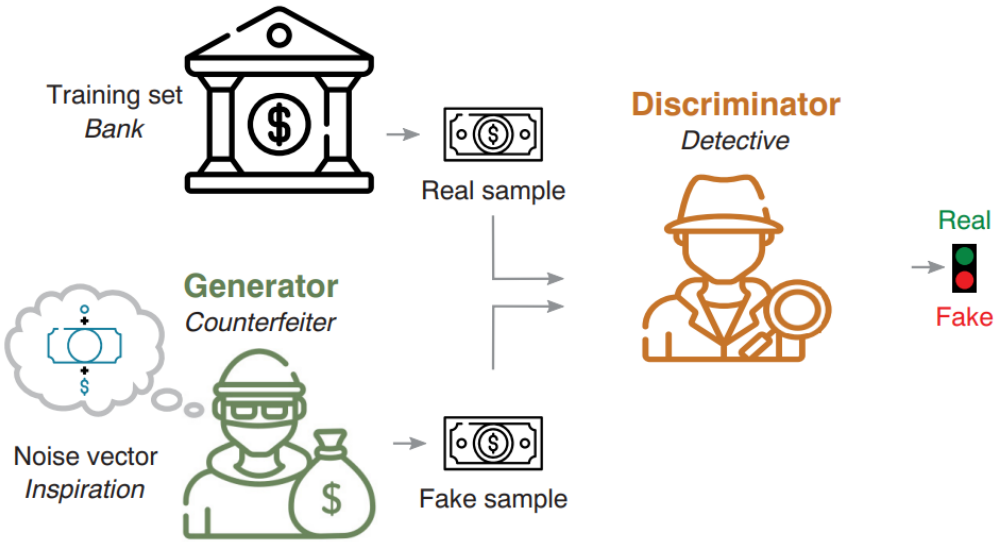


Figure 4.4: Analogy between a counterfeiter and a detective demonstrating the GAN logic, reproduced from [54]. The generator in GAN acts like the counterfeiter, trying to create fake money and constantly improving their techniques to deceive the detective. On the other hand, the discriminator embodies the role of the detective, constantly improving their methods to catch the counterfeiter. This ongoing competition between the two results in both of them becoming more proficient at their respective tasks.

transforms samples z from a fixed prior distribution p_{prior} in $\mathbb{R}^{k_{in}}$ into samples that are expected to be indistinguishable from those of the real distribution p_{real} in $\mathbb{R}^{k_{out}}$ [55]. In contrast, the discriminator D_ϕ attempts to tell apart fake and real data, and outputs a single scalar value representing the probability of the input coming from p_{real}.

The cost function of each player is defined based on the parameters of both players. Specifically, the discriminator aims to minimize $\mathcal{L}_D(\theta, \phi)$ by controlling only ϕ, while the generator aims to minimize $\mathcal{L}_G(\theta, \phi)$ by controlling only θ. As the cost of each player depends on the other player's parameters, but neither player can control the other's parameters, this situation is best described as a game rather than an optimization problem [56]. In general, the solution to an optimization problem is a (local) minimum, i.e., a point in the parameter space where all neighboring points have greater or equal cost. In contrast, the solution to a game is called a *Nash equilibrium*, which we refer to here as a *local differential Nash equilibrium* according to the terminology introduced by [57]. In such a case, a Nash equilibrium describes a combination of parameter values (θ, ϕ) that represents a local minimum of $\mathcal{L}_G$

with respect to θ and a local minimum of $\mathcal{L}_D$ with respect to ϕ.
Within the framework of GANs, there are multiple cost functions that can be utilized. Actually, all the games developed for GANs share a common cost function for the discriminator network, while the variation among these games is limited to the cost adopted for the generator.
The discriminator cost function is

$$\mathcal{L}_D(\theta, \phi) = -\mathbb{E}_{x\sim p_{real}}[\log D_\phi(x)] - \mathbb{E}_{z\sim p_{prior}}[\log(1 - D_\phi(G_\theta(z)))] \tag{4.1}$$

which is nothing but the ordinary cross-entropy cost used for training binary classifiers with a sigmoid output. However, in GAN the classifier is trained using two separate minibatches of data - one from the real dataset, with all patterns labeled as 1, and the other from the generator, where all examples are labeled as 0. Equation (4.1) is minimized by the discriminator in all GAN variations to achieve its objective. Training the discriminator in a GAN enables us to estimate the ratio

$$\frac{p_{real}(x)}{p_G(x)}$$

for every data point x, where p_G denotes the probability distribution implicitly defined by the generated data samples $G(z)$. Despite being referred to as "adversarial" due to its analysis with game theory, GANs can alternatively be seen as cooperative because the discriminator estimates such density ratio and provides feedback to the generator to improve, thus acting more like a teacher than an adversary.
The *zero-sum game* is the simplest form of the GAN framework, where the sum of costs for all players is always zero, and the generator and discriminator costs are thus related by $\mathcal{L}_G = -\mathcal{L}_D$. In this case, the entire game can be summarized with a value function V that specifies the discriminator's payoff, i.e.,

$$V(\theta, \phi) = -\mathcal{L}_D(\theta, \phi) \tag{4.2}$$

This type of game is also known as a *minimax game*, since it involves minimizing in an outer loop and maximizing in an inner loop:

$$\min_\theta \max_\phi V(\theta, \phi) = \mathbb{E}_{x\sim p_{real}}[\log D_\phi(x)] + \mathbb{E}_{z\sim p_{prior}}[\log(1 - D_\phi(G_\theta(z)))] \tag{4.3}$$

The minimax game formulation of GANs is attractive for theoretical analysis due to its simplicity. [53] demonstrated that learning in this game is similar to minimizing the Jensen-Shannon divergence [58] between the data and generated distribution, and that convergence to equilibrium is possible if the policies of both players can be updated in function space. However, these theoretical results may not apply in practice since the players are usually represented by deep neural networks and the

updates are made in parameter space, which can violate convexity assumptions. Therefore, the cost function for the generator in the minimax game (4.3) is helpful for theoretical analysis, but not very effective in practice. Moreover, equation (4.3) may not yield a strong enough gradient for the generator to learn effectively in practice. Typically, the generator performs poorly during the initial stages of learning, and its generated samples are noticeably distinct from the training data. As a result, the discriminator can confidently reject the generated samples. In such circumstances, the term $\log(1 - D(G(z)))$ saturates.
To tackle this issue, an alternative approach is to retain cross-entropy minimization for the generator but modify the target used to calculate the cross-entropy cost (rather than simply changing the sign of the discriminator's cost to derive the generator's cost). This results in the following updated cost function for the generator:

$$\mathcal{L}_G(\theta, \phi) = -\mathbb{E}_{z \sim p_{prior}}[\log(D_\phi(G_\theta(z)))] \tag{4.4}$$

In the minimax game, the generator aims to minimize the log-probability of the discriminator being correct. However, in this modified version of the game, the generator aims to maximize the log-probability of the generated samples being classified as real. This modification is based on heuristics rather than theoretical concerns, and its primary objective is to ensure that each player has a strong gradient when they are losing the game [56]. This version of the game is referred to as the *non-saturating game*. It is no longer a zero-sum game and cannot be described by a single value function.
During practical implementation, the expected values are estimated using batches of size m. As a result, the cost function for the discriminator and the generator can be expressed as

$$\mathcal{L}_D(\theta, \phi) = -\frac{1}{m} \sum_{i=1}^{m} [\log D_\phi(x^i) + \log(1 - D_\phi(G_\theta(z^i)))] \tag{4.5}$$

$$\mathcal{L}_G(\theta, \phi) = -\frac{1}{m} \sum_{i=1}^{m} [\log(D_\phi(G_\theta(z^i)))] \tag{4.6}$$

for $x^i \in X$ and $z^i \sim p_{prior}$.
The process of training a GAN hence involves finding a Nash-equilibrium of the game

$$\min_\phi \mathcal{L}_D(\theta, \phi) \tag{4.7}$$

$$\min_\theta \mathcal{L}_G(\theta, \phi) \tag{4.8}$$

Generally, the optimization process for equations (4.7) and (4.8) involves alternating update steps for the generator and discriminator parameters. However, this approach leads to non-stationary objective functions because updates to the

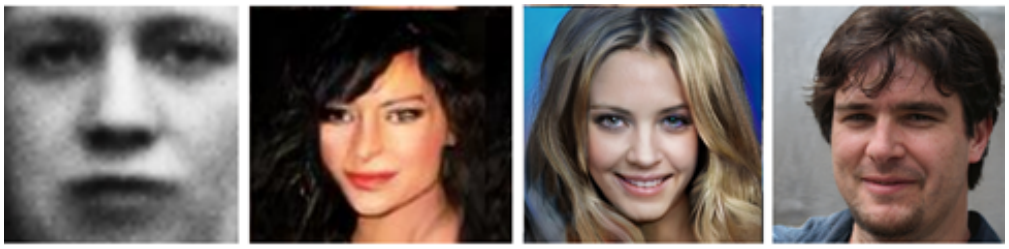


Figure 4.5: Advancements made by Generative Adversarial Networks (GANs) trained on facial images. From left to right, the images are from: GAN [53] (2014); CoGAN [61] (2016); Progressive GAN [62] (2018); StyleGAN [63] (2019).

generator's or discriminator's network parameters will also affect the loss function of the other player. To address this issue, adaptive-learning rate, gradient-based optimizers like ADAM [59] and AMSGRAD [60] are commonly used to perform the update steps. These optimizers use an exponentially decaying average of previous gradients and are well-suited for solving non-stationary objective functions. In Appendix A, we provide an overview of the most common gradient-based optimization algorithms used for training neural networks and various other machine learning algorithms.

Since the introduction of the classical GAN model, several GAN variants have been proposed that have improved the stability and quality of generated outputs. Some notable GAN variants include conditional GANs [64], which allow for control over the output by conditioning on additional information, such as labels or input images; DCGANs [65], which utilize deep convolutional architectures for both the generator and discriminator networks; CoGANs [61], which learn joint distributions of multiple domains; Progressive GANs [62], which generate high-resolution images by incrementally adding layers to the generator network; and StyleGANs [63], which use adaptive instance normalization and style transfer techniques to improve image quality and diversity. StyleGAN has also been introduced to enable the control of high-level attributes of generated images such as facial expressions, hairstyles, and clothing styles. As shown in Figure 4.5, these GAN variants have demonstrated impressive results on a variety of image generation tasks and continue to be an active area of research in the machine learning community.

4.3 Quantum Generative Adversarial Networks

One area of research in the field of QML investigates the possibilities presented by *quantum Generative Adversarial Networks* (qGANs) [66, 67, 68]. A qGAN operates similarly to a classical GAN, with a discriminator and a generator competing against each other. However, a quantum GAN incorporates a quantum model,

usually in the form of a quantum neural network (i.e., a PQC that can be trained as either generative or discriminative machine learning models), to perform part of the model. Despite the quantum aspect, training a quantum GAN follows the same principles as training a classical GAN.
The architecture of quantum GANs can vary depending on the problem being addressed, but they all share the fundamental components of a competing discriminator and generator. In general, quantum GANs can be classified into one of the following categories:

- *Use of quantum data with discriminator and generator both quantum:* qGAN architecture with a fully quantum model utilizes quantum data (i.e., quantum states), where both the generator and discriminator are quantum circuits that operate on such quantum states. Because all components of the qGAN are quantum circuits, feature maps or measurement operations in the middle of the model are not required, resulting in a unique architecture that is seamlessly integrated.

- *Use of quantum data with a quantum generator and a classical discriminator:* When a qGAN utilizes quantum data with a quantum generator and a classical discriminator, the resulting architecture will resemble that of classical GANs. The quantum generator produces quantum states that are eventually transformed into classical data through a measurement operation, allowing them to be fed into the classical discriminator network. It is important to note that the original quantum data will also need to be measured in this scenario.

- *Use of classical data with a quantum generator or discriminator:* In cases where classical data is utilized with a quantum generator or discriminator, qGANs can best resemble their classical counterparts. This approach involves replacing the generator or discriminator, or both, with a quantum model that has classical inputs and outputs. If a quantum discriminator is used, a feature map is required to load classical data into a quantum state.

Since classical data is much more readily available than quantum data, this last type of architecture has been extensively studied by the quantum computing community. The remainder of the section is devoted to discussing two different variants of qGAN that fall into this category: QuGAN [69] and qGAN for Learning and Loading Random Distributions [55]. In the first model, both the generator and the discriminator are quantum, while the second model utilizes a hybrid architecture with a quantum generator and a classical discriminator network.

4.3.1 QuGAN

QuGAN [69] is a quantum GAN architecture that utilizes *quantum state fidelity* for both the quantum discriminator and generator. This approach enables the use of quantum-based loss functions, which are calculated using a *swap test* on qubits. The swap test is a quantum computational procedure that measures the degree of difference between two quantum states, and enables the use of quantum fidelity measurements as loss functions in QuGAN.
Figure 4.6 shows the quantum circuit used in the test, which was originally introduced in the context of quantum fingerprinting [70]. This quantum circuit involves two qubit registers prepared in $|\psi\rangle \otimes |\varphi\rangle = |\psi\rangle|\varphi\rangle$, and an ancillary qubit (the qubit in the first register) in the $|0\rangle$ state. The circuit comprises three gates: two Hadamard gates H and a controlled *SWAP* gate (controlled by the first qubit), denoted by *CSWAP*. The Hadamard gates transform the $|0\rangle$ state into a superposition $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, and the $|1\rangle$ state into a superposition $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. The controlled *SWAP* operation swaps the states $|\psi\rangle$ and $|\varphi\rangle$ if the ancillary qubit is in the state $|1\rangle$. However, when the ancillary qubit is in state $|0\rangle$, the states remain in their original order. The sequence of operations performed by the circuit is thus as follows:

$$\begin{aligned} |0\rangle|\psi\rangle|\varphi\rangle \xrightarrow{H} \frac{|0\rangle + |1\rangle}{\sqrt{2}}|\psi\rangle|\varphi\rangle \xrightarrow{CSWAP} \frac{|0\rangle|\psi\rangle|\varphi\rangle + |1\rangle|\varphi\rangle|\psi\rangle}{\sqrt{2}} \\ \xrightarrow{H} \frac{1}{2}|0\rangle \otimes \Big(|\psi\rangle|\varphi\rangle + |\varphi\rangle|\psi\rangle\Big) + \frac{1}{2}|1\rangle \otimes \Big(|\psi\rangle|\varphi\rangle - |\varphi\rangle|\psi\rangle\Big) \end{aligned} \tag{4.9}$$

After completing the circuit, the state of the ancillary qubit is measured. The outcome where the measurement results in the 0 state is referred to as "outcome 0", while the outcome where the measurement results in the 1 state is referred to as "outcome 1". To determine the probability of measuring the ancillary qubit in the state 0, also known as the *acceptance probability* p_0, we isolate the coefficients of the $|0\rangle$ state and square them, obtaining

$$\begin{aligned} p_0 = \mathbb{P}(\text{ancillary qubit} = 0) &= \frac{1}{2}\Big(\langle\psi|\langle\varphi| + \langle\varphi|\langle\psi|\Big)\frac{1}{2}\Big(|\psi\rangle|\varphi\rangle + |\varphi\rangle|\psi\rangle\Big) \\ &= \frac{1}{2} + \frac{1}{2}|\langle\varphi|\psi\rangle|^2 \end{aligned} \tag{4.10}$$

When the two states are identical, the outcome is 0 with probability 1. In this case, swapping the qubit's positions has no impact, and there is no entanglement with the ancillary qubit. However, if the input states are different, both outcomes are possible; this means that outcome 1 can only occur for different states. In such instances, we say that the states "fail" the test. When two states fail the test, we can definitively conclude that they are different. Conversely, if the states "pass"

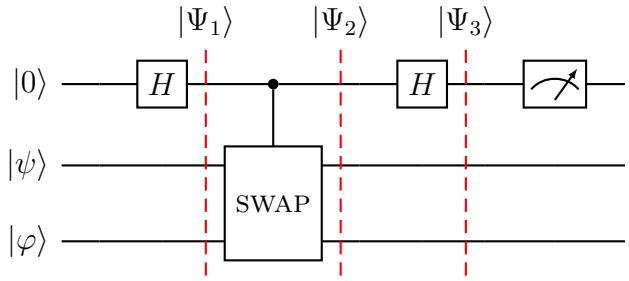


Figure 4.6: Quantum circuit for *SWAP* test. This setup includes three quantum gates: a controlled-*SWAP* gate and two Hadamard gates. $|\Psi_1\rangle$ denotes the quantum state resulting from the application of the first Hadamard gate on the ancillary qubit (the qubit in the first quantum register). This state is then transformed into $|\Psi_2\rangle$ by using the controlled-*SWAP* gate on the two registers $|\psi\rangle$ and $|\varphi\rangle$, conditioned on the ancillary qubit being in state $|1\rangle$. The final state, $|\Psi_3\rangle$, is obtained by applying the second Hadamard gate on the first qubit. At this time, the probability of observing the ancillary qubit in state 0, which represents the acceptance probability $p_0 = \frac{1}{2} + \frac{1}{2}|\langle\varphi|\psi\rangle|^2$ of the test, is computed. This quantity measures the fidelity of the two states and is bounded in $\left[\frac{1}{2},1\right]$. If the states are orthogonal, then the overlap $|\langle\varphi|\psi\rangle|^2$ will be zero and the fidelity will be $\frac{1}{2}$. On the other hand, if the states are identical, then $|\langle\varphi|\psi\rangle|^2 = 1$ and the fidelity will also be 1.

the test (outcome 0), it does not necessarily mean that they are equal.
From equation (4.10), we can notice that the acceptance probability p_0 depends on the absolute square of the inner product between states $|\varphi\rangle$ and $|\psi\rangle$. This quantity is known as *overlap* of the input states, and it provides a reliable estimate of how similar (i.e., close) the two states are. Notably, if the two states being compared are orthogonal, then $|\langle\varphi|\psi\rangle|^2 = 0$ and the probability of passing the test is $p_0 = \frac{1}{2}$. Conversely, for non-orthogonal states, the probability of passing the test increases as the states become more similar.
In the more general scenario where the two input states are mixed states represented by the density operators ψ and φ, the same method can be used, and the acceptance probability of the test is given by [71]

$$p_0 = \frac{1}{2} + \frac{1}{2}\,\mathrm{Tr}(\psi\rho) \tag{4.11}$$

It is worth noting that in this case the expression $\psi\rho$ is not an abbreviation for the tensor product, but a proper operator product. However, in the following we will consider equation (4.10) for the sake of simplicity.
Figure 4.7 depicts the architecture design of the QuGAN system, which operates

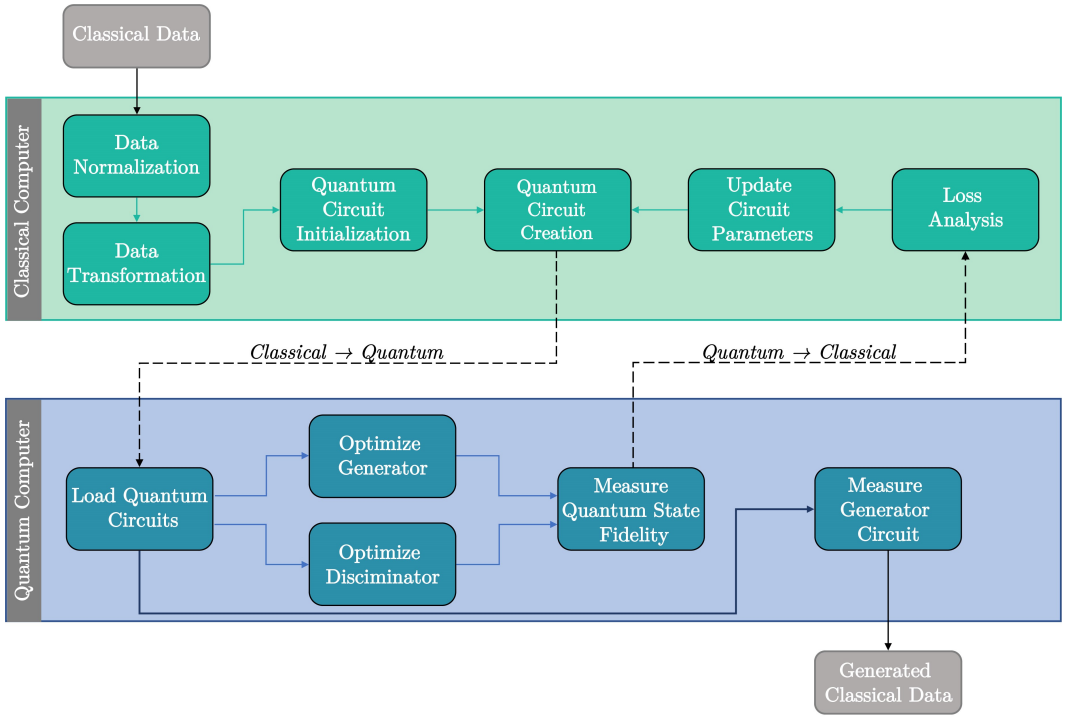


Figure 4.7: QuGAN Architecture

through iterative exchanges between a classical and quantum computer. The process involves a classical computer passing a parameterized quantum circuit to a quantum computer, which measures the quantum state fidelity and sends it back to the classical computer.
The QuGAN system begins by pre-processing classical data, which is normalized and then transformed into quantum data. This step is performed only once per dataset. Depending on the stage of the training algorithm, either a generator/discriminator circuit or a real data/discriminator circuit is passed to the quantum computer. The fidelity of the states induced by the quantum circuit is determined by measuring the expectation value of a single ancillary qubit, and this fidelity information is subsequently transmitted to the classical computer for further processing and analysis. If the system is being optimized, the fidelity is utilized to compute the gradient of each gate in relation to the cost function, and it is then used to adjust the parameters of the generator and discriminator circuits. The process of loading quantum circuits, measuring state fidelity, and updating gate parameters is repeated until the desired convergence is achieved or sufficient epochs have been completed. Furthermore, once model training is finished, the quantum computer is no longer required to return a state fidelity, but instead produces a (classical) data point by sampling from the quantum circuit of the generator.

The role of the quantum computer is dependent on the two possible types of circuits that can be prepared. If real data is being prepared, the circuit prepares the discriminator state using a PQC, while the data is encoded using some encoding scheme and fed onto the circuit alongside the discriminator. On the other hand, in the case of the generator, both the discriminator and generator states are prepared using their respective PQCs. Notably, the discriminator and generator losses can be expressed as

$$D_{loss} = -\mathbb{E}[\log(|\langle\zeta|\delta\rangle|^2)] - \mathbb{E}[\log(1 - (|\langle\gamma|\delta\rangle|^2))] \tag{4.12}$$

$$G_{loss} = -\mathbb{E}[\log(|\langle\gamma|\delta\rangle|^2)] \tag{4.13}$$

where $|\delta\rangle, |\gamma\rangle$ and $|\zeta\rangle$ denote the quantum states corresponding to the discriminator D, generator G, and real data X, respectively. The logarithmic terms in the above equations are related to the normalization of the quantum fidelity measurement to a [0,1] scale. Indeed, using a range of $[0.5, 1]$ for the input may not be effective in a logarithmic optimization problem, so the $SWAP$ test value is normalized to a range of $[0, 1]$ instead. This is why there is no $\frac{1}{2}$ term in the above equations.
To create a sample, the quantum state corresponding to the generator is induced and sampled q times. Each qubit's average measurement represents the output for a specific dimension.
We can update the network parameters in a manner similar to a classical GAN by utilizing quantum gate differentiation [72] and gradient descent (see Appendix A). By differentiating a quantum gate, we obtain the gradient of a quantum function (i.e., a function mapping gate parameters to an expectation) with respect to the gate parameters. The system measures the discriminator loss relative to both the generator and real data and then updates the parameters of the corresponding PQC to improve performance. The generator also analyzes its performance against the discriminator and updates its PQC parameters to improve performance accordingly. [72] demonstrated that the gradients of quantum measurement expectation values can be estimated using the same or nearly the same architecture that executes the original circuit. Specifically, to perform gradient descent of each gate, we consider the following parameterized differential equation

$$\frac{\partial f(\theta)}{\partial \mu} = \frac{1}{2}\left(f\left(\mu + \frac{\pi}{2}\right) - f\left(\mu - \frac{\pi}{2}\right)\right) \tag{4.14}$$

where $\mu \in \theta$ is one of the gate parameters, and f is the quantum cost function, which in our case is defined by equations (4.12) and (4.13). This method enables analytical differentiation of the Quantum Neural Network, but it has the drawback of being computationally expensive because the network must be induced twice per gate to obtain the gradient.

4.3.2 qGAN Distribution Learning

The process of loading classical data into a quantum register is recognized as a crucial challenge of NISQ devices that can undermine the effectiveness of several key quantum processing algorithms. It is widely believed that classical data loading will continue to pose a significant obstacle, unless quantum parallel access to information is possible through the use of quantum random access memories (qRAMs) [73]. Therefore, recent discussions in the literature have focused on the concept of *efficiently loading approximate data* by adapting machine learning techniques to quantum hardware.
In general, the most effective methods for loading a generic data structure into an $n-$qubit state require $\mathcal{O}(2^n)$ gates to achieve an exact representation. [55] introduced a hybrid quantum-classical algorithm that uses quantum Generative Adversarial Networks to efficiently and approximately load generic probability distributions into quantum states. This is done by learning the representation of the probability distribution underlying the data samples through a combination of a quantum channel, like a parametrized quantum circuit, and a classical neural network. Such loading process only requires $\mathcal{O}(poly(n))$ gates, which could allow for the utilization of quantum algorithms that may be advantageous, like Quantum Amplitude Estimation (QAE) [74]. Hence, efficient qGANs could prove beneficial for data loading in virtually every field that currently utilizes Monte Carlo simulations [75], as Quantum Amplitude Estimation is considered the counterparty of Monte Carlo integration in quantum computing. Indeed, QAE is capable of working with approximate state preparation because it is not significantly affected by small errors in the input state. In other words, minor deviations in the input state only result in minor deviations in the outcome.
In this situation, the goal of the qGANs is not to generate classical samples that match a set of given classical training data, but to teach the quantum generator how to produce a quantum state that accurately represents the underlying probability distribution of the data. More precisely, the quantum generator is trained to convert an $n-$qubit input state $|\psi_{in}\rangle$ to an $n-$qubit output state $|g_\theta\rangle$ described by

$$G_\theta |\psi_{in}\rangle \doteq |g_\theta\rangle = \sum_{j=0}^{2^n-1} \sqrt{p_\theta^j}\, |j\rangle \tag{4.15}$$

where p_θ^j represents the occurrence probabilities of the basis states $|j\rangle$. Therefore, the quantum generator, once trained, is expected to represent an $n-$qubit quantum state given by the equation

$$|g_{\text{trained}}\rangle = \sum_{j=0}^{M-1} \sqrt{p_j}\, |x^j\rangle, \tag{4.16}$$

In this equation, the basis states $|x^j\rangle$ correspond to the data items in the training dataset $X = \{x^0, \ldots, x^{M-1}\}$, with $M \leq 2^n$, while p_j represents the probability of sampling $|x^j\rangle$.
In order to achieve this representation, we must first map the samples from continuous probability distribution into discrete values. The number of possible values that can be represented depends on the number of qubits used in the mapping. For instance, if we use 4 qubits to represent one feature, we can represent 16 discrete values. To obtain these discretized values, we can use an affine map such as

$$\{0,1,\ldots,2^n-1\} \ni i \mapsto \frac{high - low}{2^n - 1} \cdot i + low \tag{4.17}$$

which transforms an input value i into a value within the range $[low, high]$, where the values for low and $high$ are usually selected as the 5th and 95th percentiles of the dataset X. This is done to reduce the number of required qubits for a reasonable representation of the distribution. In the training process, the affine mapping can be applied in a classical manner by measuring the quantum state. However, if the resulting quantum channel is utilized in another quantum algorithm, the mapping must be executed as part of the quantum circuit. It has been shown that this affine mapping can be implemented in a gate-based quantum circuit using a linear number of gates [76].
The quantum generator is created using a variational form that consists of a parametrized quantum circuit, while the discriminator is implemented as a classical deep neural network. To train the qGAN, the generator output state $|g_\theta\rangle$ is measured in the computational basis to obtain samples, where the measurement outcomes can be any $|j\rangle$, $j \in \{0,1,\ldots,2^n-1\}$. Unlike classical sampling, the sampling process does not require a stochastic input, as it is based on the intrinsic stochasticity of quantum measurements. The measurements produce classical information, with p_j defined as the measurement frequency of $|j\rangle$.
To extend this scheme to $d-$dimensional distributions, d qubit registers can be selected, each with n_i qubits, $i = 1, \ldots, d$ and finally a multidimensional grid is constructed.
The choice of an appropriate input state $|\psi_{in}\rangle$ can aid in reducing the complexity of the quantum generator and the number of training epochs while avoiding local optima during quantum circuit training. However, it is crucial that the preparation of such input state does not dominate the overall gate complexity, and thus it should be loadable using $\mathcal{O}(poly(n))$ gates. This is achievable for efficiently integrable probability distributions, such as log-concave distributions (e.g., the exponential and normal families), which are characterized by the property that $\frac{\partial^2}{\partial x^2} \log p(x) < 0$.
In practical applications, the statistical analysis of the training data can assist in selecting an appropriate initial state from a set of distributions that can be loaded efficiently, for instance by comparing the expected value and variance of the

distributions. The classical discriminator is a deep neural network that includes multiple layers that apply nonlinear activation functions. As in the classical case, its role is to process data samples and classify them as either real or generated.
It is important to carefully choose the topology of the networks, including the number of nodes and layers, to prevent the discriminator from becoming too dominant over the generator or vice versa.
The qGAN loss functions are determined by considering m data samples g^i generated by the quantum generator and m training data samples x^i chosen randomly, and are given by

$$\mathcal{L}_D(\theta, \phi) = -\frac{1}{m} \sum_{i=1}^{m} \left[\log D_\phi(x^i) + \log(1 - D_\phi(g^i))\right] \tag{4.18}$$

$$\mathcal{L}_G(\theta, \phi) = -\frac{1}{m} \sum_{i=1}^{m} \left[\log(D_\phi(g^i))\right] \tag{4.19}$$

Here, ϕ denotes the weight vector of the deep neural network used as the discriminator, while θ corresponds to the parameter vector of the PQC used as the generator. The training process is illustrated in Figure 4.8. To calculate the analytical gradient of the generator quantum circuit, we can exploit the fact that equation (4.19) can be rewritten as

$$\mathcal{L}_G(\theta, \phi) = -\sum_{j=0}^{2^n-1} p_\theta^j \left[\log(D_\phi(g^j))\right] \tag{4.20}$$

where

$$p_\theta^j = |\langle j|g_\theta\rangle|^2 \tag{4.21}$$

To update the parameters θ using gradient-based methods, we need to evaluate the expression

$$\frac{\partial \mathcal{L}_G(\theta, \phi)}{\partial \mu} = -\sum_{j=1}^{m} \frac{\partial p_\theta^j}{\partial \mu} \left[\log(D_\phi(g^j))\right] \tag{4.22}$$

for all gate parameters $\mu \in \theta$. As seen in section 4.3.1, quantum gradients can be evaluated according to the equation

$$\frac{\partial p_\theta^j}{\partial \mu} = \frac{1}{2}\left(p^j_{\mu+\frac{\pi}{2}} - p^j_{\mu-\frac{\pi}{2}}\right) \tag{4.23}$$

Similar to the classical scenario, the optimization of the loss functions is performed alternately by adjusting the parameters of the generator and the discriminator.

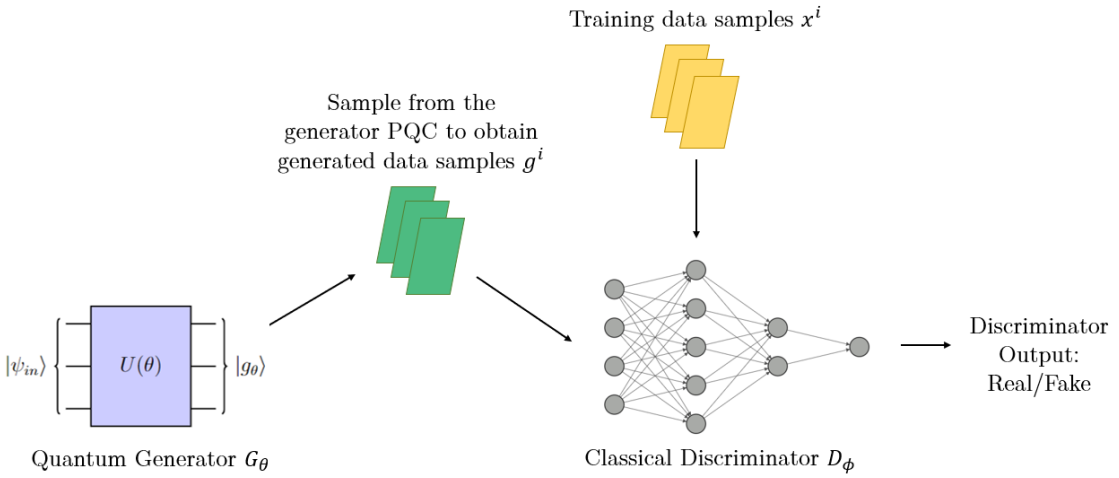


Figure 4.8: qGAN for probability distribution learning. In this case, the quantum generator - which is a parametrized quantum circuit - is trained to convert an input state $|\psi_{in}\rangle$ into an output state $|g_\theta\rangle$ that accurately reflects the probability distribution underlying the data. To train the qGAN, samples are obtained by measuring the output state $|g_\theta\rangle$ in the computational basis. Similar to the classical scenario, the discriminator aims to distinguish between the generated samples and the actual training samples. The generator and discriminator are trained in an iterative and alternating manner to improve their respective performances.

4.4 Conclusions

In this chapter, we introduced the exciting field of Quantum Machine Learning, which combines the power of quantum computing with classical machine learning techniques. Specifically, we focused on the quantum-assisted machine learning paradigm that integrates quantum computing either into the model or the training process. To provide context, we briefly described classical Feed-Forward Neural Networks, which are fundamental building blocks in classical machine learning. Additionally, we provided a detailed explanation of the classical GAN model, which has been instrumental in generative modeling in recent years. Finally, we discussed two versions of quantum GAN: QuGAN, which is a fully quantum model based on quantum state fidelity, and qGAN for distribution learning, which uses a quantum generator and a classical discriminator to load an approximate probability distribution into a quantum state. These concepts set the stage for the next chapter, where we will illustrate the application of these techniques in numerical experiments. Notably, we will consider the application of QuGAN to quantum image generation, and present the results related to the loading of a univariate and bivariate Student's

t-distribution using qGAN for distribution learning.

Chapter 5

Applications and Numerical Experiments

This chapter presents the results obtained from applying the previously defined quantum machine learning models to real datasets. Firstly, we will analyze the QuGAN performance in the classical task of image generation using the MNIST dataset [77], which comprises of 70,000 hand-drawn digit images labeled with their respective classes. Secondly, we will evaluate the effectiveness of the qGAN in loading univariate and multivariate Student's t-distributions, which have become popular models for economic and financial data due to their various applications such as modeling of stock returns [78], European option pricing [79], measuring Value-at-Risk [80] and many more.
All the numerical experiments were implemented using Python 3.9 frameworks, including Qiskit [81] and PennyLane [82]. Our primary focus is to evaluate the accuracy of the proposed methods rather than computational time, and to highlight the benefits and possible improvements of a hybrid approach over a classical one.

5.1 QuGAN applied on MNIST dataset

5.1.1 Dataset and Data Qubitization

The MNIST dataset is widely used as a benchmark for evaluating the performance of various machine learning models in image classification tasks. This dataset contains 60,000 images of handwritten digits (0-9) for training and 10,000 images for testing, each of which is labeled with its corresponding class. The images are grayscale and have a resolution of 28×28 pixels. Figure 5.1 illustrates a random sample of 14 images from the dataset.

To translate a classical numerical pixel value x_i into a quantum state, we first

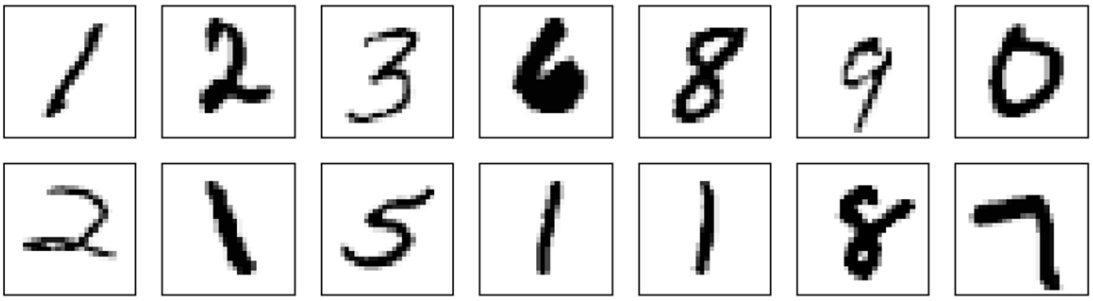


Figure 5.1: Random sample of 14 images from the MNIST dataset.

scale it to fit within the range $[0, 1]$. Afterward, we apply a rotation around the $y-$axis, where the rotation angle is defined as

$$\theta_{x_i} = 2\sin^{-1}(\sqrt{x_i}) \tag{5.1}$$

The resulting qubit expectation, when measured against the Z basis, corresponds to the x_i value of the classical data that the qubit encodes.

5.1.2 Dimensionality Reduction

The primary criterion for evaluating success in the image generation field is the perceived realism of generated images to the human eye. However, generating images with a resolution of 784 pixels would require a number of qubits that is currently infeasible for existing quantum hardware, necessitating the need for dimensionality reduction of the original images. This reduced representation is referred to as the *latent vector* of the image. Nevertheless, reducing the dimensionality while maintaining fidelity is challenging.
In this work, we implement an *autoencoder* to reduce the dimensionality of the data. An autoencoder is a type of neural network that utilizes its inherent nonlinear characteristics to enable more advanced dimensionality reduction, compared to standard techniques such as Principal Component Analysis [83].

It is composed of two main parts: an *encoder* and a *decoder*. The encoder takes the input data and maps it to a lower-dimensional representation, while the decoder takes this lower-dimensional representation and maps it back to the original input space. Mathematically, an autoencoder can be defined as follows. Let $x \in \mathbb{R}^d$ be an input data point, and let $z \in \mathbb{R}^m$ be its corresponding encoded representation, with $m \ll d$. The encoder can be represented by a function $f : \mathbb{R}^d \to \mathbb{R}^m$, which maps the input data x to its encoded representation z. The decoder can be represented by a function $g : \mathbb{R}^m \to \mathbb{R}^d$, which maps the encoded representation z back to the original input space. During training, the autoencoder learns to minimize a

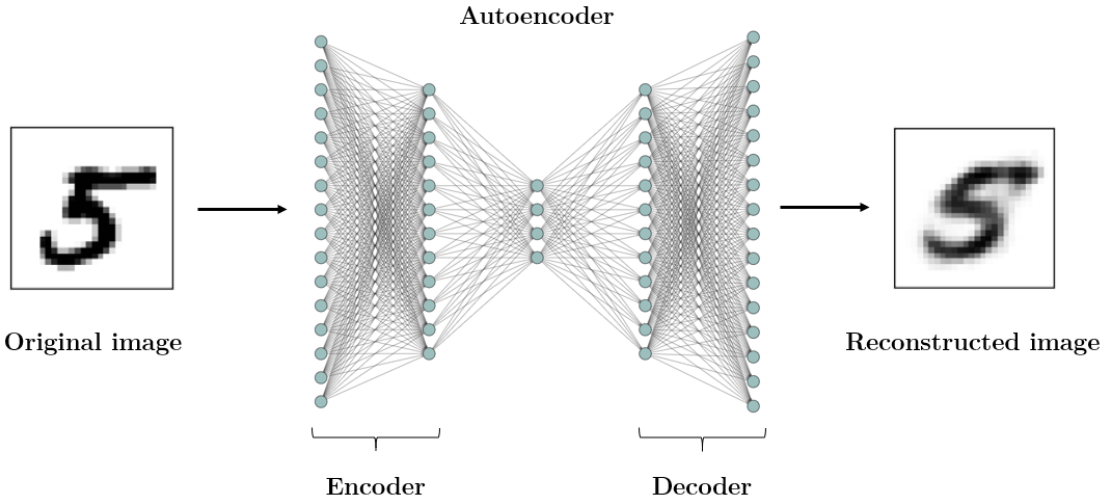


Figure 5.2: Illustration of an autoencoder architecture in action, showing the process of encoding an input image to a compressed (latent) representation and subsequently decoding it to reconstruct the original image.

reconstruction loss, which measures the difference between the original input data and its reconstructed version. One common choice of reconstruction loss is the mean squared error (MSE) loss, which is defined as follows:

$$\mathcal{L}_{MSE} = \frac{1}{M} \sum_{i=1}^{M} \left(x^i - g(f(x^i)) \right)^2 \tag{5.2}$$

where x^i is the i-th input data point, $i = 1, \ldots, M$.
The goal of the autoencoder is to learn a compressed representation that captures the most important features of the input data. Figure 5.2 illustrates an example of such network.
In this case, the QuGAN is trained to generate 4-dimensional latent vectors that can be decoded into a 784-dimensional vector that represents the pixel values of a handwritten image. The whole system is composed of both classical and quantum elements, combining the encoder, the QuGAN used for training, and the decoder utilized for inference.

5.1.3 Quantum Circuit Ansatz

Based on [69], we utilize a parameterized quantum circuit approach by considering three essential types of quantum gates, namely single qubit unitary, dual qubit unitary, and an entanglement unitary. The single qubit unitary applies an $R_y(\theta)$

gate on a single qubit, rotating its position around the $y-$axis by an angle of θ. This gate manipulates a single qubit's superposition. The dual qubit unitary $R_{yy}(\theta)$ rotates two qubits around the $y-$axis by a shared parameter θ. This gate manipulates the superposition of two qubits. The entanglement unitary employed by QuGAN is a $CR_y(\theta)$ gate, which rotates a qubit around the $y-$axis by θ if the control qubit measures 1.

The unitary matrices of these gates are given by

$$R_y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{5.3}$$

$$R_{yy}(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & 0 & 0 & i\sin\left(\frac{\theta}{2}\right) \\ 0 & \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) & 0 \\ 0 & -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) & 0 \\ i\sin\left(\frac{\theta}{2}\right) & 0 & 0 & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{5.4}$$

$$CR_y(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \tag{5.5}$$

The discriminator and generator circuits have been both designed equally, giving them the same ability to learn the ideal quantum state. Figure 5.3 provides a visualization of the architecture of the generator and discriminator, depicting three of the four main components: the discriminator, ancillary qubit, and generator. In the case of a real sample, the trainable rotations in q_5 through q_8 are replaced with the latent vector representing a data point.

5.1.4 Results and Discussion

The autoencoder used in this work consists of three layers in both the encoder and decoder, namely a dense layer with 256 units, a dense layer with 64 units, and a dense layer with 4 units. The input and output layers consist of 784 units each, representing the pixel values of the handwritten images. The total number of trainable parameters (i.e., weights) is 436,116.

To train the autoencoder, we used the Keras deep learning library [84] with the mean squared error (MSE) loss function and Adam optimizer with a learning rate of 10^{-3}. The model was trained for 15 epochs with a batch size of 128 and shuffle set to true. In the context of machine learning, a training epoch is a complete pass through the entire training dataset during the training phase of a model. The number of epochs is a hyperparameter that determines the number of times the

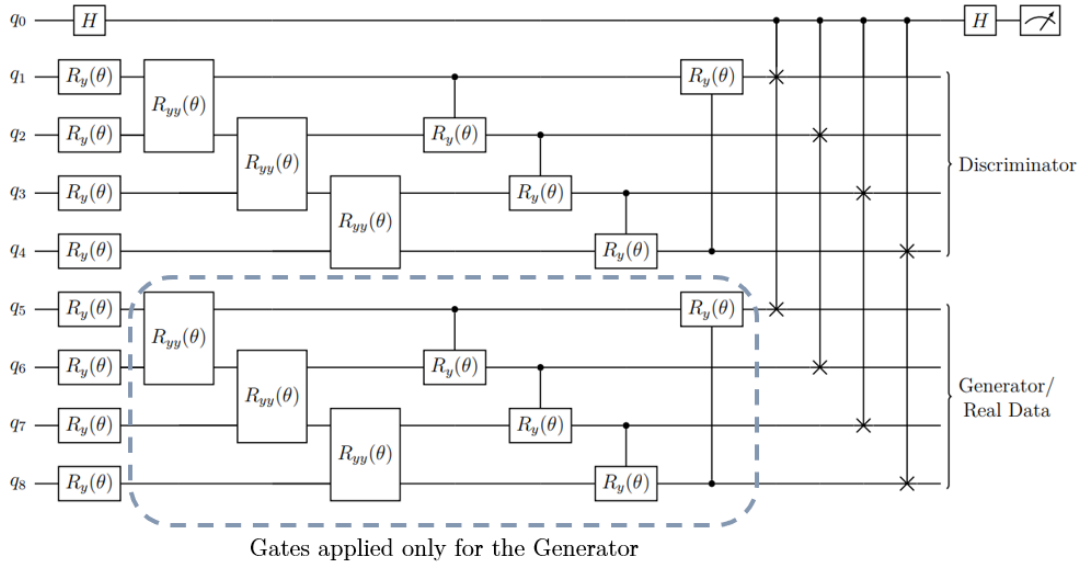


Figure 5.3: Architecture of the implemented QuGAN circuit. Qubits q_1 to q_4 serve as the discriminator, while qubits q_5 to q_5 are used for generator parameters or data loading. Qubit q_0 is used as the ancillary qubit for the *SWAP* test, allowing for measuring quantum state fidelity and enabling inter-circuit communication.

model is trained on the entire dataset. Batch size refers to the number of data points that are processed together in a single forward and backward pass through the neural network during training. A larger batch size means that more data points are processed at once, which can speed up training time, but also requires more memory. Finally, shuffling refers to the randomization of the order of data points within the training dataset before each epoch. This is typically done to prevent the model from learning spurious correlations based on the order of the data points.
The validation data used was the test set consisting of 10,000 handwritten images. After training, the autoencoder achieved a loss value of 0.0282 on the training set and a validation loss of 0.0287. Figure 5.4 shows some examples of the autoencoder's performance on handwritten digit images from the test set. We can see that the autoencoder successfully captures the essential features of the input images and generates high-quality reconstructions.

The QuGAN model was trained using a subset of the MNIST dataset. We employed PennyLane [82] to simulate the quantum circuit of the QuGAN during the training process. To verify that the QuGAN learned the intrinsic probability distribution and that the success of the autoencoder's training did not affect the QuGAN output, we trained the model on a subset of the original data: the images

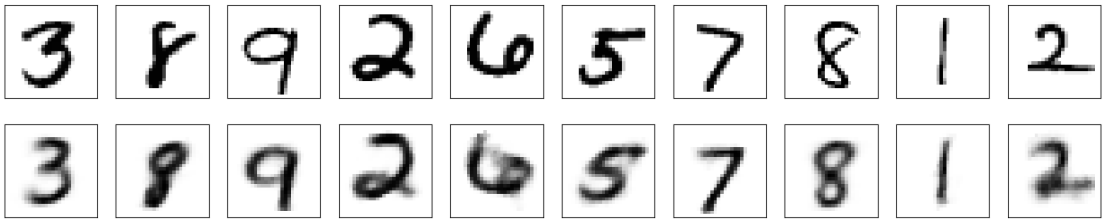


Figure 5.4: Examples of reconstructed images from the MNIST dataset with the trained autoencoder.

labeled as "0" or "5". This approach helped us confirm the QuGAN training success and independence from the autoencoder. The initial parameters for the generator and discriminator circuits were randomly initialized within the range $[0, \pi]$. The hyperparameters of the training procedure, including the batch size, learning rate, and number of iterations for the discriminator and generator circuits, were carefully selected to optimize the performance of the QuGAN. The best results in terms of image quality were obtained with a batch size of 5, a learning rate of 10^{-3}, 50 epochs, 100 iterations for the discriminator on generated data, and 15 iterations for the generator circuit.
In Figure 5.5, we present a selection of 15 images that were produced by the QuGAN using the hyperparameters and training data previously described. These images represent a combination of zeros and fives, and they were obtained by sampling the generator circuit for $q = 30$ times and computing the mean measurement per qubit. To assess the performance of the trained QuGAN, we used the Hellinger Distance [85] as a metric to measure the similarity between the generated distribution and the original dataset distribution. The Hellinger distance is defined as:

$$H(P||Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{n} (\sqrt{p_i} - \sqrt{q_i})^2} \tag{5.6}$$

where n is the size of the support and p_i and q_i are the probabilities assigned by P and Q to the $i-$th element of the support. Intuitively, the Hellinger distance measures how well one distribution can be approximated by the other. It ranges from 0 to 1, where a value of 0 indicates that the two distributions are identical, and a value of 1 indicates that they are completely dissimilar. To compute the Hellinger distance between the generated images and the original dataset, we used 1000 samples generated from the generator circuit. We converted each set of data to a probability distribution by computing a histogram of the data and normalizing it to sum to 1. The resulting Hellinger Distance value obtained was 0.29142. While this value is not zero, it indicates that the generated distribution is somewhat similar to the original dataset, demonstrating that the QuGAN has

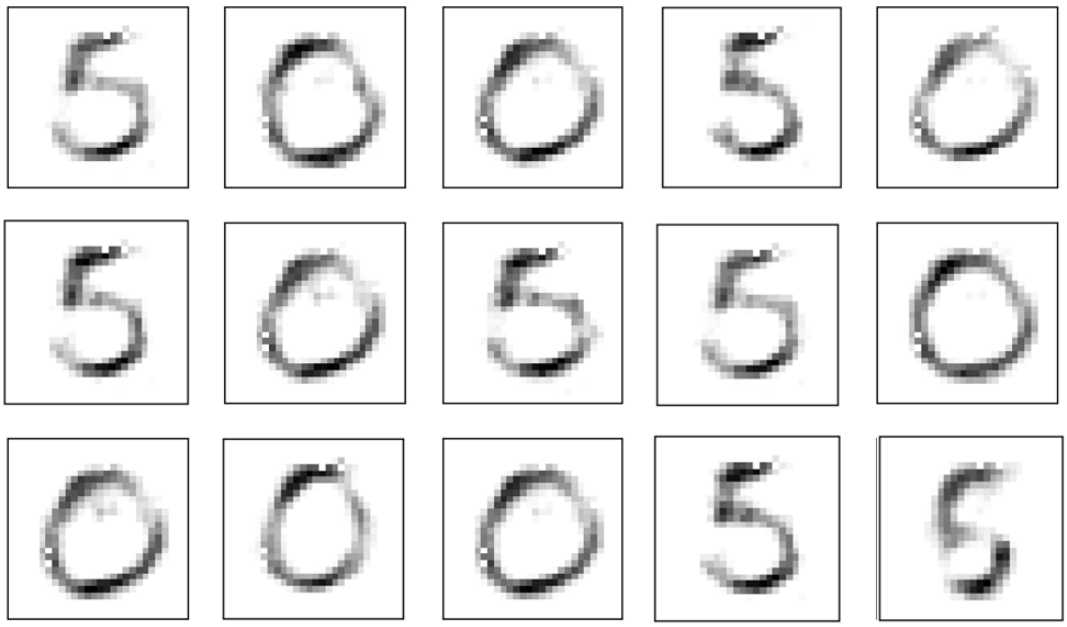


Figure 5.5: Sample of 15 images generated by the QuGAN. The images were produced using the hyperparameters and training data described in the text. The QuGAN achieved nice results with some degree of variation and detail, but also some imperfections. Overall, these results show the potential of QuGANs for image generation with fewer parameters than classical GANs.

learned some of the statistical features of the restricted MNIST dataset, even when trained on a compressed version using the autoencoder preprocessing step. Compared to classical GANs, the QuGAN requires significantly fewer parameters (only 11 rotations for both the generator and the discriminator), which highlights the potential of quantum computing for image generation tasks. As quantum computing technology continues to advance, we expect to see more applications of QuGANs with higher dimensionality and broader applicability.

5.2 qGAN for Loading Student's t-Distributions

5.2.1 Student's t-Distribution and its Generalization

The distributional form of returns on underlying assets is a critical aspect to consider in finance, particularly under derivative valuation theories. Heavy-tailed distributions are commonly used as a substitute for the Normal distribution in financial studies, with the Student's t-distribution being a popular example. These

types of distributions are important in finance and risk management because they can better model situations where extreme events, such as market crashes or major price fluctuations, are more likely to occur. Therefore, when analyzing real market data, heavy-tailed distributions offer a more suitable alternative to the Normal distribution. The Student's t-distribution is particularly useful for fitting the distributions of logarithmic asset returns with ν degrees of freedom, typically falling within the range of $3 \leq \nu \leq 5$ [86].
In its general form, the Student's t-distribution is a bell-shaped distribution that depends on three parameters, namely the degrees of freedom ν, a location parameter μ, and a dispersion parameter σ^2 [87]. The degrees of freedom determine the thickness of the tails, with larger values of ν resulting in thinner tails and smaller values of ν resulting in heavier tails. When ν tends to infinity, the Student's t-distribution approaches the Normal distribution. The location parameter μ indicates the center of the distribution, while the dispersion parameter σ^2 controls the spread of the distribution. In the case where $\mu = 0$ and $\sigma^2 = 1$, the resulting distribution is called the standard Student's t-distribution.
Given a random variable X following a Student-t distribution with parameters ν, μ, and σ^2, i.e., $X \sim \mathrm{St}(\nu, \mu, \sigma^2)$, its probability density function can be expressed as

$$f_{\nu,\mu,\sigma^2}(x) = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{1}{\sqrt{\nu\pi\sigma^2}} \left(1 + \frac{1}{\nu}\frac{(x-\mu)^2}{\sigma^2}\right)^{-\frac{\nu+1}{2}} \tag{5.7}$$

where $\Gamma(\cdot)$ is the gamma function

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt$$

The standard parameters that describe the properties of the Student's t-distribution, namely expected value, standard deviation, and skewness, are given by

$$\begin{aligned} \mathbb{E}[X] &= \mu \\ \mathrm{Sd}[X] &= \frac{\nu}{\nu-2}\sqrt{\sigma^2} \\ \mathrm{Sk}[X] &= 0 \end{aligned} \tag{5.8}$$

and they are defined for $\nu > 1, 2$, and 3 respectively.
Extending the Student's t distribution to a random vector $\mathbf{X} \in \mathbb{R}^N$ is a straightforward process, resulting in the multivariate Student's t-distribution. We denote such distribution by $\mathbf{X} \sim \mathrm{St}(\nu, \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} \in \mathbb{R}^N$ is a location parameter vector that determines the highest point of the distribution, and $\boldsymbol{\Sigma}$ is a symmetric $N \times N$ positive-definite matrix that determines the shape of the distribution around its peak. The degrees of freedom ν determine the importance of the peak of the

distribution in comparison to its tails. The probability density function of the multivariate Student's t-distribution can be expressed as

$$f_{\nu,\boldsymbol{\mu},\boldsymbol{\Sigma}}(\boldsymbol{x}) = (\nu\pi)^{-\frac{N}{2}} \frac{\Gamma\left(\frac{\nu+N}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \left(1 + \frac{1}{\nu}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right)^{-\frac{\nu+N}{2}} \tag{5.9}$$

where $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$. The expected value and the covariance matrix of $\boldsymbol{X}$ are given by

$$\begin{aligned} \mathbb{E}[\boldsymbol{X}] &= \mu \\ \mathrm{Cov}[\boldsymbol{X}] &= \frac{\nu}{\nu-2}\boldsymbol{\Sigma} \end{aligned} \tag{5.10}$$

The generic marginal distribution is still Student's t-distributed; additionally, the Student's t-distribution is invariant under affine transformations. However, unlike the Normal distribution, the conditional distribution of a Student's t-distribution is not generally a Student's t-distribution. As a result, the marginal and conditional distributions cannot be the same, leading to the conclusion that random variables that have a joint Student's t-distribution are not independent [87]. The multivariate Student's t-distribution converges to the multivariate Normal distribution in the limit as $\nu \to \infty$, similar to the one-dimensional case.

5.2.2 qGAN Architecture and Implementation

The design of the parameterized quantum circuit is a critical aspect influencing the performance of any quantum machine learning model. For the implementation of the quantum generator in our qGAN, we employed a hardware-efficient SU(2) 2-local circuit, which consists of layers of single-qubit operations from the special unitary group SU(2) and controlled-X (CX) gates for entanglement. The special unitary group SU(2) comprises 2×2 unitary matrices with determinant equal to 1, such as the Pauli rotation gates. In our implementation, we utilized $R_y(\theta)$ and $R_z(\theta)$ single-qubit gates and a *reversed linear* entanglement strategy, whereby qubit i is entangled with qubit $i+1$, $i \in \{n-2, n-3, \ldots, 1, 0\}$. This configuration provides the same unitary transformation as the *full* entanglement strategy, where each qubit is entangled with all the others, but requires fewer entangling gates, making it a more efficient choice [88]. The circuit starts with a layer of $R_y(\theta)$ and $R_z(\theta)$ rotations, followed by k repetitions of alternating CX gates and additional $R_y(\theta)$ and $R_z(\theta)$ layers, as shown in Figure 5.6. The parameter k is closely linked to a crucial property of a quantum circuit known as *circuit depth*. Circuit depth quantifies the number of layers of quantum gates executed in parallel to perform the computation prescribed by the circuit itself [89]. As quantum gates require time for execution, the depth of a circuit is closely related to the approximate execution time on a quantum computer. Notably, a shallow circuit is desirable for efficient

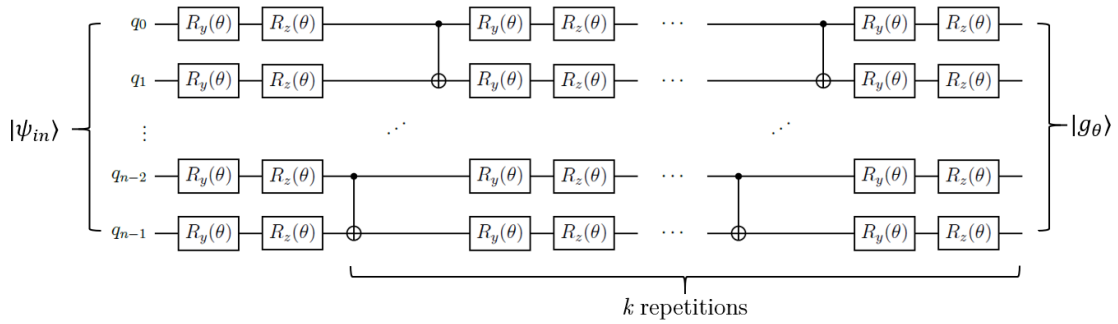
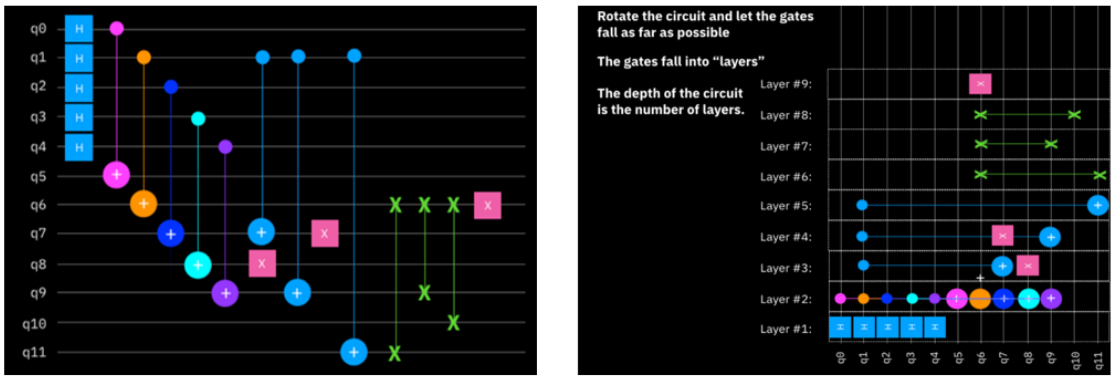


Figure 5.6: Variational form of the quantum generator. After a first layer of $R_y(\theta)$ and $R_z(\theta)$ rotations, the circuit consists of k alternating repetitions of CX layers and further layers of $R_y(\theta)$ and $R_z(\theta)$ gates.

quantum computation as it reduces the execution time and the risk of error due to decoherence. Therefore, circuit depth serves as an important metric to determine if a quantum circuit can run on a specific quantum device. Mathematically, circuit depth is defined as the length of the longest path in a directed acyclic graph (DAG) that represents the evolution over time of a qubit's state. However, to provide an intuitive understanding of circuit depth, one can draw an analogy to the popular game Tetris [90], where the falling blocks in different shapes and layers correspond to the quantum gates executing in parallel layers. By grasping this analogy, which is illustrated in Figure 5.7, one can visualize how the circuit depth impacts the execution time and resource requirements of a quantum circuit.

The variational circuit has a total of $(k + 2)n$ parametrized single-qubit gates and kn two-qubit gates if it acts on n qubits. Increasing the depth of the circuit, similar to adding layers in deep neural networks, allows the circuit to represent more complex structures and results in an increase in the number of parameters. The hardware-efficient SU(2) 2-local circuit design offers several justifications for its use in training a qGAN model. Firstly, it addresses hardware constraints by minimizing the number of physical gates required for implementation, making it well-suited for near-term quantum devices with limited connectivity and high error rates. Moreover, the hardware-efficient SU(2) 2-local circuit design sets the foundation for scalability. As quantum devices continue to advance, larger and more complex systems are being developed. By employing a circuit design optimized for hardware efficiency, the qGAN model can be readily scaled up to larger quantum systems in the future, enabling the transfer of training techniques to more advanced hardware platforms.

To perform our tests, we used the Qiskit library [81], which offers the option to run quantum circuits on different types of devices. These device types include:



(a) (b)

Figure 5.7: Analogy with Tetris game to explain the concept of circuit depth, reproduced from [89]. The depth of a quantum circuit can be compared to the height of a Tetris game board, and adding more layers of gates to a circuit is like stacking more Tetris pieces on top of each other. In the same way that it takes more time and effort to clear a higher stack of Tetris pieces, a deeper quantum circuit requires more time and resources to execute. Figure (b) summarizes the steps involved in the computation of the depth for the circuit shown in Figure (a), which is equal to 9.

1. the *statevector simulator*, which computes the exact state of the circuit under ideal conditions;
2. the *noiseless simulator*, which simulates ideal conditions, but provides an approximate output based on the number of measurements specified (a.k.a. *shots*);
3. the *noisy simulator*, which introduce a noise model to the gate executions;
4. *real quantum hardware*, which utilizes IBM's devices available in the cloud.

Initial experiments indicated that the convergence behavior of the training process differs significantly between results obtained from the statevector simulator and the noiseless simulator. This is because the limited number of measurements in the noiseless simulation approximation causes stronger oscillations in the neural network. As a result, a larger parameter space must be evaluated to determine the gradient and prevent local flatness [73]. In the noisy simulator, a basic noise model generated automatically by Qiskit was utilized, incorporating information extracted from real quantum systems snapshots. These snapshots encompass important details regarding the quantum system, including the coupling map, basis gates, and qubit properties such as the relaxation time constants and the readout error

Figure 5.8: Histogram of the initial training data, consisting of 20,000 samples drawn from a standard Student's t-distribution with $\nu = 3$ degrees of freedom.

probability. Specifically, the IBM Quantum system known as "ibmq_manila", which consists of 5 qubits, was considered for this purpose. The difference between the noiseless and noisy simulators was found to be minimal, which may be due to neural networks' ability to handle noise and the shallow depth of the implemented circuits. However, the execution time is significantly longer in noisy simulations because it requires the artificial replication of noise in the simulator. Given the current state of technology, it is challenging to conduct extensive testing of hybrid algorithms on quantum hardware due to the significant time and effort required to connect to the backends. This challenge is amplified in hybrid approaches, where iterative querying of the quantum computer is necessary, leading to an increase of the total computational time required by the algorithm. Therefore, we opted to perform our training process on the noiseless simulator as it offers the best balance between execution time and result significance for our purposes.

5.2.3 Results and Discussion

In this section, we present the results of an extensive simulation study that investigates various training settings for qGANs. The study focuses on generating two different target distributions, namely univariate and bivariate standard Student's t-distributions. For the univariate case, we used a quantum generator that operates on $n = 3$ and $n = 4$ qubits, enabling the representation of $2^3 = 8$ and $2^4 = 16$ discretized values, respectively. Figure 5.8 shows a histogram of the initial

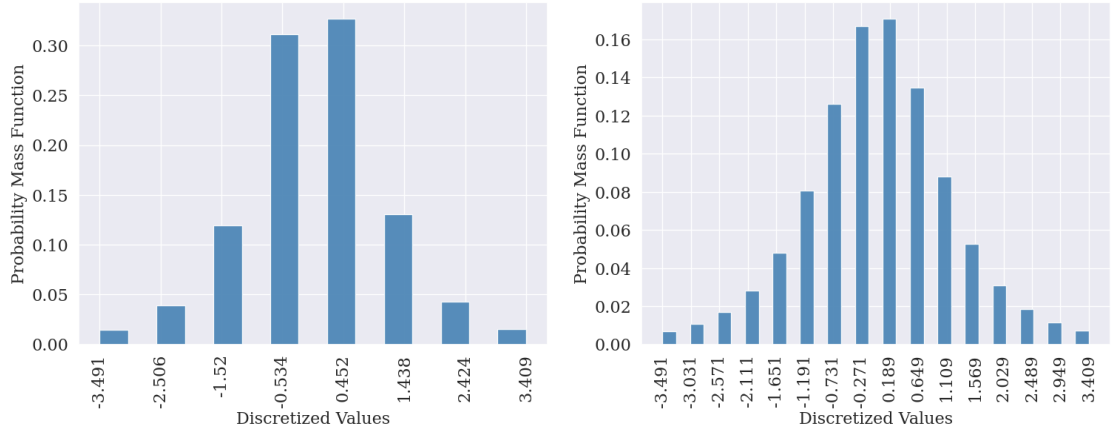


Figure 5.9: Histograms of the discretized training data for the $n = 3$ (left) and $n = 4$ (right) cases, obtained from a standard Student's t-distribution with $\nu = 3$ degrees of freedom. The discretized values were obtained using the affine map (4.17) and selecting the 2nd and 98th percentiles of the dataset as *low* and *high* values, respectively. The resulting distributions preserve the heavy-tailed nature of the target distribution while maintaining a bell shape.

training data, which consists of 20,000 samples drawn from a standard Student's t-distribution with $\nu = 3$ degrees of freedom. To discretize the samples, we applied the affine map (4.17) and selected the 2nd and 98th percentiles of the dataset as *low* and *high* values, respectively. This approach, chosen considering the limited number of qubits used in our study, not only allowed to preserve the heavy-tailed nature of the distribution but also to maintain a bell shape. The histograms of the discretized training data for $n = 3$ and $n = 4$ cases are illustrated in Figure 5.9. The initial state $|\psi_{in}\rangle$ of the generator was prepared in accordance with a discrete uniform distribution, necessitating the use of $n = 3$ and $n = 4$ Hadamard gates, i..e., one per qubit. The discriminator, implemented using PyTorch [91], is a standard deep neural network consisting of two hidden layers. Specifically, the intermediate nodes use the Leaky ReLU activation function with a negative slope of 0.2, while the output node uses sigmoid activation.
To train the qGAN, we used AMSGrad (see Appendix A) with an initial learning rate of 10^{-3}. This optimization technique is robust for non-stationary objective functions and noisy gradients due to the use of first and second momentum terms. These parameters were set to $(\beta_1, \beta_2) = (0.7, 0.99)$ for both the discriminator and the generator. During each training epoch, the generator's objective is to produce probabilities that the discriminator would classify as probabilities from the real training data distribution. To obtain such probabilities, the generator output state $|g_\theta\rangle$ is measured based on the specified number of shots s. On the other hand, the

Figure 5.10: An example of a qGAN run that appears to have converged based on loss function analysis, but that still results in a suboptimal approximation of the target distribution.

discriminator aims to distinguish between the original data distribution and the probabilities generated by the generator.
In the classical GAN literature, it is acknowledged that the convergence of the model cannot solely be determined by the analysis of loss functions. While a run may appear to have converged, it is still possible to obtain a suboptimal approximation of the target distribution, as shown in Figure 5.10. This can happen due to two main reasons: firstly, the optimizer may find a solution that is not the global optimum but a local minimum, resulting in an imperfect approximation. Secondly, the chosen ansatz may not have the capability to reach a point that is close enough to the target distribution, leading to an inadequate representation. Hence, it becomes crucial to incorporate additional metrics besides loss functions for a comprehensive evaluation of the generated samples and overall performance of the qGAN model. The *Kolmogorov-Smirnov statistic* [92] and *relative entropy* [93] are commonly used metrics to assess the training performance of quantum representations of probability distributions underlying the training data. The Kolmogorov-Smirnov statistic compares the empirical cumulative distribution functions $P(X \leq x)$ and $Q(X \leq x)$ of two probability distributions P and Q. It quantifies the largest vertical distance between these CDFs, thus serving as a measure of the goodness-of-fit between the distributions, and its expression is given by

$$D_{KS}(P||Q) = \sup_{x \in X} |P(X \leq x) - Q(X \leq x)| \tag{5.11}$$

Assuming the null hypothesis that the probability distribution obtained from $|g_\theta\rangle$ is equal to the underlying probability distribution of the training data, the KS statistic determines whether the null hypothesis should be rejected or not at a

specified confidence level $(1 - \alpha)$. To perform the test, we collected 2,000 samples from both distributions and chose a significance level of $\alpha = 0.05$. If the resulting p-value is less than 0.05, we reject the null hypothesis in favor of the alternative hypothesis, indicating that the two distributions are different. A larger KS statistic leads to a smaller p-value, providing evidence against the null hypothesis and supporting the alternative hypothesis.
The relative entropy, also known as Kullback-Leibler divergence, is an additional measure that can be utilized to assess the similarity between two discrete probability distributions $P(x)$ and $Q(x)$. This entropy-based measure is defined as follows:

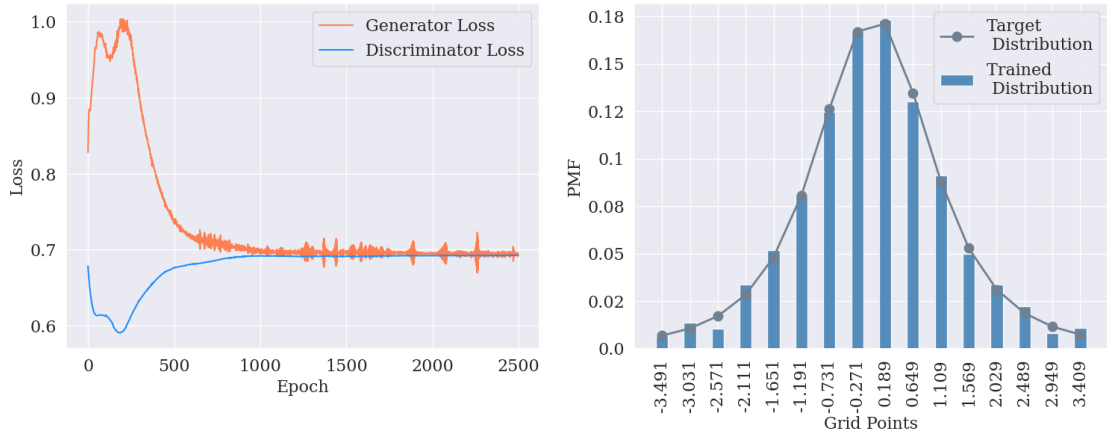
$$D_{RE}(P||Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right) \tag{5.12}$$

The relative entropy is always non-negative, i.e., $D_{RE}(P||Q) \geq 0$, and it equals zero when $P(x) = Q(x)$ for all values of x.
To implement a qGAN for a particular application, it is crucial to determine the optimal configuration that allows the trained circuit to closely approximate the target distribution. Our testing involved experimenting with various combinations of hyperparameters, as summarized in Table 5.1. For the $n = 3$ qubit case, the optimal configuration was achieved with a generator consisting of $k = 1$ repetition, $(h_1, h_2) = (50,20)$ nodes for the discriminator network, a learning rate of $\eta = 10^{-3}$ for both the generator and discriminator, a maximum of $E_{max} = 2{,}000$ epochs, and $s = 10{,}000$ shots for the generator circuit. Similarly, for $n = 4$, the optimal configuration was achieved with $k = 2$ repetitions, $(h_1, h_2) = (50,25)$ nodes, a learning rate of $\eta = 10^{-3}$, maximum epochs of $E_{max} = 2{,}500$, and $s = 10{,}000$ shots. Figures 5.11(a) and 5.11(b) show the losses of both the discriminator and generator, as well as the resulting trained distributions for the cases of $n = 3$ and

Parameters	**Tested Values**
n	3, 4
k	1, 2, 3, 4
(h_1, h_2)	(32,16), (50,20), (50,25), (50,30), (64, 32)
η	10^{-4}, $5 \cdot 10^{-4}$, 10^{-3}
E_{max}	1000, 1500, 2000, 2500
s	2000, 5000, 10000

Table 5.1: Tested hyperparameters and their corresponding values. The table includes the number of qubits n, the number of repetitions of quantum layers k, the number of nodes of discriminator hidden layers (h_1, h_2), the learning rate η, the maximum number of epochs E_{max}, and the number of shots for the quantum circuit s.

(a)



(b)

Figure 5.11: Experimental results for univariate standard Student's t-distributions obtained by training the qGAN with optimal hyperparameter configurations for $n = 3$ (Figure (a)) and $n = 4$ (Figure (b)) qubits. The figure displays the losses of both the discriminator and generator, as well as the resulting trained distributions. These results demonstrate the effectiveness of the qGAN approach in approximating the target distributions in both cases. The optimal hyperparameter configurations were determined through a systematic exploration of various combinations, as summarized in Table 5.1.

$n = 4$ qubits, respectively. These experimental results indicate that the qGAN algorithm successfully converged for both $n = 3$ and $n = 4$ qubits. The losses of the discriminator and generator reached stable values, suggesting an effective training process. Furthermore, the resulting trained distributions exhibited a high degree

of fidelity to the target distributions, demonstrating the effectiveness of qGAN in generating accurate approximations.
After completing the training process, we evaluated the performance of the qGAN models based on the metrics defined earlier. Specifically, for $n = 3$ qubits, the qGAN generated a distribution with a relative entropy of $D_{RE} = 0.00074$ and a KS statistic of $D_{KS} = 0.01899$, with a corresponding p-value equal to 0.84128. Similarly, for the $n = 4$ qubit case, the qGAN produced a distribution with a relative entropy of $D_{RE} = 0.01057$ and a KS statistic of $D_{KS} = 0.01382$, with a corresponding p-value of 0.98740. The low values of the relative entropy and KS statistic, as well as the high p-values, further demonstrate that the qGAN approach was effective in generating accurate approximations for our given target distributions.

Bivariate Case

For the bivariate standard Student's t-distribution, we discretized each dimension using 3 qubits, resulting in an 8×8 grid of data points. The (continuous) bivariate probability density function was evaluated at each grid point, and the resulting probabilities were then normalized to ensure a consistent discrete distribution. The obtained discretized PDF is visualized in Figure 5.12. Similar to the univariate scenario, training the qGAN for the bivariate case required testing various combinations of hyperparameters to achieve satisfactory results.
Table 5.2 summarizes the parameter values experimented with during the training process. Notably, the optimal configuration for the qGAN was achieved with a generator consisting of $k = 7$ repetitions, $(h_1, h_2) = (50,25)$ nodes for the discriminator network, a learning rate of $\eta = 10^{-3}$ for both the generator and discriminator, a maximum of $E_{max} = 2{,}000$ epochs, and $s = 10{,}000$ shots for the generator

Parameters	**Tested Values**
(n_1, n_2)	(3, 3)
k	3, 4, 5, 6, 7
(h_1, h_2)	(32,16), (50,20), (50,25), (50,30), (64, 32)
η	10^{-4}, $5 \cdot 10^{-4}$, 10^{-3}
E_{max}	1,000, 1,500, 2,000, 2,500
s	2,000, 5,000, 10,000

Table 5.2: Tested hyperparameters and their corresponding values for the bivariate scenario. (n_1, n_2) denotes the number of qubits used to discretize each dimension of the bivariate Student's t-distribution.

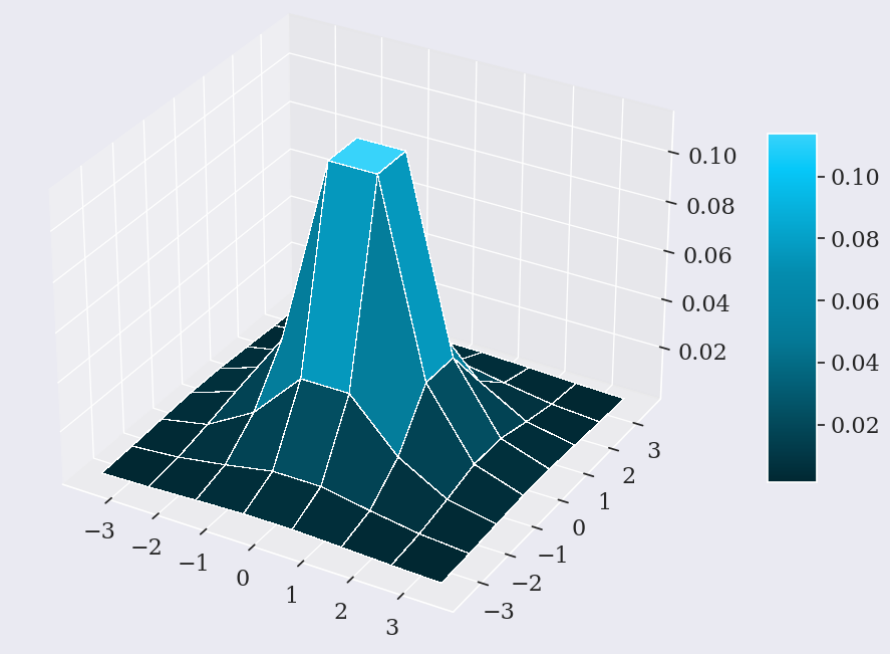


Figure 5.12: Discretized probability density function (PDF) for the bivariate standard Student's t-distribution with $\nu = 3$ degrees of freedom. The distribution was discretized using a 3-qubit representation for each dimension, which yielded an 8×8 grid of data points over the interval $(-3.5, 3.5)$.

circuit. The resulting losses of the discriminator and generator, along with the corresponding trained distribution, are illustrated in Figure 5.13. Regarding the number of repetitions in the generator circuit, we noticed that achieving satisfactory results in the bivariate case needed a higher number of repetitions compared to the univariate scenario. This increased requirement can be attributed to the fact that in the bivariate setting, the generator needs to learn 64 probabilities, which demands a more complex representation.
Upon comparing the generated probabilities to the real probabilities, we observed differences ranging from magnitudes of 10^{-3} to 10^{-5}. These variations indicate that the qGAN successfully learned the underlying bivariate distribution and approximated the true distribution with good accuracy. In evaluating the qGAN's performance for the bivariate case, we focused on using the relative entropy as evaluation metric. Unlike the univariate scenario, we did not perform a Kolmogorov-Smirnov test for goodness-of-fit. This choice was made considering that most existing goodness-of-fit tests are mostly developed for univariate distributions, with limited options available for multivariate distributions, except for cases such as multivariate normality. The obtained relative entropy value of $D_{RE} = 0.02487$

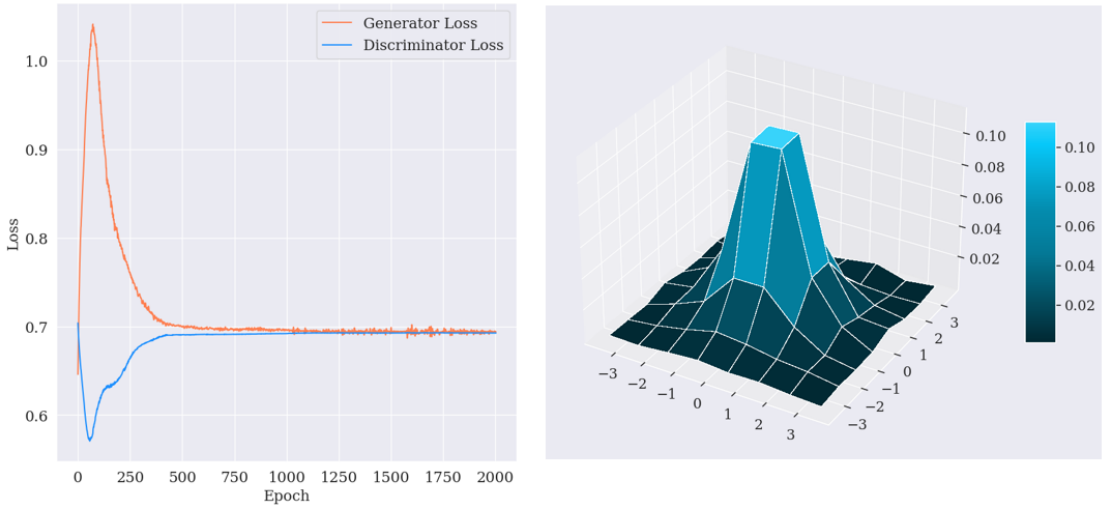


Figure 5.13: Experimental results for the bivariate standard Student's t-distributions obtained by training the qGAN with optimal hyperparameter configuration. The differences between generated probabilities and real probabilities range from from 10^{-3} to 10^{-5} in magnitude, suggesting that the qGAN successfully learned the underlying bivariate distribution.

indicated a relatively small divergence, suggesting that the qGAN performed well in approximating the true bivariate distribution. These findings highlight the potential of qGANs for modeling complex multivariate distributions and provide a foundation for further research and exploration in this field.

5.3 Conclusions

In this chapter, we investigated some practical applications of the quantum machine learning models defined in Chapter 4 using data with similar complexity to real-world applications. Firstly, we evaluated the performance of the QuGAN model in the classical task of image generation using the MNIST dataset. Despite the training was performed on a compressed version of the dataset obtained through an autoencoder preprocessing step, the generated distribution exhibited similarity to the original dataset. This indicates that the QuGAN effectively captured some of the statistical features of the restricted MNIST dataset. Notably, the QuGAN achieved this performance with significantly fewer parameters compared to classical GANs. Furthermore, the experimental results highlighted the effectiveness of the qGAN algorithm in loading Student's t-distributions, both in the univariate and bivariate cases. The convergence of the qGAN algorithm was successfully achieved, as indicated by the stable values reached for the losses of the discriminator and

generator. The resulting trained distributions exhibited a remarkable level of fidelity to the target distributions, providing a foundation for future research and application of quantum machine learning techniques in the field of economics and finance.

Chapter 6

Conclusions

We began our work by providing an overview on quantum computers, emphasizing their distinctive properties rooted in quantum mechanics. We discussed the concept of quantum supremacy, highlighting that quantum devices are not intended to surpass classical computers in every aspect. This led us to explore the areas where researchers believe quantum computers can bring significant advancements, with a particular focus on machine learning, which serves as core of this work. Furthermore, we recognized the current period as the NISQ (Noisy Intermediate-Scale Quantum) era, characterized by the availability of quantum devices with a limited number of qubits and inherent noise, highlighting the challenges and constraints associated with working on current quantum hardware.

We established the theoretical foundations of quantum computing, covering topics such as qubits, state superposition, the Bloch sphere, density operator, observables, and the dynamics of closed quantum systems. We explored single-qubit gates, two-qubit gates, gate decompositions, and the concept of entanglement. We discussed the physical realizations of quantum computers, focusing on superconducting qubits, photonic qubits, and trapped ion qubits. The significance of quantum simulators for algorithm development and testing was highlighted. We introduced Parametrized Quantum Circuits (PQCs) as versatile tools for machine learning and optimization applications. We examined hybrid quantum-classical algorithms like the Variational Quantum Eigensolver (VQE) and the Quantum Approximate Optimization Algorithm (QAOA). Data encoding methods were explored, emphasizing their impact on algorithm runtime. We further explored quantum-assisted machine learning, which combines classical data with quantum computing techniques to enhance computational capabilities and tackle complex learning tasks.

The scope of this work was to explore several quantum extensions of the classical Generative Adversarial Networks (GANs) and investigate their potential benefits in the field of quantum machine learning. Classical GANs have emerged as a powerful class of generative models that have demonstrated remarkable capabilities

in generating realistic images, texts, and other types of data. In this work, we investigated two versions of quantum GANs: QuGAN and qGAN for distribution learning. Both of these models make use of classical data, but with distinct architectures. Notably, QuGAN employs a fully quantum approach, where both the generator and discriminator are represented by PQCs. On the other hand, qGAN for distribution learning adopts a hybrid architecture, combining a quantum generator with a classical discriminator network.
QuGAN introduces the utilization of quantum state fidelity for both the quantum discriminator and generator. This approach enables the incorporation of quantum-based loss functions, computed using a swap test on qubits. This architecture exhibits stable convergence and requires significantly reduced parameter sets compared to its classical counterparts. The second quantum GAN we investigated centers around the efficient loading of approximate data onto quantum hardware. By reducing the gate complexity, this method offers a more efficient and scalable solution for loading data onto quantum systems. Additionally, the corresponding quantum channel is implemented using a gate-based quantum algorithm, allowing for seamless integration into other gate-based quantum algorithms.
The first experiment we conducted in this study aimed to assessing the performance of the QuGAN model in the task of image generation using the classical MNIST dataset. Due to the current limitations of quantum hardware, generating high-resolution images with a large number of qubits is infeasible. To address this, we employed an autoencoder, a neural network that leverages its nonlinear characteristics for advanced dimensionality reduction. This enabled us to reduce the dimensionality of the data and facilitate the training process. To validate the effectiveness of QuGAN in learning the underlying probability distribution independently from the autoencoder, we trained the model exclusively on images labeled as "0" or "5". This approach confirmed the success of training and demonstrated the model's independence from the autoencoder. The results obtained showed that the QuGAN effectively learned some of the statistical features of the restricted MNIST dataset. The generated images of the digits "0" and "5" exhibited a certain level of variation and detail, although some imperfections were present. Notably, compared to classical GANs, the QuGAN required significantly fewer parameters, demonstrating promising performance with reduced computational requirements.
In the second experiment, we investigated the effectiveness of the qGAN in loading univariate and bivariate Student's t-distributions. These distributions are particularly important in finance and risk management, as they provide a more accurate modeling of situations where extreme events are more likely to occur due to their heavy-tailed nature. To obtain satisfactory results, an extensive exploration of hyperparameter configurations was conducted. Indeed, similar to classical machine learning, it is not initially evident which model structure is most appropriate for a given problem, nor which training strategy will yield the optimal results. Despite

the necessity for careful tuning, the qGAN algorithm demonstrated successful convergence in both cases, as evidenced by the stable values reached for the discriminator and generator losses. The resulting trained distributions exhibited a remarkable level of fidelity to the target distributions, both in the univariate and bivariate cases. Moreover, the metrics utilized to evaluate the performance of the trained distributions showed satisfactory results.
In conclusion, the results obtained from our experiments provide encouraging insights and motivate further exploration in the field of quantum machine learning. They shed light on the current limitations of quantum devices and simulators, particularly in terms of the limited number of qubits that can be effectively utilized. While these limitations are evident, it is crucial to recognize the importance of ongoing research in this field. By enhancing the quality of quantum hardware, we can tackle increasingly complex tasks across various domains, ultimately improving both the quality and speed of computations.

Bibliography

[1] Peter W. Shor. «Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer». In: *SIAM J. Comput.* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397. URL: https://doi.org/10.1137/S0097539795293172.

[2] Lov K. Grover. «A Fast Quantum Mechanical Algorithm for Database Search». In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. URL: https://doi.org/10.1145/237814.237866.

[3] David G. Cory, Amr F. Fahmy, and Timothy F. Havel. «Ensemble quantum computing by NMRspectroscopy». In: *Proceedings of the National Academy of Sciences* 94.5 (1997), pp. 1634–1639. URL: https://www.pnas.org/doi/abs/10.1073/pnas.94.5.1634.

[4] Neil A. Gershenfeld and Isaac L. Chuang. «Bulk Spin-Resonance Quantum Computation». In: *Science* 275.5298 (1997), pp. 350–356. URL: https://www.science.org/doi/abs/10.1126/science.275.5298.350.

[5] Lieven MK Vandersypen, Matthias Steffen, Gregory Breyta, Costantino S Yannoni, Mark H Sherwood, and Isaac L Chuang. «Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance». In: *Nature* 414.6866 (2001), pp. 883–887.

[6] Isaac L. Chuang, Neil Gershenfeld, and Mark Kubinec. «Experimental Implementation of Fast Quantum Searching». In: *Phys. Rev. Lett.* 80 (15 Apr. 1998), pp. 3408–3411. URL: https://link.aps.org/doi/10.1103/PhysRevLett.80.3408.

[7] Ming Gong, Shiyu Wang, and et al. «Quantum walks on a programmable two-dimensional 62-qubit superconducting processor». In: *Science* 372.6545 (May 2021), pp. 948–952. URL: https://doi.org/10.1126%5C%2Fscience.abg7812.

[8] Frank Arute, Kunal Arya, and et al. «Quantum Supremacy using a Programmable Superconducting Processor». In: *Nature* 574 (2019), pp. 505–510. URL: https://www.nature.com/articles/s41586-019-1666-5.

[9] *On "Quantum Supremacy"*. https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/.

[10] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. USA: Oxford University Press, Inc., 2007. ISBN: 0198570007.

[11] Gregg Jaeger. *Entanglement, Information, and the Interpretation of Quantum Mechanics*. 1st. Springer Publishing Company, Incorporated, 2009. ISBN: 3540921273.

[12] Fabio Nicola and Luigi Rodino. «Spectral Theory». In: *Global Pseudo-Differential Calculus on Euclidean Spaces*. Basel: Birkhäuser Basel, 2010, pp. 153–201. DOI: 10.1007/978-3-7643-8512-5_6.

[13] S. E. Rasmussen and N. T. Zinner. «Simple implementation of high fidelity controlled-*i*swap gates and quantum circuit exponentiation of non-Hermitian gates». In: *Phys. Rev. Res.* 2 (3 July 2020), p. 033097. URL: https://link.aps.org/doi/10.1103/PhysRevResearch.2.033097.

[14] Deanna M. Abrams, Nicolas Didier, Blake R. Johnson, Marcus P. da Silva, and Colm A. Ryan. «Implementation of XY entangling gates with a single calibrated pulse». In: *Nature Electronics* 3.12 (Nov. 2020), pp. 744–750. URL: https://doi.org/10.1038%5C%2Fs41928-020-00498-1.

[15] Morten Kjaergaard, Mollie E. Schwartz, Jochen Braumüller, Philip Krantz, Joel I.-J. Wang, Simon Gustavsson, and William D. Oliver. «Superconducting Qubits: Current State of Play». In: *Annual Review of Condensed Matter Physics* 11.1 (2020), pp. 369–395. URL: https://doi.org/10.1146/annurev-conmatphys-031119-050605.

[16] *Rigetti Computing*. https://www.rigetti.com/about-rigetti-computing.

[17] Simon J Devitt, William J Munro, and Kae Nemoto. «Quantum error correction for beginners». In: *Reports on Progress in Physics* 76.7 (June 2013), p. 076001. URL: https://dx.doi.org/10.1088/0034-4885/76/7/076001.

[18] David P. DiVincenzo. «The Physical Implementation of Quantum Computation». In: *Fortschritte der Physik* 48.9-11 (2000), pp. 771–783. DOI: https://doi.org/10.1002/1521-3978(200009)48:9/11<771::AID-PROP771>3.0.CO;2-E.

[19] Sanskriti Joshi and Sajjad Moazeni. *Scaling up Superconducting Quantum Computers with Cryogenic RF-photonics*. 2022. arXiv: 2210.15756 [quant-ph].

[20] *Quantum-Centric Supercomputing: The Next Wave of Computing*. https://research.ibm.com/blog/next-wave-quantum-centric-supercomputing.

[21] *IBM's roadmap for building an open quantum software ecosystem*. https://research.ibm.com/blog/quantum-development-roadmap.

[22] Rajeev Acharya and et al. *Suppressing quantum errors by scaling a surface code logical qubit*. 2023. URL: https://doi.org/10.1038/s41586-022-05434-1.

[23] *Google Plans to Build a Practical Quantum Computer by 2029 at New Center*. https://www.insidequantumtechnology.com/news-archive/google-plans-to-build-a-practical-quantum-computer-by-2029-at-new-center/.

[24] E Knill, R Laflamme, and G J Milburn. «A scheme for efficient quantum computation with linear optics». In: *Nature* 409.6816 (2001), pp. 46–52. ISSN: 0028-0836. URL: https://doi.org/10.1038/35051009.

[25] Joshua W. Silverstone, Damien Bonneau, Jeremy L. O'Brien, and Mark G. Thompson. «Silicon Quantum Photonics». In: *IEEE Journal of Selected Topics in Quantum Electronics* 22.6 (2016), pp. 390–402. DOI: 10.1109/JSTQE.2016.2573218.

[26] *Xanadu*. https://www.xanadu.ai/.

[27] *PsiQuantum and GLOBALFOUNDRIES to Build the World's First Full-scale Quantum Computer*. https://psiquantum.com/news/psiquantum-and-globalfoundries-to-build-the-worlds-first-full-scale-quantum-computer.

[28] *IonQ Forte: The First Software-Configurable Quantum Computer*. https://ionq.com/resources/ionq-forte-first-configurable-quantum-computer.

[29] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik, and Jeremy L. O'Brien. «A variational eigenvalue solver on a photonic quantum processor». In: *Nature Communications* 5.1 (July 2014). URL: https://doi.org/10.1038%5C%2Fncomms5213.

[30] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv: 1411.4028 [quant-ph].

[31] Marcello Benedetti, Erika Lloyd, Stefan Sack, and Mattia Fiorentini. «Parameterized quantum circuits as machine learning models». In: *Quantum Science and Technology* 4.4 (Nov. 2019), p. 043001. DOI: 10.1088/2058-9565/ab4eb5.

[32] Shun-ichi Amari. «Backpropagation and stochastic gradient descent method». In: *Neurocomputing* 5.4 (1993), pp. 185–196. ISSN: 0925-2312. DOI: https://doi.org/10.1016/0925-2312(93)90006-O.

[33] Walter Greiner, Ludwig Neise, and Horst Stöcker. «The Models of Ising and Heisenberg». In: *Thermodynamics and Statistical Mechanics*. New York, NY: Springer New York, 1995, pp. 436–456. DOI: 10.1007/978-1-4612-0827-3_18.

[34] M. Schuld and F. Petruccione. *Machine Learning with Quantum Computers*. Quantum Science and Technology. Springer International Publishing, 2021. ISBN: 9783030830984. URL: https://books.google.it/books?id=-N5IEAAAQBAJ.

[35] Sheir Yarkoni, Elena Raponi, Thomas Bäck, and Sebastian Schmitt. «Quantum annealing for industry applications: introduction and review». In: *Reports on Progress in Physics* 85.10 (Sept. 2022), p. 104001. DOI: 10.1088/1361-6633/ac8c54.

[36] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. *Adiabatic Quantum Computation is Equivalent to Standard Quantum Computation*. 2005. arXiv: quant-ph/0405098 [quant-ph].

[37] Masuo Suzuki. «General theory of higher-order decomposition of exponential operators and symplectic integrators». In: *Physics Letters A* 165.5 (1992), pp. 387–395. ISSN: 0375-9601. DOI: https://doi.org/10.1016/0375-9601(92)90335-J.

[38] *Quantum Approximate Optimization Algorithm explained*. https://www.mustythoughts.com/quantum-approximate-optimization-algorithm-explained.

[39] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G. Rieffel. «Quantum approximate optimization algorithm for MaxCut: A fermionic view». In: *Phys. Rev. A* 97 (2 Feb. 2018), p. 022304. DOI: 10.1103/PhysRevA.97.022304.

[40] Minh Do, Zhihui Wang, Bryan O'Gorman, Davide Venturelli, Eleanor Rieffel, and Jeremy Frank. *Planning for Compilation of a Quantum Algorithm for Graph Coloring*. 2020. arXiv: 2002.10917 [quant-ph].

[41] Frank Leymann and Johanna Barzen. «The bitter truth about gate-based quantum algorithms in the NISQ era». In: *Quantum Science and Technology* 5.4 (Sept. 2020), p. 044007. DOI: 10.1088/2058-9565/abae7d.

[42] Ryan LaRose and Brian Coyle. «Robust data encodings for quantum classifiers». In: *Phys. Rev. A* 102 (3 Sept. 2020), p. 032420. DOI: 10.1103/PhysRevA.102.032420.

[43] Seth Lloyd. «Universal Quantum Simulators». In: *Science* 273.5278 (1996), pp. 1073–1078. DOI: 10.1126/science.273.5278.1073.

[44] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. «Efficient Quantum Algorithms for Simulating Sparse Hamiltonians». In: *Communications in Mathematical Physics* 270.2 (2007), pp. 359–371. ISSN: 1432-0916. DOI: 10.1007/s00220-006-0150-x.

[45] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. *Quantum algorithms for supervised and unsupervised machine learning*. 2013. arXiv: 1307.0411 [quant-ph].

[46] Esma Aïmeur, Gilles Brassard, and Sébastien Gambs. «Machine Learning in a Quantum World». In: *Advances in Artificial Intelligence*. Ed. by Luc Lamontagne and Mario Marchand. Springer Berlin Heidelberg, 2006, pp. 431–442.

[47] Ivan Glasser, Nicola Pancotti, and J. Ignacio Cirac. *From probabilistic graphical models to generalized tensor networks for supervised learning*. 2019. arXiv: 1806.05964 [quant-ph].

[48] Sepp Hochreiter. «The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions». In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (Apr. 1998), pp. 107–116. DOI: 10.1142/S0218488598000094.

[49] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. *The Expressive Power of Neural Networks: A View from the Width*. 2017. arXiv: 1709.02540 [cs.LG].

[50] Dor Bank, Noam Koenigstein, and Raja Giryes. *Autoencoders*. 2021. arXiv: 2003.05991 [cs.LG].

[51] Ruslan Salakhutdinov and Geoffrey Hinton. «Deep Boltzmann Machines». In: ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 2009, pp. 448–455. URL: https://proceedings.mlr.press/v5/salakhutdinov09a.html.

[52] Douglas A. Reynolds. «Gaussian Mixture Models». In: *Encyclopedia of Biometrics*. 2009.

[53] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[54] Lore Goetschalckx, Alex Andonian, and Johan Wagemans. «Generative adversarial networks unlock new methods for cognitive science». In: *Trends in Cognitive Sciences* 25.9 (2021), pp. 788–801. ISSN: 1364-6613. DOI: https://doi.org/10.1016/j.tics.2021.06.006.

[55] Christa Zoufal, Aurélien Lucchi, and Stefan Woerner. «Quantum Generative Adversarial Networks for learning and loading random distributions». In: *npj Quantum Information* 5.1 (Nov. 2019). DOI: 10.1038/s41534-019-0223-2.

[56] Ian Goodfellow. *NIPS 2016 Tutorial: Generative Adversarial Networks*. 2017. arXiv: 1701.00160 [cs.LG].

[57] Lillian J. Ratliff, Samuel A. Burden, and S. Shankar Sastry. «Characterization and computation of local Nash equilibria in continuous games». In: *2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2013, pp. 917–924. DOI: 10.1109/Allerton.2013.6736623.

[58] M.L. Menéndez, J.A. Pardo, L. Pardo, and M.C. Pardo. «The Jensen-Shannon divergence». In: *Journal of the Franklin Institute* 334.2 (1997), pp. 307–318. ISSN: 0016-0032. DOI: https://doi.org/10.1016/S0016-0032(96)00063-4.

[59] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[60] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. *On the Convergence of Adam and Beyond*. 2019. arXiv: 1904.09237 [cs.LG].

[61] Ming-Yu Liu and Oncel Tuzel. *Coupled Generative Adversarial Networks*. 2016. arXiv: 1606.07536 [cs.CV].

[62] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. *Progressive Growing of GANs for Improved Quality, Stability, and Variation*. 2018. arXiv: 1710.10196 [cs.NE].

[63] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2019. arXiv: 1812.04948 [cs.NE].

[64] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets*. 2014. arXiv: 1411.1784 [cs.LG].

[65] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016. arXiv: 1511.06434 [cs.LG].

[66] He-Liang Huang and et al. «Experimental Quantum Generative Adversarial Networks for Image Generation». In: *Physical Review Applied* 16.2 (Aug. 2021). DOI: 10.1103/physrevapplied.16.024051.

[67] Seth Lloyd and Christian Weedbrook. «Quantum Generative Adversarial Learning». In: *Physical Review Letters* 121.4 (July 2018). DOI: 10.1103/physrevlett.121.040502.

[68] Pierre-Luc Dallaire-Demers and Nathan Killoran. «Quantum generative adversarial networks». In: *Physical Review A* 98.1 (July 2018). DOI: 10.1103/physreva.98.012324.

[69] Samuel A. Stein, Betis Baheri, Daniel Chen, Ying Mao, Qiang Guan, Ang Li, Bo Fang, and Shuai Xu. «QuGAN: A Quantum State Fidelity based Generative Adversarial Network». In: *2021 IEEE International Conference on Quantum Computing and Engineering (QCE)*. 2021, pp. 71–81. DOI: 10.1109/QCE52317.2021.00023.

[70] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. «Quantum Fingerprinting». In: *Physical Review Letters* 87.16 (Sept. 2001). DOI: 10.1103/physrevlett.87.167902.

[71] Hirotada Kobayashi, Keiji Matsumoto, and Tomoyuki Yamakami. *Quantum Merlin-Arthur Proof Systems: Are Multiple Merlins More Helpful to Arthur?* 2008. arXiv: quant-ph/0306051 [quant-ph].

[72] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran. «Evaluating analytic gradients on quantum hardware». In: *Physical Review A* 99.3 (Mar. 2019). DOI: 10.1103/physreva.99.032331.

[73] Gabriele Agliardi and Enrico Prati. «Optimal Tuning of Quantum Generative Adversarial Networks for Multivariate Distribution Loading». In: *Quantum Reports* 4.1 (2022), pp. 75–105. ISSN: 2624-960X. DOI: 10.3390/quantum4010006.

[74] Gilles Brassard, Peter Høyer, Michele Mosca, and Alain Tapp. *Quantum amplitude amplification and estimation*. 2002. DOI: 10.1090/conm/305/05215.

[75] Gabriele Agliardi, Michele Grossi, Mathieu Pellen, and Enrico Prati. «Quantum integration of elementary particle processes». In: *Physics Letters B* 832 (Sept. 2022). DOI: 10.1016/j.physletb.2022.137228.

[76] Stefan Woerner and Daniel J. Egger. «Quantum risk analysis». In: *npj Quantum Information* 5.1 (2019), p. 15. ISSN: 2056-6387. DOI: 10.1038/s41534-019-0130-6.

[77] Li Deng. «The MNIST database of handwritten digit images for machine learning research». In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142.

[78] Emmanuel Afuecheta, Stephen Chan, and Saralees Nadarajah. «Flexible Models for Stock Returns Based on Student's T Distribution». In: *The Manchester School* 87.3 (2019), pp. 403–427. DOI: https://doi.org/10.1111/manc.12234.

[79] Daniel T. Cassidy, Michael J. Hamp, and Rachid Ouyed. «Pricing European options with a log Student's t-distribution: A Gosset formula». In: *Physica A: Statistical Mechanics and its Applications* 389.24 (2010), pp. 5736–5748. ISSN: 0378-4371. DOI: https://doi.org/10.1016/j.physa.2010.08.037.

[80] S. Huschens and J.-R. Kim. «Measuring Risk in Value-at-Risk Based on Student's t-Distribution». In: *Classification in the Information Age*. Ed. by Wolfgang Gaul and Hermann Locarek-Junge. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 453–459. ISBN: 978-3-642-60187-3.

[81] Qiskit contributors. *Qiskit: An Open-source Framework for Quantum Computing*. 2023. DOI: 10.5281/zenodo.2573505.

[82] Ville Bergholm, Josh Izaac, Maria Schuld, Thomas Fink, and Nathan Killoran. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. Version 0.18.0. 2022. URL: https://pennylane.ai/.

[83] Jonathon Shlens. *A Tutorial on Principal Component Analysis*. 2014. arXiv: 1404.1100 [cs.LG].

[84] François Chollet et al. *Keras*. https://keras.io. 2015.

[85] Frank Nielsen. «Closed-form information-theoretic divergences for statistical mixtures». In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. 2012, pp. 1723–1726. URL: https://ieeexplore.ieee.org/document/6460482.

[86] Sandya Nilmini Kumari. «Characterization of Student's T- Distribution with some Application to Finance». In: *Mathematical Theory and Modeling* 3 (Oct. 2013), pp. 1–9.

[87] A. Meucci. *Risk and Asset Allocation*. Springer Finance. Springer Berlin Heidelberg, 2007. ISBN: 9783540279044. URL: https://books.google.it/books?id=bAS63cyIp0EC.

[88] *Qiskit Documentation, EfficientSU2*. https://qiskit.org/documentation/stubs/qiskit.circuit.library.EfficientSU2.html.

[89] *Qiskit Documentation, Quantum Circuits*. https://qiskit.org/documentation/apidoc/circuit.html.

[90] *Tetris*. https://en.wikipedia.org/wiki/Tetris.

[91] *PyTorch*. https://pytorch.org/.

[92] Frank J. Massey. «The Kolmogorov-Smirnov Test for Goodness of Fit». In: *Journal of the American Statistical Association* 46.253 (1951), pp. 68–78. ISSN: 01621459. URL: http://www.jstor.org/stable/2280095.

[93] S. Kullback and R. A. Leibler. «On Information and Sufficiency». In: *The Annals of Mathematical Statistics* 22.1 (1951), pp. 79–86. DOI: 10.1214/aoms/1177729694.

[94] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[95] S. Ganguly. *Quantum Machine Learning: An Applied Approach: The Theory and Application of Quantum Machine Learning in Science and Industry*. Apress, 2021. ISBN: 9781484270974. URL: https://books.google.it/books?id=8JJizgEACAAJ.

[96] Filip Wojcieszyn. *Introduction to Quantum Computing with Q# and QDK*. Quantum Science and Technology. Springer Cham, 2022. ISBN: 978-3-030-99378-8. URL: https://doi.org/10.1007/978-3-030-99379-5.

[97] A. Jacquier, O. Kondratyev, A. Lipton, and M.L. de Prado. *Quantum Machine Learning and Optimisation in Finance: On the Road to Quantum Advantage*. Packt Publishing, 2022. ISBN: 9781801817875. URL: https://books.google.it/books?id=22SXEAAAQBAJ.

[98] K. Blum. *Density Matrix Theory and Applications*. Springer Series on Atomic, Optical, and Plasma Physics. Springer Berlin Heidelberg, 2012. ISBN: 9783642205613. URL: https://doi.org/10.1007/978-3-642-20561-3.

[99] In: *Quantum Machine Learning*. Ed. by Peter Wittek. Boston: Academic Press, 2014. ISBN: 978-0-12-800953-6. DOI: https://doi.org/10.1016/B978-0-12-800953-6.00015-3.

[100] M. Nakahara and T. Ohmi. *Quantum Computing: From Linear Algebra to Physical Realizations*. Taylor & Francis, 2008. ISBN: 9780750309837. URL: https://books.google.it/books?id=0937mAEACAAJ.

[101] Giovanni Acampora, Ferdinando Di Martino, Alfredo Massa, Roberto Schiattarella, and Autilia Vitiello. «D-NISQ: A reference model for Distributed Noisy Intermediate-Scale Quantum computers». In: *Information Fusion* 89 (2023), pp. 16–28. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2022.08.003.

[102] J.D. Hidary. *Quantum Computing: An Applied Approach*. Springer International Publishing, 2021. ISBN: 9783030832742. URL: https://link.springer.com/book/10.1007/978-3-030-83274-2.

[103] Dawid Kopczyk. *Quantum machine learning for data scientists*. 2018. arXiv: 1804.10068 [quant-ph].

[104] Benjamin Schumacher and Michael Westmoreland. *Quantum Processes Systems, and Information*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511814006.

[105] András Gyenis, Agustin Di Paolo, Jens Koch, Alexandre Blais, Andrew A. Houck, and David I. Schuster. «Moving beyond the Transmon: Noise-Protected Superconducting Quantum Circuits». In: *PRX Quantum* 2 (3 Sept. 2021), p. 030101. DOI: 10.1103/PRXQuantum.2.030101.

[106] Andreas Bengtsson and et al. «Improved Success Probability with Greater Circuit Depth for the Quantum Approximate Optimization Algorithm». In: *Physical Review Applied* 14 (Sept. 2020). DOI: 10.1103/PhysRevApplied.14.034010.

[107] M. Schuld and F. Petruccione. *Supervised Learning with Quantum Computers*. Quantum Science and Technology. Springer International Publishing, 2018. ISBN: 9783319964249. URL: https://link.springer.com/book/10.1007/978-3-319-96424-9.

[108] Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. «Encoding patterns for quantum algorithms». In: *IET Quantum Communication* 2.4 (2021), pp. 141–152. DOI: https://doi.org/10.1049/qtc2.12032.

[109] E.F. Combarro, S. Gonzalez-Castillo, and A. Di Meglio. *A Practical Guide to Quantum Machine Learning and Quantum Optimization: Hands-on Approach to Modern Quantum Algorithms*. Packt Publishing, 2023. ISBN: 9781804618301. URL: https://books.google.com/books?id=1OK0EAAAQBAJ.

[110] *Multi-layer Perceptron*. https://vistalab-technion.github.io/cs236605/lecture_notes/lecture_03/.

[111] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[112] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Vol. 4. 4. Springer, 2006, p. 738. DOI: 10.1117/1.2819119.

[113] *Adam - latest trends in deep learning optimization*. https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c.

[114] Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. *On the Convergence of A Class of Adam-Type Algorithms for Non-Convex Optimization*. 2019. arXiv: 1808.02941 [cs.LG].

[115] Juan Carlos Garcia-Escartin and Pedro Chamorro-Posada. «swap test and Hong-Ou-Mandel effect are equivalent». In: *Physical Review A* 87.5 (May 2013). DOI: 10.1103/physreva.87.052330.

[116] Lov Grover and Terry Rudolph. *Creating superpositions that correspond to efficiently integrable probability distributions*. 2002. arXiv: quant-ph/0208112 [quant-ph].

[117] Rui Li and Saralees Nadarajah. «A review of Student's t distribution and its generalizations». In: *Empirical Economics* 58.3 (2020), pp. 1461–1490. ISSN: 1435-8921. DOI: 10.1007/s00181-018-1570-0.

Appendix A

Gradient Descent Optimization Algorithms

Gradient Descent is a widely used optimization algorithm in neural networks. While it is popular, its various implementations can be difficult to understand as they are often used as black-box optimizers without a clear explanation of their strengths and weaknesses. Here, we provide an overview of different variants of gradient descent, summarize the challenges associated with them, and introduce some of the most commonly used optimization algorithms.
Gradient descent is a technique used to minimize an objective function $C(\theta)$ which relies on the parameters $\theta \in \mathbb{R}^d$ of a given model. This involves updating the parameters by moving them in the direction opposite to the gradient $\nabla_\theta C(\theta)$ of the objective function with respect to the parameters. In other words, we move the parameters in the direction of the greatest rate of decrease of the cost function. The size of the step we take towards the minimum is determined by the learning rate η, which is a hyperparameter. To put it differently, we take small steps in the direction of the negative gradient until we reach a valley on the surface defined by the objective function.

A.1 Gradient Descent Variants

Gradient descent has three distinct variants that differ based on the amount of data used to calculate the gradient of the objective function. The quantity of data used is significant since it impacts the trade-off between the accuracy of the parameter update and the time required to execute the update.

A.1.1 Batch Gradient Descent

Batch Gradient Descent calculates the gradient of the cost function with respect to the parameters θ using the entire training dataset, and updates the parameters according to the equation

$$\theta^{(\tau+1)} = \theta^{(\tau)} - \eta \nabla_\theta C(\theta^{(\tau)}) \tag{A.1}$$

where τ labels the iteration step. Since batch gradient descent needs to process the entire set in each step to evaluate $\nabla_\theta C$, this method can become very slow and is impractical for datasets that are too large. Additionally, batch gradient descent does not allow for online updates of the model, meaning that new examples cannot be incorporated as they come in.
To achieve a sufficiently accurate minimum, it might be necessary to execute a gradient-based algorithm several times, using distinct starting points chosen randomly, and then comparing the performance of each attempt on a separate validation set. If the error surface is convex, batch gradient descent is guaranteed to converge to the global minimum, while for non-convex surfaces, it will likely converge to a local minimum.

A.1.2 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD), which is also referred to as online gradient descent or sequential gradient descent, updates the weight vector of a model using a single training example at a time by applying the rule

$$\theta^{(\tau+1)} = \theta^{(\tau)} - \eta \nabla_\theta C(\theta^{(\tau)}; x_i, y_i) \tag{A.2}$$

Here, $(x_i, y_i)_{i=1,\dots,M}$ denotes the $i-$th data pattern and its corresponding label, respectively. Batch Gradient Descent involves repetitive computations for large datasets since it recomputes gradients for similar examples before each parameter update. In contrast, SGD avoids this redundancy by performing one update at a time, making it faster and suitable for online learning. However, the use of a single pattern at a time in SGD can introduce noise into the cost function, leading to fluctuations in its value, as shown in Figure A.1. In some cases, these fluctuations can be beneficial because they help the algorithm to escape from local minima and explore potentially better areas of the cost function landscape. Nonetheless, if the learning rate is too high, the algorithm may oscillate around the minimum or even diverge.
Research has demonstrated that the learning rate schedule is an important factor in determining the convergence behavior of SGD. Notably, if the learning rate is gradually decreased, SGD can achieve the same convergence behavior as Batch Gradient Descent. This implies that SGD is likely to converge to a local or the

global minimum, depending on whether the optimization problem is convex or non-convex, respectively.

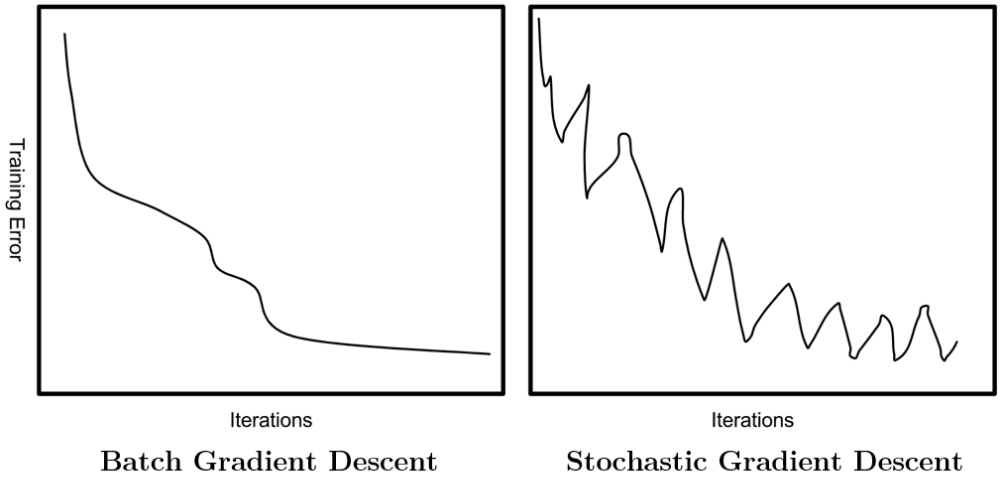


Figure A.1: Sketch of the cost function during the training process using Batch Gradient Descent and Stochastic Gradient Descent optimization methods. In BGD, the model parameters are updated using the gradient of the cost function with respect to the entire batch of training examples at once. This results in smooth updates that typically lead to a monotonically decreasing cost function. In contrast, in SGD, the high variance of the frequent updates can cause the cost function to fluctuate heavily.

A.1.3 Mini-Batch Gradient Descent

Mini-Batch Gradient Descent combines the benefits of Batch Gradient Descent and Stochastic Gradient Descent by performing an update for every mini-batch of k training examples, so that

$$\theta^{(\tau+1)} = \theta^{(\tau)} - \eta \, \nabla_\theta \, C(\theta^{(\tau)}; x_{(i:i+k)}, y_{(i:i+k)}) \tag{A.3}$$

This approach reduces the variance of the parameter updates, resulting in more stable convergence. Additionally, mini-batch gradient descent can take advantage of highly optimized matrix optimizations that are common in state-of-the-art deep learning libraries, making the computation of the gradient with respect to a mini-batch very efficient [111]. The mini-batch size typically ranges between 50 and 500 but can vary depending on the application.

Mini-batch gradient descent is commonly the preferred algorithm for training neural networks, and the term SGD is often used even when mini-batches are employed. In the following, we will omit the parameters $x_{(i:i+k)}$ and $y_{(i:i+k)}$ for the sake of simplicity.

A.2 Challenges

There are several challenges associated with mini-batch gradient descent that need to be addressed for effective optimization:

- Difficulty in choosing a proper learning rate: Selecting an appropriate learning rate is crucial for successful convergence. If the learning rate is too small, convergence will be extremely slow, while a large learning rate may cause the loss function to fluctuate around the minimum or even diverge.

- Inflexible learning rate schedules: Learning rate schedules attempt to modify the learning rate throughout training by reducing it based on a predetermined schedule or when the difference in costs between epochs falls below a specific threshold. However, these schedules and thresholds must be established beforehand, making them incapable of adapting to the unique characteristics of a dataset.

- Uniform learning rate for all parameter updates: Vanilla mini-batch gradient descent applies the same learning rate to all parameter updates. In cases where the data is sparse and features have varying frequencies, it may be more appropriate to perform updates of different magnitudes, with larger updates for infrequent features, rather than updating all features to the same extent.

- Difficulty in escaping saddle points: Minimizing highly non-convex error functions, which are common for neural networks, presents challenges due to the risk of getting trapped in suboptimal local minima or saddle points. Saddle points are points where one dimension slopes up and another slopes down, and they are typically surrounded by a plateau of the same error [111]. Since the gradient is close to zero in all dimensions near a saddle point, it is difficult for SGD to escape, which can slow down or hinder convergence.

To address these challenges, researchers have developed more advanced optimization algorithms, such as adaptive learning rate methods (e.g., AdaGrad, RMSProp, Adam, and AMSGrad), which adapt the learning rate for each parameter update and can better handle sparse data, saddle points, and complex error surfaces. In the following, we will discuss the two algorithms used during numerical experiments, namely Adam and ASMGrad.

A.3 Adam

Adam [59] is a method for efficient stochastic optimization that employs adaptive learning rates, calculating distinct learning rates for various parameters. The name stems from *adaptive moment estimation*, as it utilizes estimates of the first and second moments of the gradient to adjust the learning rate for each weight within the neural network.
Suppose we have a stochastic scalar function $C(\theta)$ that is differentiable with respect to its parameters θ. Our goal is to minimize the expected value of this function with respect to θ

$$\min_{\theta} \mathbb{E}[C(\theta)] \tag{A.4}$$

It is important to note that the gradient of the cost function for a neural network can be seen as a random variable due to its evaluation on a small random batch of data (mini-batch). To simplify the notation, we define $g^{(\tau)} \doteq \nabla_\theta C(\theta^{(\tau)})$ as the gradient of the objective function with respect to the parameters θ at time step τ. The first moment of a random variable corresponds to the mean, while the second moment corresponds to the uncentered variance. To estimate these moments, Adam uses exponentially weighted moving averages that are calculated based on the gradient evaluated on the current mini-batch:

$$m^{(\tau)} = \beta_1 m^{(\tau-1)} + (1-\beta_1) g^{(\tau)} \tag{A.5}$$

$$v^{(\tau)} = \beta_2 v^{(\tau-1)} + (1-\beta_2)(g^{(\tau)})^2 \tag{A.6}$$

for $\tau = 1, \ldots, T$. Here, $m^{(\tau)}$ and $v^{(\tau)}$ represent estimates of the first and second moment of the gradient, $(g^{(\tau)})^2$ denotes the element-wise square $g^{(\tau)} \odot g^{(\tau)}$ (meaning that each entry of the gradient is multiplied by itself), and $\beta_1, \beta_2 \in [0,1)$ are two hyperparameters that control the exponential decay rates of these moving averages [59]. At the initial time step, we fix the initial parameter vector $\theta^{(1)}$ and initialize the first and second moment vectors $m^{(0)}$ and $v^{(0)}$.
As we desire unbiased estimators for the first and second moments, we want to ensure that the following properties hold:

$$\mathbb{E}[m^{(\tau)}] = \mathbb{E}[g^{(\tau)}] \tag{A.7}$$

$$\mathbb{E}[v^{(\tau)}] = \mathbb{E}[(g^{(\tau)})^2] \tag{A.8}$$

However, since $m^{(\tau)}$ and $v^{(\tau)}$ are initially set to vectors of 0's, these estimates tend to be biased towards zero, especially at the beginning of the optimization process, and particularly when the decay rates are small (i.e., β_1 and β_2 are close to 1). To correct these biases, we compute bias-corrected estimates of the first and second moments according to:

$$\hat{m}^{(\tau)} = \frac{m^{(\tau)}}{1-\beta_1^{\tau}} \tag{A.9}$$

$$\hat{v}^{(\tau)} = \frac{v^{(\tau)}}{1 - \beta_2^\tau} \tag{A.10}$$

where $\beta_1^\tau, \beta_2^\tau$ denote β_1 and β_2 to the power τ.
To perform weight update in Adam, we use the moving averages computed in (A.9) and (A.10) to scale the learning rate of each parameter individually. This is achieved by dividing the bias-corrected first moment estimate $\hat{m}^{(\tau)}$ by the square root of the bias-corrected second moment estimate $\hat{v}^{(\tau)}$ plus a correction factor ϵ (to avoid division by 0), and then multiplying by the step size η. The update rule for Adam is hence given by

$$\theta^{(\tau+1)} = \theta^{(\tau)} - \eta \cdot \frac{\hat{m}^{(\tau)}}{\sqrt{\hat{v}^{(\tau)}} + \epsilon} \tag{A.11}$$

The default values suggested for the hyper-parameters are 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ. Empirical evidence demonstrates that Adam performs well in practical applications and is competitive with other adaptive learning algorithms.

A.4 AMSGrad

AMSGrad [60] is a variant of the Adam optimizer that addresses a potential issue with Adam where the learning rate can decay too quickly in certain situations. In Adam, the second moment estimate $v^{(\tau)}$ is used to scale the learning rate, but in AMSGrad, the maximum value of $v^{(\tau)}$ up to time τ is used instead. This modification ensures that the learning rate never decreases as τ increases, preventing a situation where the learning rate becomes too small and slows down convergence. More specifically, in AMSGrad, the estimate of the second moment is computed as:

$$\hat{v}^{(\tau)} = \max(\hat{v}^{(\tau-1)}, v^{(\tau)}) \tag{A.12}$$

To simplify the method, the authors have also eliminated the debiasing step, resulting in the following update rule:

$$\begin{aligned} m^{(\tau)} &= \beta_1 m^{(\tau-1)} + (1 - \beta_1) g^{(\tau)} \\ v^{(\tau)} &= \beta_2 v^{(\tau-1)} + (1 - \beta_2)(g^{(\tau)})^2 \\ \hat{v}^{(\tau)} &= \max(\hat{v}^{(\tau-1)}, v^{(\tau)}) \\ \theta^{(\tau+1)} &= \theta^{(\tau)} - \eta \cdot \frac{m^{(\tau)}}{\sqrt{\hat{v}^{(\tau)}} + \epsilon} \end{aligned} \tag{A.13}$$

In practice, AMSGrad has been shown to improve upon Adam in certain scenarios, such as in training generative adversarial networks or with large-scale language modeling. However, it is important to note that the performance of AMSGrad may depend on the specific problem being solved and the values chosen for the hyperparameters.